60445300

CONTROL DATA CORPORATION

# NOS VERSION 1 REFERENCE MANUAL

Volume 2 of 2

CDC® COMPUTER SYSTEMS:

CYBER 170 MODELS 172, 173, 174, 175

CYBER 70 MODELS 71, 72, 73, 74

6000 SERIES

# LIST OF MACROS



① For system origin jobs or users with system origin privileges.
② Some features of this macro require system origin privileges.
③ This macro is not available in SYSTEXT. The user must call common deck COMCMAC.
④ This macro is not available in SYSTEXT. The user must call common deck COMCCMD.

60445300

CONTROL DATA CORPORATION

# NOS VERSION 1 REFERENCE MANUAL

Volume 2 of 2

CDC® COMPUTER SYSTEMS:

CYBER 170

MODELS 171, 172, 173, 174, 175

CYBER 70

MODELS 71, 72, 73, 74

6000 SERIES

# REVISION RECORD

| REVISION | DESCRIPTION |
|---|---|
| A (6-17-75) | Manual released. |
| B (3-8-76) | Revised to update the manual to NOS 1.1 and to make typographical and technical corrections. New features documented in this manual include: 844-41 Disk Storage Subsystem support; additional security control; GETMC, MACHID, and SETSSM macros; and support of multimainframe. Section 3 has been revised to more accurately reflect magnetic tape processing and the FET descriptions. Appendix E now contains central memory tables that were previously available only in the NOS Systems Programmer's Instant Manual. This edition obsoletes all previous editions. |
| C (12-3-76) | Revised to update the manual to NOS 1.2 at PSR level 439 and to make typographical and technical corrections. New features documented in this manual include: GETASL and SETASL macros for account block SRU limit; GETJSL and SETJSL macros for job step SRU limit; revision of GETTL and SETTL macros to apply to job steps; revision of GETFLC and SETRFL macros for control of initial and maximum field lengths; SETMFL macro for setting the maximum field length limit; removal of QDL support (old section 8); and new section 8 describing file routing and the ROUTE macro. Appendix E has been revised to eliminate information sensitive to system security and that which does not apply to the applications programmer. This edition obsoletes all previous editions. |
| D (7-15-77) | Revised to update manual to NOS 1.2 at PSR level 452 and to make typographical and technical corrections. Support of CDC CYBER 170 Series, Model 171 is also included. |
| E (11-21-77) | Revised to update manual to NOS 1.2 at PSR level 460 and to make typographical and technical corrections. New features include CIO read parity error processing, the FILINFO macro for determining information about a file, the GETLOF and SETLOF list of files macros, and systems text PSSTEXT containing macros defined on common decks COMCMAC and COMCCMD. |
| Publication No. 60445300 | scanned 2/2004 by gmt |

**REVISION LETTERS I, O, Q AND X ARE NOT USED**



Address comments concerning this manual to:

**Control Data Corporation
Publications and Graphics Division
4201 North Lexington Avenue
St. Paul, Minnesota 55112**

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 | - |
| Inside Front Cover | - |
| Title Page | - |
| ii | E |
| iii | E |
| iv | E |
| v | E |
| vi | C |
| vii | D |
| viii | D |
| ix | E |
| x | E |
| xi | E |
| xii | E |
| 2-1-1 | D |
| 2-2-1 | D |
| 2-2-2 | D |
| 2-2-3 | D |
| 2-2-4 | D |
| 2-2-5 | E |
| 2-2-6 | E |
| 2-2-6.1/ 2-2-6.2 | D |
| 2-2-7 | E |
| 2-2-8 | D |
| 2-2-9 | E |
| 2-2-10 | C |
| 2-2-11 | B |
| 2-2-12 | B |
| 2-2-13 | B |
| 2-2-14 | B |
| 2-3-1 | D |
| 2-3-2 | D |
| 2-3-3 | D |
| 2-3-4 | E |
| 2-3-5 | E |
| 2-3-6 | E |
| 2-3-6.1/ 2-3-6.2 | E |
| 2-3-7 | E |
| 2-3-8 | E |
| 2-3-9 | E |
| 2-3-10 | B |
| 2-3-11 | D |
| 2-3-12 | C |
| 2-3-13 | D |
| 2-3-14 | D |
| 2-3-15 | D |
| 2-3-16 | D |
| 2-3-17 | D |
| 2-3-18 | D |
| 2-3-19 | D |
| 2-3-20 | D |
| 2-3-21 | E |
| 2-3-22 | E |
| 2-3-23 | D |
| 2-3-24 | B |
| 2-3-25 | E |

| PAGE | REV |
|---|---|
| 2-3-26 | D |
| 2-3-27 | C |
| 2-3-28 | E |
| 2-3-29 | E |
| 2-3-30 | E |
| 2-3-31 | B |
| 2-3-32 | B |
| 2-3-33 | D |
| 2-3-34 | C |
| 2-3-35 | C |
| 2-3-36 | E |
| 2-3-37 | B |
| 2-3-38 | C |
| 2-3-39 | D |
| 2-3-40 | E |
| 2-3-41 | B |
| 2-3-42 | D |
| 2-3-43 | D |
| 2-3-44 | E |
| 2-3-45 | E |
| 2-3-46 | D |
| 2-3-47 | D |
| 2-3-48 | B |
| 2-3-49 | B |
| 2-3-50 | B |
| 2-3-51 | C |
| 2-3-52 | C |
| 2-3-53 | C |
| 2-3-54 | C |
| 2-3-55 | B |
| 2-3-56 | B |
| 2-3-57 | E |
| 2-3-58 | C |
| 2-3-59 | C |
| 2-4-1 | D |
| 2-4-2 | C |
| 2-4-3 | A |
| 2-4-4 | C |
| 2-4-5 | C |
| 2-4-6 | C |
| 2-4-7 | D |
| 2-4-8 | D |
| 2-4-8.1/ 2-4-8.2 | D |
| 2-4-9 | A |
| 2-4-10 | A |
| 2-4-11 | C |
| 2-4-12 | C |
| 2-4-13 | C |
| 2-4-14 | E |
| 2-4-15 | E |
| 2-4-16 | E |
| 2-4-17 | E |
| 2-4-18 | C |
| 2-4-19 | C |
| 2-4-20 | D |
| 2-4-21 | E |
| 2-4-22 | E |
| 2-4-23 | E |

| PAGE | REV |
|---|---|
| 2-5-1 | C |
| 2-5-2 | C |
| 2-5-3 | C |
| 2-5-4 | D |
| 2-5-5 | D |
| 2-5-6 | E |
| 2-5-6.1/ 2-5-6.2 | D |
| 2-5-7 | C |
| 2-5-8 | D |
| 2-5-9 | C |
| 2-5-10 | D |
| 2-5-11 | C |
| 2-5-12 | D |
| 2-5-13 | D |
| 2-5-14 | E |
| 2-5-15 | C |
| 2-5-16 | D |
| 2-5-17 | D |
| 2-5-18 | D |
| 2-5-19 | D |
| 2-5-20 | D |
| 2-5-21 | C |
| 2-5-22 | D |
| 2-5-23 | C |
| 2-5-24 | D |
| 2-6-1 | D |
| 2-6-2 | C |
| 2-6-3 | C |
| 2-6-4 | C |
| 2-6-5 | C |
| 2-6-6 | D |
| 2-6-7 | C |
| 2-6-8 | C |
| 2-6-9 | C |
| 2-6-10 | C |
| 2-6-11 | D |
| 2-6-12 | C |
| 2-6-13 | C |
| 2-6-14 | C |
| 2-6-15 | C |
| 2-6-16 | C |
| 2-6-17 | C |
| 2-6-18 | C |
| 2-6-19 | C |
| 2-6-20 | C |
| 2-6-21 | C |
| 2-6-22 | C |
| 2-6-23 | C |
| 2-6-24 | C |
| 2-6-25 | C |
| 2-6-26 | D |
| 2-6-27 | E |
| 2-6-28 | E |
| 2-6-29 | E |
| 2-7-1 | C |
| 2-7-2 | C |
| 2-7-3 | C |
| 2-7-4 | C |
| 2-7-5 | C |

| PAGE | REV |
|---|---|
| 2-8-1 | C |
| 2-8-2 | D |
| 2-8-3 | E |
| 2-8-4 | C |
| 2-8-5 | C |
| 2-9-1 | C |
| 2-9-2 | C |
| 2-9-3 | B |
| 2-9-4 | A |
| 2-10-1 | E |
| 2-10-2 | C |
| 2-10-3 | A |
| 2-10-4 | A |
| 2-10-5 | A |
| 2-11-1 | B |
| 2-11-2 | E |
| 2-11-3 | B |
| 2-11-4 | A |
| 2-11-5 | D |
| 2-11-6 | D |
| 2-11-7 | C |
| 2-11-8 | C |
| 2-11-9 | E |
| 2-11-10 | A |
| 2-11-11 | A |
| 2-11-12 | A |
| 2-11-13 | A |
| 2-11-14 | D |
| 2-11-15 | A |
| 2-11-16 | C |
| 2-11-17 | D |
| 2-11-18 | D |
| 2-11-19 | C |
| 2-11-20 | D |
| 2-11-21 | C |
| 2-11-22 | C |
| 2-12-1 | A |
| 2-12-2 | E |
| 2-12-3 | E |
| 2-12-4 | D |
| 2-12-5 | D |
| 2-12-6 | C |
| 2-12-7 | C |
| 2-12-8 | B |
| 2-12-9 | B |
| 2-12-10 | B |
| 2-12-11 | C |
| 2-12-12 | B |
| 2-A-1 | E |
| 2-A-2 | E |
| 2-A-3 | E |
| 2-A-4 | C |
| 2-A-5 | C |
| 2-B-1 | A |
| 2-B-2 | B |
| 2-B-3 | B |
| 2-B-4 | B |
| 2-B-5 | B |
| 2-B-6 | B |

| PAGE | REV |
|---|---|
| 2-B-7 | B |
| 2-B-8 | B |
| 2-B-9 | B |
| 2-B-10 | B |
| 2-C-1 | A |
| 2-C-2 | A |
| 2-C-3 | A |
| 2-C-4 | A |
| 2-C-5 | A |
| 2-C-6 | A |
| 2-C-7 | A |
| 2-C-8 | A |
| 2-C-9 | E |
| 2-C-10 | B |
| 2-C-11 | A |
| 2-C-12 | A |
| 2-C-13 | A |
| 2-C-14 | A |
| 2-C-15 | A |
| 2-C-16 | A |
| 2-C-17 | A |
| 2-D-1 | A |
| 2-D-2 | A |
| 2-D-3 | A |
| 2-D-4 | A |
| 2-D-5 | A |
| 2-D-6 | A |
| 2-D-7 | A |
| 2-D-8 | A |
| 2-D-9 | A |
| 2-D-10 | A |
| 2-D-11 | A |
| 2-D-12 | A |
| 2-D-13 | A |
| 2-D-14 | A |
| 2-D-15 | A |
| 2-D-16 | A |
| 2-D-17 | A |
| 2-D-18 | A |
| 2-E-1 | D |
| 2-E-2 | D |
| 2-E-3 | C |
| 2-E-4 | C |
| 2-E-5 | C |
| 2-F-1 | E |
| 2-F-2 | E |
| 2-G-1 | B |
| 2-G-2 | B |
| 2-G-3 | B |
| 2-G-4 | B |
| 2-G-5 | D |
| 2-G-6 | B |
| 2-G-7 | C |
| 2-G-8 | C |
| 2-G-9 | C |
| 2-H-1 | A |
| 2-H-2 | C |
| 2-H-3 | C |
| 2-H-4 | C |

| PAGE | REV | PAGE | REV | PAGE | REV | PAGE | REV | PAGE | REV |
|---|---|---|---|---|---|---|---|---|---|
| Index-1 | E | | | | | | | | |
| Index-2 | E | | | | | | | | |
| Index-3 | E | | | | | | | | |
| Index-4 | E | | | | | | | | |
| Index-5 | E | | | | | | | | |
| Index-6 | E | | | | | | | | |
| Index-7 | E | | | | | | | | |
| Index-8 | E | | | | | | | | |
| Index-9 | E | | | | | | | | |
| Index-10 | E | | | | | | | | |
| Index-11 | E | | | | | | | | |
| Cmt Sheet | E | | | | | | | | |
| Back Cover | - | | | | | | | | |

# PREFACE

The Network Operating System (NOS) was developed by Control Data Corporation to provide network capabilities for time-sharing and transaction processing, in addition to local and remote batch processing on CONTROL DATA® CYBER 170 Series, Models 171, 172, 173, 174, and 175 Computer Systems; CYBER 70 Series, Models 71, 72, 73, and 74 Computer Systems; and 6000 Series Computer Systems.

This manual describes the external features of NOS 1.2 for the batch user. Information in this manual should be useful to those who use the programs and utilities supplied with the system and those who wish to write their own. The manual is contained in two volumes to separate information pertaining primarily to the applications programmer from that of interest to the applications COMPASS programmer.

Volume 1 (publication no. 60435400) contains information for the applications programmer. This includes general information about files, job flow and execution, control statement processing, and an extensive discussion of control statements.

Volume 2 (publication no. 60445300) contains information for those who write system or assembly language programs for use with NOS. It is primarily intended for the applications COMPASS programmer; however, several portions contain information for users of higher level languages.

Throughout this manual, cross-references to the NOS Reference Manual, Volume 1, are of the form, "refer to section (or appendix) n, volume 1." If volume 1 is not stipulated, the reference is to this manual.

This manual does not contain a description of NOS system operation, detailed descriptions of the software product set available under NOS, or descriptions of the time-sharing commands.

The user is assumed to be familiar with CDC computer systems and with operating systems in general.

Conventions for central memory word formats are as follows:

- Cross-hatching indicates a field is not used by or is not applicable to a function processor. However, CDC reserves the right to assign these fields to system use in the future.
- Fields reserved for system use are so labeled.
- Fields labeled with mnemonics indicate a specific parameter must be inserted (generally described after the word format).
- Fields with numeric identifiers indicate the actual value that is used or returned for a particular function.

For further information concerning CDC CYBER 170, CDC CYBER 70, and 6000 Series Computer Systems, the NOS time-sharing system, and the products supported by NOS, consult the following manuals.

| Control Data Publication | Publication No. |
|---|---|
| CDC CYBER 170 Computer Systems Reference Manual | 60420000 |
| CDC CYBER 70/Model 71 Computer System Reference Manual | 60453300 |
| CDC CYBER 70/Model 72 Computer System Reference Manual | 60347000 |
| CDC CYBER 70/Model 73 Computer System Reference Manual | 60347200 |
| CDC CYBER 70/Model 74 Computer System Reference Manual | 60347400 |
| CDC 6400/6500/6600 Computer Systems Reference Manual | 60100000 |
| NOS Systems Programmer's Instant | 60449200 |
| NOS Applications Programmer's Instant | 60436000 |
| NOS Installation Handbook | 60435700 |
| NOS Time-Sharing User's Reference Manual | 60435500 |
| NOS Operator's Guide | 60435600 |
| NOS Terminal User's Instant | 60435800 |
| TRANEX Version 1 Reference Manual | 60407900 |
| TRANEX Version 1/TAF Version 1 User's Guide | 60436500 |
| TAF Version 1 Reference Manual | 60453000 |
| TAF Version 1 Data Manager Reference Manual | 60453100 |
| Export/Import Reference Manual | 60436200 |
| Text Editor Reference Manual | 60436100 |
| NOS Modify Reference Manual | 60450100 |
| NOS Modify Instant | 60450200 |
| Update Reference Manual | 60449900 |
| COMPASS Reference Manual | 60492600 |
| FORTRAN Extended Reference Manual | 60497800 |
| CDC CYBER Record Manager Reference Manual | 60495700 |
| CDC CYBER Loader Reference Manual | 60429800 |
| NAM Reference Manual | 60499500 |
| RBF Reference Manual | 60499600 |

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 undefined parameters.

# CONTENTS

VOLUME 1



## APPENDIXES

VOLUME 2

## APPENDIXES



## INDEX

## FIGURES

# INTRODUCTION 1

Network Operating System (NOS) provides many utilities in the form of function processors, macros, and system routines that enable the user to perform complex operations with a minimum of coding. These routines have been thoroughly tested and optimized, and are designed to interface with the system.

## FUNCTION PROCESSORS

Several NOS system routines process user requests. Sections 3 through 11 contain descriptions of each function processor, including:

- Identification of requests (function numbers)
- System macros (or common decks† containing macros) available to issue the requests
- Common decks required to issue the requests
- Information returned from the processor for the requests

A listing of error messages for function processors, macros, and common decks is contained in volume 1.

## MACROS AND COMMON DECKS

Most system macros are available to COMPASS programmers and are used to issue requests to the function processors. Common decks are normally used with macros or function requests or as subroutines.

This manual provides the user with the information necessary to communicate with the system to perform the tasks that cannot be performed in the CPU. For information concerning batch usage and system processing in general, refer to volume 1. For information on the COMPASS assembly language, refer to the COMPASS Reference Manual. For information on calling the COMPASS assembler, refer to appendix H.

† Refer to the Modify Reference Manual for a complete description of common decks.

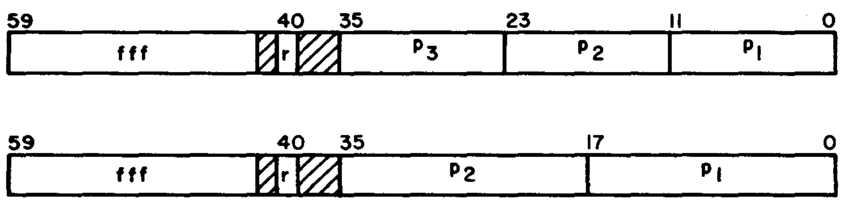
# PROGRAM/SYSTEM COMMUNICATION 2

## SYSTEM REQUESTS

All communication with the system is performed by entering a system request in location 1 (RA+1) of the field length (refer to figure 2-E-1). The system then initiates execution of that portion of the system required to satisfy the user's request. There are two types of system requests.

- Those that contain all information necessary in RA+1 (for example, RCL)
- Those that require additional areas for parameters or results from the system (for example, CIO addr)

The format of the 60-bit request is one of the following (depending on whether two or three parameters are to be passed).




| | |
|---|---|
| fff | System request name |
| r | Auto recall bit ($p_1$ must be specified) |
| $p_1, p_2, p_3$ | Parameters passed to the portion of the system that processes fff ($p_2$ in second format may contain two parameters) |

If the auto recall bit is set in a system request, CPU assignment to the job is released when the request is sensed. The CPU is not reassigned to the job until the request is complete.

### REQUEST PROCESSORS

Most system requests require some area within the user's job to contain information for the requests being processed. The request processors consist of two groups.

The following request processors are associated with a file environment table (FET) (refer to section 3).

Local file manager (LFM)

Combined input/output (CIO)

Permanent file manager (PFM)

System file manager (SFM)

Queue file manager (QFM)

The following request processors perform a job action request or provide system information.

Control point manager (CPM)

Exit processing request (ABT)

Time accumulated (TIM)

Termination request (END)

Memory requests (RFL, MEM)

System message processor (MSG)

CM dump (DMP, DMD)

Recall CPU (RCL)

Overlay request (LDR, LDV)

Fast dynamic loading (LDD, LDQ)

File routing (DSP)

Translate control statement (TCS)

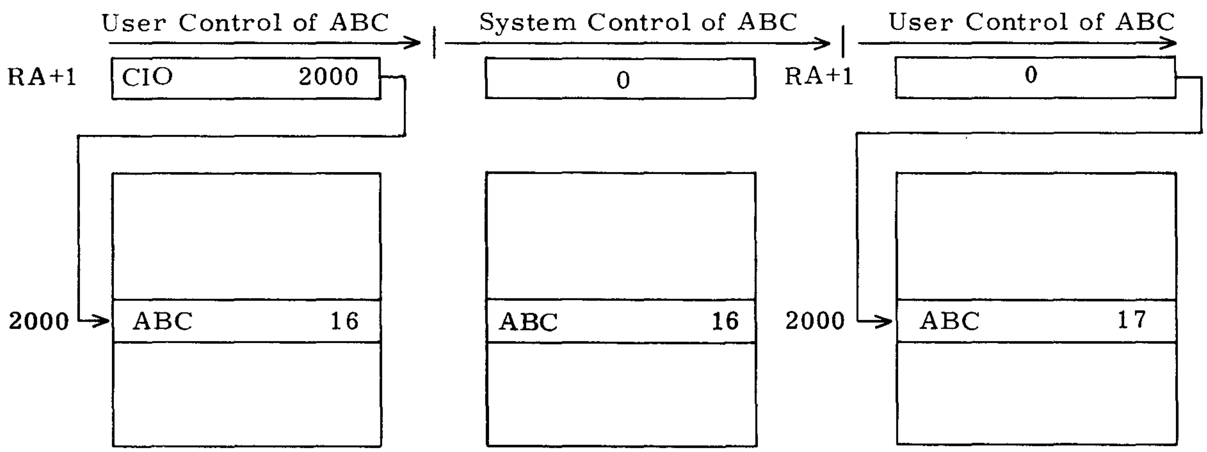
## SYSTEM REQUEST PROCESSING

Whether a system request communicates with a FET or another parameter block, the first word of this area is usually the status word. Both the system and the user use the lower portion of this word to communicate the status of the request. When bit 0 is cleared (equals 0), the system is in control of the request; when it is set (equals 1), the user is in control of the request.

For example, to write on file ABC, the program must perform the following steps.

1. Check the status of ABC.
2. If ABC is busy (bit 0 cleared), the program must wait until bit 0 is set. This is done by issuing a system request to recall (RCL).
3. When ABC is idle (bit 0 set), the program must clear bit 0 and place the request in RA+1.
4. If other processing can be performed, the program proceeds.
5. If further processing depends upon ABC being completed, the program must check the status word for completion (bit 0 set by the system).

To perform this write operation on file ABC, the user issues a system request to CIO. The following diagram illustrates the user/system control when performing this operation.



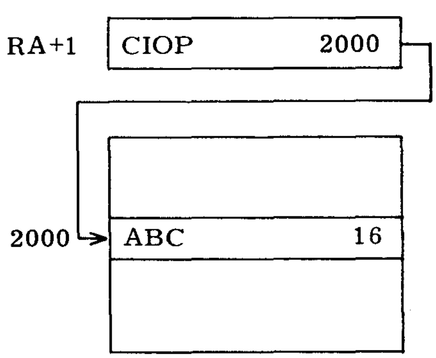

The user requests the system to take control of ABC by clearing bit 0 of the status word and entering CIO 2000 in RA+1.

The system clears RA+ 1 upon beginning processing of the request.

The system sets bit 0 of the status word upon completing the request.

In many situations, the program cannot proceed until the system request is complete (as in steps 2 and 5 when ABC is busy). When this occurs, the user can prevent execution until the status word is not busy (bit 0 set). This simplifies the programmer's job, because he does not have to check the status. It also reduces the amount of CPU time used by the job.

In the previous example, if the following request is issued:




P is the display code representation of the auto recall bit ($20_8$); that is, bit 40 of RA+1 is set.

the system would not allow the job to continue execution until bit 0 of word 2000 was set and the PPU completed its operation.

The steps in this procedure are:

1. Check the status of ABC.
2. If ABC is busy, wait until bit 0 is set and return to step 1. This can be done by issuing an RCLP function on word 2000.
3. Clear bit 0 of word 2000 and place the CIOP request in RA+1.

Processing can proceed with the assurance that the previous operation on ABC is completed (bit 0 of word 2000 does not have to be checked).

The user should be aware that many system requests require that the auto recall bit be set. This is noted in the description of the requests, when necessary.

## ISSUING RA+1 REQUESTS

When a system request is placed in RA+1, the system may process that request at any time. The central exchange jump (CEJ) instruction (XJ), if available, provides much faster response to system requests.

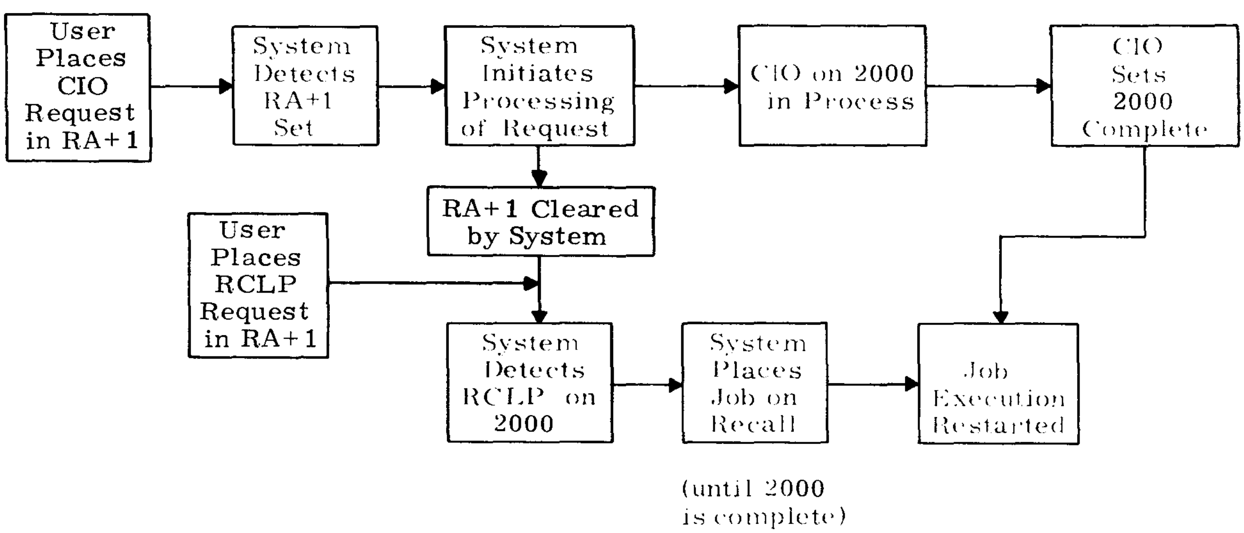
The system processes program requests by scanning location RA+1 of all user programs in the system. This is done on a periodic basis. If the system detects a request in RA+1 that can be processed, RA+1 is cleared and processing begins.

When the user program issues a request, it must check RA+1 to determine if the request is accepted by the system before it issues another request.

The following example illustrates the steps taken by the user and the system to process a system request.

1. The user issues a CIO request on word 2000 to RA+1 (CIO 2000).
2. The user proceeds with his processing.
3. The user now wishes to wait until the request is complete before continuing. He does this by issuing an RCLP request on word 2000. However, he must first check RA+1 to ensure that the request to CIO is accepted before he issues the RCLP request.

The system performs the following steps in processing this user request.




When the XJ instruction is available, the user places the system request in RA+1 and executes an XJ instruction. This causes the CPU to be exchanged to the system program CPUMTR from the user's program. The system can act upon the request immediately.

When the system initiates processing of the request, CPU control returns to the user, unless the request is made with auto recall.

The user can determine whether the XJ instruction is available by checking if bit 59 of word 66 (XJPR) of his field length is set (refer to SYSCOM macro and figure 2-E-1).

If the request in the previous example is performed in this manner, the following steps apply.

1. The user places the CIO request on word 2000 in RA+1 and executes an XJ instruction. The central processor is reassigned to the job when the request is accepted.

2. The user proceeds with his processing.

3. The user now wishes to wait until this request is completed before continuing. This is done by placing the request RCLP 2000 in RA+1 and issuing another XJ instruction. It is not necessary to determine if the previous request is accepted.

Many macros and common decks are provided to assist the programmer in performing this interaction with the system.

## NOS SYSTEMS TEXTS

The following systems texts are available to the NOS user.

- SYSTEXT
- PPTEXT
- NOSTEXT
- PSSTEXT

SYSTEXT contains system communication macros that are used by the CPU COMPASS programmer. PPTEXT contains symbol definitions used by all SYSTEM PP routines for intercommunication. NOSTEXT contains all system communication macros and symbol definitions that are found in SYSTEXT and PPTEXT.

PSSTEXT contains product set support macros that are defined on system OPL common decks COMCMAC and COMCCMD (refer to appendix A).

By selecting the correct systems text (for the applications COMPASS programmer, this usually will be SYSTEXT and/or optionally PSSTEXT), the user can reduce the amount of system resources needed for assemblies.

To obtain listings of the systems texts, enter one or more of the following control statements after accessing the system OPL.

```
MODIFY(Q, CL, CS=0, Z)/*EDIT, SYSTEXT
MODIFY(Q, CL, CS=0, Z)/*EDIT, PPTEXT
MODIFY(Q, CL, CS=0, Z)/*EDIT, NOSTEXT
```

To obtain listings of the common decks on PSSTEXT, enter the following statement after accessing the system OPL.

```
MODIFY(Q, CL, Z)/*EDIT, CALLCPU
```

## MACRO USAGE

Macros are available for issuing most requests to the function processors. If macros are not available, the user can define them.

A macro is a predefined sequence of code that can be used in the user's programs. † If the user wishes to use predefined macros, he may do so by specifying the location of these definitions to COMPASS with the S or G parameter. For example:

```
COMPASS(I,B,S=XYZTEXT)
```

This call causes all macro definitions in XYZTEXT to be available for assembly of the program. If no S parameter is specified, COMPASS uses the system default system text SYSTEXT. In the descriptions in this manual, unless otherwise noted, all macros are defined in SYSTEXT and NOSTEXT.

Some macros are defined in common decks; therefore, the common deck must be called into the text of the program (refer to Common Deck Usage). If the decks are named COMCMAC or COMCCMD, the user has the option of specifying the alternate systems text PSSTEXT. For example:

```
COMPASS(I,B, S,S=PSSTEXT)
```

This makes available all macro definitions in PSSTEXT (that is, those defined in common decks COMCMAC and COMCCMD) for the assembly of the program.

In addition to the macros available in SYSTEXT, an integer divide operation definition is provided for the user's convenience.

Integer Division

IXi Xj/Xk

Divide Xj by Xk and place the result in Xi

Destroys Xj, Xk, B7

When a macro parameter refers to an address, the parameter may be a register name, a relocatable address, an external symbol, or an absolute address. The user should consult the expansions of the system macros to determine the optimum use of registers when using macros. The user is responsible for ensuring that a register used as a parameter contains only the parameter (for example, if an 18-bit address is specified by an X register, the user must ensure that the upper 42 bits of the register are zero).

NOS system macros and common decks assume that the contents of the following registers are preserved.

A0, X0, A5, and X5

Also, upon exit, the contents of registers B1 and X2 are as follows:

| | |
|---|---|
| B1 | 1 |
| X2 | FET address (refer to section 3) if a macro specifies the FET address as a parameter |

The contents of the B1 register is assumed to be 1 upon entry only if the SYSCOM B1 macro is called or the B1=1 COMPASS pseudo-instruction is defined.

† The COMPASS Reference Manual provides instructions for defining macros.

## SYSCOM

Many system common decks called from macros assume the contents of the B1 register is equal to 1. Other common decks assume B1 is equal to 1 only if the macro SYSCOM B1 or COMPASS pseudo instruction B1=1 is defined. If this macro is used, it should be set as an initial step in the program. If the B1 parameter is not included in the SYSCOM call or if there is no SYSCOM call, these common decks then generate additional code to set B1 equal to 1. If SYSCOM B1 is used, it is the user's responsibility to set the B1 register equal to 1.

Macro Format:

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | SYSCOM | B1 |

B1 If present, COMPASS pseudo instruction B1=1 is defined

The specification of SYSCOM (with or without specifying B1) also makes available the system communication symbols (refer to figure 2-E-1). These are:

| Symbol | Value | Description |
|---|---|---|
| ARGR | 2 | Address of the first argument |
| SPPR | $27_8$ | Special program parameter area (locations $27_8$ through $47_8$) |
| PGNR | $64_8$ | Program name (bits 59 through 18) |
| ACTR | $64_8$ | Argument count (bits 17 through 0) |
| CMUR | $65_8$ | Compare move unit (CMU) available flag (bit 59) |
| LWPR | $65_8$ | LWA+1 of the assigned program space (bits 17 through 0) |
| FWPR | $66_8$ | FWA of the assigned program space (bits 17 through 0) |
| JOPR | $66_8$ | Job origin type (bits 35 through 24) |
| XJPR | $66_8$ | Central exchange available (bit 59) |
| CSMR | $67_8$ | System character set mode flag (bit 59); set if 64 character set mode |
| LDRR | $67_8$ | LDR completion (bit 29) |
| CCDR | $70_8$ | Control statement image (eight locations) |
| LINP | $60_{10}$ | Lines/printer page |

## COMMON DECK USAGE

A system common deck is a COMPASS subroutine or group of macro or symbol definitions that have been tested, optimized, and placed on a program library where they can be accessed by the user.

All of the common decks supplied with the system are available to any user who has access to the system OPL. The OPL is a collection of records of text which is accessed randomly by the Modify utility program (refer to Modify control statement, section 14, volume 1 or the Modify Reference Manual). Normally the system OPL is available as a direct access permanent file and the user must issue an attach command before Modify runs. The user should contact installation personnel to determine how to access the OPL.

Several classes of system common decks exist; however, the user need only be concerned with the CPU common decks. Documentation of these common decks can be obtained with the following control statements, after the system OPL is accessed.

```
MODIFY(Z)/*EDIT, CALLCPU
DOCMENT.
```

If a complete listing of the routines is desired, the following control statement can be entered.

```
MODIFY(Q, CL, Z)/*EDIT, CALLCPU
```

Appendix A contains a list of the CPU common decks of general interest and describes their functions. Appendix C contains further information on common decks and their usage.

CPU common deck names have the following format.

COMCxxx

| | |
|---|---|
| COM | Indicates a common deck |
| C | Specifies CPU common deck |
| xxx | Entry point name (sometimes defined as xxx=) |

The typical COMPASS programmer who is writing relocatable programs is generally unaware of the common deck interface, and need only be concerned if a common deck of unique nature (for example, the common deck that converts display code to octal representations) is required, or if the specified common deck is not on the system library (SYSLIB). Refer to appendix A for a list of common decks available in relocatable form on SYSLIB.

Two common decks (COMCMAC and COMCCMD) are also available on systems text PSSTEXT. Thus, the user has the option of either calling them from the system OPL or specifying the alternate systems text PSSTEXT as described under Macro Usage.

Most system macros require that the common deck related to a function processor be available in the program; however, they need not be specifically called by the user when assembling relocatable programs, since all macros specify entries to common decks as external symbols. When the relocatable subroutines are loaded, the routines required at execution time (such as LFM= and CIO=) are loaded from SYSLIB.

For example, a subroutine uses the following macro.

```
  .
  .
  READ    ABC, R
  .
```

This macro requires routines CIO= and SYS=; however, the READ macro generates:

```
    RJ      =XCIO=
```

and CIO= generates:

```
    RJ      =XSYS=
```

Since these are flagged as external references (=X), the loader satisfies them from SYSLIB if they are not locally satisfied.

If a program is not relocatable or if the desired common deck is not on SYSLIB, then a system common deck must be accessed from the system OPL.

To use a common deck, the programmer must insert the Modify directive *CALL in the text of his program, use the COMPASS pseudo-op XTEXT, or reference it in a relocatable program so that it is loaded and linked from SYSLIB.

**NOTE**

> When using a common deck, the user should be aware of the format of the common deck and any registers it uses.

The following example illustrates the procedure for using the *CALL directive.

```
JOB1.
USER(USERNUM,PASSWRD,FAMILY)
COPYBR(INPUT,XFILE)           COPY PROGRAM TO XFILE
ATTACH(OPL/UN=ABC123,PN=PACKC) ATTACH OPL
RFL,45000.
MODIFY(Q,CL)                  CALL MODIFY PROGRAM
LGO.                          LOAD AND EXECUTE COMPASS BINARY
-EOR-
PROG                          DECK NAME TO BE EDITED
          IDENT   PROG,TME
          ABS                 BODY OF ABSOLUTE COMPASS PROGRAM
          ENTRY   START         THAT OUTPUTS THE CURRENT TIME
          SYSCOM  B1
          ORG     102B
 TME      BSS     1
 OBUF     BSS     101B
 OUTPUT   FILEC   OBUF,101B
 START    SB1     1
          CLOCK   TME         CLOCK MACRO REQUIRES
          SA5     TME           COMMON DECK COMCSYS
          BX6     X5
          WRITEO  OUTPUT      WRITEO MACRO CALLS COMMON
          WRITER  OUTPUT        DECK COMCWTO, WRITER CALLS
          ENDRUN                COMCCIO, ENDRUN CALLS COMCSYS
*CALL     COMCCIO             MODIFY DIRECTIVES TO INSERT
*CALL     COMCWTO               SPECIFIED COMMON DECKS
*CALL     COMCSYS
          END     START
-EOR-                         MODIFY INPUT DIRECTIVES--
*REWIND XFILE                 REPOSITION XFILE TO BOI
*CREATE XFILE                 CREATE PROGRAM LIBRARY
*EDIT PROG                    MODIFY DECK PROG
-EOI-
```

After the OPL is attached, the Modify utility edits the COMPASS deck by inserting the common decks in place of the respective *CALL statements. The Q parameter on the Modify statement causes Modify to call COMPASS to assemble the resultant COMPASS program. The CL parameter specifies COMPASS list output. The COMPASS listing from this program does not contain a listing of the called common decks. Instead, the following is provided.

```
     .
     .
     ENDRUN
     CTEXT     COMCCIO  - I/O FUNCTION PROCESSOR.
     CTEXT     COMCWTO - WRITE ONE WORD.
     CTEXT     COMCSYS  - PROCESS SYSTEM REQUEST.
     END       START
```

To have the specified common decks listed in the program, the user must use the COMPASS LIST pseudo-op.

The net effect of the *CALL statement is that the text for the specified common deck is inserted at that position in the program text. In the previous example, the assembler would receive the following text.

```
                       .
                       .
                       WRITER OUTPUT
                       ENDRUN
            CIO1       RECALL X2              WAIT COMPLETION
            ERP$       IF      -DEF,ERP$
                       .
                       .
                       .
            CIO=       PS                     ENTRY/EXIT
COMCCIO                .
                       .
                       .
                       EQ      CIO=           RETURN
                       .
                       .
                       .
            WTO1       SA6     X1             STORE WORD
                       SX2     A1-2
                       .
                       .
                       .
            WTO=       PS                     ENTRY/EXIT
COMCWTO                .
                       .
                       .
                       RJ      =XCIO=
                       .
                       .
                       .
            SYSA       BSS     0
                       LOC     *+2
                       .
                       .
COMCSYS                .
            SYS=       PS                     ENTRY/EXIT
                       .
                       .
                       .
                       END     START
```

The procedure for using the COMPASS pseudo-op XTEXT to obtain common decks is detailed in the following example.

```
JOB2.
USER(USERNUM,PASSWRD,FAMILY)
ATTACH(OPL/UN=ABC123,PN=PACKC)
COMPASS.
LGO.
-EOR-
PROG
          IDENT   PROG,TME
          ABS
          ENTRY   START
          SYSCOM  B1
          ORG     102B
 TME      BSS     1
 OBUF     BSS     101B
 OUTPUT   FILEC   OBUF,101B
 START    SB1     1
          CLOCK   TME
          SA5     TME
          BX6     X5
          WRITEO  OUTPUT
          WRITER  OUTPUT
          ENDRUN
 OPL      XTEXT   COMCSYS
 OPL      XTEXT   COMCWTO
 OPL      XTEXT   COMCCIO
          END     START

-EOI-
```

After attaching the OPL, a call is made to the COMPASS assembler. The XTEXT pseudo instruction provides a means of obtaining source statements from a file other than that being used for input. COMPASS transfers the text from the external source and assembles it before taking the next statement from the program.

A COMPASS listing from a program using the XTEXT instruction also does not list the inserted common decks. These can be obtained with the COMPASS LIST instruction or by using list options on the COMPASS control statement.

The conflict of tags is the one problem that may arise when using common decks. All tags and routine names within common decks conform to the NOS programming specifications (refer to appendix C), and as such, the routine names relate to their particular functions. If a user's routine name conflicts with the name of the common deck being used, he should rename the routine. If a conflict still exists, he should qualify the entire common deck (refer to the COMPASS Reference Manual).

For example, if the programmer wishes to use the space-fill-name common deck, COMCSFN, he can use the method shown in the following sample program.

```
DUPLICT
          IDENT   DUPLICT
          ENTRY   DUPLICT

 QUAL$    EQU     1            SUPPRESS COMMON DECK QUALIFICATION

*         THE *SPACE FILL NAME* COMMON DECK MUST BE QUALIFIED
*         BECAUSE THE PROGRAM HAS A SUBROUTINE *SFN* AND THE
*         PROGRAMMER DOES NOT WISH TO RENAME THE SUBROUTINE.


 DUPLICT  SB1     1            ENTRY
          SA1     DUPA         SET SYMBOL NAME IN MESSAGE
          MXØ     -18          CLEAR VALUE PORTION OF WORD
          SX5     X1+          SAVE VALUE
          BX1     XØ*X1                          Transfers control to
          RJ      /SFILL/SFN   SPACE FILL WORD   COMCSFN, qualified
          SA6     DUPC                           by SFILL
          SX1     X5+          CONVERT VALUE
          RJ      CDD
          SB2     3            SHIFT VALUE 3 CHARACTERS LEFT
          RJ      SFN
          SA6     DUPD         ISSUE MESSAGE     Transfers control to
          MESSAGE DUPB,,R                        the user's shift
          ENDRUN                                 register routine, SFN

 DUPA     CON     ØLNAME+123456
 DUPB     CON     1ØH   SYMBOL
 DUPC     CON     Ø            SYMBOL NAME
          CON     1ØHHAS VALUE
 DUPD     CON     Ø            CONVERTED SYMBOL VALUE
          CON     Ø            MESSAGE TERMINATOR
          SPACE   4
**        SFN - SHIFT REGISTER BY *N* CHARACTERS.
*
*         ENTRY   (X6) = REGISTER TO SHIFT.
*                 (B2) = NUMBER OF CHARACTERS TO SHIFT.
*
*         EXIT    (X6) = SHIFTED LEFT.
*
*         USES    B - 2.
*                 X - 6, 7.
          SPACE   4
 SFN      PS                   ENTRY/EXIT
          SX7     B2           SET NUMBER OF BITS TO SHIFT
          SB2     B2+B2
          LX7     2
          SB2     B2+X7
          LX6     X6,B2        SHIFT REGISTER
          JP      SFN          RETURN


*         COMMON DECKS.

*CALL     COMCSYS
*CALL     COMCCDD
          QUAL    SFILL        QUALIFY *SFN*
*CALL     COMCSFN
          SPACE   4
          END     DUPLICT
```

The QUAL$ tag can also be used to suppress the qualification of common decks called more than once in overlays. In the following example, the QUAL pseudo instruction is used in each of the two primary overlays to allow an unqualified reference to CDD.

```
 OVER
          IDENT   MAIN,MAIN,MAIN          MAIN (0,0) OVERLAY
          ABS
          TITLE   MAIN (0,0) OVERLAY.
          ORG     110B

 QUAL$    EQU     1            SUPPRESS COMMON DECK QUALIFICATION

*         LOAD TWO OVERLAYS - ONE AT A TIME - EACH WILL USE
*         ITS OWN COPY OF *CDD*, BUT WILL USE *SYS=* AND
*         *MSG=* THAT RESIDE IN THE (0,0) OVERLAY.

 MAIN     MESSAGE MAIA,,R      * MAIN RUNNING.*
          SB1     1
          OVERLAY MAIB,0100B LOAD *OVL1* OVERLAY
          SB2     X1           EXECUTE *OVL1*
          JP      B2
 MAI1     OVERLAY MAIB,0200B LOAD *OVL2* OVERLAY
          SB2     X1           EXECUTE *OVL2*
          JP      B2
 MAI2     MESSAGE MAIC,,R      * MAIN COMPLETE.*
          ENDRUN END

 MAIA     DIS     ,* MAIN RUNNING.*
 MAIB     CON     0LLGO        OVERLAY FILE NAME
 MAIC     DIS     ,* MAIN COMPLETE.*

*CALL     COMCSYS
*CALL     COMCOVL

 OVER     BSS     0            OVERLAY AREA BASE
          TTL     PRIMARY (1,0) OVERLAY.
          EJECT
          QUAL    OVL1         DEFINE QUALIFICATION FOR THIS OVERLAY
          IDENT   OVL1,OV1,OV1,1,0        PRIMARY (1,0) OVERLAY
          ORG     OVER         START AT OVERLAY AREA
```

```
*          READ THE REAL TIME CLOCK AND ISSUE A DAYFILE MESSAGE
*          INDICATING WHEN OVERLAY WAS CALLED.

 OV1       RTIME   OV1A         GET CURRENT TIME
           SA1     OV1A         CONVERT MILLISECONDS TO DISPLAY CODE
           MX0     -36
           BX1     -X0*X1
           RJ      CDD
           SA6     OV1C
           MESSAGE OV1B,,R      * OVL1 CALLED AT    NNNNNN*
           JP      MAI1         RETURN

 OV1A      CON     0            REAL TIME CLOCK
 OV1B      DATA    20H OVL1 CALLED AT
 OV1C      CON     0            CONVERTED MILLISECONDS
           CON     0            MESSAGE TERMINATOR

*CALL      COMCCDD

           TTL     PRIMARY (2,0) OVERLAY.
           EJECT
           QUAL    OVL2         DEFINE QUALIFICATION FOR THIS OVERLAY
           IDENT   OVL2,OV2,OV2,2,0      PRIMARY (2,0) OVERLAY
           ORG     OVER         START AT OVERLAY AREA

*          READ THE REAL TIME CLOCK AND ISSUE A DAYFILE MESSAGE
*          INDICATING WHEN OVERLAY WAS CALLED.

 OV2       RTIME   OV2A         GET CURRENT TIME
           SA1     OV2A         CONVERT MILLISECONDS TO DISPLAY CODE
           MX0     -36
           BX1     -X0*X1
           RJ      CDD
           SA6     OV2C
           MESSAGE OV2B,,R      * OVL2 CALLED AT    NNNNNN*
           JP      MAI2         RETURN

 OV2A      CON     0            REAL TIME CLOCK
 OV2B      DATA    20H OVL2 CALLED AT
 OV2C      CON     0            CONVERTED MILLISECONDS
           CON     0            MESSAGE TERMINATOR

*CALL      COMCCDD

           END
```

## SECURITY CONSIDERATIONS

Unless a job is of system origin type, or the user is validated for system origin privileges and DEBUG mode has been set at the system display console (refer to NOS Operator's Guide), the following requirements or restrictions are placed on user jobs in the interests of system security.

- If a job step does not contain an SSJ= entry point (refer to appendix F) and secure system memory (SSM) status is set, then the job step cannot request system functions through RA+ 1 calls that are processed by programs containing a DMP= special entry point (refer to appendix F).
- If a job step has SSM set, it cannot be followed by a control statement that calls a DMP= processor.

Violation of these restrictions results in the job step being aborted and the following dayfile message being issued.

SECURE MEMORY, DUMP DISABLED.

Programs containing an SSJ= entry point are permitted to make RA+ 1 calls to DMP= processors with SSM status set.

Secure system memory status is set either automatically by the system whenever a program containing SSJ= or SSM= special entry points (refer to appendix F) is loaded or by a program using a SETSSM macro (refer to section 6).

RA+ 1 calls processed by DMP= programs include the following.

CKP - Checkpoint request

DMP - Dump field length request

DMD - Dump field length request with display code

REQ - Request equipment assignment

LFM - For the ASSIGN, LABEL, and REQUEST functions (refer to section 4)

PFM - For any access to removable auxiliary devices (refer to section 5)

Refer to section 3, volume 1, for a list of control statements that cannot follow job steps with SSM status set.

# FILE CREATION AND INPUT/OUTPUT 3

This section describes the process of performing input/output from a COMPASS program and the creation of files to accomplish these and other tasks.

The file environment table (FET) portion describes circular buffer concepts and FET creation macros. The discussion of FETs is important, not only for input/output and file creation, but also for system routines that perform file processing in general.

The combined input/output (CIO) part includes those macros needed for file creation, read and write functions, file positioning, and data transfer.

The user can perform I/O directly through FETs using CIO, or he can use CDC CYBER Record Manager facilities that are available to COMPASS users through COMPASS macro calls. A brief description of Record Manager features is provided later in this section.

## FILE ENVIRONMENT TABLE (FET)

The FET is the standard communication area or parameter block for the system file processors. The COMPASS programmer must define the FET, whereas the higher level languages (COBOL and FORTRAN, for example) automatically establish and use this area.

Depending on the processor being used, certain areas of the FET must be defined and used in communicating with that processor. The minimum length of a FET is five words.

## CIRCULAR BUFFERS

All input/output is performed by passing data between a user's program circular buffers (central memory) and a peripheral device (mass storage or magnetic tapes, for example).

A circular buffer is a temporary central memory storage area that contains data during input/output operations. It is called a circular buffer because routines that process input/output treat the first word of the buffer area as contiguous to the last word of the buffer area. The buffer parameters (FIRST, IN, OUT, and LIMIT) in the FET describe the circular buffer (refer to figure 2-3-1).

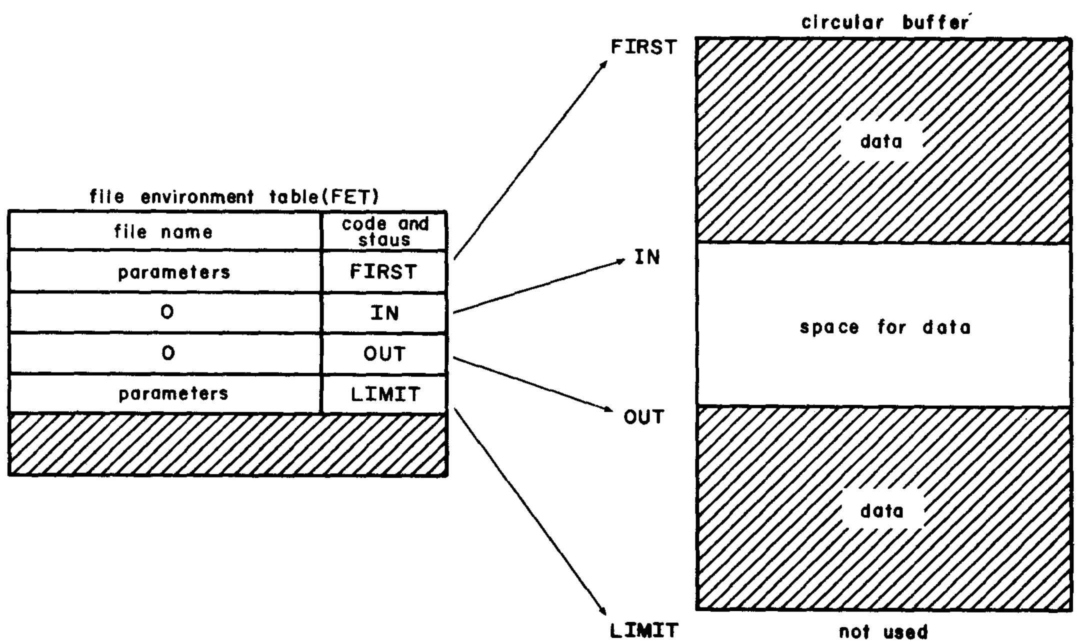



Figure 2-3-1. Circular Buffer

### FIRST Address

FIRST is the first word address of the buffer area. Routines that perform input/output never change the value of FIRST.

### LIMIT Address

LIMIT is the last word address plus 1 of the buffer areas. Data is not stored in LIMIT since LIMIT is not part of the circular buffer. When LIMIT is reached, the next available address for storage is FIRST. Routines that perform input/output never change the value of LIMIT.

### OUT Address

OUT is the next location to read to remove data from the circular buffer. Either the system or the user's program changes OUT depending on whether the operation is a write or a read (refer to figures 2-3-2 and 2-3-3).

### IN Address

IN is the next location to write data into the circular buffer. Either the system or the user program changes IN depending on whether the operation is a read or a write operation (refer to figures 2-3-2 and 2-3-3). When IN=OUT, the buffer is empty. When IN=OUT-1, or IN=LIMIT-1 and OUT=FIRST, the buffer is full.

That is, one location is left empty in a full buffer to distinguish an empty buffer from a full buffer. A buffer is normally initialized with IN=OUT=FIRST, and IN and OUT circle the buffer as data is inserted and extracted.

Refer to the FET creation macros in this section for a description of minimum and recommended buffer sizes.

Figure 2-3-2 is an example of a read operation.

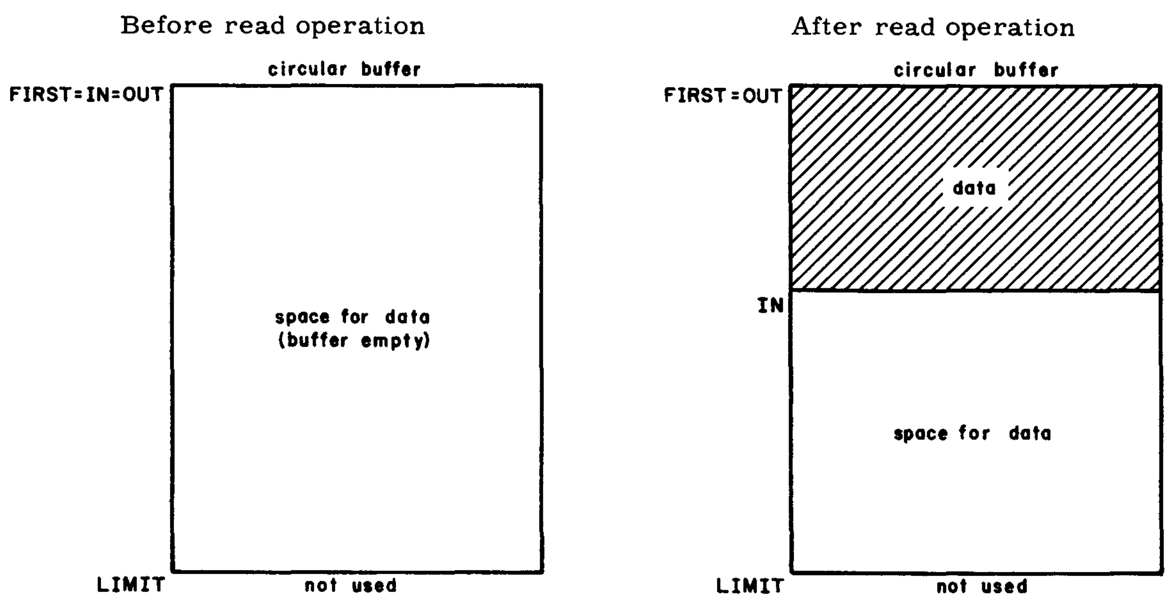



Figure 2-3-2. Read Operation

Figure 2-3-3 is an example of a write operation.

