60454300

CONTROL DATA CORPORATION

NOS VERSION 1
INTERNAL
MAINTENANCE
SPECIFICATION

VOLUME 2 OF 3

CDC® COMPUTER SYSTEMS:
CYBER 170 SERIES
CYBER 70
MODELS 71, 72, 73, 74
6000 SERIES

REVISION RECORD	
REVISION	DESCRIPTION
A	Manual released. Manual reflects NOS 1.3.
(06/26/78)	
B	Revised to update manual to NOS 1.4 and to make
(08/03/79)	typographical and technical corrections. New
	features documented in this manual include: extended
	character set/print train support; expanded ECS
	status; on-line ECS diagnostic support; retry on
	time/SRU limit: IAF enhancements; deadstart from mass
	storage; CYBER 170 Model 176 support; extended TIM
	function; 885 Disk Storage Subsystem support; task
	initiated K.DUMP; TAF internal XJP trace; LIBTASK
	enhancements; TAF CYBER Record Manager support; and
	TAF/COBOL interface enhancements. This revision
	obsoletes all previous editions.

Publication No.
60454300

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

Printed in the United States of America

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

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 Computer Systems; CDC CYBER 70 Series, Models 71, 72, 73, and 74 Computer Systems; and CDC 6000 Series Computer Systems.

AUDIENCE

This internal maintenance specification (IMS) provides the systems analyst with detailed internal documentation of NOS. Included are detailed descriptions of system routines and the system interfaces, tables, and flowcharts of these routines. Some user interfaces are mentioned, but these are fully described in other NOS manuals.

CONVENTIONS

Extended memory for the CYBER 170 Models 171, 172, 173, 174, 175, 720, 730, 750, and 760 is extended core storage (ECS). Extended memory for CYBER 170 Model 176 is large central memory (LCM) or large central memory extended (LCME). ECS and LCM/LCME are functionally equivalent, except as follows:

- LCM/LCME cannot link mainframes and does not have a distributive data path (DDP)capability.
- LCM/LCME transfer errors initiate an error exit, not a half exit. Refer to the COMPASS Reference Manual for complete information.

The Model 176 supports direct LCM/LCME transfer COMPASS instructions (octal codes 014 and 015). Refer to the COMPASS Reference Manual for complete information.

In this manual the acronym ECS refers to all forms of extended memory on the CYBER 170 Series. However, in the context of a multimainframe environment or DDP access, the Model 176 is excluded.

In this manual, the order of importance of headings is denoted as follows.

LEVEL 1 HEADINGS ARE FULL CAPS AND UNDERLINED

LEVEL 2 HEADINGS ARE FULL CAPS

Level 3 Headings are First-Capped and Underlined

Level 4 Headings are First-Capped

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.

RELATED PUBLICATIONS

For further information concerning CYBER 170, CYBER 70, and 6000 Series Computer Systems, the NOS time-sharing systems, and the user interface for NOS, consult the following manuals.

Control Data Publication	Publication No.
CYBER 170 Computer Systems Reference Manual	60420000
CYBER 170 Computer Systems Models 720, 730, 750, and 760 Model 176 (Level B)	60456100
CYBER 70/Model 71 Computer System Reference Manual	60453300
CYBER 70/Model 72 Computer System Reference Manual	60347000
CYBER 70/Model 73 Computer System Reference Manual	60347200
CYBER 70/Model 74 Computer System Reference Manual	60347400
Modify Reference Manual	60450100
Network Products Interactive Facility Version 1 Reference Manual	60455250
Network Products Transaction Facility Version 1 Reference Manual	60455340
Network Products Transaction Facility Version 1 User's Guide	60455360
Network Products Transaction Facility Version 1 Data Manager Reference Manual	60455350

Control Data Publication	Publication No.
Network Products Transaction Facility Version 1 CYBER Record Manager Data Manager Reference Manual	60456710
Network Products Network Access Method Version 1 Reference Manual	60499500
Network Products Network Access Method Version 1 Internal Maintenance Specification	60490110
Network Products Remote Batch Facility Version 1 Reference Manual	60499600
NOS Version 1 Installation Handbook	60435700
NOS Version 1 Operator's Guide	60435600
NOS Version 1 Reference Manual Volume 1	60435400
NOS Version 1 Reference Manual Volume 2	60445300
NOS Version 1 System Maintenance Reference Manual	60455380
NOS Version 1 System Programmer's Instant	60449200
NOS Version 1 Time-Sharing User's Reference Manual	60435500
NOS Version 1 Export/Import Reference Manual	60436200
TAF/TS Version 1 Reference Manual	60453000
TAF/TS Version 1 User's Guide	60436500
TAF/TS Version 1 Data Manager Reference Manual	60453100
TAF/TS Version 1 CYBER Record Manager Data Manager Reference Manual	60456700
6400/6500/6600 Computer System Reference Manual	60100000

DISCLAIMER

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

CONTENTS

FIGURES

FIGURES (Continued)

FIGURES (Continued)

FIGURES (Continued)

TABLES

TABLES (Continued)

Control point management enables the user to alter or interrogate parameters in the job's control point area. Control point management consists of a set of control statements, macros, and the PP routine control point manager (CPM).

CPM is called as follows:

- A user can issue an RA+1 call to CPM or the appropriate macro which generates an RA+1 call.
- A user can use the appropriate control statement, which causes CPU program CONTROL to be loaded in the user's field length. CONTROL then uses the appropriate macro to generate the RA+1 call to CPM.

The macro definitions for CPM functions are found in SYSTEXT (and NOSTEXT), PSSTEXT, or in common decks COMCCMD and COMCMAC. Those defined in SYSTEXT are general use macros, such as EREXIT and USERNUM, while those macros defined in PSSTEXT and COMCCMD and COMCMAC have special system usage, such as MODE, GETJA, and SETUI. All CPM macros require common decks COMCCPM and COMCSYS. A detailed description of each user CPM function and its macro call is available in the NOS Reference Manual, volume 2.

The PP routine CPM performs the requested CPM function except for those CPM functions that are performed by CPUMTR because of their high usage. The functions are outlined in table 10-1.

TABLE 10-1. CPM FUNCTIONS

Code	Description	CPM Overlay	CPM Entry Point
0	Set queue priority	CPM	SQP
1	Set CPU priority	CPM	SPR
2	Set exit mode	CPM	SEM
3	Set time and SRU limit	3CB	SLL
4	Set error exit return address	3CC	SEE
5	Set K-display register	CPM	SDA
6	Rollout job	CPM	ROC
7	Set no exit/on exit bit	CPM	NEX
10	Set secure system memory	CPM	SSM
11	Turn on sense switches	CPM	ONS
12	Turn off sense switches	CPM	OFS
13	Read job name	CPM	RJN
14	Read queue priority	CPM	RQP
15	Read CPU priority	CPM	RPR
16*	Read exit mode	CPM	PRS1
17	Retrieve limit	3CB	RLM
20	Enter demand index	CPM	EDI
21	Set user index	3CA	SUI
22	Set loader control word	CPM	SLC
23	Set last RFL (CM/ECS)	CPM	RFL
24*	Read job control word	CPM	PRS1

* Processed by CPUMTR

TABLE 10-1. CPM FUNCTIONS (CONTINUED)

Code	Description	CPM Overlay	CPM Entry Point
25*	Set job control word	CPM	PRS1
26	Set subsystem flag	CPM	SSF
27	Read job origin code	CPM	ROT
30	Read accounting information	3CB	RAI
31	Select CPU to execute in	CPM	SCP
32*	Return user number	CPM	PRS1
33*	Read FL control word	CPM	PRS1
34	Enter event into system event table	CPM	EET
35	Set pack name	CPM	SPN
36	Return pack name	CPM	RPN
37*	Set subsystem flag	CPM	PRS1
40	Validate user number	3CA	VAN
41	Enter family name	3CA	FAM
42	Special charge functions	3CB	SCF
43*	Disable SSJ= control	CPM	PRS1
44	Return version name	CPM	RVN
45*	Get loader control word	CPM	PRS1
46	Get global library set	3CC	GLS
47	Set global library set	3CC	SLS
50*	Return machine ID	CPM	PRS1

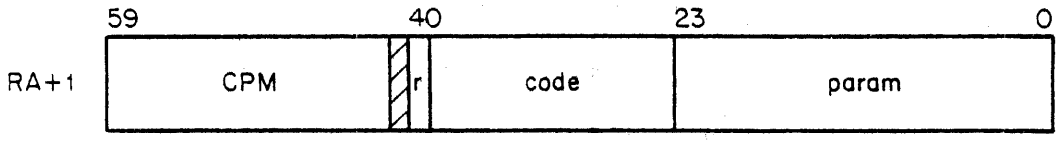
* Processed by CPUMTR

TABLE 10-1. CPM FUNCTIONS (CONTINUED)

Code	Description	CPM Overlay	CPM Entry Point
51	Return job activity information	CPM	RAC
52	Set maximum field length (CM/ECS)	CPM	MFL
53	Toggle SRU calculation	3CB	CSC
54	Set job class	CPM	SJC
55*	Read ECS Control word	CPM	PRS1
56	Validate user	3CA	VAL
57	Get permanent file parameters from control point area	CPM	GPF
60	Set permanent file parameters in control point area	3CA	SPF
61*	Get list of files address	CPM	CKA1
62*	Set list of files address	CPM	CKA1
63-72	Reserved for CPUMTR	CPM	PRS1
73	Decrement family user count	3CA	DFC
74	Job control information	CPM	JCI
75	Set/clear job control flags	CPM	PRO
76	Set/clear override required to drop job flag	CPM	SOV

* Processed by CPUMTR

The format of the RA+1 call to CPM is as follows:



r	Autorecall bit
code	CPM function code
parameter	Parameter for the function

FUNCTION PROCESSING

CPUMTR checks in the table of CPM function codes for those function codes that are processed in CPUMTR. The remainder of the CPM functions are processed in PP routine CPM. CPM consists of a preset routine (PRS) and processors for the individual functions. Preset performs function code and special parameters verification and calls the appropriate function processor. Although the CPM functions processed by CPUMTR are defined in CPM's function table, they are all processed as errors if those function codes were recognized by CPM.

CPM ORGANIZATION

CPM consists of the main program (CPM) and the processors and subroutines listed in table 10-1.

The following subroutines are used by CPM.

Subroutine and Overlay	Description
CFN/CPM	Compare family names
DSC/3CA	Decrement security count
CKA/CPM	Check address
CUF/3CB	Check for profile file update failure
LBD/3CC	Search library directory for special entry
PMP/CPM	Process memory parameters
RLN/3CC	Return library name to user program
RLW/3CC	Read library name from user program
RUI/3CA	Read user identification word
SPP/3CB	Set profile parameters

Subroutine and Overlay	Description
UFC/CPM	Update family activity counts
UPF/3CB	Update project profile file using overlay OAU
ERR/CPM	Process error

The following common decks are used by CPM.

Common Deck/Overlay	Description
COMPECX/3CA	Compute ECS maximum field length
COMPCMX/3CA	Compute maximum field length
COMPSFE/3CA	Set family equipment
COMPCVI/CPM	Convert validation indexes
COMPCRA/3CB	Convert random address
COMPGTN/3CB	Generate terminal number
COMPFAT/3CB	Search for fast-attach file
COMPRJC/CPM	Read job control word
COMPSTA/3CB	Set terminal table address
COMPVFN/CPM	Verify file name

CPM organization is completed by the preset routine (PRS), the function table (TFCN), and common deck COMPCRS (check recall status).

A file is a collection of data saved on a storage medium. It can be tape or mass storage. The data is written in groups of blocks or sectors. The system controls and designates a file by its FNT and keeps its position by the FST.

There are basically two kinds of files.

- Explicit files are defined by an FNT/FST.
- Implicit files are known only by a track reservation in the TRT. They are actually track chains and, as such, are managed by the owner. They are, more specifically, files unknown to the system. The best example of this type of file is the indirect permanent file track chain.

This section describes those files known to the system explicitly.

These files all have an FNT/FST entry. The FST is used basically for file positioning information, with exceptions for queue type files. Refer to Section 2 for formats of FNT/FST entries.

The FNT/FST is created for several reasons. With the exception of 1TA for TELEX rollout files, all files are created (that is, an FNT/FST entry is built) by the PP routine OBF (begin file). With no exceptions, FNT/FST entries are cleared (that is, the files are dropped) by the PP routine ODF (drop files).

An FNT entry is considered empty if the lfn=0. When a new file is created, an empty entry is found and used for this new file. Routine OBF creates a file with any local file name, even those consisting of special characters. Only a PP routine can use OBF, so a PP routine can create a file with any 1- to 7-character name. CPU programs, however, must request CIO to create a file entry for them. CIO requires that a name be composed of 1 to 7 alphanumeric characters. If CIO finds a special character in a file name which is to be created, it aborts the program. However, CIO accepts files that were previously created with an illegal name for reading, writing, or positioning. This allows a DMP= special entry point routine to use the file DM*, which is created by 1RO. Once CIO determines that the lfn is legal, it calls OBF to create the FNT/FST entry.

The job origin field always contains the origin code of the creator of the file (SYOT=0, BCOT=1, for example).

Bit 5 of the FNT is set for mass storage files whose system sector contains information for later processing (for example, input and output queue files).

The control point number field of the FNT contains the control point number of the current user of the file.

The file type field defines what type of file it is. If the control point number is zero, then the file is not assigned to a control point; this is usually the case for files in a queue. This field is set up by OBF in the following manner. A table of file names is kept in OBF's FL. The file type is set to the corresponding file name. If the caller of OBF desires that the file have a file type different than OBF generates, the caller must change it himself. PFM changes the type to LOFT for the GET command, even if the name was one of those in the OBF table, and changes type to PMFT for ATTACH commands.

The table is at location TSFN in OBF and contains the following.

File Name	File Type
INPUT	LOFT
OUTPUT	PRFT
PUNCH	PHFT
PUNCHB	PHFT
P8	PHFT
LGO	LOFT
Any other name	LOFT

FILE TYPES

The following file types are defined by NOS.

The queue file types are as follows:

Type	Value	Description
INFT	0	Input
ROFT	1	Rollout
PRFT	2	Print
PHFT	3	Punch
TEFT	4	Timed/event rollout

Special queue file types are as follows:

Type	Value	Description
S1FT	5	Special file type 1
S2FT	6	Special file type 2
S3FT	7	Special file type 3

Other file types include the following.

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

File types INFT, ROFT, PRFT, PHFT, and TEFT are described in Sections 5 and 6.

Types S1FT, S2FT, and S3FT are special queue files that are reserved for use by NOS.

Library files (LIFT) are locked common files that are currently assigned to a control point. An FNT/FST entry for a file of type LIFT always has a control point assignment. The library file type is also used by system utilities while dumping and loading queues, dayfiles, and permanent files.

Primary files (PTFT) are created with the OLD or PRIMARY control statements. When the user issues the OLD command, a copy of the indirect access permanent file is created with PTFT type in unlocked mode. The NEW command creates a scratch file with type PTFT in unlocked mode. The PRIMARY command can be used to change a file of type LOFT to that of PTFT.

The time-sharing user can additionally create a primary file with the LIBRARY command. The LIBRARY command requests a copy of an indirect access file from the user number LIBRARY (user index 377776B). It is equivalent to the following command.

```
OLD,lfn=pfn/UN=LIBRARY.
```

When the user issues an ATTACH command, he gets the file pointed to by an FNT of type PMFT. If the file is attached in read mode, then many users can each get an FNT of type PMFT pointing to the same file. If a user attaches the file in write mode, it will be the only FNT pointing to the file. The DEFINE command is used to create a PMFT type file.

Fast attach files (FAFT) are files which have an FNT always in the FNT table. PFM searches the FNTs first on an ATTACH command, and if it finds it there, PFM gives the user his own FNT/FST entry for the file and this entry has file type PMFT.

System type files are files which are used by the system for special functions. Examples of SYFT files are VALIDUs, RSXDid, and RSXVid, which are created by the deadstart procedures and permanently remain at FNT ordinals 1, 3, and 4, respectively. These file types are changed to FAFT whenever ISF is run. If the ISF(R=lfn) is used, then the lfn specified, if of type FAFT, is changed to type SYFT.

Local type files (LOFT) are generally scratch files. They include any file created locally at a control point and any copy of an indirect access file created by the GET command. These files are automatically released by 1CJ at job completion time. Tape files are also considered local files.

NOS supports common files (CMFT) only in locked mode. When a user wants the use of this type of file, he issues the COMMON command after locking a local file. LFM finds the FNT and creates a new FNT/FST for the user of type LIFT. This file is in read-only mode. Many users can read this file simultaneously, each with his own FNT/FST pointing to the same file. The user must be validated to access library files (bit CASF set in AACW).

Locked common files are CMFT files with the write lockout bit (bit 12) set in the FNT. The bit is set by the LOCK command and unset by the UNLOCK command. However, when the creator of the file returns it or terminates, and if the write lockout bit is set, then the file can never be unlocked or released, except by a level-zero deadstart, or with the console memory entry commands. It is not possible to create an unlocked common file. The local LIFT FNT/FST is released at return or end-of-job time, but the CMFT entry remains in the FNT and the file space is not dropped.

LOCAL FILE MANAGER

Local file management consists of a set of macros, control statements, and the PP routine LFM. LFM can be called with the following.

- An RA+1 call to LFM.
- A macro which generates an RA+1 call to LFM.
- A control statement which causes either RESEX or FILES to be loaded in the user's FL. RESEX is loaded for the ASSIGN, LABEL, and REQUEST control statements (refer to Section 12 for the discussion of tape assignment). Local mass storage file assignment uses the same procedure as tape assignment. FILES issues the appropriate macro (RENAME, COMMON, or RELEASE, for example) to generate an RA+1 call to LFM.

When called to a control point, LFM locates or creates the FNT/FST for the desired file. It makes the appropriate changes in the FNT/FST entry for the function specified in the call. LFM also interfaces to RESEX if necessary.

The routine FILES also does file skipping, rewinding, and WRITER and WRITEF commands. It calls CIO for these tasks.

The macro definitions for LFM functions are found in SYSTEXT (and NOSTEXT) or in common deck COMCMAC or COMCCMD. Those macros defined in SYSTEXT are general use macros, such as RENAME, STATUS, and LABEL, while those macros defined in COMCMAC and COMCCMD have special usage, such as the ACCSF, ENCSF, and PSCSF macros used by the control language. All LFM macros require common decks COMCLFM and COMCSYS. A detailed description of each LFM function and its macro call is available in the NOS Reference Manual, volume 2.

The PP program LFM consists of a group of overlays that performs the requested function. The functions and their corresponding LFM overlays are outlined in table 11-1.

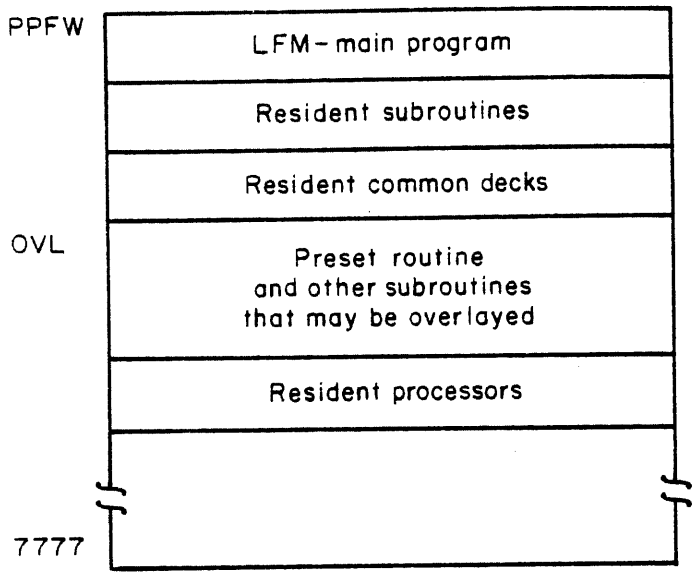
TABLE 11-1. LFM OVERLAYS

Code	Function	Overlay	Entry Point
0	Rename file	LFM	RNI
1	Assign common file	3LD	ACF
2	Enter common file	3LD	ECF
4	Release print file	3LE	RPR
5	Release 026 punch file	3LE	RPH
6	Release PUNCHB file	3LE	RPB
7	Release P8 file	3LE	RP8
10	Lock file	3LB	LCK
11	Unlock file	3LB	ULK
12	Return file status	3LB	RLS
13	Return current position	3LB	RCP
14	Request equipment	LFM	RQI
15	Assign equipment	3LC	AEQ
16	Release files	3LE	REL
17	Set file ID code	3LE	SID
20	Access library file	LFM	ALF
21	Attach control statement file	3LF	ACS
22	Enter control statement file	3LF	ECS
23	Position control statement file	3LF	PCS
24	LABEL request	LFM	LBI
25	Get all local FNTs	3LG	GTF

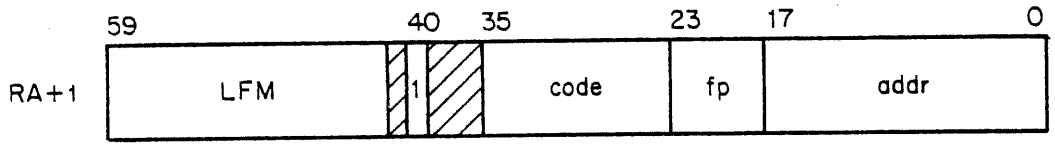
TABLE 11-1. LFM OVERLAYS (Continued)

Code	Function	Overlay	Entry Point
26	Request operator assignment of equipment	3LC	QAE
27	Enter VSN entry file	3LC	VSN
30	Release 029 punch file	3LE	RPN
31	Make file primary	3LG	PRI
32	Return file information	3LB	RFI

An outline of the LFM memory map is as follows. The map represents LFM code prior to loading a 3Lx overlay. All 3Lx overlays are loaded at location OVL.



The RA+1 call for LFM has the following format.



code	Function code
fp	Function parameter (used by SETID function 17, STATUS function 13, and equipment request function 26)
addr	Address of the FET

The main program (LFM) calls the preset subroutine (PRS) and then jumps to the appropriate subroutine to process the requested function (a 3Lx overlay is loaded, if required). LFM also contains the common return point, LFMX, for all function processors. LFMX sets the file status not busy, completes the FET status, and drops the PP.

The resident subroutines are called from the various function processors and include the following subroutines.

Subroutine	Description
CKE	Check error processing (bit 44 of FET+1)
CPA	Compute parameter address
DEQ	Drop (release) equipment
DRF	Drop file (calls ODF)
EFN	Enter new file name in FNT (calls OBF)
ABT	Abort job
ERR	Process error (calls 3LA)
RCL	Recall LFM
SFS	Set file status
SIS	Seach for and interlock file
SPB	Set/clear pause bit
SVF	Search for VSN entry file

Resident common decks include the following.

- COMPSAF Search for assigned file
- COMPSEI Search for end-of-information
- COMPSFB Set file busy
- COMPSRA Set random address
- COMPVFN Verify file name

PRS checks the input register for a valid call to LFM, determines what overlay is needed to process the requested function, initializes some memory cells, and returns to LFM. Other subroutines, besides PRS, that are overlayed include the following.

- CRX Determine if RESEX has been called.
- TFCN Function table. Specifies the entry point address of the routine to process the requested function, and in which overlay that entry point is defined.

Resident processing routines are those routines not requiring the loading of a special 3Lx overlay or requiring special processng before loadng a 3Lx overlay. They are resident in LFM due to high volume use or special initialization processing. They include the following.

- ALF Access library file (function 20)
- RNI Rename (function 0)
- RQI Request equipment (function 14)
- LBI LABEL request (function 24)

Also part of the resident processing routines are common deck COMPCRS (check recall status) and subroutine CRF (create relocatable routine FNT) and SSD (set system devices). The resident processing routines are also overlaid by 3Lx overlays.

LFM OVERLAYS

The following paragraphs describe the LFM overlays.

3LA - ERROR PROCESSOR

Overlay 3LA contains the code and messages for error processing.

3LB - LOCAL FILE FUNCTIONS

Overlay 3LB contains the rename (0), lock (10), unlock (11), return last status (12), return current position (13), and return file information (32) functions. The routines in 3LB are as follows:

Routine	Description
RNM	Rename file
LCK	Lock file
ULK	Unlock file
RLS	Return last status
RCP	Return current position
TUW	Transfer UDT words
RFI	Return file information
SFN	Seach FNT for assigned file

3LC - EQUIPMENT REQUESTS

Overlay 3LC contains those functions used in assigning an equipment to a given file. These functions are request equipment (14), assign equipment (15), label tape request (24), request operator assignment of equipment (26), and build tape file FNT/FST (27). The routines in 3LC are as follows:

Routine	Description
REQ	Request equipment
AEQ	Assign equipment
LBR	Label tape request
OAE	Request operator assignment of equipment
VSN	Build tape file FNT/FST
ASF	Assign nontape file
CEN	Check equipment number
FTE	Find tape equipment
REA	Request equipment assignment
ROA	Request operator assignment
SEQ	Search for equipment
VAE	Validate assigned equipment
VSJ	Validate SSJ=
VSO	Validate system origin privileges
VTE	Verify tape entry

Common decks COMPACS (assemble character string) and COMPC2D (convert 2 octal digits to display code) are also used in 3LC.

3LD - COMMON FILE FUNCTIONS

Overlay 3LD contains the assign common file to job (1) and enter common file (2) functions. The routines in 3LD are as follows:

Routine	Description
ACF	Assign common file to job
ECF	Enter common file
SCF	Search for common file
USS	Update system sector

Common deck COMPWSS (write system sector) is also used in 3LD.

3LE - FILE DISPOSAL FUNCTIONS

Overlay 3LE contains those functions related to disposing a local file to an output queue. These functions are release file to print queue (4), release file to 026 punch queue (5), release file to binary punch queue (6), release file to 80-column binary punch queue (7), release file to 029 punch queue (30), release file to corresponding queue (16), and set file ID code (17). The routines in 3LE are as follows:

Routine	Description
RPR	Release file to PRINT queue
RPH	Release file to 026 PUNCH queue
RPB	Release file to PUNCHB queue
RP8	Release file to P8 queue
RPN	Release file to 029 PUNCH queue
REL	Release file to corresponding queue
SID	Set file ID code
ROF	Release output file
COL	Check output file limit
COT	Change origin type
ISB	Initialize system sector buffer

Overlay 3LE contains the following common decks.

Common Deck	Description
COMPC2D	Convert two octal digits to display code
COMPRSS	Read system sector
COMPCUN	Compare user numbers
COMPCVI	Convert validation indexes
COMPSSE	System sector error processor
COMPUSS	Update system sector for disposable files
COMPWSS	Write system sector
COMPRJC	Read job control word

3LF - CONTROL STATEMENT FILE FUNCTIONS

Overlay 3LF contains the functions used to manipulate the control statement record by the job control language. These functions are attach control statement file (21), replace control statement file (22), and position control statement file (23). The routines in 3LF are as follows:

Routine	Description
ACS	Attach control statement file under name from FET
ECS	Replace control statement file
PCS	Position control statement file
PCF	Position control statement file

Common decks COMPCRA (convert random address) and COMPRNS (read next sector) are included in 3LF.

3LG - GETFNT AND PRIMARY FUNCTIONS

Overlay 3LG contains the GETFNT (25) and PRIMARY (31) functions. The routines in 3LG are as follows:

Routine	Description
GTF	Return table with FNT/FST entries for all working files
PRI	Process primary function
CCP	Crack calling parameters in FET and CGNT
CFS	Check file selectivity
MFF	Modify file's FST
PCF	Process checkpoint file

Overlay 3LG contains the following common decks.

Common Deck	Description
COMPWEI	Write EOI sector
COMPWSS	Write system sector

Resource control involves the allocation of the system magnetic tape and removable pack resources. The control of these resources is handled by the NOS resource executive (RESEX).

RESEX manages the assignment of magnetic tape and removable pack resources to user jobs without causing a deadlock to occur for those resources.

This section describes the concept of resource overcommitment and the scheduling (assigning) of resource units by RESEX.

OVERCOMMITMENT

A portion of RESEX deals with managing the use of tape units and disk packs in such a way as to prevent deadlocks from occurring. The whole process is generally referred to as overcommitment, although the overcommitment of tape and disk resources is just a part of it.

Before continuing the discussion of overcommitment, the following definitions are necessary.

- A deadlock or potential deadlock condition exists when two or more jobs demand tape and pack resources such that no more resources are available (deadlock), or there are not enough free resource units available to satisfy the resource requirements of any of these jobs (potential deadlock).
- An internal conflict condition exists if a job's current tape request or increased resource demands would deadlock the job itself. This can occur only when using PE resources (1600 cpi, 9-track tapes).
- The resource demand for a job is the number of resource units that the job requires concurrently. This demand may be for a single resource type (for example, 7-track tape) or a combination of resources (for example, 7- and 9-track tape and a DI-2).
- Demand count and assigned count indicate the number of resource units required (resource demand) and the number of resource units assigned to the job respectively.
- The demand file is a fast attach permanent file manipulated by RESEX and its associated program (for example, ORF) which contains the assigned and demand information for each job using tape and pack resources. The entry in this file for a job is called the demand file entry.
- A resource unit is any magnetic tape or removable pack.

- The overcommitment algorithm is a subroutine in RESEX that processes demand file entries and determines whether jobs can be completed based on current resource usage. It also refers to the process of presetting tables that are used during the execution of the algorithm.

- The resource environment consists of all magnetic tape and removable disk drives currently defined in the system and the status of the tape or pack mounted on these drives. The environment is rarely static, as the assignment, mounting/dismounting, and status of the drives change during normal system operation. RESEX builds two working tables, the resource equipment table (RET) and the environment VSN buffer (EVSB), from information contained in the EST, MST, and MAGNET unit descriptor tables (UDT).

- The share table provides RESEX with the information concerning which job is using which removable device. The active user count for a removable pack found in the MST tells only how many direct access files are being accessed, but does not indicate who is using them. An entry is made for each pack assigned to a job in the job's share table. The share table is part of the demand file entry.

DEADLOCK PREVENTION

RESEX is able to prevent deadlock situations by requiring that a RESOURC control statement be included in any job that uses more than one resource unit concurrently. Using the information provided on the RESOURC statement, RESEX builds a demand file entry for the job and writes it to the demand file. Each job that uses tape or pack resources must have a demand file entry. RESEX automatically creates a demand file entry for any job requesting tape or pack assignment, if an entry does not already exist for that job. Part of creating a demand file entry consists of calling subroutine DEI (demand exceeds installation) to check whether there are enough resource units available to satisfy the job's demand, and calling subroutine CRV (check resource validation) to verify that this user has been validated for tape or pack resource usage.

To control overcommitment and prevent deadlock for 800/1600 and 1600/6250 cpi, 9-track tape drives, resource identification by density is provided. The resource identifier NT is retained for upward compatibility; however, it can be used only to satisfy 800 and 1600 cpi tape requests and cannot be specified concurrently in the same job with density resource identifiers HD (800 bpi, 9-track), PE (1600 cpi, 9-track), and GE (6250 cpi, 9-track). Assignment of a 9-track tape with no prior demand count defaults to the density resource specification. Density resource identifiers LO, HI, and HY are also provided for 7-track tape units; however, they all map into the MT resource demand entry.

In the overcommitment algorithm, unsatisfied NT resource demands are logically satisfied by 800/1600 cpi, 9-track drives; however, if at request time the desired 1600 cpi tape is found mounted on a 1600/6250 cpi drive, the assignment is made if it does not cause overcommitment.

For jobs with HD, PE, or GE resource demands, RESEX attempts to satisfy the PE demand (in the overcommitment algorithm) with any 9-track drives remaining after the job's HD and GE demands have been logically satisfied.

For a job with a PE resource demand, if the desired 1600 cpi, 9-track tape is found mounted on any 9-track drive, the assignment is made, unless it causes overcommitment or an internal conflict. Internal conflict is when tapes are attempting to be assigned to a job such that the current request would deadlock the job. For example, on a system with two 800/1600 and two 1600/6250 cpi, 9-track drives, a user job with resource requirements of PE=2 and GE=2 deadlocks itself if a PE tape found mounted on a 1600/6250 drive is actually assigned. Since this assignment cannot be allowed and the job cannot continue until that tape is dismounted from that drive type, RESEX informs the operator that this situation has occurred via the E,P display.

OVERCOMMITMENT ALGORITHM

The overcommitment algorithm has two basic parts, the environment and overcommitment determination. The environment and the demand for its resources are examined by RESEX to determine if the assignment of resources in the environment is such that a deadlock exists. The algorithm itself determines that there does or does not exist a sequence of job completions such that all jobs with assigned resources will complete. If all jobs with resources can eventually complete, then a deadlock does not exist.

Subroutine BRE (build resource environment) examines the EST, MST, and the UDT from MAGNET and builds two working tables, the RET and EVSB, for use by the overcommitment algorithm. Figure 12-1 contains the flowchart for BRE.

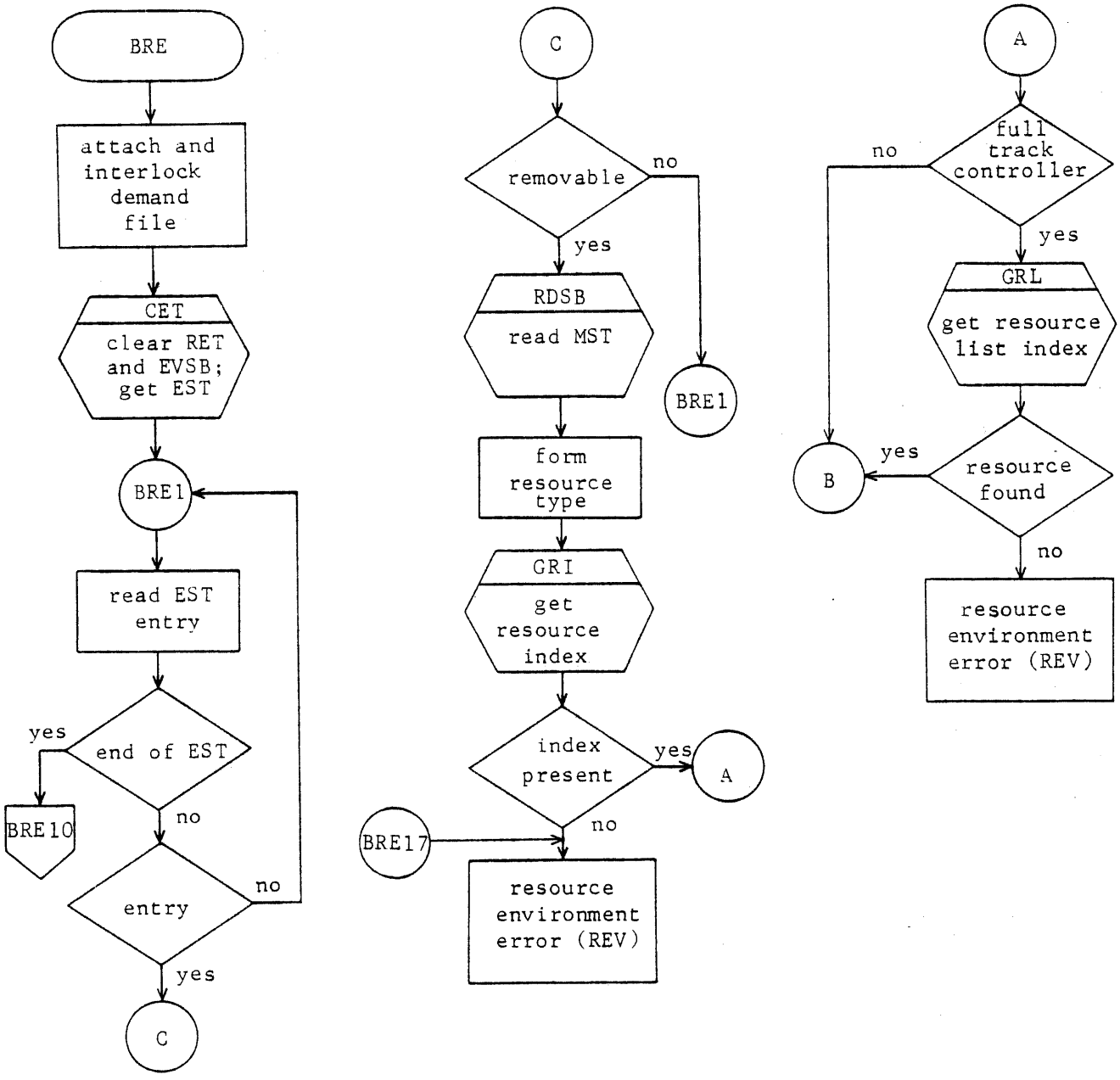


Figure 12-1. BRE - Build Resource Environment

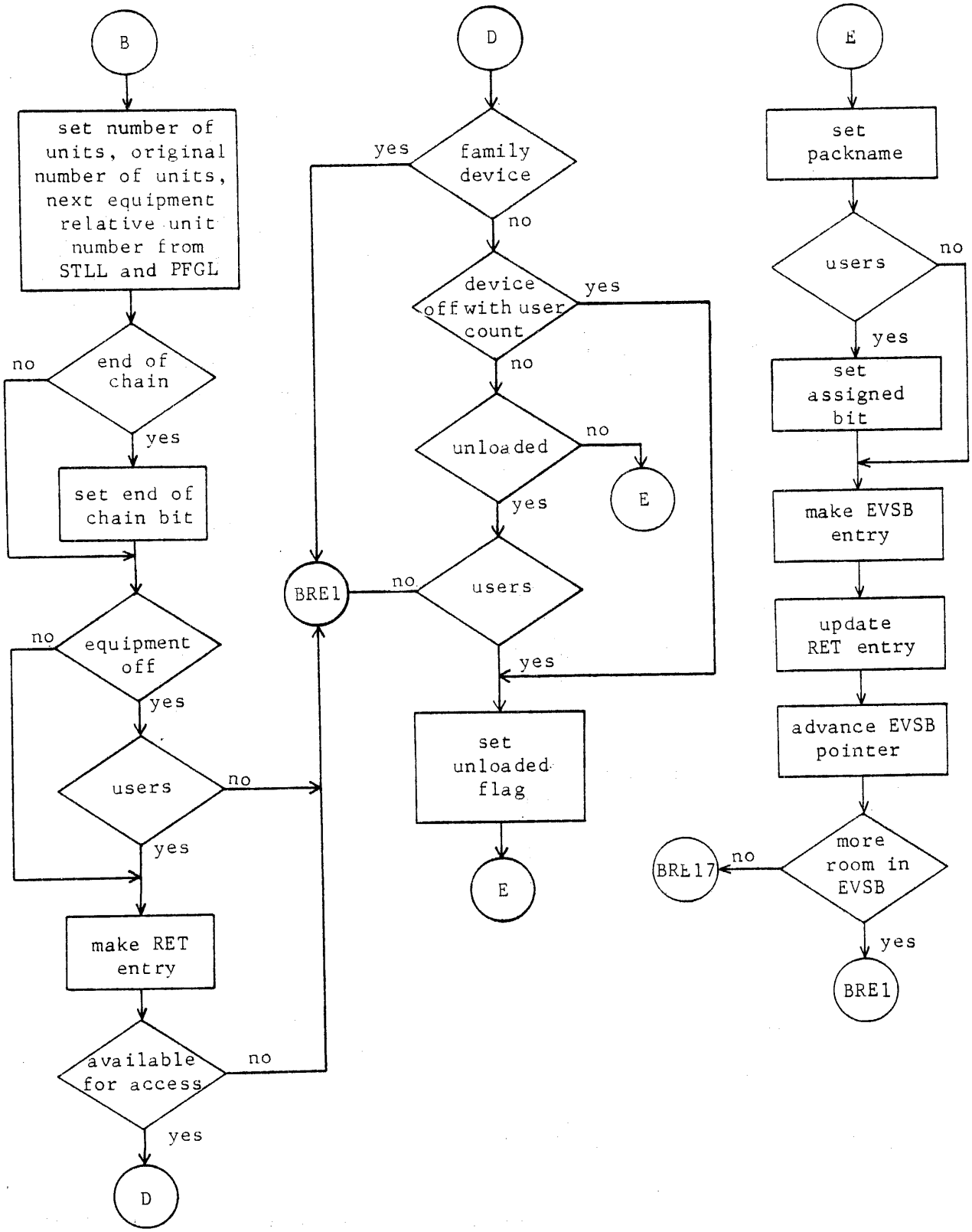


Figure 12-1. BRE - Build Resource Environment (Continued)

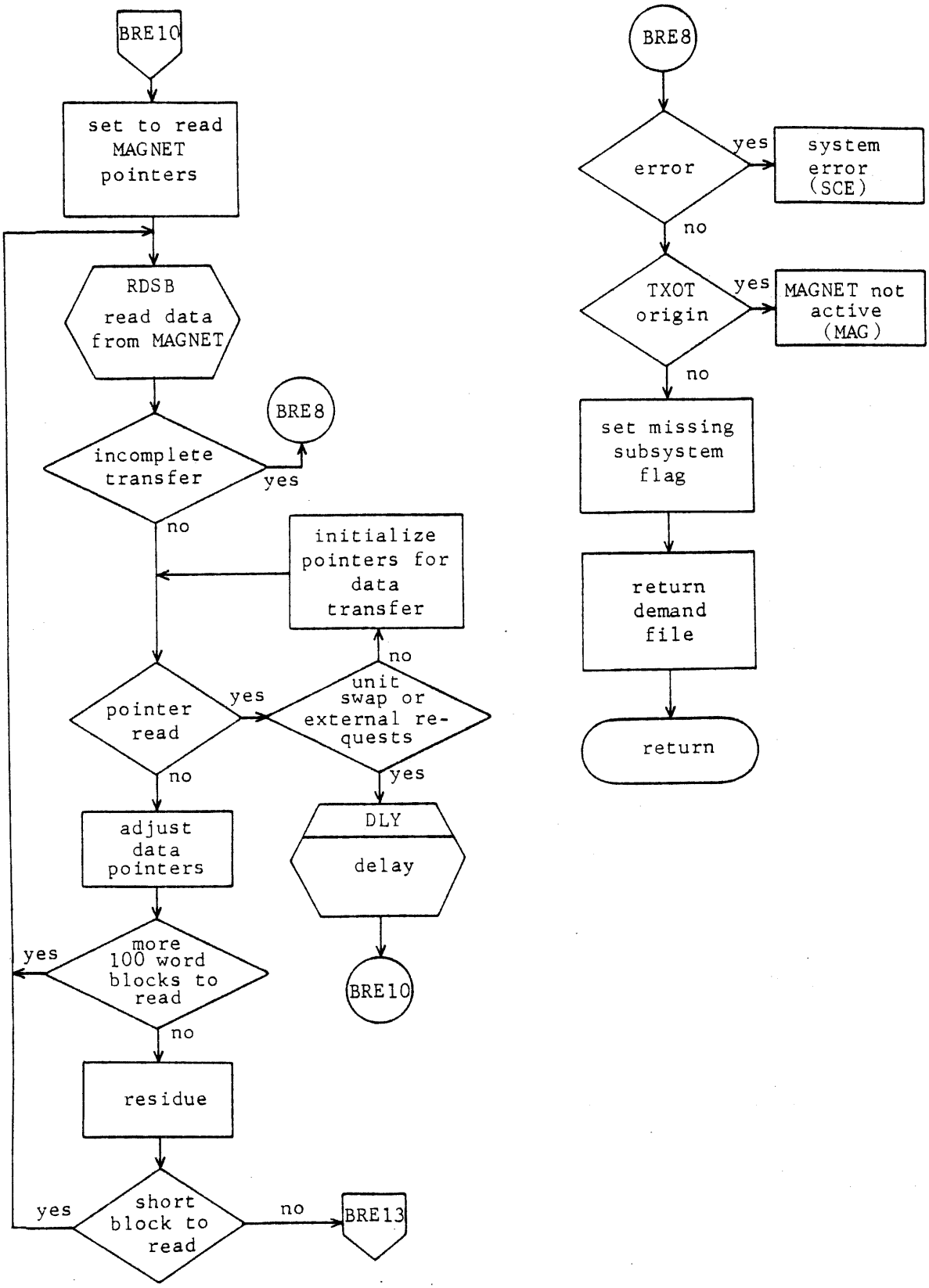


Figure 12-1. BRE - Build Resource Environment (Continued)

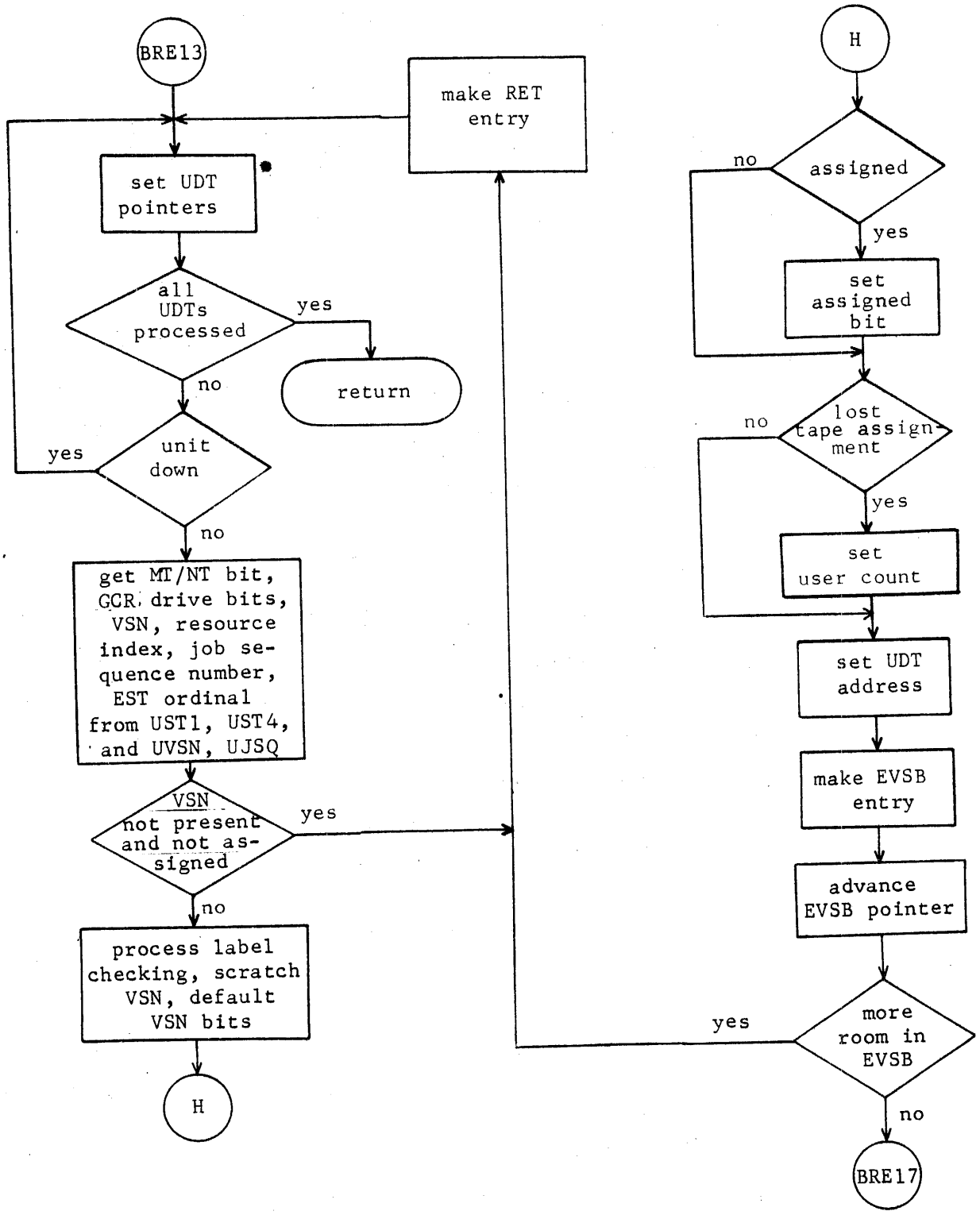
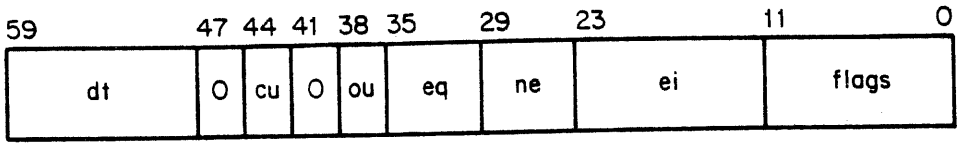


Figure 12-1. BRE - Build Resource Environment (Continued)

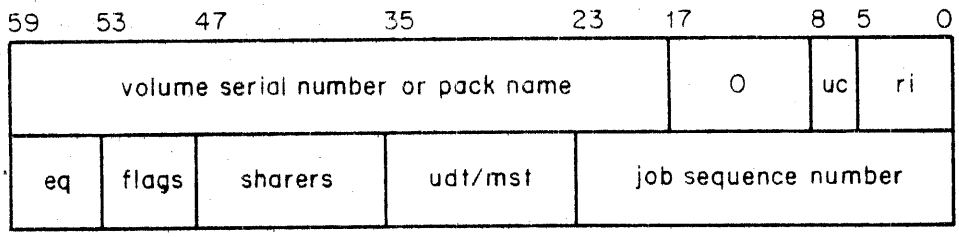
The RET consists of a combination of data collected from the EST, MST, and UDT tables. It contains one-word entries and is the same length as the EST. The format of an RET entry is as follows:



Field	Description	Origin: Tapes	Origin: Packs
dt	Device type if bit 59 is zero; otherwise dt is an index into a table of equivalent resource types	UDT/UST1	EST
cu	Current number of units in chain		EST
ou	Original number of units in chain		MST/STLL
eq	Equipment number (EST ordinal)	UDT/UST4	EST
ne	Pointer to EST entry of next pack in chain		MST/STLL
ei	EVSB index + 2 (if any)		
flags	Each bit defined as follows:		

Bit	Description		
11	Checking labels done by MAGNET	UDT/UVSN	
10-4	Unused		
3	LDAM equipment		EST
2	End of chain of packs		MST/PFGL
1	Unused		
0	Unit logically assigned		

The environment VSN buffer contains data relating to mounted magnetic tapes and removable packs. An EVSB entry is two words of the following format.



Field	Description
ri	Resource index; points to a word in the demand file entry between RMTP and RQPD
uc	Unit count (1 to 8) for packs; always 1 for tapes
eq	Equipment number
flags	Each bit defined as follows:

Bit	Description
53	Assigned
52	Scratch VSN
51	Unloaded pack with users
50	Default VSN
49-48	Unused

sharers	Number of users sharing pack
udt/mst	UDT address if tape, or MST address/10B if disk

The overcommitment algorithm considers only those jobs that have currently assigned resources. A scratch file is built from the demand file entries for those jobs. Each scratch file entry is then processed sequentially. If there are unsatisfied demands for a given entry, an attempt is made to determine if sufficient logically free resources are available to fulfill the demand. If all the resource demands for the job are satisfiable, then the job is said to complete. Jobs that complete have their resources logically freed for use by other jobs.

If a job's demands cannot be satisfied, the demand entry is written to a second scratch file. Processing continues with the next entry in the first scratch file until it is exhausted. At the end of the first scratch file (a pass through the algorithm) if there are no uncompleted jobs (second scratch file is empty), then the algorithm has successfully completed and, based on the snapshot of the environment used during the execution of the algorithm, no deadlocks on resources will occur. If entries had been written to the second scratch file and some job did complete during this pass (the number of entries on the first and second scratch files are not equal), the scratch files are interchanged and another pass through the algorithm is made. If no job completed on a pass through the algorithm (both scratch files are identical), then there is a deadlock on the resources and overcommitment is said to have occurred. The overcommitment algorithm determines whether or not a deadlock will occur; the action taken by RESEX to prevent the deadlock is detailed later in this section. The overcommitment algorithm, OCA, is shown in figure 12-2.

RESOURCE FILES

To aid in the allocation of magnetic tape and removable pack resources, RESEX maintains two mass storage files. These files are known as the resource files and are fast attach, direct access permanent files. The resource demand file (RSXDid) contains the maximum concurrent demand for each resource type. It also contains the share table and information for the preview display. The format of the RSXDid file is in figure 12-3.

In figure 12-3, RVAL contains the validation limits (that is, the number of pack and tape units allowed to be assigned to this user). This is obtained from location ALMS in the user's control point area. RMTP through RGEP are five words for 7- and 9-track tape parameters. RRPP is the beginning address of 10 words of removable pack parameters (leftmost 12 bits are in display code), where each AD(n) contains the assigned and demand counts for this resource with n physical units (for example, DI-1, DI-2,..., DI-8) amd mpu defines the maximum number of physical units allowed for this resource. Finally, RRPS begins the share table, which is a list of removable packs assigned to the job.

The eq field of each share table entry contains the EST ordinal of the shared device; ri and uc specify the resource index and unit count for this device (defines a location in RRPP).

The lost tape assignment flag in RVAL is set by MAGNET1 when MAGNET is dropped with tapes assigned. RESEX uses this flag and an indicator in the UDT to adjust the environment to reflect those equipments whose assignments are lost, so these lost tapes do not affect other user jobs. Until a user job returns all of its lost tapes, it will be aborted with the following message issued when attempting a subsequent RESEX operation.