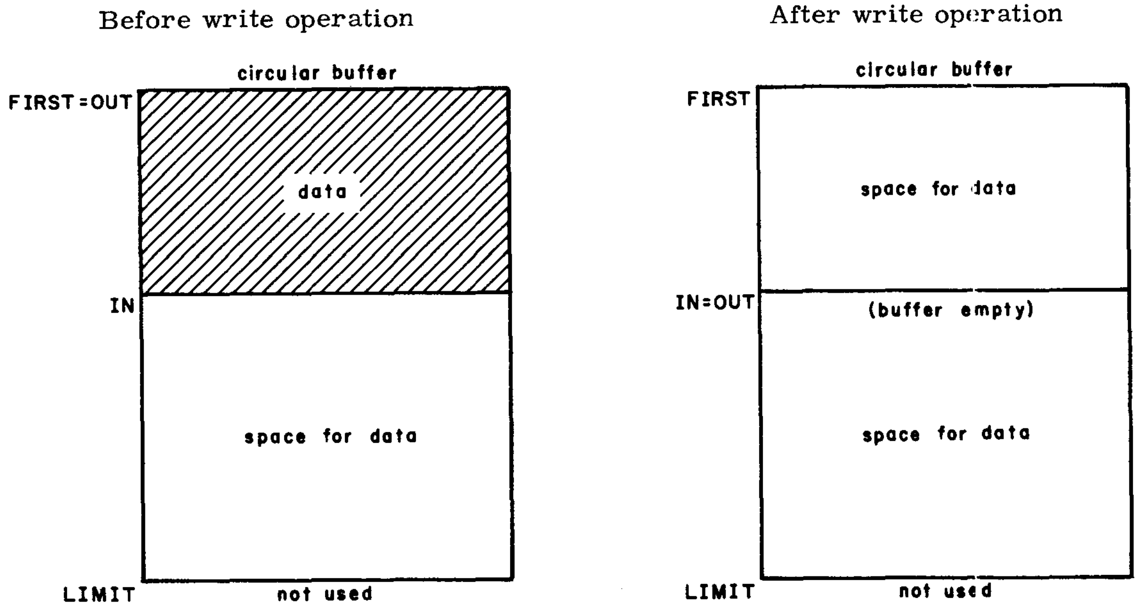



Figure 2-3-3. Write Operation

## FET DESCRIPTION

There are two basic file environment table (FET) formats. Figure 2-3-4 illustrates the standard FET for mass storage files; figure 2-3-5 illustrates the standard FET for magnetic tape files. The figures are followed by a description of the FET fields. When a field is used by only one of the file processors, it is noted in the description.

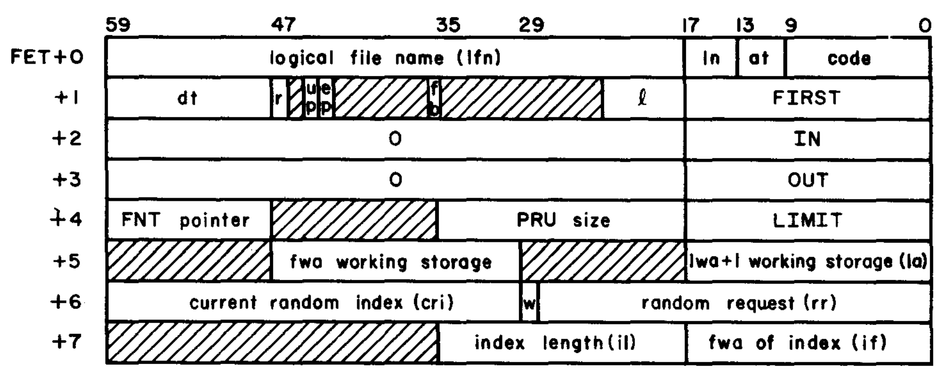



Figure 2-3-4. Standard FET for Mass Storage File

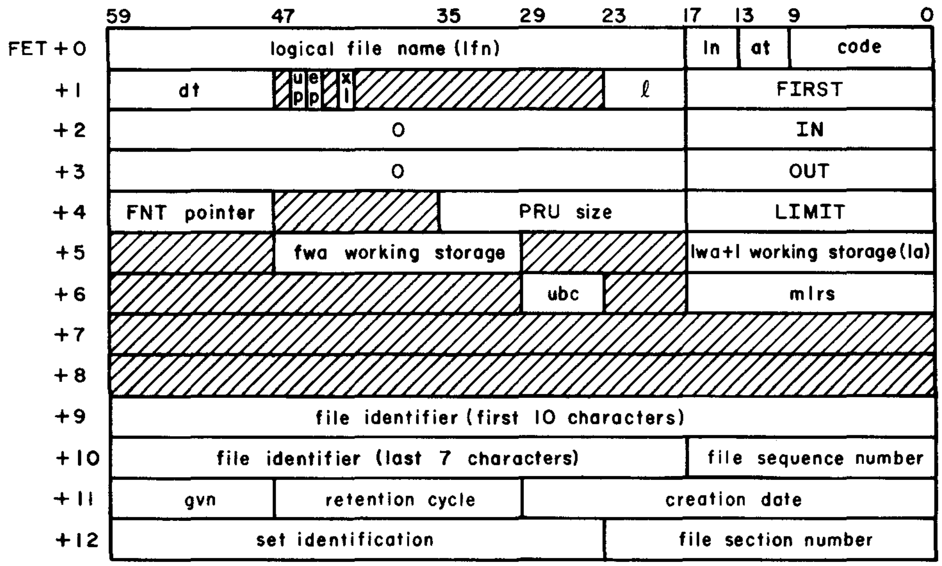



Figure 2-3-5. Standard FET for Labeled Magnetic Tape File (CIO)

Refer to the LABEL and OPEN macros for a description of the FET fields used in processing ANSI labels.

| Parameter | Word | Position | Description |
|---|---|---|---|
| Logical file name (lfn) | 0 | 59-18 | The lfn field contains 1 to 7 alphanumeric display code characters, left-justified; unused characters are zero-filled. The lfn is the common reference point for all system communication concerning the file. |
| Level number (ln) | 0 | 17-14 | This is the level number for an EOR/EOF operation on the file. NOS uses this field for CIO operations and for distinguishing interactive input from noninteractive input (refer to CIO and the discussion on writing interactive programs, section 12). |
| Abnormal termination Codes (at) | 0 | 13-10 | Status information returned by the function processor when an abnornal situation or error occurs. This field is usually set by the processor when the error processing bit (ep) is set in word 1 of the FET. For some processors, at is returned in bits 17 through 10. This field is set to $11_8$ by CIO if the ep bit is not set and an error condition is present. |
| Code | 0 | 9-0 | Request/return code. The user (or macro) sets this code for the request desired. The function processors alter it only if the request is not completed. For example, the user requests a read (CIO code 0010) but encounters an EOR. CIO returns a status code of 0021. These codes are detailed in the function processor descriptions. |

| Parameter | Word | Position | Description |
|---|---|---|---|
| | | | The following are subfields for the code parameter.<br><br>Bit 9: end-of-information (eoi). This bit is set when an end-of-information is encountered on a read request.<br><br>Bits 4-3: set to binary 10 if an end-of-record is encountered on a read request; set to binary 11 if end-of-file encountered.<br><br>Bits 1: file mode (fm). File mode for input/output operations (S, L, or SI tape formats only):<br>0 Coded<br>1 Binary<br><br>Bit 0: Interlock (ilk). FET interlock bit. Used to indicate system/user access to data associated with the file. The user sets this bit to 0 (busy or not complete) and the processor sets it to 1 when completed. |
| Device type (dt) | 1 | 59-48 | The 12-bit display code of the type of device on which the file is or will be residing. If bit 59 is set, it indicates a nonallocatable device (refer to appendix E). If an S, L, or SI file is opened, this field contains NOS/BE compatible return information (refer to the OPEN macro). |
| Random access (r) | 1 | 47 | This bit is set if random processing is to be performed on the file. If this is set, the FET must be at least seven words in length. |
| User processing (up) | 1 | 45 | The user sets this bit if he desires to perform his own end-of-reel or end-of-device processing. When CIO encounters an end-of-reel/end-of-device, it returns an abnormal termination code of 1 (bits 13 through 10 of word 0). For further information about end-of-reel processing, refer to the CIO CLOSER macro. The up bit is checked only if the FET is at least six words in length. |
| Error processing (ep) | 1 | 44 | This bit is used to indicate to the function processor that the calling program processes errors that occur, such as parity errors or errors in requests to the file managers. The function processor returns the error code in the at field of word 0 (bits 13 through 10). Function processor descriptions should be consulted for the error codes returned. If this bit is not set, the function processor either aborts the job or requests operator intervention. The ep bit is checked only if the FET is at least six words in length. |

| Parameter | Word | Position | Description |
|---|---|---|---|
| | | | If an unrecovered parity or block too large error occurs during a magnetic tape read operation (with the ep bit set) or after a read parity error occurs on a mass storage read operation (with ep bit set), the following steps occur.<br>1. The data in the bad block is stored in the user's circular buffer.<br>2. The value of the IN pointer prior to the read is stored in the next word in the circular buffer (pointer to the beginning of the bad data block).<br>3. The parity error code is set in FET+0.<br>4. The IN pointer is updated in the FET. This IN pointer value does not include the additional word (pointer to the beginning of the bad data block) stored in the buffer.<br>5. The FET completion bit is set.<br><br>The pointer to the bad data is returned on all reads processed by CIO for a mass storage file. If no data is transferred (correct PRU not read), the pointer points to itself, and no update of IN occurs.<br><br>If tape error processing is inhibited (refer to the LABEL macro, section 4), the preceding steps are not performed regardless of whether or not the ep bit is set. |
| Extended label processing (xl) | 1 | 41 | Specifies standard (xl = 0) or extended (xl = 1) tape label processing. |
| Flush bit (fb) | 1 | 36 | Specifies that the file's circular buffer is to be flushed upon abnormal termination (mass storage files only).<br><br>Files that are pointed to by the list of files (refer to the SETLOF macro, section 6) and meet the following criteria are flushed with an end of record write.<br>● No buffer parameter errors; that is:<br>Complete FET within user's field length<br>LIMIT .LT. FL<br>OUT .LT. LIMIT<br>OUT .GT. FIRST<br>IN .LT. LIMIT<br>IN .GT. FIRST<br>● Write bit is set in the FET or the FET is unused (bits 9 through 3 are 0). |

| Parameter | Word | Position | Description |
|---|---|---|---|
| | | | • No CIO error code exists in FET.<br>• Data is in buffer. |
| FET length ($\ell$) | 1 | 23-18 | Specifies the additional length of the FET over normal size (five words). For example, if $\ell$ = 3, FET length = 8 and FET + 7 is the last usable word. Function processors require varying lengths for particular parameters. However, it is recommended that the FET lengths be at least six words for most efficient system processing. |
| FIRST | 1 | 17-0 | First word address of input/output buffer. |
| IN | 2 | 17-0 | The next available location for entering data into the buffer. Note that the upper 42 bits should never be used since the function processors read and write the entire word. |
| OUT | 3 | 17-0 | The next available location for removing data from the buffer. Note that the upper 42 bits should never be used since the function processors read and write the entire word. |
| FNT pointer | 4 | 59-48 | Current position of the file name table (FNT) pointer. This is used by the system to reduce overhead when processing a file. It is set only if the FET length is greater than 5. |
| Physical record unit size | 4 | 35-18 | Number of CM words in PRU of the device to which the file is assigned. The PRU size for mass storage is always 64 CM words. The PRU size for magnetic tape varies according to the data format selected.<br><br>This is set only if the file is opened using the CIO OPEN macro and if the FET length is greater than 5. Refer to the CIO OPEN macro for information on magnetic tape PRU size. |
| LIMIT | 4 | 17-0 | Last word address plus 1 of the buffer. Data is never placed in or removed from LIMIT. |
| First word address of working storage | 5 | 47-30 | First word address of working storage. Working storage is used by several of the compilers to control input/output in specific formats (blocking/unblocking). This parameter is not used by the system or the NOS common decks which refer to working storage areas. Working storage areas for use by macros (READS, READC, etc.) require the user to define his own working storage area and specify it on each macro request. Pointers to working storage can be placed here for reference. |

| Parameter | Word | Position | Description |
|---|---|---|---|
| Working storage last word address +1 | 5 | 17-0 | Last word address plus 1 of working storage. |
| List address (la) | 5 | 17-0 | List address. This points to the table of the relative sector addresses for CIO READLS and RPHRLS macros. |
| Current random index (cri) | 6 | 59-30 | The current random index for the mass storage file being randomly accessed. The system returns the current position of the file after a random input/output request. This is in the form of a relative sector address (rsa) from the beginning of the file. For any nonrandom read or write operation, the system updates this field by adding the number of sectors transferred to the existing contents of the field. For any random access or positioning operation, the value is recalculated. cri is ignored if r (word 1) is not set. |
| Random rewrite request (w) | 6 | 29 | This bit is set to indicate a write-in-place operation. If not set, the write takes place at the current position with rr being the address for the return of rsa, where the write began. This is ignored if r (word 1) is not set. |
| Unused bit count (ubc) | 6 | 29-24 | Specifies the unused bit count for S and L format tapes (refer to Data Formats, section 10, volume 1). |
| Random request (rr) | 6 | 28-0 | Relative sector address (rsa) for a random input/output request. An exception is if w = 0 and it is a write request, then it is the address for the return of the starting rsa of the write (previous EOI). If the error processing bit (FET+1, bit 44) is set and an error occurs, the system returns detailed error status information in FET+6, bits 11 through 0. For further information, refer to the description of CIO. rr is ignored if r (word 1) is not set. |
| Maximum logical record size (mlrs) | 6 | 17-0 | Specifies the maximum physical record size for S and L format tapes. For S format, if mlrs = 0, the value of the maximum PRU is assumed to be 512 ($1000_8$) words. For L format, if mlrs = 0, the assumed maximum PRU is LIMIT-FIRST-1 for standard reads or writes. LIMIT-FIRST-2 for READN or WRITEN, and LIMIT-FIRST-3 for READCW and WRITECW. Refer to Data Formats, section 10, volume 1. |
| Index length (il) | 7 | 35-18 | Random index length. This must be set by the user when requesting CIO OPEN to load the random index of a file or CIO CLOSE to dump the random index of a file. If r (word 1) is not set, il is ignored. |

| Parameter | Word | Position | Description |
|---|---|---|---|
| Index first word address (if) | 7 | 17-0 | First word address of the index buffer. This is the area where CIO OPEN stores the index when opening a file or the area from which CIO CLOSE takes the index when closing a file. If r (word 1) is not set, if is ignored. |
| File identifier | 9 | 59-0 | File identifier (first 10 display code characters, left-justified with binary zero or blank fill). † |
| File identifier | 10 | 59-18 | File identifier (last seven display code characters, left-justified with binary zero or blank fill). † |
| File sequence number | 10 | 17-0 | 1- to 3-digit numeric display code file sequence number (right-justified with display code zero fill). † |
| Generation version number (gvn) | 11 | 59-48 | 1- to 2-digit numeric display code generation version number (right-justified with display code zero fill). † |
| Retention cycle | 11 | 47-30 | 1- to 3-digit numeric display code retention cycle (right-justified with display code zero fill). † |
| Creation date | 11 | 29-0 | Creation date (2-digit numeric display code value for the year followed by a 3-digit numeric display code value for the day within the year). † |
| Set identification | 12 | 59-24 | 1- to 6-character set identification (left-justified with binary zero or blank fill). † |
| File section number | 12 | 23-0 | 1- to 4-digit numeric display code file section number (right-justified with display code zero fill). † |

## FET CREATION MACROS

The following four macros are defined to initialize FETs. They can be used to create sequential coded files, sequential binary files, random coded files, and random binary files.

### FILEB

The FILEB macro creates a FET for a binary sequential file.

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| lfn | FILEB | fwa, length, $p_1$, $p_2$, . . . , $p_n$ |

† Refer to appendix G, volume 1, HRD1 label.

## FILEC

The FILEC macro creates a FET for a coded sequential file.

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| lfn | FILEC | fwa, length, $p_1, p_2, \ldots, p_n$ |

## RFILEB

The RFILEB macro creates a FET for a binary random file.

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| lfn | RFILEB | fwa, length, $p_1, p_2, \ldots, p_n$ |

## RFILEC

The RFILEC macro creates a FET for a coded random file.

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| lfn | RFILEC | fwa, length, $p_1, p_2, \ldots, p_n$ |

The following parameters apply to each FET creation macro.

lfn — File name.

fwa — First word address of CIO buffer.

length — Length of the CIO buffer. Because the buffer is full when IN = OUT- 1 or IN = LIMIT- 1 and OUT = FIRST, the buffer length should not be an exact multiple of PRU size. The following are the minimum and recommended buffer sizes for mass storage,. tape, and terminal input/output.

| Device | Minimum (octal) | Recommended (octal) |
|---|---|---|
| Mass storage | 101 (without control words†) | 1001 to 2001 (without control words) |
| | 103 (with control words) | 1021 to 2041 (with control words) |
| Tape (I, SI, and X formats) | 1001 (without control words) | 3n + 1 to 4n + 1, where: n = 1000 (without control words) |
| | 1003 (with control words) | n = 1002 (with control words) |
| Time-sharing terminal | | 101 for input.<br>301 for output |

† Refer to the CIO READCW (200) request for a description of control words.

Refer to Data Formats, section 10, volume 1, for a description of PRU sizes for S, L, E, B, and F tape formats.

pi The following parameters can be used to set fields in the FET. They can be specified in any order.

| | |
|---|---|
| DTY = dt | Sets the device type to dt. User must specify display code equivalent of dt. For example, set device type to TT by specifying DTY = 2RTT or DTY = 2424B. Refer to appendix E for legal equipment codes. |
| EPR | Sets the error processing bit (FET+1, bit 44). |
| FET = $\ell$ | Sets the length of the FET to $\ell$ . |
| IND = addr,$\ell$ | Sets the index first word address to addr and the index length to $\ell$. |
| LBL | Sets the FET length to $13_{10}$ for tape label processing. |
| OWN = iaddr, jaddr | Sets the OWNCODE EOI address to iaddr and the error exit address to jaddr. If jaddr is included, the error processing bit is also set. |
| UPR | Sets the user processing bit (FET+1, bit 45). |
| XL | Sets the extended label processing bit (FET+1, bit 41). and sets the FET length to $13_{10}$. |
| WSA = addr,$\ell$ | Sets the first word address of working storage to addr and the length of working storage to $\ell$. |

The following parameters are used for setting PFM communication words (refer to section 5) in the FET.

| | |
|---|---|
| PFN=name | Sets the permanent file name |
| USN=usernum | Sets the optional user number usernum |
| PWD=passwrd | Sets the permanent file password |
| UCW=usercon | Sets the user control bits (bit 59 must be set to indicate that the word contains user control information) |
| PKN=packname | Sets the packname for access to permanent files residing on auxiliary devices |

The following example illustrates the use of FET creation macros to create an FET for sequential input/output operations.

```
BUFL    EQU     2001B
TAPE1   FILEB   BUF,BUFL
BUF     BSS     BUFL
        .
        .
```

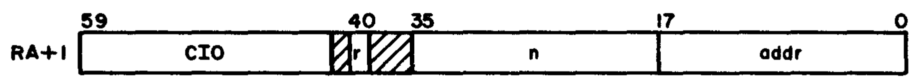
The following example creates an FET for retrieving a file from permanent file storage, loading the index block, and performing random input/output operations.

```
BUFL     EQU      2001B
INDXL    EQU      100B
TAPE1    RFILEB   BUF,BUFL,(FET=10D),(PFN=STOCKS),(IND=INDX,INDXL)
BUF      BSS      BUFL
INDX     BSS      INDXL
           .
           .
```

## CIO — COMBINED INPUT/OUTPUT

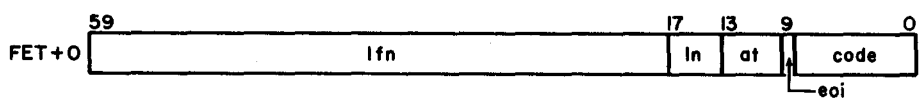
The CIO read/write requests are used to transfer data between a file and a CIO circular buffer. The read requests transfer input files from a system storage medium to a CIO circular buffer. The write requests transfer output from a CIO circular buffer to a system storage medium. Also included in this group of requests are those which open and close files, those which update records in an existing mass storage file, and those used to control positioning of the file.

The format of the call to CIO is:




| | |
|---|---|
| r | Auto recall, if desired |
| n | Count for skip operations |
| addr | Address of the FET |

Word 0 of the FET contains the following information.




| | |
|---|---|
| lfn | Logical file name |
| ln | Level number ($0 \leq ln \leq 17_8$) for an EOR/EOF operation on the file: |

| | |
|---|---|
| 0 | EOR operation |
| $1\text{-}16_8$ | Same as level 0 |
| $17_8$ | EOF operation |

at — Abnormal termination code returned by CIO:

- 01 — End-of-reel (magnetic tape) or end-of-device (mass storage)
- 02 — Parity error
- $11_8$ — Other error (applies only to mass storage files; refer to FET+6, dec field)

eoi — End of information bit

code — Request/return code:

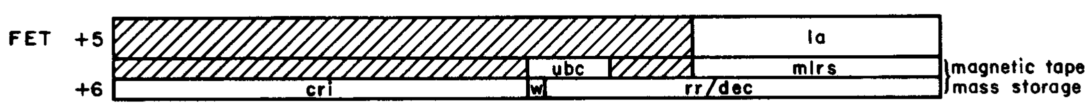
- xx1 or xx3 — Operation complete
- xx2 — Binary operation (applies only to SI, S, and L formatted tapes)
- xx0 — Coded operation (applies only to SI, S, and L formatted tapes)

The file mode (fm) bit (bit 1) of FET+0 is not actually part of the status response, although it is returned as such. The fm bit is used by tape drivers, in some cases, to determine parity (7-track) or whether conversion is required between character sets (9-track). For disk I/O the bit is carried for compatibility with tape I/O, but is meaningless. The bit is set by the FILEB or FILEC macros or directly by the user. After this it is masked in as part of the return code.

The CPU program is expected to issue an even request code (bit 0 = 0). If it does not, a completed operation may not be detected.

Words 1 through 7 of the FET contain the following information.

Words 1 through 4 and word 7 of the FET are the same as figures 2-3-4 and 2-3-5. Words 5 and 6 contain the following information.




la — Address of a list of random addresses to be used with READLS or RPHRLS mass storage operations.

ubc — Unused bit count for S and L format tapes.

mlrs — Maximum PRU size for S and L format tapes.

cri — Current random index (for mass storage files only).

w — Random rewrite request (for mass storage files only).

† These fields apply only to S and L format tapes.

| | | |
|---|---|---|
| rr/dec | rr | Random request (for mass storage files only): If rr $\neq$ 0, and the request is a read request, rr is the random index.<br>If rr $\neq$ 0, w=0, and the request is a write request, rr is the address for return of random index (the write operation is at the current position).<br>If rr $\neq$ 0, w=1, and the request is a write request, rr is the random index. |
| | dec | Detail error return code (for mass storage files only): |

| Code | Type of Error |
|---|---|
| x001 | Parity error |
| x002 | Address error |
| x003 | Device status error |
| x004 | 6681 function reject or function sent to mass storage device that timed out with no response |
| x005 | Device reserved |
| x006 | Device not ready |
| 4007 | Track limit (device full) |

If, after a read error (with ep bit set), the system determines that the correct PRU was read (although it may contain incorrect data) then x above is zero, the data is placed in the buffer, and the file is positioned to the next PRU of the file. If the correct PRU is not read, then x is 4, no data is placed in the buffer, and the file is not repositioned. The cri is set as usual.

Equipment which may be accessed by CIO includes:

Mass storage

Magnetic tape units

Card reader

Card punch

Line printers

Communications terminals through the time-sharing executive

## CIO FUNCTION PROCESSING

All of the CIO macros require two common decks for system interface.

- COMCCIO
- COMCSYS

These common decks are available to the user in relocatable form on the user library SYSLIB.

Error processing for functions issued to CIO involves processing only those errors that occur on the specified devices which includes read and write parity errors to magnetic tape. If a mass storage device returns an error status or the device driver detects an error, the system places the error status in the FET+0 status field. If the error processing bit is set in the FET+1 ep field and the FET length is greater than 5, a detail error code is returned to the user in the FET+6 dec field.

## RANDOM PROCESSING

A file that resides on mass storage can be randomly accessed by the user; that is, any PRU (64 60-bit words) on the file can be read without reading the entire file. † The random address of a PRU is the number of PRUs that precede the PRU on the file. For example, on file TEST:

| | TEST | |
|---|---|---|
| | 0 | BOI |
| A | 1 | |
| B | 2 | |
| C | 3 | |
| D | 4 | |
| E | 5 | |
| F | 6 | |

† Refer to Accessing Files, section 2, volume 1.

Record E has random address 5. The first random address that can be read or written is address 1. Sector 0 (for mass storage, a PRU is equivalent to a sector) on all mass storage files is reserved for system use.

The user can request that the system perform the specified read or write request at the file position specified by the random request word (FET+6). If the file specified resides on mass storage and the random processing bit is set (r parameter in word 1), then random access to the file can be performed. For a random write operation, the remainder of the file (the portion following the data written) is not released. On a sequential write operation, this portion of the file is released.

The user is responsible for managing the random addresses. For any CIO operation with r set in word 1, the system returns the current random index (cri) in FET+6. The cri is the position of the file when the operation is completed.

For a write operation, if rr=0, a sequential write is performed at the current position. If rr≠0, it represents either:

- The address at which the random write is to be performed (w=1), or
- The CM address which receives the position of the file before the write operation is performed (w=0). In this case, the write operation takes place at the current EOI.

For a REWRITE operation, if rr=0, a random write is performed at the current position. If rr≠0, a random write is performed at rr.

For the random file described in section 2, volume 1, the following random requests are necessary to perform the operations described.

| Operation | rr | w | cri Returned | Description |
|---|---|---|---|---|
| READ | 10 | 0 | 11 | Read directory |
| WRITER | 5 | 1 | 8 | Write new record 3 in place; |
| or | | | | |
| REWRITER | 5 | 0 | 8 | Word count = 138 |
| WRITER | 2000 | 0 | 15 | Write new record at EOI. Location 2000 is set to 138 to indicate where the write occurred. |
| WRITEF | 15 | 1 | 17 | Rewrite directory |
| or | | | | |
| REWRITEF | 15 | 0 | 17 | |

The user must account for the extra sector written for EORs and EOFs when specifying rewrite-in-place operations.

The system computes the number of sectors written as follows:†

† All numbers are in octal.

| Operation | Formula | Word Count | Example | PRUs Written | Data Remaining in Buffer |
|---|---|---|---|---|---|
| Buffer write | n/100 | 243 | 243/100 | 2 | 43 |
| EOR write | $\frac{(n+100)}{100}$ | 243 | $\frac{(243+100)}{100}$ | 3 | 0 |
| | | 300 | $\frac{(300+100)}{100}$ | 4 | 0 |
| EOF write | $\frac{(n+200)}{100}$ | 243 | $\frac{(243+200)}{100}$ | 4 | 0 |
| | | 300 | $\frac{(300+200)}{100}$ | 5 | 0 |

## CIO OPEN AND CLOSE FUNCTIONS

Two macros are available for opening files.

OPEN is applicable to all files

POSMF is applicable only to labeled multifile tapes

Two macros are available for closing files.

CLOSE is applicable to all files

CLOSER is applicable only to tape files

POSMF is described in the discussion of file positioning functions. OPEN, CLOSE, and CLOSER are described in the following paragraphs.

### OPEN

OPEN creates a file or determines certain information about a file for a job. If the file does not exist before the request to OPEN, it is created.

Macro Format:

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | OPEN | addr, type, r |

addr — Address of the FET for the OPEN request.

type — Type of function to be performed:

| Type | Function (With Code in Octal) |
|---|---|
| READNR | Read, no rewind (100) |

| | |
|---|---|
| READ | Read and rewind (140) |
| WRITENR | Write, no rewind (104) |
| WRITE | Write and rewind (144) |
| NR | No rewind (120) |
| ALTERNR | Alter, no rewind (120) |
| ALTER | Alter and rewind (160) |
| REELNR | Read reel, no rewind (300) |
| REEL | Read reel and rewind (340) |

If the user specifies a function type of REEL (or REELNR) and the file resides on mass storage, CIO assumes the type is READ (or READNR). If the type is not specified, ALTER is assumed. The functions listed do not change the system read/write lock on the file.

r — Auto recall, if desired.

The information supplied by OPEN includes:

- The PRU size for the device to which the file is assigned (FET+4, bits 35 through 18). The PRU sizes returned for a magnetic tape file depends on the tape data format.

| Tape Format | PRU Size in Words |
|---|---|
| I | 1000 octal |
| SI | 1000 octal |
| X | 1000 octal |
| E | 16 octal by default (136 characters rounded up) or value calculated from FC or C parameter value on ASSIGN, LABEL, or REQUEST control statement. |
| B | 17 octal default (150 characters) or value calculated from FC or C parameter value on ASSIGN, LABEL, or REQUEST control statement. |
| F | Value calculated from required FC or C parameter values on ASSIGN, LABEL, or REQUEST control statement. |
| S | If mlrs = 0 (FET +6, bits 17 through 0), PRU size is 1000 octal; if mlrs ≠ 0, PRU size = mlrs. |
| L | If mlrs = 0, PRU size is LIMIT minus FIRST minus 1; if mlrs ≠ 0, PRU size is mlrs. |

- The type of device on which the file resides (FET+1, bits 59 through 48). With one exception, the device types returned are the same as those returned for the STATUS macro. For magnetic tape operations, the device type returned for STATUS is either MT (7-track) or NT (9-track). For OPEN, the device type is MT or NT for I, X, E, B, and F format tapes; for SI, S, and L format tapes, the device type is:

| | | |
|---|---|---|
| 40nn | Seven-track tape, where nn specifies the recording technique. The 6-bit binary representation of nn is: | |
| | xxxx10 | HY density (800 bpi)† |
| | xx00xx | Unlabeled |
| | xx01xx | Standard ANSI labels |
| | 00xxxx | SI data format |
| | 10xxxx | S data format |
| | 11xxxx | L data format |
| 41nn | Nine-track tape, where nn specifies the recording technique. The 6-bit binary representation of nn is: | |
| | xxxx10 | HD density (800 cpi)† |
| | xx00xx | Unlabeled |
| | xx01xx | Standard ANSI labels |
| | 00xxxx | SI data format |
| | 10xxxx | S data format |
| | 11xxxx | L data format |

  This device type format is used for compatibility with the NOS/BE 1 Operating System.

The PRU size and device type are returned only when the FET length is greater than five words (refer to FET Description).

The operations performed by OPEN depend on the type of device being used.

### Mass Storage Operations

For mass storage files, if random processing is specified and proper pointers are set in FET +7, the last record of the file is loaded into the buffer specified. The random index on a file that is to be OPENed is expected to be the last record before the EOI. No EOFs may intervene. No indication of the length of the index loaded is returned other than a zero word in the buffer (OPEN clears the buffer before loading the index). If the buffer is too small to accommodate the entire index, the excess data is lost. The random access bit (FET + 1, bit 47) is cleared during an OPEN operation if:

1. The last record before the EOI is empty.
2. Index area FWA is less than 2 (FET+7, bits 17 through 0).
3. The file does not reside on mass storage.

For all OPEN functions on mass storage, the index for a file is loaded into the index area specified.

† Density is always returned as 800 bpi or cpi.

**Unlabeled Tape Processing**

If a no rewind option is specified (codes 100, 104, 120, or 300), the tape remains at its current position. If a rewind option is specified (codes 140, 144, 160, or 340), the tape is rewound to the load point of the current volume.

**Nonstandard Labeled Tape Processing**

Since the system cannot write nonstandard labels, the job aborts with the following dayfile message if a WRITE (144) or WRITENR (104) function is specified.

ILLEGAL LABEL TYPE.

If a no rewind option is specified (codes 100, 120, or 300), the tape remains at its current position. If a rewind option is selected (codes 140, 160, or 340), the tape is rewound to the load point of the current volume.

Nonstandard labels are not read or returned to the CIO buffer. If the tape is at the load point, a subsequent read operation skips to the first tape mark before the read occurs.

**ANSI Standard Labeled Tape Processing**

When an ANSI labeled tape file is opened, the action the system takes depends on the xl bit (FET+1, bit 41). If xl is 0, standard label processing is performed; extended label processing is performed if xl is 1.

For standard label processing, all optional labels are ignored. If the FET for the file is at least 13 words long, FET+9 through FET+12 (refer to FET Description) contain the HDR1 data. These fields are described in detail in appendix G, volume 1, HDR1 label. All fields contain alphanumeric or numeric display code values. Alphanumeric fields are left-justified with binary zero or blank fill. Numeric fields contain display code numeric digits and are right-justified with display code zero fill.

The tape remains at its current position if a no rewind option is specified (codes 100, 104, 120, or 300). If a rewind option is selected (codes 140, 144, 160, or 340), the tape is rewound to the load point of the current volume.

The system reads and/or verifies the HDR1 label if the tape is at the load point and a READ, REEL, or ALTER option is selected (codes 100, 120, 140, 160, 300, or 340), with the following restrictions or requirements.

- If the FET length is less than 13 words, the system accepts standard label without verification.
- If the FET length is at least 13 words, the HDR1 data in FET+9 through FET+12 is compared with that in the HDR1 label. Binary zero fields are not compared, although the actual value read is returned to the field. If any nonzero field does not match, the job aborts and the following message is issued to the dayfile.

  LABEL PARAMETER CONFLICT ON OPEN, fff AT nnn.

  FIELD BEGINNING AT CHARACTER yy NO COMPARE.

  In this message, fff is the file name, nnn is the FET address, and yy is the decimal character position in HDR1 at which the field begins.

A nonzero retention cycle (FET+11, bits 47 through 30) is used to calculate an expiration date that is compared with that HDR1 field on the tape.

The HDR1 label is transferred to the CIO buffer (as space permits), although IN and OUT pointers are not updated to reflect the label information in the buffer.

- All optional labels are ignored.

The system writes a new HDR1 label if the tape at load point is opened for the first time with a WRITE operation specified (codes 104 or 144), and the FET length is at least 13 words. Subsequent OPEN/WRITE operations (following an OPEN/READ or OPEN/WRITE) do not update the label information, even if the file is opened, closed, and then reopened. The following restrictions or requirements apply.

- If the FET is less than 13 words in length, the previous HDR1 label information is not changed.
- If the FET length is at least 13 words, the system uses the information in FET+9 through FET+12 for the HDR1 label. If any of the FET HDR1 fields are binary zero, the system uses the default value. The current date is used instead of the create date field in the FET. A nonzero retention cycle field is used to calculate the expiration date (default is current date).
- If a nonnumeric value is encountered in a numeric field, the job is aborted and the following message is issued to the dayfile.

  ILLEGAL XL BUFFER/FET LABEL FIELD, lfn AT nnn.

- Previous HDR1 label expiration date is enforced; if not expired, the job aborts with the following message issued to the dayfile.

  LABEL NOT EXPIRED.

### Extended Label Processing

For extended label processing, a user label buffer, rather than the FET, is used to hold labels for processing. Extended label processing requires a FET length of at least 10 words, and an extended label buffer. Extended label processing is disabled if these requirements are not met.

The buffer location must be specified in FET+9 as follows:

| | 59 | 35 | 17 0 |
|---|---|---|---|
| FET+9 | | length of label buffer | fwa of label buffer |

Within the buffer, each label must be preceded by a status word.

| 59 | 11 0 |
|---|---|
| | characters in label |

Only bits 11 through 0 should be set by the user to show the number of characters in the label. This value must be 80 ($120_8$). If it does not equal 80, the job is aborted and the following message is issued to the dayfile.

ILLEGAL XL BUFFER/FET LABEL FIELD, lfn AT nnn.

Remaining fields may be used by the label processor. The last label should be followed by a status word containing zeros in bits 11 through 0. Each label in the buffer appears, in display code, with the same format it has on the tape.

The tape remains at its current position if a no rewind operation is specified (codes 100, 104, 120, or 300). If a rewind option is selected (codes 140, 144, 160, or 340), the tape is rewound to the load point of the current volume.

If the tape is at the load point and READ, REEL, or ALTER is selected (codes 100, 120, 140, 160, 300, or 340), the system reads all labels to the first tape mark and verifies the HDR1 label, with the following exceptions.

- If the label buffer does not contain an HDR1 label, the system accepts the standard label without verification.

- If the label buffer contains an HDR1 label, any nonzero field in the label buffer is compared with that HDR1 field on the tape. The job aborts with the following message if any nonzero field does not match.

  LABEL PARAMETER CONFLICT ON OPEN, fff AT nnn.

  FIELD BEGINNING AT CHARACTER yy NO COMPARE.

  In this message, fff is the file name, nnn is the FET address, and yy is the decimal character position in HDR1 at which the field begins.

- All labels from VOL1 through the first tape mark are transferred to the label buffer as space permits. Verification of additional labels are the user's responsibility.

The system writes a new HDR1 label if the tape at load point is opened for the first time with a WRITE operation specified (codes 104 or 144). If the file has been opened prior to the OPEN/WRITE, the label information is not updated, even if the file is opened, closed, and reopened. Further requirements and restrictions are as follows.

- VOL1 labels in the label buffer are ignored.

- If an HDR1 label is not present in the label buffer, the system uses default values to create the HDR1 label for the tape.

- If an HDR1 label is present in the label buffer, it is used to generate the HDR1 label on the tape. If any field in the label buffer is binary zero, the default value for that field is used. The current date is used instead of the create date in the buffer. The expiration date field is used; if zero, it defaults to the current date.

- If a numeric field in the HDR1 label contains a nonnumeric value, the job is aborted and the following message is issued to the dayfile.

  ILLEGAL XL BUFFER/FET LABEL FIELD, lfn AT nnn.

- Previous HDR1 label expiration date is enforced; if not expired, the job aborts with the following message issued to the dayfile.

  LABEL NOT EXPIRED.

- All user labels to be written must be present in the label buffer. All user volume labels (UVL1-9), additional file header labels (HDR2-9), and user header labels (UHL) in the label buffer are written to the tape. Nonapplicable labels are ignored.

## CLOSE

CLOSE terminates operations on a file.

Macro Format:

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | CLOSE | addr, type, r |

addr — Address of the FET for the CLOSE request

type — Type of function to be performed:

| Type | Function (With Code in Octal) |
|---|---|
| NR | File is not rewound (130) |
| REWIND | File is rewound (150) |
| UNLOAD | Tape file is rewound and unloaded; mass storage file is released (170) |
| RETURN | Tape file is rewound and the number of tape units scheduled for the job is decreased by one; mass storage file is released (174) |

If type is not specified, rewind is assumed.

r — Auto recall, if desired

If the file resides on mass storage, the random processing bit is set, and the proper parameters are set in FET+7, CLOSE writes the data in the buffer specified in FET+7 at the EOI of the file. This normally is the index for the file. A random index is written at the EOI only if the file is random, the file has been written on since the last OPEN request, the file is not locked, and an index area is specified in FET+7, bits 17 through 0.

If the name of the file in FET+0 is PUNCH, PUNCHB, or P8, the file is entered in the punch queue. The file is punched in O26 or O29 mode, depending upon the origin of the job issuing the CLOSE request. If the job is not local batch origin, then the coded file is punched according to the system default keypunch mode (as specified by the installation). If the job is local batch origin, the initial keypunch mode of the job's control statement record (that is selected by the job statement or installation parameter) is the mode in which the deck is punched.

When a magnetic tape file is closed, the action the system takes depends on the last I/O operation performed.

The system responds to a CLOSE request on a magnetic tape file in the following manner.

Step 1. Last operation: Write

a. If the tape is unlabeled and the data format is X, S, L, E, B, or F, the system writes four tape marks.

b. If the tape is unlabeled and the data format is I or SI, the system writes a tape mark, an EOF1 label, and two tape marks.

c. If the tape is labeled and standard label processing is in effect (xl, bit 41 of FET +1, not set), the system writes a tape mark, an EOF1 label, and two tape marks.

d. If the tape is labeled and extended label processing is in effect (xl, bit 41 of FET +1, set), the system writes a tape mark, an EOF1 label, all user end-of-file labels (EOF2-9) and user trailer labels (UTL) present in the extended label buffer, and two tape marks. All nonapplicable labels, including EOF1 and EOV1 labels, in the extended label buffer are ignored. Refer to extended label processing under the OPEN macro for a description of the label buffer.

Step 2. Last operation: Read

If the tape is labeled, extended label processing is in effect (xl, bit 41 of FET +1, set), and a tape mark immediately follows, all labels from this tape mark (beginning with EOF1) through the next tape mark are transferred to the extended label buffer, as space permits, beginning at the first word of the buffer.

Step 3. No rewind option selected

Tape remains positioned at or is repositioned to the same point as before the CLOSE was issued (to prevent user from going past EOI).

Step 4. Rewind option selected

The system rewinds the tape to the beginning of data of the current file. This operation is performed automatically even if the current file begins on another reel.

Step 5. Unload option selected

The system rewinds and unloads the current tape reel, releasing job and file attachment.

Step 6. Return option selected

a. The system rewinds and unloads the current tape reel, releasing job and file attachment.

b. The number of tapes scheduled for the job is decremented only if the total concurrent resource demand (tapes and packs) has been satisfied.

## CLOSER

The CLOSER macro closes a magnetic tape reel.

Macro Format:

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | CLOSER | addr, type, r |

addr — Address of the FET

type — Type of operation to be performed:

| Type | Function (With Code in Octal) |
|---|---|
| NR | No rewind (330) |
| UNLOAD | Unload (370) |
| omitted | Rewind (350) |

If the specified file resides on mass storage, the CLOSER 350 operation sets the file to BOI and 370 releases the file.

r — Auto recall, if desired

The CLOSER macro enables the user to control end-of-reel processing. The definition of end of reel varies according to the processing option the user selects (refer to End-of-Tape/End-of-Reel Conditions, section 10, volume 1). The action the system takes in response to a CLOSER request depends on two factors:

- The last I/O operation performed
- The user processing option (FET+1, bit 45)

and is defined as follows:

Step 1. Last operation: Write

a. If the tape is labeled, unlabeled SI, or unlabeled I format, the system writes a tape mark followed by an EOV1 label and two tape marks. If the user has specified the vsn of the next reel, † an EOV2 label containing that value is also written following the EOV1. User trailer labels present in the label buffer are written if extended label processing is in effect. Refer to Extended Label Processing, OPEN macro for a description of the label buffer.

b. If the tape is unlabeled and the data format is X, S, L, E, B, or F (refer to section 10, volume 1), the system writes four tape marks.

Step 2. Last operation: Read

a. If extended label processing is in effect, all labels following from EOV1 to the next tape mark are returned to the extended label buffer.

b. If the tape has an EOV2 label (except for SI unlabeled tapes) and was written under NOS, the system extracts the vsn and proceeds to step 3.

† Refer to section 10, volume 1 for a description of the VSN control statement.

c. If the user or operator has specified the vsn of the next reel, the system proceeds to step 3 or 4.

d. If the vsn of the next reel has not been specified and the tape has no EOV2 label, the system proceeds to step 3 or 4.

Step 3. User processing option selected.

The system returns control to the user with end-of-reel status in the FET. The tape is positioned so that if the user attempts to perform another I/O operation, the result is the same as for the previous read or write. This prevents a read or write operation from running off the end of the tape.

Step 4. User processing option not selected.

The system sets the completion bit in the FET. In addition, if the tape is labeled and the FET length is at least 14 words, the system increments the file section number (FET+12, bits 23 through 0).

Step 5. The system rewinds or unloads the tape as specified by the type parameter.

Step 6. If the vsn of the next reel is known, the I/O operation continues. If the vsn is not known, the system requests the operator to supply it and then continues the operation on the next reel.

## CIO READ FUNCTIONS

The following read functions are processed by CIO.

### RPHR (000)

RPHR causes one PRU to be transferred into the circular buffer.

The status responses (bits 9 through 0 of word 0 of the FET) are:

| | |
|---|---|
| 0001 | Full sector |
| 0021 | EOR encountered |
| 0031 | EOF encountered |
| 1031 | EOI encountered |

Macro Format:

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | RPHR | addr, r |

| | |
|---|---|
| addr | Address of the FET |
| r | If r is specified, control is returned only upon completion of the operation. |

## READ (010)

The READ function reads information into the circular buffer. If there is room in the buffer for at least one physical record, the system initiates reading and continues until:

- The buffer is full.
- An end of record or end of file is encountered.
- The end of information is encountered.
- For S and L format tapes, one PRU is read.

The status responses (bits 9 through 0 of word 0 of the FET) are:

| | |
|---|---|
| 0011 | Buffer filled |
| 0021 | EOR encountered |
| 0031 | EOF encountered |
| 1031 | EOI encountered |

Data is not transferred after an EOR or EOF mark is encountered. For tapes that do not have a defined EOI (refer to section 10, volume 1), an operation that normally would terminate at EOI terminates instead at EOF. Also, for S and L format tapes, the unused bit count is returned to FET+6, bits 29 through 24, when the read is complete.

Macro Format:

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | READ | addr, r |

| | |
|---|---|
| addr | Address of the FET |
| r | If r is specified, control is not returned until operation is complete. |

## READSKP (020)

READSKP performs a read function until the buffer if filled or until an EOR or EOF is encountered. If the buffer is filled before an EOR is encountered, CIO positions the file at the next EOR, EOF, or EOI, whichever is encountered first.

The status responses are:

| | |
|---|---|
| 0021 | Buffer filled or EOR encountered |
| 0031 | EOF encountered |
| 1031 | EOI encountered |

For tapes that do not have a defined EOI (refer to section 10, volume 1), an operation that normally would terminate at EOI will terminate instead at EOF.

This is the only read function that performs the read operation if less than one PRU of space is available in the buffer.

Macro Format:

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | READSKP | addr, level, r |

addr — Address of the FET

level — Level number (0 through $17_8$) specified in FET+0, bits 17 through 14; if a level number is specified, information is skipped until the occurrence of an EOR with a level number greater than or equal to the one specified:

| | |
|---|---|
| 0 | After the buffer is full, skip to the next EOR. |
| $17_8$ | After the buffer is full or a full record is placed in the buffer, skip to the next EOF. |

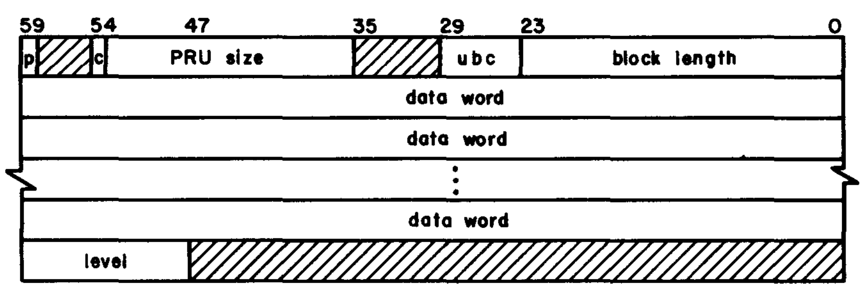
For S and L format tapes, only a request with level $17_8$ is recognized; any other level in the request is ignored.

r — If r is specified, control is returned only upon completion of the operation.

For S and L format tapes, the user should set the mlrs field (FET+6, bits 17 through 0) before issuing the READSKP function. If mlrs is 0, the system assumes 512 words for an S tape and LIMIT-FIRST-1 for an L tape. For L format tapes, mlrs must be set to a value at least as long as the largest block on the tape.

## READCW (200)

The READCW function performs a nonstop read of PRUs bounded by control words. The PRU format is:



| | |
|---|---|
| p | Parity error indication (is set for each block in error when reading with the ep bit set in the FET). |
| c | Bit 54 is set if coded operation (tape operations only). |
| PRU size | Number of CM words in each PRU on the device (refer to the OPEN macro for a description of PRU sizes). |
| ubc | Unused bit count ($0 \leq ubc \leq 11$). Ignored for mass storage files. Checked for all tape formats except I, SI, and X; for I, SI, and X format tapes, ubc must be 0. For read and write operations on tape, ubc is processed as accurately as possible within the constraints of the hardware. |
| block length | Count of the number of 12-bit data bytes in the PRU. For mass storage files and I, SI, and X format tapes, it must be equal to five times the number of CM words occupied by the data. |
| level | Logical record level number: |

| | |
|---|---|
| 0 or absent | The PRU is an EOR |
| $17_8$ | The PRU is an EOF |

This function allows the user to read nonstop and detect EORs and EOFs without having to recall CIO for the next sequential read. Reading terminates normally if the buffer becomes full or if the EOI is detected. If the request is made with the level number equal to $17_8$ (that is, FET+0, bits 17 through 14), reading will stop at the next EOF. This function may only be used with mass storage and magnetic tape devices.

Macro Format:

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | READCW | addr, level, r |

| | |
|---|---|
| addr | Address of the FET |
| level | Termination level |

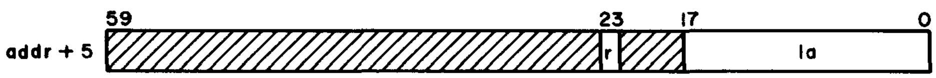
| | |
|---|---|
| 0 | Continue reading over EOFs (stops at EOI or buffer full). EOFs are returned in data as zero length block with level $17_8$. |
| $17_8$ | Stop reading at next EOF. EOF status is returned to the FET, but no EOF data block (zero length and level $17_8$) is returned to the CIO buffer. |

| | |
|---|---|
| r | If r is specified, control is returned only upon completion of the operation. |

**READLS (210)**

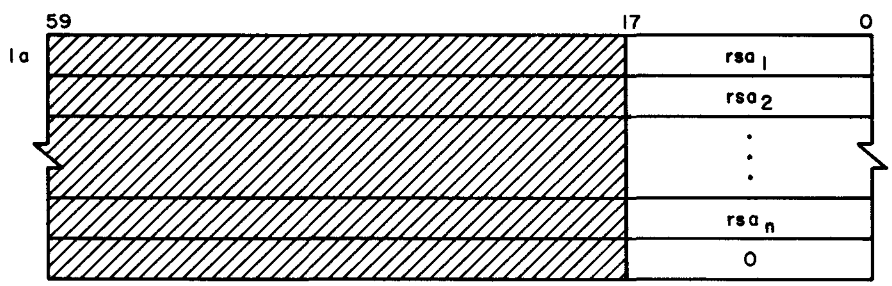
The READLS function reads the group of mass storage logical records specified by a list supplied by the user. The user must supply the address of the list in the lower 18 bits of word 5 of the FET specified by addr. READLS continues reading until the list is exhausted or the buffer is filled.

Word 5 of the FET contains the address of the list in the following format.




| | |
|---|---|
| r | Set by CIO if this is not the initial READLS call |
| la | Address of the list |

The list is in the following format.




| | |
|---|---|
| $rsa_i$ | Random sector address |

The status responses are:

| | |
|---|---|
| 0031 | Operation complete. Entire list read. The value of la in FET+5 is set to terminator word. |
| 0211 | Operation not complete. The value of la in FET+5 is set to next entry of list to be processed. If the buffer is full, r is set to 1. If the buffer is not full, CIO has reached an internal limit and has stopped processing the list. After emptying the buffer, CIO should be called again to continue processing the list. |

Macro Format:

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | READLS | addr, r |

addr Address of the FET

r If r is specified, control is returned only upon completion of the operation.

**RPHRLS (230)**

The RPHRLS function reads the group of mass storage PRUs specified by a list supplied by the user. This function performs the same operation as READLS except that each address in the list specifies a single PRU instead of a record. After the single PRU specified by each list entry is placed in the buffer, the list position is advanced.

Macro Format:

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | RPHRLS | addr, r |

addr Address of the FET

r If r is specified, control is returned only upon completion of the operation.

### READNS (250)

The READNS function reads a file from the current position to an EOF.

The status responses are:

| | |
|---|---|
| 0251 | Buffer full |
| 0031 | EOF encountered |
| 1031 | EIO encountered |

Macro Format:

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | READNS | addr, r |

| | |
|---|---|
| addr | Address of the FET |
| r | If r is specified, control is returned only upon completion of the operation. |

### READN (260)

The READN function reads data from an S or L format tape into the circular buffer. Reading continues until:

- The buffer is full.
- An EOF is encountered.
- The EOI is encountered.

Status responses are:

| | |
|---|---|
| 0261 | Buffer full |
| 0271 | EOF encountered |
| 1271 | EOI encountered |

Macro Format:

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | READN | addr, r |

addr    Address of the FET

r    If r is specified, control is returned only upon completion of the operation.

Before this function is issued, the mlrs field in FET+6, bits 17 through 0, must be set to the largest physical record that will be encountered. For S format, if mlrs=0, the value of the maximum block is assumed to be 512 words. For L format, if mlrs=0, the assumed maximum block is LIMIT-FIRST-2. In addition, the file mode (FET+0, bit 1) must be set.

Each physical record in the circular buffer is preceded by a header word. This word is generated by the system; it does not exist on the tape. The format of the header word is:

| 59 | 29 | 23 | 17 0 |
|---|---|---|---|
| | ubc | | block size |

ubc    Unused bit count. Number of bits in the last word that are not valid data; ubc may range from 0 to 55.

block size    Number of CM words in the physical record

After each complete physical record has been placed in the buffer, the system moves the IN pointer to reflect both the header and the data.

**READEI (600)**

The READEI function reads information into the circular buffer. Reading continues until an EOI mark is encountered or the buffer is filled. The status response is EOI encountered (1031). The file is positioned at EOI.

Macro Format:

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | READEI | addr, r |

addr    Address of the FET

r    If r is specified, control is returned only upon completion of the operation.

## CIO WRITE FUNCTIONS

The following write functions are processed by CIO.

### WPHR (004)

WPHR writes one physical record from the circular buffer. Unless the buffer contains at least one PRU, no operation occurs.

Macro Format:

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | WPHR | addr, r |

addr    Address of the FET

r    If r is specified, control is not returned until the operation is complete.

### WRITE (014)

WRITE transfers the contents of the circular buffer to the specified file. Writing continues until the buffer contains less than one PRU of data (a WRITER request empties the buffer).

Macro Format:

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | WRITE | addr, r |

addr    Address of the FET

r    Auto recall option:

*    If the symbol WRIF$ is defined in assembly, the common decks reissue the CIO request set in the FET instead of a WRITE function. The * option sets a WRITE request ($14_8$) in the FET. When the * option is used, CIO is not called.

other    Control is not returned until the operation is complete.

For S and L format tapes, only one record is written for each request. The length the record is determined by the value of the IN and OUT pointers.

## WRITER (024)

WRITER writes the entire contents of the buffer to the file specified.† The last PRU is written as a short PRU (refer to Data Formats, section 10, volume 1). If the data exactly fills the last PRU, the system adds a PRU with no data to indicate the end of the record. A WRITER request with level $17_8$ set in FET+0, bits 17 through 14, performs the same operation as a WRITEF request.

Macro Format:

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | WRITER | addr, r |

addr    Address of the FET

r    If r is specified, control is not returned until the operation is complete.

## WRITEF (034)

WRITEF writes the entire contents of the buffer to the file specified. † The last PRU written is the end of file. If data does not exactly fill the last PRU, the system writes a short PRU (refer to Data Formats, section 10, volume 1) and an EOF. If the buffer is empty and the last operation was an EOR or EOF, the system writes an EOF; otherwise a PRU with no data (EOR) and an EOF is written. For S and L format tapes, data in the buffer is transferred to tape and followed by a tape mark.

Macro Format:

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | WRITEF | addr, r |

addr    Address of the FET

r    If r is specified, control is not returned until the operation is complete.

†The OUT pointer is updated. The IN pointer is not changed.

## WRITECW (204)

The WRITECW function performs a nonstop write of PRUs bounded by control words. The PRUs are in the same format as specified for READCW. Data written using this function is stored on the device in the same format as if it had been written with any other write function (that is, the control words are not part of the data).