PRIOR TAPE ASSIGNMENT LOST.

The job aborts with the following mssage when attempting a subsequent I/O operation on a lost tape (any CIO function other than RETURN, UNLOAD, or EVICT).

I/O SEQUENCE ERROR ON FILE, fff AT nnn.

After the user job returns all of its lost tape files, RESEX clears the lost tape assignment flag in the user's demand file entry and the job can resume normal execution (pertinent to terminal and DIS users).

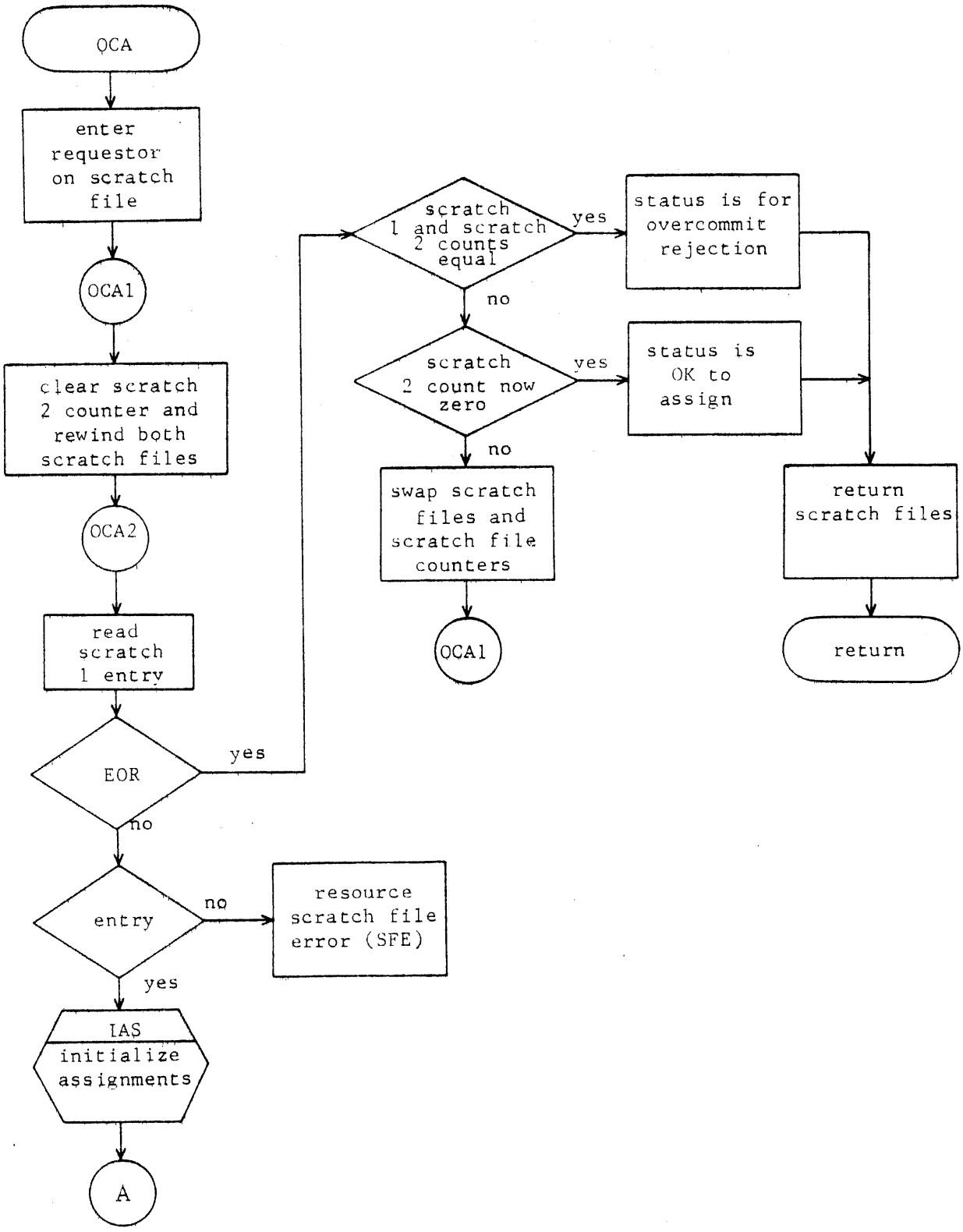


Figure 12-2. OCA - Overcommitment Algorithm

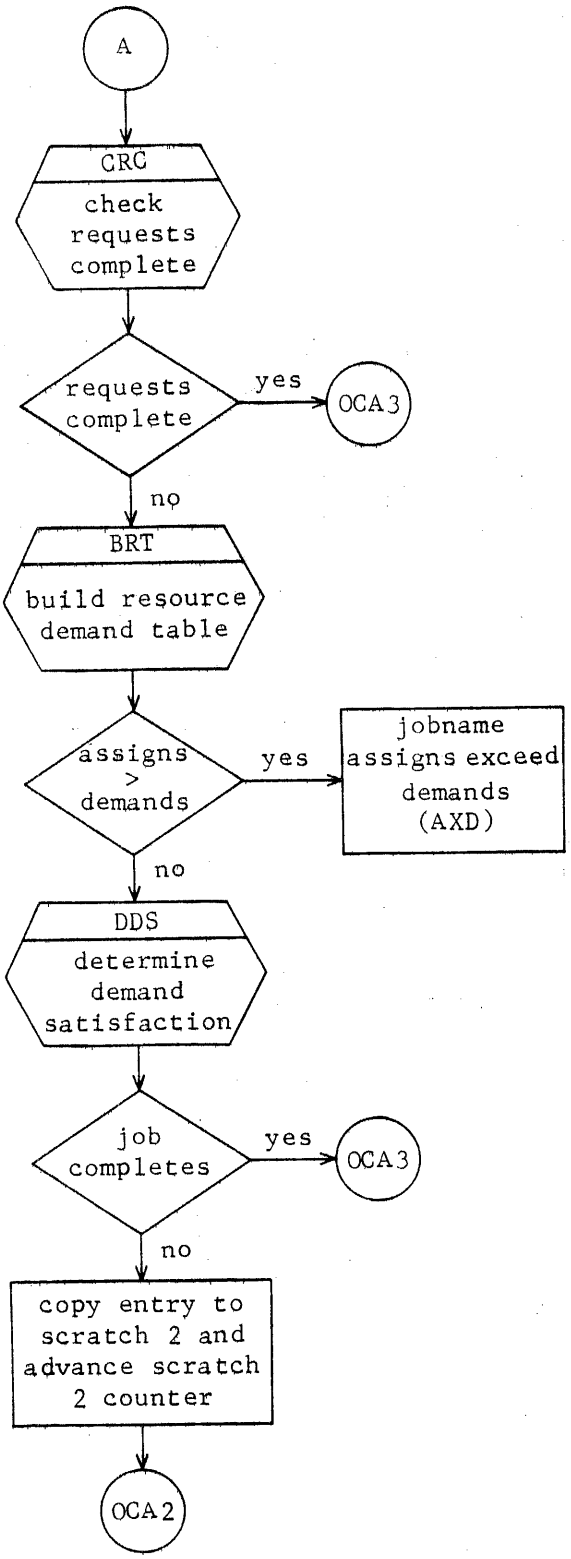


Figure 12-2. OCA - Overcommitment Algorithm (Continued)

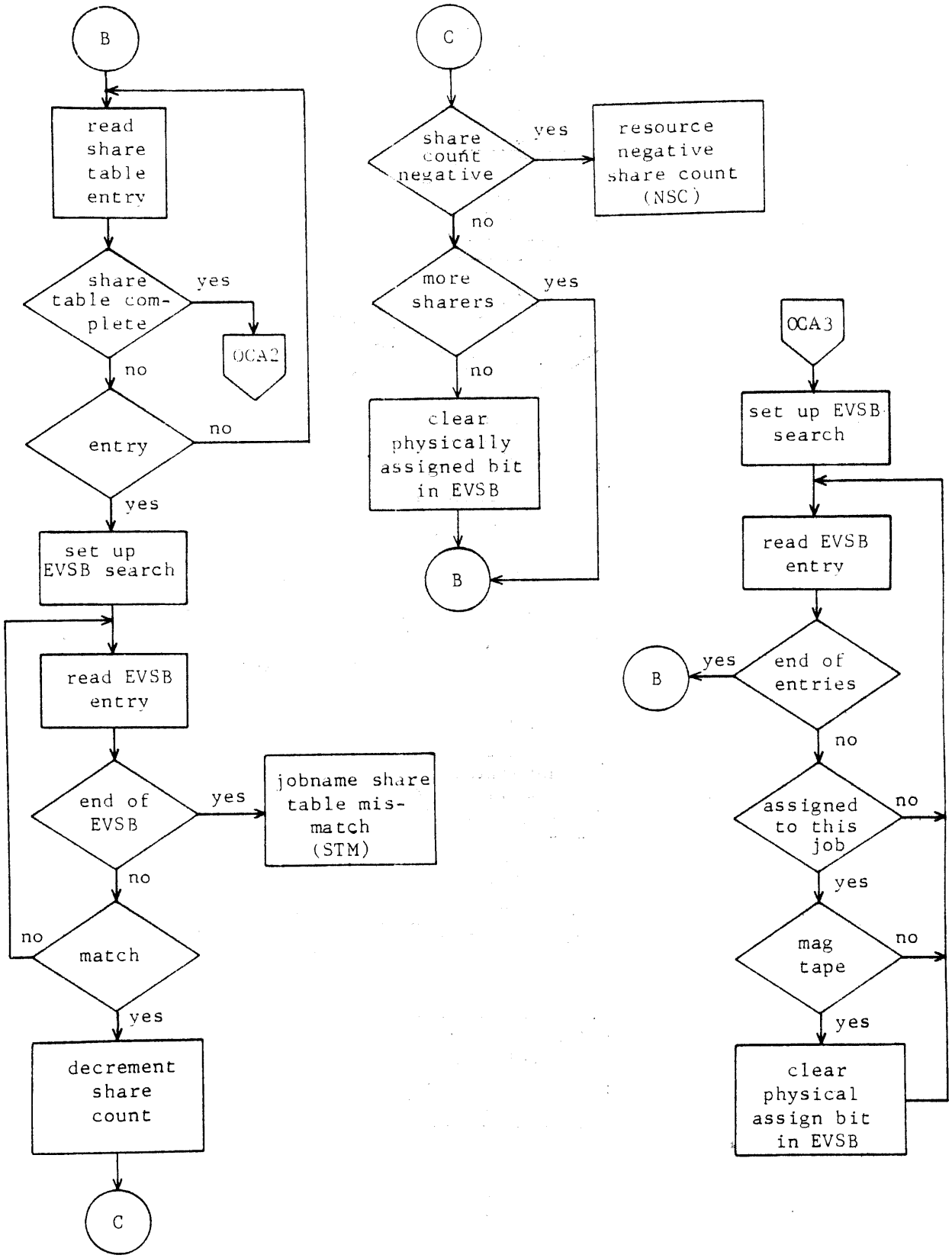


Figure 12-2. OCA - Overcommitment Algorithm (Continued)

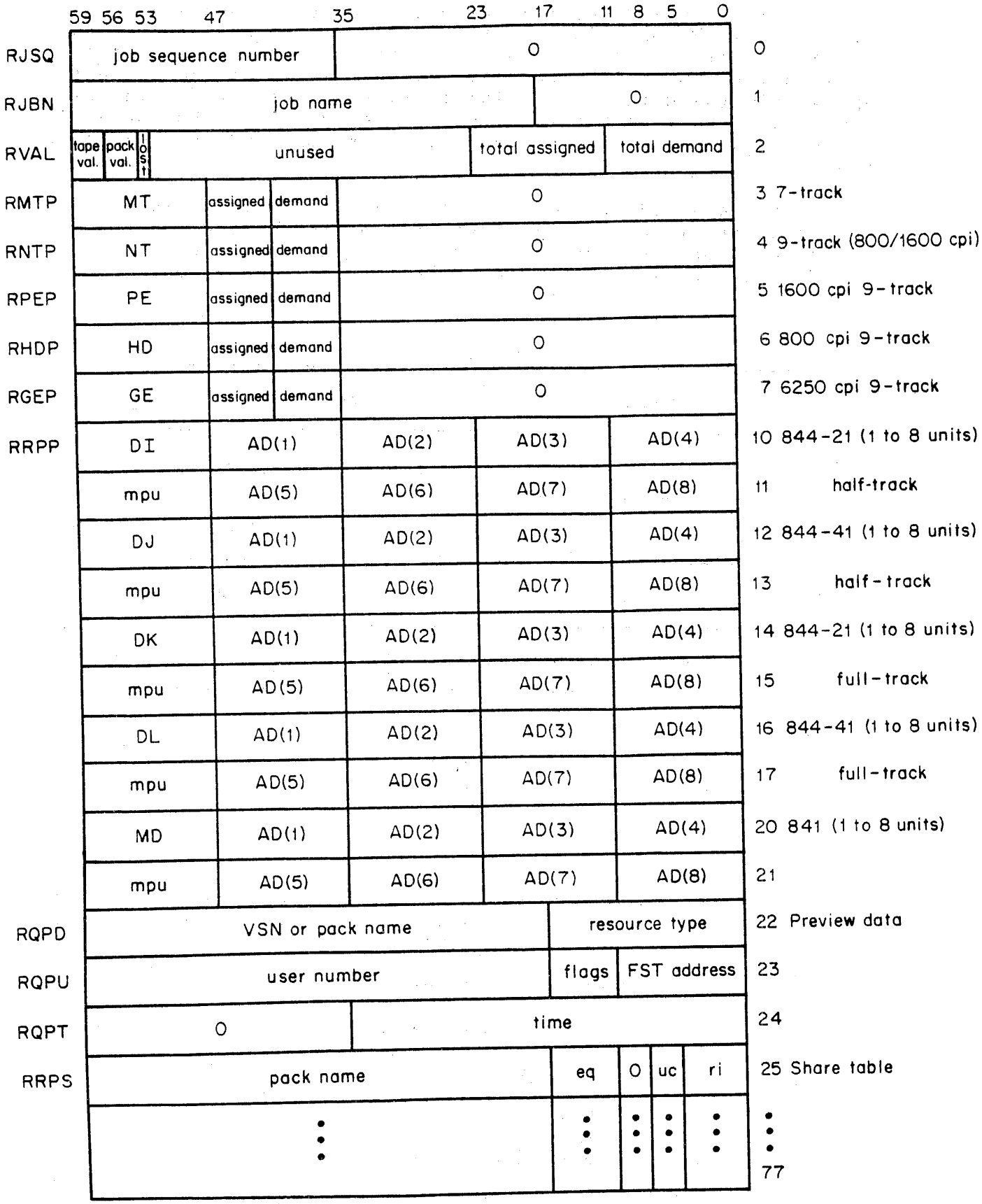
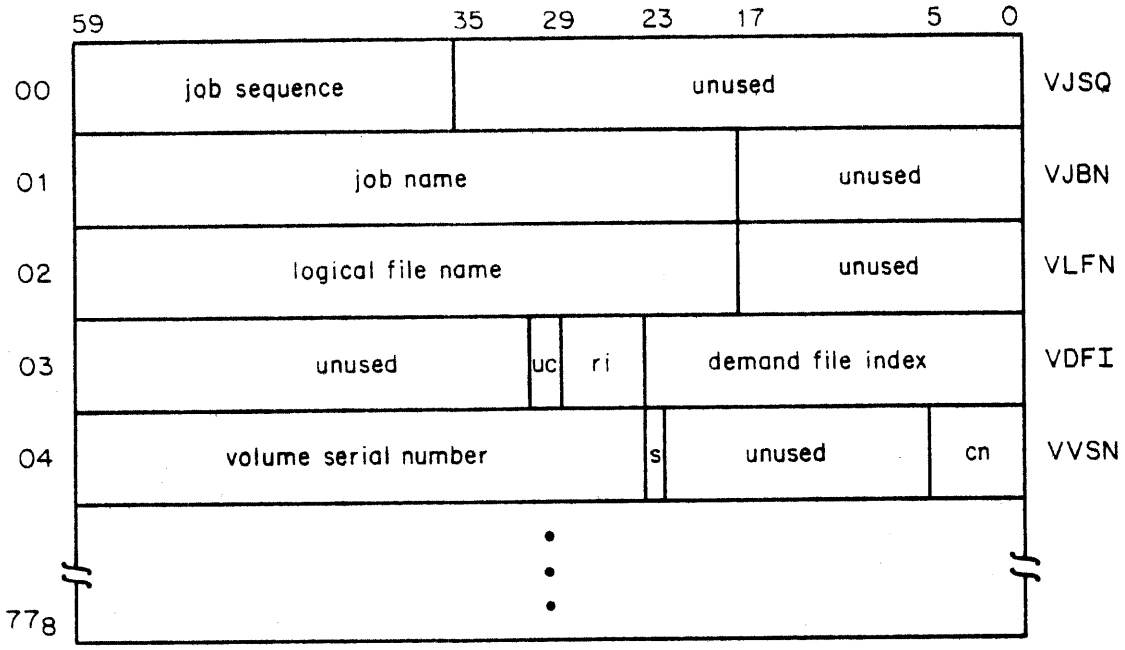


Figure 12-3. Resource Demand File Entry (RSXVid)

The VSN file (RSXVid) contains the volume serial numbers associated with a particular file (refer to figure 12-4). The entries in the VSN file contain the random index of the job's entry in the demand file and the resource index of the assigned tape within the demand file. The entries in these two files are associated with a particular job and are identified by the job's sequence number. Entries in both files are one PRU (64 words) in length.

These files are updated by RESEX and PP routine ORF. Routine ORF (described later in this section) updates demand and VSN file entries upon the return/unload of a tape file or the job's last direct access file on a pack and at job completion.



uc — Unit count (always 1)

ri — Resource index; points to a word in the demand file entry between RMTP and RGEP; zero if tape is not yet assigned

s — Scratch VSN (if set)

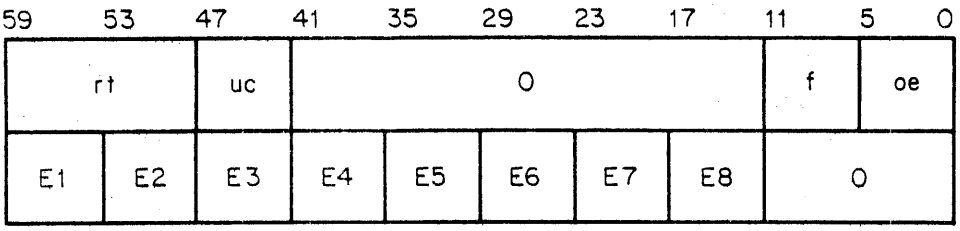
cn — Control byte as follows:

Value	Meaning
/	Multireel
=	Alternate reel
0	End of entries

Figure 12-4. VSN File Entry (RSXVid)

Resource Satisfaction

The logical satisfaction of unfulfilled demands is done by subroutine determine demand satisfaction (DDS). DDS works with the RET and a temporary table called the resources demanded table (RDT), which is formatted as follows:



rt	Resource type
uc	Unit count (1 to 8 for packs, 1 for tapes)
f	Flags
oe	Original equipment
E1-E8	Equipments being used to (logically) satisfy the demand

The RDT consists of the number of unsatisfied resource demands for the job. The satisfaction of tape demands by DDS is simplified since there is only one user for each tape. However, with removable packs there can be sharing and combinations. Sharing is having more than one accessor for each pack. The number of user jobs sharing a pack is reflected in the EVSB entry for that pack. It contains a count of the number of demand file entries that contain a share table entry for that pack. However, it cannot be assumed that a pack is going to be shared.

The combination of removable packs allows the use of two DI-1s, for example, as a DI-2. The ability to combine removable equipments requires that a best bit determination be done by DDS; thus, the removable mass storage devices are used in a most efficient manner while logically satisfying demands. Mass storage demands are satisfied in a largest-number-of-spindles-first order. This assures that the best fit is based on the smallest unused spindle residue of combinations of removable devices. The operator's choice of removable pack mounting can cause an excessive amount of overcommitment, as the best fit of removable packs may not be available for a given job. Refer to figure 12-5 for a flowchart of DDS.

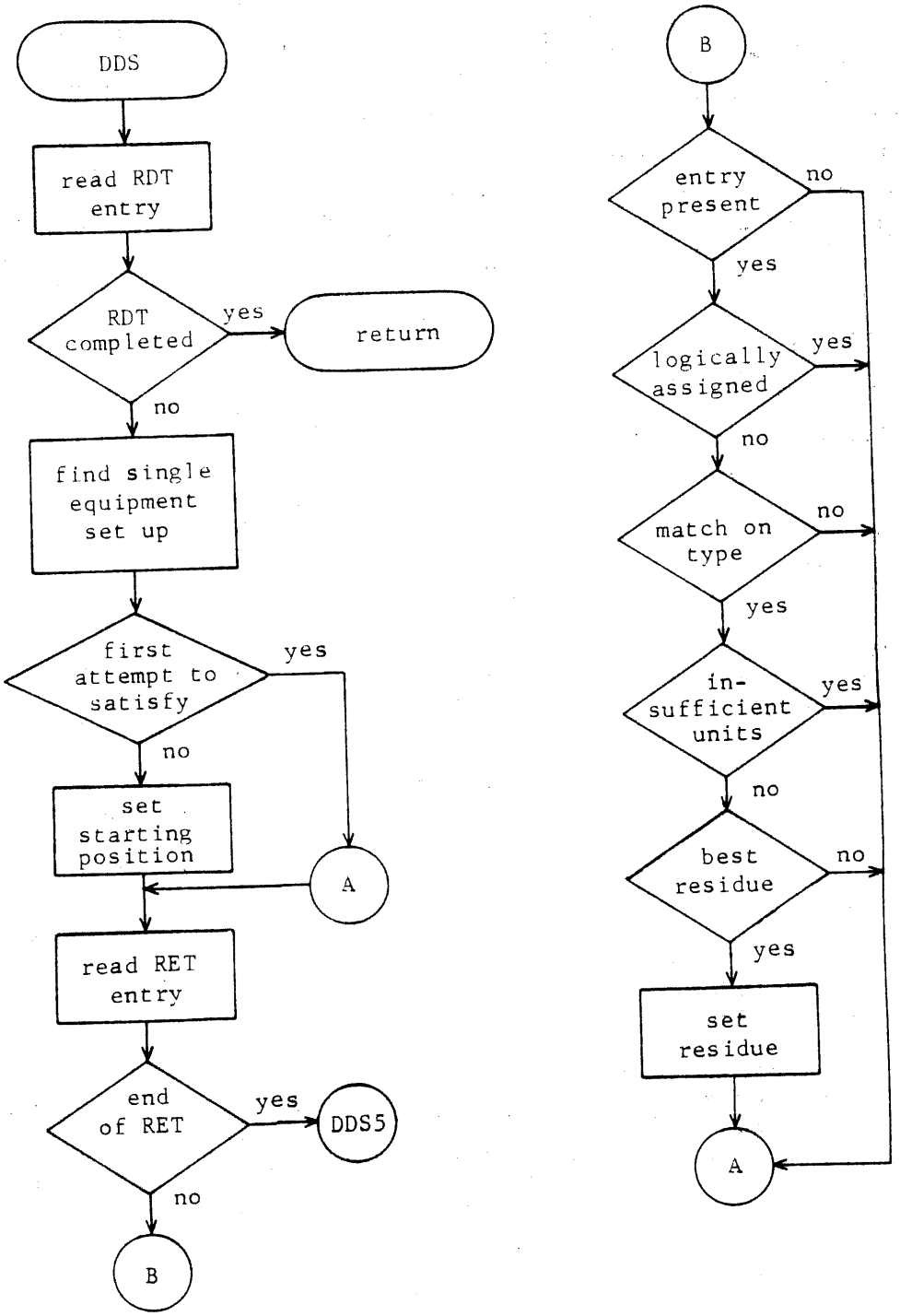


Figure 12-5. DDS - Determine Demand Satisfaction

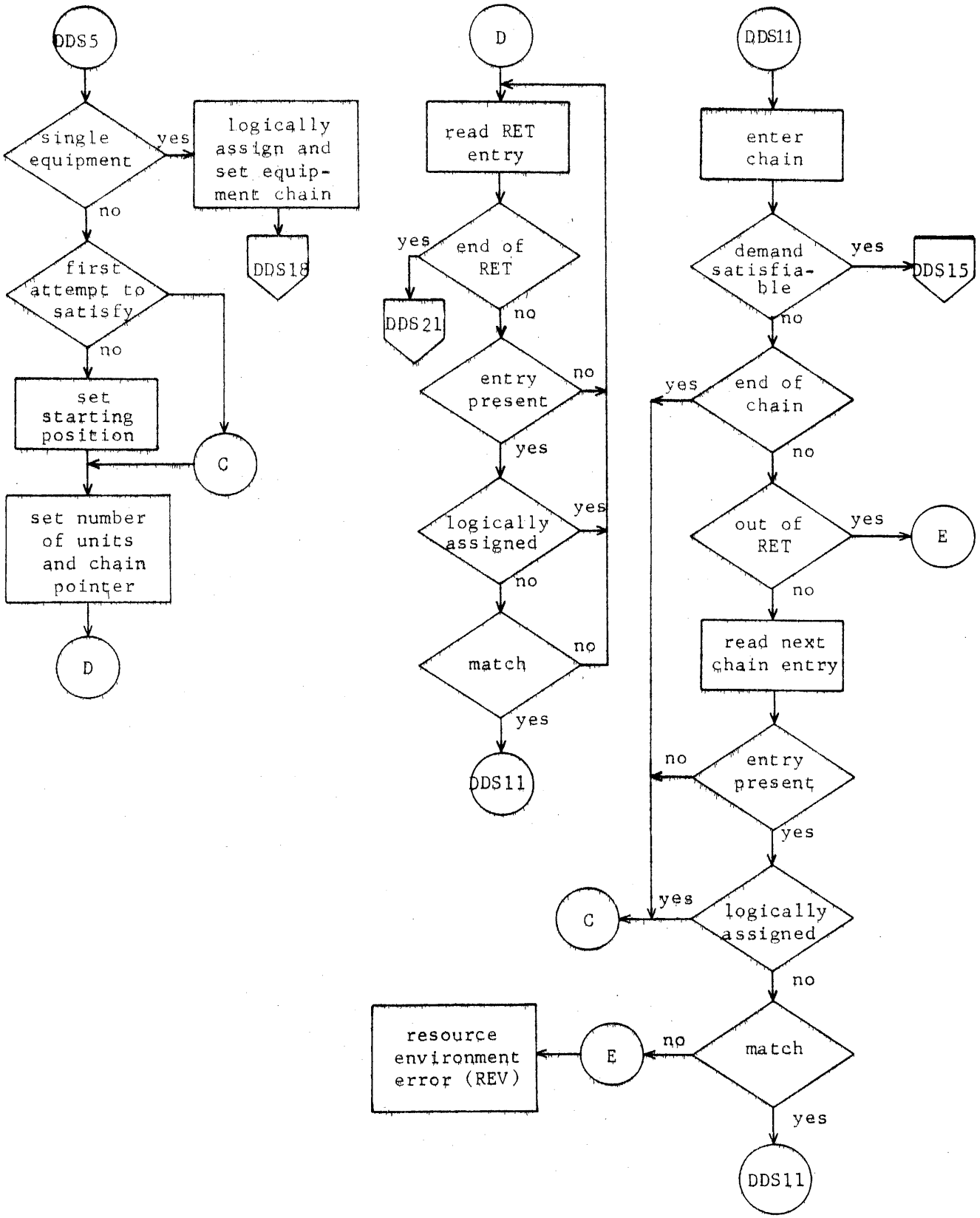


Figure 12-5. DDS - Determine Demand Satisfaction (Continued)

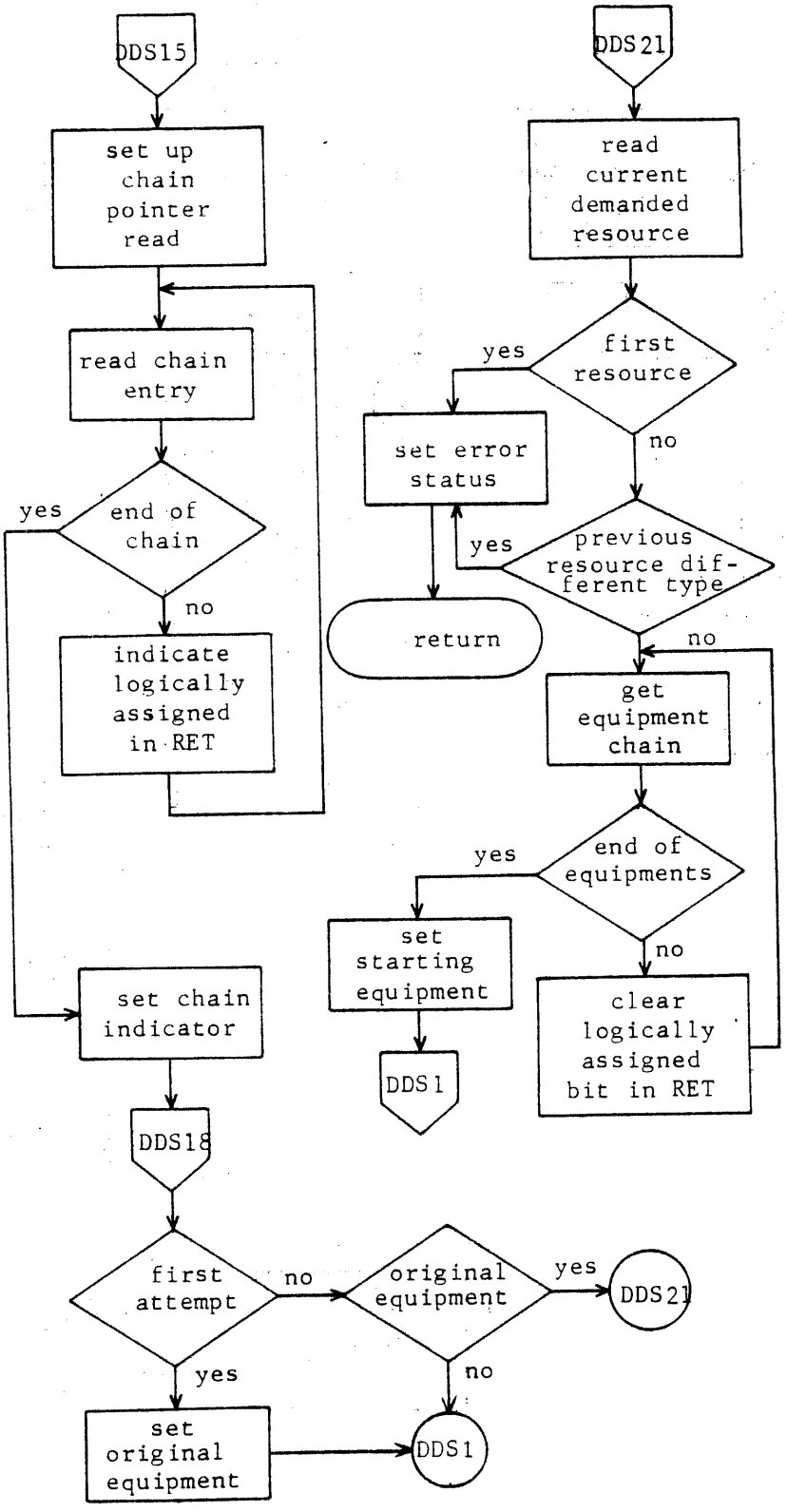


Figure 12-5. DDS - Determine Demand Satisfaction (Continued)

Resource Assignment Counts

The method of assigning units depends on the resource type. All tapes and private packs not accessible by alternate users can only be assigned to one job at a time. All public packs and those private packs accessible by alternate users can be shared and, therefore, assigned to several jobs at the same time.

On indirect access file requests, the pack is charged (assigned) to the job in fulfilling its resource demand only if the requested pack must be mounted. For direct access file requests, the pack is charged to the job when the first direct access (ATTACH or DEFINE) is made.

A unit is assigned to a job until the job terminates, or when all the direct access files residing on that unit that are assigned to the job are returned/unloaded, or a tape file is returned/unloaded.

The system keeps track of the resource usage in the demand file through the assigned and demand counts for each job. These values may vary during job processing. A user can increase or decrease demands through additional RESOURC statements. However, if the user attempts to decrease his demand below his assigned count or increase his demand such that a deadlock would occur, a diagnostic is issued to the user's dayfile and the job is aborted.

The demand and assign counts are also adjusted when files are returned or unloaded. When a magnetic tape file or the last file assigned to the job residing on a removable pack is unloaded, the resource assign count is decremented (by routine ORF). If the file was returned, the resource demand count is also decremented, but only if all of the job's demands have been satisfied. Whenever there is only one concurrent resouce demanded by the job, both the assign and demand counts are cleared regardless of the RETURN or UNLOAD function used to drop the file

RESOURCE EXECUTIVE

The remainder of this section is devoted to the internals of RESEX.

RESEX processes requests for magnetic tapes and removable packs and manages the usage of these resources. The following control statements are processed by RESEX.

- ASSIGN
- LABEL
- REQUEST
- RESOURC
- VSN

These control statements are described in detail in the NOS Reference Manual, volume 1. The control statements for removable packs are part of CPU routine PFILES and are also described in the reference manual.

CONTROL STATEMENT PROCESSING

The control statements processed by RESEX fit into three groups: assignment (ASSIGN/LABEL/REQUEST), resource declaration (RESOURC), and VSN association (VSN).

Assignment Statements

The assignment statements perform the same function but with variations in default values and parameter processing. REQUEST and LABEL are virtually identical except that the parameters associated with tape labels can only be specified on the LABEL statement. ASSIGN operates somewhat differently as it works with a specified device type or device number (EST ordinal), and must process these accordingly. All assignment statements transfer control to address LAB1 (refer to figure 12-6) for control statement processing. If a VSN is specified, RESEX automatically assigns this tape after it is mounted. If the request is for a tape, but no VSN is specified or if the request is not for a tape, RESEX calls LFM to return an operator equipment assignment. If the equipment selected by the operator is a tape unit, RESEX must then check for overcommitment before completing the tape assignment. A flowchart of the assignment control statement is in figure 12-6.

Resource Declaration

The specification of a job's resource demand is accomplished via a RESOURC control statement. The processing of the RESOURC statement includes the validation of resource types and determining if a change in resource specification will produce a deadlock or internal conflict. Figure 12-7 is a flowchart of the RESOURC routine.

VSN Association

The purpose of the VSN control statement is to associate magnetic tape volume serial numbers with the name of the file that will use these reels. The processing of the VSN statement includes building a VSN file entry for the file (and a demand file entry, if necessary). The processing of the VSN statement does not depend on MAGNET being active. The routine to process the VSN statement is also entered to process the VSN= control statement parameter that may be used on assignment cards. The flowchart for the VSN processor is shown in figure 12-8.

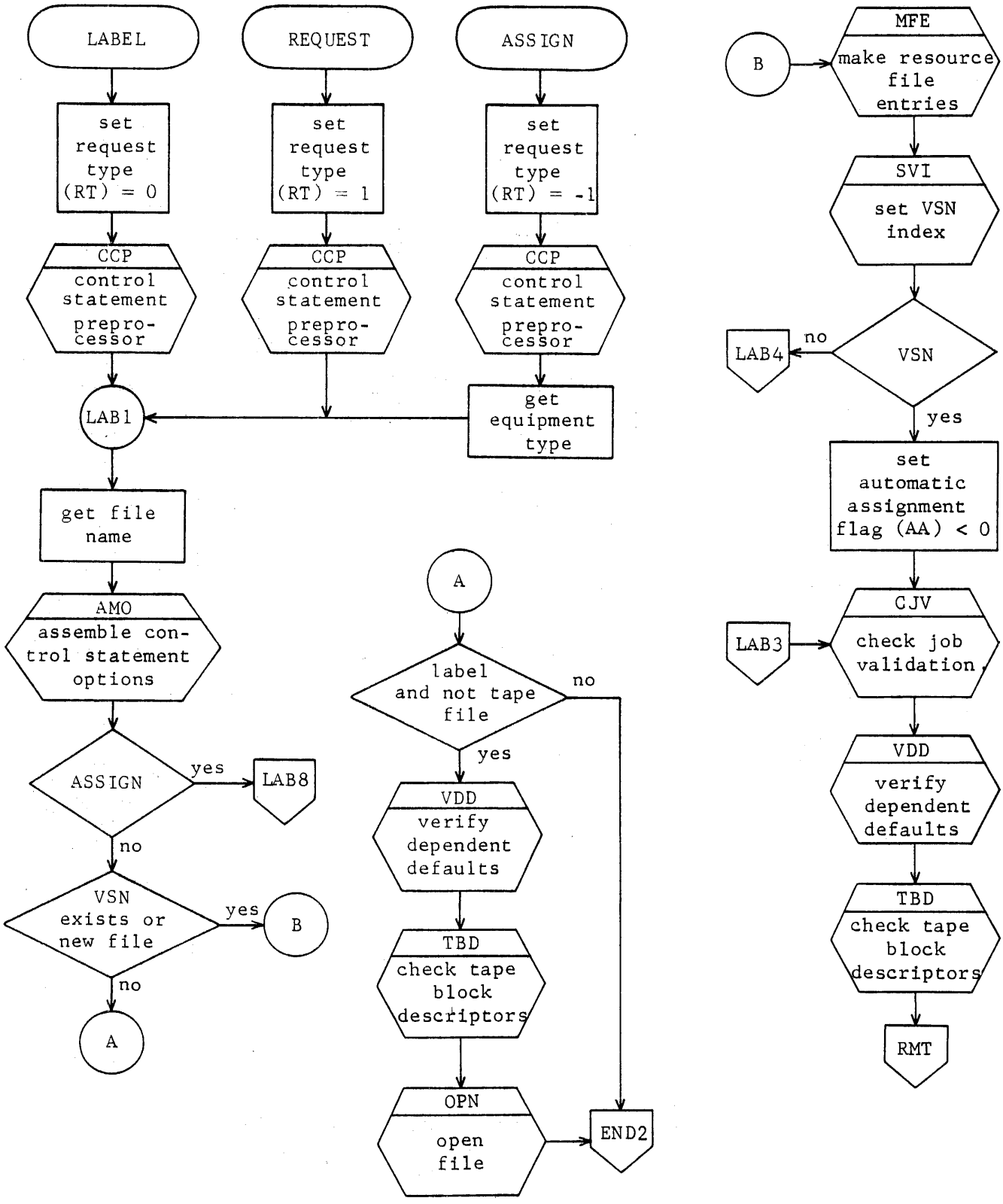


Figure 12-6. ASSIGN/LABEL/REQUEST - Assignment Control Statements

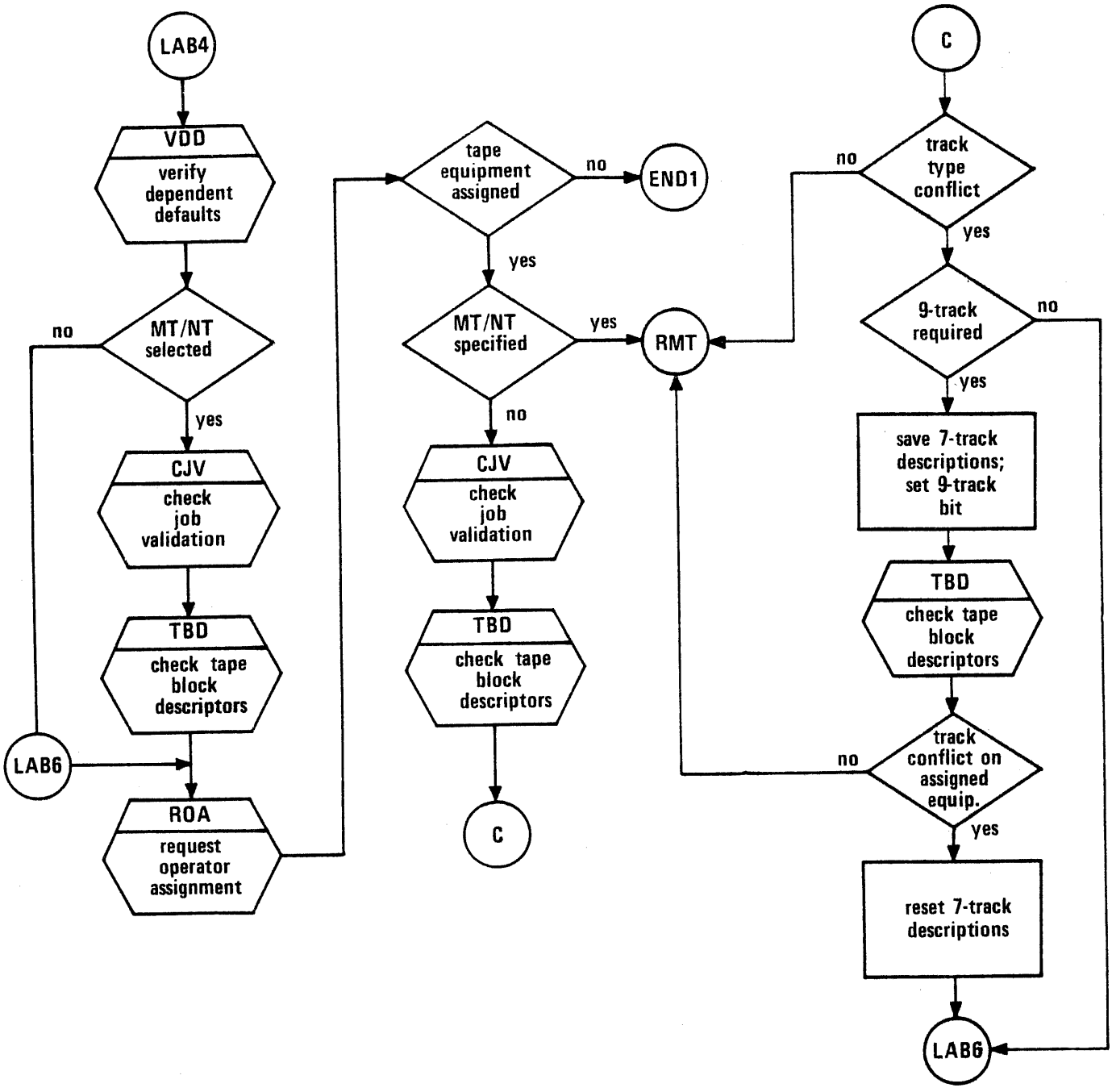


Figure 12-6. ASSIGN/LABEL/REQUEST - Assignment Control Statements (Continued)

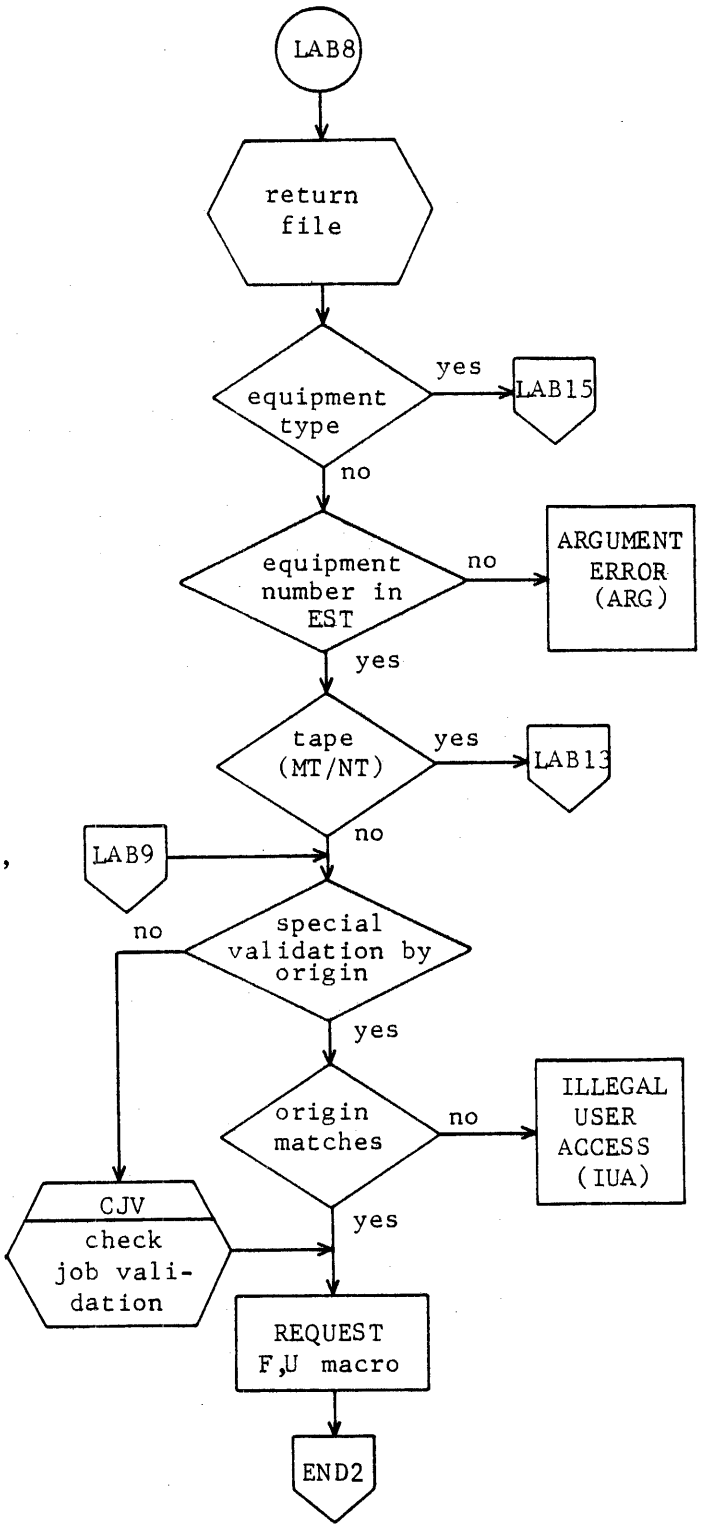


Figure 12-6. ASSIGN/LABEL/REQUEST - Assignment Control Statements (Continued)

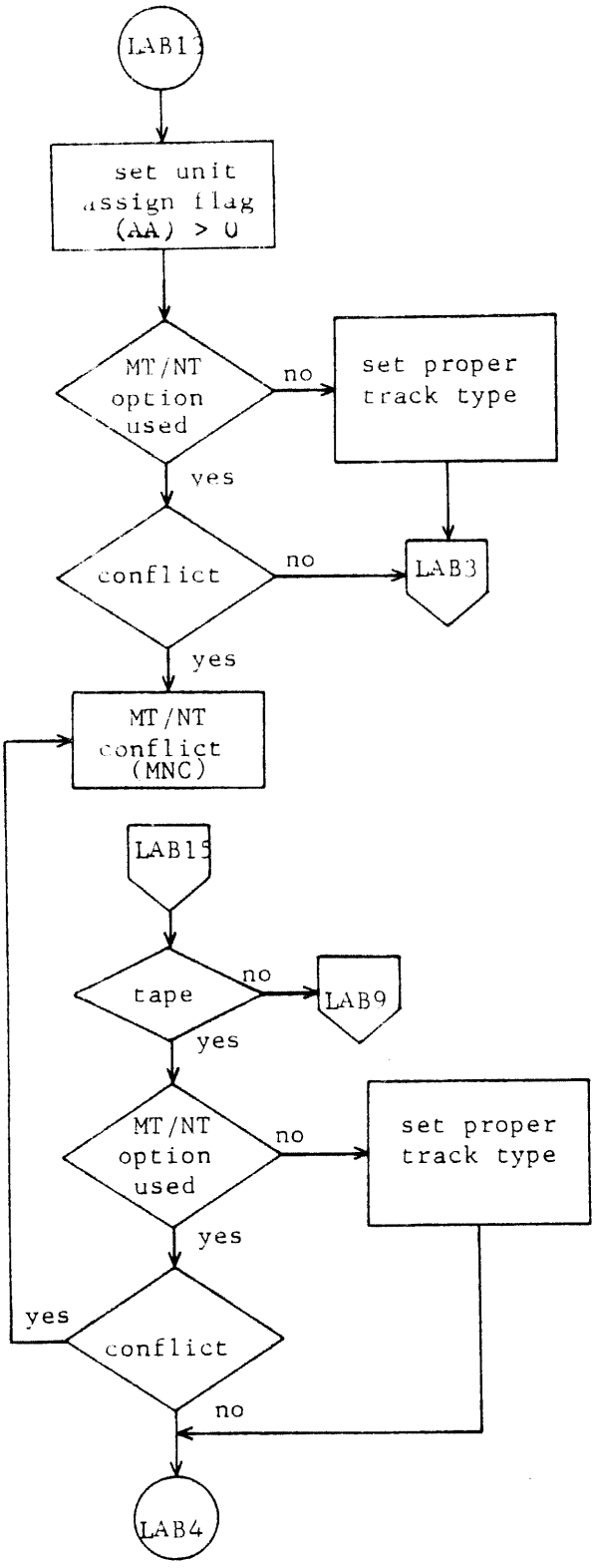


Figure 12-6. ASSIGN/LABEL/REQUEST - Assignment Control Statements (Continued)

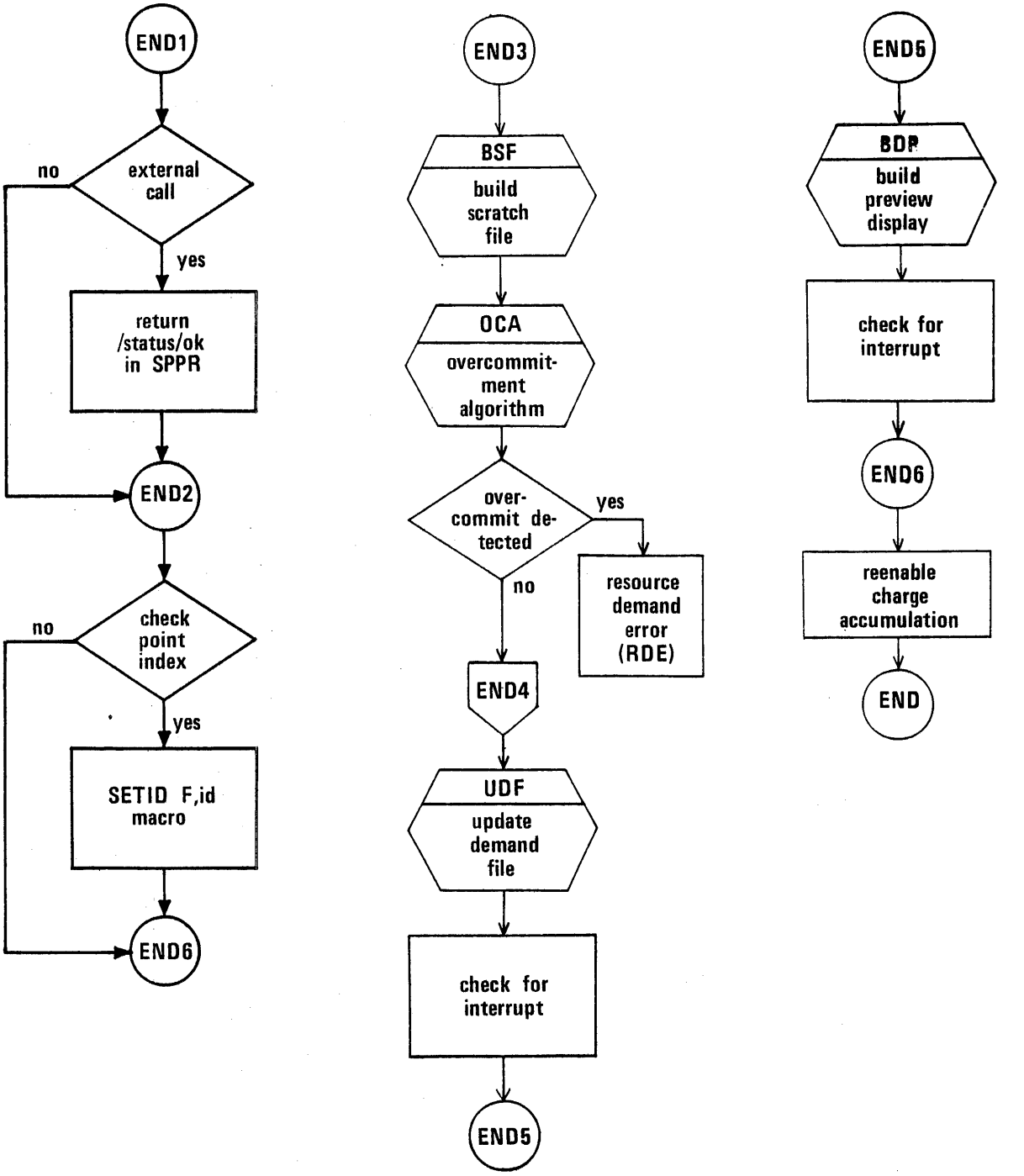


Figure 12-6. ASSIGN/LABEL/REQUEST - Assignment Control Statements (Continued)