WRITECW may only be used with mass storage and magnetic tape devices.

Macro Format:

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | WRITECW | addr, r |

addr    Address of the FET

r    Auto recall option:

- *    If the symbol WRIF$ is defined in assembly, the common decks reissue the CIO request set in the FET instead of a WRITE function. The * option sets a WRITECW request ($204_8$) in the FET. When the * option is used, CIO is not called.
- other    Control is not returned until the operation is complete.

## REWRITE (214)

REWRITE performs the same operation as the WRITE function with the exception that it causes the system to process the operation as a random function; that is, that portion of the file following the portion written is not destroyed. If the random parameters (r, rr, and w) are not specified in the FET, the write operation takes place at the current position. If the random parameters rr and w are specified, the normal random addressing procedures are followed. The file to be rewritten must reside on mass storage.

**NOTE**

REWRITE does not check if the record being rewritten in place is the same size or less than the original record.

Macro Format:

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | REWRITE | addr, r |

addr     Address of the FET

r     If r is specified, control is not returned until the operation is complete.

**REWRITER (224)**

REWRITER performs the same task as WRITER with the exceptions noted for REWRITE. The file must reside on mass storage. If the level number is $17_8$, REWRITER performs the same operation as REWRITEF.

Macro Format:

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | REWRITER | addr, r |

addr     Address of the FET

r     If r is specified, control is not returned until the operation is complete.

**NOTE**

A short PRU is written even if the last PRU is exactly full.

**REWRITEF (234)**

REWRITEF performs the same task as WRITEF with the exceptions noted for REWRITE. The file specified must reside on mass storage.

Macro Format:

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | REWRITEF | addr,r |

addr — Address of the FET

r — If r is specified, control is not returned until the operation is complete.

**NOTE**

An extra PRU is written to specify an EOF.

## WRITEN (264)

The WRITEN macro writes nonstop on an S or L formatted magnetic tape. S and L formatted tapes are described in Data Formats, section 10, volume 1. Writing continues until the buffer is empty or end of reel is encountered.

Macro Format:

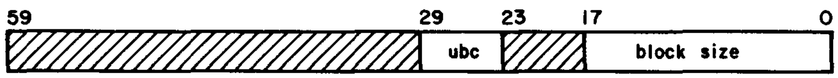
| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | WRITEN | addr, r |

addr — Address of the FET

r — Auto recall option:

- * — If the symbol WRIF$ is defined in assembly, the common decks reissue the CIO request set in the FET instead of a WRITE function. The * option sets a WRITEN request ($264_8$) in the FET. When the * option is used, CIO is not called.
- other — Control is not returned until the operation is complete.

The user must provide a header word immediately preceding each record in the buffer. This header is not physically written on the tape. Its format is:




ubc — Unused bit count. Number of bits that are not valid data in the last word; ubc may range from 0 to 55.

block size — Number of CM words in the physical record

The system compares the mlrs and ubc fields in FET+6 using information from this header.

The OUT pointer is not changed to reflect the move until after each complete record has been written to tape.

## FILE POSITIONING FUNCTIONS

The following functions control the positioning of a file. If the FET indicates that the file is being accessed randomly, the random address of the new position (cri) always returned.

### BKSP (040)

BKSP causes a file to be backspaced one logical record. If BOI is encountered before backspacing is complete, a rewind status is returned (05x). If the backspace causes the file to be positioned exactly at BOI, a backspace status is returned (041).

Macro Format:

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | BKSP | addr, r |

addr — Address of the FET

r — If r is specified, control is not returned until the operation is complete.

**BKSPRU (044)**

BKSPRU causes the file to be backspaced the specified number of physical records. If BOI is encountered before the specified number of PRUs is backspaced, a rewind status is returned (05x). If the operation causes the file to be positioned at BOI, the backspace status is returned (045). The skip count is set in the RA+1 call to CIO. (Refer to the format of the call to CIO.)

Macro Format:

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | BKSPRU | addr, n, r |

addr — Address of the FET

n — Number of PRUs to backspace

r — If r is specified, control is not returned until the operation is complete.

**REWIND (050)**

REWIND causes a mass storage file to be positioned at BOI and a magnetic tape file to be positioned at the beginning of the current file. If the file does not exist, no operation is performed.

Macro Format:

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | REWIND | addr, r |

addr — Address of the FET

r — If r is specified, control is not returned until the operation is complete.

For a mass storage file, if the random processing bit is set in the FET, the current random index (cri) is returned as the beginning of the file (random address 1).

If the file resides on magnetic tape, the action the system takes depends on the last I/O operation performed.

The system responds to a REWIND request as follows:

Step 1. Last operation: Write

a. If the tape is labeled, the system writes a tape mark, an EOF1 label, and three tape marks.

b. If the tape is unlabeled and the data format† is X, S, L, E, B, or F, the system writes four tape marks.

c. If the tape is unlabeled and the data format is I or SI, the system writes a tape mark, an EOF1 label, and three tape marks.

Step 2. Last operation: Read

The system proceeds to step 3.

Step 3. The system rewinds the tape to the beginning of data of the current file. This operation is performed automatically even if the current file begins on another reel.

## UNLOAD (060)

UNLOAD causes the specified file to be rewound and unloaded. If the file resides on mass storage, UNLOAD performs the same function as RETURN.

Macro Format:

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | UNLOAD | addr, r |

addr Address of the FET

r If r is specified, control is not returned until the operation is complete.

If the file resides on magnetic tape, the action the system takes depends on the last I/O operation performed.

The system responds to an unload request as follows:

Step 1. Last operation: Write

a. If the tape is labeled, the system writes a tape mark, an EOF1 label, and three tape marks.

b. If the tape is unlabeled and the data format is X, S, L, E, B, or F, the system writes four tape marks.

† Refer to section 10, volume 1.

c. If the tape is unlabeled and the data format is I or SI, the system writes a tape mark, an EOF1 label, and three tape marks.

Step 2. Last operation: Read

a. The system proceeds to step 3.

Step 3. The system rewinds and unloads the tape.

If an end-of-reel exists (was encountered on a previous CIO function while the user processing option was selected), a subsequent UNLOAD writes the end-of-volume trailer (refer to end-of-reel processing, section 10, volume 1) before rewinding and unloading the reel.

### RETURN (070)

RETURN causes the specified file to be released from control of the job. The operation performed depends on the type of file.

| Type | Operation |
|---|---|
| Input | The file name is changed to INPUT*; file space is not released and file INPUT* remains attached to the job as a local file. |
| Print | File space and job attachment are released. |
| Punch | File space and job attachment are released. |
| Local | File space and job attachment are released. |
| System | Job attachment is released but file space remains. |
| Library | Job attachment is released but file space remains. |
| Primary | File space and job attachment are released. |
| Permanent | Job interlock (read/write) is cleared; job attachment is released; file space remains. |

If the file resides on magnetic tape, the RETURN macro performs the same function as the UNLOAD macro. In addition, the RETURN of a magnetic tape file or the user's last direct access file for a particular disk pack decrements the resource demand count (as scheduled by the RESOURC control statement) only if the total concurrent resource demand (tapes and disk packs) has been satisfied. If the file is a deferred routed file (refer to section 8), the file space and job attachment are released.

Macro Format:

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | RETURN | addr, r |

addr Address of the FET

r If r is specified, control is not returned until the operation is complete.

## POSMF (110)

The POSMF macro opens and/or positions standard ANSI-labeled multifile magnetic tape sets to a member of the set. The file to be opened is determined by the contents of the label fields of the FET, or if the xl bit is set, the contents of the HDR1 label in the extended label buffer. The relative position of the file within the multifile set is specified by the file sequence number field.

Macro Format:

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | POSMF | addr, r |

addr — Address of the FET†

r — If r is specified, control is not returned until the operation is complete.

### Standard ANSI-Labeled Multifile Set Processing

If the FET length is less than 13 words, the job aborts with the following dayfile message.

BUFFER ARGUMENT ERROR ON fff AT nnn.

In this message, fff is the file name and nnn is the FET address.

The PRU size (FET+4, bits 35 through 18) and device type (FET+1, bits 59 through 48) are returned to the FET as described under the OPEN macro. However, the device type returned by POSMF for SI, S, and L formatted multifile set tapes is in the following format.

42nn — 7-track; recording technique same as for OPEN (40nn)

43nn — 9-track; recording technique same as for OPEN (41nn)

A multifile set is positioned to read an existing file if the file sequence number in the FET is not display code 999. The following restrictions and requirements also apply.

- The tape is positioned to the first member of the multifile set whose HDR1 fields match the HDR1 data in FET + 9 through FET + 12. Binary zero fields are not compared. A nonzero retention cycle is used to calculate an expiration data that is compared with that on the tape. If all fields do not match, the search continues.

† To accommodate COBOL, if the initial name of the assigned tape file is a 6-character name (for example, ABCDEF) and a POSMF to a file named ABCDEFn (n is alphanumeric character) is performed, the 6-character file name is automatically changed to the 7-character name. Subsequent POSMFs to other 7-character file names with the same first six characters also cause the file name to be changed. COBOL uses the first characters of the file name as the SETID.

- If the explicit or implicit position number is greater than that of the last member file (that is, the matching HDR1 was not found), POSMF is terminated, returning an end-of-set status (21 octal set in bits 13 through 9 of FET+0) and updating the file sequence number field in the FET to one greater than that of the last member file. The HDR1 for the last member file is transferred to the CIO buffer, although IN and OUT are not updated.

- If the desired file is located, the HDR1 label is transferred to the CIO buffer, as space permits, although the IN and OUT pointers are not updated to reflect the label information in the buffer. Actual values read are returned to the label FET fields.

- All optional labels are ignored.

A multifile set is positioned to write a new file if the file sequence number in the FET is display code 999. Further requirements are as follows.

- The first file of a multifile set is created if all of the following are true (refer to examples 1 and 2):

  1. The tape is positioned to the first file.

  2. The last operation was not a write.

  3. The set identification field in the initial HDR1 label on the tape is all blanks (the set identification field in the FET must be nonblank if the set is to be extended in the future).

- The multifile set is positioned to extend the set in all other cases. The tape is positioned to after the last file in the multifile set.

- The system writes a new HDR1 label to the tape with the following results.

  1. The file sequence number of the new HDR1 label is set to display code 0001 for the first file or set to the last member sequence number plus 1 for extended files.

  2. The section number in the HDR1 label is always set to display code 0001.

  3. With the above exceptions, the information in FET+9 through FET+12 is used to create the HDR1 label. The default value is used for any FET HDR1 field that is binary zero. If a numeric field contains a nonnumeric value, the job is aborted and the following message is issued to the dayfile.

     ILLEGAL XL BUFFER/FET LABEL FIELD, lfn AT nnn.

     The creation date field in the FET is ignored; the current date is always used.

## Extended Label Multifile Set Processing

If the FET length is less than 10 words, the job aborts with the following message issued to the dayfile.

BUFFER ARGUMENT ERROR ON fff AT nnn.

In this message, fff is the file name and nnn is the FET address.

The PRU size (FET+4, bits 35 through 18) and device type (FET+1, bits 59 through 48) are returned to the FET as described under standard multifile set processing.

To avoid a fatal error, an HDR1 label must be present in the extended label buffer (refer to extended label processing under the OPEN macro for a description of the extended label buffer).

A multifile set is positioned to read an existing file if the file sequence number in the buffer HDR1 label is not display code 9999. The following requirements also apply.

- The tape is positioned to the first member of the multifile set whose HDR1 label matches that in the label buffer. Binary zero fields are not compared. If the fields do not match, the search continues.

- If the explicit or implicit position number is greater than that of the last member file (that is, the matching HDR1 was not found), POSMF is terminated, returning an end-of-status (21 octal in FET+0, bits 13 through 9).

- If the desired file is located, all labels from HDR1 through the next tape mark are transferred to the label buffer, as space permits. Verification of additional labels is the user's responsibility.

A multifile set is positioned to write a new file if the file sequence number in the buffer HDR1 label is display code 9999. Further requirements or restrictions are as follows.

- The first file of a multifile set is created if all of the following are true (refer to examples 1 and 2):

    1. The tape is positioned to the first file.

    2. The last operation was not a write.

    3. The set identification field in the existing HDR1 label on the tape is all blanks (the set identification field in the buffer HDR1 label must be nonblank if the set is to be extended in the future).

- The multifile set is positioned to extend the set in all other cases. The tape is positioned after the last file in the multifile set.

- The system writes a new HDR1 label and all additional user labels to the tape with the following results.

    1. The file sequence number in the HDR1 label of the new file is set to display code 0001 for the first file or set to the last member sequence number plus 1 for extended files.

    2. The file section number in the HDR1 label is always set to display code 0001.

    3. With the above exceptions, the information in the extended label buffer is used to generate the HDR1 label on the tape. The default value is used for any buffer HDR1 field that is binary zero. If a numeric field contains a nonnumeric value, the job is aborted and the following message is issued to the dayfile.

        ILLEGAL XL BUFFER/FET LABEL FIELD, lfn AT nnn.

        The creation date field in the buffer HDR1 label is ignored; the current date is always used.

    4. All additional user file header labels (HDR2-9) and user header labels (UHL) in the label buffer are written to the tape. Nonapplicable labels are ignored.

Example 1:

Step 1. BLANK(VSN=TEST,D=HY,MT)

Creates a labeled tape with SETID defaulted to blanks.

Step 2. LABEL(TAPE,VSN=TEST,LB=KL,F=S,MT)

Causes tape to be assigned and positioned at file 1.

Step 3. LGO.

Executes program where POSMF on file TAPE with recall specified is performed with sequence number set to 999 (or 9999) and a nonblank SETID set in the FET (or xl buffer). This causes the first file of the multifile set to be created using the label fields specified in the FET (or xl buffer).

Subsequent POSMFs with sequence number set to 999 (or 9999) cause the file set to be extended since the SETID field on the tape is now nonblank.

Example 2:

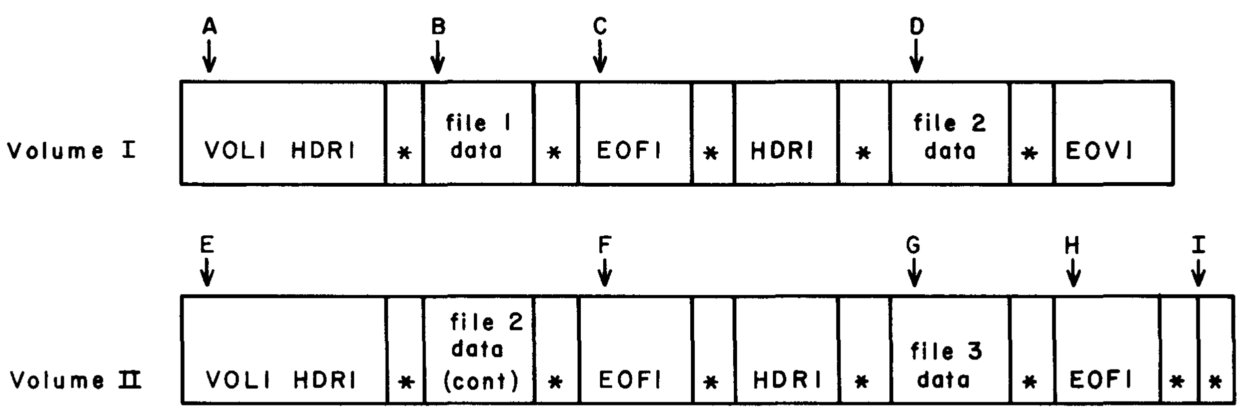
Repeat the above example except that a

LABEL(TAPE,VSN=TEST,SI=TESTAA,F=S,MT,W)

is performed at step 2. The POSMF in the third step extends the file set since the SETID is nonblank on the tape.

Example 3:

The structure of multifile labels is outlined in appendix G, volume 1. Given the following example, note the operation.



- To create this file, the following sequence is used.

| | | |
|---|---|---|
| 1. | OPEN or LABEL | With section number = 1, sequence number not specified (default is 1) |
| 2. | Write data | (File set 1) |
| 3. | POSMF | With sequence number set to 999 |
| 4. | Write data | (Goes over end of reel) (file set 2) |
| 5. | POSMF | With sequence number set to 999 |
| 6. | Write data | (File set 3) |
| 7. | REWIND | This causes the trailer label at H to be written. The tape is then rewound to the beginning of Volume II (position E) and then the file is positioned forward to G. The tape is not necessarily at load point following a rewind of a multifile set member, rather at the start of the multifile set member. The only means, at this point, to position to the beginning of Volume I is to issue a POSMF with sequence number 1. |

- If located at the end of file set member 2 (F), a rewind positions the tape to the beginning of Volume I since file set member 2 begins on Volume I, and then positions Volume I to D.

- By writing over file set member 2, file set member 3 is destroyed.

- A POSMF 999 followed by a WRITE creates a file set member at I.

- To copy all three file set members, the following technique may be used.

  OPEN with display code 001 (or 0001) sequence number and all other label fields binary zero.

  READ to EOI. (C)

  POSMF with display code 002 (or 0002) sequence number and all other label fields binary zero. This positions to next file set member at (D).

  READ to EOI. (F)

  POSMF with display code 003 (or 0003) sequence number and all other label fields binary zero.

  READ to EOI. (H)

The SI (M) parameter must be present for multifile label positioning using control statements. If the QN (P) parameter is present, the multifile set is positioned to the file set member that matches the specified sequence number. If QN is not specified and the FI (L) parameter is present, the multifile set is positioned to the file set member that matches the file identifier specified. If both QN and FI are specified, a match must occur on both sequence number and file identifier. If neither QN nor FI is specified, an OPEN is done instead of a POSMF.

To extend a multifile set, QN must be set to 9999.

If the SI parameter is not specified, then file positioning is not done. The R and W parameters on the LABEL statement are ignored if SI is specified. The exception is if the W parameter is specified and QN $\leq$ 1, and it is the first OPEN on the file, then an OPEN/WRITE is performed.

Although the sequence number field in the HDR1 label is four characters in length, only the rightmost three characters are used to differentiate between 999 and another valid sequence number. This is because on an open (POSMF) the FET field for sequence is only three characters. Therefore, if extended labels are not being used, a limit of 998 file set members per file set is enforced.

## EVICT (114)

The EVICT macro is similar to the RETURN macro in that it releases file space for the specified file. It differs from RETURN in that EVICT does not release the file attachment to the job.

Macro Format:

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | EVICT | addr, r |

addr — Address of the FET

r — If r is specified, control is not returned until the operation is complete.

The operation that EVICT performs depends on the file type. For permanent files, all file space except the first track is released, job attachment remains, and an EOI is written on the first sector of the first track. For all other mass storage file types, file space is released and job attachment remains. Files for which write lockout is set are returned to the system. An EVICT of a tape file performs the same functions as the UNLOAD macro.

## SKIPF (240)

SKIPF causes the file to be positioned n records forward from the current position. The operation terminates when the skip count is satisfied or when EOI is encountered on a mass storage file.

The status response is:

0021 — Last record skipped over was EOR

0031 — Last record skipped over was EOF

1031 — EOI encountered

Macro Format:

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | SKIPF | addr, n, r |

addr — Address of the FET

n — Number of records to skip; if n is omitted, 1 is assumed.

r — If r is specified, control is not returned until the operation is complete.

**SKIPFF (240)**

SKIPFF skips forward the specified number of files.

The status responses are:

0021 — Last record skipped over was EOR

0031 — Last record skipped over was EOF

1031 — EOI encountered

Macro Format:

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | SKIPFF | addr, n, r |

addr — Address of FET

n — Number of files to skip; if n is omitted, 1 is assumed.

r — If r is specified, control is returned only upon completion of operation.

### SKIPEI (240)

SKIPEI causes the file to be positioned at EOI. The skip count in RA+1 is set to $777777_8$ to indicate a skip to EOI. The status returned is 103x. On magnetic tapes where no EOI is defined, the operation stops at an EOF.

Macro Format:

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | SKIPEI | addr, r |

| | |
|---|---|
| addr | Address of the FET |
| r | If r is specified, control is not returned until the operation is complete. |

### SKIPB (640)

SKIPB causes the file to be backspaced n logical records. If BOI is encountered before the operation is complete, a rewind status (05x) is returned. If the file is positioned exactly at BOI, backspace status is returned (64x).

Macro Format:

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | SKIPB | addr, n, r |

| | |
|---|---|
| addr | Address of the FET |
| n | Number of records to skip backward. If n is not specified, 1 is assumed. If n=$777777_8$, file is rewound. |
| r | If r is specified, control is not returned until the operation is complete. |

### SKIPFB (640)

SKIPFB causes the specified file to be backspaced n files from the current position. If the skip count specified is $777777_8$, the file is rewound. If the operation positions the file to BOI (even if it was originally at BOI), a rewind status (05x) is returned. Otherwise a skip status (02x) is returned.

Macro Format:

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | SKIPFB | addr, n, r |

| | |
|---|---|
| addr | Address of the FET |
| n | Skip count; if n is omitted, 1 is assumed. |
| r | If r is specified, control is not returned until the operation is complete. |

## DATA TRANSFER MACROS

NOS provides a set of macros to aid the programmer in manipulating data that is to be processed after reading from or to be written on a file. To use the data manipulation macros with the input/output macros, the user must define input/output buffers and working buffers. The data manipulation macros enable the user to transfer data to/from the working buffer or from/to the input/output buffer without having to be concerned with the input/output buffer FET pointers. The macros and the associated common decks perform all activities involved with the circular pointers. Figure 2-3-6 illustrates a typical buffer arrangement:

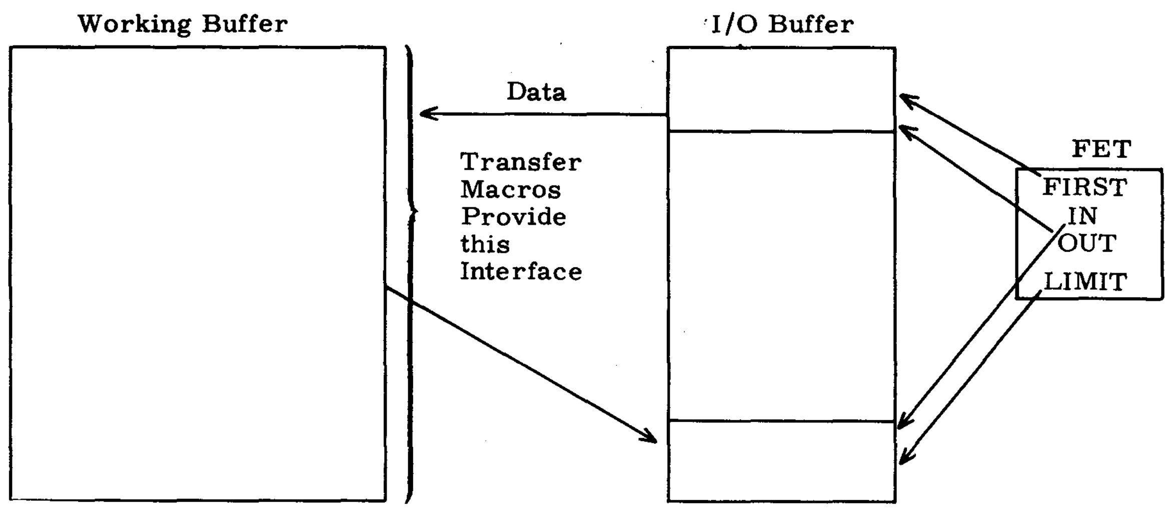



Figure 2-3-6. Data Transfer Buffer Arrangement

The working buffer is usually significantly smaller than the input/output buffer.

The user must issue the input/output functions to initiate reading or writing between the input/output buffer and the device. However, once the function is issued, the common decks called by the data transfer macros initiate the subsequent input/output requests required to maintain the data flow until the initial function is satisfied.

Macro read processors return the following information.

- (X1) = 0 if the transfer to the working buffer is complete.
- (X1) = -1 if an EOF is encountered when reading the file.
- (X1) = -2 if an EOI is encountered when reading the file.
- (X1) > 0 if an EOR is encountered on the file before the transfer to the working buffer is complete; this number is the address plus 1 in the working buffer of the last word transferred.

Data is not transferred after an EOR/EOF/EOI is encountered (for example, READEI or READNS). Depending on the read code, EOR and/or EOF may not be returned. B6 contains the address plus 1 in the input/output buffer of the last word of data transferred.

The following sample program rewinds and copies one file to another, modifying data as desired.

```
                 IDENT   MANIP
                 ABS
                 ENTRY   MANIP
                 .
                 .
                 .
        IBUFL    EQU     1001B        INPUT BUFFER LENGTH
        OBUFL    EQU     2001B        OUTPUT BUFFER LENGTH
        WBUFL    EQU     100B         WORKING BUFFER LENGTH

        I        FILEB   IBUF,IBUFL
        O        FILEB   OBUF,OBUFL

        MANIP    SB1     1
                 REWIND O
                 REWIND I
        TAG1     READ    I            INITIATE READ OF LOGICAL RECORD
                 RECALL O
        TAG2     READW   I,WBUF,WBUFL
                 ZR      X1,TAG3      IF NOT EOR/EOF
                 NG      X1,TAG4      IF EOF OR EOI ENCOUNTERED
                 .                    END-OF-RECORD ENCOUNTERED
                 .                    PERFORM DATA MANIPULATION
                 .                    OF DATA IN WBUF
                 .
                 WRITEW O,WBUF,X1-WBUF  WRITE LAST PORTION OF
                 WRITER O                 RECORD READ
                 EQ      TAG1         LOOP TO INITIATE READ OF NEXT
                                      RECORD
        TAG3     .
                 .                    PERFORM DATA MANIPULATION
                 .                    OF DATA IN WBUF
                 .
                 .
                 WRITEW O,WBUF,WBUFL  LOOP TO TRANSFER MORE
                 EQ      TAG2           DATA TO WORKING BUFFER

        TAG4     WRITEF O             WRITE END-OF-FILE
                 ENDRUN               END PROGRAM

       **        BUFFERS.

        WBUF     EQU     *            WORKING BUFFER
        IBUF     EQU     WBUF+WBUFL   INPUT BUFFER
        OBUF     EQU     IBUF+IBUFL   OUTPUT BUFFER

                 END
```

In the previous example, if the RECALL function was not specified for the output file, the following steps could occur.

1. The user program issues the REWIND O function.

2. Since auto recall is not specified, the user program continues execution.

3. At some point, the user program issues WRITEW which moves data from WBUF to the input/output buffer (O) starting at the current IN pointer in the FET. The WRITEW macro advances the IN pointer to reflect the amount of data transferred into the buffer.

4. At this point, the system completes the REWIND function and accordingly updates the FET IN and OUT pointers to point to FIRST (empty buffer). This destroys the IN pointer updated in step 3. The data placed in the buffer is ignored.

5. On subsequent EOR operations (WRITER), the same situation could occur if the RECALL function was placed elsewhere.

```
WRITER O
  .
  .          must place RECALL O here.
  .
WRITEW O
  .
  .
  .
```

Write requests are not issued with auto recall specified because other operations (READ) can be performed before it is necessary to have the write operation completed.

In the previous example, if an EOR or EOF mark is not detected, it is not necessary to reissue READ requests to fill the input/output buffer because the macros and the associated common decks detect when the buffer threshold is reached. If this threshold is reached, a request is automatically issued to CIO. This occurs on the READ macro when the empty space in the buffer exceeds the threshold or on the WRITE macro when data in the buffer exceeds the threshold. The threshold used is half the buffer size. That is, if the buffer is more than half empty, a read request is issued; if it is more than half full, a write request is issued.

Assuming no EOR is encountered by the fifth time through the loop, the READW function issues another CIO request to transfer data from file I to buffer IBUF; since more than 256 ($400_8$) words are removed from IBUF, it is now more than half empty. On the ninth time through the loop, the WRITEW function issues a WRITE request to file O since OBUF is now more than half full.

For the data transfer macros, the common decks required for absolute assemblies, in addition to those specified with each macro, are:

- COMCCIO
- COMCSYS
- COMCWTW for write functions (except WRITEO)
- COMCRDW for read functions (except READO)

For relocatable assemblies, these decks are satisfied by default from the library SYSLIB.

In all of the macros described, the following parameter definitions apply.

| | |
|---|---|
| addr | Address of the FET |
| buf | Working buffer address |
| n | Working buffer word count |

## READC

The READC macro reads one coded line from the input/output buffer to the working buffer. Data is transferred until the end of the line (0000 in bits 11 through 0) is sensed or until n words are transferred.

Macro Format:

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | READC | addr, buf, n |

Common deck required: COMCRDC

## WRITEC

The WRITEC macro transfers a coded line image from the working buffer to the input/output buffer. Data is transferred until the end of the line (0000 in bits 11 through 0) is sensed.

Macro Format:

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | WRITEC | addr, buf |

Common deck required: COMCWTC

## READH

The READH macro reads a coded line with space fill from the input/output buffer to the working buffer. Data is transferred until the end of the line (0000 in bits 11 through 0) is sensed or until n words are transferred.

Macro Format:

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | READH | addr, buf, n |

Common deck required: COMCRDH

## WRITEH

The WRITEH macro writes a coded line, deleting all trailing spaces, from the working buffer to the input/output buffer.

Macro Format:

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | WRITEH | addr, buf, n |

Common deck required: COMCWTH

## READO

The READO macro reads one word from the input/output buffer to X6. (X1)=1 if an EOR is encountered.

Macro Format:

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | READO | addr |

Common deck required: COMCRDO

## WRITEO

The WRITEO macro writes one word from X6 to the input/output buffer.

Macro Format:

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | WRITEO | addr |

Common deck required: COMCWTO

## READS

The READS macro reads a line image to a character buffer. The words are unpacked and stored in the working buffer, right-justified, one character per word, until the end-of-line byte (0000) is detected. If the coded line terminates before n characters are stored, the working buffer is blank-filled.

Macro Format:

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | READS | addr, buf, n |

Common deck required: COMCRDS

## WRITES

The WRITES macro writes a line image from the character buffer. Characters are packed 10 characters per word. Trailing spaces are deleted before the characters are packed.

Macro Format:

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | WRITES | addr, buf, n |

Common deck required: COMCWTS

**READW**

The READW macro fills the working buffer from an input/output buffer. READW may transfer data to beyond the end of the working buffer. This could cause the program to abort if the last word address of the buffer is within four words of FL. The n parameter must be specified.

Macro Format:

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | READW | addr, buf, n |

Common deck required: COMCRDW

**WRITEW**

The WRITEW macro transfers data from the working buffer to the input/output buffer. WRITEW may transfer data from beyond the end of the working buffer. This could cause the program to abort if the last word address of the buffer is within four words of FL. The n parameter must be specified.

Macro Format:

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | WRITEW | addr, buf, n |

Common deck required: COMCWTW

## CDC CYBER RECORD MANAGER I/O

CDC CYBER Record Manager (CRM) consists of a group of routines providing input/output facilities common to several products. User programs written in several higher level languages (for example, COBOL and FORTRAN Extended) can communicate with the Record Manager through compiler language calls. COMPASS users communicate through Record Manager macros (refer to the CYBER Record Manager Reference Manual).

Features of CRM include the following.

- Consistent error checking
- Accommodation for various label checking
- Support of several file organizations

CDC CYBER Record Manager supports the following file organizations.

- Sequential files in physical order
- Word addressable files on mass storage with continuous nonblocked data
- Indexed sequential files in which records are physically and logically ordered by symbolic keys
- Direct access files containing records in fixed length blocks; record location is determined by hashing a key to identify a block
- Actual key files in which each record is stored in a location specified by the key associated with that record

For a complete description of COMPASS macros, file organizations, and record and block formats supported by CRM, refer to the CDC CYBER Record Manager Reference Manual.

# LOCAL FILE MANAGER 4

Local file manager (LFM) performs requests associated with the control of a user's files. The user can issue requests to LFM and have control returned if certain error conditions occur. To do this, the error processor bit (ep) must be specified in word 1 of the FET. The following error codes are returned in the error code field of the FET word 0, bits 17 through 10.

| Error Codes | Description | Error Codes | Description |
|---|---|---|---|
| 1 | File not found | 14 | Illegal ID code |
| 2 | File name error | 15 | Resource executive (RESEX) detected an error |
| 3 | Illegal file type | 16 | I/O sequence error |
| 4 | File empty | 17 | Output file limit |
| 5 | MAGNET not active | 20 | Local file limit |
| 6 | Duplicate library file name | 21 | No mass storage available |
| 7 | Illegal equipment | 22 | Illegal file mode |
| 10 | Equipment not available | 23 | Illegal change in file/origin type |
| 11 | Duplicate file name | 24 | FET too short |
| 12 | Illegal user access | 25 | GETFNT table too large |
| 13 | Illegal user number | | |

The following error causes the job to be aborted regardless of the presence of the user error processing bit in the FET.

LFM ILLEGAL REQUEST.

This message is issued if any of the following situations occur.

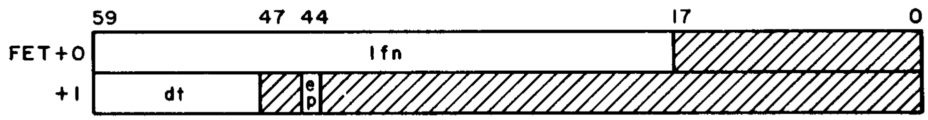
- An LFM function is issued without the auto recall bit set. All LFM calls must be specified with recall.
- An LFM function detected is not recognized as a legal function.

The format of the call to LFM is:

| | 59 | 40 | | 35 | 23 | 17 | 0 |
|---|---|---|---|---|---|---|---|
| RA+1 | LFM | | 1 | | code | id | addr |

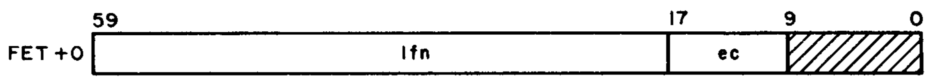
| | |
|---|---|
| code | Function code |
| id | File id number (refer to SETID, function code 017) |
| addr | Address of the FET |

LFM uses the following information from the FET.




| | |
|---|---|
| lfn | File name |
| dt | Device type |
| ep | Error processing bit |

After the request is completed, the first word of the FET contains the following information.




| | |
|---|---|
| ec | Error code |

The common decks required in absolute assemblies for the macros processed by LFM are:

- COMCLFM
- COMCSYS

For relocatable assemblies, these decks are satisfied by default from the library SYSLIB.

The LFM functions are described on the following pages.

## RENAME (000)

The RENAME function enables the user to change the name of a file currently attached to the job to the name specified in word 6 of the FET. This does not change the names of files in the permanent file system.

If a file by the new file name already exists, it is returned to the system. However, there are certain conditions under which the file type of the old file is changed to that of the returned file.

- If the old file is a local mass storage file and the returned file is a print, punch, or primary type file, the file type of the old file is changed to that of the returned file.
- If the old file is a local mass storage file and the returned file is not a print, punch, or primary type file, the old file is renamed but its file type is not changed.
- If the old file is not a local file or does not reside on mass storage, an

  ILLEGAL FILE TYPE.

  error message is issued.

Macro Format:

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | RENAME | lfn, addr |

lfn	Address of the FET (FET+0 contains the old file name)

addr	Address of new file name:

If addr = X1, X1 contains the new file name.

If addr is not specified, word 6 of the FET contains the new file name.

## ASSIGN (001)

The ASSIGN function enables the user to access a library file. If a file by the requested name is already local to the user's job, no action is taken. If the requested library file is not found, the following message is issued.

FILE NOT FOUND.

If the error processing bit is set, the message is not issued and error code 1 is returned.

The user must be validated to access library files or the following message is issued.

ILLEGAL USER ACCESS.

Macro Format:

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | ASSIGN | lfn |

lfn	Address of the FET

## COMMON (002)

The COMMON macro changes the file type of the specified file to library. The file must be a locked local file residing on a mass storage device. If it is not, the following message is issued.

ILLEGAL FILE TYPE.

If a file of the same name already exists as a library file, the following message is issued.

DUPLICATE COMMON FILE NAME.

If a file by the requested name is not found, the following message is issued.

FILE NOT FOUND.

If the equipment the file is assigned to is not mass storage, the following message is issued.

ILLEGAL EQUIPMENT.

The user must be validated to access library files or the following message is issued.

ILLEGAL USER ACCESS.

The file is no longer assigned to the user's job if the operation is successful.

Macro Format: †

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | COMMON | lfn |

lfn    Address of the FET of the file to be entered as a library file

† This macro is not available in SYSTEXT. The user must call common deck COMCCMD (refer to appendix A).

## RELEASE (004, 005, 006, 007, 016, 030)†

This function enables the user to release files to any of the output queues for processing before job termination.

Any of the following file types can be released.

- Local files
- Print files
- Punch files

If any other type is released or the file is not a mass storage file, the following message is issued.

ILLEGAL FILE TYPE.

If the file is unused, the following message is issued.

FILE EMPTY.

If an attempt is made to change the file type or origin type of a deferred routed file (refer to section 8), the following message is issued.

ILLEGAL CHANGE IN FILE/ORIGIN TYPE.

The file must be routed to a scratch file (DC=SC) before it can be released to the desired file and origin type.

If the number of files released to the output queue by the job has exceeded the limit for which the user is validated, the following message is issued and the job is aborted.

OUTPUT FILE LIMIT.

If the file is an execute only file, the following message is issued.

ILLEGAL FILE MODE.

Macro Format:

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | RELEASE | lfn, type, ot, un |

† The user should use the ROUTE macro instead of RELEASE (refer to section 8).

| | | | |
|---|---|---|---|
| lfn | Address of the FET | | |
| type | One of the following: | | |
| | PRINT | Release file to the PRINT queue (function 004) | |
| | PUNCH | Release file to the PUNCH O26 queue (function 005) | |
| | PUNCH9 | Release file to PUNCH O29 queue (function 030) | |
| | PUNCHB | Release file to the PUNCHB queue (function 006) | |
| | P8 | Release file to the P8 queue (function 007) | |
| ot | Specifies disposition of output: | | |
| | BC | Release file to the local batch queue (function 016) | |
| | EI | Release file to the remote batch queue (function 016) | |
| un | Address containing the user number of the remote batch user to which the file is to be disposed (ignored if ot is BC). This parameter is valid only if the user is allowed deferred batch jobs. Also, un must match the user number of the user performing the RELEASE on all character positions except those containing an asterisk (*). | | |

If the ot and un parameters are not specified, a remote batch job disposes the file to a remote batch terminal from which the job is submitted and all other origin types dispose the file to the central site output device. If ot is BC, the un parameter is ignored and the file is disposed as usual to the central site device.

The user may release coded punch files in either O26 or O29 mode, independent of the job's initial keypunch mode (refer to appendix F, volume 1).

## LOCK (010)

This function enables the user to prevent writing on a file by setting the write lockout bit for the file. The file specified must be a local file; if it is not, the following message is issued.

ILLEGAL FILE TYPE.

Macro Format:

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | LOCK | lfn |

| | |
|---|---|
| lfn | Address of the FET |

## UNLOCK (011)

The UNLOCK function clears the write lockout bit for the specified file. The file must be a local file; if it is not, the following message is issued.

ILLEGAL FILE TYPE.

Macro Format:

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | UNLOCK | lfn |

lfn Address of the FET

## STATUS (012)

The STATUS macro determines if a file exists. Zero is returned in bits 11 through 1 of the FET status word if the file is not found. A nonzero quantity is returned in bits 11 through 1 if the file is found. To determine the current position and status of a file, use STATUS (013).

Macro Format:

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | STATUS | lfn |

lfn Address of the FET

## STATUS (013)

This function returns the current position and status of a mass storage or magnetic tape file. This function can be used to determine the current position of the file, the type of file, and the device on which it resides. If the file is a magnetic tape file, information is returned to FET+8 as described for the LABEL macro. † If the file does not exist, the following message is issued.

FILE NOT FOUND.

The device type is returned in word 1 of the FET; if the device is a nonmass storage device, the upper bit is set. Refer to appendix E for a list of legal device types and equipment codes.

† This feature of the STATUS macro is subject to change in future versions of NOS.

In general, the device type returned for a STATUS request is the same as that returned for a CIO OPEN request. For SI, S, and L formatted tapes, however, the device type returned by OPEN differs from the MT or NT device type returned by status. For a description of this special device type format, refer to the CIO OPEN macro, section 3.

The FNT entry of the file is returned in word 5 of the FET.

The FST entry of the file is returned in word 6 of the FET. For NOS, the previous function code is not retained in the FST and the following status information from byte 4 of the FST is returned to the FET.

| Bit(s) | Description | |
|---|---|---|
| 0 | Set if the file is not busy | |
| 1 | Set if the last operation was a write | |
| 2-3 | If the last operation was a read: | |
| | 0 | Incomplete (buffer full) |
| | 1 | EOR encountered |
| | 2 | EOF encountered |
| | 3 | EOI encountered |
| | If the last operation was a file positioning or write operation: | |
| | 0 | Incomplete (no EOR/EOF or buffer exhausted) |
| | 1 | Complete |
| 4-5 | Not used | |
| 6 | Set if the file is written on since attachment or creation | |
| 7 | Set if the file is written on since last opened | |
| 8 | Set if the file is opened | |
| 9-10 | Not used | |
| 11 | Set if labeled tape | |

Macro Format:

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | STATUS | lfn, P, T† |

lfn Address of the FET

P If P is specified, the current position is returned. If this parameter is omitted, LFM function 12 is executed.

T† If T is specified (in addition to P) and lfn refers to a magnetic tape file, FET+8 receives information as described in FET+8 of the FET used by the LABEL macro (although the block size is returned in CM words, rather than frames). If lfn is not a magnetic tape file, the request for additional information is ignored. Refer to the LABEL macro description in this section for further details on the contents of the FET+8 fields.

If the FET is not at least nine words in length, the following message is issued.

FET TOO SHORT.

† The T parameter is subject to change in future versions of NOS.

## REQUEST (014)

The REQUEST function requests operator assignment of equipment to the file. If the file is previously assigned, the function is ignored. If dt≠0 in word 1 of the FET, the device assigned must be this type. If dt=MS, a mass storage device must be assigned. If dt=MT, the following options may be present in word 1 of the FET.

| | 59 | 47 | 41 | 39 | 35 | 23 0 |
|---|---|---|---|---|---|---|
| FET+1 | dt | ////// | f | d | cc | ////// |

f Format

0 = system

1 = external BCD, line image, cc = 136D

2 = external BCD, blocked, cc = 150D

d Density

0 = no change

1 = HI (556 bpi)

2 = LO (200 bpi)

3 = HY (800 bpi)

cc Character count/record (BCD format)

These options apply only to 7-track tapes; the user should not request 9-track tape with this function. It is recommended that the programmer use the LABEL macro rather than the REQUEST macro for tape request.

The user must be validated to access the assigned equipment; if he is not, the following message is issued.

ILLEGAL USER ACCESS.

Macro Format:

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | REQUEST | lfn |

lfn Address of the FET

If REQUEST is to be used for checkpoint dumps, FET+7 specifies the checkpoint mode.

| | 59 | 53 | 0 |
|---|---|---|---|
| FET+7 | cm | | |

| | |
|---|---|
| cm | Checkpoint† mode (left-justified); indicates that lfn is to be used as a checkpoint file: |

| | |
|---|---|
| $75_8$ | Each time a checkpoint dump is taken, the new information is written at the BOI of lfn. |
| $76_8$ | Each time a checkpoint dump is taken, the new information is written at the previous EOI of lfn. |

The user can alternately write dumps on two checkpoint files by issuing two REQUESTs with cm=$75_8$. If cm=$76_8$ for alternate files or if more than two checkpoint files are specified, the job is aborted and the following message is issued to the user's dayfile.

CHECKPOINT FILE ERROR.

## REQUEST (015)

This function assigns a file to the specified equipment. If the file already exists, the following message is issued.

DUPLICATE FILE NAME.

The FET parameters specified for this request are the same as for function 014. The user may also specify a numeric value for the equipment ordinal. This is the EST ordinal of the device.

The job must be of system origin or the user must be validated for system origin privileges or the following message is issued.

ILLEGAL USER ACCESS.

If the user has this validation, he may use function 015 for checkpoint dumps. The checkpoint mode is specified in FET+7 in the same format as that for function 014.

† For further information, refer to the description of the CHECKPT macro, section 10.

Macro Format:

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | REQUEST | lfn, U |

lfn Address of the FET

U If U is specified, the device type in word 1 of the FET is the device to which the file is assigned. If the device type in word 1 of the FET is numeric, this is the EST ordinal to which the file is assigned. Absence of this parameter causes LFM function 014 to be issued.

For tape requests, this macro applies only to 7-track tapes; the user should not request a 9-track tape with this function. It is recommended that the programmer use the LABEL macro instead of the REQUEST macro for tape requests.

## SETID (017) †

This function sets the identifier code for a file. This enables the user to direct an output file to a particular device. The output queue processors assign files to devices only if the ID codes are the same. If the file does not exist, it is created by this function. The file must be: input (INFT), print (PRFT), local (LOFT) or punch (PHFT), or the following message is issued.

ILLEGAL FILE TYPE.

Macro Format:

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | SETID | lfn, n |

lfn Address of the FET

n Identifier code; $0 \leq n \leq 67_8$

† The user should use the ROUTE macro instead of SETID (refer to section 8).

## ASSIGN (020)

This function is used to access a library file. If the file is accessed from a system or library file, the return status code equals 0.

The random address of the directory is stored in word 6 of the FET.

The address bias for the directory is stored in word 7 of the FET.

This function enables the user to access user libraries that exist either on files attached to the job or the system. The local files are searched first. Because of the structure of the system file, the address bias for the directory must be specified. For example, if the directory for SYSLIB is specified at random address 2000 (FET+6) on the system file, the bias for all entries in this directory is 1777 (FET+7). This is the address to be added to the random addresses of all routines in this directory to access the routines from the system library.

If the file specified is a system procedure file, the sign bit in FET+7 is set. If the file specified is not a system file, FET+6 and FET+7 are returned as zero. If the file specified is a relocatable file, FET+6 and FET+7 are unchanged.

Macro Format:

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | ASSIGN | lfn, L |

lfn — Address of the FET

L — If L is specified, the file is assigned from the system file. The absence of this parameter causes LFM function 001 to be issued.

## ENCSF (022)

This function replaces the control statement file. If the file is not defined, the control statement file is cleared.

If the file specified is empty, the following message is issued.

FILE EMPTY.

Macro Format:†

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | ENCSF | addr |

addr — Address of the FET for the file that is to replace the current control statement file

## PSCSF (023)

This function gives the user the ability to control the execution of the job control statements by positioning the next statement to be executed.

Macro Format:†

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | PSCSF | addr |

addr — Address of the FET for the request

Word 6 of the FET contains the parameters in the following format:

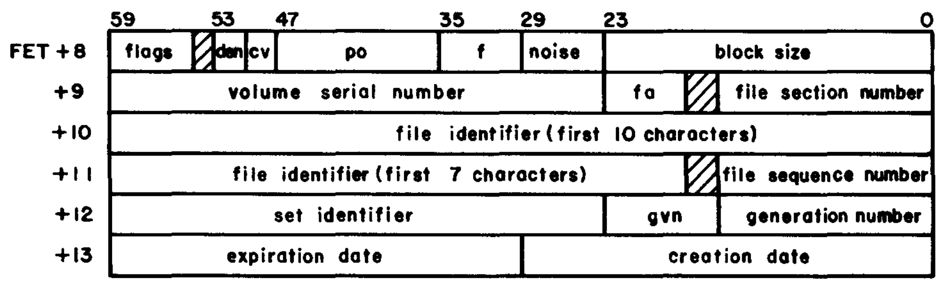

cs — Statement count

wc — Word count from the beginning of the file; if the word count specified exceeds the length of the file, the file is positioned at the beginning.

† This macro is not available in SYSTEXT. The user must call the common deck COMCMAC to use this macro (refer to appendix A).

## LABEL (024)

The LABEL macro assigns a file to magnetic tape and processes labeled and unlabeled tapes. LABEL uses the information in FET+8 through FET+13 to create new and access existing 7- or 9-track tape files. The FET information is defined as follows:




| Parameter | Word | Position | Description |
|---|---|---|---|
| Read/write (rw) | 8 | 59 | Specifies the type of label processing to be performed. This bit is set if an OPEN/WRITE is to be performed; it is not set if an OPEN/ALTER is to be performed. |
| Label (lb) | 8 | 58 | If this bit is set, the tape is labeled or is to be labeled. If the bit is not set, the tape is unlabeled. |
| Nonstandard (ns) | 8 | 57 | If this bit is set, the tape has or is to have nonstandard labels. If the bit is not set and bit 58 is set, the file is or is to be an ANSI-labeled tape. |
| Tracks (tr) | 8 | 56 | If this bit is set, a 9-track tape is to be used. If the bit is not set, a 7-track tape is to be used. |
| Density (den) | 8 | 53-51 | Tape density:<br>0 Installation default<br>1 556 bpi (7-track)<br>2 200 bpi (7-track)<br>3 800 bpi (7-track) or 800 cpi (9-track)<br>4 1600 cpi (9-track) |

| Parameter | Word | Position | Description |
|---|---|---|---|
| Conversion mode (cv) | 8 | 50-48 | Conversion mode for 9-track tapes:<br>0 Installation default<br>1 ASCII/USASII conversion<br>2 EBCDIC conversion |
| Processing option (po) | 8 | 47-36 | Processing options:† |

| Bit Set | Description |
|---|---|
| 36 | Abort on an irrecoverable read or write parity error even if the error processing bit is set. |
| 37 | Do not abort on an irrecoverable read or write parity error regardless of the error processing bit. |
| 38 | Inhibit error processing. |
| 39 | If the tape is mounted with write ring in, job processing is suspended until the operator remounts the tape correctly. |
| 40 | If the tape is mounted with the write ring out, job processing is suspended until the operator remounts the tape correctly. |
| 41 | Do not unload tape at end of usage. |
| 42 | Directs the system to write system noise blocks when performing write error recovery. This option is ignored for 1600-cpi tapes and should not be used for tapes which are to be interchanged with other systems. |
| 44-43 | Not used. |

† If neither bit 36, 37, nor 38 is set, abort on read or write parity error only if the error processing bit is not set. For further information about these processing options, refer to the equipment/file assignment control statements in section 10, volume 1.

| Parameter | Word | Position | Description |
|---|---|---|---|
| Processing option (po) | 8 | 47-36 | **Processing options:** |

| Bit Set | Description |
|---|---|
| 47-45 | These bits define the end-of-tape/end-of-reel conditions and are defined as follows: |

| Bit 47 | Bit 46 | Bits 45 | Option† |
|---|---|---|---|
| 0 | 0 | 1 | 3 |
| 0 | 1 | 0 | 2 |
| 1 | 0 | 0 | 1 |
| 0 | 0 | 0 | Default option selected according to format |

All other combinations are illegal.

| Parameter | Word | Position | Description |
|---|---|---|---|
| Data format (f) | 8 | 35-30 | Data format (refer to section 10, volume 1): |

| | |
|---|---|
| 0 | I |
| 1 | SI |
| 2 | X |
| 3 | S |
| 4 | L |
| 5 | E |
| 6 | B |
| 7 | F |

| Parameter | Word | Position | Description |
|---|---|---|---|
| Noise | 8 | 29-24 | Noise size in frames; any block containing fewer than the specified number of frames is considered noise and is discarded by the system (refer to Data Formats in section 10, volume 1). A noise specification of zero causes the default noise size to be used. |
| Block size (PRU size) | 8 | 23-0 | Maximum block size in frames (refer to Data Formats in section 10, volume 1). |

† For further information, refer to options 1 through 3 described in End-of-Tape/End-of-Reel Conditions, section 10, volume 1.

| Parameter | Word | Position | Description |
|---|---|---|---|
| Volume serial number | 9 | 59-24 | 1 to 6 display code characters that uniquely identify a reel of tape (refer to appendix G, volume 1, VOL1 label, and VSN control statement in section 10, volume 1). |
| File accessibility (fa) | 9 | 23-18 | 1 display code character indicating who may access the tape file. † |
| File section number | 9 | 14-0 | 15-bit binary file section number. † |
| File identifier | 10 | 59-0 | File identifier (first 10 display code characters, left-justified with binary zero or blank fill). † |
| File identifier | 11 | 59-18 | File identifier (last seven display code characters, left-justified with binary zero or blank fill). † |
| File sequence number | 11 | 14-0 | 15-bit binary file sequence number. † |
| Set identifier | 12 | 59-24 | 6 display code characters specifying the multifile set identifier.† |
| Generation version number (gvn) | 12 | 23-15 | 9-bit binary generation version number. † |
| Generation number | 12 | 14-0 | 15-bit binary generation number. † |
| Expiration date | 13 | 59-30 | 5 display code characters specifying the expiration date. † |
| Creation date | 13 | 29-0 | Creation date (2-digit numeric display code value for the year followed by a 3-digit numeric display code value for the day within the year). † |

The specified file is assigned to tape automatically if the volume serial number is specified either in FET+9 or via VSN control statement. If the vsn in FET+9 is zero and no VSN control statement for the file was included, the system requests the operator to assign a unit.

For ANSI-labeled tapes, FET+9 through FET+13 contain the values LABEL uses to process HDR1 labels. †

Macro Format:

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | LABEL | addr |

addr Address of the FET

† Refer to appendix G, volume 1, HDR1 label.

If the specified file already exists, the following action is taken.

- If the file is assigned to a device other than a tape unit, no assignment is made and job processing continues. If the user wishes to assign the file to tape, he should return the existing file to the system before issuing the LABEL request.

- If the file resides on magnetic tape and FET+8, bit 59, is not set, the system verifies the HDR1 label by issuing an OPEN/ALTER request.

- If the file resides on magnetic tape and FET+8, bit 59, is set, the system checks if the volume serial number in VOL1 matches that specified by the user in FET+9 or on a VSN control statement. If the vsns do not match, the job is aborted. If the vsns match, the system writes an HDR1 label by issuing an OPEN/WRITE request.

The following error messages may be issued to the user's dayfile in response to a LABEL request.

| | |
|---|---|
| EQUIPMENT NOT AVAILABLE. | The equipment to which the specified file was to be assigned is not available. |
| RESEX DETECTED ERROR. | The resource executive (RESEX) detected an error. |
| MT/NT CONFLICT. | The device type specified in FET+1 conflicts with the track type specified in FET+8, bit 56. A device of MT implies 7-track tape and NT implies 9-track tape. If dt=MT and bit 56 is set or if dt=NT and bit 56 is not set, this message is issued. |

If the file is to be used for checkpoint dumps and the dumps are to be written on labeled tape, the checkpoint mode can be specified in FET+7, bits 59 through 56. For further information about checkpoint dumps, refer to the REQUEST macro (function 014).

## GETFNT (025)

With the GETFNT macro, the user can generate a table of FNT/FST entries for his local files.

For mass storage files, bytes 2 and 3 of the FST (bits 35 through 12) can be modified with either a random index (converted from current track and sector), or the file length (number of sectors). For tape files, the FST entry can be modified with MT in byte 1 (bits 47 through 36) and the block number in bytes 2 and 3.

Macro Format:†

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | GETFNT | addr |

addr Address of the FET

†This macro is not available in SYSTEXT. The user must call common deck COMCMAC (refer to appendix A).

GETFNT obtains its input parameters from FET+ 8, in the following format.

| | 59 | 47 | 41 | 23 | 20 | 17 | 0 |
|---|---|---|---|---|---|---|---|
| FET + 8 | nf | | sb | | cb | ta | |

nf — Maximum number of 2-word FNT/FST entries to return to table. Table size must be at least (nf*2 + 1). Default is $200_8$ entries.

sb — File type selection bits. A bit set implies the corresponding file type is selected for entry into the table. Bit positions and corresponding file types are as follows:

| Bit | File Type | Description |
|---|---|---|
| 37 | LOFT | Local |
| 36 | SYFT | System |
| 35 | FAFT | Fast attach file |
| 34 | PMFT | Direct access permanent file |
| 33 | PTFT | Primary terminal |
| 32 | LIFT | Library |
| 31 | -- | Reserved |
| 30 | -- | Reserved |
| 29 | -- | Reserved |
| 28 | TEFT | Timed/event rollout |
| 27 | PHFT | Punch |
| 26 | PRFT | Print |
| 25 | ROFT | Rollout |
| 24 | INFT | Input |
| 23 | -- | Reserved |

Default (all bits set to zero) is selection of all file types.

cb — Control bits:

| Bit | Significance |
|---|---|
| 18 | If bit 18 is set, no modifications are made on the FST entries. If bit 18 is not set, pertinent modifications are made (such as the block number for tape files or that specified by bit 19 for mass storage files). |
| 19 | If bit 19 is set, mass storage file FST entries are modified with file lengths in bytes 2 and 3. If bit 19 is not set, the FST entry is modified with the random index in bytes 2 and 3. Bit 18 must be zero or bit 19 is ignored. |
| 20 | If bit 20 is not set (zero), checkpoint file FNT entries are returned in FET+9, instead of the table (FET length must be at least 13 words). If bit 20 is set, checkpoint file FNT/FST entries are returned to the table (FET length must be at least 9 words). |

| | |
|---|---|
| ta | Address of table. |

Upon return from GETFNT, the FET is as follows:

| | 59 ... 0 |
|---|---|
| FET + 8 | ia |
| + 9 | cfnt |
| +10 | cfst |
| +11 | acfnt |
| +12 | acfst |

| | |
|---|---|
| ia | ia is less than zero if an error is encountered (bit 18 of FET+8 not set) |
| cfnt | FNT entry of the checkpoint file; cfnt is zero if there is no checkpoint file; less than zero if more than two checkpoint files encountered (bit 20 of FET+ 8 not set) |
| cfst | FST entry of checkpoint file |
| acfnt | FNT entry of the alternate checkpoint file; acfnt is zero if there is no alternate checkpoint file |
| acfst | FST entry of the alternate checkpoint file |

## PRIMARY (031)

The PRIMARY macro enables the user to create or change a primary file (refer to the Time-Sharing User's Reference Manual for a description of primary files). The current primary file (if any) is returned and the local mass storage file specified in FET+0 is made the primary file. If the specified file does not exist, an empty primary file is created.

Macro Format: †

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | PRIMARY | addr |

| | |
|---|---|
| addr | Address of the FET for the new primary file |

† This macro is not available in SYSTEXT. The user must call common deck COMCMAC (refer to appendix A).

## FILINFO (032)

The FILINFO macro returns information about a file to a specified parameter block.

Macro Format:

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | FILINFO | addr |

addr	Address of a 5-word parameter block

Before the FILINFO macro is issued, the first word of the parameter block must contain the following.

| | 59–18 | 17–12 | 11–1 | 0 |
|---|---|---|---|---|
| addr | logical file name | len | 0 | 1 |

len	Length of parameter block (at least five words)

The parameter block returned has the following format.

| | 59–48 | 47–36 | 35–30 | 29–18 | 17–12 | 11–6 | 5–1 | 0 |
|---|---|---|---|---|---|---|---|---|
| addr+0 | logical file name (lfn) | | | | len | 0 | | c |
| +1 | dt | status | | | | | ft | |
| +2 | equipment no. | 0 | | | | | | |
| +3 | file length in sectors | | 0 | current random address | | | 0 | |
| +4 | reserved | | | | | | | |

c	Completion bit (set when operation is complete)

dt	The 12-bit display code of the type of device on which the file resides (refer to appendix E) status Status bits:

| Bits | Description |
|---|---|
| 47-24 | Reserved |
| 23 | File at EOI |
| 22 | File at EOF |
| 21 | File at BOI |
| 20 | Labeled tape file |
| 19 | 9-track tape file |
| 18 | 7-track tape file |
| 17 | Reserved |
| 16 | File assigned to time-sharing terminal |
| 15 | File on mass storage |
| 14-13 | Reserved |
| 12 | File assigned in execute mode † |
| 11 | File assigned in read/append mode † |
| 10 | File assigned in read/modify mode† |
| 9 | File assigned in modify mode † |
| 8 | File assigned in append mode † |
| 7 | File assigned in write mode † |
| 6 | File assigned in read mode † |

ft File type in one of the following octal values.

| File Type | Value |
|---|---|
| Local | 0 |
| Input | 1 |
| Print | 2 |
| Punch | 3 |
| Direct access | 4 |
| (Reserved) | 5 |
| Primary | 6 |
| Library | 7 |
| Others | 77 |

File position is returned for mass storage files only. EOF status is returned if the last operation was a read, and EOF was encountered.

† Refer to Permission Modes, File Categories in section 5.

For mass storage files, bits 6 through 12 of the status field are set, depending upon the file permission mode in the file name table. For other files, only the write lockout bit in the file status table is checked. If the write lockout bit is set, read-only permission is assumed. If the write lockout bit is not set, read and write permission is set. Bits 8 through 12 apply only to mass storage files. If the following permission mode is set in the FNT, the indicated bits are set in the status field.

| FNT Mode | Status Bits Set |
|---|---|
| Read | 6, 12 |
| Write | 6, 7, 8, 9, 12 |
| Append | 6, 8, 12 |
| Modify | 6, 8, 9, 12 |
| Read/modify | 6, 8, 10, 12 |
| Read/append | 6, 11, 12 |
| Execute | 12 |

For example, if read mode is set in the FNT, the user can read (bit 6 set) or execute (bit 12 set) the file.

Word 4 (addr+4) of the parameter block is currently not used; however, it is reserved for future expansion of the FILINFO macro.

If the specified file is not local to the user's job, the following message is issued.

FILE NOT FOUND.

If any of the parameter block is beyond the user's field length, the following message is issued.

ADDRESS OUT OF RANGE.

If the length specified for the parameter block is not at least five words, the following message is issued.

FET TOO SHORT.

If this function is issued as an RA+1 request without using the FILINFO macro, the completion bit must not be set to 0. If it is, the following message is issued.

PARAMETER BLOCK BUSY.

If the file is busy (cannot be interlocked), the following message is issued.

FILE BUSY.

# PERMANENT FILE MANAGER 5

Permanent file manager (PFM) processes all permanent file requests. The user can issue requests to PFM and have control returned if certain error conditions occur. To do this, the error processing bit (ep) must be specified in word 1 of the FET. The following error codes are returned in the error code field (bits 17 through 10) in word 0 of the FET.

| Error Codes | Description |
|---|---|
| 1 | The specified direct access file is attached in the opposite mode. |
| 2 | One of the following:<br>• The specified permanent file could not be found.<br>• The specified user number could not be found.<br>• The user is not allowed to access the specified file.<br>• The user issued an indirect access file command on a direct access file.<br>• The user issued a direct access file command on an indirect access file.<br>If this error occurs in response to the SAVE macro, the specified local file is not attached to the control point, is a direct access file, or is an execute-only file. |
| 3 | The file specified on a SAVE macro contains no data. |
| 4 | The file to be saved is not on mass storage; the first track of the file is not recognizable. |
| 5 | The user has already saved or defined a file with the name specified. |
| 6 | The user attempted to define a file that is not a local file. |
| 7 | File name contains illegal characters. |
| 10 | The user is not validated to create direct access or indirect access files or to access auxiliary devices. |
| 11 | The device type (r parameter in macro calls) specified on a request for an auxiliary device cannot be recognized or does not exist in the system.<br>If the auxiliary device specified by the pn parameter is not the same type as the system default, the r parameter must be included; if not, this error code is returned. |

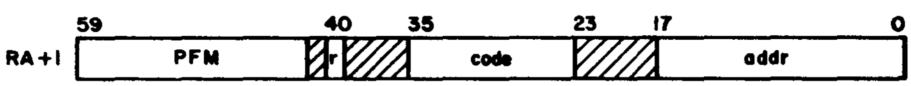
| Error Codes | Description |
|---|---|
| 12 | The local file specified for a SAVE, REPLACE, or APPEND command or macro exceeds the length allowed, or the direct access file specified for an ATTACH in WRITE, MODIFY, or APPEND mode exceeds the direct access file length limit for which the user is validated. |
| 13 | One of the following:<br>• Illegal function code passed to PFM<br>• Illegal permit mode or catalog type specified<br>• CATLIST request has permit specified without a file name<br>• PERMIT attempted on a library file |
| 14 | Access to the permanent file device requested is not possible. |
| 15 | The device on which the file resides may not contain direct access files because:<br>1. The device is not specified as a direct access device in the catalog descriptor table.<br>2. The device is not specified as ON and initialized in the catalog descriptor table.<br>3. The device is a dedicated indirect access permanent file device. |
| 16 | Because a permanent file utility is currently active, the operation is not attempted; the user should retry the operation. |
| 17 | An error occurred in a read operation during a file transfer. |
| 20 | The number of files in the user's catalog exceeds the limit (refer to LIMITS control statement, volume 1, section 6). |
| 21 | The cumulative size of the indirect access files in the user's catalog exceeds the limit (refer to LIMITS control statement, volume 1, section 6). |
| 22 | The number of PRUs specified via the s parameter on the DEFINE macro is not available. |
| 23 | A request is attempted on a local file that is currently active. This error can occur, for example, if the user creates two FETs for the same file and issues a second request before the first is complete. |
| 24 | The job's local file limit has been exceeded by an attempt to GET or ATTACH the file. |
| 25 | The job's mass storage PRU limit is exceeded during preparation of a local copy of an indirect access file. |

| Error Codes | Description |
|---|---|
| 26 | Permit limit is exceeded for a private file. |
| 30 | The resource executive detected a fatal error. |
| 31 | No allocatable tracks remain on equipment xx, where xx is the EST ordinal. |
| 32 | The length of a file does not equal the catalog length; the action taken depends on the type of request issued. |

| Request | Action |
|---|---|
| GET | A local file is created with length being the actual length retrieved. |
| SAVE | If file length is longer than TRT specification, file is truncated. |
| REPLACE | Same as for SAVE. |

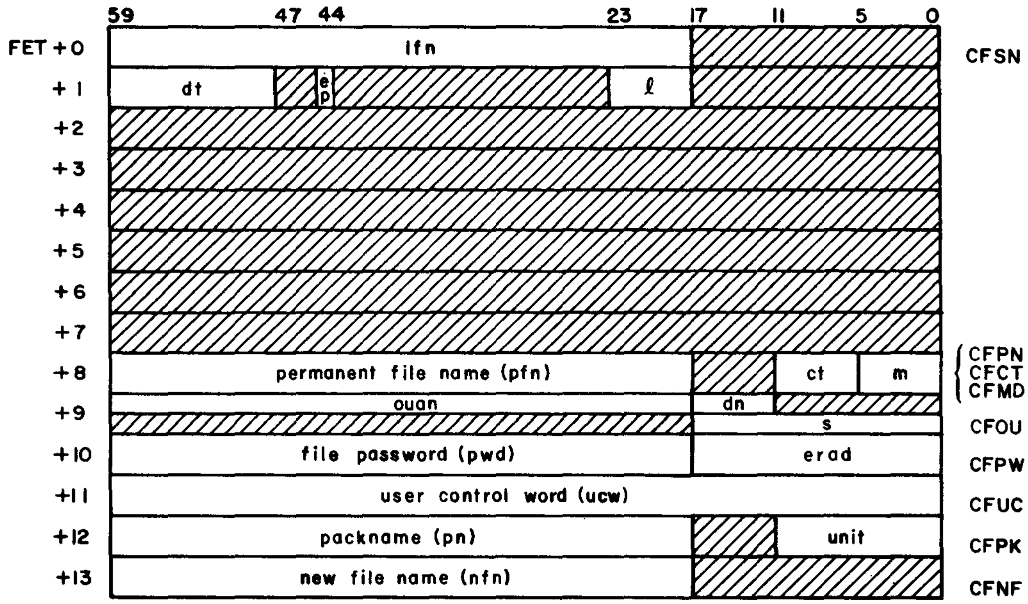
| Error Codes | Description |
|---|---|
| 33 | PERMIT random address error. |
| 34 | The system sector data for the file does not match the catalog data. |
| 35 | The same file is found twice during a catalog search. This error can occur for APPEND or REPLACE requests after a file is found and purged and the catalog search is continued. |
| 36 | Error flag detected at PFM control point. |
| 37 | An error is encountered in reading a portion of the permanent file catalog or permit information. |

The format of an RA+1 call to PFM is as follows.



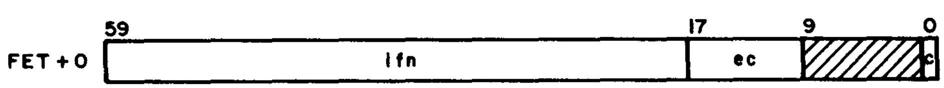

| | |
|---|---|
| r | Auto recall bit (must be set) |
| code | Function code |
| addr | Address of the FET |

The FET used by all PFM requests is of the following form. For a more detailed description, refer to the discussion of the FET in section 3.




| | |
|---|---|
| lfn | Local file name |
| dt | Device type |
| ep | Error processing bit (bit 44) |
| ℓ | FET length minus 5 |
| pfn | Permanent file name |
| ct | File category (refer to Permission Modes, File Categories) |
| m | File access mode (refer to Permission Modes, File Categories) |
| ouan | Option user number |
| dn | Device number for CATLIST option |
| s | Number of PRUs (octal) desired for the file |
| pwd | Optional file password |
| erad | Error message return address |
| ucw | User control word |
| pn | Pack name of auxiliary device |
| unit | Number of units of multiunit device |
| nfn | New file name |

After a request to PFM is complete, the first word of the FET contains the following information.




| | |
|---|---|
| lfn | local file name |
| ec | error code |
| c | Bit 0 is set to 1 upon completion of the request |

The FET length may be five words, if no special options are required, or up to 14 words, depending on the special options required.

When a PFM macro request is issued, the parameter values specified are placed in their corresponding fields in the FET.

If the permanent file name is not specified in a macro, word 8 of the FET contains the permanent file name. If X1 is specified, X1 contains the permanent file name left-justified and zero-filled. If word 8 does not contain a file name, the contents of word 0 is used as the permanent file name.

If the optional user number is not specified in a macro, word 9 of the FET contains the name of the alternate catalog. If X3 is specified, X3 contains the user number left-justified and zero-filled.

If the file password is not specified in a macro, word 10 of the FET contains the file password. If X2 is specified, X2 contains the file password left-justified and zero-filled.

If the user control word is not specified in a macro, word 11 of the FET contains the user control information. Bit 59 must be set if word 11 of the FET contains data for PFM to process.

If the pack name is not specified in a macro, word 12 of the FET contains the pack name.

If the new file name is not specified in the CHANGE macro, word 13 of the FET contains the new file name.

The address of the FET must be supplied in the addr parameter. The call to PFM must be made with the auto recall bit set. If the user wishes to process PFM errors, the error processing bit in the FET must be set. The error codes are returned in the abnormal termination field of the FET (word 0, bits 17 through 10). If the user specifies erad, the error message is returned at this address instead of being issued to the user's dayfile. A maximum of three central memory words are returned. System errors (error codes 30 through 37) are still issued to the system and error log dayfiles, however, even if erad is specified.

The common decks required for an absolute assembly of a program containing PFM requests are COMCPFM and COMCSYS.

## PERMISSION MODES, FILE CATEGORIES

Several methods are available to the user to specify the m and/or ct parameters when they are used in PFM macros. The user can either specify a 1- or 2-character key or an address that contains an integer value that corresponds to a 1- or 2-character key. If an address is specified, the value must be right-justified and zero-filled in a 60-bit word. The values can be established with the mnemonics in the following list. The user must call common deck COMSPFM to use these mnemonics (relocatable or absolute assembly). The valid mnemonics, keys, and values for the m and ct macro parameters are listed.

| Parameter | Mnemonic | Key | Value | Description |
|---|---|---|---|---|
| m | | | | File or user permission mode: |
| | PTWR | W | 0 | Allows the user to write, read, append, execute, modify, and/or purge the file. This mode can be specified for direct or indirect access files. |
| | PTRD | R | 1 | Allows the user to read and/or execute the file. This mode can be used for direct or indirect access files. |
| | PTAP | A | 2 | Allows the user to append information to the end (EOI) of the file. This mode can be specified for direct or indirect access files. |
| | PTEX | E | 3 | Allows the user to execute the file. This mode can be specified for direct or indirect access files. |
| | PTNU | N | 4 | Removes permission previously granted via PERMIT macros. This mode can be specified for direct or indirect access files. |
| | PTMD | M | 5 | Allows the user to modify, append, read, and/or execute a direct access file. Adding new information within the existing boundaries of the file is legal, but the file size must be maintained. |

| Parameter | Mnemonic | Key | Value | Description |
|---|---|---|---|---|
| | PTRM | RM | 6 | Allows the user to read and/or execute a direct access file with the implication that another user may currently be accessing the same file in M (modify) mode. This mode can be specified only for direct access files. |
| | PTRA | RA | 7 | Allows the user to read and/or execute a direct access file with the implication that another user may currently be accessing the same file in A (append) mode. This mode can be specified only for direct access files. |
| ct | | | | File category: |
| | FCPR | P | 0 | Private. Private files are available for access only by the originator or those to whom the originator has explicitly granted permission (refer to the PERMIT macro). |
| | FCSP | S | 1 | Semiprivate. Semiprivate files are available for access by all users who know the file name, user number, and password. The system records in the originator's catalog the user number of each user who accessed the file, the number of accesses, and the data and time of the last access. |
| | FCPB | PU | 2 | Public. Public files are available for access by all users who know the file name, user number, and password. The system records the number of times the file was accessed but does not record user numbers or the last access date and time. |

System tags are also available in COMSPFM for the PFM functions and the FET parameter words. The function mnemonics are specified in the macro descriptions. The FET parameter words are shown in the PFM file environment table diagram.

## AUXILIARY DEVICE REQUEST

Unless the user explicitly declares otherwise, all permanent files reside on family devices. As stated in section 2, volume 1, the user may wish to supplement the mass storage provided by his family devices by retaining his files on auxiliary devices. There are four parameters (pn, r, un, and pwd) that uniquely identify file lfn on an auxiliary device.

- The pn parameter specifies the 1- to 7-character system-defined pack name of the auxiliary device. The device can be either public or private, as defined by the installation.
- The r parameter specifies the type of auxiliary device on which the file resides or is to reside. An auxiliary device is any supported device which an installation defines as auxiliary; it need not be physically removable as the pack name implies.

  If the device is physically removable, its device type is one of the following.

  | r | Device |
  |---|---|
  | DIn | 844-21 Disk Storage Subsystem ($1 \leq n \leq 8$) |
  | DJn | 844-41/44 Disk Storage Subsystem ($1 \leq n \leq 8$) |
  | MDn | 841 Multiple Disk Drive ($1 \leq n \leq 8$) |

  If the user needs two or more physically removable auxiliary devices at any one time during his job, he must include a RESOURC control statement (refer to section 6, volume 1). An installation can provide additional continuous storage on a DI, DJ, or MD type device by combining from two to eight physical units into one logical unit. A device so defined is known as a multiunit device. To specify such a device, the r parameter must include the number of units. For example, if four 844-21 units have been combined as one multiunit device, the r parameter must be DI4. If it is not, the job is aborted and PFM error message 11 is issued to the user's dayfile.

  ILLEGAL DEVICE REQUEST, AT nnn.

  However, if r is DI, DJ, or MD but n is omitted, the unit count is assumed to be 1.

  The r parameter is required only if the desired device has a device type different from that of the available device, and the installation has defined the desired device as removable. However, if the user always specifies the r parameter, he can be sure that he is accessing the proper device. If r is specified and it conflicts with that of the available device, PFM error message 11 is issued. For all PFM macros, if pn is specified but the device is not available, the job is aborted. By setting the error processing bit (FET+1, bit 44), the user can bypass the abort and direct the system to make available the device with packname pn and device type r.

- The un parameter specifies the 1- to 7-character optional user number. If the device is public, files are accessed in the same manner as specified for family devices. That is, the un parameter must be included only if the user wishes to access files which another user has explicitly or implicitly permitted him to use. If the device is private, there is only one owner. All other users who have the proper validation can access files on the device, but the system prevents them from creating files.
- The pwd parameter specifies the 1- to 7-character password associated with the file. As with files on family devices, the pwd parameter must be included only if the originator requires that a password be specified.

## SAVE (001, CCSV)

The SAVE macro enables the user to retain a copy of a local file in the permanent file system as an indirect access file. The original file is rewound when completed.

Macro Format:

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | SAVE | lfn, pfn, pwd, ucw, ct, m, pn, r, fo |

lfn — Address of the FET; the local file name must be set in word 0 of the FET.

pfn — Address containing the name of the file; name the file is given in the permanent file catalog.

pwd — Address containing the password to be placed with the file in the permanent file catalog.

ucw — Address containing user control word to be placed with the file in the permanent file catalog. Bit 59 of FET+11 must be set if this information is to be retained. If this word exists in the catalog, it is returned to FET+11 when the file is accessed (default=0).

ct — File category:

- P — Private file (default value)
- S — Semiprivate file
- PU — Public file

If a value other than those listed is supplied, PFM assumes it is the address of the location containing the file category, right-justified, in bits 2 through 0; P is 0, S is 1, and PU is 2 (refer to Permission Modes, File Categories in this section).

m — File mode. This mode defines the type of access alternate users may have for semiprivate or public files.

- W — Read, write, purge, and execute (default value)
- R — Read and execute
- A — Append
- E — Execute
- N — None

If a value other than those listed is supplied, PFM assumes it is the address of the location containing the permission mode, right-justified, in bits 3 through 0; W is 0, R is 1, A is 2, E is 3, and N is 4 (refer to Permission Modes, File Categories in this section).

pn — Address containing 1- to 7-character pack name of the auxiliary device on which the file is to be saved.

r — Type of auxiliary device on which the file is to be saved (refer to the DEFINE macro).

fo — Family option:

- IP — The pack name specified by a PACKNAM macro or pn parameter is ignored. PFM accesses the user's family.

DF The pack name specified by a PACKNAM macro or pn parameter and the family name specified on the USER statement are ignored. PFM accesses the system default family. This option can be used only by programs that have an SSJ= entry point.

An example of the use of the SAVE macro is:

```
        SAVE     FILE,PF,,,PU,R
          .
          .
          .
FILE    FILEB    BUF,BUFL,(FET=9)
PF      VFD      24/0LDATA,36/0
```

This sequence of code saves local file FILE in the permanent file system as a public file named DATA with read permission. The same results could be accomplished by the following.

```
        SA1      PF
        SAVE     FILE,X1,,,FCAT,MDE
          .
          .
          .
FCAT    CON      FCPB
MDE     CON      PTRD
FILE    FILEB    BUF,BUFL,(FET=9)
          .
          .
          .
```

## GET (002, CCGT)

The GET macro enables the user to generate a working copy of an indirect access permanent file. If a local file by the same name already exists, it is returned as if the RETURN macro had been issued even if the GET is unsuccessful. The new file is set to rewound status. No interlock is provided to prevent other users from obtaining working copies of the same file simultaneously.

Macro Format:

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | GET | lfn, pfn, un, pwd, pn, r, fo |

lfn Address of the FET; the local file name must be set in word 0 of the FET.

pfn Address containing the name of the file; name the file is given in the permanent file catalog.

| | |
|---|---|
| un | Address containing the number of the alternate user whose catalog is to be searched for the file specified; if this parameter is specified, the permission mode is that which the user has been permitted for private files or that specified in the catalog for semiprivate and public files. |
| pwd | Address containing the password of the file; required if un $\neq$ 0 and the file requires a password. |
| pn | Address containing 1- to 7-character pack name of the auxiliary device from which the file is to be retrieved. |
| r | Type of auxiliary device on which the indirect access permanent file resides (refer to the DEFINE macro). |
| fo | Family option:<br>IP The pack name specified by a PACKNAM macro or pn parameter is ignored. PFM accesses the user's family.<br>DF The pack name specified by a PACKNAM macro or pn parameter and the family name specified on the USER statement are ignored. PFM accesses the system default family. This option can be used only by programs that have an SSJ= entry point. |

## PURGE (003, CCPG)

The PURGE macro enables the user to remove the specified file from the permanent file system. To purge a file in an alternate user's catalog, the user must have write permission to the file or the file must be a semiprivate or public file with write mode.

Macro Format:

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | PURGE | lfn, un, pwd, pn, r, fo |

| | |
|---|---|
| lfn | Address of the FET; the local file name must be set in word 0 of the FET. |
| un | Address containing the user number of the alternate catalog for the file to be purged. |
| pwd | Address containing the password of the file; required if un $\neq$ 0 and the file requires a password. |
| pn | Address containing 1- to 7-character pack name of the auxiliary device on which the file resides. |
| r | Type of auxiliary device identified by the pn parameter (refer to the DEFINE macro). |

fo Family option:

IP The pack name specified by a PACKNAM macro or pn parameter is ignored. PFM accesses the user's family.

DF The pack name specified by a PACKNAM macro or pn parameter and the family name specified on the USER statement are ignored. PFM accesses the system default family. This option can be used only by programs that have an SSJ= entry point.

An example of the use of this macro is:

```
        PURGE     F
          .
          .
          .
F       FILEB     BUF,BUFL
```

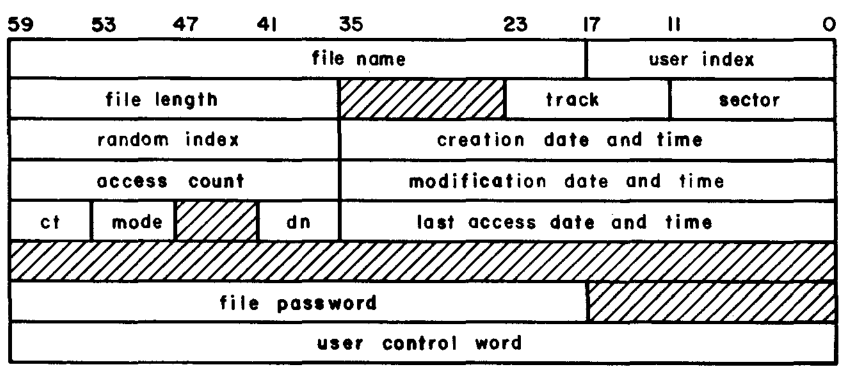
These instructions purge permanent file F.

## CATLIST (004, CCCT)

The CATLIST macro enables the user to:

- Determine the contents of his permanent file catalog.
- Determine which files in the specified alternate catalog he is allowed to access.
- Determine the alternate user information for any files that an alternate user can access or has accessed in his catalog (permit data).

For the first type of request, the entire catalog entry is returned to the user. The format of the catalog entry is illustrated.



| | |
|---|---|
| file name | Permanent file name |
| user index | User index of file creator |
| file length | Length in PRUs of the file |
| track | Beginning track of file |
| sector | Beginning sector of file (4xxx for a direct access file) |
| random index | Random disk address of permit sector |
| access count | Count of accesses to file |
| ct | File category (private, semiprivate, or public); refer to Permission Modes, File Categories in this section |
| mode | Mode of access for semiprivate and public files<br>0 Write, read, execute, append, modify, and/or purge<br>1 Read and/or execute<br>2 Append<br>3 Execute<br>4 Negate previous permission<br>5 Modify<br>6 Read and/or execute, allow modify<br>7 Read and/or execute, allow append<br>Refer to Permission Modes, File Categories in this section. |
| dn | Device number (0 through $77_8$); each device within a family of permanent file devices is identified by a device number |
| password | Optional password |
| user control word | User control information (FET+11) |

For the second type of request, the entire catalog except the user index and the password is returned to the user. The user obtains the file names of all semiprivate and public files and all private files he is permitted to use.

For the third type of request, the alternate user access information for file pfn is returned in the following format.

| 59 | 41 | 35 | 17 0 |
|---|---|---|---|
| usernum | | | ////////// |
| ac | m | ad | at |

| | |
|---|---|
| usernum | Alternate user number |
| ac | Number of accesses the alternate user has made to the file |
| m | Permission mode (bit 40 set if this was an accounting permit and was not created by a PERMIT control statement or macro; bit 40 clear indicates an explicit permit set by PERMIT statement or macro; bits 39 through 36 are same as mode described in first type of request) |

| | |
|---|---|
| ad | The last date the user accessed the file |
| at | The time of day the user last accessed the file |

Common deck COMSPFM is required for relocatable and absolute assemblies of this macro.

Macro Format:†

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | CATLIST | lfn, pfn, un, m, pn, r, fo, sr |

| | |
|---|---|
| lfn | Address of the FET for the CATLIST function |
| pfn | Address containing the file name; if this parameter is omitted, CATLIST information for all files is returned |
| un | Address containing the name of the alternate user catalog to be searched for the catalog information |
| m | If m is specified, access information (permit data) for user un on file pfn is returned. The pfn is required if this parameter is specified. If un is not specified, all permit data for pfn is returned. |
| pn | Address containing 1- to 7-character pack name of the auxiliary device that contains catalog information for all users with information on that device |
| r | Type of auxiliary device identified by the pn parameter (refer to the DEFINE macro) |
| fo | Family option:<br>IP The pack name specified by a PACKNAM macro or pn parameter is ignored. PFM accesses the user's family.<br>DF The pack name specified by a PACKNAM macro or pn parameter and the family name specified on the USER statement are ignored. PFM accesses the system default family. This option can be used only by programs that have an SSJ= entry point. |
| sr | Request CATLIST of files on device number dn (FET+9, bits 17 through 12). |

† This macro is available in common deck COMCMAC (refer to appendix A).

If the status returned in FET+0 (lfn parameter of the CATLIST macro), bits 9 through 0, is 0033 (EOF encountered), the user should reissue the CATLIST macro after the buffer of entries has been processed (refer to example 1). The user should continue this until a status of 1033 (EOI encountered) is returned. CATLIST uses the current random index field (FET+6, bits 59 through 30) to keep track of its position for continued calls. If the user changes this field, the results of a CATLIST request may be undefined.

Information is placed in the buffer starting at IN until IN = LIMIT minus 1, at which time buffer full (0033) status is set in FET+0. PFM does not process the buffer circularly; therefore the user must reset IN=OUT=FIRST before reissuing the CATLIST macro.

Example 1: The following program creates a binary file named F with CATLIST information for all files in the user's catalog. The information can be examined using the TDUMP control statement.

```
CAT
         IDENT  CAT
         ENTRY  START
         SYSCOM B1

***      COMMON DECKS REQUIRED.
*CALL    COMSPFM
*CALL    COMCMAC

 F       FILEB  BUFF,101B
 G       FILEB  BUFG,101B,FET=10D

 START   SB1    1
         MX0    50D

 STA1    CATLIST G
         SA1    G              DETERMINE ERROR CODE
         BX1    -X0*X1         (X1)=ERROR CODE
         SX1    X1-1033B
         ZR     X1,STA2        IF EOI (CATLIST COMPLETE)
         READW  G,WBUF,100B
         WRITEW F,WBUF,100B
         SA1    G+B1           RESET FET POINTERS
         SX7    X1
         SA7    A1+B1          SET IN=FIRST
         SA7    A7+B1          SET OUT=FIRST
         EQ     STA1

 STA2    SA1    G+2            (X1)=IN
         SA2    A1+B1          (X2)=OUT
         IX5    X1-X2          (X5)=WORDS IN CATALOG ENTRY
         READW  G,WBUF,X5
         WRITEW F,WBUF,X5
         WRITER F
         ENDRUN

 BUFF    BSS    101B
 BUFG    BSS    101B
 WBUF    BSS    100B

         END    START
```

```
Example 2:                 CATLIST     F,PF,AUN,M
                              .
                              .
           F               FILEB       BUF,BUFL,(FET=10D)
           PF              VFD         42/0LDATA013, 18/0
           AUN             VFD         42/0LUSERABC, 18/0
```

This returns all access information about the USERABC for file DATA013.

```
Example 3:                 CATLIST     F,,AUN
                              .
                              .
                              .
           F               FILEB       BUF,BUFL,(FET=10D)
           AUN             VFD         42/0LUSERABC, 18/0
```

This returns a list of all files that the user can access in the catalog of user USERABC.

## PERMIT (005, CCPM)

The PERMIT macro enables a user to explicitly permit another user to access a private file in his permanent file catalog. If the user wishes to remove permission previously granted, the negate mode should be selected.

Macro Format:

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | PERMIT | lfn, pfn, un, m, pn, r, fo |

**lfn** — Address of the FET; the local file name must be set in word 0 of the FET.

**pfn** — Address containing the name of the file; name of the file is given in the permanent file catalog.

| | |
|---|---|
| un | **Address containing the name of the user to whom permission is being granted.** |
| m | Mode of permission being granted; if no mode is specified, W (write) is assumed. Refer to Permission Modes, File Categories in this section. |
| pn | Address containing 1- to 7-character pack name of the auxiliary device on which the specified file resides. |
| r | **Type of auxiliary device identified by the pn parameter (refer to the DEFINE macro).** |
| fo | **Family option:** |

| | |
|---|---|
| IP | **The pack name specified by a PACKNAM macro or pn parameter is ignored. PFM accesses the user's family.** |
| DF | **The pack name specified by a PACKNAM macro or pn parameter and the family name specified on the USER statement are ignored. PFM accesses the system default family. This option can be used only by programs that have an SSJ= entry point.** |

An example of the use of this macro is:

```
         PERMIT    F,PF,AUN,2
           .
           .
           .
F        FILEB     BUF,BUFL,(FET=10D)
PF       VFD       42/0LDATA012,18/0
AUN      VFD       42/0LUSERABC,18/0
```

This allows user USERABC to have append permission to file DATA012.

## REPLACE (006, CCRP)

The REPLACE macro enables the user to place a copy of the specified local file in the permanent file system as an indirect access file. If the specified permanent file name already exists in the catalog, that file is purged and the new file placed in the catalog as the same type of file. If the file does not exist in the catalog, the new file is placed in the catalog as a private file. Permission information and alternate user access data are not lost when the file is replaced.

Macro Format:

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | REPLACE | lfn, pfn, un, pwd, ucw, pn, r, fo |

| | |
|---|---|
| lfn | Address of the FET; local file name must be set in word 0 of the FET. |
| pfn | Address containing the name of the permanent file to be replaced in the permanent file catalog. |
| un | Address containing the name of the alternate user's catalog where the file resides. |
| pwd | Address containing the password of the file being replaced (required if un $\neq$ 0); if no previous file existed, the password is saved with the file. |
| ucw | Address containing user control word to place with the file; bit 59 of word 11 of the FET must be set to retain the word with the file in the catalog. |
| pn | Address containing 1- to 7-character pack name of the auxiliary device on which the file is to be placed. |
| r | Type of auxiliary device identified by the pn parameter (refer to the DEFINE macro). |
| fo | Family option:<br>IF The pack name specified by a PACKNAM macro or pn parameter is ignored. PFM accesses the user's family.<br>DF The pack name specified by a PACKNAM macro or pn parameter and the family name specified on the USER statement are ignored. PFM accesses the system default family. This option can be used only by programs that have an SSJ= entry point. |

## APPEND (007, CCAP)

The APPEND macro enables a user to write the contents of the specified working files at the end of the specified indirect access permanent file. The logical structure of the two files is retained; that is, EORs and EOFs are appended as well as data. If the working file is appended to a file in an alternate user's catalog, a password must be supplied if one is required.

Macro Format:

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | APPEND | lfn, pfn, un, pwd, pn, r, fo |

| | |
|---|---|
| lfn | Address of the FET; name of the local file to be appended must be set in word 0 of the FET. |
| pfn | Address containing the name of the permanent file to which information is to be appended. |

un — Address containing the name of the alternate user whose catalog contains the permanent file .

pwd — Address containing the password of the file to which information is to be appended. This must be specified if un $\neq 0$ and the file requires a password.

pn — Address containing 1- to 7-character pack name of the auxiliary device on which the specified permanent file resides.

r — Type of auxiliary device identified by the pn parameter.

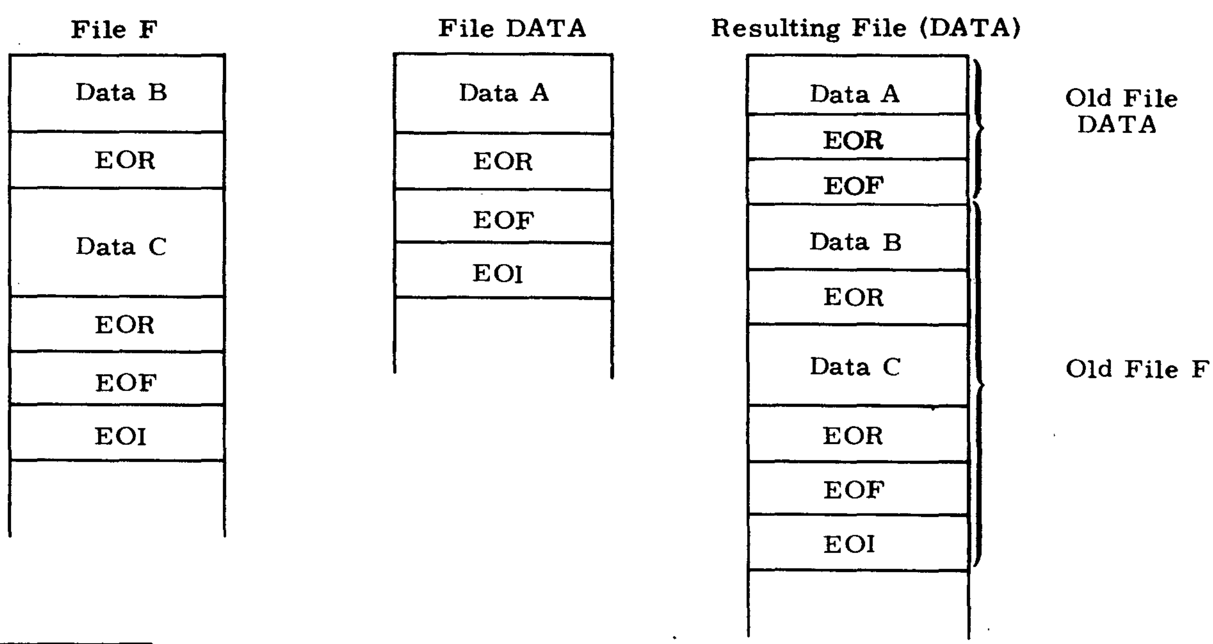
fo — Family option:

- IP — The pack name specified by a PACKNAM macro or pn parameter is ignored. PFM accesses the user's family.
- DF — The pack name specified by a PACKNAM macro or pn parameter and the family name specified on the USER statement are ignored. PFM accesses the system default family. This option can be used only by programs that have an SSJ= entry point.

An example of the use of this macro is:

```
          APPEND     F
            .
            .
            .
F         FILEB      BUF,BUFL,(PFN=DATA)†
            .
            .
```

The following diagram illustrates the structure of file DATA after the APPEND macro is issued.




† The permanent file name can be set by FET creation macros as well as being set in the permanent file macros.

## DEFINE (010, CCDF)

The DEFINE macro enables the user to specify a file as a direct access permanent file. The file specified must be a local file that resides on a permanent file device. If the file does not exist, a zero-length file is created either on the master device (refer to Permanent File Devices, section 2, volume 1, for a description of the master device), another device if the master device cannot be used, or that device specified by the r and/or pn parameters.

Macro Format:

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | DEFINE | lfn, pfn, pwd, ucw, r, ct, m, pn, s, fo |

| | |
|---|---|
| lfn | Address of the FET. Word 0 of the FET must contain the name of the file to be made a direct access file; name used when the job accesses the file. |
| pfn | Address containing the name of the permanent file entered in the catalog. If pfn=0, pfn=contents of word 0 of the FET. |
| pwd | If pwd is specified, the password is retained with the file; all alternate users must supply this password when attaching file. |
| ucw | Address containing user control information, bit 59 of word 11 of the FET must be set to retain this data. If this parameter is supplied, it is returned in word 11 of the FET when the file is attached. |

Specifies the type of family or auxiliary device on which the permanent file resides or is to reside; r can be any of the following.

| r | Display Code | Device |
|---|---|---|
| DE | 0405 | Extended Core Storage |
| DIn | 0411 | 844-21 Disk Storage Subsystem ($1 \le n \le 8$) |
| DJn | 0412 | 844-41/44 Disk Storage Subsystem ($1 \le n \le 8$) |
| DP | 0420 | Distributive Data Path to ECS |
| MDn | 1504 | 841-n Multiple Disk Drive ($1 \le n \le 8$) |

If r is not one of these values, the value specified represents the address that contains the display code equivalent, left-justified. If the device is a multiunit type device (DIn, DJn or MDn), n is the number of physical units which the installation has combined into one logical unit; n is converted from display code to octal and placed in FET+12, bits 11 through 0. Also, if r is DI, DJ, or MD but n is omitted, the unit count is assumed to be 1. If r is not specified, the system default device tvpe is used.

| | |
|---|---|
| ct | File category; ct can be one of the following. |
| | P Private file (default value) |
| | S Semiprivate file |
| | PU Public file (ct=L can also be used) |
| | If a value other than those listed is supplied, PFM assumes it is the address of the location containing the file category, right-justified in bits 2 through 0; P is 0, S is 1, and PU is 2. Refer to Permission Modes, File Categories in this section. |
| m | Access mode; this mode defines the type of access alternate users may have for semiprivate and public files. |
| | W Write, read, append, execute, modify, and/or purge (default value) |
| | R Read and/or execute |
| | A Append |
| | E Execute |
| | N None |
| | M Modify, append, read, and/or execute |
| | RM Read and/or execute, allow modifications |
| | RA Read and/or execute, allow extensions |
| | If a value other than those listed is supplied, PFM assumes it is the address of the location containing the permission mode, right-justified in bits 3 through 0. Refer to Permission Modes, File Categories in this section. |
| pn | Address containing the 1- to 7-character pack name of the auxiliary device on which the direct access file is to reside. |
| s | Address containing the number of PRUs desired for the direct access file. The number of PRUs is in octal, right-justified. DEFINE places this value in FET+9, bits 23 through 0 (CFOU). |
| fo | Family option: |
| | IP The pack name specified by a PACKNAM macro or pn parameter is ignored. PFM accesses the user's family. |
| | DF The pack name specified by a PACKNAM macro or pn parameter and the family name specified on the USER statement are ignored. PFM accesses the system default family. This option can be used only by programs that have an SSJ= entry point. |

If lfn does not exist at the time the DEFINE request is issued, the device on which pfn resides depends on the r and s parameters.

| r | s | Residency |
|---|---|---|
| Specified | Not specified | The file resides on the device of type r with the most space available. |
| Specified | Specified | The file resides on the device of type r with the most space available, provided that device has as many PRUs available as specified by the s parameter. |
| Not specified | Specified | The file resides on the device with the most space available, provided that device has as many PRUs available as specified by the s parameter. |
| Not specified | Not specified | The file resides on the device with the most space available. |

An example of the use of the DEFINE macro is:

```
          DEFINE    F,PF,,,DI ,PU,R
            .
            .
            .
F         FILEB     BUF,BUFL,(FET=9)
PF        VFD       24/0LDATA, 36/0
```

This defines file F to reside on the 844 Disk Storage Subsystem as a read-only public file. The name of the entry in the permanent file catalog is DATA. (It is assumed that the file did not exist before the DEFINE request was issued.)

The same operation could be accomplished by the following sequence of instruction.

```
          SA1       PF
          DEFINE    F,X1,,,RES,FCAT,R
            .
            .
            .
F         FILEB     BUF,BUFL,(FET=9)
PF        VFD       24/0LDATA, 36/0
RES       VFD       12/0LDI, 48/0
FCAT      CON       FCPU
```

## ATTACH (011, CCAT)

The ATTACH macro enables the user to attach the specified direct access file to his control point.

A read/write interlock is provided to ensure that only one user at a time accesses the file in write mode. Several users may access the file in read mode simultaneously. The user should return the file as soon as possible to enable other users to access the file. If the working file name specified is attached to the job, it is returned as if a RETURN macro were issued.

Macro Format:

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | ATTACH | lfn, pfn, un, pwd, m, pn, r, fo, fa |

lfn — Address of the FET. Word 0 of the FET must contain the temporary name of the file while attached to the job.

pfn — Address containing the name of the file in the permanent file catalog. If pfn=0, pfn=contents of word 0 of the FET.

un — Address containing name of the alternate user catalog on which the file resides.

pwd — Address containing file password; entered only if un≠0 and the file requires a password.

m — Mode of access desired:

| | |
|---|---|
| W | Write, read, append, execute, modify, and/or purge; the last modification date is updated whenever a file is attached in W mode even if the file is not altered during attachment (default). |
| R | Read and/or execute |
| A | Append |
| E | Execute |
| N | None |
| M | Modify, append, read, and/or execute |
| RM | Read and/or execute, allow modifications |
| RA | Read and/or execute, allow extensions |

If a value other than those listed is supplied, PFM assumes it is the address of the location containing the permission mode right-justified, in bits 3 through 0. Refer to Permission Modes, File Categories in this section.

pn — Address containing 1- to 7-character pack name of the auxiliary device on which the specified permanent file resides (refer to the DEFINE macro).

r — Type of auxiliary device identified by the pn parameter (refer to the DEFINE macro).

fo — **Family option:**

- IP — The pack name specified by a PACKNAM macro or pn parameter is ignored. PFM accesses the user's family.
- DF — The pack name specified by a PACKNAM macro or pn parameter and the family name specified on the USER statement are ignored. PFM accesses the system default family. This option can be used only by programs that have an SSJ= entry point.

fa — If this parameter is specified (any value may be used), the file being attached must be a fast attach permanent file.

If the user issues an ATTACH macro and the file is busy, the system aborts the request. The user can bypass the abort by specifying error processing (FET+1, bit 44). If ep is set and the file is busy, the system returns control to the user. He may then suspend his job by issuing the ROLLOUT macro. Normally, when a user issues a ROLLOUT macro to roll out his job subject to time/event dependencies, he must include an address specifying the time period and/or event. However, whenever file busy status is returned, PFM sets up a time/event entry for the user, specifying a default rollout time period of $360_8$ seconds.

Refer to the ATTACH control statement for a description of the resulting current access modes when the user attempts to attach an active file.

## CHANGE (012, CCCG)

The CHANGE macro allows the originator of a direct or indirect access file to alter any of several parameters without having to attach and redefine the file or retrieve and save it. The pwd, ct, and m parameters should be specified only if a change in the value associated with that parameter is desired. The pn and r parameters cannot be used to specify a new auxiliary device. They are used only to specify the auxiliary device on which ofn resides.

Macro Format:

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | CHANGE | lfn,ofn, nfn, pwd, ucw, ct, m, pn, r, fo |

lfn — Address of the FET; the local file name must be set in word 0 of the FET.

ofn — Address containing the old permanent file name. If ofn is not specified, the contents of FET+0 is used instead.

nfn — Address containing the new permanent file name. If nfn is not specified, no name change takes place.

| | |
|---|---|
| pwd | Address containing new password; replaces the old password or adds a new password if there was none before. If the user does not wish to change the current password, pwd should be set to 42/77777777777777B,18/0. |
| ucw | Address containing user control information; bit 59 of FET+11 must be set if this information is to be retained. |
| ct | New file category. If this value is set in the FET, it must be entered in the format 4x where x is the new category. Refer to Permission Modes, File Categories in this section. |
| m | New file mode. If this value is set in the FET, it must be entered in the format 4y where y is the new mode. Refer to Permission Modes, File Categories in this section. |
| pn | Address containing the 1- to 7-character pack name of the auxiliary device on which the file resides. |
| r | Address containing the type of auxiliary device identified by the pn parameter (refer to the DEFINE macro). |
| fo | Family option: |
| | IP — The pack name specified by a PACKNAM macro or pn parameter is ignored. PFM accesses the user's family. |
| | DF — The pack name specified by a PACKNAM macro or pn parameter and the family name specified on the USER statement are ignored. PFM accesses the system default family. This option can be used only by programs that have an SSJ= entry point. |

An example of the use of the CHANGE macro is:

```
        CHANGE   F,PF,NF,,,,R,PKN
          .
          .
          .
F       FILEB    BUF,BUFL,(FET=13)
PF      VFD      42/0LDATA,18/0
NF      VFD      42/0LFILE,18/0
PKN     VFD      42/0LPACK1,18/0
```

This changes file DATA on auxiliary device PACK1 to FILE with read-only access.

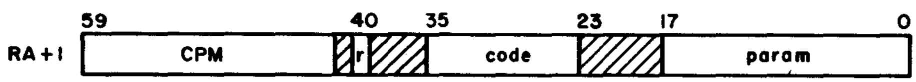
Control point manager (CPM) enables the user to alter or interrogate parameters in the job control point area which control his job in the system. All CPM functions must be issued with auto recall specified. If it is not, the following message is issued.

CPM ILLEGAL REQUEST.

All errors encountered by CPM cause the job to be aborted; no user error processing is available. Unless otherwise noted, the following message is issued upon job termination.

CPM ARG. ERROR.

The format of the call to CPM is:




| | |
|---|---|
| r | Auto recall bit |
| code | CPM function code |
| param | Parameter for the function |

Common decks required for absolute assemblies by the functions processed by CPM are COMCCPM and COMCSYS. For relocatable assemblies, these decks are satisfied by default from the library SYSLIB.

## SETQP (000)

The SETQP macro allows the user to alter the queue priority of his job to the parameter specified. The queue priority controls the scheduling of the job to and from the rollout queue. Lowering the queue priority may cause the job to be rolled out more often.

Macro Format:†

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | SETQP | n |

| | |
|---|---|
| n | Queue priority; $100_8 \le n \le$ current priority; ($100_8 \le n \le 7761_8$ for a system origin job). If n is greater than 7761, the priority is set to 7761. If n=0 is specified, the queue priority is set to the rollout upper bound for the current job origin. |

† This macro is not available in SYSTEXT. The user must call the common deck COMCMAC (refer to appendix A).

## SETPR (001)

The SETPR macro allows the user to alter the CPU priority of his job. The CPU priority controls the assignment of the CPU to active jobs. If the CPU priority is lower than that of other jobs, the job is assigned to the CPU only when jobs of a higher priority do not need it. The user is validated for a maximum CPU priority. If he requests a value that exceeds this value or $70_8$ (the maximum CPU priority), the maximum for which he is validated is used.

Macro Format: †

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | SETPR | n |

n CPU priority; ($1 \le n \le 70_8$); if n exceeds that for which the user is validated, it is reduced to that value.

## MODE (002)

The MODE macro allows the user to set the exit mode flags for CPU execution of his job (refer to figure 2-E-2). Refer to the applicable hardware reference manual for a description of all bits indicated. Section 5, volume 1, contains a summary of these descriptions.

Macro Format: †

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | MODE | m, n |

m CPU program error exit mode ($0 \le m \le 7$).

n CPU hardware error exit mode ($0 \le n \le 7$).
If n is not specified, the system assumes n=7.††

## SETASL (003)

The SETASL macro allows the user to specify the account block SRU limit for a job. The account block limit is the maximum number of SRUs that can be accumulated by a job. If this limit is reached, the following message is issued.

ACCOUNT BLOCK LIMIT.

Each user and each charge/project number is validated for a maximum SRU limit. If the limit is exceeded, the message

SL NOT VALIDATED.

is issued.

†This macro is not available in SYSTEXT. The user must call the common deck COMCCMD or COMCMAC (refer to appendix A).

††Applicable to CDC CYBER 170 Series only.

If $1 \leq s \leq 77777_8$ is not satisfied, the following message is issued.

ILLEGAL USER ACCESS.

A value of $s = 77777_8$ specifies an infinite account block SRU limit.

For the time-sharing user, this function defines the number of SRUs allowed for a job before entry of another CHARGE statement is required. For batch or time-sharing jobs, s represents the maximum SRU accumulation between two CHARGE statements, or between one CHARGE statement and the end of the job. The user may not set the account block limit to a value less than the current job step limit. If this is attempted, the following message is issued.

JOB STEP EXCEEDS ACCOUNT BLOCK:

Macro Format: †

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | SETASL | s |

s    Account block SRU limit

## SETJSL (003)

The SETJSL macro allows the user to specify the job step SRU limit for each step of a job, including the current job step. The job step limit is the maximum number of SRUs that can be accumulated by a single job step. If this limit is reached, the following message is issued.

JOB STEP LIMIT.

The job step limit cannot exceed the account block limit (refer to SETASL macro). If this is attempted, the job step is aborted and the following message is issued.

JOB STEP EXCEEDS ACCOUNT BLOCK.

If $1 \leq s \leq 77777_8$ is not satisfied, the following message is issued.

ILLEGAL USER ACCESS.

A value of $s = 77777_8$ specifies an infinite job step SRU limit.

† This macro is not available in SYSTEXT. The user must call common deck COMCCMD (refer to appendix A).

Macro Format: †

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | SETJSL | s |

s Job step SRU limit

## SETTL (003)

The SETTL macro changes the user's CPU job step time limit for each job step, including the current job step. The CPU job step time limit is the amount of time (in seconds) that a job step is allowed to use the CPU. If this limit is reached, the following message is issued.

TIME LIMIT.

The user is validated for a maximum time limit (refer to the LIMITS control statement, section 6, volume 1). If the maximum is exceeded, the following message is issued.

TL NOT VALIDATED.

If $1 \leq t \leq 77777_8$ is not satisfied, the following message is issued.

ILLEGAL USER ACCESS.

A value of $t = 77777_8$ specifies an infinite job step time limit.

Macro Format: †

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | SETTL | t |

t Job step time limit ($1 \leq t \leq 77777_8$); t is accurate to the nearest second.

† This macro is not available in SYSTEXT. The user must call common deck COMCCMD (refer to appendix A).