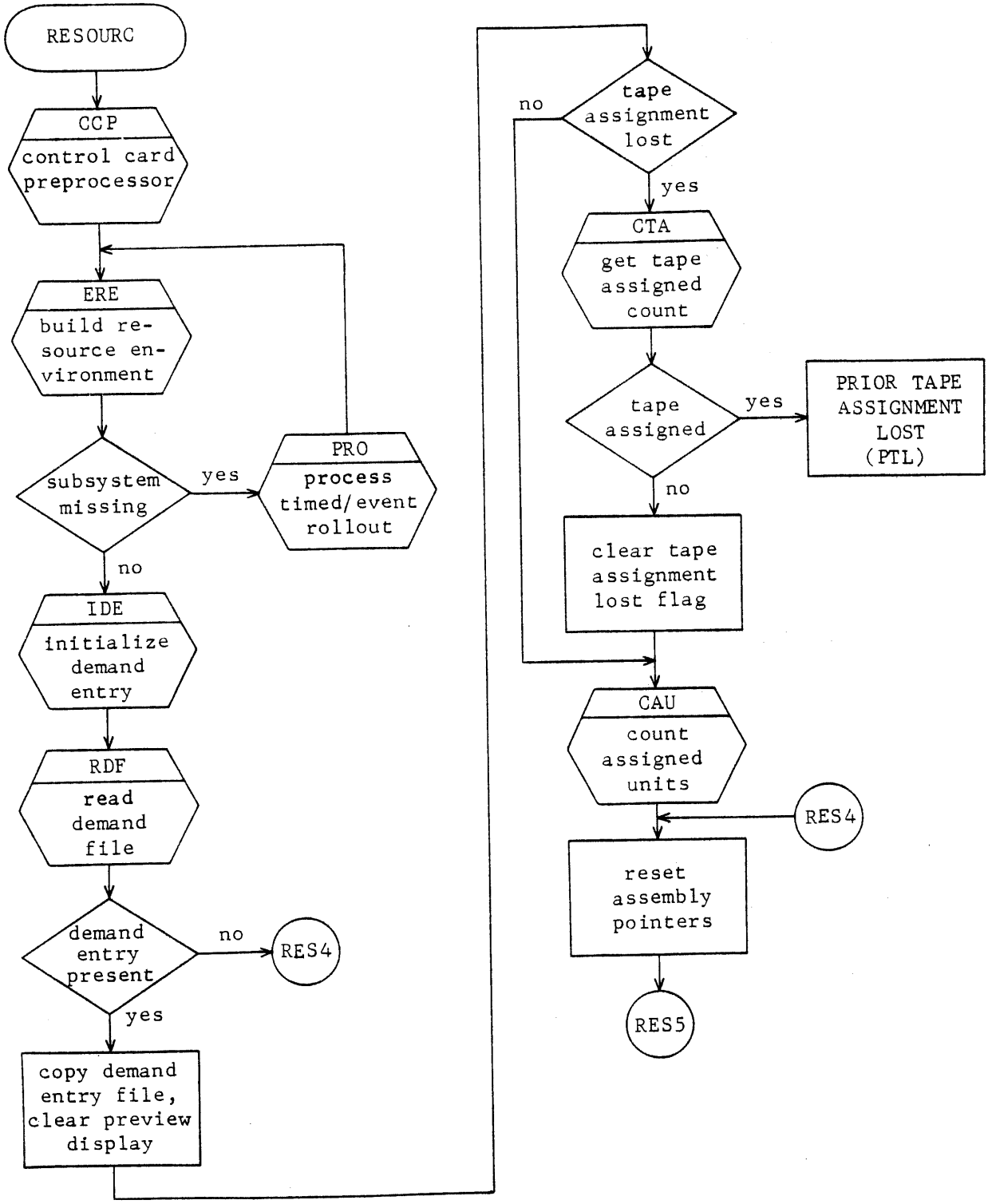


Figure 12-7. RESOURC Control Statement

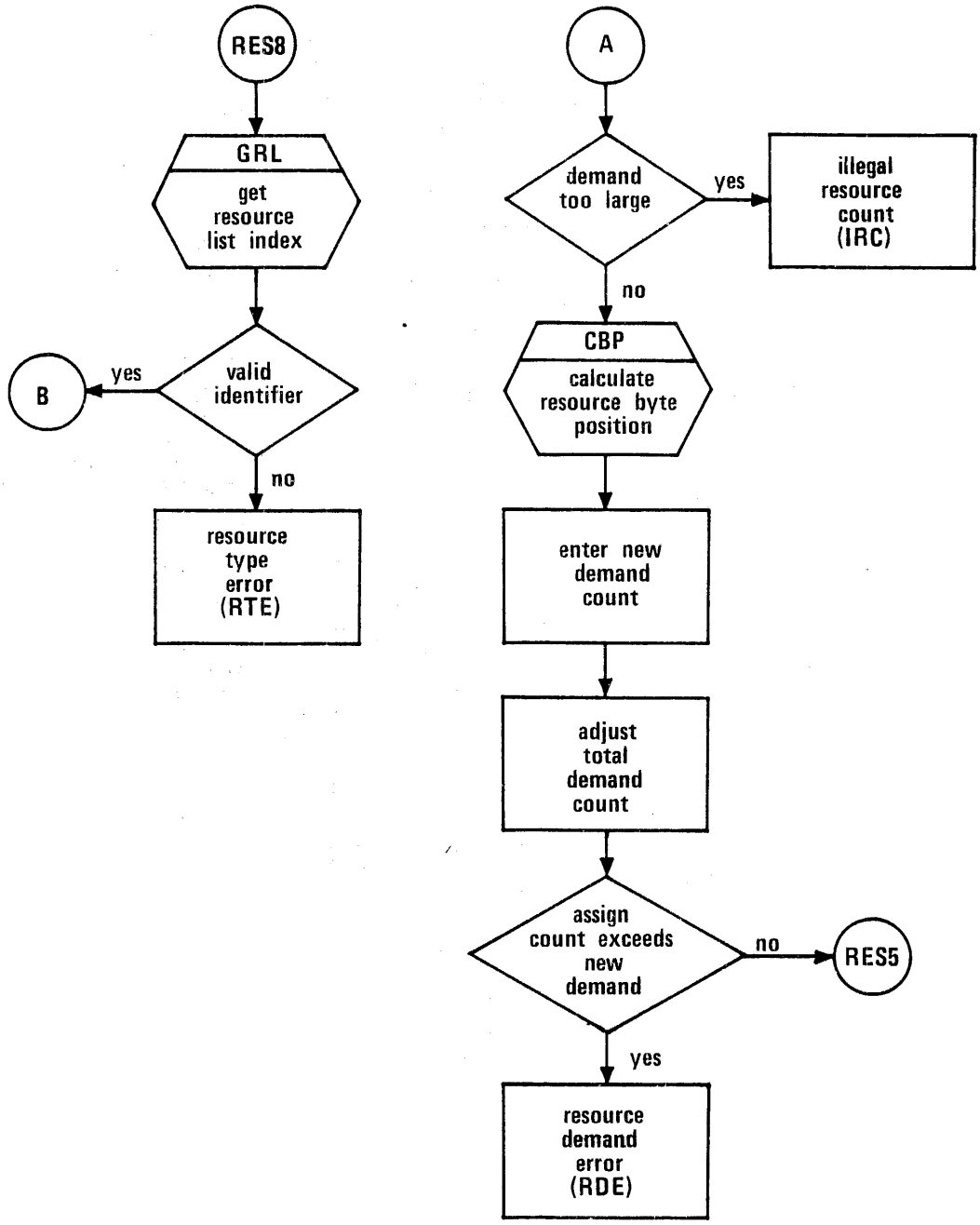


Figure 12-7. RESOURC Control Statement (Continued)

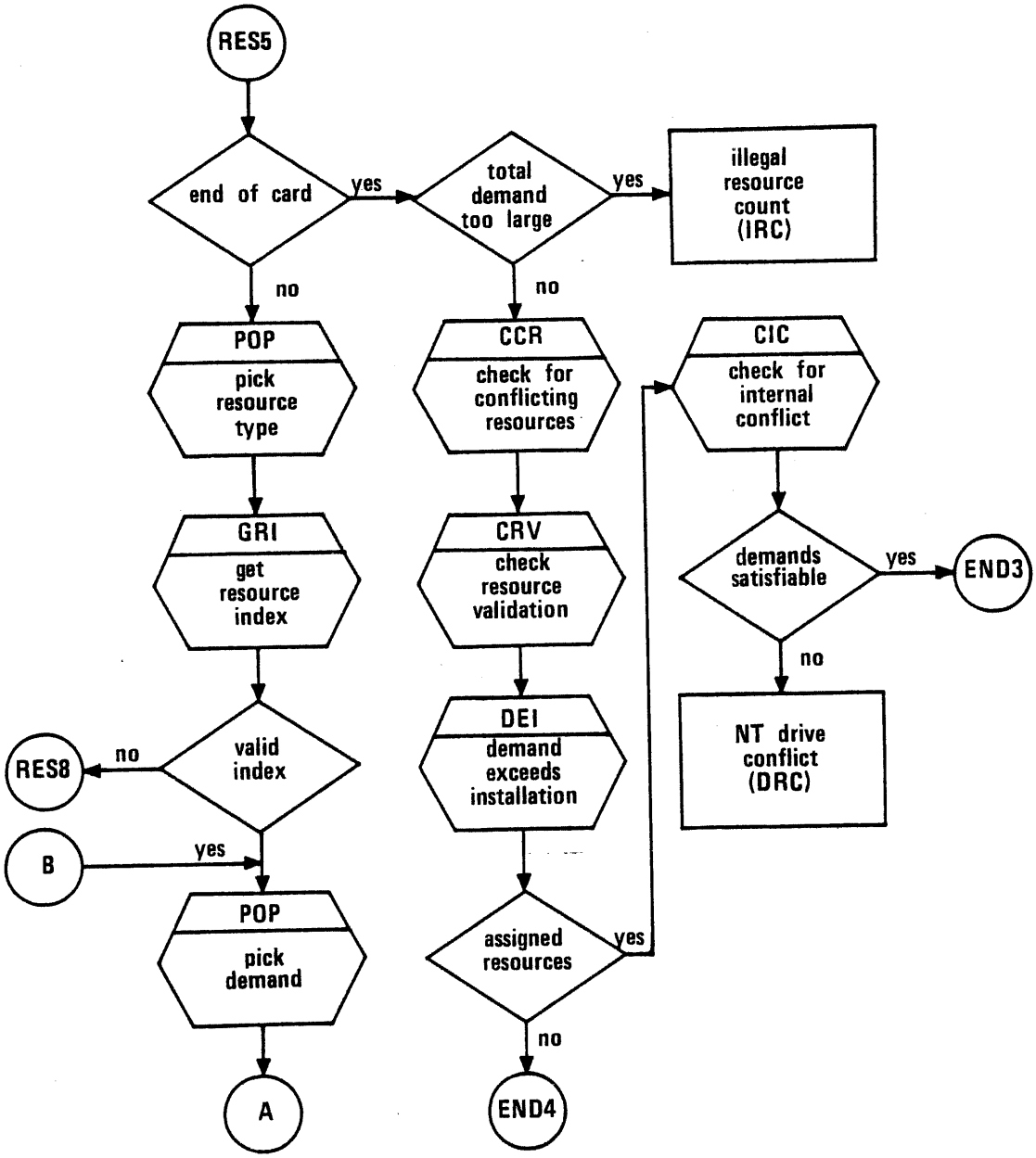


Figure 12-7. RESOURC Control Statement (Continued)

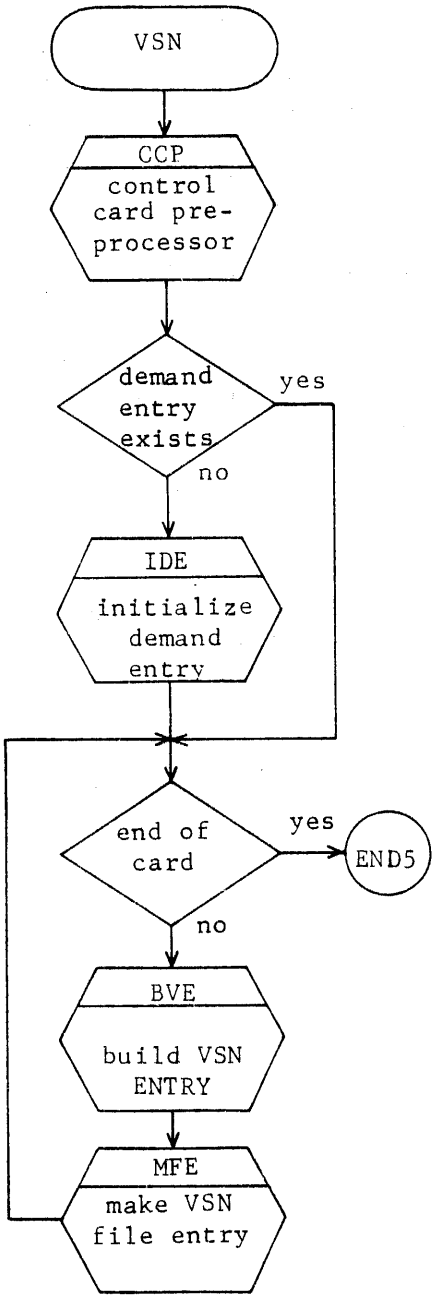


Figure 12-8. VSN Control Statement

EXTERNAL CALLS

The assignment of removable packs and magnetic tapes is also available to the user at the macro and RA+1 function level. Tape requests use LFM functions and removable pack requests use PFM functions.

If the user issues an LFM or PFM RA+1 call either directly or through a macro to assign a resource unit to the job, the PPU processor (LFM, PFM) initiates a call to RESEX at that job's control point. To avoid destroying the user's field length when RESEX is invoked from the PPU, the DMP= special entry point is defined in RESEX. The calling of RESEX is done via the SPCW word in the job's control point area.

The SPCW calling mechanism places, beginning at address SPPR of the field length, a copy of the SPCW word and a 20B word block of data (usually the user FET); RESEX is loaded and begins execution at the entry point specified in the SPCW call. Upon completion, RESEX sets a status in SPPR which is returned to the calling PP when RESEX relinquishes control.

In addition to LFM and PFM, RESEX processes the REQ RA+1 call in order to maintain compatibility with NOS/BE. The call to RESEX through SPCW is initiated by PP routine SFP.

External call refers to the processing of LFM, PFM, and REQ calls by RESEX. This is opposed to an internal call in which RESEX calls the tape assignment routines directly after cracking the tape assignment control statement.

Figure 12-9 flowcharts the LFM external call and figure 12-10 is a flowchart of the REQ external call.

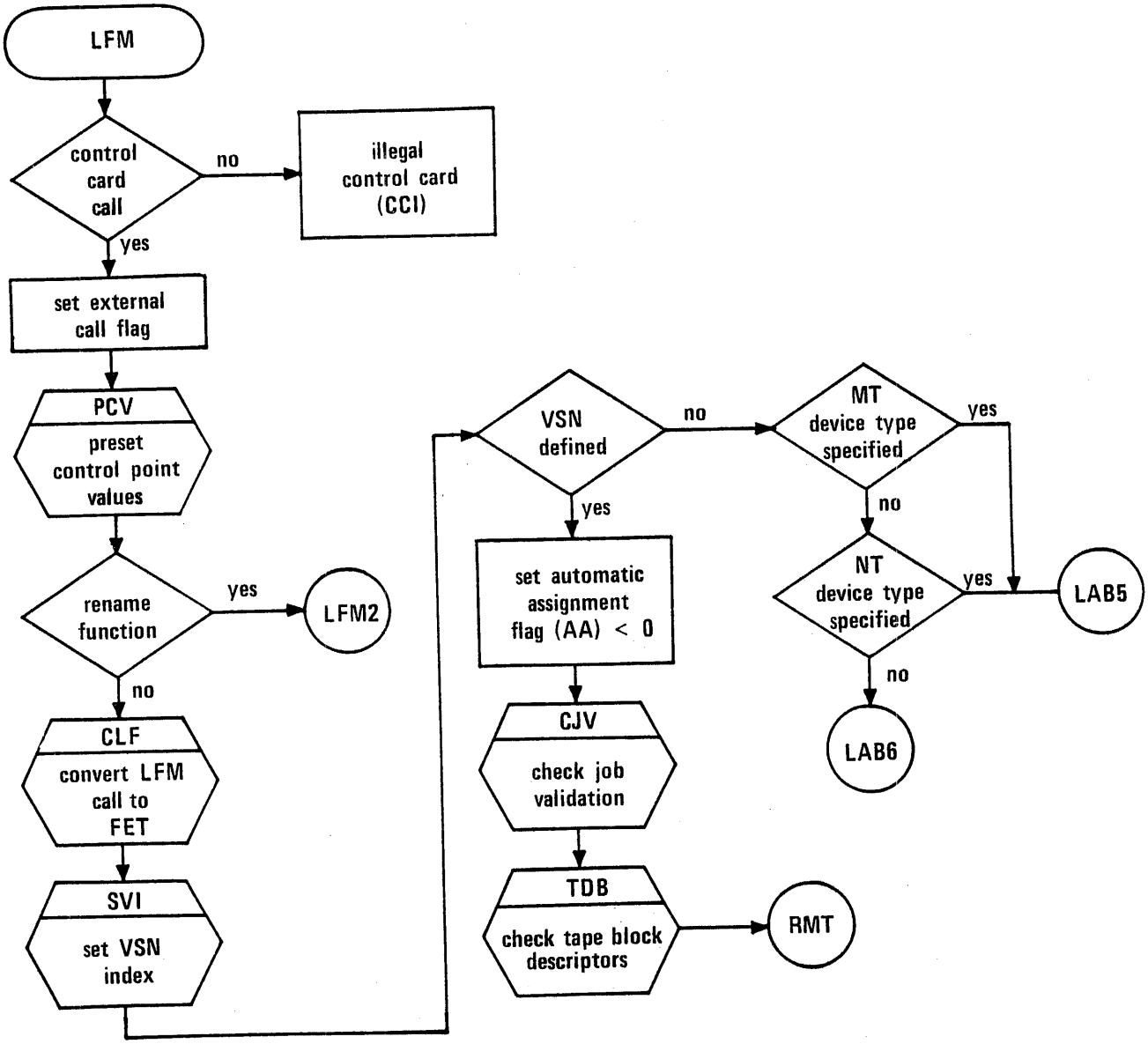


Figure 12-9. LFM External Call Processor

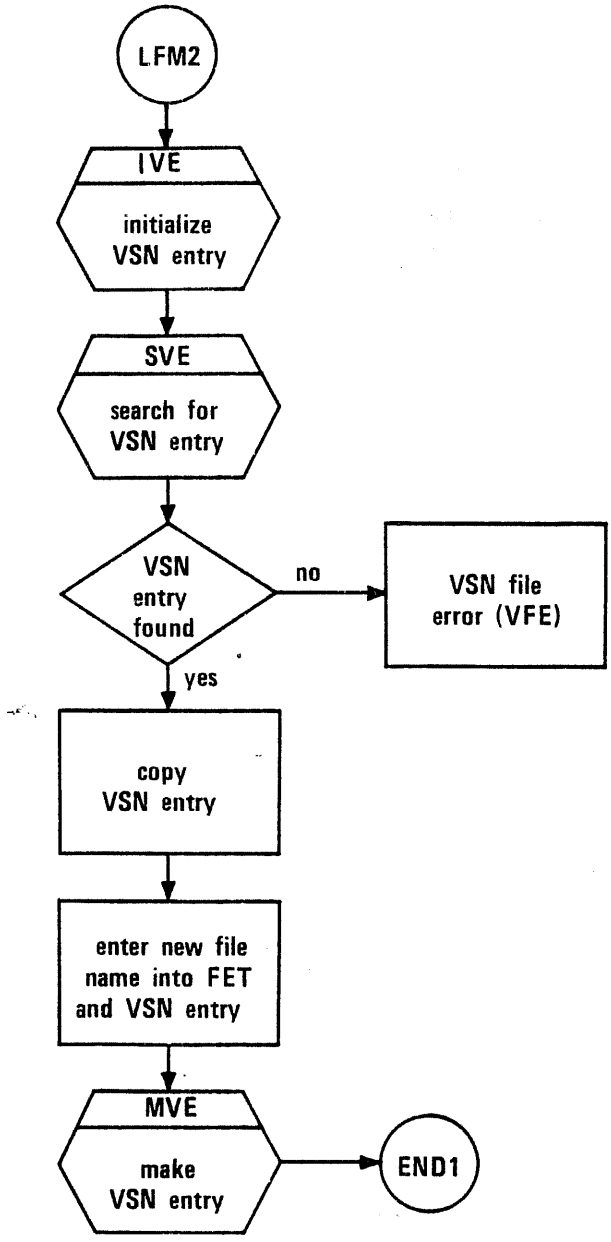


Figure 12-9. LFM External Call Processor (Continued)

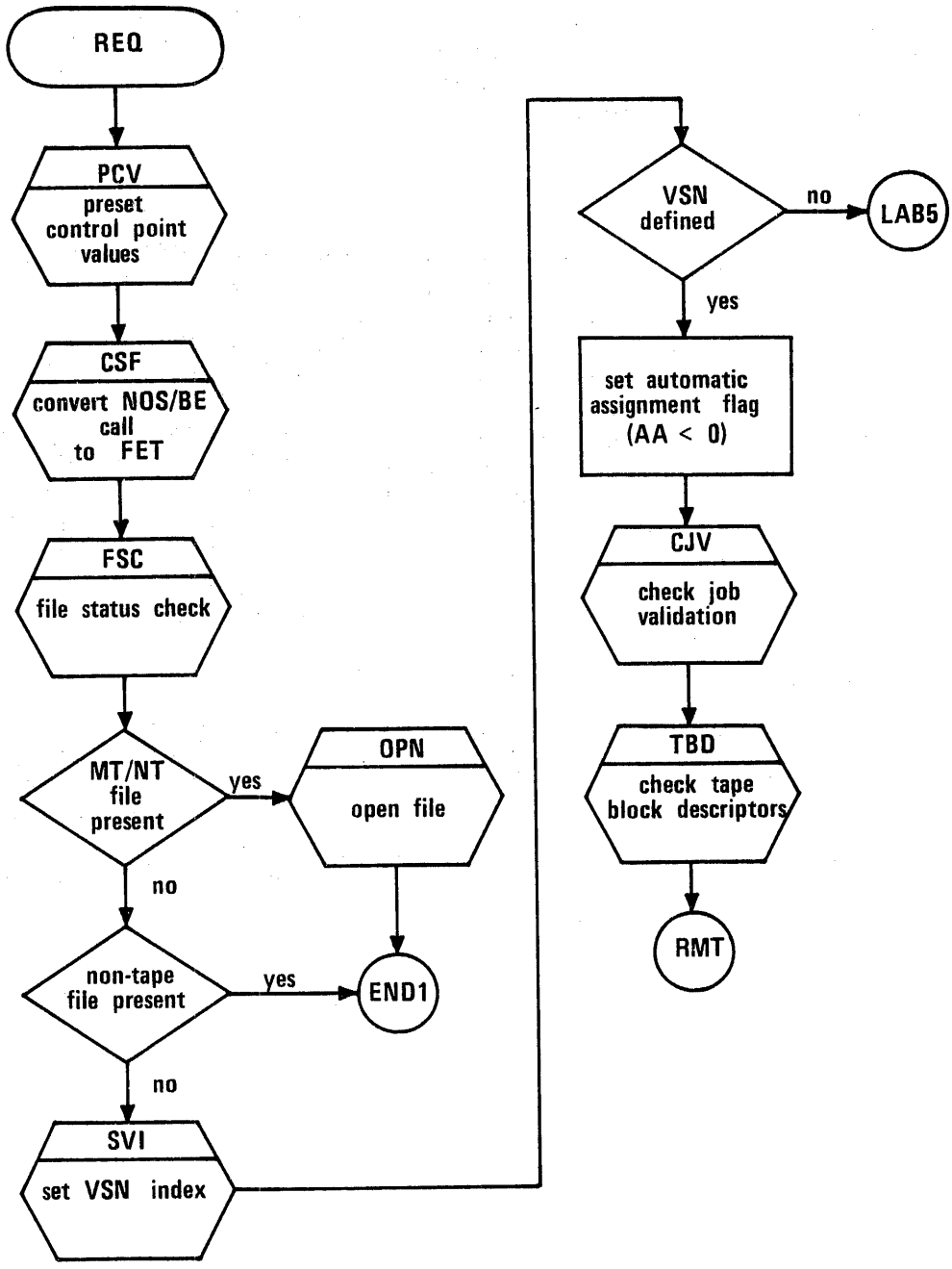


Figure 12-10. REQ External Call Processor

RESOURCE ASSIGNMENT

The assignment of resource units to a job revolves around two routines, RRP (request removable pack) and RMT (request magnetic tape). Each routine processes the request using data contained in a requesting FET. The FET used may be the actual user FET (LFM calls for tapes and all pack requests) or one built by RESEX as the result of processing ASSIGN, LABEL, or REQUEST control statements.

REMOVABLE PACKS

All removable pack requests are processed by RESEX upon entry at the PFM entry point as an external call. The user's FET (as copied to SPPR) is used to provide the pack name and resource type being requested. From these values or their defaults, input parameters (request blocks) are built for use by the overcommitment algorithm. PFM then transfers control to subroutine RRP. RRP issues the call to the COM subroutine. If a missing subsystem status is returned, RESEX rolls out with a missing subsystem event and requeues the request. If overcommitment occurs and the user has selected to do error processing (ep bit set in the FET), then RESEX rolls out with an overcommitment event and requeues the request. If ep is not set, RESEX aborts the job with the following diagnostic.

REMOVABLE PACKS OVERCOMMITMENT

If the pack is not mounted and the ep bit is set, an event is formatted from the pack name and RESEX rolls out using this event and requeues the request. If ep is not set, control is returned to PFM. If everything is correct, then the demand file is updated to reflect this assignment, the preview display built, and a successful status is returned to PFM. The flowchart of PFM/RRP is contained in figure 12-11.

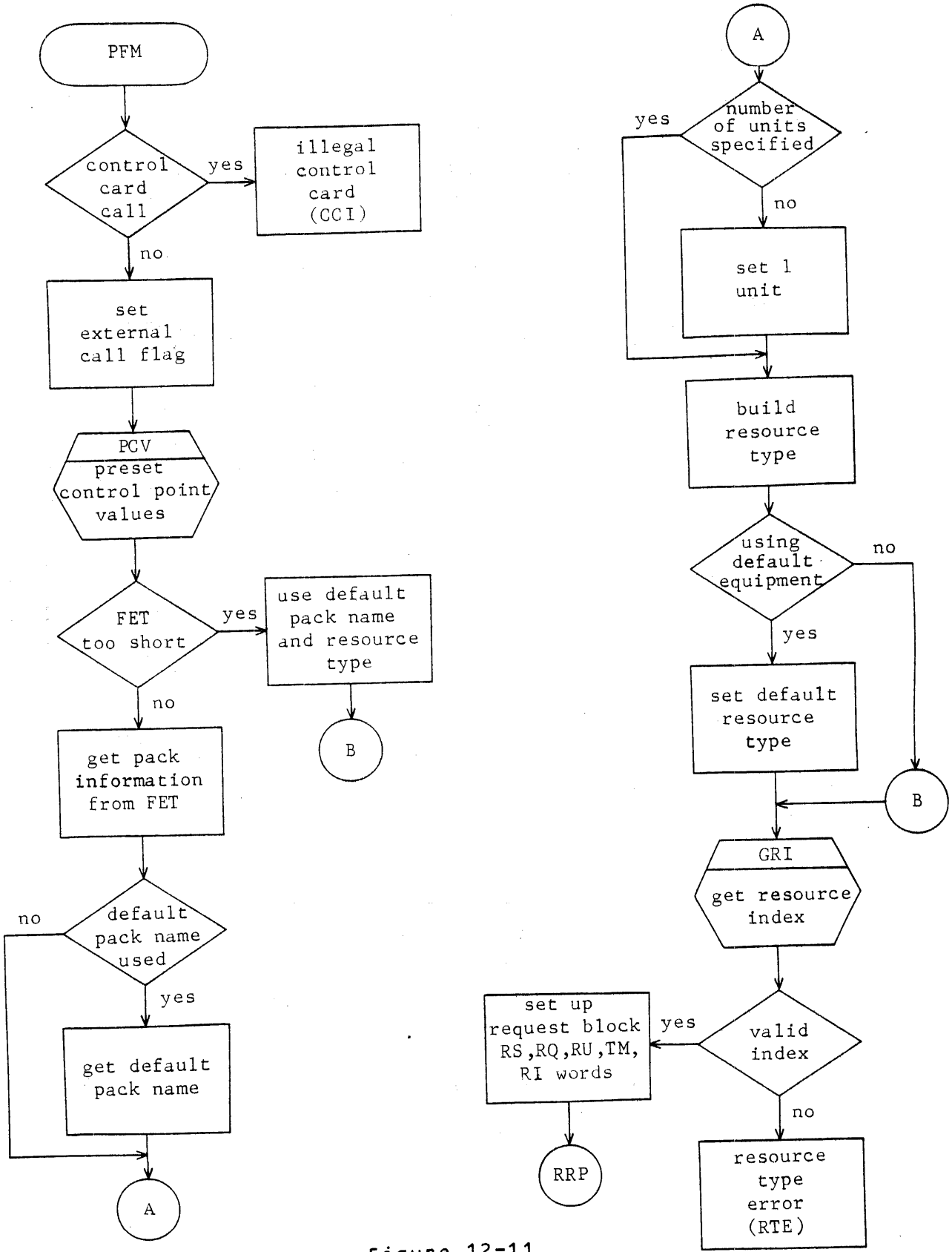


Figure 12-11.
PFM - PFM External Call Processor and
RRP - Request Removable Pack

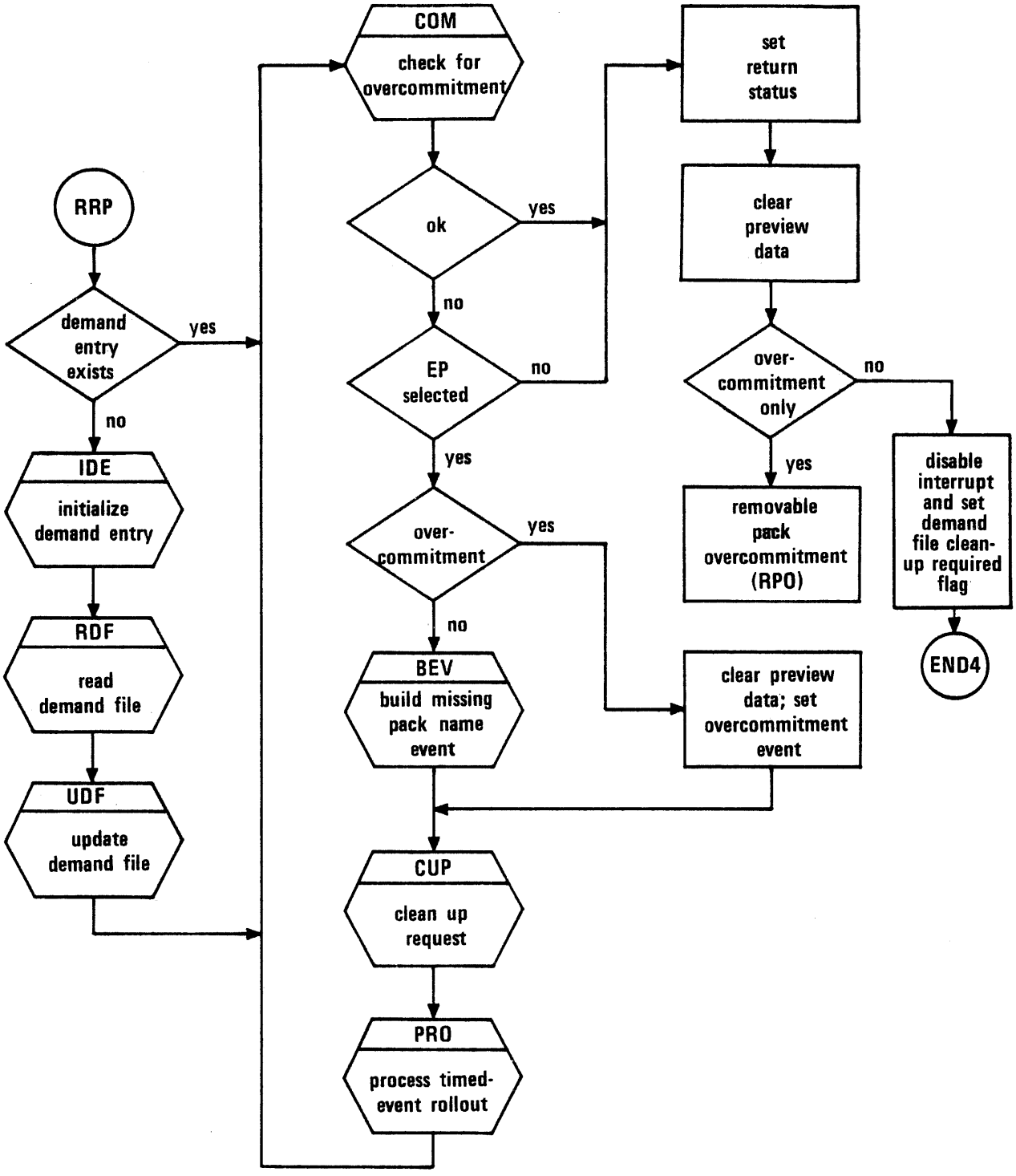


Figure 12-11.
PFM - PFM External Call Processor and
RRP - Request Removable Pack
(Continued)

RMT, flowcharted in figure 12-12, is called to request a magnetic tape from MAGNET. Before exiting to RMT, subroutine TBD is called by the assignment control statement processor and from LFM and REQ external functions. TBD builds the tape block definition (TB) by mapping portions of the tape descriptors (FET+8) into values for use in the RESEX/MAGNET call block. TBD computes the word count, overflow, unused bit count, noise size, and fill according to the requested format, frame size, and noise size. TBD also establishes density and conversion mode using installation supplied values (TDEN, TCVM, and TDTR) as necessary. Finally, TBD ensures that combinations of parameters do not yield conflicting results, such as specifying both ring-in and ring-out processing options. For 9-track tape requests, TBD sets the original 9-track density display (HD, PE, or GE) into request block word DD.

RMT builds the request block (refer to figure 12-13). The request block (words RQ, RS, RU RI, DD, and OI) forms a common set of input parameters to the overcommitment algorithm for both tapes and packs. RMT guarantees that the requestor has a demand file entry and a VSN file entry for the desired tape file. All tape requests must have these two entries before attempting to satisfy a request.

RMT then calls subroutine COM. This subroutine identifies the unit that satisfies the request and performs deadlock prevention. COM returns three results: missing VSN, overcommitment, and assignment permitted. If the VSN is missing, RMT formats an event using the VSN and rolls out using this event. If overcommitment occurs, the preview data is cleared and RMT performs a rollout with the overcommitment event.

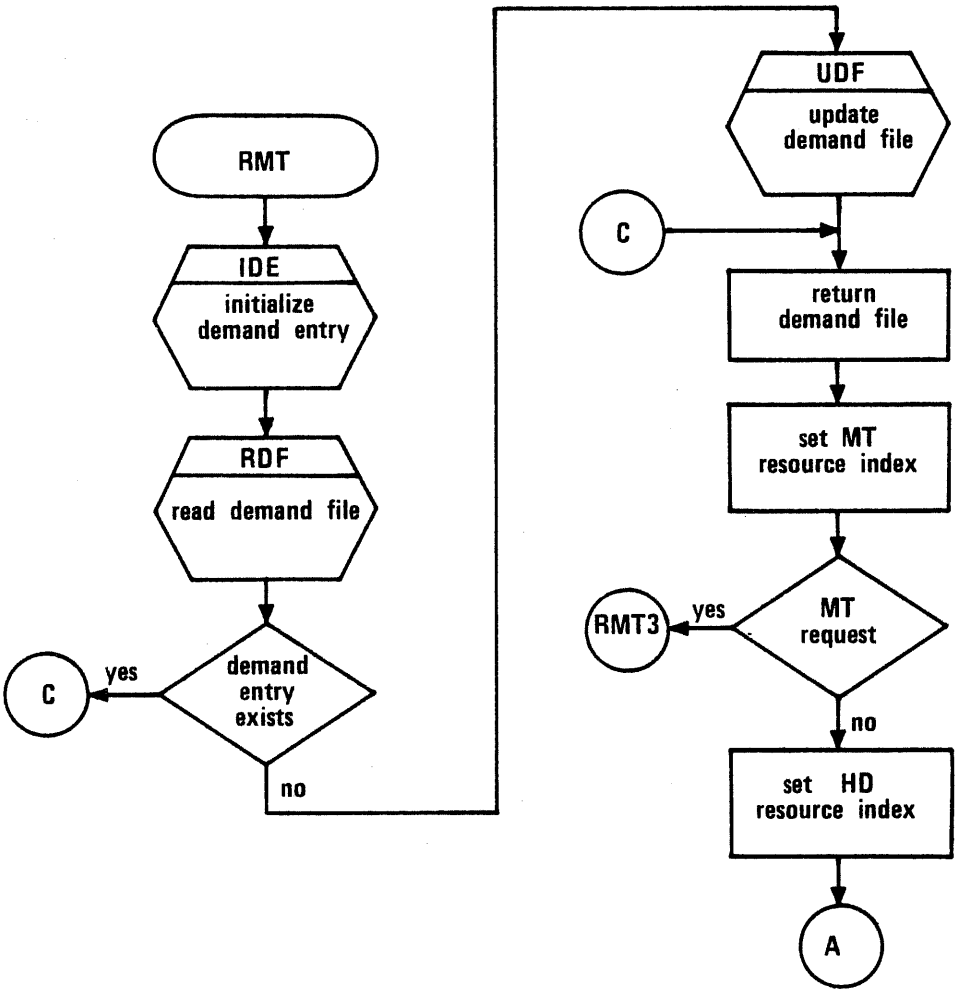


Figure 12-12. RMT - Request Magnetic Tape

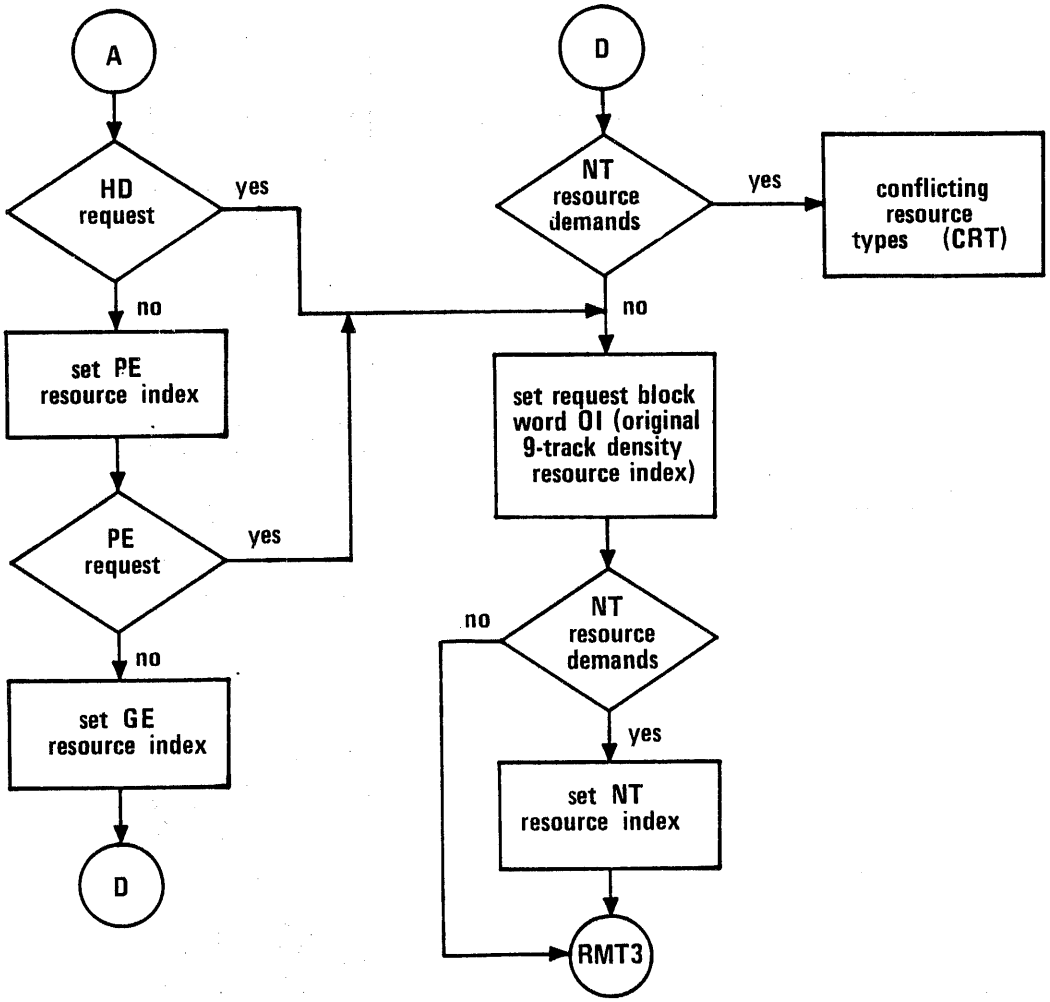


Figure 12-12. RMT - Request Magnetic Tape (Continued)

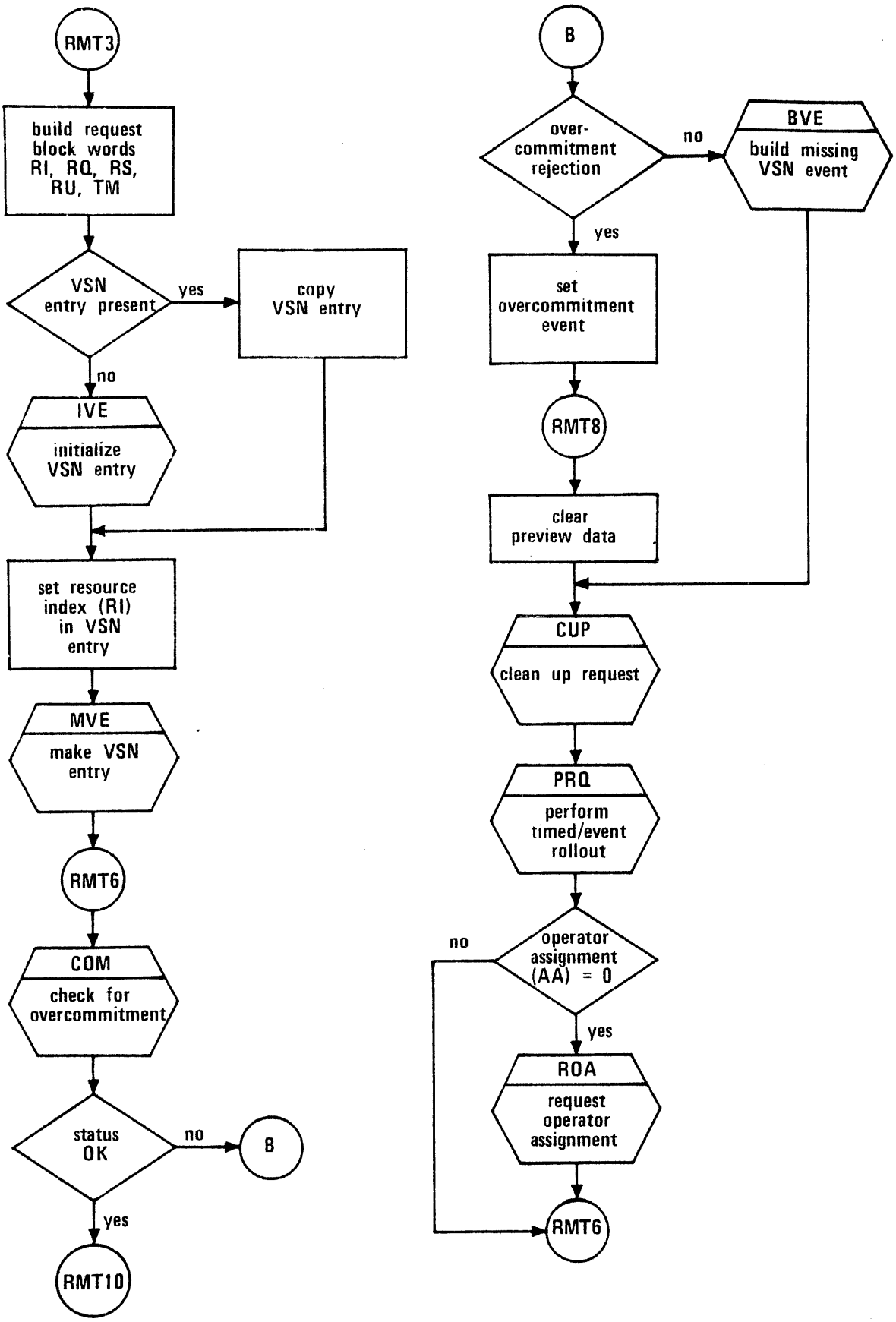


Figure 12-12. RMT - Request Magnetic Tape (Continued)

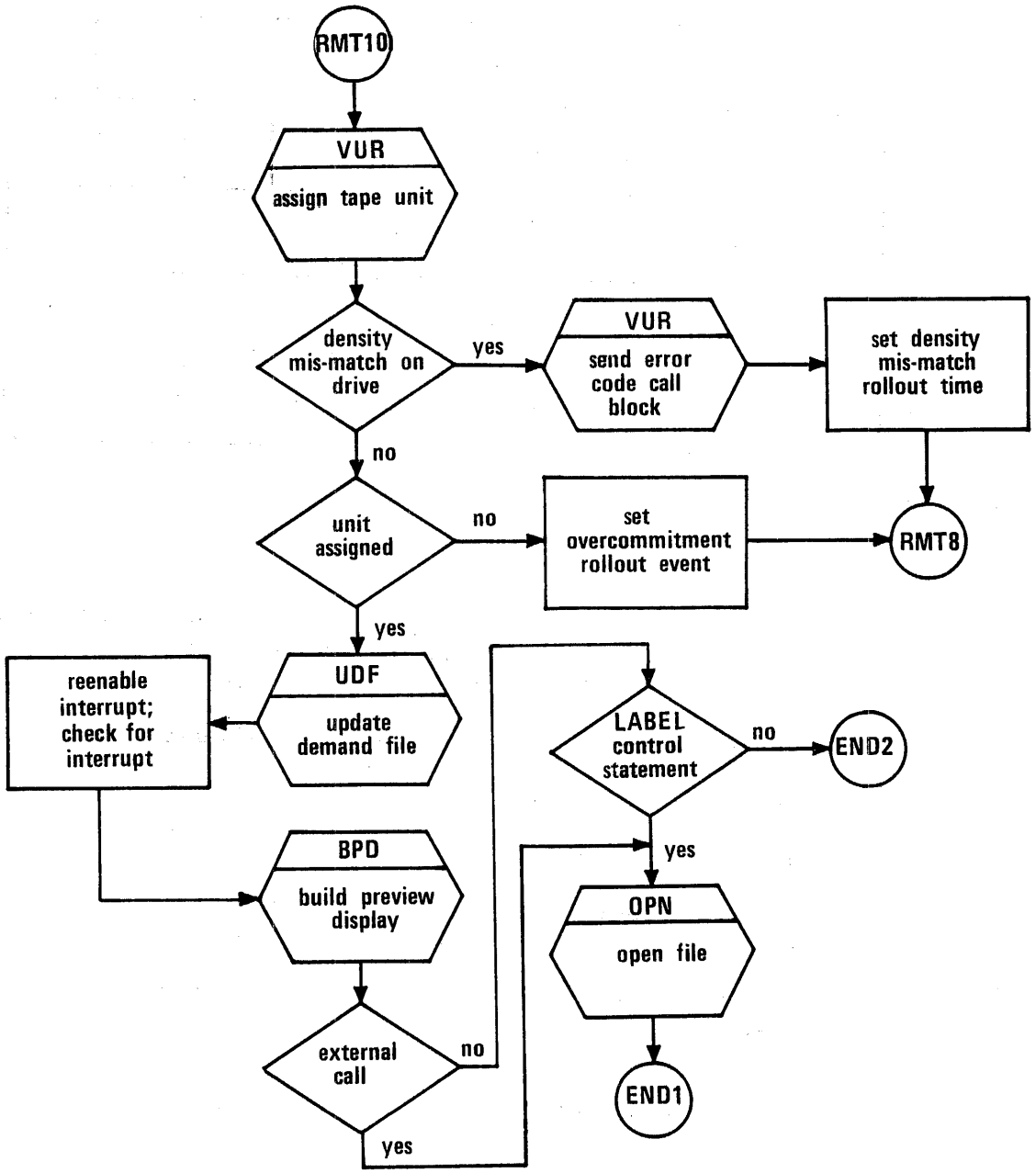
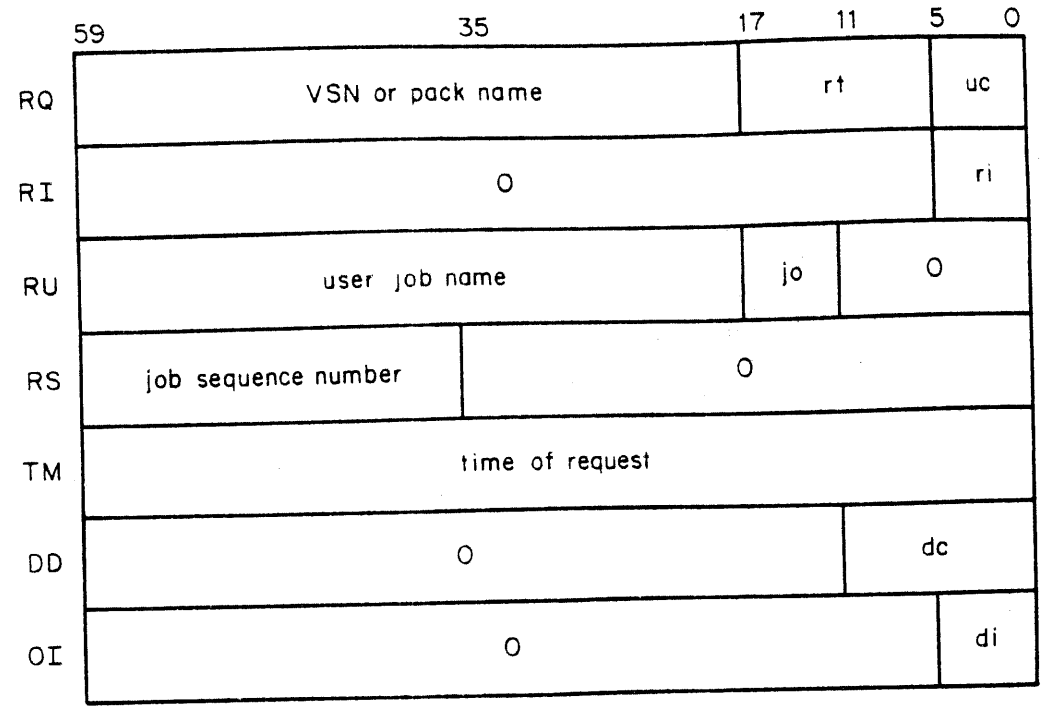


Figure 12-12. RMT - Request Magnetic Tape (Continued)



rt — Resource type (left-justified in the 12-bit field). Values for RT are MT, NT, HD, PE, GE, DI, DJ, etc., as in the demand file entry

uc — Unit count (1 to 8) for packs; always 1 for tapes

ri — Index into demand file entry

jo — Job origin type

dc — Display code for density (HD, PE, or GE) of 9-track tape request

di — Original density resource index for 9-track tape request

Figure 12-13. Request Block (RQ)

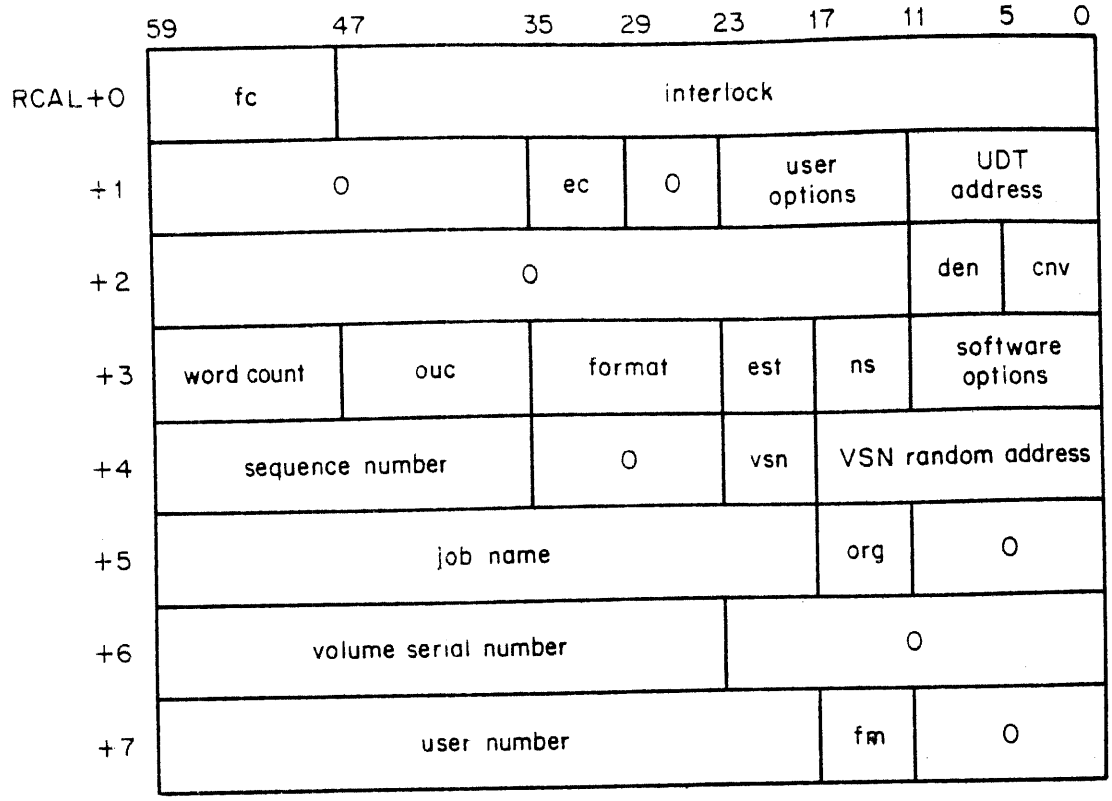
If COM returns the status indicating the assignment is allowed, subroutine VUR is called. VUR reads the UDT from MAGNET to verify that the VSN has not changed and the unit has not been assigned to another job since the environment was established. If this is not the case, RMT processes this situation as if it were an overcommitment case.