## EREXIT (004)

NOS allows a user program to continue execution after an error occurs (except where prohibited), if the error return address is specified in the job control point area. When an error occurs, the system clears the error exit address and restarts the job at the CPU address specified. The contents of RA returned when an error is encountered is:

| 59 | 53 | 47 | 29 | 23 | 11 | 5 0 |
|---|---|---|---|---|---|---|
| 0 | mo | aaaaaa | ef | 0 | ssw | 0 |

mo — Mode of CPU error exit

aaaaaa — CPU address of the job when the error occurred

ef — Error flag:

| Error Flag | Mnemonic | Description |
|---|---|---|
| 1 | TLET | Time limit. Job is allowed an additional 1 to 10 seconds of CPU time for error exit processing. |
| 2 | ARET | Arithmetic error. mo is the mode of the error. Refer to section 3, volume 1, for a description of modes. |
| 3 | PPET | PPU abort. A PPU program requested that the job be aborted (CIO, PFM, etc.). |
| 4 | CPET | CPU abort. The job issued an ABT request (error exit processing does not occur for this error). |
| 5 | PCET | PP call error. The job called a nonexistent or illegal system request. |
| 6 | ODET | The operator dropped the job. |
| 7 | PSET | A program stop was encountered by the CPU. |
| 10B | FLET | File limit. More active files were assigned to the job than are allowed by the validation parameter. |

| Error Flag | Mnemonic | Description |
|---|---|---|
| 11B | TKET | Track limit. The job requested mass storage space on a device with none available. |
| 12B | SYET | System abort. |
| 13B | FSET | Forced error. |
| 14B | PEET | CPU or CM parity error. † |
| 15B | ORET | Override of error condition. |
| 16B | SSET | Subsystem abort. |
| 17B | SRET | SRU limit. Job is allowed an additional 10 SRUs to complete error exit processing. |
| 20B | RRET | Job rerun. |
| 21B | OKET | Operator killed job. |

ssw Status of sense switches

Macro Format:

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | EREXIT | addr |

addr Address for error exit return

As an example, the common procedure is for the program's preset routines to set the error exit processing.

```
                              .
                              .
                              .
Routine to          ERR       SA1       BO       READ ERROR RETURN INFO
complete                      .
processing                    .
when an error                 .
is encountered                ENDRUN             TERMINATE PROGRAM
                    PRS       SUBR               PROGRAM PRESET ROUTINE
                              EREXIT    ERR      SET ERROR EXIT ADDRESS
                              .
                              .
                              .
                              EQ        PRSX
                              .
                              .
                              END
```

† Applicable to CDC CYBER 170 Series only.

## CONSOLE (005)

NOS provides a method of user/operator communication through the K and L console displays. With these displays, a job can place information on the console screen and receive information from the console keyboard.

The address specified points to the request word which is formatted as follows:

| | 59 | 53 | 35 | 17 0 |
|---|---|---|---|---|
| addr | | ib | rb | lb |

ib — Keyboard buffer address. This specifies the area where data entered by the operator is placed by the system. The data is terminated by a zero word and is always entered starting at ib. A maximum of 40 characters is transferred to this buffer.

rb — Address of the specially formatted buffer to be displayed on the right screen of the console when the K or L display is assigned to the job.

lb — Address of the specially formatted buffer to be displayed on the left screen of the console when the K or L display is assigned to the job.

Examples of the use of this macro can be found in the program STAGE.

Macro Format:†

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | CONSOLE | addr |

addr — Address of the console parameter word

## ROLLOUT (006)

The ROLLOUT macro enables a user to place his job in the rollout queue until a specified event occurs or for a fixed period of time. This function may be useful if a program requires a direct access permanent file that is currently busy.

† This macro is not available in SYSTEXT. The user must call the common deck COMCMAC (refer to appendix A).

Macro Format:†

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | ROLLOUT | addr |

addr — Address containing the time and/or event dependencies; if addr is not specified, the job is rolled out without time or event dependencies.

The format of addr is:

| addr | 59 … 30 | 29 … 12 | 11 … 0 |
|---|---|---|---|
| | 0 | evd | rtp |

evd — Event descriptor. System programs use the EESET macro†† to make entries in a system event table indicating the occurrence of an event. The job scheduler compares the specified descriptor, evd, with events recorded in the table. If a match is detected, the scheduler initiates rollin.

- evd≠o — evd and rtp are placed in the control point area (TERW). When the job rolls out, the scheduler waits for the occurrence of evd or for the time period, rtp, to elapse before initiating rollin. Because the job may roll in for two different reasons, it is the user's responsibility to verify whether the specified event actually occurred.
- evd=0 — The job scheduler checks rtp in RA+addr and evd in the control point area. This option allows the user to roll out while waiting for a system-specified event.
- evd=$7700xx_8$ — Specifies extended time rollout with no event dependency. The job rolls out for $7777*xx+rtp_8$ seconds.

rtp — Rollout time period in job scheduler delay intervals (assume 1-second intervals) where

$o \le rtp \le 7777_8$

- rtp=0 — The job rolls out for a time determined by the system to ensure that the job will be rolled in even if the specified event is not detected or never occurs.

† This macro is not available in SYSTEXT. The user must call the common deck COMCMAC (refer to appendix A).

† † Use the DOCMENT control statement to obtain internal documentation of the EESET macro and the ROLLOUT macro for a description of valid event descriptors.

## GETASL (007)

The GETASL macro returns the account block SRU limit for the job (refer to SETASL macro) to the specified address.

| | 59 | 17 0 |
|---|---|---|
| addr | 0 | s |

s Account block SRU limit

Macro Format: †

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | GETASL | addr |

addr Address to receive account block SRU limit

## GETJSL (007)

The GETJSL macro returns the job step SRU limit for the current job step (refer to SETJSL macro) to the specified address.

| | 59 | 17 0 |
|---|---|---|
| addr | 0 | s |

s Job step SRU limit

Macro Format: †

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | GETJSL | addr |

addr Address to receive job step SRU limit

† This macro is not available in SYSTEXT. The user must call common deck COMCMAC (refer to appendix A).

## SETSSM (010)

The SETSSM macro enables the user to set or clear the secure system memory (SSM) flag. Setting this flag prevents the dumping of any portion of the job field length. This flag is automatically set for programs containing an SSJ= or SSM= entry point (refer to appendix F). If a request is made to clear SSM status by an SSM= program, the job step is aborted and the following message is issued.

CPM ILLEGAL REQUEST.

**NOTE**

While the SSM flag is set, no programs with a DMP= special entry point (refer to appendix F) can be called to the control point (refer to Security Considerations, section 2) unless the calling program contains an SSJ= entry point.

Macro Format:†

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | SETSSM | p |

p Clear SSM flag if p = 0; set SSM flag if p ≠ 0

† This macro is not available in SYSTEXT. The user must call common deck COMCMAC (refer to appendix A).

## ONSW (011)

The ONSW macro enables the user to set the sense switches for his program (refer to figure 2-E-1). This allows the user to set switches for options for subsequent tasks in the job.

The sense switches reside both in the user's control point area and in bits 11 through 6 of RA+0 of the job field length. These two fields are maintained separately. The only transfer of information occurs at the start of each job step and when an error flag is detected. At these times the control point area field is copied into RA+0. The field in RA+0 is used when the user reads sense switch settings. An ONSW (or OFFSW) request updates both fields separately.

The bit position specifies the switch to be set.

Macro Format:

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | ONSW | n |

n — Switches to be set; $0<n<77_8$; bit 0 corresponds to switch 1, bit 1 corresponds to switch 2, etc. If a bit is set, the corresponding switch is set; for example, ONSW $52_8$ sets switches 2, 4, and 6. If n=$77_8$, all switches are set. If n=0, all switches remain unchanged.

## OFFSW (012)

The OFFSW macro enables the user to clear the sense switches in RA. Refer to the description of the ONSW macro for a discussion of sense switch settings.

Macro Format:

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | OFFSW | n |

n — Switches to be cleared; $0<n<77_8$; bit 0 corresponds to switch 1, bit 1 corresponds to switch 2, etc. If a bit is set, the corresponding switch is cleared; for example, OFFSW $52_8$ clears sense switches 2, 4, and 6. If n=$77_8$, all switches are cleared. If n=0, all switches remain unchanged.

## GETJN (013)

The GETJN function allows the user to determine the job name of the job. For the format of the job name, refer to section 3, volume 1.

Macro Format:†

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | GETJN | addr |

addr	Address to receive job name, left-justified

| | 59 | 17 0 |
|---|---|---|
| addr | jobname | 0 |

## GETQP (014)

The GETQP function allows the user to determine the queue priority of the current job.

Macro Format:†

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | GETQP | addr |

addr	Address to receive queue priority

| | 59 | 11 0 |
|---|---|---|
| addr | 0 | qp |

qp	Job queue priority

## GETPR (015)

This function allows the user to determine the CPU priority of the current job.

Macro Format:†

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | GETPR | addr |

addr	Address where the CPU priority is returned

| | 59 | 11 0 |
|---|---|---|
| addr | 0 | pr |

pr	CPU priority

† This macro is not available in SYSTEXT. The user must call the common deck COMCMAC (refer to appendix A).

## GETEM (016)

The GETEM macro enables the user to determine under what exit mode control the job is currently running.

Macro Format:†

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | GETEM | addr |

addr          Address for return of current exit mode in bits 11 through 0

## GETTL (017)

The GETTL macro returns the time limit for the current job step.

Macro Format:†

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | GETTL | addr |

addr          Address to receive time limit

| | 59 | 11 0 |
|---|---|---|
| addr | 0 | tl |

tl          CPU job step time limit (in seconds)

## SETUI (021)

The SETUI macro allows the user to set the permanent file catalog control parameter (user index). This macro should not be used unless a validation file is unavailable. This macro is normally used only by the system utilities that have access to special portions of the permanent file system. If the job is not of system origin, the following message is issued.

CPM ILLEGAL REQUEST.

† This macro is not available in SYSTEXT. The user must call the common deck COMCCMD or COMCMAC (refer to appendix A).

Macro Format:†

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | SETUI | ui |

ui User index; $0 \le ui \le 377777_8$

# SETLC (022 )

The SETLC macro enables the user to set the loader control word for subsequent loader requests. The format of the loader control word is:


v Map validation bit. If bit is not set, default map option is used.

mc Map control bits (octal):

| | |
|---|---|
| 00 | No map is produced |
| 01 | Statistics and errors |
| 02 | Block assignments |
| 03 | Partial map providing statistics, errors, and block assignments |
| 04 | Entry points |
| 14 | External references and entry points |
| 17 | Full map providing information given individually by control bits 01, 02, 03, and 14 |

n Prevent field length reduction after relocatable load.

g Global library set indicators.

The global library set indicators, contained in the loader control word (location LB1W in the user's control point area), are maintained by the SETLC macro. The user should call the GETLC macro before using the SETLC macro to avoid destroying the library set information. With the GETLC macro, the user can obtain the current loader control word and include bits 23 through 0 in the control word used by SETLC.

Macro Format†

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | SETLC | addr |

addr Address of the loader control word to be set

† This macro is not available in SYSTEXT. The user must call the common deck COMCMAC (refer to appendix A).

## SETRFL (023)

The SETRFL macro sets the initial field length for a job step. This value is used unless the system encounters one of the following.

- A routine with an RFL= or MFL= special entry point (refer to appendix F).
- A routine that specifies the amount of field length required in a 54 loader table.
- An MFL or RFL control statement (refer to section 6, volume 1) or a subsequent SETRFL macro.

If the user does not issue the SETRFL function or the RFL control statement, the operating system determines how much field length to assign initially for each job step. If the user attempts to set the field length above that for which he is validated, the following message is issued.

CM NOT VALIDATED.

The field length is rounded upward to a multiple of $100_8$ words.

Macro Format: †

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | SETRFL | n |

n — New field length restoration parameter

## GETJCR (024)

The GETJCR macro enables the user to interrogate the job control registers associated with his job (refer to the description of the system control language, section 4, volume 1). The last error flag encountered can also be determined. Information is returned in the following format.

| | 59–54 | 53–36 | 35–18 | 17–0 |
|---|---|---|---|---|
| addr | pef | R3 | R2 | R1 |

pef — The last error flag encountered (refer to the EREXIT function for the values)

R1 — Job control register 1

R2 — Job control register 2

R3 — Job control register 3

† This macro is not available in SYSTEXT. The user must call the common deck COMCCMD (refer to appendix A).

Macro Format:†

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | GETJCR | addr |

addr Address for return of the job control registers

## SETJCR (025)

The SETJCR macro enables the user to set the job control registers for the job. Refer to the description of the GETJCR macro for the format of the word.

Macro Format:†

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | SETJCR | addr |

addr Address of the word containing the job control registers to be set

| | 59–54 | 53–36 | 35–18 | 17–0 |
|---|---|---|---|---|
| addr | pef | R3 | R2 | R1 |

pef Previous error flag

R1 Job control register 1

R2 Job control register 2

R3 Job control register 3

## SETSS (026)

The SETSS macro enables the user to specify the subsystem under which he is currently executing, provided he is validated for that subsystem. If he is not, then the operation is aborted and the user is issued the following error message.

ILLEGAL USER ACCESS.

†This macro is not available in SYSTEXT. The user must call common deck COMCMAC (refer to appendix A).

Macro Format:†

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | SETSS | ss |

ss Subsystem ordinal (refer to the TSTATUS macro, section 12, for a description of subsystem ordinals)

## GETJO (027)

This function reads the job origin code from the user's control point area and returns the code right-justified at the address specified by the user.

Macro Format:†

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | GETJO | addr |

addr Address to return job origin code. Refer to appendix E for legal job origin codes.

| | 59–6 | 5–0 |
|---|---|---|
| addr | 0 | jo |

jo Job origin code

## GETJA (030)

This function transfers the job accounting information to the address specified by the user.

The data is returned in the following form.

| | 59–36 | | 35–20 | 19–0 |
|---|---|---|---|---|
| addr+0 | 0 | SRU accumulator (micro units/10) | | |
| +1 | 0 (59–42) | 0 (41–36) | CP accumulator (milli units) | |
| +2 | MS accumulator (59–40) | MT accumulator (39–20) | | PF accumulator |
| +3 | 0 | | | adder accumulator |

† This macro is not available in SYSTEXT. The user must call the common deck COMCMAC (refer to appendix A).

Macro Format:†

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | GETJA | addr |

addr    Address to receive job accounting data

## USECPU (031)

The USECPU macro specifies which central processor is to be used when more than one is available.

Macro Format:

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | USECPU | n |

n=0    Either central processor can be used

n=1    CPU 0 is to be used (the CPU with functional units on a 6700 or CDC CYBER 74-2x)

n=2    CPU 1 is to be used (the CPU without functional units on a 6700 or CDC CYBER 74-2x)

## USERNUM (032)

The USERNUM macro returns the user number the job is running under to the specified address.

Macro Format:

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | USERNUM | addr |

addr    Address to receive the user number, left-justified and zero-filled

† This macro is not available in SYSTEXT. The user must call the common deck COMCMAC (refer to appendix A).

## GETFLC (033)

The GETFLC macro returns the field length control word from the user's control point area to the specified address.

Macro Format: †

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | GETFLC | addr |

addr    Address to receive the field length control word

The field length control word is returned in the following format.

| | 59 | 47 | 35 | 23 | 11 0 |
|---|---|---|---|---|---|
| addr | mfl | rfl | jmfl | rifl | flir |

| | |
|---|---|
| mfl | Maximum field length for the current job step. This value may be reset with the MFL control statement (refer to section 6, volume 1). |
| rfl | Initial running field length for a job step. This value is always less than or equal to mfl and is set with the RFL control statement or SETRFL macro. A value of 0 indicates that the system controls the field length. |
| jmfl | Maximum field length for the entire job. The jmfl represents the upper bound on mfl. |
| rifl | Field length for the job when it is to be rolled in. |
| flir | Field length increase requested. |

## PACKNAM (035)

The PACKNAM 035 request places the specified pack name in the user's control point area. It allows the user to omit the pn parameter from PFM requests for files residing on the auxiliary device. However, if permanent files on another auxiliary device are to be accessed, the pn parameter can be specified in the request or another PACKNAM macro can be issued.

† This macro is available only in COMCMAC (refer to appendix A).

Macro Format: †

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | PACKNAM | addr |

addr — Address containing the pack name of the auxiliary device to be accessed in subsequent PFM requests; if the pack name is set to zero, the current default pack name in the control point area is cleared. The user can then access permanent files residing on the normal system.

## PACKNAM (036)

The PACKNAM 036 request obtains the default pack name from the control point area and returns it to the specified address.

Macro Format: †

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | PACKNAM | addr, N |

addr — Address to which the pack name is to be returned

N — Indicates that the pack name is to be returned

## GETSS (037)

With this macro a user can determine the current subsystem. The subsystem ordinal is returned to an address specified by the user.

Macro Format: †

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | GETSS | addr |

addr — Address to receive subsystem ordinal (refer to TSTATUS macro, section 12, for a description of subsystem ordinals)

† This macro is not available in SYSTEXT. The user must call common deck COMCCMD or COMCMAC (refer to appendix A).

The subsystem ordinal is returned in the following format.

| | 59 | 2 0 |
|---|---|---|
| addr | 0 | ss |

ss — Subsystem ordinal

## VERSION (044)

The VERSION macro returns the version name of the operating system from central memory to a location specified by the user. Location addr contains parameters specifying the disposition of the version name.

| | 59 | 47 | 35 | 23 | 17 0 |
|---|---|---|---|---|---|
| addr | bc | sb | bp | | waddr |

| | |
|---|---|
| bc | Number of bytes to return (1 to 10 from 2-word entry) |
| sb | Byte in source field (CM location containing version name) to begin transfer at (0 to 9); the sum of bc and sb must be less than 11 |
| bp | Byte position within receiving field (waddr) to begin transfer at (0 to 4) |
| waddr | Address of first word to receive data |

Macro Format: †

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | VERSION | addr |

addr — Address of word containing macro parameters

## GETLC (045)

The GETLC macro enables the user to determine the loader control word. Refer to the SETLC macro for the format of the loader control word.

Macro Format: †

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | GETLC | addr |

addr — Address to receive the loader control word

† This macro is not available in SYSTEXT. The user must call the common deck COMCCMD or COMCMAC (refer to appendix A).

## GETGLS (046)

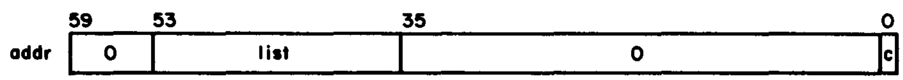
The GETGLS macro returns the global library set from the user's control point area. Refer to the CDC CYBER Loader Reference Manual for a discussion of global library sets. A parameter word is used to specify where a list of logical file names is to be placed.

Macro Format:†

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | GETGLS | addr |

addr Address of parameter word

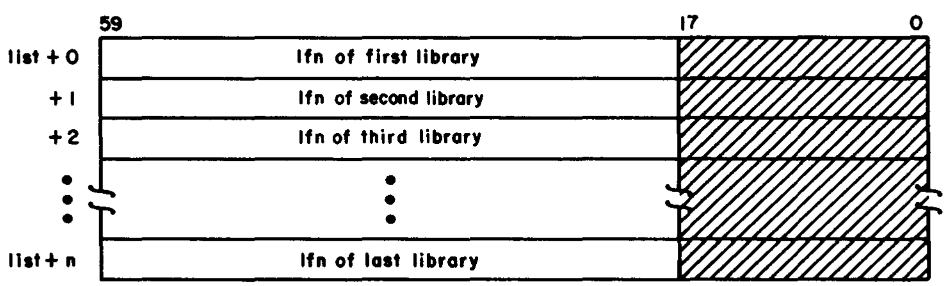
Parameter word format:




list Beginning address to which the logical file names (left-justified) contained in the global library set are written. The value of this parameter is updated to the address of the last library (list+n) upon completion.

c Completion bit.

Upon return from the GETGLS macro, locations list through list+n are updated as follows:




n The number of libraries minus one defined in the global library set

†This macro is not available in SYSTEXT. The user must call common deck COMCMAC (refer to appendix A).

## SETGLS (047)

The SETGLS macro enables the user to define the global library set indicators in the user's control point area. Refer to the CDC CYBER Loader Reference Manual for a discussion of global library sets. A parameter word specifies where the list of logical file names used to define the global library set is located.

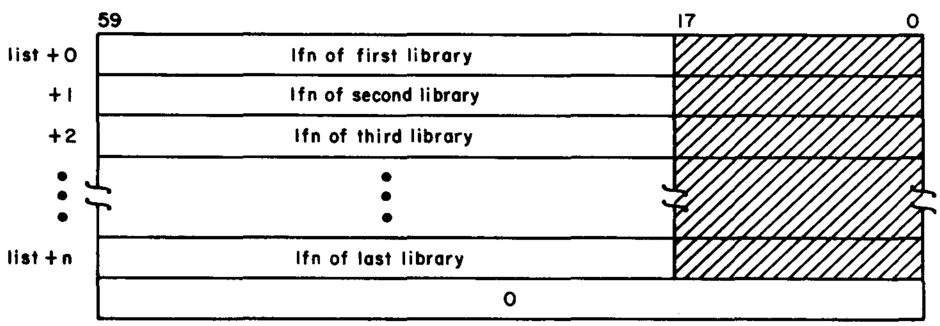
Macro Format:†

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | SETGLS | addr |

addr Address of parameter word

Refer to the GETGLS macro for the format of the parameter word.

Before calling the SETGLS macro, locations list through list+n+1 must be as follows:




n The number of libraries minus one in the global library set

†This macro is not available in SYSTEXT. The user must call common deck COMCMAC (refer to appendix A).

## MACHID (050)

The MACHID macro enables the user to determine the one- or two-character machine identification that is established at deadstart time.

Macro Format: †

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | MACHID | addr |

addr Address to receive machine identification

The machine identification is returned in the following format.

| | 59 … 12 | 11 … 0 |
|---|---|---|
| addr | 0 | id |

## GETMC (051)

The GETMC macro allows the user to obtain information about machine characteristics and the system environment.

Macro Format: †

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | GETMC | addr |

addr Address to receive machine characteristics

† This macro is not available in SYSTEXT. The user must call the common deck COMCCMD or COMCMAC (refer to appendix A).

The machine characteristics are returned in the following format.

| | 59 | | 47 | 35 | 23 | 17 | 11 0 |
|---|---|---|---|---|---|---|---|
| addr | | cs | ecs | ppu | | mc | cm |

| | |
|---|---|
| cs | If bit 48 is set, system is operating in 64 character set, and 63 character set if bit 48 is clear |
| ecs | Extended core storage (ECS) size/1000 |
| ppu | Number of PPUs in system |
| mc | Machine characteristics |

| Bit No. | Description |
|---|---|
| 17 | Unused |
| 16 | Set if machine is CDC CYBER 170 |
| 15 | Set if compare move unit (CMU) option is present |
| 14 | Set if CEJ/MEJ option is present |
| 13 | Set if CPU0 has instruction stack |
| 12 | Set if CPU1 is present |

cm Central memory size/100

## SETMFL (052)

The SETMFL macro enables the user to change the job maximum field length boundary (refer to GETFLC and SETRFL macros for discussion of maximum field length limit).

Macro Format:†

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | SETMFL | n, m |

| | |
|---|---|
| n | New maximum central memory field length limit for entire job |
| m | New maximum ECS FL/$1000_8$ limit for job †† |

† This macro is not available in SYSTEXT. The user must call common deck COMCCMD (refer to appendix A).

†† Currently not supported.

## GETPFP (057)

The GETPFP macro enables the user to read his permanent file parameters (current family name, pack name, user number, and user index).

Macro Format:†

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | GETPFP | addr |

addr — Address to receive 3-word table containing permanent file parameters

The current permanent file parameters are returned in the following format.

| | 59–18 | 17–0 |
|---|---|---|
| addr +0 | family name | 0 |
| +1 | pack name | 0 |
| +2 | user number | user index |

† This macro is not available in SYSTEXT. The user must call common deck COMCMAC (refer to appendix A).

## GETLOF (061)

The GETLOF macro returns the address of the list of files from the user's control point area to the specified address. The address of the list of files is set in the control point area with the SETLOF macro.

Macro Format: †

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | GETLOF | addr |

addr    Address to receive the list of files pointer

Upon return from the GETLOF macro, location addr has the following format.

| | 59–48 | 47–30 | 29–1 | 0 |
|---|---|---|---|---|
| addr | 0 | pointer | 0 | c |

pointer    Address of list of files table established with the SETLOF macro

c    Completion bit

If the system returns 0 in the pointer field, no list of files address has been previously set (refer to description of SETLOF macro).

† This macro is not available in SYSTEXT. The user must call common deck COMCCMD (refer to appendix A).

## SETLOF (062)

The SETLOF macro enables the user to specify a pointer to a list of files whose circular buffers will be flushed at job step abort or for terminal files, when the job is rolled out.

Macro Format:†

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | SETLOF | addr |

addr — Address containing the pointer to the list of files table

The format of location addr before issuing the SETLOF macro must be as follows:

| | 59–48 | 47–30 | 29–1 | 0 |
|---|---|---|---|---|
| addr | 0 | pointer | 0 | c |

pointer — Address of list of files table

c — Completion bit

The list of files table has the following format.

| | 59 | 58–18 | 17–0 |
|---|---|---|---|
| pointer +0 | m | 0 | len |
| +1 | $lfn_1$ | | $fet_1$ |
| +2 | $lfn_2$ | | $fet_2$ |
| ⋮ | ⋮ | | ⋮ |
| +n | $lfn_n$ | | $fet_n$ |
| | 0 | | |

m — Common memory manager (CMM) block indicator; setting this bit indicates that the file list is CMM controlled

len — Length of the list of files table

$lfn_i$ — File name

$fet_i$ — FET address

† This macro is not available in SYSTEXT. The user must call common deck COMCCMD (refer to appendix A).

The length of the list of files list must be less than the length of the FNT. If the list length exceeds the length of the FNT, the following warning message is issued when the system processes the list.

LIST OF FILES LENGTH TOO LONG.

When used with a time-sharing job step, the first file in the list must be the terminal output file.

During job step completion and after an error condition has occurred, the system uses the list of files to determine which files to complete for the user. If the file name is OUTPUT or the FET has the flush bit set and the flushing criteria are met (refer to the description of the flush bit, FET Description in section 3), the data in the circular buffer is written to the specified file.

# QUEUE FILE MANAGER 7

Queue file manager (QFM) performs functions associated with queue and dayfile protection. The majority of the QFM functions are for special system jobs (require SSJ= entry points; refer to appendix F). Thus, only those functions available to the typical user are discussed in this section. These include user requests to:

- Allow a job to be rerun or not rerun in the event of a system failure
- Release a file from a job to either the remote or local batch input queues
- Assign a file to a queue device

The following QFM functions require SSJ= entry points.

| Function Code | Description |
|---|---|
| 001 | Attach preserved file |
| 002 | Detach preserved file |
| 003 | Purge preserved file |
| 004 | Set IQFT file (preserved queue file) |
| 005 | Initialize IQFT file |
| 006 | Requeue FNT/FST list |
| 007 | Release FNT/FST list |
| 010 | Dequeue FNT/FST list |
| 011 | Attach queued file |
| 012 | Read system sector |
| 013 | Attach inactive queued file |
| 014 | Requeue inactive queued file |

To obtain documentation of these functions, enter the following control statements, after accessing the system OPL.

```
MODIFY(Z)/*EDIT,QFM
DOCMENT.
```

Enter the following control statement to obtain an entire listing of QFM.

```
MODIFY(Q,CL,Z)/*EDIT,QFM
```

Input/output queue protection identifies all I/O queue files and system dayfiles (system, account, and error log) as preserved files and retains them during system deadstart. Because queue protection can be inhibited at deadstart time, the user should contact installation personnel to determine if it is in effect.

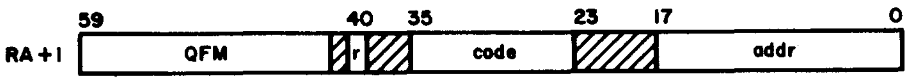
All requests to QFM must be made with the auto recall bit set; if it is not, the job is aborted and the following message is issued.

```
QFM ILLEGAL REQUEST.
```

If the parameter addresses are not within the user's field length, the job is aborted and the following message is issued.

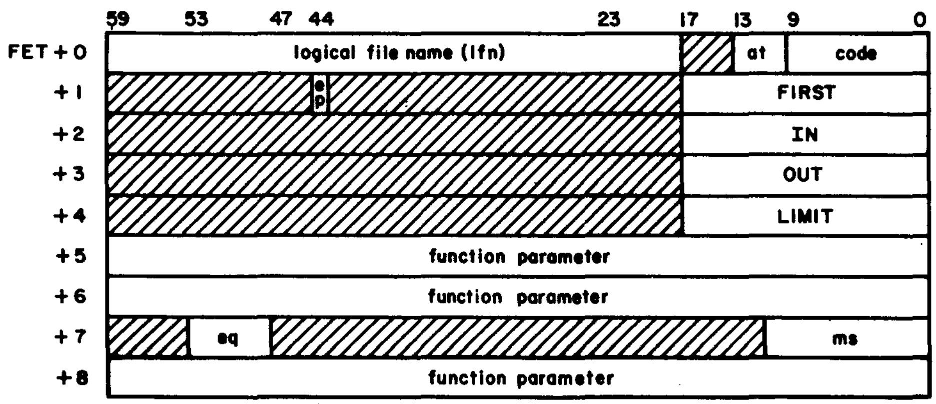
```
QFM ARGUMENT ERROR.
```

The format of the call to QFM is:




| | |
|---|---|
| r | Auto recall bit |
| code | Function code |
| addr | Address of the FET for the file |

The format of the FET used by QFM is:




| | |
|---|---|
| at | Abnormal termination code |
| code | Completion code |
| ep | Error processing bit |
| eq | Equipment number |
| ms | Mass storage error code |

## RERUN (015)

The RERUN macro sets the job rerun status, indicating that the job may be rerun in the event of system failure. This macro need not be used unless queue protection was disabled by a previous NORERUN macro, since all jobs in the input queue are initially given rerun status. The RERUN macro has no effect if used from a time-sharing origin job unless it is used in conjunction with the SUBMIT macro or SUBMIT control statement (refer to section 6, volume 1).

Macro Format:†

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | RERUN | |

†This macro is not available in SYSTEXT. The user must call common deck COMCMAC (refer to appendix A).

## NORERUN (016)

The NORERUN macro clears the job rerun status (initially enabled) and prevents a job from being rerun as the result of queue protection. With this macro the user can specify when a job may be safely rerun. For example, if a job has just updated a critical data base, it is probably desirable not to rerun the job. Hence, the user can use the NORERUN macro to prevent this. This macro affects input files only. The NORERUN macro has no effect if used from a time-sharing origin job unless it is used in conjunction with the SUBMIT macro or SUBMIT control statement (refer to section 6, volume 1).

Macro Format:†

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | NORERUN | |

## SUBMIT (017)

The SUBMIT macro enables the user to release a file from a job to either the remote or local batch input queues. The user must be validated to use this function (refer to the DB field of validation limits in section 6, volume 1), but does not need system origin privileges. This file must have the normal job file format. If the user wishes to reformat a file before submitting it for processing, he should use the SUBMIT control statement (refer to section 6, volume 1).

If a submitted job contains an illegal first USER statement, the job entering the SUBMIT function is aborted via a system error (no exit processing) and the following messages are issued to the user dayfile.

ILLEGAL USER CARD.
SYSTEM ABORT.

The following message is issued to the account dayfile.

SIUN, usernum.

Terminal users are immediately logged off with no dayfile message and without recovery.

† This macro is not available in SYSTEXT. The user must call common deck COMCMAC (refer to appendix A).

The occurrence of an illegal user statement also decrements the security count for the user number of the job that initiated the SUBMIT function. The security count is the number of security violations a user number has remaining before it is denied access to the system. If a user number's security count is exhausted, the following message is issued at job initiation.

ILLEGAL USER NUMBER - CONTACT SITE OPR.

The security count for the user number must be reset before access to the system is allowed for that user number.

Entry condition:

| | 59 ... 24 | 23 ... 12 | 11 ... 0 |
|---|---|---|---|
| FET+6 | 0 | id | ot |

A FET buffer of at least 1 PRU must be provided to allow QFM to read first sector for job control statement translation.

id — Identification code to be assigned to the job

ot — Origin type of the job to be submitted; must be local batch (BCOT,1) or remote batch (EIOT,2)

The job name given to the deferred job is returned in bits 59 through 18 of word 6 of the FET. This job name is derived from the user number specified on the user statement of the submitting job. This is the same as the method used for time-sharing and remote batch origin jobs (refer to section 3, volume 1).

This macro requires the common decks COMCQFM and COMCSYS.

Macro Format:†

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | SUBMIT | addr |

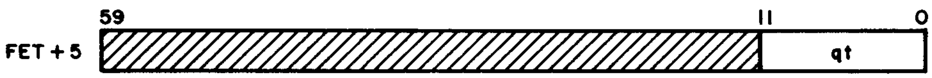
addr — Address of the FET that specifies the file to be submitted

† This macro is not available in SYSTEXT. The user must call common deck COMCMAC (refer to appendix A).

## ASSIGN FILE TO QUEUE DEVICE (020)

Function 020 enables the user to assign a file to a queue device (a device to which queue type files are assigned). A queue file of the specified type is assigned to the mass storage device specified in the mass storage allocation (MSAL) entry.

Entry condition:




qt Queue type. The qt field contains the queue-type value.

| Type | Value |
|---|---|
| INFT | 0 |
| PRFT | 2 |
| PHFT | 3 |

No macro is available for this function. The user should call it with the SYSTEM macro in the following manner.

```
SYSTEM    QFM,R,addr,2000B
```

| | |
|---|---|
| R | Auto recall bit (must be set) |
| addr | Address of the FET |
| 2000B | Function code (020) * $100_8$ |

If the MSAL control is set (refer to the NOS Operator's Guide for a discussion of MSAL), the specified file is assigned to the device. Otherwise, no action is taken.

## DSP FUNCTION

File routing under NOS is performed by the dispose processor (DSP). DSP is called by an RA+1 call or the ROUTE macro. Similar file routing capabilities are available with the SUBMIT and RELEASE macros, but ROUTE provides many more features.

The DSP function places a file in an input or output queue of either the central site or a remote batch site. The function can be issued from jobs of any job origin type.

**NOTE**

> If a file is routed to an input queue, it must contain a valid user statement or the job initiating the route will be aborted without exit processing. Refer to the SUBMIT macro, section 7, for further information concerning illegal user statements.

The format of the call to DSP is:

| | 59 | 40 | | 35 | 17 | 0 |
|---|---|---|---|---|---|---|
| RA+1 | DSP | r | | 0 | addr | |

| | |
|---|---|
| r | Auto recall bit (must be set) |
| addr | First word address of parameter block |

The user must define a parameter block as described below before issuing the DSP call or ROUTE macro.

| | 59 | 47 | 35 | 23 | | 17 | 11 | 0 |
|---|---|---|---|---|---|---|---|---|
| addr+0 | lfn | | | | | ec | | c |
| +1 | 0 | forms | disp | ex | ic | flags | | |
| +2 | reserved | | | TID | | | | |
| +3 | reserved | | | | b | priority | | |
| +4 | spacing | reserved | | | | rc | reserved | |

| | |
|---|---|
| lfn | Local file name of file to be routed (must be a print, punch, input, or local file; must not reside on a removable device; and must have read permission). |
| ec | Error code returned by system when bit 12 of flags field is set (error codes described later in this section). |

| | |
|---|---|
| c | Completion bit (must be zero when function is issued; system sets bit to one when operation complete). |
| forms | Forms code or input flags. Two display code characters or specific bits set identifying the forms to be used for this file or other processing options. If file is routed to input queue, this field is defined as follows. |

| Bits | Description |
|---|---|
| 47-46 | Unused |
| 45 | Do not protect input file |
| 44 | Reserved |
| 43 | Send file to input queue even if job statement error |
| 42-36 | Reserved |

Forms codes are two alphanumeric characters and are assigned by each installation. The user should contact installation personnel to determine what forms codes are available (if any).

| | |
|---|---|
| disp | Disposition code. Two alphanumeric characters specifying the disposition of the routed file. |

| Code | Description |
|---|---|
| IN | Release file to input queue |
| LP | Print on any line printer |
| LQ | Print on 512 line printer |
| LR | Print on 580-12 line printer |
| LS | Print on 580-16 line printer |
| LT | Print on 580-20 line printer |
| PB | Punch system binary |
| PH | Punch coded |
| PR | Same as LP |
| PU | Same as PH |
| P2 | Same as LQ |
| P8 | Punch 80 column binary |
| SB | Same as PB |
| SC | Rescind prior routing and change file type to local (LOFT) |

| | |
|---|---|
| ex | External characteristics (bits 23 through 21) code translated as follows. |

| Value | Print File | Punch File |
|---|---|---|
| 0 | (default) | (default) |
| 1 | Unused | SB |
| 2 | A4 | 80COL |
| 3 | B4 | Unused |
| 4 | B6 | O26 |
| 5 | A6 | O29 |
| 6,7 | Reserved | Reserved |

The mnemonics in the above table are defined as follows.

| Mnemonic | Description |
|---|---|
| A4 | ASCII 48-character set |
| A6 | ASCII 64-character set |

| Mnemonic | Description |
|---|---|
| B4 | Display code 48-character set |
| B6 | Display code 64-character set |
| O26 | Punch O26 mode |
| O29 | Punch O29 mode |
| SB | Punch system binary |
| 80COL | Punch 80-column binary |

ic — Internal characteristics (bits 19 through 18) code translated as follows:

| Value | Description |
|---|---|
| 0 | Display code |
| 1 | ASCII code |
| 2 | Binary |
| 3 | Reserved |

flags — Each bit set indicates that a parameter is specified.

| Bit | Description |
|---|---|
| 17 | File name assigned by system is returned to addr+0, bits 59 through 18 |
| 16 | Unused |
| 15 | Spacing code |
| 14 | Repeat count |
| 13 | Reserved |
| 12 | No dayfile message and return error code to addr+0, bits 17 through 12 |
| 11 | Reserved |
| 10 | Forms code |
| 9 | Priority |
| 8 | Internal characteristics |
| 7 | External characteristics |
| 6-5 | Reserved |
| 4 | Disposition code |
| 3 | Reserved |
| 2 | TID |
| 1 | Route to central site |
| 0 | End-of-job (deferred ROUTE) |

TID — For routing to an export/import queue, the TID should contain the complement of the address of a two-word block. The first word of the block contains the family name and the second word contains the user number (both left-justified and zero-filled). This is the user number that must be used to log in at a remote terminal to get the routed file.

For routing to the local batch queue, the TID contains an ID code (right-justified).

b — Must be set if priority specified

priority — If job priority is greater than 7760 octal, the specified priority is used for output files (otherwise field is ignored)

spacing — Spacing code for output files (580 print format control support)

rc — Repeat count

## ROUTE

The format of the ROUTE macro is as follows:

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | ROUTE | addr, r |

| | |
|---|---|
| addr | Address of parameter block |
| r | Auto-recall |

## ERROR PROCESSING

When an error occurs in processing a ROUTE macro or DSP call, either a dayfile message explaining the error is issued, or an error code is returned in bits 17 through 12 of addr+0. If the address of the parameter block is outside the field length of the job or if the completion bit is set when the function is issued, the job aborts. For all other errors, the function is not executed, but error processing continues. If bit 12 of the flags field is set, an error code is returned and no dayfile message is issued. If bit 12 is not set, a dayfile message is issued and no error code is returned.

When a diagnostic is issued for the ROUTE macro, the message

ERROR IN ROUTE FUNCTION LFN = lfn.

is issued, followed by the message describing the error.

The error codes that can be returned are as follows:

| Error Codes | Description |
|---|---|
| 1 | File name error |
| 2 | File not on mass storage |
| 3 | Illegal file type |
| 4 | Output file limit |
| 5 | Route to input not immediate |
| 6 | Immediate routing - no file |
| 7 | Invalid disposition code |
| 10 | Reserved |
| 11 | Reserved |
| 12 | Illegal request (unconditional abort) |
| 13 | Reserved |
| 14 | Reserved |
| 15 | Reserved |
| 16 | Cannot route job input file |
| 17 | Completion bit already set (unconditional abort) |
| 20 | File on removable device |
| 21 | Invalid TID |
| 22 | Forms code not alphanumeric |
| 23 | Reserved |
| 24 | Reserved |
| 25 | Reserved |
| 26 | This routing not allowed |

| Error Codes | Description |
|---|---|
| 27 | FNT/device full |
| 30 | Local file limit |
| 31 | I/O sequence error (unconditional abort) |
| 32 | Job statement error |
| 33 | Too many deferred batch jobs |
| 34 | Illegal user statement |
| 35 | Device unavailable |
| 36 | Illegal file mode |
| 37 | Invalid external characteristics |
| 40 | Illegal origin type |

For a complete listing of error messages, refer to the NOS Reference Manual, volume 1, appendix B.

# SYSTEM FILE MANAGER 9

System file manager (SFM) performs functions for CPU programs that access files that are not accessible using the CIO and LFM function processors. Most SFM functions require that the job be of system origin or that the user be validated to have system origin privileges. Several SFM functions require that the job be a special system job (the program must have an SSJ= entry point; refer to appendix F). These are:

| Function Code | Description |
|---|---|
| 000 | Terminate active dayfile |
| 007 | Protect active dayfile |
| 010 | Clear dayfile byte |
| 011 | Enter local fast attach file |
| 012 | Delete fast attach file (local and global) |
| 014 | Attach inactive dayfile |
| 015 | Enter global fast attach file |

SFM functions requiring SSJ= entry points are not discussed in this section since they are not available to the typical user. To obtain documentation of these functions, enter the following control statements after accessing the system OPL.

```
MODIFY(Z)/*EDIT,SFM
DOCMENT.
```

Enter the following control statements to obtain a complete listing of SFM.

```
MODIFY(Q,CL,Z)/*EDIT,SFM
```

All requests to SFM must be made with the auto recall bit set; if it is not, the job is aborted and the following message is issued.

SFM ILLEGAL REQUEST.

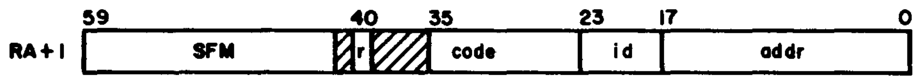
If the parameter addresses are not within the user's field length, the job is aborted and the following message is issued.

SFM ARGUMENT ERROR.

Errors encountered by SFM cause the job to abort; no user error processing is provided. Those operations that use central memory buffers do not treat the buffers as circular buffers, storing at IN and reading from OUT. All data is read from or entered at FIRST.

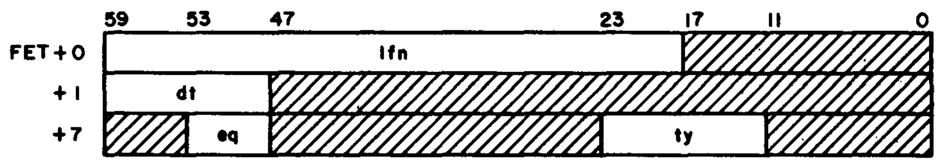
Common decks required by all SFM macros are COMCSFM (relocatable and absolute assemblies) and COMCSYS (absolute assembly).

The format of the call to SFM is:




| | |
|---|---|
| r | Auto recall bit |
| code | Function code |
| id | File identification number |
| addr | Address of the FET for the file |

The format of the FET used by SFM is:




| | |
|---|---|
| lfn | File name |
| dt | Device type (refer to STATUS macro, section 4) |
| eq | Equipment number |
| ty | Dayfile type: |

1 System dayfile
2 Account dayfile
3 Error log dayfile

## DAYFILE (001, 002, 003, 005)

The DAYFILE macro enables the user to access the dayfiles available in the system. This is useful for on-line accounting programs and dayfile dumps. A FET of at least seven words must be specified for this function.

A portion of the dayfile specified resides in the system central memory buffers. This is transferred to the buffer in the user's field length specified in the FET (FIRST). The user must ensure that the buffer specified is large enough to accommodate the central memory portion of dayfile. Since buffer sizes can vary (deadstart parameter), the user should check with installation personnel for the buffer size of the desired dayfile. If the buffer is too small, the following message is issued.

SFM ARGUMENT ERROR.

That portion of the dayfile that resides on mass storage is made available to the user as a library file attached to the control point in read-only mode. If the file name specified already exists, it is returned. The file is positioned at BOI. If the dayfile specified does not exist, the following message is issued.

SFM ILLEGAL REQUEST.

The user must be validated for system origin privileges or the job must be a system origin job for all dayfile types except USER (SFM function 005).

The track interlock is left set for all dayfile types except USER. The track interlock is cleared when the dayfile is returned to the system.

Macro Format:†

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | DAYFILE | lfn, type |

| | |
|---|---|
| lfn | Address of the FET for the call; name given to the mass storage portion of the file |
| type | DAYFILE (function 001); attach system dayfile |
| | ACCOUNT (function 002); attach account dayfile |
| | ERRLOG (function 003); attach error log dayfile |
| | USER (function 005); attach user's dayfile. This is the dayfile of the job currently running. |

The reply to this macro for functions 001, 002, and 003 is:

| | 59 … 0 |
|---|---|
| FET+5 | recovered dayfile length |
| +6 | current dayfile length |

†This macro is available in COMCMAC (refer to appendix A).

For examples of the use of these macros, refer to a listing of DAYFILE.

## ESYF (004)

The ESYF macro enables the user to enter a file attached to his control point as a system file. This is a read-only file that contains special system information. The job must be of system origin or the user must be validated for system origin privileges to issue this macro. If the file specified already exists as a system file, the old file is released and the new file is entered.

Macro Format:†

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | ESYF | addr |

addr Address of the FET for the request to SFM

## RDVT (006)

The RDVT macro allows the user to determine the type of device on which the specified file resides.

Macro Format: †

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | RDVT | addr, dt |

addr Address of the FET for the file

dt The device number; if the dt parameter is set in word 1 of the FET, this parameter is not required.

† This macro is available only in COMCMAC (refer to appendix A). For system origin jobs or users with system origin privileges.

## TRANSLATE CONTROL STATEMENT

Translate control statement (TCS) processes user requests to read a control statement from or place a control statement in the control statement stream. The only common deck required for absolute assembly is COMCSYS.

The format of the RA+1 call is:

| | 59 | 40 | | 35 | 23 | 17 | 0 |
|---|---|---|---|---|---|---|---|
| RA+1 | TCS | | r | code | sf | addr | |

r — Auto recall bit

code — Function code:

| Code | Macro |
|---|---|
| 004 | CONTROL |
| 005 | EXCST |

sf — Subfunction code for the CONTROL macro. This field is not used for the EXCST macro.

| sf | Action |
|---|---|
| 00 | Read the next control statement and advance the control statement pointer. |
| 01 | Read the next control statement only if it is not a local file call. Do not advance the control statement pointer. |
| 02 | Read the next control statement even if it is a local file call. If the statement is a local file call, set bit 17 of $RA+64_8$. |
| 4x | If the system determines that the next control statement must be processed in product set format, sf is set to 4x, where x can be 0, 1, or 2, corresponding to one of the above options. If bit 23 of RA+1 is not set, control statement parameters are to be processed in NOS format. |

addr — First word address of the buffer in which the control statement is to be stored or from which the control statement is to be read.

### CONTROL (004)

The CONTROL macro allows the user to read the next control statement in the control statement stream and transfer it to the address specified. The control statement is checked for syntax errors, and all parameters are stored as if a program load had actually taken place. (Refer to section 5, volume 1 for a description of the manner in which parameters are stored.) If no control statement exists or if the next control statement is a comment line of the form:

    *comments

a zero statement is stored.

Macro Format:

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | CONTROL | addr, rss, lf, nbf |

addr — First word address of the buffer in which the next control statement is to be stored. The user should allow room for $10_8$ words (80 characters). If addr+7 is not less than the job's field length, the following message is issued to the user's dayfile.

BUFFER ARG. ERROR.

rss — If rss is specified (any value may be used), the control statement pointer is not advanced. This allows the user to determine what the next statement is and still allow it to be processed. If rss is not specified, the control statement pointer is advanced as if the statement had been processed.

lf — If lf is specified (any value may be used), the next control statement is read even if the statement calls for the execution of a local file; the control statement pointer is not advanced. If lf is specified and the call is for a local file, the system sets bit 17 in RA+$64_8$ (ACTR). If lf is not specified, the next control statement is read only if it is not a local file call. The rss parameter is required if the lf parameter is specified.

nbf — If nbf is specified, parameters are unpacked in NOS/BE format.

**EXCST (005)**

With the EXCST macro the user specifies a buffer containing a control statement. Control is transferred from the calling program to the system, which reads the control statement, places it in the control statement stream, and processes it. Control is not returned to the calling program. The control statement must conform to NOS control statement format conventions described in section 5, volume 1.

Macro Format: †

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | EXCST | addr |

addr — First word address of the buffer containing the control statement to be processed. The system begins reading at addr and continues until the end of statement (zero byte) or end of buffer (80 characters) is reached. The control statement must be left-justified with zero fill. If the buffer extends past the job's field length, the following message is issued to the user's dayfile.

BUFFER ARG. ERROR.

† This macro is not available in SYSTEXT. The user must call the common deck COMCMAC (refer to appendix A).

## CHECKPOINT/RESTART

A job may be terminated at any time as the result of system, operator, or programmer error. For some jobs it becomes more advantageous to accept the overhead of checkpoint procedures than to run the risk of losing the entire job output. The checkpoint/restart feature is implemented through the CKP control statement or CHECKPT macro and the RESTART control statement. Refer to section 12, volume 1, for discussions of the CKP and RESTART control statements.

### CHECKPT

The CHECKPT macro is used for taking checkpoint dumps. The dump is written on the tape or mass storage checkpoint file specified on a REQUEST, ASSIGN, or LABEL control statement or REQUEST or LABEL macro. For a general description of checkpoint dumps, refer to section 12, volume 1. The CHECKPT macro provides the user greater control than the CKP control statement in specifying the type of copy to be performed.

Macro Format:†

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | CHECKPT | param, sp |

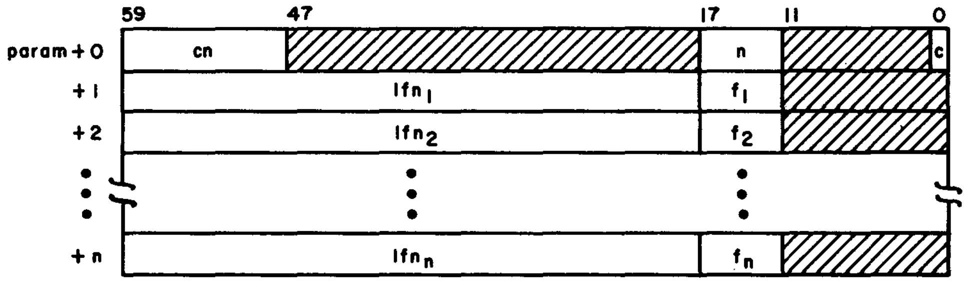
param — Address of the parameter list identifying the files to be checkpointed

sp — Flag indicating whether or not all files assigned to the job are to be checkpointed:

- sp = 0 — All files are to be checkpointed.
- sp ≠ 0 — The files identified in the parameter list are to be checkpointed.

†This macro is not available in SYSTEXT. The user must call common deck COMCMAC (refer to appendix A).

Parameter List Format:




cn — Latest checkpoint number

n — Octal number of entries in the parameter list: $(0 \leq n \leq 77_8)$

c — Completion bit; set by CHECKPT when the checkpoint is complete

$lfn_i$ — Identifies file to be checkpointed; $lfn_i$ is left-justified

$f_i$ — Specifies the manner in which $lfn_i$ is to be copied:

$f_i$=0 — The file is copied from the BOI to its position at checkpoint; only that portion is available for restart. RESTART positions the file at the latter point.

$f_i$=1 — The file is copied from its position at checkpoint to the EOI; only that portion is available for restart. RESTART positions the file at the former point.

$f_i$=2 — The entire file is copied. RESTART sets the file to its position at checkpoint time.

$f_i$=3 — The last operation on the file determines how the file is copied.

| Last Operation | f Selected |
|---|---|
| Write | $f_i$ = 0 |
| Read (EOI detected) | No copy |
| Read (EOI not detected) | $f_i$ = 2 |

$f_i$=4 — The information table associated with the file is copied but the file itself is not copied. The information table contains FNT/FST information and the random address of the file. RESTART retrieves the file and sets it to its position at checkpoint time. If $f_i$=4 and the file is a mass storage file, RESTART assumes it is an indirect access file and issues a GET macro to obtain a working copy.

The following list shows the type of operation CHECKPT performs as sp and n vary.

| sp | n | Operation |
|---|---|---|
| sp=0 | n=0 | All files assigned to the job at checkpoint time are copied according to the last operation performed. |
| sp≠0 | n=0 | All files assigned to the job at checkpoint time are copied according to the last operation performed. |
| sp=0 | n≠0 | All files assigned to the job at checkpoint time are copied. The n files included in the parameter list are copied according to their respective f values. All other files are copied according to the last operation performed. |
| sp≠0 | n≠0 | The n files specified in the parameter list are copied according to their respective f values. |

The INPUT, OUTPUT, PUNCH, PUNCHB, and LGO files are always checkpointed; they are copied according to the last operation performed (refer to $f_i$=3), regardless of the sp and n values.

For all other files except direct access files, the default copy type is f=4 when n≠0. For direct access files, the type of copy CHECKPT makes depends on the access mode.

| | Mode | User Option | Default |
|---|---|---|---|
| W | Write | Any type of copy† | Copied (f=3) |
| R | Read-only | Any type of copy | Not copied (f=4) |
| E | Execute-only | Only f=4 | Not copied (f=4) |
| A | Append-only | Any type of copy† | Copied (f=3) |
| M | Modify | Any type of copy† | Copied (f=3) |
| RA | Read and append | Any type of copy† | Copied (f=3) |
| RM | Read and modify | Any type of copy† | Copied (f=3) |

For a random file the copy type must be f=2 or it will be copied according to the last operation performed (f=3).

†If f=4 is selected, the user must retrieve the file himself at restart time and select the NA and FC options on the RESTART control statement (refer to section 12, volume 1).

## SYSTEM REQUESTS

The following requests enable the user to perform miscellaneous tasks associated with his job. Most of the requests are processed by the system monitor directly rather than by a specific function processor. The calling format is shown for each macro. Unless otherwise noted, the only common deck required is COMCSYS.

### ABORT

The user can request the monitor to set the error flag at the control point when a program error occurs by using the ABORT function. If the control statement record of the job deck contains an EXIT statement, the system continues job processing with the control statement that immediately follows the EXIT statement.

The format of the RA+1 call for this function is:

| | 59 | 35 0 |
|---|---|---|
| RA+1 | ABTP | |

Macro Format:

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | ABORT | |

### CLOCK

In response to the CLOCK function, the system returns the current time of day in display code in location stat.

| | 59 | 53 | 41 | 35 | 23 | 17 | 5 0 |
|---|---|---|---|---|---|---|---|
| stat | Δ | hh | . | mm | . | ss | . |

The format of the RA+1 call for this function is:

| | 59 | 35 | 23 | 17 0 |
|---|---|---|---|---|
| RA+1 | TIMP | 0002 | 0 | stat |

Macro Format:

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | CLOCK | stat |

stat Address to receive the clock reading

**DATE**

In response to the DATE function, the system returns the current date in display code format in location stat.

| | 59 | 53 | 41 | 35 | 23 | 17 | 5 0 |
|---|---|---|---|---|---|---|---|
| stat | Δ | yy | / | mm | / | dd | . |

The RA+1 call for this function is:

| | 59 | 35 | 23 | 17 0 |
|---|---|---|---|---|
| RA+1 | TIMP | 0001 | 0 | stat |

Macro Format:

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | DATE | stat |

stat Address to receive the date

**EDATE**

EDATE takes the packed date and converts it to display code.

| | 59 | 17 | 11 | 5 0 |
|---|---|---|---|---|
| d | 0 | yy | mm | dd |

yy Year minus 1970

mm Month

dd Day

Upon completion, X6 contains the following.

| | 59 | 53 | 41 | 35 | 23 | 17 | 5 0 |
|---|---|---|---|---|---|---|---|
| (X6) | Δ | yy | / | mm | / | dd | . |

Macro Format:†

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | EDATE | d |

d Packed date to be converted

This macro requires the common deck COMCEDT.