Next, VUR checks for wrong density if the write ring is out of the tape or the user requested the write ring out. If an 800 bpi tape is mounted on a 1600/6250 bpi drive or a 6250 bpi tape is mounted on an 800/1600 bpi drive, VUR returns to RMT with density mismatch status. RMT then sends an error code block to MAGNET to advise the operator of this condition via the E,P display and processes the assignment rejection as an overcommitment case. If VUR detects any other combination of wrong density on a 9-track drive, a warning message (MIS-MATCHING DENSITY) is issued.

Next, the conversion mode from the UDT is compared with that requested by the user; a warning (MIS-MATCHING CONVERSION) is issued if these conversion modes are not compatible. Then the file accessibility and volume accessibility are validated to guarantee system security is maintained by not permitting unauthorized operations to be performed on the tape. The RESEX/MAGNET call block (figure 12-14) is then built, the equipment interlocked through LFM, and the call block sent to MAGNET to activate unit assignment. The UDT is reread from MAGNET to verify that the assignment is successful and then RESEX completes the FNT/FST entry for the tape file through LFM. The message

EQuu ASSIGNED TO lfn, VSN= vsn.

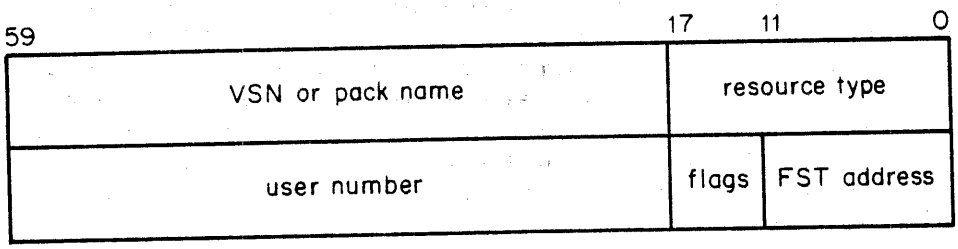
is issued to the user and system dayfiles indicate the assignment of the unit to the job.



fc	Function code (0=assign unit, 1=set error code in UVSN word of UDT)
ec	Error code for function 1
den	Density
cnv	Conversion
ouc	Overflow unused characters
est	EST ordinal
ns	Noise
vsn	VSN index
org	Job origin
fm	Family ordinal

Figure 12-14. RESEX/MAGNET Call Block

RMT then updates the job's demand file entry, builds the preview display buffer, and performs any file opening required. RMT exits and control is returned to the routine that called RESEX. The format of the preview display buffer is as follows



flags Each bit defined as follows:

Bit	Description
14	Unlabeled
13	Ring out
12	Ring in

COM SUBROUTINE

The subroutine COM controls the identification of the tape or pack requested and the processing of the overcommitment algorithm. COM begins by calling subroutine BRE to build the working resource environment tables (RET, EVSB) from which the desired resource unit can be identified and the algorithm executed. COM performs a timed/event rollout with missing subsystem event if BRE detected that MAGNET is not active. If demand file busy rollout occurred when BRE attached the demand file, COM rerequests operator assignment of equipmnt if no VSN was supplied (contents of AA equals 0).

Subroutine CRQ is called to identify the desired VSN to be used during tape assignment and to set up the demand file entry and preview data for further processing.

Subroutine DEI is then called to determine if enough resource units are available in the environment to satisfy the job's demands.

Then subroutine BSF is called to build a scratch file with the demand file entries for all jobs with assigned resources except the requestor. Subroutine OCA later adds the requestor's updated demand file entry on the scratch file before executing the overcommitment algorithm. Subroutines CRQ and BSF call subroutine CAU (count assigned units) to adjust the environment to reflect those equipments that are currently assigned.

Next, subroutine CFU is called to assign the resource unit to the job. This allocation involves the advancing of the assigned count for the resource type being assigned, updating RESEX working tables, and building the share table entry (if a removable pack is being assigned). If the desired tape or pack is not found, pseudoentries are made in order that the overcommitment algorithm may still be executed. If a pseudo entry must be made or a SCRATCH tape request is being satisfied, CFU guarantees that the selected equipment will not cause internal conflict for the job.

Subroutine CIC is called to check for an internal conflict. The requestor job is in a state of deadlocking itself if the requestor's remaining demands cannot be satisfied by the environment remaining after the requestor's currently assigned resources and the unit selected by CFU are eliminated. This can occur when the current request is for a 1600 bpi 9-track tape (PE resource) that is mounted on the wrong drive type (800/1600 or 1600/6250) such that the remaining 800 bpi (HD resource) or 6250 bpi (GE resource) 9-track tape requirements cannot be satisfied. In this case, COM rejects the assignment and sends an error code call block to MAGNET to inform the operator of the conflict via the E,P display, so that the tape can be mounted on the correct drive type.

Subroutine CRC is called to determine whether the requesting routine completes with the assignment of this resource unit. If so, the overcommitment algorithm is not entered as this job cannot cause a deadlock to occur.

The overcommitment algorithm is now executed by calling subroutine OCA. COM returns a status indicating overcommitment (/STATUS/OV), missing VSN or packname (/STATUS/MV), or assignment permitted (/STATUS/OK) to the calling routine.

COM is flowcharted in figure 12-15.

Subroutine CRQ exits to COM6 if rerequest operator assignment of equipment is necessary; to COM7 if rerequest operator selection for duplicate VSN is necessary; or to COM10 if wait for ready on selected unit is necessary. Subroutine CFU also exits to COM7 if operator selection for duplicate VSN is necessary.

RESEX calls subroutine R0A to request the operator to make an equipment assignment via LFM function 26. R0A uses FET+1 to pass LFM the required device type (MT, NT, or if zero, any equipment), and for 9-track tape requests, and the display code for the required density (HD, PE, or GE) to be entered in the operator request message. If an operator choice for a duplicate VSN is necessary, R0A also passes the required VSN in FET+9 for display in the operator request message. LFM returns the device type of the assigned equipment, and for tape equipment assignments, the EST ordinal of the assigned unit in FET+1.

PREVIEW DISPLAY

The preview display (E,P) involves RESEX, MAGNET, and DSD. Information for formatting the preview display is put into a job's demand file entry if the desired tape or pack is not mounted, and the job can use the resource immediately (that is, a drive is available for the desired tape or pack and that

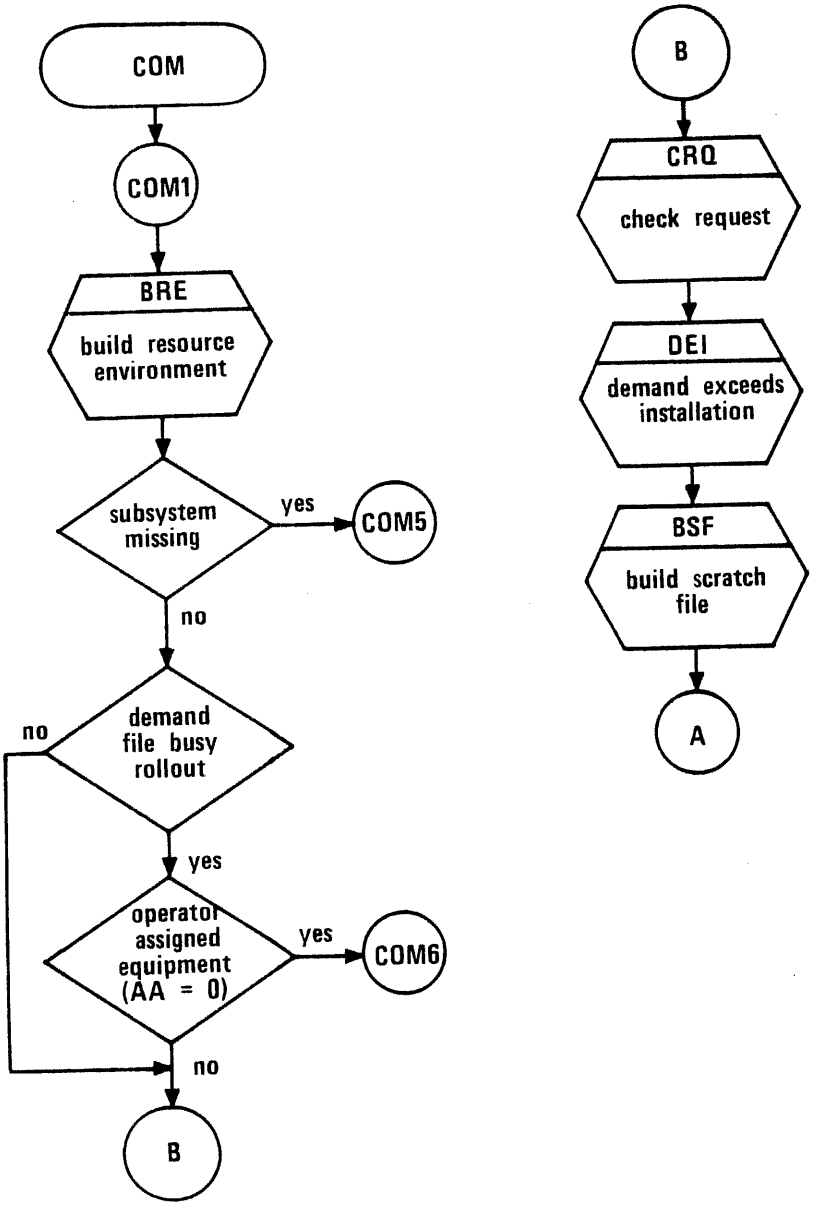


Figure 12-15. COM

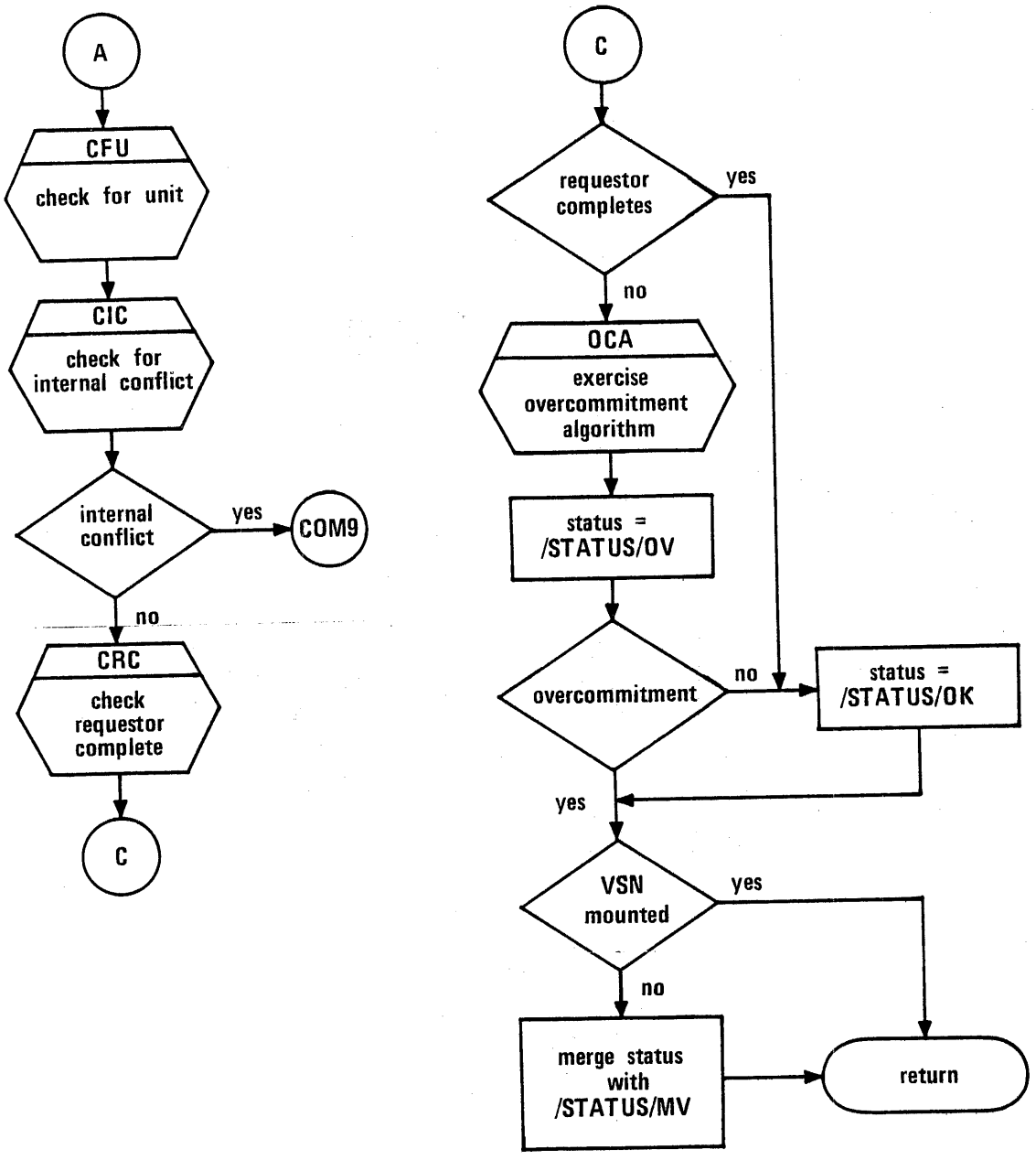


Figure 12-15. COM (Continued)

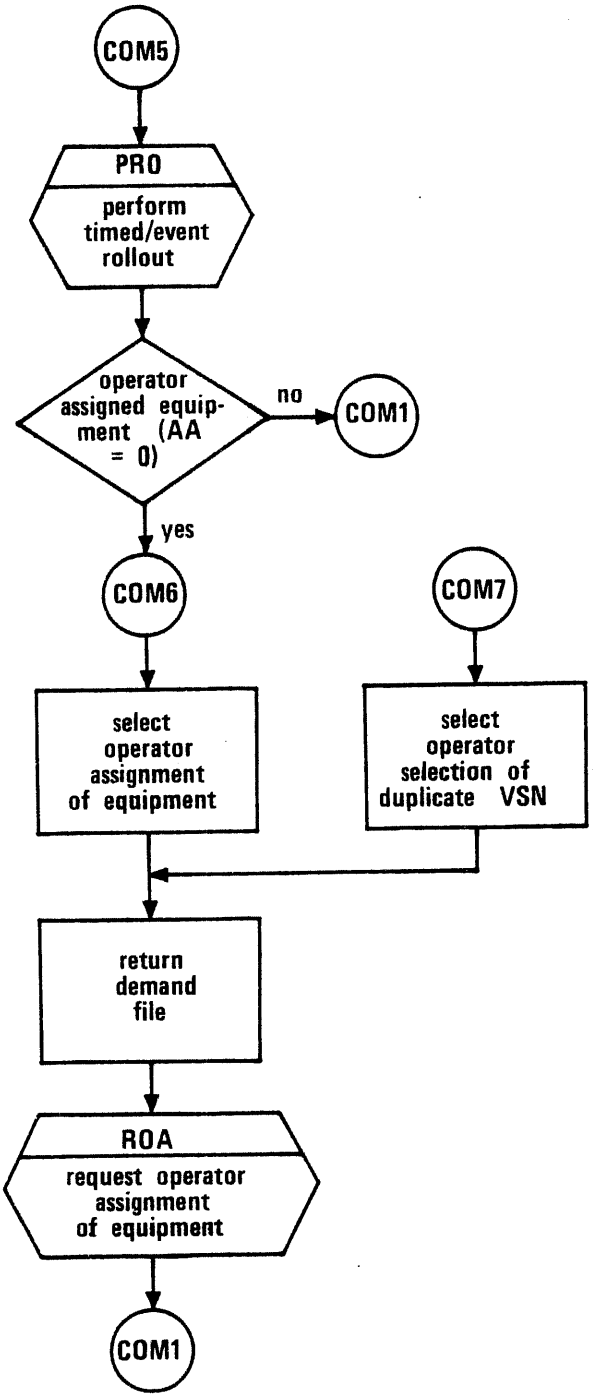


Figure 12-15. COM (Continued)

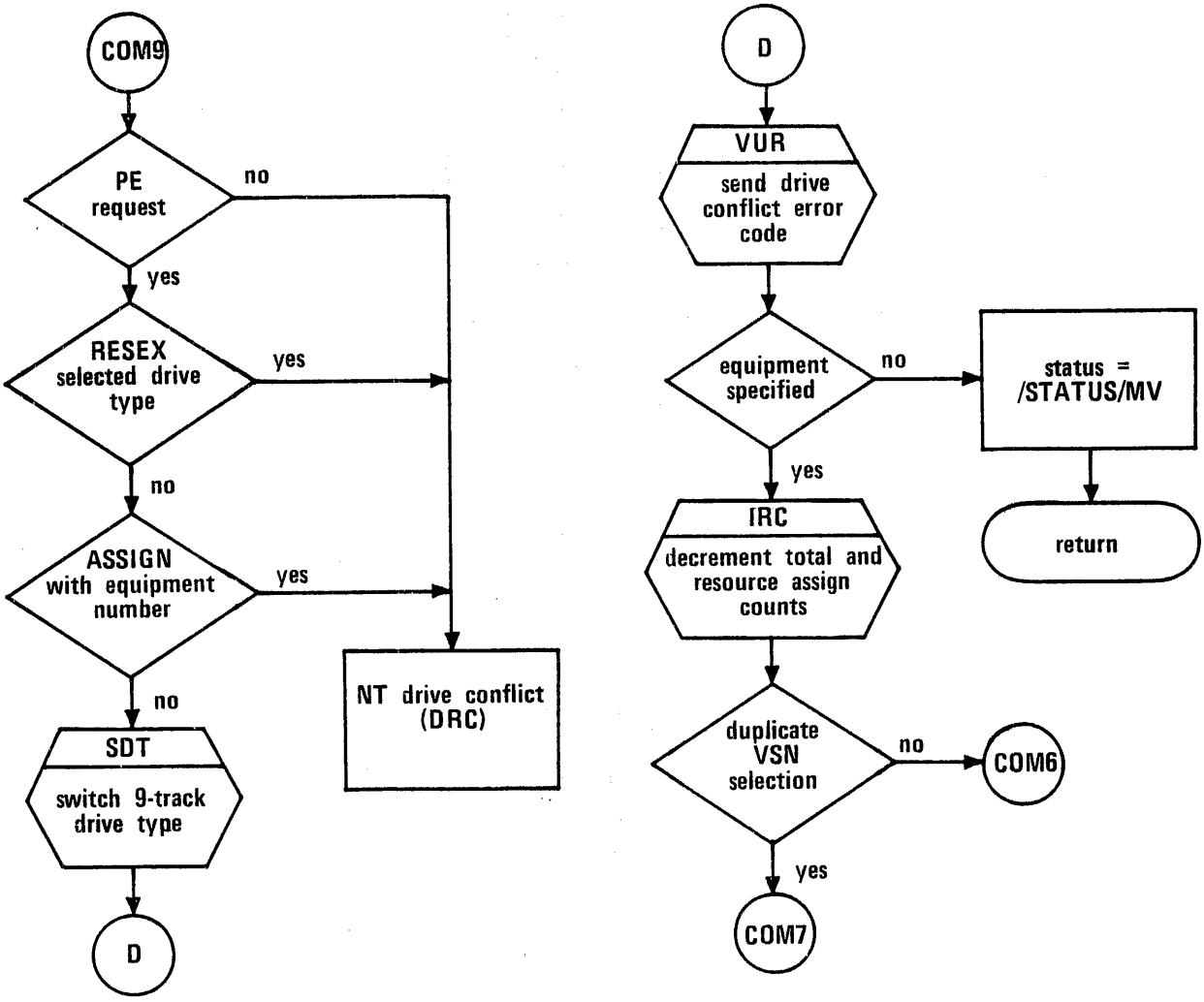


Figure 12-15. COM (Continued)

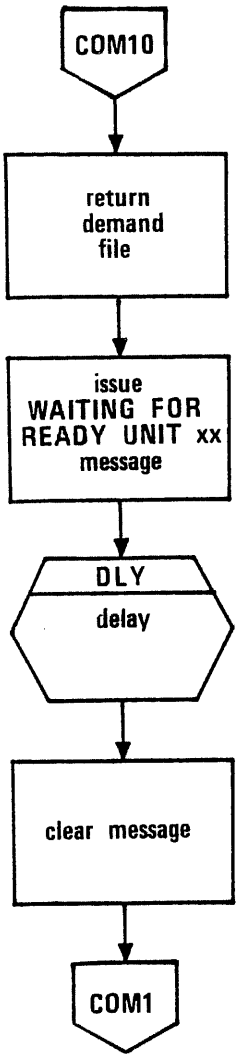


Figure 12-15. COM (Continued)

resource can be assigned to the job without causing a deadlock). RESEX reads the demand file in sequential order and if the job has preview data, it becomes a candidate for previewing. The preview buffer is built in an order based on resource usage and time of request. The order is determined using the highest values computed by the following formula.

H=((A+1)*T)/(D-A)

A Assigned count (total of all resources)

D Demand count (total of all resources)

T Time the request has been waiting

The preview buffer is transferred to MAGNET via the SIC monitor function and is limited to 63 words. This limits the number of jobs previewed by RESEX to 31 jobs. DSD formats the E,P display using the data in MAGNET's preview buffer and the UDTs. All jobs appearing in the preview buffer and those jobs which require a tape to be mounted due to reel swap, ring conflict, density mismatch, or internal conflict are displayed.

The preview buffer is updated by RESEX each time RESEX is called into execution. Thus, the preview buffer is refreshed only by an activation of RESEX. System activity, such as dropping a job with a request previewed from the rollout queue, does not cause that entry to be removed from the display until RESEX is activated by another job.

REPRIEVE PROCESSING

During initialization, RESEX requests extended reprieve processing to be in effect for errors set by a terminal user interrupt, an operator drop, kill, or rerun, or a DIS ERR command. RESEX contains several critical code sequences that must be completed to prevent dependent conditions from getting out of syncyhronization, such as the demand/VSN file entry and corresponding index or MAGNET and FST entry equipment assignments. If RESEX is interrupted while processing a critical code sequence, the reprieve processor (routine RPV) sets an interrupt flag and returns control to the prior executing code. After the sequence of critical code is completed, the interrupt flag is checked and the interrupt is processed. For noncritical code sequences, RPV immediately transfers control to the PIT routine to process the interrupt. PIT performs demand file and E,P display cleanup, if necessary, before issuing an appropriate RESEX ABORT message and aborting. If the job was rolled out with VSN or pack name missing, the preview data in the demand file entry should be cleared and the E,P display rebuilt. If the demand file attach fails, RESEX displays the following message at the user control point and retries the attach several times before giving up, in which case the preview data and E,P display are not cleaned up.

WAITING FOR DEMAND FILE.

In rare instances the demand file assign count must also be decremented to keep the user resource assign counts and file assignments in synchronization. In this case, RESEX continues to retry the demand file attach until it is successful.

ROUTINE ORF

PP program ORF updates the demand file entry upon job completion or upon the return of tape or pack resources. The return (or unloading) of a tape file clears that file's VSN file entry.

Routine ORF is called by REC (during deadstart recovery of MAGNET), ODF (return/unload or last disk file), 1CJ (clear demand file entry on job completion), 1DS (clear demand file entry on job RERUN or PURGE), 1TA (clear demand file entry on SALVARE cleanup), and PFM (decrement demand entry for GET on removable pack).

Routine ORF has three modes of operation: clearing VSN file entries, updating demand file entries, and clearing the demand file entry. The clearing of the VSN file entry automatically prompts the updating of the demand file for tapes. Routine ODF determines whether the direct-access file being returned is the user's last file on the removable pack and prompts ORF to update the share table entry for the removable pack and adjust the resource counts for the resource type involved. Finally, ORF is prompted to clear the demand file entry for a job when that job is removed from the system.

The flowchart of ORF is in figure 12-16.