**ENDRUN**

The ENDRUN function requests normal termination of a program. NOS examines the control statement record of the job deck and begins execution with the next unused control statement. If there are no more control statements or if the next statement is an EXIT statement, the system terminates the job.

The format of the RA+1 call for this function is:

| | 59 | 35 0 |
|---|---|---|
| RA+1 | ENDP | 0 |

Macro Format:

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | ENDRUN | |

† This macro is available in COMCMAC (refer to appendix A).

## ETIME

ETIME takes the packed time and converts it to display code.

| | 59 … 18 | 17 … 12 | 11 … 6 | 5 … 0 |
|---|---|---|---|---|
| ptime | ////////// | hh | mm | ss |

| | |
|---|---|
| hh | Hours |
| mm | Minutes |
| ss | Seconds |

Upon completion, X6 contains the following.

| | 59 … 54 | 53 … 42 | 41 … 36 | 35 … 24 | 23 … 18 | 17 … 6 | 5 … 0 |
|---|---|---|---|---|---|---|---|
| (X6) | Δ | hh | . | mm | . | ss | . |

Macro Format:†

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | ETIME | ptime |

| | |
|---|---|
| ptime | Packed time to be converted |

This macro requires the common deck COMCEDT.

## JDATE

JDATE returns the current Julian date in stat. The format of the RA+1 call for this function is:

| | 59 … 36 | 35 … 24 | 23 … 18 | 17 … 0 |
|---|---|---|---|---|
| RA+1 | TIMP | 0003 | 0 | stat |

† This macro is available in COMCMAC (refer to appendix A).

| | 59 | 29 | 17 0 |
|---|---|---|---|
| stat | 0 | yy | ddd |

yy Julian year

ddd Julian day

Macro Format:

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | JDATE | stat |

stat Address which receives the Julian date

## MEMORY

Using the MEMORY function, the user can determine or change the amount of central memory assigned to his control point or determine the maximum amount of memory he can request.

The format of the RA+1 call for this function is:

| | 59 | 40 | | 35 | 17 0 |
|---|---|---|---|---|---|
| RA+1 | RFL | r | | stat | words |

Alternate call format: †

| | 59 | 40 | | 17 0 |
|---|---|---|---|---|
| RA+1 | MEM | r | 0 | stat |

Macro Format:

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | MEMORY | type, stat, r, words, na |

type CM or null for central memory; either is legal. ECS for extended core storage. ††

stat Status word address. May be omitted only if the words parameter is specified and na is null.

† To maintain compatibility with NOS/BE.
†† Not currently implemented. ECS size is zero and cannot be changed.

| | | | | |
|---|---|---|---|---|
| | 59 | 29 | 3 | 0 |
| stat | val | 0 | rs | c |

r — If specified, control is not returned until the request is complete. If omitted, control may be returned before the request is complete. In this case, bit 0 of the stat word is set to indicate completion.

words — Desired new field length which, if specified, overrides the stat word. If words is specified, the MEMORY macro sets the upper 30 bits of the stat word to the value of words. A negative value may not be used for words.

na — If null, the program is aborted if the user's request exceeds the current maximum which he is allowed. If this parameter is specified, the program is not aborted if the request exceeds the current maximum. Instead, the macro does not change the field length, and sets the stat word as defined below.

The stat word (if used) has the following format.

val — Prior to the macro call, val is used to specify the desired new field length (words parameter can override this value). If val (or words) is +0, then the current field length for the specified type is returned in this field. If bit 47 of val is set and specified type is CM, a memory reduction is honored even if no reduce has been selected (that is, no reduce override is in effect).† If val is -1, then the current maximum field length for the specified type is returned. If val (or words) is greater than zero for the macro call, it contains the actual value assigned upon return. If val is -0 and type is ECS, the field length is set to 0.

rs — These bits are reserved for system usage.

c — Completion bit. The system sets this bit when the request is complete.

If a request is given for an amount greater than the current maximum (refer to the Installation Handbook for a description of the current maximum field length) and if na is specified, then no field length change occurs and control is returned with val set to the current field length value.

If the reserved bits (rs) in the status word are used, the MEMORY request or a subsequent MEMORY request may be aborted with the following dayfile message.

ILLEGAL COMMON MEMORY MANAGER REQUEST.

†Refer to the REDUCE (-) control statement in the CYBER Loader Reference Manual.

## MESSAGE

The MESSAGE function allows the user to display a message on the system console display and enter it in a dayfile.

If the job is of system origin, the message may be flashed on the B display by including a dollar sign as the first character of the message.

The maximum length of a message is 80 characters; up to 40 characters per line are displayed. The message ends with either the first word containing 12 bits of zeros in any byte or at the eightieth character. Before issuing the MESSAGE function, the user must pack the display coded message in sequential locations, beginning at location addr.

The format of the RA+1 call for this function is:

| | 59 | 40 | | | 35 | 23 | 17 | 0 |
|---|---|---|---|---|---|---|---|---|
| RA + 1 | MSG | | r | 0 | x | 0 | addr | |

Macro Format:

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | MESSAGE | addr, x, r |

addr — Beginning address of the message. If the upper 12 bits of the location specified by this address are zero, then the next 18 bits (bits 47 through 30) of this location are assumed to contain the beginning address of the message.

x — Message routing option

- x=0 — Message is placed in the system dayfile, the user dayfile, and displayed at line 1 of the control point
- x=1 — Message is displayed at line 1 of the control point
- x=2 — Message is displayed at line 2 of the control point
- x=3 — Message is placed in the user dayfile and displayed at line 1 of the control point
- x=4 — Message is placed in the error log dayfile if job is special system job (that is, has an SSJ= entry point) or is of system origin; or user dayfile if not SSJ= or system origin
- x=5 — Message is placed in the account dayfile if job is special system job or is of system origin; or user dayfile if not SSJ= or system origin
- x=6 — Message is placed in the system dayfile, the user dayfile, and displayed at line 1 of the control point†
- x=7 — Message is placed in the user dayfile and displayed at line 1 of the control point†

† Provided for compatibility with NOS/BE.

If x is not specified or is an illegal value, x=0 is assumed. If x is not defined, x=1 is assumed. If x is the character string LOCAL, x=3 is used.

The control point message areas (lines 1 and 2) provide the user with the ability to display concurrently messages that enter the dayfile and those that require operator action. Line 2 is normally used to display information about the current status of the executing program; for example,

SKIPPING lfn

COPYING lfn

ASSEMBLING TEST

⋮

Only messages that are of interest to other than the job, such as the control statements processed and compilers used, should be placed in the system dayfile (x=0). All messages of interest to the job, such as the path taken by the programs and the number of records copied, should be placed only in the user dayfile (x=3). All messages placed in the user dayfile (x=0 and x=3) are counted by the system. If the number of messages issued by the job exceeds the limit for which the user is validated, the following error message is issued to the user dayfile and the job is aborted.

MESSAGE LIMIT.

r — If r is specified, control is not returned until the operation is complete.

## MOVE

The MOVE macro moves a block of data from $addr_1$ to $addr_2$. This macro requires the common deck COMCMVE for absolute assemblies.

Macro Format:

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | MOVE | count, $addr_1$, $addr_2$ |

| | |
|---|---|
| count | Number of words in the block to be moved |
| $addr_1$ | Address of the first word of the block to be moved |
| $addr_2$ | Address of the first word of the destination |

MOVE allows overlap in data moves; in other words, $addr_2$ can be less than $addr_1$ plus count.

## PDATE

PDATE returns the current date and time in binary packed format. The user can unpack the parameters or use the EDATE and ETIME macros to do the unpacking.

The format of the RA+1 call for this function is:

| | 59 | 35 | 23 | 17 0 |
|---|---|---|---|---|
| RA+1 | TIMP | 0006 | 0 | stat |

| | 59 | 35 | 29 | 23 | 17 | 11 | 5 0 |
|---|---|---|---|---|---|---|---|
| stat | 0 | yy | mo | dd | hh | mm | ss |

| | |
|---|---|
| yy | Year minus 1970 |
| mo | Month |
| dd | Day |
| hh | Hours |
| mm | Minutes |
| ss | Seconds |

Macro Format:

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | PDATE | stat |

stat    Address to receive the packed date

## RECALL

The RECALL function enables the user to relinquish the CPU until a function is completed or the CPU recall time has elapsed (delay time is set by the installation, usually 1/2 second). If the stat parameter is included in the call, control is not returned to the program until bit 0 of the word specified by stat is set. If stat is not included in the macro call, the program relinquishes the CPU only until the next pass through the recall loop.

The format of the RA+1 call for this function is:

| | 59 | 40 | | 17 | 0 |
|---|---|---|---|---|---|
| RA+ 1 | RCL | r | 0 | stat | |

Macro Format:

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | RECALL | stat |

stat    If this parameter is present, control is returned to the program when bit 0 of the word specified by the address stat is set.

## RTIME

In response to the RTIME function, the system returns the real-time clock reading in location stat. This is the elapsed time since deadstart.

| | 59 | 35 | 0 |
|---|---|---|---|
| stat | seconds | milliseconds | |

The format of the RA+1 call for this function is:

| | 59 | 35 | 23 | 17 0 |
|---|---|---|---|---|
| RA+1 | TIMP | 0005 | 0 | stat |

Macro Format:

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | RTIME | stat |

stat  Address to receive the clock reading

**STIME**

With the STIME macro, the user can determine his accumulated system resource units (SRU). Refer to the NOS Installation Handbook for a description of SRUs.

Macro Format:

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | STIME | stat |

stat  Address to receive SRU value

| | 59 | 35 0 |
|---|---|---|
| stat | 0 | milli units |

milliunits  Accumulated system resource units/1000

## SUBR

The subroutine macro enables the user to distinguish between entering a subroutine and exiting from a subroutine even though control is transferred to the same address. Transfers to the subroutine are of the form:

RJ tag

Exits from the subroutine are of the form:

EQ tagX

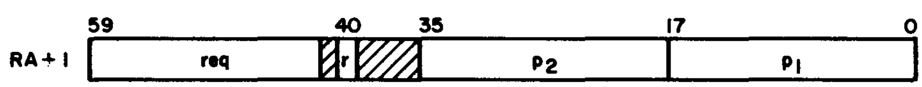
Macro Format:†

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| tag | SUBR | |

## SYSTEM

With the SYSTEM function, the user can request the system to process any three-character requests. This can be either the functions that MTR performs, such as TIM, or a PPU program, such as CIO (refer to File Environment Table, section 3, for a list of request processors). The SYSTEM function destroys the contents of X2. A PPU program can be called from a CPU program if the first character of the name is alphabetic. Documentation of these programs (refer to DOCMENT statement, section 7, volume 1) should be consulted for the functions available (for example, LFM, CPM). These should be used when macros do not exist to issue the functions desired.

The format of the RA+1 call is:




† This macro is available in COMCMAC (refer to appendix A).

Macro Format:

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | SYSTEM | req, r, $p_1$, $p_2$ |

| | |
|---|---|
| req | Three-character system request |
| r | If specified, control is returned only after the request is completed. |
| $p_1$ | Bits 17 through 0 of the request |
| $p_2$ | Bits 35 through 18 of the request |

Example 1:

If the user wishes to dump the contents of locations 1000 to 4000 and recall the CPU when the dump is completed, the SYSTEM request is:

```
SYSTEM DMP, R, 4000B, 1000B
```

Example 2:

If the user wishes to dump the display code equivalent of the data dumped in example 1, the SYSTEM request is:

```
SYSTEM DMD, R, 4000B, 1000B
```

Example 3:

If the user wishes to issue a CIO function request, he can do so with the following SYSTEM request:

```
SYSTEM CIO, r, n, addr
```

where addr is the address of the FET for the file being read and n is the count for skip operations. If r is specified, the request is made with auto recall. When performing a CIO request in this manner, the user must set the function code in FET+0.

## TIME

The TIME function returns the accumulated central processor time used by the job in location stat.

| | 59 | 47 | 35 | 11 0 |
|---|---|---|---|---|
| stat | 2000 | 0000 | seconds | milliseconds |

The format of the RA+1 call for this function is:

| | 59 | 35 | 17 0 |
|---|---|---|---|
| RA+1 | TIMP | 0 | stat |

Macro Format:

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | TIME | stat |

stat   Address to receive the CPU time

## LOADER REQUESTS

The system provides routines to aid the user in loading overlays or capsules at specific points during program execution. The overlays or capsules can reside on files attached to the user's job or in system libraries.

### OVERLAY

The OVERLAY macro processes a system request to the LDR processor. LDR provides the ability to load overlays to specified areas of the user's program area. Depending on the parameters specified and the level of the overlay, control may or may not be returned to the calling program.

The format of the RA+1 call for this function is:†

| | 59 | 40 | | 17 | 0 |
|---|---|---|---|---|---|
| RA+1 | LDR | | r | 0 | addr |

r   Auto recall, if desired

addr   Address of the request

The load request consists of two to four words. The two- to four-word block must be defined by the user for RA+1 calls, but is defined by the system when called from the OVERLAY macro (only two words used).

† LDV is processed by the system as the LDR call. This is to provide compatibility with the common product set.

<table>
<tr><td></td><td>59</td><td>53</td><td>47</td><td></td><td>40</td><td></td><td></td><td></td><td>35</td><td>17</td><td>0</td></tr>
<tr><td>addr</td><td colspan="9">name</td><td colspan="2">0</td></tr>
<tr><td></td><td>$\ell_1$</td><td>$\ell_2$</td><td>n</td><td>0</td><td>u</td><td>v</td><td>0</td><td>e</td><td>lwa</td><td colspan="2">fwa</td></tr>
<tr><td></td><td colspan="9">ovlname</td><td colspan="2">0</td></tr>
<tr><td></td><td colspan="9">eptname</td><td colspan="2">0</td></tr>
</table>

| | |
|---|---|
| name | Source of name depending on the u and n options |
| $\ell_1$ | First overlay level |
| $\ell_2$ | Second overlay level |
| n | Number of words in request-2 (bits 47-46) |
| u | Load option (bit 42) |
| v | Overlay flag (must be set to 1) (bit 41) |
| e | Call completion flag (bit 36) |
| lwa | Last word address available for load |
| fwa | First word address of the overlay |
| ovlname | Name of overlay to be loaded (if n≠0) |
| eptname | Entry point name when loading multiple entry point overlay (if n=2) |

The operation performed depends on the value of u, n, fwa, $\ell_1$, and $\ell_2$.

1. If u=0, n is ignored and name is the name of the file containing the overlay ($\ell_1$ and $\ell_2$ are required).
2. If u=1 and n=0, name is the name of the overlay from the system library ($\ell_1$ and $\ell_2$ are ignored).
3. If u=1 and n≠0, ovlname is the name of the overlay from the system library ($\ell_1$ and $\ell_2$ are ignored).
4. If fwa=0, the overlay is loaded at the address specified by the overlay.
5. If $\ell_1=\ell_2=0$, the (0, 0 overlay) control is returned to the called overlay; otherwise, control is returned to the caller with fwa=entry address.
6. If e=1, control transfers to the specified entry point, eptname, in the overlay.

Upon completion of the load, information is returned to addr as follows:

| | 59 | 53 | 47 | 17 0 |
|---|---|---|---|---|
| addr | name | | | 0 |
| | $\ell_1$ | $\ell_2$ | 0 | eptaddr |
| | ovlname | | | 0 |
| | eptname | | | 0 |

eptaddr Entry point address of the overlay; if n=2, eptaddr is the address of eptname.

Macro Format:

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | OVERLAY | nam, lev, SYSTEM, fwa |

nam Address of file name in L format (display code, left-justified)

lev Level of overlay. If not specified, level 0, 0 is assumed and control is automatically transferred to transfer address encountered on overlay load. (Usually specified on IDENT instruction of ABS programs.) For overlay level (i, j), level is defined as: $lev=i*100_8+j$.

SYSTEM If SYSTEM is specified, file is loaded from system library and nam is name of overlay desired.

fwa If this parameter is specified, fwa is the address where the overlay is to be loaded. The file is loaded at the address specified on the overlay if this parameter is not specified.

Common decks required are COMCOVL and COMCSYS.

After the macro is processed, X1 contains the address of the entry in the overlay.

Example 1:

```
            .
            .
            .
            OVERLAY TEST
            .
            .
TEST        CON       4LTEST
```

This sequence of code loads overlay 00 from file name TEST and begins execution at the entry address specified by TEST.

Example 2:

```
            OVERLAY A,0100B,S,LDA
            SB2       X1
            JP        B2
            .
            .
A           CON       5LARITH
            .
            .
LDA         EQU       *
```

This sequence of code loads the ARITH routine from the system (level 1, 0) library at address LDA and transfers control to the routine.

If the request calls for the overlay to be loaded and executed and no fwa load address is specified in the request, the entry point address specified in the load table is used as the entry point address. If an fwa load address is specified in the request, bias the entry point address from the load table with the difference of the fwa load address specified by the request and the fwa load address specified by the load table; that is:

Entry point address = (entry point address from load table) + (fwa load address specified in the request) - (fwa load address from load table)

If the request requires just an overlay load, the entry point address returned is the entry point address in the load table.

## LOADD

The LOADD macro allows the user to locate fast dynamic load (FDL) capsules and have pertinent information returned to a specified address. Fast dynamic loading is a method of loading preprocessed binary routines.

The format of the RA+1 call for this function is:

| | 59 | 40 | | 17 0 |
|---|---|---|---|---|
| RA+1 | LDD | r | 0 | addr |

| | |
|---|---|
| r | Auto recall bit |
| addr | Address of parameter block |

The parameter block consists of two words in the following format.

| | 59 | 47 | 29 | 17 | 8 0 |
|---|---|---|---|---|---|
| addr + 0 | group name | | | stat | fc |
| +1 | 0 | liblist | dirlen | directory | |

group name — Name of the group of capsules for which a directory is requested

stat — Status of call (ignored during request). Upon completion of call, stat is set to one of the following values.

| Value | Description |
|---|---|
| 0 | Function completed without error |
| 1 | Illegal function code |
| 2 | Bad directory address or length (address plus length must be less than or equal to field length) |
| 3 | Bad liblist address or length |

The following errors may be combined with those above.

| | |
|---|---|
| $10_8$ | An entry in the library list did not correspond to any known local or system library name, or an entry specified the name of a file which was not a mass storage library |
| $20_8$ | The specified directory space was not large enough to contain the entire directory |

fc — Function code (must be zero during request). LDD sets bit zero to one when the request is complete.

liblist — Address of a list of libraries to be searched after the global library set. Zero if no set is specified.

| | |
|---|---|
| dirlen | Length, in central memory words, of the area to receive the generated directory. Upon completion of the call, it is set to the actual length that was needed for the complete directory (may be less than or equal to the value of the original call). |
| directory | Address of the area to receive the generated directory. |

When called, LDD searches first the global library set (refer to the CDC CYBER Loader Reference Manual for a description of global library sets) and then the library set specified in the call. If a library file is found to contain one or more capsules belonging to the given group, an entry is made in the directory. This entry is one of two different forms, depending on whether the library is a system library or a local file library.

For a local file library the format is:

| 59 | | 17 0 |
|---|---|---|
| l | local file name † | 0 |

For a system library the format is:

| 59 | 47 | 23 | 17 0 |
|---|---|---|---|
| 7777 | 0 | libord | 0 |

| | |
|---|---|
| libord | The library ordinal of the library containing the capsule |

For each capsule found that belongs to the given group, LDD makes the following entry in the directory.

| 59 | 56 | 35 | | 17 0 |
|---|---|---|---|---|
| capsule name | | | | faddr |
| | r | caddr | daddr | length |

| | |
|---|---|
| faddr | Address, relative to the beginning of the directory, of the word containing the file entry associated with this capsule. |
| r | Residence of capsule:<br>0 Mass storage<br>1 Mass storage and CM<br>2 Mass storage and ECS |
| caddr | CM or ECS address of capsule |
| daddr | Disk address (relative PRU) of capsule |

† The first character of the local file name cannot be numeric.

| | |
|---|---|
| length | Length of the capsule, including header, code image, and relocation and linking information, but excluding the prefix table. |

Macro Format:†

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | LOADD | addr,r |

| | |
|---|---|
| addr | Address of parameter block |
| r | Auto recall |

## LOADQ

The LOADQ macro loads fast dynamic load (FDL) capsules or overlays from specified files.

The format of the RA+1 call for this function is:

| | 59 | 40 | | 17 | 0 |
|---|---|---|---|---|---|
| RA+1 | LDQ | r | 0 | addr | |

| | |
|---|---|
| r | Auto recall bit |
| addr | Address of parameter block |

The four-word parameter block must be defined as follows.

| | 59 | 17 | 8 0 |
|---|---|---|---|
| addr +0 | file name | stat | fc |
| +1 | group name | | |
| +2 | capsule or overlay name | fwa | |
| +3 | random address | lwa+1 | |

| | |
|---|---|
| file name | Name of file containing capsule or overlay |
| stat | Status of LDQ call (ignored during request). Upon completion of call, stat is set to one of the following values. |

| Value | Description |
|---|---|
| 0 | Function completed without error |
| 1 | Illegal function code |

† This macro is not available in SYSTEXT. The user must call common deck COMCMAC (refer to appendix A).

| Value | Description |
|---|---|
| 2 | Bad address (must have fwa<lwa+l<field length) |
| 3 | Nonexistent file or file not on mass storage |
| 4 | Bad disk address (out of file bounds) |
| 5 | Capsule or overlay not found at specified location |
| 6 | Insufficient space provided for capsule or overlay |

If either error 5 or 6 occurs, the contents of the loadable area are undefined.

fc — Function code:

| | |
|---|---|
| 0 | Load capsule |
| 2 | Load overlay |

LDQ sets bit zero to one when the request is complete.

group name — Name of capsule group; zero for overlay load

capsule or overlay name — Name of desired capsule or overlay

fwa — First word address of the area into which the capsule or overlay is to be read

random address — Location of capsule or overlay on specified file

lwa+1 — Last word address plus 1 of area for capsule or overlay

LDQ reads a capsule or overlay from the specified mass storage location, removing the prefix table, but not altering the record otherwise. LDQ ensures that the location contains a capsule (56 table), if a capsule load is requested, or an overlay (50, 51, 53, or 54 table), if an overlay load is requested. LDQ also determines that the entire capsule or overlay fits into the specified area and that the name is correct.

Macro Format:†

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | LOADQ | addr,r |

addr — Address of parameter block

r — Auto recall

† This macro is not available in SYSTEXT. The user must call the common deck COMCMAC to use this macro (refer to appendix A).

## MEMORY ALLOCATION FOR OVERLAY LOADERS

The overlay loaders load records into core. COS-type overlays are always loaded at RA, and the program address is set to the number of parameters plus three. If a record begins with an overlay table ($50_8$-, $51_8$-, and $53_8$- identification word), it is loaded according to the relative address given in the table (figure 2-11-1). The beginning load address must be greater than $100_8$. For a $50_8$ single entry point overlay, loading starts at origin minus one; for a $51_8$ multiple entry point overlay, loading starts at origin minus wc minus one, where wc is the number of entry points.

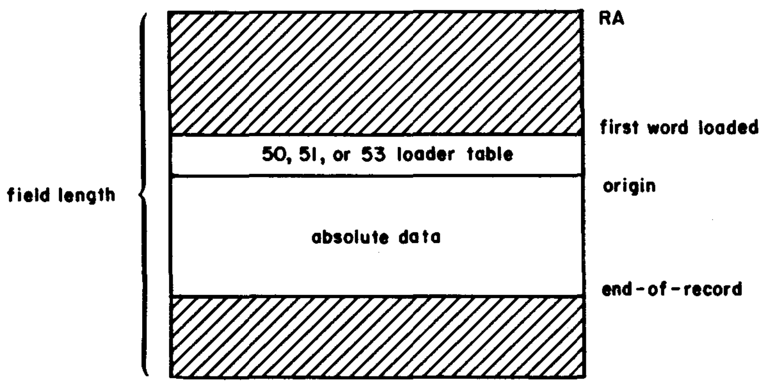



Figure 2-11-1. Absolute Loader Memory Assignment

## WRITING PROGRAMS UNDER NOS

To observe the techniques necessary to use the features described, the user should read programs that use the macros and common decks. Appendix D contains a listing of the COPYB program that illustrates the use of these macros. Other programs that use common decks and macros can be cross-referenced by the KRONREF control statement.

If the user follows the coding specifications described in appendix C, the documentation utilities can be used to document his program. These specifications are supplied as a guide to enable the user to understand the format of system programs and to give him an idea of how to structure programs.

The programmer should organize his program area, buffers, and working storage areas in a manner that uses the least amount of memory for the shortest period of time. This provides him with better performance and more efficient use of system resources. Areas of code that are used once, such as preset routines, should be overlayed with buffers that are used later in the program. All main programs should be set to load past location $100_8$ since areas before this address contain program and system parameters. † Placing FETs, table pointers, and other critical data areas at the beginning of the program aids in debugging because these areas do not change addresses as code in the program is changed.

## WRITING INTERACTIVE PROGRAMS

As a terminal user becomes more familiar with the use of NOS, he may wish to write interactive programs in languages other than those oriented specifically to the terminal (for example, COMPASS). He may also desire to convert some batch application programs to interactive programs. The following description of system conventions acquaint the user with potential problem areas in performing this conversion or in writing interactive programs.

### CONVERSION PROBLEMS

The major difference between an interactive job and a batch job is in the handling of files assigned to a terminal. Since the system is designed with a common control interface to all types of equipment, the programmer need not be concerned about the specific piece of equipment with which he is communicating.

Control Data application programs not written with terminal interaction in mind may present problems to the user who attempts to convert them to interactive programs. These problems are:

- Each line of terminal input is considered a logical record. This causes two basic problems: First, any program that terminates input processing on an EOR processes only one line of input. Second, programs that read ahead on input place a special significance on an EOR, and therefore, cannot effectively interact while accepting these one-line records.

  This problem is minimized by the design of NOS input/output. However, when the problem does occur, it is fatal to program execution.

† Refer to appendix E.

- Many programs output more than 72 characters per line. This may result in unacceptable output; however, this is easily corrected by using the LO72 control statement or by reformatting the output in the program.

- Many programs place format control in column 1 which is printed at the terminal. Again, although this may cause unacceptable output, it is easily corrected.

- Many programs perform unnecessary write requests on output. This can cause high system overhead.

Portions of the NOS standard product set are modified to correct these last three problems.

## DEFAULT FILE ASSIGNMENTS AND SPECIAL FILE TREATMENT

All jobs interacting with a terminal normally have input and output files assigned to the terminal unless the user assigns them to a different device. A user may assign a file to the terminal by assigning a file to equipment TT using the ASSIGN control statement or REQUEST macro; conversely, the user may assign his terminal input or output file to mass storage by specifying a device type of MS.

Normally, an interactive program need not check for EOI on INPUT; however, INPUT file status should be checked for EOI if INPUT has been assigned to mass storage.

## SPECIAL HANDLING

When a read request is made on a terminal input file, an output file's circular buffer can be automatically flushed (with an EOR write) before the system issues the read request.

The system uses one of the following methods in determining when output data is to be sent to the terminal.

- FET pointers can be stored in RA+2 through RA+$63_8$, which specify the files to be flushed. The FET pointer is the file name left-justified with zero-fill in bits 59 through 18 and the address of the FET in bits 17 through 0.

- A list of files can be specified (refer to the SETLOF macro, section 6) from which the system determines the appropriate file name.

In using either of the previous methods, the system uses the first file that meets the following criteria.

- The file must either be assigned to the terminal (device type equals TT in the FET) or if unassigned must be named OUTPUT.

- The file must meet the conditions described under the flush bit (refer to description of flush bit, FET Description in section 3).

## OTHER SPECIAL HANDLING

The following additional considerations should be noted by those attempting to write interactive programs.

- An EOR or EOF write on a terminal file has no special significance except that it ensures that the buffer is dumped to the terminal.

- When terminal input data is passed to an executing program, the following convention is followed.

  If an odd number of characters is entered, the last character plus 1 is a space.

  If the input data consists of data followed by a carriage return, the program has a five-word FET, the system supplies an EOR level of zero (level number is in bits 17 through 14 of the FET code and status word). If the program has a six-word or larger FET, the system supplies an EOR level of 1. If there is no data input but only a carriage return, an EOF is supplied by the system.

  Thus, a program can determine if input is from a terminal by using six-word or longer FETs and by sensing an EOR level of 1. The system input/output macros and common decks are coded to handle this case properly. Thus, most of the system utilities interact with a terminal. It should be noted that this interaction for FORTRAN may depend on the library used to satisfy externals (SYSLIB is the FORTRAN object-time library that supports interactive communication).

  Input to an executing program is handled in the same manner as the CIO READSKP request. If the user's buffer is not large enough to accommodate a full line of input, the data is truncated and the excess is lost.

- A program that is interacting with a terminal should not do a recall on OUTPUT if it does an EOF write on OUTPUT to clear the buffer (do not specify the r option on WRITEF macro). If recall is specified, an extra rollout of the program is required before the program terminates. If the conventions mentioned earlier concerning the status of output are followed, it is not mandatory to write an EOF to clear the buffer; however, if it is done (to remain compatible for batch use, for example) as the last thing before placing END in RA+1 (ENDRUN macro) without recall, little system overhead is incurred.

- If the conventions for special handling are followed, a job being time-sliced by the system on either a CPU or central memory usage has all data in the output buffer sent to the terminal.

- If a job terminates because an error flag is set, the contents of the control point area's first message buffer is sent to the terminal as part of the output. When the user is in the BATCH subsystem, this message buffer is always sent to the terminal on job termination. Messages can be placed in this area using the MESSAGE macro. Messages longer than 48 characters are truncated to 48 characters.

- If a buffer argument error is detected on an output buffer when output is being issued automatically, the output is ignored and the FET is not acted on. This error usually indicates that the executing program has destroyed part of its own field length.

- If the user program destroys its input FET after the FET has been validated by CIO, but before the data is actually passed, the input request is ignored. The probability of either of these occurrences is extremely remote, and if the CIO request is made with recall, it is impossible.

- A COMPASS program can determine whether it is interacting with a terminal by checking the origin type that is passed or by checking the type of equipment to which the file was assigned (byte 0 of word 1 of the FET). Refer to the common deck COMCSTF, appendix A.

## PROGRAM CONTROL OF TERMINAL ACTIVITY

The user can design an interactive program to control terminal activity in two ways.

- Include control bytes in his output to control the positioning of the printing element, define alternate input modes, etc.
- Issue a DISTC macro to disable the terminal operator's control of his program during critical phases of execution.

### CONTROL BYTES

A control byte is a 12-bit quantity, right-justified in bit positions 0, 12, 24, 36, or 48 of a CM word.

**NOTE**

> The user must be careful that data is not mistaken for a control byte. For example, the characters :D typed at the end of a line may log-off the user, since the code 0004 is transmitted.

The following paragraphs describe the bytes available to the user and their functions.

#### End-of-Line

End-of-line generates a carriage return and line feed, positioning the terminal printing element at the beginning of the next line. An end-of-line consists of 12 to 66 bits of zero, right-justified in one or two central memory words.

#### End-of-Block (0001 or 0002)

This byte prevents the positioning of the terminal printing element at the beginning of the next line. An end-of-block byte can be used to allow the terminal operator to enter input on the same line as the input request is printed. This byte must be followed by an end-of-line. If not followed immediately by a read request, any output following this byte may be lost.

#### Auto Input (0003)

This byte is intended for use by the time-sharing executive for auto mode input,† but the user may include it in his output. The preceding characters in the word in which this byte occurs are sent to the terminal and are also retained as the first characters of the input line. This byte must be followed by an end-of-line. The next terminal operation must be an input request. The terminal prompt (a question mark) is suppressed.

#### Log-Off User (0004)

This byte disconnects the terminal user's telephone lines. This byte must be the first byte of a line and must be followed by an end-of-line.

---

† Refer to the NOS Time-Sharing User's Reference Manual for further information about auto mode input.

### Set Transparent Input Mode (0005)

This byte changes the input mode from normal or ASCII (refer to appendix F, volume 1) to transparent mode. A 0005 byte directs the driver to translate all characters to the 6/12 bit internal codes as in ASCII mode. Also, no input control characters except RETURN and INTERRUPT (delete, backspace, STEXT, and ETEXT) are processed. Rather, all characters are passed to the program as data. INTERRUPT is passed as NULL followed by end-of-line. RETURN is passed as end-of-line. End-of-line terminates transparent input mode. This byte must be the first byte of the first word of a line and must be followed by an end-of-line.

### Set Binary Input Mode (0006)

This byte changes the input mode from normal or ASCII (refer to appendix F, volume 1) to binary mode. The control byte must be byte 0 of the first word of a line and must be followed by an end-of-line. Bytes 1 and 2 are defined as follows:

| | |
|---|---|
| Byte 1 | Specifies the maximum number (octal) of characters to be received before input is terminated. Byte 1 cannot exceed 150. If byte 1 is 0 or is greater than 150, the number is assumed to be 1. |
| Byte 2 | Specifies the termination code. When a character is received from the terminal that matches the lower 7 bits of this byte, the input operation is terminated. If bit 11 of this byte is set, no termination character is assumed. The eighth bit (parity bit) is not checked. |

This conversion mode packs the 8 bits of data as the lower 8 bits of a 12-bit byte and sets the upper bit (bit 11). The exhaustion of the character count or the occurrence of the termination code causes the end-of-line condition to be set. A 0007 byte is forced as the first byte of input so the data is transmitted as binary if it is listed.

### Initiate Binary Output (0007)

This byte initiates binary output. If the user wishes to output data formatted as described for binary input, a 0007 byte must precede the data. This mode continues until an end-of-line or nonbinary output data byte is detected. Termination by an end-of-line, however, does not cause a carriage return and line feed. The 0007 byte must be byte 0 (bits 59 through 48) of the first word of a line.

A binary output data byte is in the format 4xxx octal, where xxx is the 8-bit octal code for the character being printed. The upper 4 bits of a binary data byte are always set to $1000_2$. If this pattern is not detected, the terminal is switched to standard output mode. Thus, the user can output standard mode data immediately following binary mode data without explicitly specifying a termination (end-of-line). Note that the characters 5, 6, 7, and 8 (display codes 40, 41, 42, and 43 octal) match the binary mode bit pattern when in the upper half of a byte and are interpreted as binary data. Therefore, these characters cannot be used to terminate binary output. Once binary output mode is terminated it remains cleared unless resumed by a 0007 control byte.

The following word, when output from a time-sharing terminal, produces a line feed and prints a question mark. There is no carriage return or line feed after the question mark.

    0007 4012 4077 0000 0000

The 4012 byte produces a line feed (012 is the eight-bit octal code for ASCII standard print) and 4077 is the question mark. The binary output is terminated by the 24 bits of trailing zeros, constituting an end-of-line. The same output can be obtained with the following.

    0007 4012 7100 0000 0000

Here 71 is the display code for a question mark and 7100 is a nonbinary data byte. Binary output is terminated at the question mark and the end-of-line produces a carriage return and line feed.

A control byte (0003, 0005, 0006) that changes the terminal input mode prevents the system from printing a question mark in response to a program request for input. However, in all other cases, a read request on the input buffer causes the system to print a question mark at the terminal.

### End of Transaction Block (0010)

This byte is used by the transaction subsystem (TAF/TRANEX) to indicate the end of a transaction block. This byte must be followed by an end-of-line.

### Initiate ASCII Output (0011)

This byte is used to initiate ASCII output. This byte must be byte 0 (bits 59 through 48) and applies only to the line currently being output.

### End of Transaction Block with Response (0012)

This byte is used by the transaction subsystem to indicate the end of a transaction block with a completion response to be sent back to the transaction subsystem. This byte must be followed by an end-of-line.

### End of String (0013)

This control byte allows a user to terminate a line of output data without repositioning the terminal carriage. This byte must be followed by an end-of-line (which is ignored) and output continues with subsequent data.

### Internal End-of-Block (0014)

This control byte is the first byte of a word and is followed by an end-of-line. This byte is reserved by NOS and should not be used since it may cause loss of data.

## CONTROL OF PROGRAM EXECUTION

By entering various keys or commands, the terminal user can exercise control over an interactive program during all phases of execution. The following is a list of these keys and commands and their effect on the executing program.

| Phase | Key/Command | Effect |
|---|---|---|
| Waiting for Input | STOP | The program is terminated and the terminal is placed in command mode. All other entries are passed to the program as data. For all input modes except binary, carriage return is passed as end-of-line. If no data is entered, a null input line is passed to the program. STOP is not detected in binary input mode. |
| Generating Output† | S | The S key has the same effect as the STOP command in the input phase. |
| | I | The output operation ceases and the terminal is placed in suspended mode. The time-sharing executive recognizes only:<br>• The P key causing a return to program control, discarding any data in the program buffer prior to the entry of the I key.<br>• A carriage return causing output to be continued and program control to continue normally. Part of the current line of output may be lost when output is suspended.<br>• Any other key is interpreted as the STOP command. |
| | INTERRUPT or BREAK | This key has the same effect as the I key. |
| Executing† | STOP | This command has the same effect as the STOP command in the input phase. |
| | INTERRUPT or BREAK | INTERRUPT has the same effect as the I key during output. |

## DISTC MACRO

The DISTC macro enables an interactive program to prevent the time-sharing executive from processing terminal control keys and commands.

† Certain correspondence code terminals do not allow any input except INTERRUPT during output or execution.

**Macro Format:†**

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | DISTC | st, addr, INT |

st — Control status:

- ON — Activates program terminal control
- OFF — Deactivates program terminal control

addr — Specifies an interrupt address which the system uses to inform the program that a terminal operator attempted to terminate the program. This parameter is valid only if st=ON. If addr is not specified, the program is not notified of attempted terminal control.

INT — If INT is specified when terminal control is attempted, the system copies the program operating registers (exchange package)†† to the $20_8$-word area starting at addr. The program address is changed to addr + $20_8$. If INT is not specified, the contents of addr are set to one. The program can then check the location periodically to determine if the terminal operator has attempted to interrupt or terminate the program.

The interrupt address remains the same after the program is notified of an attempted interrupt. Once a terminal control is set or cleared, that status remains in effect until:

- The program issues a DISTC macro that changes the status.
- The program terminates, which returns the terminal to command mode.
- The user logs off or disconnects the terminal. If the terminal was disconnected, the user may recover. If the executing program is continued after recovery, the terminal control status remains as it was prior to the disconnect. If the executing program is not continued, the interrupt address is cleared.

† This macro is not available in SYSTEXT. The user must call the common deck COMCMAC.

†† A system request, XJR, allows the user to restore the operating registers and resume normal processing after the interrupt processing has been completed.

The following is a list of the same keys and commands described previously and their effect when the user selects the disable terminal control feature.

| Phase | Key/Command | Effect |
|---|---|---|
| Waiting for Input | STOP | The user may enter this command at the beginning of any input line unless binary input mode has been selected. STOP is passed to the program via the input file. If an interrupt address was specified, the program is notified of attempted control. |
| Generating Output | S | If an interrupt address was specified, the program is notified of attempted control. If the program issued an input request before the operator attempted terminal control, the system takes the same action as during the input phase. |
| | I | Same as for S. |
| | INTERRUPT | Same as for S. |
| Executing | STOP | If an interrupt address was specified, the program is notified of attempted control. |
| | INTERRUPT | Same as for STOP. |

## CSET MACRO

The CSET macro sets the terminal's initial and current character set mode to either ASCII or NORMAL.

Macro Format: †

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | CSET | arg, C |

arg — Argument:

| Mode | Description |
|---|---|
| ASCII | Set extended character set mode |
| NORMAL | Set 64/63 character set mode |
| RESTORE | Set current terminal character mode to initial terminal character mode |

C — Change initial and current terminal character mode to that specified; if C is omitted, only the current mode is changed.

**NOTE**

The use of this macro may cause the character mode of the terminal to switch prior to the printing of all previous output. This can be prevented by preceding the macro call with an input request.

† This macro is not available in SYSTEXT. The user must call the common deck COMCMAC (refer to appendix A).

## PARITY MACRO

The PARITY macro sets the terminal to the indicated parity.

Macro Format: †

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | PARITY | arg |

arg EVEN selects even parity; ODD selects odd parity.

**NOTE**

The use of this macro may cause the parity of the terminal to switch prior to all previous output being printed at the terminal. This can be prevented by preceding the macro call with an input request.

## TSTATUS MACRO

The TSTATUS macro returns the status of the terminal. Information returned includes terminal type, subsystem being used, the terminal number, parity, character set information, duplex and tape mode status, and the current interrupt address.

Macro Format: †

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | TSTATUS | addr |

addr Address of 2-word status return buffer.

† This macro is not available in SYSTEXT. The user must call the common deck COMCMAC (refer to appendix A).

Exit conditions:

| | 59 | 35 | 17 | 11 0 |
|---|---|---|---|---|
| addr + 0 | tid | | sys | tn |
| + 1 | 0 | int | tc | st |

tid Left-justified, zero-filled display code terminal type identification:

| Type | Description |
|---|---|
| TTY | ASCII code terminal with standard print |
| TTYD | TTY with 61 character set |
| MEMAPL | Memorex® 1240 (ASCII code) terminal with APL print |
| MEMAPLD | MEMAPL with 61 character set |
| CORAPL | Correspondence code terminal with APL print |
| CORAPLD | CORAPL with 61 character set |
| COR | Correspondence code terminal with standard print |
| CORD | COR with 61 character set |
| BLKEDT | Block transmission (ASCII code) terminal with standard print. |
| BLKEDTD | BLKEDT with 61 character set |

sys Subsystem ordinal:

| Code | Description |
|---|---|
| 0 | NULL subsystem |
| 1 | BASIC subsystem |
| 3 | FTNTS subsystem |
| 4 | EXECUTE subsystem |
| 5 | BATCH subsystem |
| 6 | ACCESS subsystem |
| 7 | TRANACT subsystem |

tn Terminal number (octal)

int Current interrupt address, if set (refer to DISTC) tc Transmission code:

| Code | Description |
|---|---|
| 0 | ASCII terminal |
| 1 | Correspondence terminal |
| 2 | NIXDORF terminal |

st Terminal status bits:

| Code | Description |
|---|---|
| 0 | Parity (odd if set) |
| 1 | Initial character set (ASCII if set) |
| 2 | Current character set (ASCII if set) |
| 3 | Set indicates full duplex |
| 4 | Set indicates tape mode |

# CPU COMMON DECKS A

Appendix A lists the CPU common decks of general interest to the COMPASS programmer, including a list of those common decks that are available in relocatable form on the system library (SYSLIB). The user can obtain documentation of all CPU common decks by entering the following control statement after accessing the system OPL.

```
MODIFY(Z)/*EDIT,CALLCPU
DOCMENT.
```

In addition, by using the KRONREF utility to determine which common decks are associated with which programs, the user can determine the program listings to obtain for examples of the use of particular common decks.

Since common decks are continually being changed or updated, it is impossible to maintain a complete and current description of each common deck in this appendix. Thus, the user should consult system listings as described above for specific information.

## COMCMAC

The COMCMAC common deck contains macros (not available in SYSTEXT) for issuing system functions for system-oriented programs. The common decks for the processors used must also be called by the user.

The macros defined in common deck COMCMAC are available in systems text PSSTEXT. Therefore, the user can access these macros either through the system OPL or by specifying the alternate systems text PSSTEXT.

The following COMCMAC macros are described in the indicated sections.

| Macro Name | Description | Section |
|---|---|---|
| CATLIST | Catalog user's permanent files | 5 |
| CHECKPT | Create checkpoint dump | 10 |
| CONSOLE | Set K display control | 6 |
| CSET | Set terminal character set mode | 12 |
| DAYFILE | Access dayfile | 9 |
| DISTC | Disable terminal control | 12 |
| EDATE | Edit packed date | 11 |
| ENCSF | Enter new control statement file | 4 |
| ESYF | Enter system file | 9 |
| ETIME | Edit packed time | 11 |
| EXCST | Execute control statement | 10 |
| GETASL | Get account block SRU limit | 6 |
| GETFLC | Read field length control word | 6 |
| GETFNT | Read FNT/FST entry table | 4 |
| GETGLS | Get global library set | 6 |
| GETJA | Read job accounting words | 6 |
| GETJCR | Read job control registers | 6 |

| Macro Name | Description | Section |
|---|---|---|
| GETJN | Read job name | 6 |
| GETJO | Read job origin code | 6 |
| GETJSL | Get job step SRU limit | 6 |
| GETLC | Get loader control word | 6 |
| GETMC | Read machine characteristics | 6 |
| GETPFP | Get permanent file parameters | 6 |
| GETPR | Read CPU priority | 6 |
| GETQP | Read job queue priority | 6 |
| GETSS | Get subsystem | 6 |
| GETTL | Read time limit | 6 |
| LOADD | Load fast dynamic load capsule directory | 11 |
| LOADQ | Load fast dynamic load capsules | 11 |
| NORERUN | Clear rerun status | 7 |
| PARITY | Set terminal parity | 12 |
| PRIMARY | Make file primary | 4 |
| PSCSF | Position control statement file | 4 |
| RDVT | Return device type | 9 |
| RERUN | Set rerun status | 7 |
| ROLLOUT | Roll out job | 6 |
| SETGLS | Set global library set | 6 |
| SETJCR | Set job control registers | 6 |
| SETLC | Set loader control word | 6 |
| SETPR | Set CPU priority | 6 |
| SETQP | Set queue priority | 6 |
| SETSS | Set subsystem | 6 |
| SETSSM | Set secure system memory | 6 |
| SETUI | Set user index | 6 |
| SUBMIT | Enter job in input queue | 7 |
| SUBR | Create subroutine tag | 11 |
| TSTATUS | Return terminal status | 12 |

## COMCCMD

The COMCCMD common deck contains macros (not available in SYSTEXT) for issuing special job control and accounting function requests.

The macros defined in common deck COMCCMD are available in systems text PSSTEXT. Therefore, the user can access these macros either through the system OPL or by specifying the alternate systems text PSSTEXT.

The following COMCCMD macros are described in the indicated sections.

| Macro Name | Description | Section |
|---|---|---|
| COMMON | Change file type to library (LIFT) | 4 |
| GETEM | Read current exit mode | 6 |
| GETLOF | Get list of files pointer | 6 |
| MACHID | Read machine identification | 6 |
| MODE | Set exit mode flags | 6 |
| PACKNAM | Request pack name | 6 |
| SETASL | Set account block SRU limit | 6 |
| SETJSL | Set job step SRU limit | 6 |
| SETLOF | Set list of files pointer | 6 |
| SETMFL | Set job maximum field length | 6 |
| SETRFL | Set job step initial field length | 6 |
| SETTL | Set job step time limit | 6 |
| VERSION | Read operating system version name | 6 |

## OTHER COMMON DECKS

The following common decks are also available to the user.

| Common Deck | Description |
|---|---|
| COMCARG | Processes an argument list by the use of an equivalence table |
| COMCARM | Processes multiple word arguments |
| COMCCDD | Converts decimal digits to display code with leading zero suppression |
| COMCCFD | Converts a 30-bit integer to display code in FORTRAN F10.3 format |
| COMCCIO | Performs I/O functions through the PPU program CIO |
| COMCCOD | Converts octal digits to display code with leading zero suppression |
| COMCCPM | Calls the PPU program CPM to perform tasks involving control point activity |
| COMCDXB | Converts one word of display code digits to a binary value |
| COMCEDT | Edits an 18-bit packed date or time into a 10-character display coded date or time |
| COMCFCE | Edits a permanent file catalog entry into a two-line output format |
| COMCLFM | Processes requests for the PPU program LFM |
| COMCMTM | Contains macros for generation, allocation, and processing of managed tables |

| Common Deck | Description |
|---|---|
| COMCMTP | Contains routines for processing managed tables |
| COMCMVE | Moves a block of data |
| COMCOVL | Requests the PPU program LDR to load a specified overlay |
| COMCPFM | Performs permanent file action functions by calls to the PPU program PFM |
| COMCPOP | Obtains parameters from a string buffer |
| COMCQFM | Processes requests for the PPU program QFM |
| COMCRDC | Reads one coded line from a CIO buffer to a working buffer |
| COMCRDH | Reads one coded line from a CIO buffer to a working buffer with trailing space fill |
| COMCRDO | Reads one word from a CIO buffer to the X6 register |
| COMCRDS | Reads one coded line from a CIO buffer to a working buffer where it is stored one character per word |
| COMCRDW | Reads the specified number of words from a CIO buffer to a working buffer |
| COMCSFM | Processes requests for the PPU program SFM |
| COMCSFN | Replaces trailing 00 codes with 55 codes in a word |
| COMCSRT | Identifies the format of a record from the first 64 words located in a working buffer |
| COMCSSN | Skips a sequence number on a coded line if present |
| COMCSST | Sorts a table into ascending order using a shell-sorting technique |
| COMCSTF | Determines if a file is, or will be, assigned to a terminal |
| COMCSYS | Contains routines for processing certain system requests |
| COMCUPC | Unpacks a control statement to individual parameters |
| COMCWOD | Converts a word to octal display code by an in-line sequence of shifts and masks |
| COMCWTC | Transfers one coded line in C format from a working buffer to a CIO buffer |
| COMCWTH | Transfers one coded line in H format from a working buffer to a CIO buffer; trailing spaces are deleted |
| COMCWTO | Writes one word to a CIO buffer from X6 |
| COMCWTS | Transfers one coded line from a string buffer to a CIO buffer with trailing space suppression |
| COMCWTW | Transfers data from a working buffer to a CIO buffer |
| COMCZTB | Replaces all 00 codes with 55 codes in a word |

## SYSLIB

The following common decks are available in relocatable form on the system library SYSLIB.

| Common Deck | Entry Points | Description |
|---|---|---|
| COMCCIO | CIO= | I/O function processor |
| COMCCPM | CPM= | Control point manager processor |
| COMCLFM | LFM= | Local file manager processor |
| COMCMVE | MVE= | Move block of data |
| COMCOVL | OVL= | Overlay load processor |
| COMCPFM | PFM= | Permanent file processor |
| COMCRDC | RDC= | Read coded line, -C- format |
| COMCRDH | RDH= | Read coded line, -H- format |
| COMCRDO | RDO= | Read one word |
| COMCRDS | RDS= | Read coded line to string buffer |
| COMCRDW | LCB=, RDW=, RDX= | Read words to working buffer |
| COMCSYS | MSG=, RCL=, | |
| | SYS=, WNB= | Process system request |
| COMCWTC | WTC= | Write coded line, -C- format |
| COMCWTH | WTH= | Write coded line, -H- format |
| COMCWTO | WTO= | Write one word |
| COMCWTS | WTS= | Write coded line from string buffer |
| COMCWTW | DCB=, WTW=, WTX= | Write words from working buffer |

# EXAMPLES OF RANDOM I/O B

Appendix B contains several programs that perform the following random I/O operations.

- Create a random file
- Read a random file with list
- Write (replacing records) on a random file

The COMPASS program illustrated in figure 2-B-1 reads the input deck in figure 2-B-2 and creates a random file in the format illustrated in figure 2-B-3. The resulting index directory record is also illustrated.

An example of a program which uses the READLS (read with list) macro to retrieve (read) a list of records from the same random file is illustrated in figure 2-B-4.

Figure 2-B-5 illustrates two ways of writing (replacing) records on a random file. The new record 1 is written at EOI; that is, at the same random address. Record 3 is written to the file by using the REWRITER macro, which rewrites in place.

```
CREATE
          IDENT  CREATE,FWA
          ABS
          SST
          TITLE  CREATE - CREATE RANDOM FILE.
          ENTRY  CREATE
          ENTRY  RFL=
          SYSCOM B1
*COMMENT  CREATE RANDOM FILE.
          COMMENT COPYRIGHT CONTROL DATA 1976.
          SPACE  4
***       CREATE - CREATE RANDOM FILE.
*         PROGRAMMER NAME.  76/3/8.
          SPACE  4
***              THIS PROGRAM CREATES A RANDOM FILE AND DIRECTORY FROM
*         DATA SUPPLIED ON FILE INPUT.
*         FORMAT OF INPUT -
*                CARD 1      COLUMN 1-2   RECORD NUMBER RIGHT JUSTIFIED
*                CARD 1      COLUMN 11-20 RECORD NAME
*                CARD 2-N    COLUMN 1-80  RECORD DATA
*                CARD N      EOR
*                REPEAT SEQUENCE -
*                MAX. 64 RECORDS
*
*         FILE IS WRITTEN TO RANFILE.
          SPACE  4
***       DAYFILE MESSAGES.
*
*         * INDEX OUT OF RANGE.* = INDEX INPUT .GE. 64.
*         * RANDOM FILE CREATED.* = PROGRAM COMPLETE.
          SPACE  4
**        PROGRAM CONSTANTS.


 IBUFL    EQU    101B        INPUT BUFFER LENGTH
 INDL     EQU    101B        INDEX LENGTH
 WBUFL    EQU    10B         WORKING STORAGE BUFFER LENGTH
 RBUFL    EQU    1001B       FILE CREATION BUFFER
          TITLE  MAIN PROGRAM.
          ORG    103B


 FWA      BSS    0
 I        BSS    0           INPUT FET
 INPUT    FILEB  IBUF,IBUFL
 R        BSS    0
 RANFILE  RFILEB RBUF,RBUFL,(IND=INDEX,INDL)
 MAXI     CON    0           MAXIMUM INDEX


 CREATE   SB1    1
          REWIND R
 CR1      READ   I,R         READ NEXT RECORD
          READO  I
          NG     X1,CR4      IF END OF FILE
          MX0    12
          BX5    X0*X6
```

Figure 2-B-1. COMPASS Program to Create a Random File (Sheet 1 of 3)

```
         SB7     B0          SET OCTAL CONVERSION
         RJ      DXB         CONVERT DIGIT
         NZ      X4,ERR      IF ERROR IN RECORD NUMBER
         BX5     X6          SAVE RECORD NUMBER


         RECALL  R
         READO   I           READ RECORD NAME
         NG      X1,CR4      IF END OF FILE
         MX0     42          SET ADDRESS OF RECORD IN INDEX
         SA1     R+6
         LX1     30
         BX6     X0*X6
         BX1     -X0*X1
         BX6     X6+X1
         SA6     INDEX+X5
         SA1     MAXI
         IX1     X1-X5
         PL      X1,CR2      IF NOT LARGEST INDEX
         BX6     X5          SET NEW MAX. INDEX
         SA6     A1
CR2      READC   I,WORK,WBUFL
         NG      X1,CR4      IF END OF FILE
         NZ      X1,CR3      IF END OF RECORD
         WRITEC  R,WORK,WBUFL
         EQ      CR2
CR3      WRITER  R
         EQ      CR1         READ NEXT RECORD

**       WRITE INDEX.

CR4      RECALL  R
         SA1     MAXI
         WRITEW  R,INDEX,X1+1
         WRITER  R,R
         WRITEF  R,R
         REWIND  R
         MESSAGE (=C*RANDOM FILE CREATED.*),,R
         ENDRUN

ERR      MESSAGE (=C*INDEX OUT OF RANGE.*),,R
         ABORT
         SPACE   4
**       COMMON DECKS.

*CALL    COMCRDO
*CALL    COMCCIO
*CALL    COMCWTW
*CALL    COMCWTC
*CALL    COMCRDC
*CALL    COMCSYS
*CALL    COMCRDW
*CALL    COMCDXB
```

Figure 2-B-1. COMPASS Program to Create a Random File (Sheet 2 of 3)

```
          USE     LITERALS

**        BUFFERS.

 INDEX    BSSZ    INDL         INDEX BUFFER
 IBUF     EQU     *            INPUT BUFFER
 RBUF     EQU     IBUF+IBUFL   RANDOM FILE BUFFER
 WORK     EQU     RBUF+RBUFL   WORKING BUFFER
 RFL=     EQU     WORK+WBUFL   JOB FL
          END
```

Figure 2-B-1. COMPASS Program to Create a Random File (Sheet 3 of 3)

```
10         RECOR10
REC10
THIS IS RECORD NUMBER 10.
-EOR-
03         RECORD3
REC3
THIS IS RECORD NUMBER 3.
-EOR-
07         RECORD7
REC7
THIS IS RECORD NUMBER 7.
-EOR-
02         RECORD2
REC2
THIS IS RECORD NUMBER 2.
-EOR-
05         RECORD5
REC5
THIS IS RECORD NUMBER 5.
-EOR-
01         RECORD1
REC1
THIS IS RECORD NUMBER 1.
-EOR-
04         RECORD4
REC4
THIS IS RECORD NUMBER 4.
-EOR-
06         RECORD6
REC6
THIS IS RECORD NUMBER 6.
-EOR-
-EOI-
```

Figure 2-B-2. Input File for Program Creating a Random File (Figure 2-B-1)

| |
|---|
| RECORD 10 |
| RECORD 3 |
| RECORD 7 |
| RECORD 2 |
| RECORD 5 |
| RECORD 1 |
| RECORD 4 |
| RECORD 6 |
| INDEX |
| EOF |
| EOI |

Random File Format

| Record Name | RSA |
|---|---|
| RECORD 1 | 6 |
| RECORD 2 | 4 |
| RECORD 3 | 2 |
| RECORD 4 | 7 |
| RECORD 5 | 5 |
| RECORD 6 | 10 |
| RECORD 7 | 3 |
| RECORD 10 | 1 |
| 0 | 0 |
| . | |
| . | |
| . | |

Index Record

**Figure 2-B-3. Structure of the Random File Created**

```
RLIST
          IDENT  RLIST,FWA
          ABS
          SST
          TITLE  RLIST - READ SELECTED LIST FROM RANDOM FILE.
          ENTRY  RLIST
          ENTRY  RFL=
          SYSCOM B1
*COMMENT  READ LIST OF RANDOM FILE.
          COMMENT COPYRIGHT CONTROL DATA 1976.
          SPACE  4
***       RLIST - READ SELECTED LIST FROM RANDOM FILE.
*         PROGRAMMER NAME.   76/2/14.
          SPACE  4
***              THIS PROGRAM READS A SELECTED LIST OF RECORDS
*         FROM RANDOM FILE AND WRITES THEM TO FILE OUTPUT.
*         SELECTED LIST IS -
*                3
*                7
*                4
*                10
          SPACE  4
***       DAYFILE MESSAGES.
*
*         *INDEX OVERFLOW.* = INDEX BUFFER OVERFLOWED.
*         *LIST COPIED.* = PROGRAM COMPLETE.
          SPACE  4
**        PROGRAM CONSTANTS.


 INDL     EQU    101B          INDEX LENGTH
 RBUFL    EQU    101B          RANDOM FILE BUFFER LENGTH
 OBUFL    EQU    101B          OUTPUT BUFFER LENGTH
 WBUFL    EQU    10B           WORKING BUFFER LENGTH
          TITLE  MAIN PROGRAM.
          ORG    103B


 FWA      BSS    0

 R        BSS    0
 RANFILE  RFILEB RBUF,RBUFL,(IND=INDEX,INDL)
 O        BSS    0
 OUTPUT   FILEB  OBUF,OBUFL
 LIST     BSSZ   10B           LIST TO READ


 RLIST    SB1    1
          SKIPEI R,R           POSITION TO INDEX
          BKSP   R,R
          BKSP   R,R
          READ   R,R           READ INDEX
          READW  R,INDEX,INDL
          ZR     X1,ERR        IF INDEX TOO LARGE

**        SET LIST OF RECORDS.
```

Figure 2-B-4. COMPASS Program using READLS Macro to Retrieve a List of Records from a Random File (Sheet 1 of 2)

```
         MX0     42
         SA1     INDEX+3      RECORD 3
         BX6     -X0*X1
         SA6     LIST
         SA1     INDEX+7      RECORD 7
         BX6     -X0*X1
         SA6     A6+B1
         SA1     INDEX+4      RECORD 4
         BX6     -X0*X1
         SA6     A6+B1
         SA1     INDEX+10B    RECORD 10
         BX6     -X0*X1
         SA6     A6+B1
         SX6     LIST         SET ADDRESS OF LIST
         SA6     R+5

**       READ LIST.

 RLI1    READLS R,R
 RLI2    READW  R,WORK,WBUFL
         NG     X1,RLI3      IF END OF FILE
         WRITEW O,WORK,WBUFL
         EQ     RLI2

 RLI3    SX1    B6-WORK
         WRITEW O,WORK,X1
         WRITER O
         REWIND R
         MESSAGE (=C*LIST COPIED.*),,R
         ENDRUN

 ERR     MESSAGE (=C*INDEX OVERFLOW.*),,R
         ABORT
         SPACE  4
**       COMMON DECKS.

*CALL    COMCCIO
*CALL    COMCWTW
*CALL    COMCRDW
*CALL    COMCSYS
         SPACE  4
         USE    LITERALS

**       BUFFERS.

 INDEX   EQU    *            INDEX BUFFER
 RBUF    EQU    INDEX+INDL   RANDOM FILE BUFFER
 OBUF    EQU    RBUF+RBUFL   OUTPUT BUFFER
 WORK    EQU    OBUF+OBUFL   WORKING BUFFER
 RFL=    EQU    WORK+WBUFL   DEFAULT FL
         END
```

Figure 2-B-4. COMPASS Program using READLS Macro to Retrieve a List of Records from a Random File (Sheet 2 of 2)

```
REWRITE
          IDENT   REWRITE,FWA
          ABS
          SST
          TITLE   REWRITE - REWRITE RANDOM FILE.
          ENTRY   REWRITE
          ENTRY   RFL=
          SYSCOM B1
*COMMENT  REWRITE RANDOM FILE
          COMMENT COPYRIGHT CONTROL DATA 1976.
          SPACE   4
***       REWRITE - REWRITE RANDOM FILE.
*         PROGRAMMER NAME.   76/2/14.
          SPACE   4
***               THIS PROGRAM UPDATES RECORDS ON A RANDOM FILE.
*                 RECORD 3     IS UPDATED IN PLACE
*                 RECORD 1     IS REWRITTEN AT EOI
*                 THE DIRECTORY IS ALSO REWRITTEN
          SPACE   4
***       DAYFILE MESSAGES.
*         *INDEX OVERFLOW.* = INDEX TOO LARGE FOR BUFFER
          SPACE   4
**        PROGRAM CONSTANTS.


 INDL     EQU     101B         INDEX LENGTH
 RBUFL    EQU     101B         RANDOM FILE BUFFER LENGTH
          TITLE   MAIN PROGRAM.
          ORG     103B


 FWA      BSS     0

 R        BSS     0
 RANFILE  RFILEB RBUF,RBUFL,(IND=INDEX,INDL)
 INDS     CON     0


 REWRITE  SB1     1
          SKIPEI R,R           POSITION TO INDEX
          BKSP    R,R
          BKSP    R,R
          READ    R,R          READ INDEX
          READW   R,INDEX,INDL
          ZR      X1,ERR       IF INDEX TOO LARGE
          BX6     X1           SAVE INDEX SIZE
          SA6     INDS


**        CHANGE RECORD 3 IN PLACE.


          SA1     INDEX+3      SET ADDRESS OF RECORD 3
          MX0     42
          BX6     -X0*X1
          SA6     R+6
          WRITEC R,(=C*REC3-1*)
          WRITEC R,(=C*THIS IS UPDATED RECORD 3.*)
          REWRITER R,R
```

Figure 2-B-5. COMPASS Program to Replace Certain Records on a Random File (Sheet 1 of 2)

```
**        REWRITE RECORD 1 AT EOI.


          SKIPEI R,R          POSITION FILE AT EOI
          SA1    R+6
          MX0    42
          LX1    30
          SA2    INDEX+1
          BX6    X0*X2        PICK RECORD NAME
          BX1    -X0*X1
          BX6    X1+X6        REWRITE NEW ADDRESS
          SA6    A2

          WRITEC R,(=C*REC1-1*)

          WRITEC R,(=C*THIS IS UPDATED RECORD 1.*)
          WRITEC R,(=C*WRITTEN AT EOI BECAUSE LENGTH IS EXTENDED.*)
          WRITER R,R


**        REWRITE INDEX.


          SA1    INDS
          WRITEW R,INDEX,X1
          WRITER R,R
          WRITEF R,R
          REWIND R
          MESSAGE (=C*RANFILE UPDATED.*),,R
          ENDRUN


 ERR      MESSAGE (=C*INDEX OVERFLOW.*),,R
          ABORT
          SPACE  4
**        COMMON DECKS.


*CALL     COMCCIO
*CALL     COMCWTC
*CALL     COMCRDW
*CALL     COMCWTW
*CALL     COMCSYS
          SPACE  4
          USE    LITERALS


**        BUFFERS.


 INDEX    EQU    *
 RBUF     EQU    INDEX+INDL   RANDOM FILE BUFFER
 RFL=     EQU    RBUF+RBUFL   DEFAULT FL


          END
```

Figure 2-B-5. COMPASS Program to Replace Certain Records on a Random File (Sheet 2 of 2)

# CODING SPECIFICATIONS C

All software product documentation produced under NOS follows a prescribed set of standards and specifications. Appendix C briefly describes these standards and specifications.

## INTERNAL AND EXTERNAL DOCUMENTATION

NOS documentation is of two types.

- External — Produced for the general user
- Internal — Describes the internal characteristics of a program (such as register usage, subroutine entry, and exit conditions)

For a detailed explanation of how to obtain internal and external documentation, refer to the description of the DOCMENT control statement in section 7, volume 1.

## DOCUMENT ON COMMENT STATEMENTS

The following rules apply to the format of comment statements for documentation.

- All comment statements begin with an asterisk in column 1. The text of the comment is contained in columns 11 through 72. If a comment statement has an asterisk in column 1 only, it is a continuation of internal and external documentation, or it is a comment statement not included in formal documentation.
- Comment statements with asterisks in columns 1, 2, and 3 indicate that this statement and all following comment statements are internal and external documentation.
- Comment statements with asterisks in columns 1 and 2 indicate internal documentation.
- A statement with four asterisks beginning in column 1 indicates that all following statements are internal documentation (whether they are comment statements or not). Documentation ends when another comment statement containing four asterisks is encountered. This can be used to describe tables.
- All stand-alone comment statements are preceded and followed by a blank line and are terminated by a period.
- Comments describing the function of a section of a code appear on the first line of a sentence.

Figure 2-C-1 is an example of external documentation for program COPYB. Figure 2-C-2 is an example of internal documentation for COPYB.

```
COPYB - BINARY FILE COPIES.
G. R. MANSFIELD.  70/12/20.
J. C. BOHNHOFF. CPD. 73/03/01.
R. F. TATE.  CPD.  73/04/03.



DAYFILE MESSAGES.


* ILLEGAL COUNT.* = OPTIONAL RECORD/FILE COUNT ILLEGAL FORMAT.

* ILLEGAL TERMINATION CONDITION.* = ILLEGAL FORMAT ON
RECORD TERMINATOR FOR *COPYX*.

* END OF INFORMATION ENCOUNTERED.* = END OF INFORMATION WAS
ENCOUNTERED BEFORE THE SPECIFIED COPY OPERATION WAS
COMPLETED.




COPY (IFILE,OFILE,V,C)
COPY FILES FROM MEDIUM TO MEDIUM IN BINARY MODE
THROUGH AN EMPTY FILE.

       IFILE       INPUT FILE NAME.
       OFILE       OUTPUT FILE NAME.
       V           IF PRESENT, REWIND AND VERIFY BOTH FILES.
                   JOB WILL ABORT IF VERIFY ERRORS
       C           COPY IN CODED FORMAT (SI,S,L FORMAT TAPES)

IF IFILE = OFILE, FILES ON IFILE ARE SKIPPED.

ASSUMED PARAMETERS.
       IFILE = *INPUT*
       OFILE = *OUTPUT*
       V NOT PRESENT
       C NOT PRESENT




COPYBF (IFILE,OFILE,N,C)
       IFILE       NAME OF INPUT FILE.
       OFILE       NAME OF OUTPUT FILE.
       N           NUMBER OF FILES TO COPY.
       C           COPY IN CODED FORMAT (SI,S,L FORMAT TAPES)

IF IFILE = OFILE, FILES ON IFILE ARE SKIPPED.

ASSUMED PARAMETERS.
```



Figure 2-C-1. External Documentation of COPYB (Sheet 1 of 3)

```
        IFILE = *INPUT*
        OFILE = *OUTPUT*
        N = 1
        C NOT PRESENT



COPYBR (IFILE,OFILE,N,C)
COPY RECORDS FROM MEDIUM TO MEDIUM IN BINARY MODE.
A FILE MARK IS COUNTED AS A RECORD.

        IFILE        INPUT FILE NAME.
        OFILE        OUTPUT FILE NAME.
        N            NUMBER OF RECORDS TO BE COPIED.
        C            COPY IN CODED FORMAT (SI,S,L FORMAT TAPES)

IF IFILE = OFILE, RECORDS ON IFILE ARE SKIPPED.

ASSUMED PARAMETERS.
        IFILE = *INPUT*
        OFILE = *OUTPUT*
        N = 1
        C NOT PRESENT



COPYEI (IFILE,OFILE,V,C)
COPY FILES FROM MEDIUM TO MEDIUM IN BINARY MODE TO END-OF-
INFORMATION.

        IFILE        INPUT FILE NAME.
        OFILE        OUTPUT FILE NAME.
        V            IF PRESENT, REWIND AND VERIFY BOTH FILES.
                     JOB WILL ABORT IF VERIFY ERRORS.
        C            COPY IN CODED FORMAT (SI,S,L FORMAT TAPES)

IF IFILE = OFILE, IFILE IS SKIPPED TO EOI.

ASSUMED PARAMETERS -
        IFILE = *INPUT*
        OFILE = *OUTPUT*
        V NOT PRESENT.
        C NOT PRESENT



COPYX (IFILE,OFILE,TERM,BKSP,C)
COPYX (IFILE,OFILE,TYPE/NAME,BKSP,C)

COPY RECORDS FROM MEDIUM TO MEDIUM IN BINARY MODE
```

CONTROL DATA CORPORATION DOCUMENTATION.

Figure 2-C-1. External Documentation of COPYB (Sheet 2 of 3)

```
        UNTIL A SPECIFIED TERMINATION CONDITION IS REACHED.
               IFILE        INPUT FILE NAME.
               OFILE        OUTPUT FILE NAME.
               TERM         TERMINATION CONDITION.
                            *00* = ZERO RECORD
                            *N* = N RECORDS
                            *NAME* = NAME OF A RECORD
               TYPE         MNEMONIC FOR RECORD TYPE.
               NAME         RECORD NAME.
               BKSP         BACKSPACE CONTROL.
                            *0* = NO BACKSPACE
                            *1* = BACKSPACE FIRST MEDIUM
                            *2* = BACKSPACE SECOND MEDIUM
                            *3* = BACKSPACE BOTH MEDIA
               C            COPY IN CODED FORMAT (SI,S,L FORMAT TAPES)

        IF IFILE = OFILE, RECORDS ON IFILE ARE SKIPPED.

        ASSUMED PARAMETERS.
               IFILE = *INPUT*
               OFILE = *OUTPUT*
               TERM = 1
               BKSP = 0
               C NOT PRESENT
 CONTROL DATA CORPORATION DOCUMENTATION.
```

Figure 2-C-1. External Documentation of COPYB (Sheet 3 of 3)

```
          COPYB - BINARY FILE COPIES.
          G. R. MANSFIELD.  70/12/20.
          J. C. BOHNHOFF. CPD. 73/03/01.
          R. E. TATE.  CPD.  73/04/03.



          DAYFILE MESSAGES.

          * ILLEGAL COUNT.* = OPTIONAL RECORD/FILE COUNT ILLEGAL FORMAT.

          * ILLEGAL TERMINATION CONDITION.* = ILLEGAL FORMAT ON
          RECORD TERMINATOR FOR *COPYX*.

          * END OF INFORMATION ENCOUNTERED.* = END OF INFORMATION WAS
          ENCOUNTERED BEFORE THE SPECIFIED COPY OPERATION WAS
          COMPLETED.



          ASSEMBLY CONSTANTS.

 BUFL     EQU    100B       WORKING BUFFER LENGTH
 BUFLC    EQU    1003B      WORKING BUFFER LENGTH FOR CONTROL WORDS
 IBUFL    EQU    4011B
 OBUFL    EQU    4011B



          COPY (IFILE,OFILE,V,C)
          COPY FILES FROM MEDIUM TO MEDIUM IN BINARY MODE
          THROUGH AN EMPTY FILE.

                 IFILE      INPUT FILE NAME.
                 OFILE      OUTPUT FILE NAME.
                 V          IF PRESENT, REWIND AND VERIFY BOTH FILES.
                            JOB WILL ABORT IF VERIFY ERRORS
                 C          COPY IN CODED FORMAT (SI,S,L FORMAT TAPES)

          IF IFILE = OFILE, FILES ON IFILE ARE SKIPPED.

          ASSUMED PARAMETERS.
                 IFILE = *INPUT*
```



Figure 2-C-2. Internal Documentation of COPYB (Sheet 1 of 5)

```
          OFILE = *OUTPUT*
          V NOT PRESENT
          C NOT PRESENT



     COPYBF (IFILE,OFILE,N,C)
          IFILE        NAME OF INPUT FILE.
          OFILE        NAME OF OUTPUT FILE.
          N            NUMBER OF FILES TO COPY.
          C            COPY IN CODED FORMAT (SI,S,L FORMAT TAPES)

     IF IFILE = OFILE, FILES ON IFILE ARE SKIPPED.

     ASSUMED PARAMETERS.
          IFILE = *INPUT*
          OFILE = *OUTPUT*
          N = 1
          C NOT PRESENT



     COPYBR (IFILE,OFILE,N,C)
     COPY RECORDS FROM MEDIUM TO MEDIUM IN BINARY MODE.
     A FILE MARK IS COUNTED AS A RECORD.

          IFILF        INPUT FILE NAME.
          OFILE        OUTPUT FILE NAME.
          N            NUMBER OF RECORDS TO BE COPIED.
          C            COPY IN CODED FORMAT (SI,S,L FORMAT TAPES)

     IF IFILE = OFILE, RECORDS ON IFILE ARE SKIPPED.

     ASSUMED PARAMETERS.
          IFILE = *INPUT*
          OFILE = *OUTPUT*
          N = 1
          C NOT PRESENT



     COPYEI (IFILE,OFILE,V,C)
     COPY FILES FROM MEDIUM TO MEDIUM IN BINARY MODE TO END-OF-
     INFORMATION.

          IFILE        INPUT FILE NAME.
          OFILE        OUTPUT FILE NAME.
          V            IF PRESENT, REWIND AND VERIFY BOTH FILES.
                       JOB WILL ABORT IF VERIFY ERRORS.
          C            COPY IN CODED FORMAT (SI,S,L FORMAT TAPES)
```



Figure 2-C-2. Internal Documentation of COPYB (Sheet 2 of 5)

```
IF IFILE = OFILE, IFILE IS SKIPPED TO EOI.

ASSUMED PARAMETERS -
       IFILE = *INPUT*
       OFILE = *OUTPUT*
       V NOT PRESENT.
       C NOT PRESENT



COPYX (IFILE,OFILE,TERM,BKSP,C)
COPYX (IFILE,OFILE,TYPE/NAME,BKSP,C)

COPY RECORDS FROM MEDIUM TO MEDIUM IN BINARY MODE
UNTIL A SPECIFIED TERMINATION CONDITION IS REACHED.
       IFILE        INPUT FILE NAME.
       OFILE        OUTPUT FILE NAME.
       TERM         TERMINATION CONDITION.
                    *00* = ZERO RECORD
                    *N* = N RECORDS
                    *NAME* = NAME OF A RECORD
       TYPE         MNEMONIC FOR RECORD TYPE.
       NAME         RECORD NAME.
       BKSP         BACKSPACE CONTROL.
                    *0* = NO BACKSPACE
                    *1* = BACKSPACE FIRST MEDIUM
                    *2* = BACKSPACE SECOND MEDIUM
                    *3* = BACKSPACE BOTH MEDIA
       C            COPY IN CODED FORMAT (SI,S,L FORMAT TAPES)

IF IFILE = OFILE, RECORDS ON IFILE ARE SKIPPED.

ASSUMED PARAMETERS.
       IFILE = *INPUT*
       OFILE = *OUTPUT*
       TERM = 1
       BKSP = 0
       C NOT PRESENT



CCF - COPY CONTROL WORD FORMAT FILE.

ENTRY  NONE.

EXIT   (X1)< 0 IF EOI ENCOUNTERED.
       (X5)= NUMBER OF RECORDS COPIED.

USES   ALL.
```

CONTROL DATA CORPORATION DOCUMENTATION.

Figure 2-C-2. Internal Documentation of COPYB (Sheet 3 of 5)

```
COPYB - BINARY FILE COPIES.                         75/01/21. INTERNAL

          CALLS  RDW=,MES,WTW=.


          CPR - COPY RECORD.

          ENTRY  (X1) = FIRST BLOCK STATUS.

          EXIT   (X1) < 0 IF EOI.
                 (X1) = 0 IF EOF.

          USES   X - 2.
                 B - NONE.
                 A - 2.

          CALLS  NONE.


          END - END PROGRAM.


          MES - MESSAGE HEADER.
          FINDS RECORD NAME IN *BUF*, ISSUES MESSAGE AND INCREMENTS
          RECORD COUNT.

          ENTRY  (X5) = WORD COUNT IN *BUF*.

          EXIT   (MESA) = RECORD COUNT.
                 (X0) = SAME AS ENTRY.


          SEM - SEND EOI MESSAGE.

          ENTRY  NONE.

          EXIT   (EI) = 1 = MESSAGE SENT.


          BUFFERS.


          PRS - PRESET PROGRAM.
CONTROL DATA CORPORATION DOCUMENTATION.
```

Figure 2-C-2. Internal Documentation of COPYB (Sheet 4 of 5)

```
EXIT    (B7) = REMAINDER ARGUMENT COUNT.
        (A5) = LAST ARGUMENT ADDRESS.
        (X7)= 0 IF COPY CAN BE DONE WITH CONTROL WORDS.


CDT - CHECK DEVICE TYPE.

ENTRY   (X1)= (FET+1).

EXIT    (X7)= 0 IF CONTROL WORD READ/WRITE NOT SUPPORTED ON
         DEVICE.

USES    B - NONE.
        A - 2.
        X - 0,1,2,6,7.

CALLS   NONE.


STC - SET TERMINATION CONDITION.


ERR - PROCESS ERRORS.
CONTROL DATA CORPORATION DOCUMENTATION.
```

Figure 2-C-2. Internal Documentation of COPYB (Sheet 5 of 5)

## SPECIAL DOCUMENTATION STATEMENTS

The following points describe special statements used to produce documentation.

- Documentation statements containing cE in columns 1 and 2 cause the page to be ejected.
- Documentation statements containing cT in columns 1 and 2 produce a table.

The character c preceding an E or T statement is the key character specified by the C parameter on the DOCMENT control statement (the default is an asterisk).

The cE and cT statements are recognized only if they appear within a set of consecutive statements beginning with ccc (external) or cc (internal), where c is the key character.

For example, the statement

    *T EXAMPLE 24/PP PROGRAM, 18/PARAMETER 1, 18/PARAMETER 2

generates the following table.

```
                  5         4         3         2         1
         987654321098765432109876543210987654321098765432109876543210
         ------------------------------------------------------------
EXAMPLE /PP PROGRAM               /PARAMETER 1        /PARAMETER 2        /
         ------------------------------------------------------------
```

The bit position header is generated each time a new block of *T statements is encountered. The header is not listed for consecutive table statements or for any statement containing a nonblank character in column 3 of the first *T statement in a block. The identifier EXAMPLE is optional.

In the statement format shown, a slash must immediately follow a bit count field; however, spaces before the bit count are ignored. All bit counts for field widths can be specified in either octal or decimal. Decimal counts are assumed in the absence of a post radix B. All table entry description statements within an *T block represent the same total number of bits as the first statement. A maximum table width of 60 bits is allowed.

The slash separates fields in the table. The bit position that the slash occupies is included in the field to its left. Single bit fields are not listed with a slash separator. Instead, they have a + below the field position. If only one table entry is listed, the + is listed both above and below the field position.

- All loader control statements (overlay, section, etc.) are considered special, and their images are placed with the page number at the foot of each subprogram to which the directive applies.
- All END statements are considered special since they terminate a chapter.
- In COMPASS, the first TITLE statement has special meaning. Its contents (if nonblank) replace the page header. All subsequent TITLE statements are ignored.
- In COMPASS, the LIST statement has special meaning. The parameters X and L on a COMPASS control statement are processed normally. If a -L is encountered, however, all documentation is suppressed until a LIST, L statement is encountered. If a -X is encountered, no documentation is processed on common test CTEXT until a LIST, X statement is found.

  The maximum number of these LIST statements that can be processed by DOCMENT is 24. If more than 24 LIST X, -X, L, or -L statements are encountered, the following message is issued.

  LIST CARD LIMIT - CARD IGNORED.

- The CTEXT and ENDX statements are bracket statements surrounding common text. No documentation is listed unless a LIST,X statement is encountered.

## COMPASS OPERATION STATEMENTS

COMPASS coding standards are:

- Location field begins in column 2
- Operation field begins in column 11
- Address field begins in column 18
- Comment field is in columns 30 through 72

If a field extends into the next field, two blank columns separate the fields. If a field is filled exactly, one blank separates it from the next field.

Avoid using column 72 when possible. Information in column 72 abuts the line sequence number and causes difficulty in reading comments.

## COMMENT FIELD DOCUMENTATION

Beginning in column 30 of the program instructions, the programmer defines the purpose of the instruction. Every instruction need not be documented in this manner; however, as many comments as possible should be included.

## HEADER DOCUMENTATION

The variable field of TITLE statement is terminated with a period. The assembler inserts the field in the header line of the COMPASS listing. The location field of the SPACE statement and the variable field of the USE statement are also inserted in the header line.

The first two comment statements in the program are the routine name, and the author's name and the date. These are followed by a SPACE 4 statement. For example:

```
***      XXX - PROGRAM NAME AND DESCRIPTION.
*        PROGRAMMER NAME.     YY/MM/DD.
         SPACE  4
```

The body of the documentation includes the:

- Function of the routine
- Method used (if applicable)
- Entry conditions
- Exit conditions
- Messages issued
- Data areas used
- Routines called
- Registers used or saved for each entry point (CP programs)

This information appears in the order specified, with a SPACE 4 statement separating each section. An EJECT or TITLE statement follows this documentation.

Each subroutine has a header describing entry and exit conditions, registers used and routines called. The header is followed by two blank lines and the subroutine text. For example:

```
**            SFS - SEARCH FILE FOR STRING.
*
*             ENTRY  (X0)  =  0  SEARCH BEFORE WRITE/READ.
*                         .NE. 0 WRITE/READ BEFORE SEARCH.
*                    (X6)  =  STRING POINTER ADDRESS.
*
*             EXIT   (X7).NE.  0 IF STRING FOUND.
*                          =  0  IF EOF.
*
*             USES A - 0, 2, 4, 6, 7.
*                  B - 2, 6, 7.
*                  X - 0, 2, 4, 5, 6, 7.
*
*             CALLS  RDC=, SLS, WTC=.
```

If a field is not used, it is omitted rather than specifying NONE. If all of a type of CPU registers are used, list ALL rather than each register. For example:

```
*             USES A - 0, 2, 4, 6.
*                  X - ALL.
```

## PROGRAM TEXT

The program text documentation includes the:

- Local macro definitions.
- Definition of local symbols.
- Main loop of the routine.
- Primary subroutines in the routine: a header describing the entry/exit conditions, registers used or saved, and the memory locations used. This is followed by two blank statements and the subroutine text.
- Secondary and utility subroutines in the routine: a header describing the entry/exit conditions, registers used or saved, and the memory locations used. This is followed by two blank statements and the subroutine text.

### ROUTINE NAME CONVENTIONS

The conventions followed in naming routines are:

- Peripheral processor routine names are three-character mnemonics.
- Central processor routine names are any length up to seven characters, preferably four or more characters to distinguish them from peripheral processor routines.
- Routine names are mnemonics which pertain to the function of the routines.

## RESERVED NAME CONVENTIONS

In the following list, the x parameter represents any legal character, and n is any digit. Any conflict between user name categories and system name categories is resolved in favor of the user name categories.

| | | |
|---|---|---|
| • | Ux | Reserved for user equipment types |
| • | 6xx | Reserved for PP resident-callable mass storage drivers |
| • | 7xx | Reserved for mass storage driver error processing overlays |
| • | 0xx | Reserved for location-free overlays |
| • | Uxx | Reserved for user PP program names (CPU-callable) or for user-defined PP resident entry points |
| • | nUx | Reserved for user PP program names (PPU-callable) or overlay names |
| • | UxxL, UxxM, UxxP, and UxxW | Reserved for user PPCOM symbols |
| • | 9AA - 9Z9 | Used by the system |
| • | 90A - 929 | Used by diagnostics |

## SUBROUTINE TAG CONVENTIONS

Subroutine tags conform to the following standards.

- Subroutine names consist of three-character mnemonics.
- Jump tags consist of the subroutine name with 1 through 99 appended, in sequential order, with no gaps in numbering. The exception to this is when the exit point of a return jump is the subroutine name appended with an X (refer to SUBR macro, section 11). In cases where modifications are made and it becomes necessary to insert new tagged instructions, it is sometimes permissible to use fractional numbers following the tag (for example, PCL1.1). However, this practice is not recommended.
- Local data tags consist of the subroutine name with an A through Z appended, in alphabetical order, with no gaps in lettering.
- System tags (defined when the SST statement is used in the program) are used instead of constants whenever possible. This prevents problems when system routines such as PPR and mass storage drivers are referenced.
- Common decks used must avoid conflict with names in user's programs.

## DATA TAG CONVENTIONS

Data tags conform to the following standards.

- Temporary M core constants (locations other than direct locations in PP code) and data names are four alphabetic characters to distinguish them from direct locations and entry point tags.
- Table tags begin with a T and have a three-character table name appended.
- Table length tags have an L appended to the table tag.
- Peripheral processor direct location tags are two characters in length. Permanent direct location assignments are defined in PPCOM and are called by the SST pseudo-op at the beginning of the program. Other direct locations are defined with the EQU pseudo-op in one block of code at the beginning of the program immediately following the initial documentation.
- Assembly option names, micro names, and names of items used to control code generation are five characters in length.

## GENERAL CONSIDERATION FOR ALL PROGRAMS

The following items apply to all programs.

- Major subroutines are preceded by a TITLE statement (subtitle) so that they are completely separated from other subroutines.
- Each subroutine is headed by comment statements for documentation purposes. This documentation heading is followed by two blank statements and the text of the subroutine.
- Subroutines not on separate pages are separated by SPACE 4 statements. SPACE 4 statements have a tag in the tag field for COMPASS header purposes.
- SPACE 1 and SPACE 2 statements are not used. One or two blank statements are used instead.
- Jumps are not made to addresses like *+3. A tag is used. An exception to this is in very short delay loops. For example:

```
LCN     0       DELAY
SBN     1
NJN     *-1
```

- Comments on jump instructions are the if condition. For example:

```
MJN     TAG     IF TABLE FULL
```

- A comment describing a block of code appears at the beginning of that block.
- All display code data (character data) is in one of the allowable forms for specifying character data. Octal values are not used.
- Numeric data is specified in its natural form (that is, readable and understandable). If conversion considerations make this impossible, the comment field contains the natural form.
- Statement tags should reference actual instructions, not BSS 0 instructions.

- Only one piece of data is specified per line of code. Placing several pieces of data after a DATA pseudo-op makes the listing difficult to read. The same consideration applies to the ENTRY pseudo-op.

- The VFD pseudo-op does not contain more fields than required to generate one word (60 bits) of data. For example:

  ```
  VFD       10/V1,10/V2,10/V3,10/V4,2/V5,18/V6
  ```

  is acceptable while

  ```
  VFD       60/TAG1,60/TAG2
  ```

  is not acceptable.

- The EXT pseudo-op is not used. References to external names are =Xname.

- Macros for code generation are used sparingly to aid readability. When they are used, macros are structured to make use of data already in the registers. Extensive use of macros makes code difficult to read.

- Literals are used for data that is referenced only once.

- Shift counts used for testing bits in a field are of the form:

  ```
  value+bit       or       value-bit
  ```

  where bit is the bit number in the register, word, or field to be tested, and value is whatever expression is necessary to specify the correct shift count. For example, to test the upper bit of byte 1 of the central X3 register (bit 47), instead of using an

  ```
  LX3       12
  ```

  instruction, the following instruction is used.

  ```
  LX3       59-47
  ```

  Further, assume that one desires to test original bit 58 of the shifted X3 register. The following instruction can be used for this purpose.

  ```
  LX3       59+47-58
  ```

## COMMON DECK USAGE

Common decks enable a set of code to be used in several routines. They can be used whenever the code is in a form acceptable to the editing routines (Modify or Update). To insert a common deck into a program, a

```
*CALL     xxxxxxx
```

card is needed, and common deck, xxxxxxx, must be available to the editing routine. The common deck name begins in column 11. For further information concerning common deck usage, refer to section 2 and appendix A.

Common decks are used:

- To increase efficiency in writing code. If there is a need for code which was written for other applications, and the same code can be used in a new application, there is no need to regenerate the same code. It can be inserted into the new program by use of common deck calls and editing programs.

- To ensure uniformity of code. Since the code in the common deck is to be used without modification, it is completely uniform whenever the common deck is used.

- To decrease debugging time. System routines such as CIO have been thoroughly checked out and provide all of the necessary interface with the system, and therefore, need no debugging.

## SYSTEM MACROS

Macros are available whenever a program communicates with another part of the system. For example, when issuing a monitor function in PP code, the macro MONITOR DTKM performs the following.

```
LDN   DTKM
RJM   FTN
```

Extensive CP code macros are available to perform input/output or issue system requests (such as requesting storage or requesting common files).

## SYSTEM INTERFACE RULES

The following rules apply when writing central processor (CP) and peripheral processor (PP) routines for system interface.

- PP routines should be efficient and their use held to a minimum.

- PP programs that interface with CP programs should have some form of validation to ensure proper calling procedures or parameters. An example of this is CIO which checks all buffer parameters before performing any of the tasks requested. Great care should be taken to ensure that errors in arguments set up by the CP program do not cause the PP program to destroy an area other than the local FL of the calling program.

- PP routines waiting for some system resource to become available always pause for storage relocation to allow other system processing to proceed.

- Use of other PPs to assist in performing the assigned task is a practice that should be used only when absolutely necessary and with great caution. The fact that other PPs may not be available must be considered. The situation in which several PPs are waiting for pool processors should be prevented. The only sure way to do this is not to request helper PPs.

- PP overlays can be used without difficulty. The advantage of overlays is that a minimal amount of coding is loaded (from disk or CMR) every time a program is executed. Areas of programs that are used infrequently can be made overlays (for example, error processors). Certain system overlays are available for use by any program and are designed to be location-free (refer to programs OBF and ODF). For use of the overlays, refer to the SEGMENT pseudo-op.

- PP memory cells 70 through 73 and 75 through 77 are set when PPR is loaded. Care should be taken not to destroy these cells, as these locations are not initialized after each program load. Cell 74 (CP) is set by PPR when a program is loaded through an input register request.

- PP routines preparing for input/output to allocatable devices should request mass storage space needed before reserving the channel. Also, when input/output is complete, the channel should be dropped before indicating which area of mass storage was not used.

- PP routines doing input/output to a device should perform all housekeeping possible before operation is initiated. Unnecessary reservation of channels prevents other programs from using that channel for input/output.

- Buffers used by PP and CP programs should be defined by EQU statements at the end of the program text and not made part of the text by using BSS statements. This eliminates loading of the core area.

- PP programs or overlays are loaded from disk. The last sector of a program or overlay must be loaded at an address above that which causes wrap-around.

# PROGRAM EXAMPLE D

Appendix D contains a listing of the system program COPYB. This listing is supplied for two basic reasons.

- To illustrate the use of many of the system macros described in this manual
- To illustrate the format and documentation standards used in system programs

## COPYB — BINARY FILE COPIES

```
COPYB - BINARY FILE COPIES.


          IDENT  COPYB,FETS
          ABS
          ENTRY  COPY
          ENTRY  COPYBF
          ENTRY  COPYBR
          ENTRY  COPYEI
          ENTRY  COPYX
          ENTRY  RFL=
          SYSCOM B1              DEFINE (B1) = 1
  COMMENT 73/05/24. 74/11/23. BINARY FILE COPIES.
          COMMENT COPYRIGHT CONTROL DATA CORP. 1970.


***       COPYB - BINARY FILE COPIES.
*         G. R. MANSFIELD.  70/12/20.
*         J. C. BOHNHOFF. CPD. 73/03/01.
*         R. E. TATE.  CPD.  73/04/03.


***       DAYFILE MESSAGES.
*
*
*         * ILLEGAL COUNT.* = OPTIONAL RECORD/FILE COUNT ILLEGAL FORMAT.
*
*         * ILLEGAL TERMINATION CONDITION.* = ILLEGAL FORMAT ON
*         RECORD TERMINATOR FOR *COPYX*.
*
*         * END OF INFORMATION ENCOUNTERED.* = END OF INFORMATION WAS
*         ENCOUNTERED BEFORE THE SPECIFIED COPY OPERATION WAS
*         COMPLETED.
```

```
COPYB - BINARY FILE COPIES.
COMMON DATA

****      ASSEMBLY CONSTANTS.


 BUFL     EQU     100B        WORKING BUFFER LENGTH
 BUFLC    EQU     1003B       WORKING BUFFER LENGTH FOR CONTROL WORDS
 IBUFL    EQU     4011B
 OBUFL    EQU     4011B
****
                               COMPASS 3.74259.     75/01/20. 15.20.14.
                               DATA


          ORG     120B
 FETS     BSS     0

 I        BSS     0
 INPUT    RFILEB IBUF,IBUFL,(FET=8)

 O        BSS     0
 OUTPUT   RFILEB OBUF,OBUFL,(FET=8)

 CT       CON     1           COUNT
 EI       CON     0           END OF INFORMATION MESSAGE SENT
 SK       CON     0           SKIP FLAG
 VF       CON     0           VERIFY FLAG

 TM       CON     1S59        COPYX TERMINATION
 RN       CON     0           RECORD NAME
          CON     1S59        RECORD TYPE
 BK1      CON     0           FILE 1 BACKSPACE
 BK2      CON     0           FILE 2 BACKSPACE
```

COPYB - BINARY FILE COPIES.
MAIN PROGRAMS.

```
***        COPY (IFILE,OFILE,V,C)
*          COPY FILES FROM MEDIUM TO MEDIUM IN BINARY MODE
*          THROUGH AN EMPTY FILE.
*
*                 IFILE        INPUT FILE NAME.
*                 OFILE        OUTPUT FILE NAME.
*                 V            IF PRESENT, REWIND AND VERIFY BOTH FILES.
*                              JOB WILL ABORT IF VERIFY ERRORS
*                 C            COPY IN CODED FORMAT (SI,S,L FORMAT TAPES)
*
*          IF IFILE = OFILE, FILES ON IFILE ARE SKIPPED.
*
*          ASSUMED PARAMETERS.
*                 IFILE = *INPUT*
*                 OFILE = *OUTPUT*
*                 V NOT PRESENT
*                 C NOT PRESENT


 COPY      SB1    1            (B1) = 1
           RJ     PRS          PRESET PROGRAM
           BX5    X7           SAVE CONTROL WORD FLAG
           ZR     B7,CPY1      IF NO VERIFY REQUESTED
           SA2    ARGR+2
           ZR     X2,CPY1      IF NULL PARAMETER
           SX6    B1           SET VERIFY FLAG
           SA6    VF
           REWIND I
           REWIND O

 CPY1      NZ     X5,CPY4      IF CONTROL WORD COPY

*          COPY RECORDS.

 CPY2      SX5    0            CLEAR EOF FLAG
 CPY3      READ   I            BEGIN READ
           RECALL O
           WRITE  O,*          PRESET WRITE FUNCTION
           READW  I,BUF,BUFL   READ FIRST BLOCK
           RJ     CPR          COPY RECORD
           ZR     X1,CPY2      IF NOT EOF/EOI
           NG     X1,CPY7      IF EOI
           NZ     X5,CPY7      IF LAST RECORD WAS EOF
           SX5    1            SET EOF FLAG
           JP     CPY3         LOOP TO EMPTY FILE/EOI

*          COPY FILES WITH CONTROL WORDS.

 CPY4      RECALL O            PRESET WRITE FUNCTION
           WRITECW O,*
 CPY5      READCW I,17B        BEGIN READ
           RJ     CCF          COPY CONTROL WORD FILE
           NG     X1,CPY6      IF EOI
           ZR     X5,CPY7      IF EMPTY FILE
           WRITEW O,CPYC,B1+B1  WRITE END OF FILE TO BUFFER
           JP     CPY5
```

```
 CPY6      RJ     SEM             SEND EOI MESSAGE

*          PROCESS VERIFY OPTION FOR COPY AND COPYEI.

 CPY7      SA2    VF              CHECK VERIFY REQUESTED
           ZR     X2,END          IF NO VERIFY REQUESTED
           RECALL I
           RECALL O
           MX0    42
           SA1    I
           BX6    X0*X1
           SA6    ARGR
           SA1    CPYA            COPY ARGUMENTS
           BX6    X1
           SA6    ARGR+2
 CPY8      SA1    A1+B1
           BX6    X1
           SA6    A6+B1
           NZ     X1,CPY8         IF MORE ARGUMENTS
           SX7    6               ARGUMENT COUNT = 6
           SA7    ACTR
           MESSAGE CPYB,1,R
           SYSTEM LDR,R,CPYB      EXECUTE VERIFY
           PS     0

 CPYA      CON    1LN+1R=
           CON    0LO
           CON    0LA
           CON    0LR
           CON    0               END OF ARGUMENTS

 CPYB      CON    0LVERIFY
           CON    1408S36

 CPYC      CON    0               END OF FILE CONTROL WORDS
           VFD    12/17B,48/0



***        COPYBF (IFILE,OFILE,N,C)
*                 IFILE           NAME OF INPUT FILE.
*                 OFILE           NAME OF OUTPUT FILE.
*                 N               NUMBER OF FILES TO COPY.
*                 C               COPY IN CODED FORMAT (SI,S,L FORMAT TAPES)
*
*          IF IFILE = OFILE, FILES ON IFILE ARE SKIPPED.
*
*          ASSUMED PARAMETERS.
*                 IFILE = *INPUT*
*                 OFILE = *OUTPUT*
*                 N = 1
*                 C NOT PRESENT


 COPYBF    SB1    1               (B1) = 1
           RJ     PRS             PRESET PROGRAM
```

```
          SA7     CBFA          SAVE CONTROL WORD FLAG
          ZR      B7,CBF1       IF NO 3RD ARGUMENT
          SA2     ARGR+2
          ZR      X2,CBF1       IF NULL PARAMETER
          SA5     A5+1          CONVERT FILE COUNT
          RJ      DXB
          NZ      X4,ERR1       IF ILLEGAL COUNT
          ZR      X6,ERR1       IF COUNT = 0
          SA6     CT            SET FILE COUNT

 CBF1     SA1     CBFA          CHECK CONTROL WORD COPY
          NZ      X1,CBF3       IF CONTROL WORDS

*         COPY RECORDS.

 CBF2     READ    I             BEGIN READ
          RECALL  O
          WRITE   O,*           PRESET WRITE FUNCTION
          READW   I,BUF,BUFL    COPY RECORD
          RJ      CPR
          ZR      X1,CBF2       IF NOT EOF/EOI
          NG      X1,END        IF EOI
          SA2     CT            DECREMENT FILE COUNT
          SX6     X2-1
          SA6     A2+
          NZ      X6,CBF2       LOOP FOR REQUESTED FILE COUNT
          JP      END

*         COPY FILES WITH CONTROL WORDS.

 CBF3     WRITECW O,*           PRESET WRITE FUNCTION
 CBF4     READCW I,17B          BEGIN READ
          RJ      CCF           COPY CONTROL WORD FILE
          SA2     SK
          SX5     X1
          NZ      X2,CBF5       IF SKIPPING
          WRITEW O,CPYC,B1+B1   WRITE EOF
          WRITECW O             FLUSH BUFFER
 CBF5     NG      X5,CBF6       IF EOI
          SA2     CT            DECREMENT FILE COUNT
          SX6     X2-1
          SA6     A2+
          ZR      X6,END        IF ALL FILES COPIED
          JP      CBF4

 CBF6     RJ      SEM           SEND EOI MESSAGE
          JP      END1

 CBFA     CON     0


***       COPYBR (IFILE,OFILE,N,C)
*         COPY RECORDS FROM MEDIUM TO MEDIUM IN BINARY MODE.
*         A FILE MARK IS COUNTED AS A RECORD.
*
```

COPYB - BINARY FILE COPIES.
MAIN PROGRAMS.

```
*               IFILE          INPUT FILE NAME.
*               OFILE          OUTPUT FILE NAME.
*               N              NUMBER OF RECORDS TO BE COPIED.
*               C              COPY IN CODED FORMAT (SI,S,L FORMAT TAPES)
*
*         IF IFILE = OFILE, RECORDS ON IFILE ARE SKIPPED.
*
*         ASSUMED PARAMETERS.
*               IFILE = *INPUT*
*               OFILE = *OUTPUT*
*               N = 1
*               C NOT PRESENT


 COPYBR   SB1    1              (B1) = 1
          RJ     PRS            PRESET PROGRAM
          ZR     B7,CBR1        IF NO 3RD ARGUMENT
          SA2    ARGR+2
          ZR     X2,CBR1        IF NULL PARAMETER
          SA5    A5+B1          CONVERT COUNT
          RJ     DXB
          NZ     X4,ERR1        IF ILLEGAL COUNT
          ZR     X6,ERR1        IF COUNT = 0
          SA6    CT             SET COUNT

 CBR1     READ   I              BEGIN READ
          RECALL O
          WRITE  O,*            PRESET WRITE FUNCTION
          READW  I,BUF,BUFL
          RJ     CPR            COPY RECORD
          NG     X1,END         IF EOI
          SA2    CT             DECREMENT COUNT
          SX6    X2-1
          SA6    A2
          NZ     X6,CBR1        LOOP FOR ALL RECORDS
          EQ     END            TERMINATE PROGRAM



***       COPYEI (IFILE,OFILE,V,C)
*         COPY FILES FROM MEDIUM TO MEDIUM IN BINARY MODE TO END-OF-
*         INFORMATION.
*
*               IFILE          INPUT FILE NAME.
*               OFILE          OUTPUT FILE NAME.
*               V              IF PRESENT, REWIND AND VERIFY BOTH FILES.
*                              JOB WILL ABORT IF VERIFY ERRORS.
*               C              COPY IN CODED FORMAT (SI,S,L FORMAT TAPES)
*
*         IF IFILE = OFILE, IFILE IS SKIPPED TO EOI.
*
*         ASSUMED PARAMETERS -
*               IFILE = *INPUT*
*               OFILE = *OUTPUT*
*               V NOT PRESENT.
*               C NOT PRESENT
```

```
COPYEI   SB1     1              (B1)= 1
         RJ      PRS            PRESET PROGRAM
         BX5     X7             SAVE CONTROL WORD FLAG
         ZR      B7,CEI1        IF NO VERIFY REQUESTED
         SA2     ARGR+2
         ZR      X2,CEI1        IF NULL PARAMETER
         SX6     B1             SET VERIFY FLAG
         SA6     VF
         SA2     =8L999999      SET INFINITE FILE COUNT
         BX6     X2
         SA6     CPYA+1
         REWIND I
         REWIND O

CEI1     NZ      X5,CEI3        IF CONTROL WORD COPY

*        COPY RECORDS.

CEI2     READ    I              BEGIN READ
         RECALL O
         WRITE   O,*            PRESET WRITE FUNCTION
         READW   I,BUF,BUFL     READ FIRST BLOCK
         RJ      CPR            COPY RECORD
         PL      X1,CEI2        IF NOT EOI
         JP      CEI4

*        COPY FILES WITH CONTROL WORDS.

CEI3     RECALL O
         WRITECW O,*            PRESET WRITE FUNCTION
         READCW I               BEGIN READ
         RJ      CCF            COPY CONTROL WORD FILE

*        CHECK VERIFY OPTION.

CEI4     SA2     VF
         ZR      X2,END         IF NO VERIFY REQUESTED
         JP      CPY7           PROCESS VERIFY CALL



***      COPYX (IFILE,OFILE,TERM,BKSP,C)
*        COPYX (IFILE,OFILE,TYPE/NAME,BKSP,C)
*
*        COPY RECORDS FROM MEDIUM TO MEDIUM IN BINARY MODE
*        UNTIL A SPECIFIED TERMINATION CONDITION IS REACHED.
*               IFILE          INPUT FILE NAME.
*               OFILE          OUTPUT FILE NAME.
*               TERM           TERMINATION CONDITION.
*                              *00* = ZERO RECORD
*                              *N* = N RECORDS
*                              *NAME* = NAME OF A RECORD
*               TYPE           MNEMONIC FOR RECORD TYPE.
*               NAME           RECORD NAME.
```

```
*               BKSP        BACKSPACE CONTROL.
*                           *0* = NO BACKSPACE
*                           *1* = BACKSPACE FIRST MEDIUM
*                           *2* = BACKSPACE SECOND MEDIUM
*                           *3* = BACKSPACE BOTH MEDIA
*               C           COPY IN CODED FORMAT (SI,S,L FORMAT TAPES)
*
*         IF IFILE = OFILE, RECORDS ON IFILE ARE SKIPPED.
*
*         ASSUMED PARAMETERS.
*               IFILE = *INPUT*
*               OFILE = *OUTPUT*
*               TERM = 1
*               BKSP = 0
*               C NOT PRESENT


 COPYX    SB1     1           (B1) = 1
          RJ      PRS         PRESET PROGRAM
          ZR      B7,CPX1     IF NO 3RD ARGUMENT
          SA2     ARGR+2
          ZR      X2,CPX1     IF NULL PARAMETER
          RJ      STC         SET TERMINATION CONDITION

 CPX1     READ    I           BEGIN READ
          RECALL  O
          WRITE   O,*         PRESET WRITE FUNCTION
          READW   I,BUF,BUFL
          PL      X1,CPX2     IF NO EOF
          SX7     1
          SA7     EI
          MESSAGE  (=C* END OF FILE ENCOUNTERED.*)
          SA2     SK
          NZ      X2,END      IF SKIPPING
          WRITEF  O
          EQ      END

 CPX2     BX6     X1          SAVE EOR STATUS
          SA2     TM          CHECK TERMINATION CONDITION
          SA6     CPXA

*         PROCESS ZERO RECORD.

          NZ      X2,CPX3     IF NOT ZERO RECORD REQUEST
          SB6     BUF         CHECK WORD COUNT OF READ
          SB7     X1
          SX6     B1
          NE      B6,B7,CPX5  IF NOT ZERO RECORD
          SA6     CT          SET TERMINATION
          EQ      CPX5

*         PROCESS RECORD NAME.

 CPX3     NG      X2,CPX5     IF NOT RECORD NAME
          SX2     BUF
          RJ      SRT         SET RECORD TYPE
          SA1     RN          CHECK RECORD NAME
```

```
          BX2     X7-X1
          SA3     A1+B1
          NZ      X2,CPX5         IF NO MATCH
          NG      X3,CPX4         IF NO TYPE REQUESTED
          SX2     X6              CHECK TYPE
          BX7     X2-X3
          NZ      X7,CPX5         IF NO MATCH
 CPX4     SX6     B1              SET TERMINATION
          SA6     CT

*         COPY RECORD.

 CPX5     SA1     CPXA            RESTORE EOR STATUS
          RJ      CPR             COPY RECORD
          SA1     CT              DECREMENT COUNT
          SX6     X1-1
          SA6     A1
          NZ      X6,CPX1         LOOP FOR ALL RECORDS
          SA1     BK1
          ZR      X1,CPX6         IF NO BACKSPACE FOR FILE 1
          BKSP    I
 CPX6     SA2     SK
          SA1     BK2
          NZ      X2,END          IF SKIPPING RECORDS
          ZR      X1,END          IF NO BACKSPACE FOR FILE 2
          BKSP    O
          EQ      END

 CPXA     CON     0
```

```
**        CCF - COPY CONTROL WORD FORMAT FILE.
*
*         ENTRY  NONE.
*
*         EXIT   (X1)< 0 IF EOI ENCOUNTERED.
*                (X5)= NUMBER OF RECORDS COPIED.
*
*         USES   ALL.
*
*         CALLS  RDW=,MES,WTW=.


 CCF10.1  ZR     X0,CCF11     IF NO DATA IN BUFFER
          SA4    O+4
          BX5    X1
          SX3    5
          AX4    18
          IX7    X0*X3
          SX4    X4
          LX4    36
          BX6    X7+X4
          SA6    BUF
          WRITEW 0,A6,X0+2
 CCF11    SX0    X1+1
          SA1    SK
          NZ     X1,CCF12     IF SKIPPING
          WRITECW 0           FLUSH BUFFER
 CCF12    SX1    X0+
          SA5    MESA         SET RECORD COUNT

 CCF      PS                  ENTRY/EXIT
          SA1    CCFA         CHECK MEDIUM FLAG
          SX6    B0           CLEAR RECORD COUNT
          SX7    -B1          SET EOR FLAG
          SA6    MESA
          SA7    CCFB
          NG     X1,CCF4      IF NOT TAPE TO TAPE OR DISK TO DISK

*         COPY RECORD DISK TO DISK OR TAPE TO TAPE.

 CCF1     READW  I,BUF,B1     READ CONTROL WORD
          NZ     X1,CCF11     IF EOF/EOI
          SA1    BUF          FIND BLOCK SIZE
          SX2    5
          SX3    X1+4         ROUND UP
          IX5    X3/X2
          READW  I,BUF+1,X5+B1 READ FIRST BLOCK
          SA1    CCFB
          PL     X1,CCF2      IF LAST BLOCK NOT EOR
          RJ     MES          DISPLAY RECORD NAME
 CCF2     SA1    SK
          NZ     X1,CCF3      IF SKIP SET
          WRITEW 0,BUF,X5+2
 CCF3     SA1    BUF          SET EOR FLAG
          AX1    36
          SX2    X1
          IX6    X5-X1
```

```
          SA6    CCFB
          JP     CCF1

*         COPY DISK TO TAPE OR TAPE TO DISK.

 CCF4     SX0    B0+
 CCF5     READW  I,BUF,B1      READ CONTROL WORD
          NZ     X1,CCF10.1    IF EOF/EOI
          SA1    BUF           FIND WORD COUNT
          SX2    5
          SX3    X1+4          ROUND UP
          IX5    X3/X2         WORD COUNT IN BLOCK
          READW  I,BUF+1+X0,X5+B1  READ REST OF BLOCK
          SA1    CCFB
          PL     X1,CCF6       IF LAST BLOCK NOT EOR
          RJ     MES           DISPLAY RECORD NAME
 CCF6     SA1    BUF
          SA2    O+4           GET BLOCK SIZE FOR FILE
          IX0    X0+X5         INCREMENT BUFFER WORD COUNT
          AX1    36
          IX6    X5-X1
          AX2    18
          SX3    X2
          IX4    X0-X3
          SA6    CCFB          SAVE EOR FLAG
          NG     X6,CCF7       IF EOR
          NG     X4,CCF5       IF DATA READ .LT. OUTPUT BLOCK SIZE
 CCF7     SA5    SK            PRESET WORDS WRITTEN
          NZ     X5,CCF4       IF SKIPPING
 CCF8     SA1    O+4           GET BLOCK SIZE FOR OUTPUT
          IX2    X0-X5         REMAINING WORDS TO WRITE
          AX1    18
          NZ     X2,CCF9       IF MORE WORDS TO WRITE
          SA3    CCFB
          PL     X3,CCF4       IF NOT EOR ON INPUT
 CCF9     SX1    X1            SET OUTPUT BLOCK SIZE
          IX3    X2-X1         WORDS REMAINING - BLOCK SIZE
          NG     X3,CCF10      IF EOR BLOCK
          BX4    X1            BUILD CONTROL WORD
          LX1    36
          SX3    5             CALCULATE BYTE COUNT
          IX7    X4*X3
          BX6    X1+X7         CONTROL WORD
          SA6    BUF+X5        STORE CONTROL WORD
          SX4    X4
          IX5    X5+X4         UPDATE WORD COUNT
          WRITEW O,A6,X4+B1
          WRITEW O,CCFC,B1     WRITE TRAILER CONTROL WORD
          JP     CCF8

 CCF10    LX1    36
          SX3    5
          IX4    X2*X3         CALCULATE BYTE COUNT
          BX6    X1+X4         CONTROL WORD
          SA6    BUF+X5        STORE CONTROL WORD
          WRITEW O,A6,X2+2
          JP     CCF4
```

```
 CCFA      CON     0               MEDIUMS COMPARISION FLAG
 CCFB      CON     -0              EOR FLAG
 CCFC      CON     0               CONTROL WORD



**         CPR - COPY RECORD.
*
*          ENTRY   (X1) = FIRST BLOCK STATUS.
*
*          EXIT    (X1) < 0 IF EOI.
*                  (X1) " 0 IF EOF.
*
*          USES    X - 2.
*                  B - NONE.
*                  A - 2.
*
*          CALLS   NONE.


 CPR       PS                      ENTRY/EXIT
           NG      X1,CPR5         IF EOF
           SX2     BUF
           BX6     X1              SAVE EOR STATUS
           SA6     CPRA
           RJ      SRT             SET RECORD TYPE
           SA7     CPRB+1          ENTER NAME IN MESSAGE
           MESSAGE A7-B1,1
           SA1     CPRA            RESTORE EOR STATUS
           NZ      X1,CPR3         IF EOR
 CPR1      SA2     SK
           NZ      X2,CPR2         IF SKIP SET
           WRITEW  O,BUF,BUFL
 CPR2      READW   I,BUF,BUFL
           ZR      X1,CPR1         LOOP IF NO EOR/EOF
           NG      X1,CPR5         IF EOF

 CPR3      SA2     SK
           NZ      X2,CPR4         IF SKIP SET
           WRITEW  O,BUF,X1-BUF
           WRITER  O               END RECORD
 CPR4      SX1     B0              RETURN WITH EOR
           EQ      CPR

 CPR5      SA2     SK
           NZ      X2,CPR6         IF SKIP SET
           WRITEF  O
 CPR6      SA2     I               CHECK FILE STATUS
           LX2     59-9
           SX1     B1              SET EOF
           PL      X2,CPR          RETURN IF NOT EOI
           RJ      SEM             SEND EOI MESSAGE
           SX1     -B1             SET EOI
           EQ      CPR
```

```
COPYB - BINARY FILE COPIES.
SUBROUTINES.


 CPRA      CON    0             EOR STATUS

 CPRB      DATA   10H  COPYING
           CON    0,0




**         END - END PROGRAM.


 END       SA1    EI
           NZ     X1,END1       IF EOI MESSAGE SENT
           MESSAGE  (=C* COPY COMPLETE.*)
 END1      ENDRUN



**         MES - MESSAGE HEADER.
*          FINDS RECORD NAME IN *BUF*, ISSUES MESSAGE AND INCREMENTS
*          RECORD COUNT.
*
*          ENTRY  (X5) = WORD COUNT IN *BUF*.
*
*          EXIT   (MESA) = RECORD COUNT.
*                 (X0) = SAME AS ENTRY.


 MES       PS                   ENTRY/EXIT
           SA1    MESA          INCREMENT RECORD COUNT
           BX6    X0            SAVE X0
           SX7    X1+B1
           SA6    MESB
           SX1    X5            WORD COUNT IN *BUF*
           SA7    A1
           SX2    BUF+1
           RJ     SRT
           SA7    CPRB+1
           MESSAGE A7-B1,1      DISPLAY RECORD NAME
           SA1    MESB
           BX0    X1
           JP     MES

 MESA      CON    0             RECORD COUNT
 MESB      CON    0



**         SEM - SEND EOI MESSAGE.
*
*          ENTRY  NONE.
*
*          EXIT   (EI) = 1 = MESSAGE SENT.
```

```
COPYB - BINARY FILE COPIES.
SUBROUTINES.


 SEM      PS      0              ENTRY/EXIT
          SX7     1
          SA7     EI
          MESSAGE  (=C* END OF INFORMATION ENCOUNTERED.*)
          EQ      SEM            RETURN



*         COMMON DECKS.


 WRIF$    EQU     1              SELECT *RE-ISSUE CURRENT WRITE*
          CTEXT   COMCCIO - I/O FUNCTION PROCESSOR.
          CTEXT   COMCRDW - READ WORDS TO WORKING BUFFER.
          CTEXT   COMCSRT - SET RECORD TYPE.
          CTEXT   COMCSYS - PROCESS SYSTEM REQUEST.
          CTEXT   COMCWTW - WRITE WORDS FROM WORKING BUFFER.



**        BUFFERS.


 BUFFERS  BSS     0
          USE     //
          SEG
 BUF      BSS     BUFLC
 IBUF     BSS     IBUFL
 OBUF     BSS     OBUFL
 RFL=     BSS     0



**        PRS - PRESET PROGRAM.
*
*         EXIT    (B7) = REMAINDER ARGUMENT COUNT.
*                 (A5) = LAST ARGUMENT ADDRESS.
*                 (X7)= 0 IF COPY CAN BE DONE WITH CONTROL WORDS.


          ORG     BUF

 PRS3     OPEN    I,READNR,R     CHECK CONTROL WORD COPY POSSIBLE
          OPEN    O,ALTERNR,R
          SA1     I+1            CHECK INPUT DEVICE TYPE
          SA2     O+1
          BX6     X2-X1          SET DEVICE COMPARSION FLAG
          SA6     CCFA
          RJ      CDT
          ZR      X7,PRS         RETURN - IF CONTROL WORDS NOT SUPPORTED
          SA1     O+1            CHECK OUTPUT DEVICE TYPE
          RJ      CDT
```

```
 PRS      PS                      ENTRY/EXIT
          SX6     IBUF            ENTER POINTER TO INPUT BUFFER
          SA6     0
          SA1     ACTR            CHECK ARGUMENT COUNT
          MX4     42
          SB7     X1
          ZR      B7,PRS3         IF NO ARGUMENTS
          SB6     B7-ARGR-2
          NG      B6,PRS0.2       IF NO CODED PARAMETER
          SA1     PRS             CHECK IF COPYX CALL
          AX1     30
          SX2     X1-COPYX-1
          NZ      X2,PRS0.1       IF NOT COPYX
          ZR      B6,PRS0.2       IF NO CODED PARAMETER
          SA1     ARGR+2          CHECK FOR COPYX *TYPE/NAME* PARAMETER
          SX2     X1-1R/
          SB6     B6-B1
          NZ      X2,PRS0.1       IF COPYX *TERM* PARAMETER
          SB6     B6-B1
          NG      B6,PRS0.2       IF NO CODED PARAMETER
 PRS0.1   SA1     I
          SA2     0
          SX3     B1+B1
          BX6     -X3*X1
          BX7     -X3*X2
          SA6     A1
          SA7     A2+

*         PROCESS IFILE NAME.

 PRS0.2   SA5     ARGR            GET FILE NAME
          SA2     I
          BX7     X4*X5
          SX3     X2
          ZR      X7,PRS1         IF BLANK ARGUMENT
          IX7     X7+X3
          SA7     A2

*         PROCESS OFILE NAME.

 PRS1     SB7     B7-B1
          ZR      B7,PRS2         IF 1 ARGUMENT
          SA5     A5+B1           SET OFILE NAME
          SA2     0
          BX7     X4*X5
          SB7     B7-B1
          ZR      X7,PRS2         IF BLANK ARGUMENT
          IX7     X7+X3
          SA7     A2

*         CHECK FILE NAMES.

 PRS2     SA1     I               CHECK FILE NAMES
          SA2     0
          IX7     X1-X2
          NZ      X7,PRS3         IF IFILE " OFILE
          SX6     B1              SET SKIP FLAG
```

```
          SA1     PRSA
          BX7     X1
          SA6     SK
          SA7     CPRB
          JP      PRS3

 PRSA     DATA    10H SKIPPING



**        CDT - CHECK DEVICE TYPE.
*
*         ENTRY   (X1)= (FET+1).
*
*         EXIT    (X7)= 0 IF CONTROL WORD READ/WRITE NOT SUPPORTED ON
*                  DEVICE.
*
*         USES    B - NONE.
*                 A - 2.
*                 X - 0,1,2,6,7.
*
*         CALLS   NONE.


 CDT2     LX1     12                CHECK *TT*
          BX6     -X0*X1
          SX7     X6-2RTT

 CDT      PS                        ENTRY/EXIT
          MX0     -12
          PL      X1,CDT2           IF ALLOCATABLE
          LX1     12
          SA2     CDTA              SEARCH DEVICE TABLE
          SX7     0                 ASSUME NO FIND
 CDT1     ZR      X2,CDT            RETURN - IF NOT FOUND
          BX6     X1-X2
          AX2     12
          BX6     X2*X6
          SA2     A2+B1
          NZ      X6,CDT1           IF NOT MATCH
          SX7     1                 INDICATE CONTROL WORD POSSIBLE
          JP      CDT               RETURN

 CDTA     VFD     36/,12/7703B,12/4082B
          VFD     36/,12/7703B,12/4102B
          VFD     36/,12/7777B,12/2RMT+4000B
          VFD     36/,12/7777B,12/2RNT+4000B
          CON     0



**        STC - SET TERMINATION CONDITION.


 STC      PS                        ENTRY/EXIT
```

```
         SA5   A5+B1         TERMINATION CONDITION
         SA1   =2L00
         BX6   X1-X5
         ZR    X5,STC5       IF BLANK ARGUMENT
         ZR    X6,STC4       IF *00*
         RJ    DXB           CONVERT NUMBER
         NZ    X4,STC1       IF ASSEMBLY ERROR
         ZR    X6,ERR1       IF COUNT = 0
         SA6   CT            SET COUNT
         EQ    STC5
STC1     SA5   A5            SET NAME
         MX0   42
         SX3   X5-1R/        CHECK SEPARATOR
         BX6   X0*X5
         MX7   1
         NZ    X3,STC3       IF NO TYPE SPECIFIED
         SB7   B7-B1
         SA5   A5+B1         RECORD NAME
         LX3   X6
         SA2   STCA          CHECK TYPE
         BX6   X0*X5
STC2     ZR    X2,ERR2       IF TYPE NOT IDENTIFIED
         BX7   X2-X3
         SA2   A2+B1
         NZ    X7,STC2
         SX7   A2-STCA-1     SET TYPE
STC3     SA6   RN            SET RECORD NAME
         SA7   A6+B1         SET TYPE
         SX6   B1
STC4     SA6   TM            SET TERMINATION CONDITION
         MX7   1             SET HIGH COUNT
         SA7   CT

*        PROCESS BACKSPACE CONTROL.

STC5     SB7   B7-B1
         ZR    B7,STC        RETURN IF 3 ARGUMENTS
         SA1   A5+B1         CHECK BACKSPACE ARGUMENT
         AX1   54
         SB2   X1-1R0
         ZR    X1,STC        IF BLANK ARGUMENT
         ZR    B2,STC        RETURN IF ZERO
         NG    B2,ERR2       IF ALPHA
         SX6   B0
         SX7   B0
         SB2   B2-B1
         NZ    B2,STC6       IF NOT *1*
         SX6   B1            SET FILE 1 BACKSPACE
STC6     SB2   B2-B1
         NZ    B2,STC7       IF NOT *2*
         SX7   B1            SET FILE 2 BACKSPACE
STC7     SB2   B2-B1
         NZ    B2,STC8       IF NOT *3*
         SX6   B1            SET BOTH FILES BACKSPACE
         SX7   B1
STC8     SA6   BK1
         SA7   BK2
```

```
COPYB - BINARY FILE COPIES.
SUBROUTINES.


          EQ      STC          RETURN

 STCA     BSS     0

          CON     0LTEXT
          CON     0LPP
          CON     0LCOS
          CON     0LREL
          CON     0LOVL
          CON     0LULIB
          CON     0LOPL
          CON     0LOPLC
          CON     0LOPLD
          CON     0LABS
          CON     0LPPU
          CON     0


**        ERR - PROCESS ERRORS.


 ERR1     SX0     ERRA
          EQ      ERR

 ERR2     SX0     ERRB

 ERR      MESSAGE X0
          ABORT
 ERRA     DATA    C* ILLEGAL COUNT.*

 ERRB     DATA    C* ILLEGAL RECORD TERMINATION.*


*         COMMON DECKS.


          CTEXT   COMCDXB - DISPLAY CODE TO BINARY CONVERSION.


          END
```

Appendix E provides special information available to the applications programmer. The following topics are described.

- Job communication area
- Exchange package area
- File types and job origin codes
- Equipment codes

## JOB COMMUNICATION AREA

Figure 2-E-1 illustrates the first $101_8$ words of the user's field length.

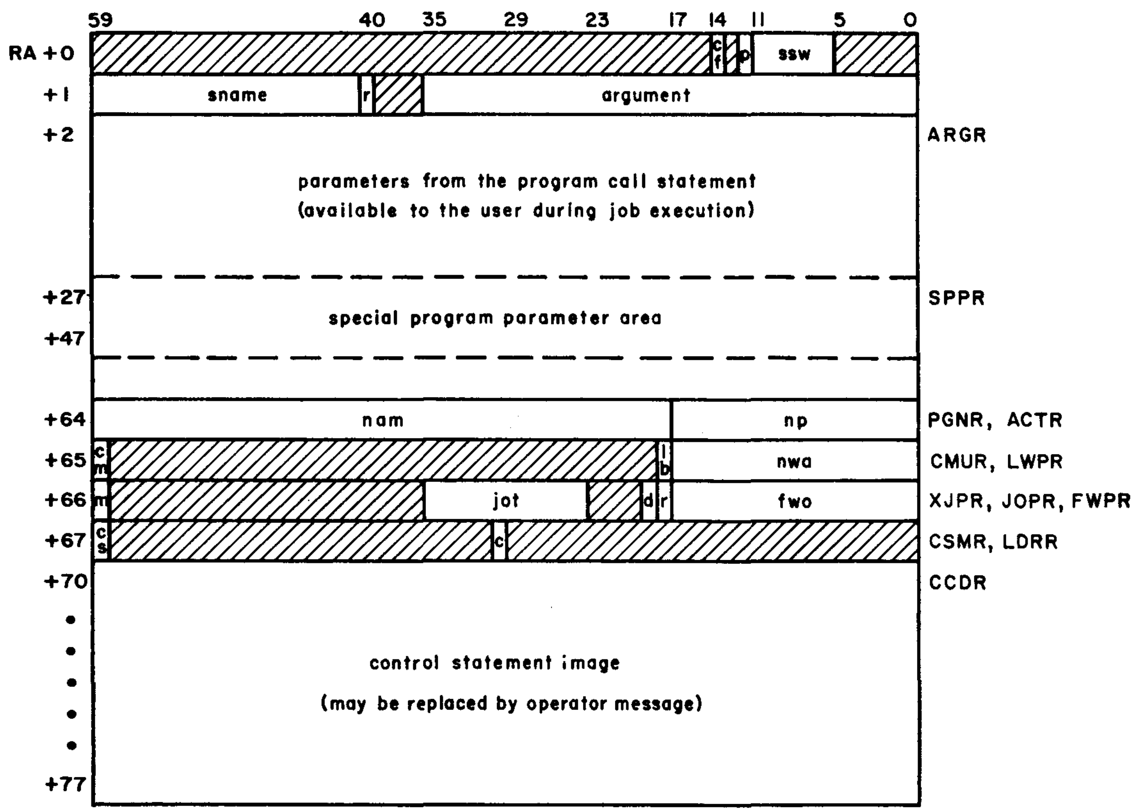



Figure 2-E-1. Job Communication Area

| Word | System Identifier | Bits | Field | Significance |
|---|---|---|---|---|
| RA+0 | | 59-15 | reserved | Reserved |
| | | 14 | cf | CFO bit |
| | | 13 | reserved | Reserved |
| | | 12 | p | Pause flag |
| | | 11-06 | ssw | Sense switches |
| | | 05-00 | reserved | Reserved |
| RA+1 | | 59-41 | sname | System request name (such as CIO) |
| | | 40 | r | Auto recall flag |
| | | 39-36 | unused | Reserved for future system use |
| | | 35-00 | arguments | Parameters passed to that portion of the system that processes the sname request |
| RA+2 through RA+$63_8$ | ARGR | 59-00 | params | Parameters from the program call statement; available to the user during execution |
| RA+$27_8$ through RA+$47_8$ | SPPR | 59-00 | params | Special program parameter area used by SSJ= entry point programs to store parameter blocks. Any job step can use this area for its own purpose, but if it is followed by an SSJ= program, the contents may be destroyed. |
| RA+$64_8$ | PGNR | 59-18 | nam | Name of program called by control statement |
| | ACTR | 17-00 | np | Number of parameters in control statement call |
| RA+$65_8$ | CMUR | 59 | cm | Set if the compare/move unit (CMU) is present |
| | LWPR | 58-19 | unused | Reserved for future system use |
| | | 18 | lb | Library flag:<br>0 Load from a file<br>1 Load from library |
| | | 17-00 | nwa | Address of next word available for loading |
| RA+$66_8$ | XJPR | 59 | m | Indicates if hardware feature CEJ/MEJ is available:<br>1 Available<br>0 Not available |
| | | 58-36 | unused | Reserved for future system use |
| | JOPR | 35-24 | jot | Job origin type |
| | | 23-20 | unused | Reserved for future system use |
| | | 19 | d | DIS flag |
| | | 18 | r | RSS flag |
| | FWPR | 17-00 | fwo | First word of object program |
| RA+$67_8$ | CSMR | 59 | cs | Set if system is running in 64 character set mode |
| | LDRR | 58-30 | unused | Reserved for future system use |
| | | 29 | c | Completion flag:<br>0 Load not completed<br>1 Load completed |
| | | 28-00 | unused | Reserved for future system use |
| RA+$70_8$ through RA+$77_8$ | CCDR | 59-00 | control statement image | Image of control statement currently being executed |

# EXCHANGE PACKAGE AREA

Figure 2-E-2 illustrates the exchange package area.

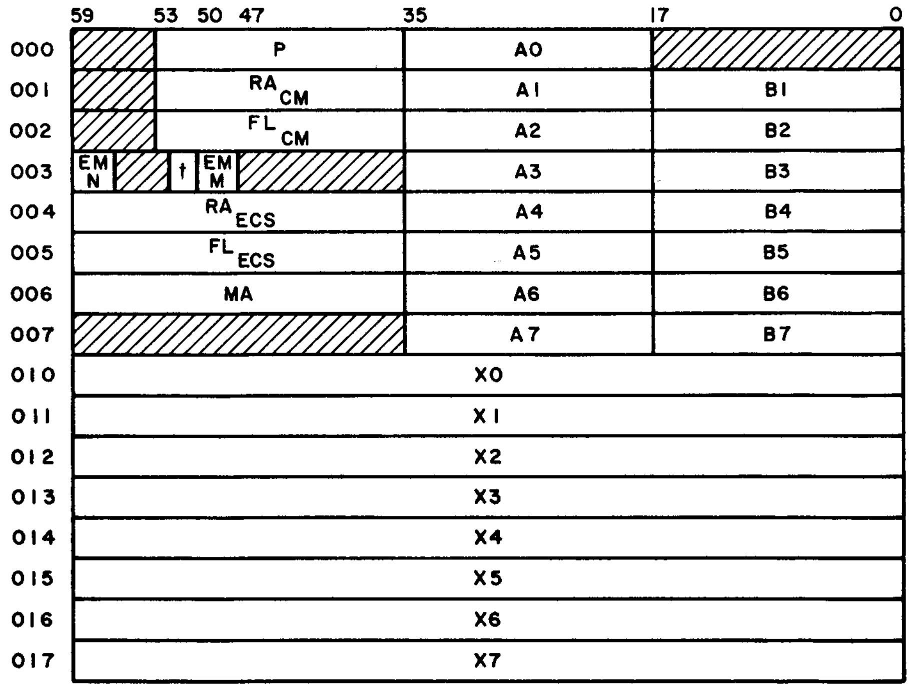



Figure 2-E-2. Exchange Package Area

| | |
|---|---|
| P | Program address |
| Ai | Address registers |
| RA(CM) | Reference address central memory |
| Bi | Increment registers |
| FL(CM) | Field length for central memory |
| EM-N | CPU hardware exit mode (CDC CYBER 170 series only): |

0 Disable hardware exit mode
1 ECS flag register operation parity error
2 CMC input error
3 1 or 2
4 CM data error
5 1 or 4
6 2 or 4
7 1 or 2 or 4

| | |
|---|---|
| EM-M | CPU program exit mode: |

0 Disable program exit mode
1 Address out of range
2 Operand out of range
3 1 or 2
4 Indefinite operand
5 1 or 4
6 2 or 4
7 1 or 2 or 4

† Bits 52 and 51, hardware error exit status bits on CDC CYBER 70 Model 74.

| | |
|---|---|
| RA(ECS) | Reference address ECS |
| FL(ECS) | Field length for ECS |
| MA | Monitor address |
| Xi | Operand registers |

## FILE TYPES AND ORIGIN CODES

The following file types and origin codes are used in many NOS system routines.

The queue file types are:

| Type | Value | Description |
|---|---|---|
| INFT | 0 | Input |
| ROFT | 1 | Rollout |
| PRFT | 2 | Print |
| PHFT | 3 | Punch |
| TEFT | 4 | Timed/event rollout |
| -- | 5 | Reserved |
| -- | 6 | Reserved |
| -- | 7 | Reserved |

Other file types include:

| Type | Value | Description |
|---|---|---|
| LIFT | 10 | Library |
| PTFT | 11 | Primary terminal |
| PMFT | 12 | Direct access permanent file |
| FAFT | 13 | Fast attach file |
| SYFT | 14 | System |
| LOFT | 15 | Local |
| -- | 16 | Reserved |

Following are the job origin codes.

| Type | Value | Description |
|---|---|---|
| SYOT | 0 | System |
| BCOT | 1 | Local batch |
| EIOT | 2 | Remote batch (Export/Import) |
| TXOT | 3 | Time-sharing |
| MTOT | 4 | Multiterminal |

# EQUIPMENT CODES

Equipment codes for device types supported by NOS are as follows.

| Code | Equipment |
|---|---|
| CP | 415 Card Punch |
| CR | 405 Card Reader |
| DE | Extended core storage |
| DI-n | 844-21 Disk Storage Subsystem (1 to 8 units) |
| DJ-n | 844-41/44 Disk Storage Subsystem (1 to 8 units) |
| DP | Distributive data path to ECS |
| DS | Display console |
| LP | 512 or 580 Line Printer |
| LQ | 512 Line Printer |
| LR | 580-12 Line Printer |
| LS | 580-16 Line Printer |
| LT | 580-20 Line Printer |
| MD-n | 841 Disk Drive (1 to 8 units) |
| MS | Mass storage device |
| MT | 7-track magnetic tape drive |
| NE | Null equipment |
| NP | 2550 Host Communications Processor |
| NT | 9-track magnetic tape drive |
| ST | 6671 or 2550-100 multiplexer |
| TT | 6676, 6671, or 2550-100 multiplexer |

# SPECIAL ENTRY POINTS F

The following entry points enable the ABS type system programs that contain them to perform special functions. These functions are independent of user control. They are described here only to provide background information to discussions of entry points elsewhere in the manual. COMPASS users should note that their programs may contain entry points of the same name as those described in this appendix, but that no special functions will be performed.

**NOTE**

Either the RFL= or the MFL= entry point is required in any program that uses special entry points.

ARG= System programs that do their own processing of control statement parameters contain an ARG= entry point. The parameters are not stored in the user's field length beginning at RA+2 (ARGR) as would normally occur, but rather are passed to RA+$70_8$ (CCDR). In addition to the parameters, the control statement image at RA+70 includes the statement name (such as PERMIT) and any prefix options ($ or /). This allows certain control statements to contain parameters that do not conform to the normal constraints of NOS control statement syntax.

DMP= Certain system programs called to perform special functions (for example, manage tapes or packs, checkpoint a job, etc.) contain a DMP= entry point. When such a program is loaded, the user's control point area, all or a portion of his central memory, and optionally his FNT/FST entries are dumped to a special rollout file. When the operation is completed, NOS transfers the contents of the rollout file back into central memory and resumes job processing.

MFL= System programs use the MFL= entry point to specify the minimum field length they require for execution. This entry point differs from RFL= in that RFL= always changes the running field length to the specified value. If the value of MFL= is greater than 200000, the program field length is set to the last job statement field length (byte 1, word FLCW of the control point area) or the value of MFL= (minus 200000), whichever is greater. If the value of MFL= is less than 200000, the program field length is set to the greater of the existing field length or the value of MFL=.

RFL= When a system program with an RFL= entry point is loaded, the running field length for the job is changed to the value of RFL=, rounded to the next highest multiple of $100_8$.

| | |
|---|---|
| SDM= | System programs that issue their own dayfile messages contain an SDM= entry point. This allows NOS to ensure that privileged information such as a user's password does not appear in the dayfile. |
| SSJ= | Programs that contain an SSJ= entry point are defined as special system jobs. A special system job is able to perform functions beyond the user's normal validation. It transfers the user's validation information from the control point area into a temporary storage area within its field length and then makes the necessary changes to the control point area. After the special job completes its operation, it restores the user's validation information to the control point area.<br><br>When a program that contains an SSJ= entry point is loaded, secure system memory (SSM) status is set (refer to Security Considerations, section 2). |
| SSM= | When a program that contains an SSM= entry point is loaded, secure system memory (SSM) status is set (refer to Security Considerations, section 2). SSM status cannot be cleared by any program containing an SSM= entry point. |
| VAL= | For jobs of any origin type except SYOT, if user validation is enabled (that is, a USER statement must be included), NOS allows only those system programs containing VAL= entry points to execute. The system uses this feature to ensure that the USER statement, and if required, CHARGE statement are properly processed. |

# BINARY FORMATS G

Appendix G describes the following binary formats.

| | |
|---|---|
| PP | CDC CYBER 170/70 or 6000 series PPU absolute |
| OPL | Modify old program library deck |
| OPLD | Program library directory |
| ULIB | User library group |
| TEXT | Unrecognized as binary |

For binary formats of loader tables, refer to the CDC CYBER Loader Reference Manual.

## PP – CDC CYBER 170/70 OR 6000 SERIES PPU ABSOLUTE

Binary output for a CDC CYBER 170/70 or 6000 series PPU program or overlay is a logical record that may contain the following.

Prefix table

CDC CYBER 170/70 or 6000 series PPU program control table

PPU text in five PPU words per 60-bit CPU word

The format of the control table is:

| 59 | 41 | 35 | 23 | 11 0 |
|---|---|---|---|---|
| name | 00 | fwa | 0000 | length |

| Bits | Field | Description |
|---|---|---|
| 59-42 | name | Program name, 1-to 3-display code characters, left-justified with zero fill. |
| 41-36 | none | Reserved for future system use. |
| 35-24 | fwa | Origin minus 5; address at which the header word is loaded. |
| 23-12 | none | Reserved for future system use. |
| 11-0 | length | Number of CPU words in program image (1/5 the number of PPU words). |

## OPL — MODIFY OLD PROGRAM LIBRARY DECK

A Modify old program library deck is a record on a Modify library file (figure 2-G-1) consisting of a prefix table, a modification table, and text. The prefix table contains the library creation date in word 2 and the latest modification date in word 3.

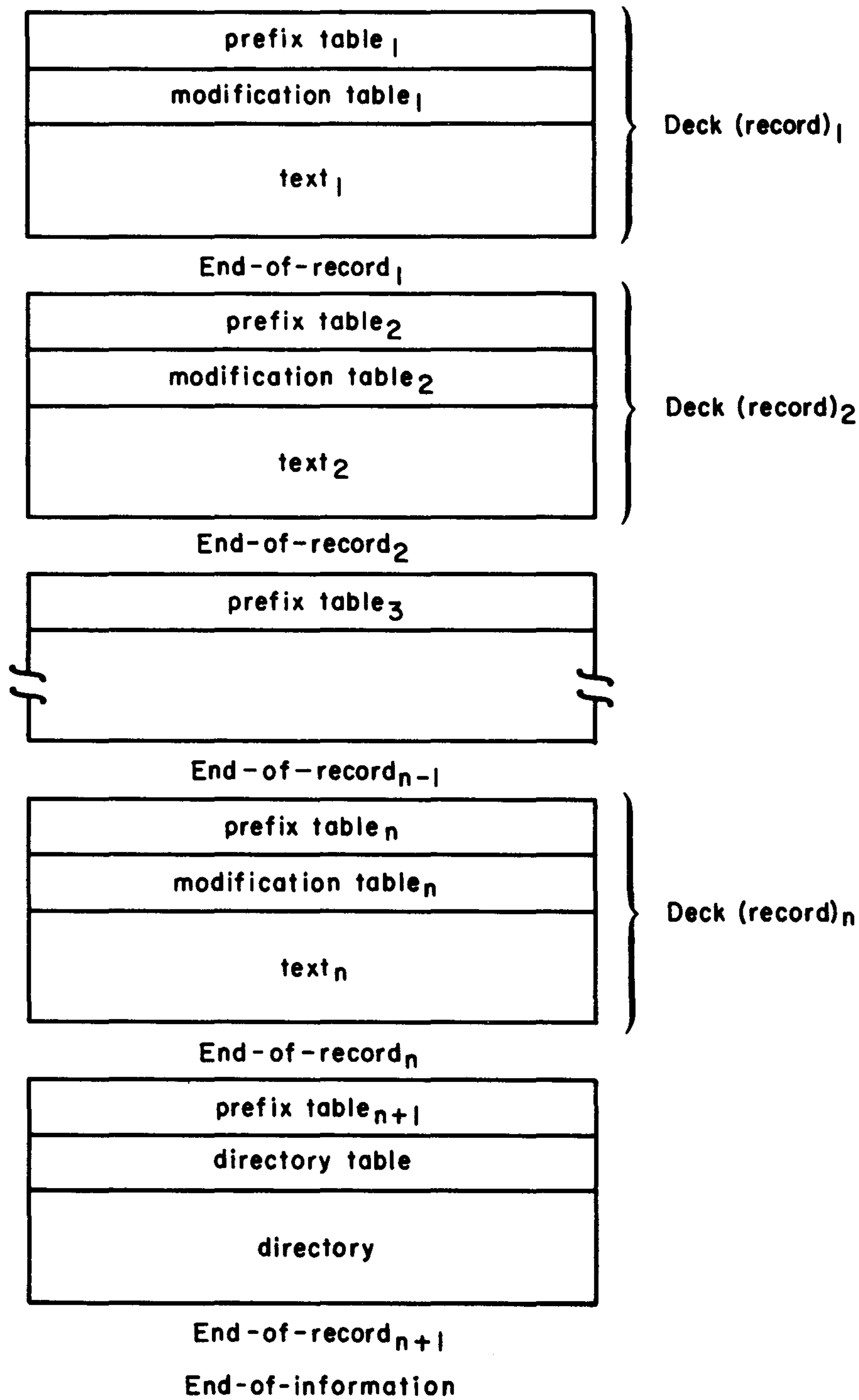



Figure 2-G-1. Modify Library File Format

## MODIFICATION TABLE FORMAT

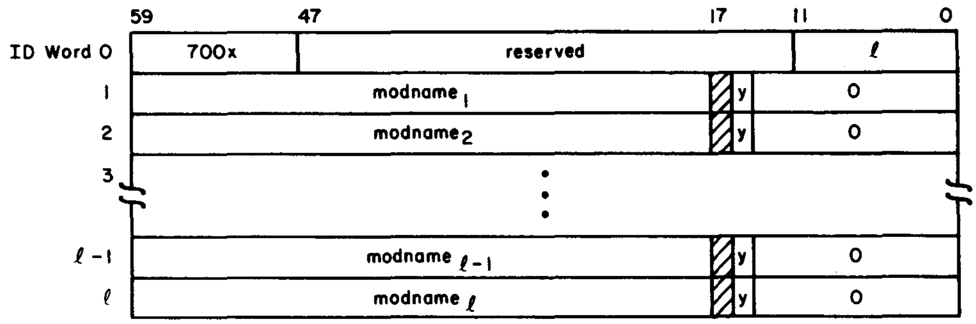



Figure 2-G-2. Modification Table Format

| Word | Bits | Field | Description |
|---|---|---|---|
| ID | 59-48 | 700X | Identifies Modify deck. Least significant digit indicates whether or not the deck is common as follows:<br>1 Deck is not common<br>2 Deck is common |
| | 47-12 | none | Reserved for future system use. |
| | 11-0 | $\ell$ | Number of modification names in table. |
| $word_i$ | 59-18 | $modname_i$ | 1-to 7-character modification set name. Each modification to a deck causes a new entry in this table. |
| | 16 | $y_i$ | YANK flag:<br>0 Modifier not yanked<br>1 Modifier yanked |

## TEXT FORMAT

Text is an indefinite number of words that contain a modification history and the compressed image of each line in the deck. Text for each line is in the following format.

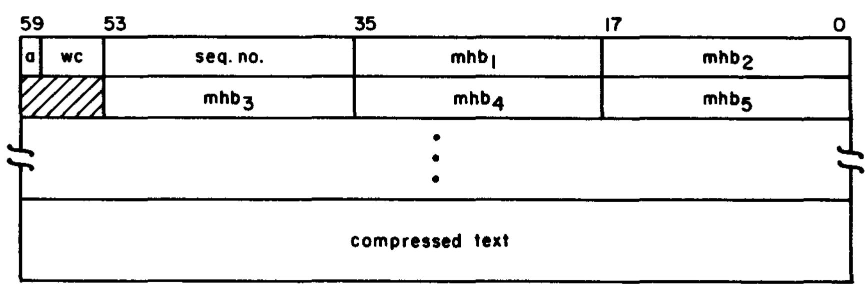
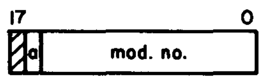



Figure 2-G-3. Modify Text Format

| Bits | Field | Description |
|---|---|---|
| 59 | a | Activity bit:<br>0 Line is inactive<br>1 Line is active |
| 58-54 | wc | Number of words of compressed text. |

| Bits | Field | Description |
|---|---|---|
| 53-36 | seq.no. | Sequence number of card (octal) according to the position in the deck or modification set. |
| 35-18 and subsequent 18-bit bytes | $mhb_i$ | Modification history byte. Modify creates a byte for each modification set that changes the status of the card. Modification history bytes continue to a zero byte. Since this zero byte could be the first byte of a word and the compressed card image begins a new word, the modification history portion of the text could terminate with a zero word. |

The format of $mhb_i$ is:




| | |
|---|---|
| a | Activate bit:<br>0 Modification set deactivated the card<br>1 Modification set activated the card |
| mod.no. | Index to the entry in the modification table that contains the name of the modification set that changed the card status. |

Following the modification history bytes is the compressed image of the card in display code. A single space is represented by $55_8$; it is not compressed. Two or more embedded spaces are replaced in the image as follows:

3 spaces replaced by 0002

4 spaces replaced by 0003

. . .

. . .

. . .

64 spaces replaced by $0077_8$

65 spaces replaced by $007755_8$

66 spaces replaced by $00770001_8$

67 spaces replaced by $00770002_8$, etc.

Trailing spaces are not considered as embedded and are not included in the card image. A 12-bit zero byte marks the end of the card.

**NOTE**

Existing 63 character set program libraries may have two spaces represented as a 0001 byte.

## OPLD — PROGRAM LIBRARY DIRECTORY

The library file directory contains an identification table followed by a table containing a two-word entry for each deck in the library. Directory entries are in the same sequence as the decks on the library.

| | 59 | 47 | 29 | 17 0 |
|---|---|---|---|---|
| ID Word 0 | 7000 | 0 | | $\ell$ |
| 1 | $\text{deckname}_1$ | | | $\text{type}_1$ |
| 2 | $p_1$ | | $\text{random address}_1$ | |
| 3 | $\text{deckname}_2$ | | | $\text{type}_2$ |
| 4 | $p_2$ | | $\text{random address}_2$ | |
| | ⋮ | | | |
| $\ell$-1 | $\text{deckname}_{\ell/2}$ | | | $\text{type}_{\ell/2}$ |
| $\ell$ | $p_{\ell/2}$ | | $\text{random address}_{\ell/2}$ | |

Figure 2-G-4. Library File Directory Table

| Word | Bits | Field | Description |
|---|---|---|---|
| ID | 59-48 | Table type | Identifies the table as a program library directory.. |
| | 17-0 | $\ell$ | Directory length excluding ID word. |
| 1,3, ..., | 59-18 | $\text{deckname}_i$ | Name of the program library deck; 1 to 7 characters, left justified. If $\text{type}_i$ equals $16_8$, this field contains the capsule name. |
| $\ell$-1 | 17-0 | $\text{type}_i$ | Type of record (octal).<br>0 Unrecognizable as binary program (TEXT)<br>1 CDC CYBER 170/6000 peripheral processing unit program (PPU)<br>3 Relocatable central processor program (REL)<br>4 Central processor overlay (OVL)<br>5 User library program (ULIB)<br>6 Old program library deck (OPL)<br>7 Old program library common deck (OPLC)<br>10 Old program library directory (OPLD)<br>11 Multiple entry point overlay (ABS)<br>16 Fast dynamic load capsule (CAP) |
| 2,4, ...,$\ell$ | 59-30 | $p_i$ | Zero for all record types except CAP ($16_8$). If $\text{type}_i$ equals $16_8$ this field is defined as follows:<br>**Bits** / **Description**<br>59-48 Ordinal to group name in ULIB record (refer to ULIB record)<br>47-30 Length of capsule |
| | 29-0 | random $\text{address}_i$ | Address of the deck relative to the beginning of the file |

## ULIB — USER LIBRARY GROUP

A user library group (figure 2-G-7) consists of all records beginning with a ULIB-type record through the next OPLD-type record.

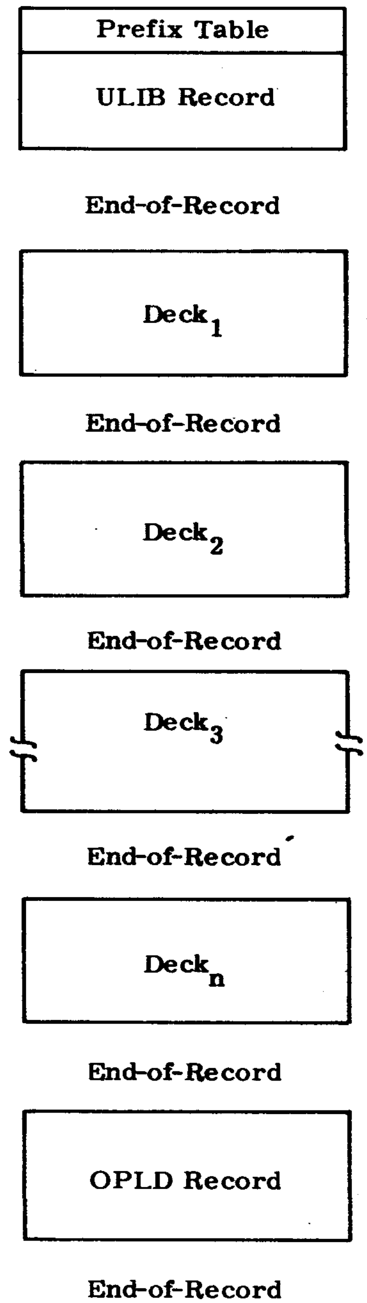
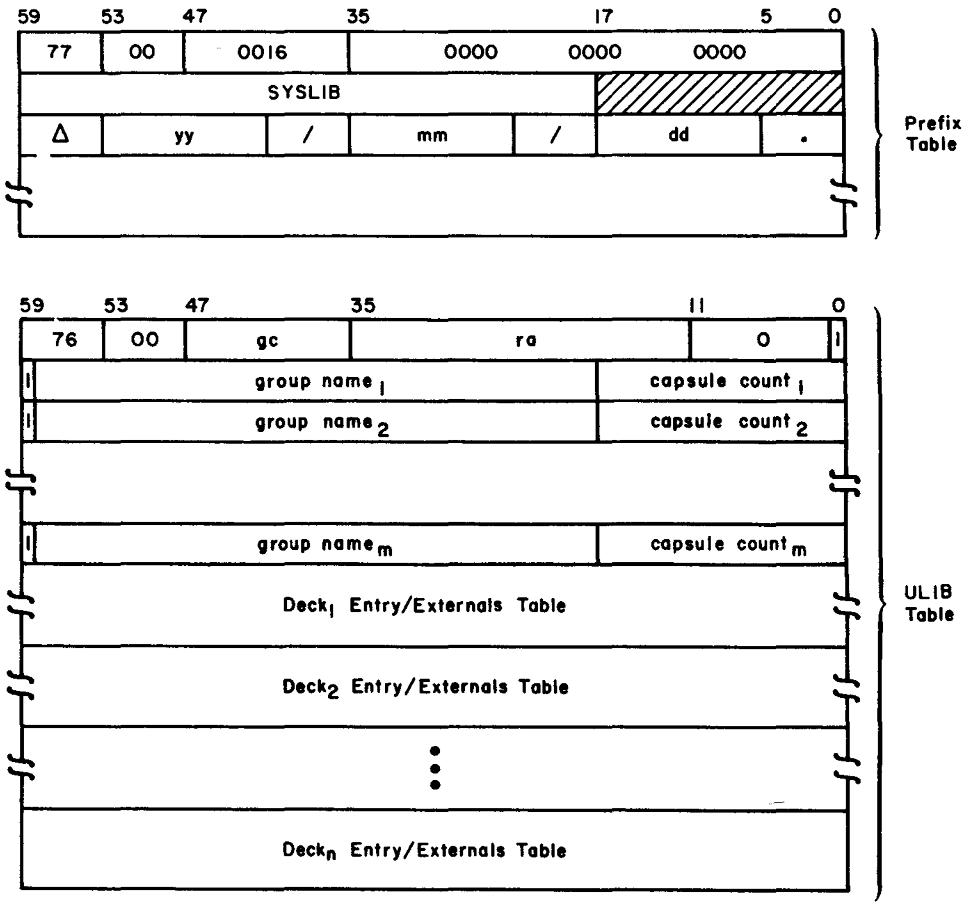



Figure 2-G-5. User Library (ULIB) Format




| | |
|---|---|
| gc | Group count (count of different group names in ULIB) equal to m |
| ra | Random address of OPLD |
| i | Cross reference flag<br>0 ULIB table contains cross references.<br>1 ULIB table does not contain cross references. |
| group $name_j$ | Name of group |
| capsule $count_j$ | Number of capsules with group $name_j$ |

Figure 2-G-6. ULIB Record Format

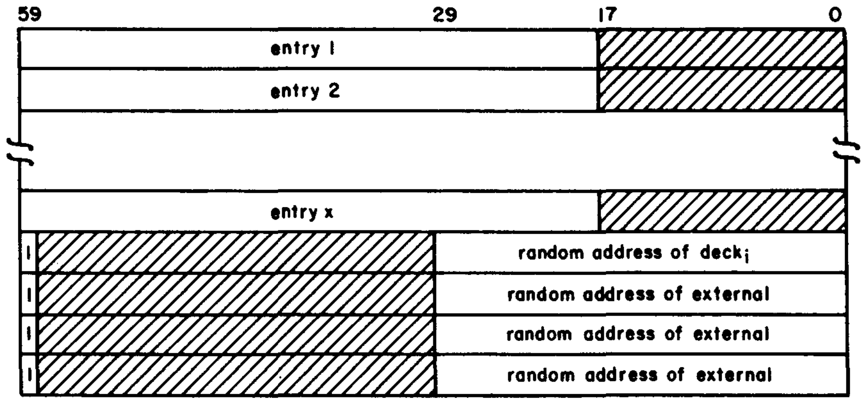



Figure 2-G-7. ULIB Deck Entry/Externals Format

## TEXT – UNRECOGNIZED AS BINARY

A text record consists of a one-word header containing the record name in display code, left-justified with zero fill. Text immediately follows. The record is terminated by an end-of-record mark.

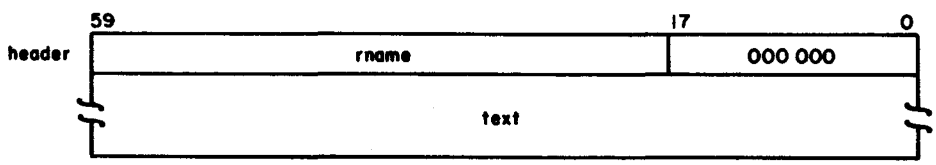



Figure 2-G-8. Text Record Format

rname  1 to 7 alphanumeric characters, left-justified with zero fill.

## COMPRESSED COMPILE FILE

A compressed compile file has a one-word header.

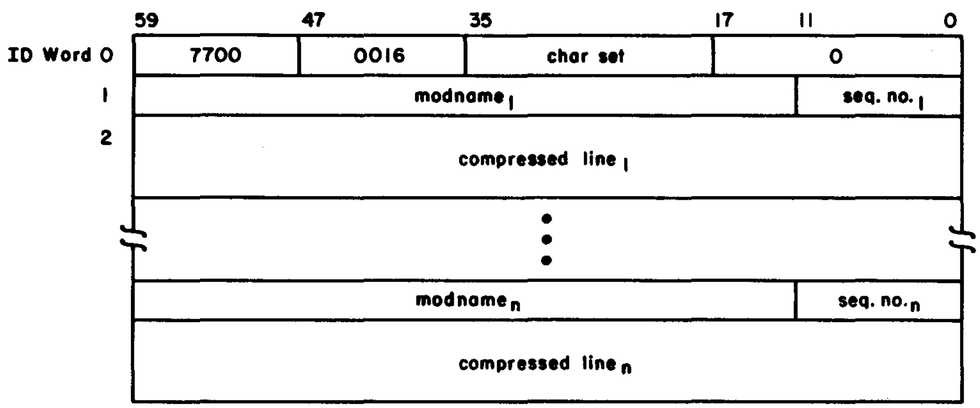



Figure 2-G-9. Compressed Compile File Format

| | |
|---|---|
| CS = 0 | 63 character set compressed compile file |
| CS = 64 | 64 character set compressed compile file |

# COMPASS CONTROL STATEMENT H

The COMPASS control statement calls the COMPASS assembler to the control point. The minimum memory requirement for a COMPASS assembly is 42,000 octal locations.

The control statement format is:

COMPASS($p_1, p_2, \ldots, p_n$)

$p_i$ Parameters are order-independent and can be in the following format.

a

a = fname

a = 0

a = chars

The following parameters can be supplied.

| Option | Meaning |
|---|---|
| A | Abort to EXIT control statement if assembly errors are detected. If no A parameter is supplied and assembly errors are detected, job processing continues. |
| B | Object code is written on file LGO. If no B parameter is supplied, this option is assumed. |
| B=fname | Object code is written on file fname. |
| B=0 | No object code is generated. |
| D | Object code is generated and written on the file specified by the B parameter even if assembly errors occur. D is ignored if B=0. If the D parameter is omitted, assembly errors inhibit object code generation. |
| F | Specifies that the COMPASS assembler was called by a COMPASS control statement. If this parameter is omitted, COMPASS is assumed. |
| F=n | n = 0 COMPASS was called by the COMPASS control statement.<br>n = 2 COMPASS was called by the FORTRAN Extended (FTN) compiler. |
| F=name | The above options can be specified by supplying the names COMPASS or FTN. |
| G | Use system text from local file SYSTEXT. |
| G=fname† | System text is first overlay on local file fname. If G is omitted or G=0, no system text is loaded. |

†A maximum of seven G parameters is allowed.

| Option | Meaning |
|---|---|
| G=fname/ovl | System text is the named overlay, ovl, on local file fname. |
| I | Source deck is to be read from file COMPILE. If this parameter is omitted, the source deck is read from file INPUT. |
| I=fname | Source deck is to be loaded from file fname. |
| L | Writes full assembly listing on file OUTPUT. When the L parameter is omitted, this option is assumed. |
| L=fname | Lists output on the specified file. When the full list is on a different file from that specified for the short list, the listing for each subprogram is written as a single logical record preceded by a one-word header consisting of an asterisk and the first six characters of the subprogram name. This header identifies the subprogram as a convenience for sorting and cataloging. Also, refer to the O option. |
| L=0 | No full list is generated. |
| LO=0 | Selects list options B, L, N, and R only. If the LO parameter is omitted, these options are assumed. |
| LO | Selects list options C, F, G, and X and deselects R. |
| LO=$$$$ | Selects all options. |
| LO=chars | Selects or deselects a maximum of nine of the following list options. Inclusion of B, L, N, and R deselects the corresponding option; otherwise, inclusion of the character selects the specified option. |

A — Lists statements actually assembled. When A is not selected, a line containing concatenation and micro substitution marks is listed containing the marks exactly as presented to the assembler. When the A option is selected, however, the assembler lists the line before and after the editing takes place. Selecting A also causes the listing of lines of code resulting from the R= pseudo-instruction.

B — Lists binary control statements. When B is selected, the listing includes SEG, SEGMENT, IDENT, and END pseudo-instructions.

C — Lists control statements. When C is selected, the listing includes EJECT, SPACE, TTL, and TITLE pseudo-instructions. A listing instruction that causes an EJECT is listed as the first line of the new page after the EJECT takes place.

D — Includes details. Selection of the D option causes listing of the following items not normally listed.

- Second and subsequent lines of DATA and DIS
- Code assembled remotely when HERE or END causes its assembly
- Literals block
- Default symbols

| Option | Meaning |
|---|---|
| E | Includes echoed lines. Selection of E causes a listing of all iterations of code duplicated as a result of DUP and ECHO. |
| F | Lists IF-skipped lines. When F is selected, the listing includes all lines skipped by IF, IFop, IFC, IFPP, IFCP, SKIP, and ELSE. In addition, the symbolic reference table contains references to symbols in IF statements. |
| G | Lists generated code. Selection of this option causes a listing of all code generating lines regardless of list controls other than L. Instructions listed include symbolic machine instructions and BSS, BSSZ, CON, DATA, DIS, R=, and VFD. |
| L | Master list control. This option is normally selected. When L is canceled, the long list contains error-flagged lines, an error directory, and LIST and END pseudo-instructions only, regardless of selection of any other options. |
| M | Lists macros and opdefs. Selection of M causes all lines generated by calls to macros and opdefs other than those defined by the system to be listed. |
| N | Lists nonreferenced symbols. This option is normally selected. Cancellation of this option causes any nonsystem symbol for which no reference has been accumulated (for example, all occurrences are in IF statements with the F option deselected or are between CTEXT or ENDX with the X option deselected) to be omitted from the symbolic reference table. |
| R | Accumulates and lists references. This option is normally selected. When R is canceled, COMPASS does not accumulate references. R should not be canceled if a complete symbolic reference table is desired. If R is canceled at assembly end, no symbolic reference table is produced. |
| S | List system macros and opdefs. Selection of S causes all lines generated by calls to system-defined macros and opdefs to be listed. |
| T | List nonreferenced system symbols. Selection of this option causes a symbol defined through SST to be included in the symbolic reference table even if there are no accumulated references. |
| X | List XTEXT lines. When X is selected, all statements assembled as a result of an XTEXT pseudo instruction are listed. CTEXT and ENDX provide means of alternately turning this external designator off and on. |

| Option | Meaning |
|---|---|
| ML | MODLEVEL is equal to JDATE. When ML is omitted, this option is assumed. |
| ML=chars | MODLEVEL is equal to the 9-character string, chars. |
| N | No eject; suppresses ejects caused by normal listing control. If this option is omitted, normal page ejects occur. |
| O | Short list; suppressed if full list is directed to the same file or if no assembly errors occur. However, if the full list and short list are on different files (for example, the full list is written on OUTPUT and the short list is written on the named file), the short list contains all statements having error flags. If an error line is generated by a macro call, the macro call may also be in the short list. |
| O=fname | Short list is written on file fname. |
| O=0 | No short list is provided. |
| PC | PCOMMENT micro is 30 blanks. When PC is omitted, this option is assumed. |
| PC=chars | PCOMMENT is the 30-character string, chars. |
| P | Selects consecutive page numbering. When P is omitted, page numbering begins with 1 at the start of each subprogram. |
| S | Text is SYSTEXT. |
| S=ovl† | Text is named overlay, ovl. If S is omitted and G is not specified, text is on SYSTEXT overlay. S=0 indicates no system text. |
| S=lib/ovl | Text is named overlay, ovl, on the specified library, lib. |
| X | External text is on file OPL. When X is omitted, XTEXT is satisfied from file OLDPL. |
| X=fname | External text is to be obtained from file fname. |

† A maximum of seven S parameters is allowed.

# INDEX

# COMMENT SHEET

MANUAL TITLE CDC NOS Version 1

Reference Manual, Volume 2

PUBLICATION NO. 60445300 REVISION E

**FROM:** NAME: __________

BUSINESS
ADDRESS: __________

CUT ALONG LINE

PRINTED IN USA

AA3419 REV. 7/75

NO POSTAGE STAMP NECESSARY IF MAILED IN U. S. A.

FOLD ON DOTTED LINES AND STAPLE

STAPLE

FOLD

FOLD

FIRST CLASS
PERMIT NO. 8241

MINNEAPOLIS, MINN.

**BUSINESS REPLY MAIL**
NO POSTAGE STAMP NECESSARY IF MAILED IN U.S.A.

POSTAGE WILL BE PAID BY

CONTROL DATA CORPORATION
Publications and Graphics Division
ARH219
4201 North Lexington Avenue
Saint Paul, Minnesota 55112

CUT ALONG LINE

FOLD

FOLD

CORPORATE HEADQUARTERS, P.O. BOX 0, MINNEAPOLIS, MINNESOTA 55440
SALES OFFICES AND SERVICE CENTERS IN MAJOR CITIES THROUGHOUT THE WORLD

LITHO IN U.S.A.

CONTROL DATA CORPORATION