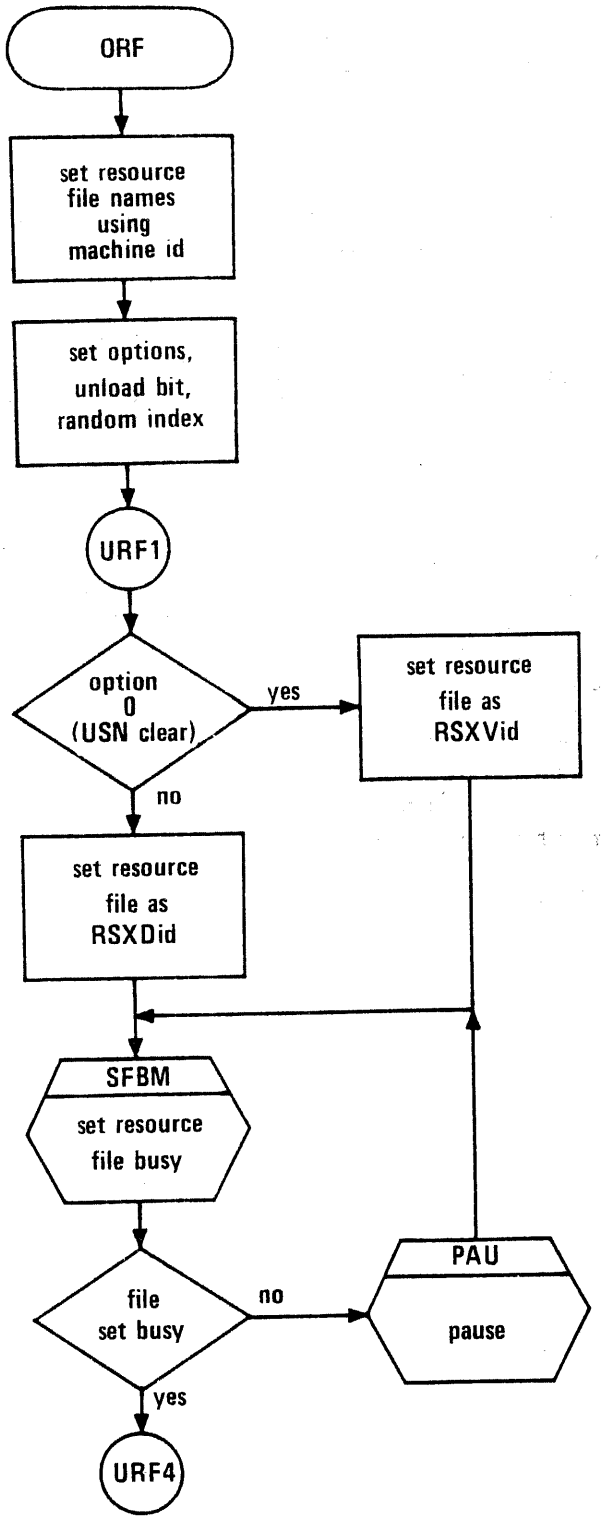


Figure 12-16. ORF - Update Resource Files

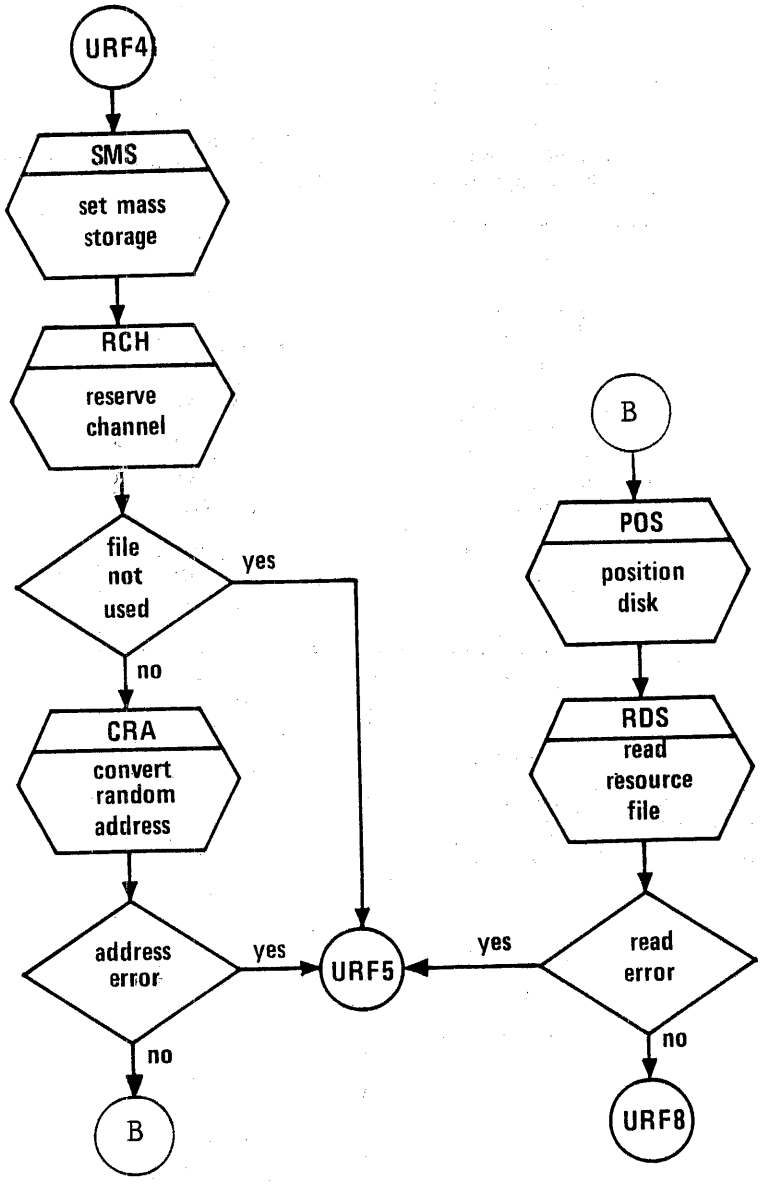


Figure 12-16. ORF - Update Resource Files (Continued)

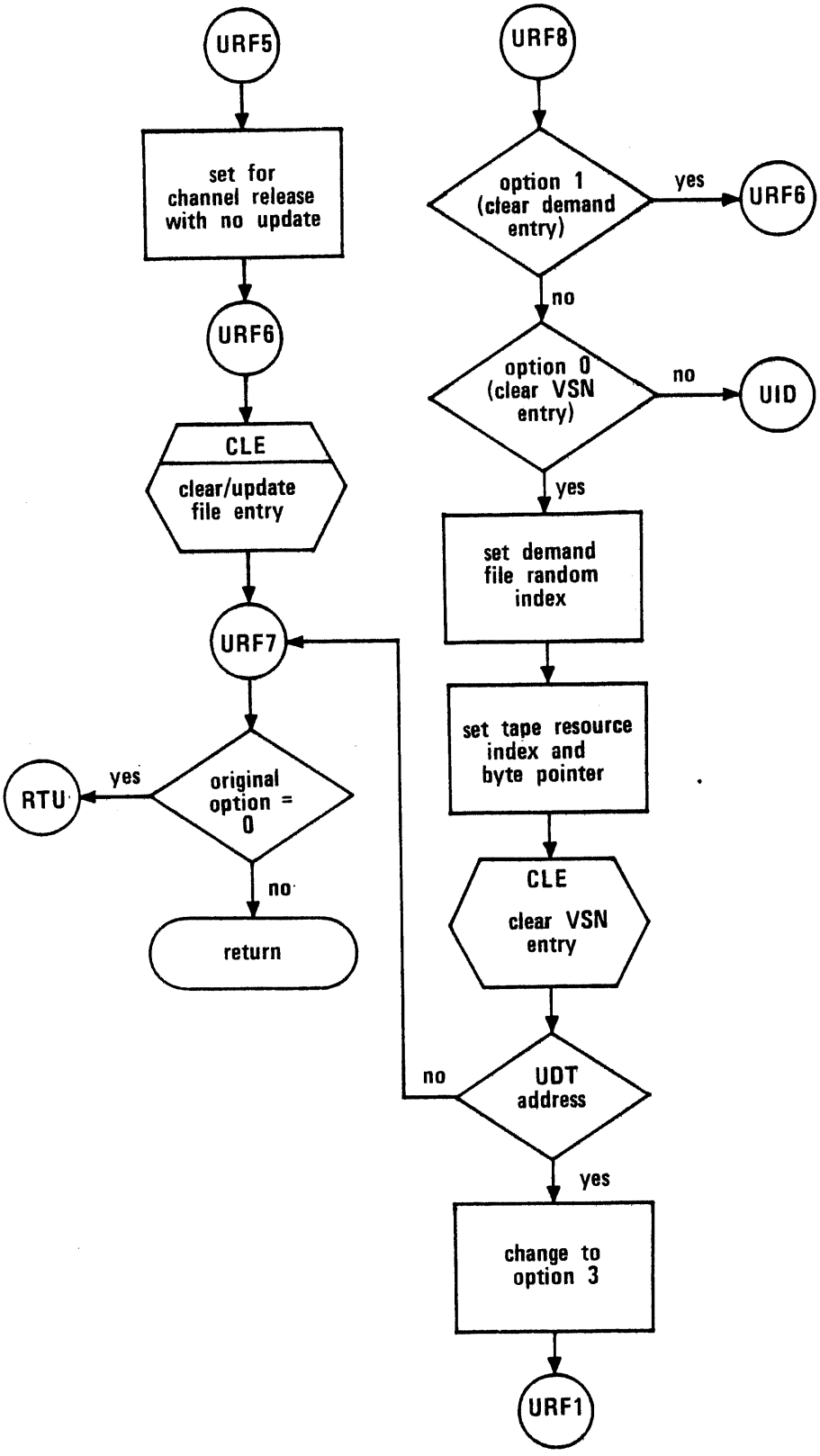


Figure 12-16. ORF - Update Resource Files (Continued)

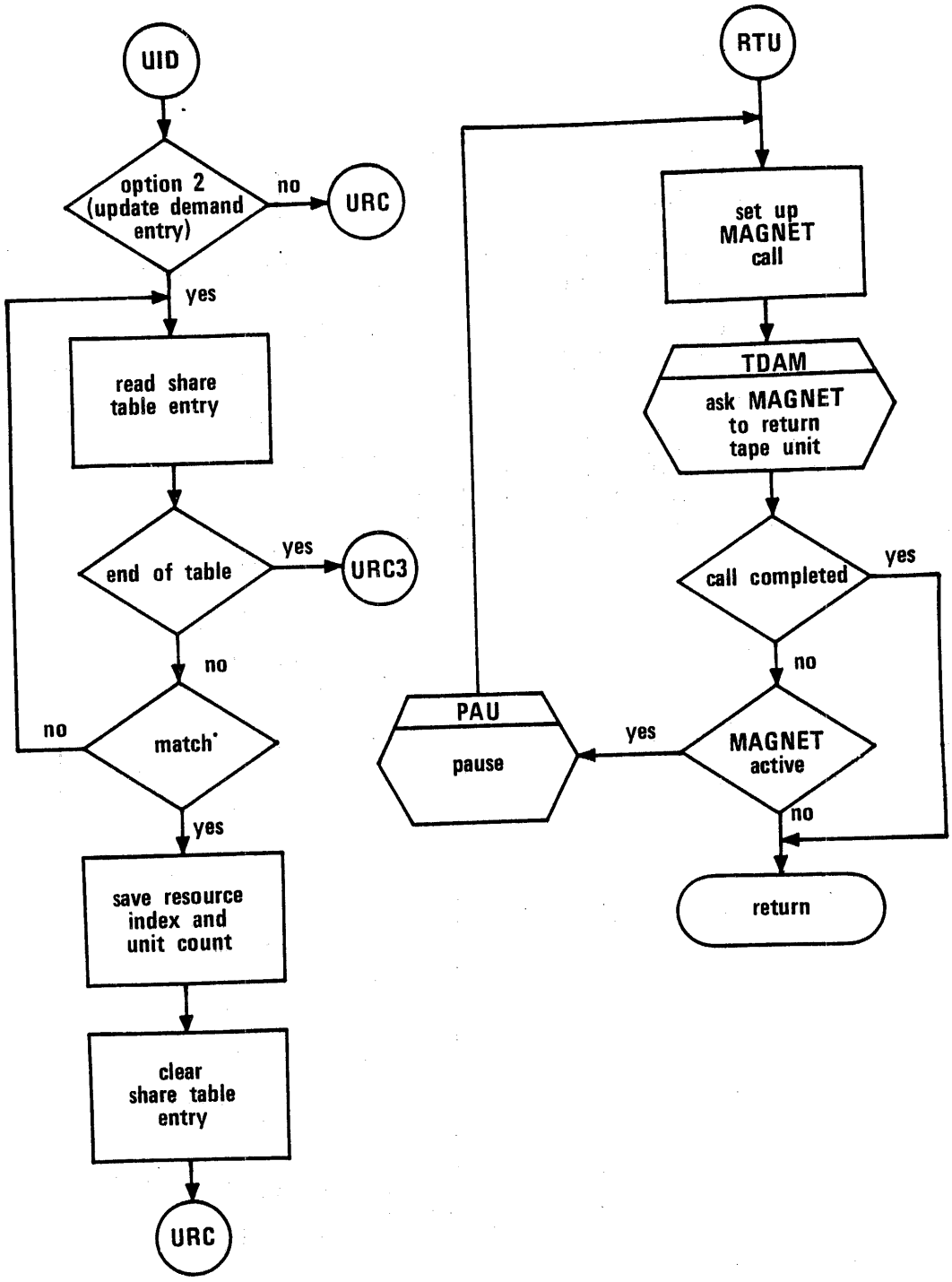


Figure 12-16. ORF - Update Resource Files (Continued)

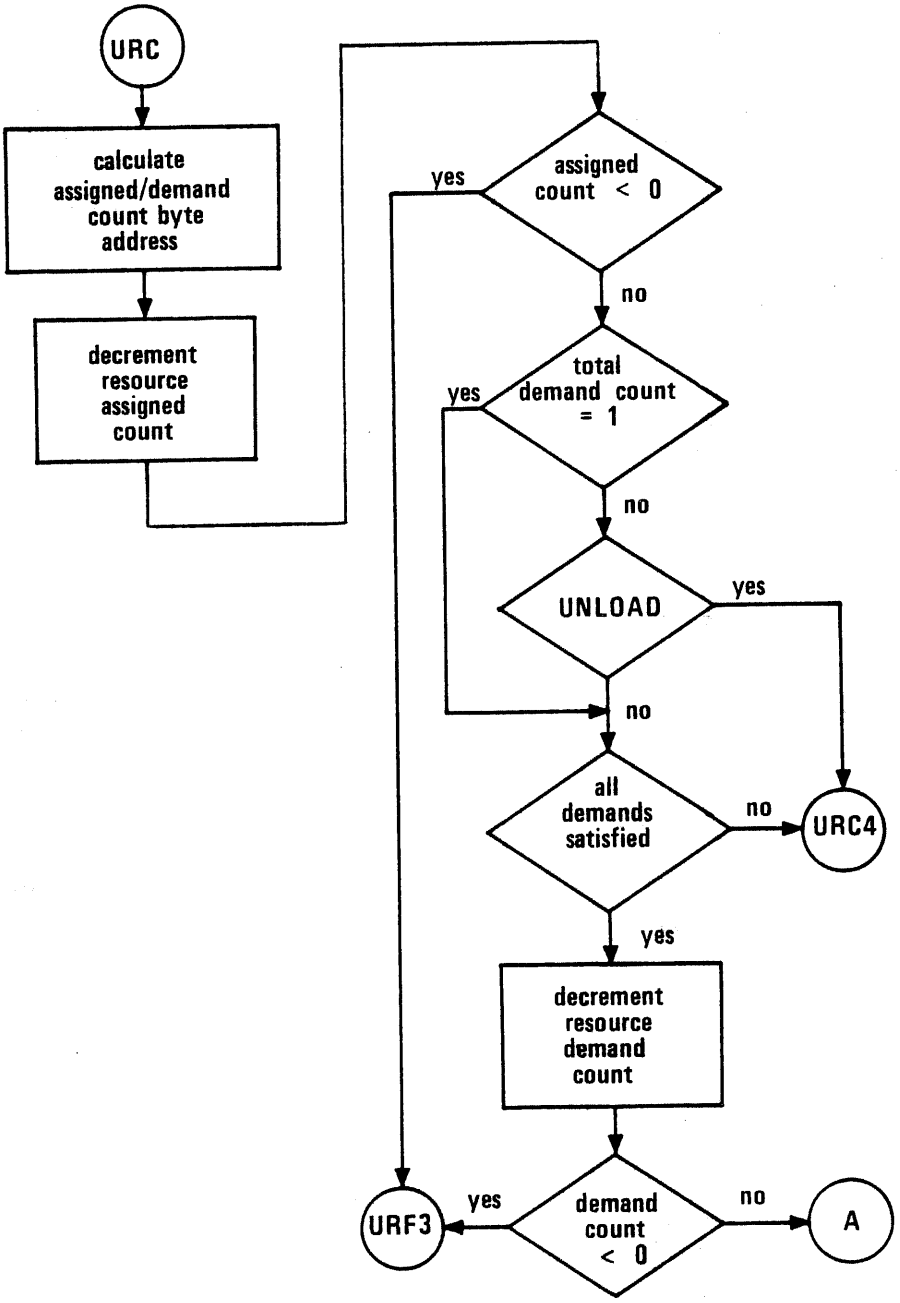


Figure 12-16. ORF - Update Resource Files (Continued)

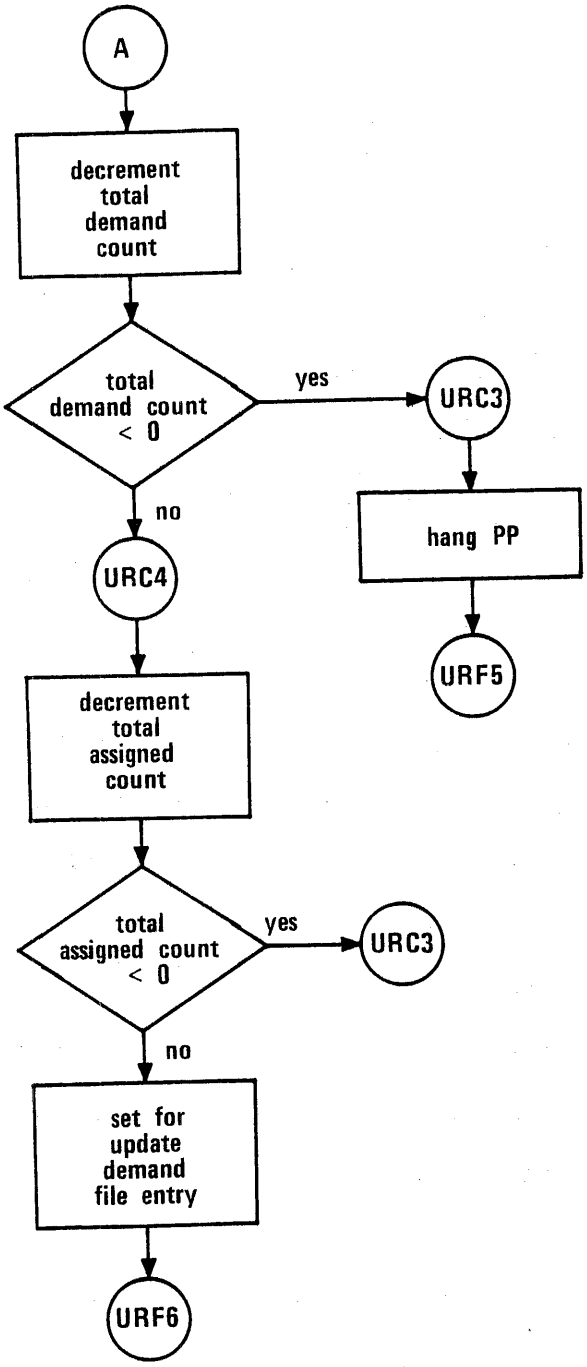


Figure 12-16. ORF - Update Resource Files (Continued)

RESEX ORGANIZATION

An outline of the subroutines and storage areas contained in RESEX follows.

- FETs for:
  - Requested file
  - VSN entry file
  - Resource demand file (RSXDid)
  - VSN file (RSXVid)
  - Scratch files
- SSJ= parameter block
- Control point area parameters
- Temporary storage (flags)
- COMSRSX
- Event skeletons
- Resource request block
- Overcommitment algorithm control (COM) and subroutines:
  - BRE Build resource environment
  - BRT Build resource demand table (RDT)
  - BSF Build scratch file
  - CFU Check for unit
  - CIC Check for internal conflict
  - CRC Check requestor completer
  - CRQ Check request
  - DEI Demand exceeds installation check
  - OCA Overcommitment algorithm

- Overcommitment utility subroutines:
  - CAU Count assigned units
  - DDS Determine demand satisfaction
  - DLY Delay
  - IAS Initialize assignments
  - SDT Switch 9-track drive type
- Resource reservation subroutines:
  - RMT Request magnetic tape
  - RRP Request removable pack
  - ROA Request operator assignment
  - VUR Verify unit request
- Resource file subroutines:
  - MVE Make VSN file entry
  - RDF Read demand file
  - SVE Search for VSN file entry
  - UDF Update demand file
  - URF Update resource files
- Preview display subroutines:
  - BPD Build preview display
  - EPD Enter preview buffer entry
- Utility subroutines:
  - BEV Build rollout event
  - CBP Calculate resource byte position
  - CET Copy EST
  - CFA Check file attach
  - CLB Clear buffer
  - CRM Check for resource match
  - CRV Check resource validation

CTA Count total assigns and demands

CTL Check track limit

CUP Clean up request

END Ending sequences

- Utility subroutines (Continued):

  ERR Error processing

  GFN Get family name

  GRI Get resource index

  GRL Get resource list index

  IDE Initialize demand entry

  IRC Increment resource count

  IVE Initialize VSN entry

  OPN Open file

  PER Process error message

  PNE Process jobname error message

  PRO Process timed/event rollout

  PIT Process interrupt

  RPV Reprieve processor

  RSB Read subsystem block

  WSB Write subsystem block to MAGNET

- Common decks
- Buffers (overlay subsequent routines)
- Control statement processors (second overlaid group):

  LABEL

  RESOURC

  VSN

- External call processors LFM and REQ
- Control statement utility routines:

  BVE Build VSN file entry

  CCR Check for conflicting resources

  CJV Check job validation

  MFE Make resource file entries

  SVI Search for VSN index

  TBD Tape block definition

  VDD Validate dependent defaults

- Preset code temporaries
- Preset code common decks
- Control statement processors (first overlaid group) ASSIGN and REQUEST
- External call processor PFM
- Control statement preprocessors CCP (control statement preprocessor) and PCV (preset control point values)
- Assemble magnetic tape options (AMO) which call any of the following processors:

SCD	SCI	SNS
CRD		SPO
FID	SCV	STD
	SFA	STF
NMD	SFS	STK
RTC	SID	VSP
RTD	SLT	WRL

- Control statement processing subroutines:

  AOP Analyze optional parameters

  CLP Call POP (pick out parameters)

  ENF Enter numeric label field

  FSC File status check

  GRD Generate retention date

  ILF Initialize label FET

- External request subroutines CLF (convert LFM call to FET) and CSF (convert NOS/BE call to FET)

All magnetic tape operations for NOS are controlled by the magnetic tape executive (MAGNET). MAGNET executes at a control point and maintains information for RESEX (resource executive) and the system. The E,P display is updated in the MAGNET field length by RESEX and is displayed by DSD. E,T display information is taken from UDTs (unit descriptor table) which are maintained by MAGNET. Certain tape commands (such as UP/DOWN channel and ON/OFF unit) are processed by MAGNET via the TDAM monitor function. CIO places tape read/write requests in the UDT in MAGNET also via the TDAM function. In addition, MAGNET statuses unassigned tape units for conditions such as tape labeled/unlabeled and ring in/out. Refer to section 12 for information concerning RESEX.

MAGNET/1MT STRUCTURE

The tape subsystem consists of CP routines MAGNET and MAGNET1, and PP routine 1MT. MAGNET is the run-time executive and processes requests from DSD, RESEX, and CIO. MAGNET1 is the MAGNET termination processor.

The PP portion of the subsystem consists of 1MT (main routine) and the following overlays:

Overlay	Description
3MA	Initialize tape executive
3MB	Function reject processor
3MC	ERRLOG message processor
3MD	MTS/ATS ERRLOG message processor
3ME	MTS/ATS special message processor
3MF	Load conversion memory
3MG	Drop PP processor
3MH	Control point/coded preset
3MI	Complete user FET
3MJ	Issue user messages
3MK	User job operations
3ML	Read function processor
3MM	Read long block processor
3MN	Read label processor
3MO	Multifile auxiliary processor
3MP	Open operations
3MQ	Tape positioning operations
3MR	MMTC Read error processor
3MS	MTS/ATS read error processor
3MT	Write function processor
3MU	Write long block processor
3MV	Write label processor
3MW	MMTC Write error processor
3MX	MTS/ATS write error processor
3MY	Tape monitoring preset
1LT	Long block helper processor

Routine 1MT overlays itself extensively to conserve space. For example, it uses the 5-byte header on PP routines; therefore, extreme care must be taken when attempting modifications.

All magnetic tape equivalences are defined in common deck COMSMTX. These equivalences are used by MAGNET, RESEX, 1MT, CIO, DSD, ORF, and 1DS.

MAGNET CONTROL POINT INITIALIZATION

The control point for MAGNET is initialized in the same manner as TELEX. That is, DSD calls 1DS to process the operator typein, n.MAGNET. Routine 1DS then calls 1MT to initialize the executive. Routine 1MT determines that this is an initial call and executes overlay 3MA to perform control point initialization. Overlay 3MA is also called for a level 3 recovery. Routine 1MT determines that it is an initialization call when the next statement index (byte 3) of word CSPW is zero. If this value is greater than or equal to CSBW+2, then it is a level 3 recovery call. The following is performed by 3MA for an initialization call.

- Calls INI (initialize CPU) to store the job name of MAGNET in control point area word JNMW with system origin type set. Sets CPU priority to 76B. Sets MFL and MAXFL to 50K.

- Calls SCC (set up control cards) to set the following control statements in CSBW of the control point area:

  ```
  MAGNET.
  MAGNET1.
  EXIT.
  MAGNET1.
  ```

  In addition, the next statement index pointer in CSPW (byte 3) is set to CSBW for initialization or CSBW+1 for level 3 recovery.

- Calls PCC (process control cards) to do the following:

  1. Requests 100B words of field length.

  2. Sets byte 4 of UITW in MAGNET to 1 as an interlock word.

  3. Requests 1AJ to process the next control statement. For initialization, this is MAGNET and for a level 3 recovery, MAGNET1.

  4. Waits until MAGNET clears interlock word.

- Calls LFT (load function table) to write the 1MT function table and DSD error messages to MAGNET starting with TFUN.

- Calls SCS (set character tables). If the 63-character set is selected, the 64-character table is modified.

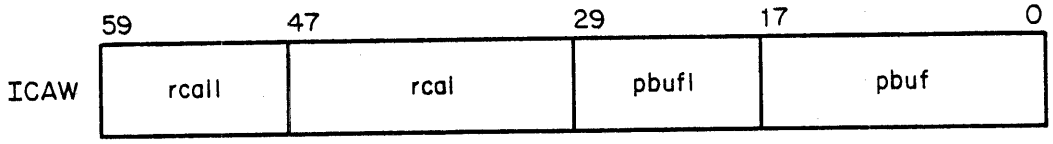
- Calls EST (pre-process EST). The EST is searched for tape equipment and the channel numbers are saved. A bit for every tape channel is set in word CHAN in MAGNET (maximum of 4 channels).

- Calls LBC to load MTS (66x) buffer controller and to load conversion tables to MTS and ATS controllers.

- Calls BDW (build equipment definition words) to build byte 0 of UST1 of the UDT for each tape unit and store in an initialization buffer in MAGNET1 starting with UINT. This is used later to build the UDTs.

- Calls BIW (build interlock words) to build the MAGNET/1MT communication words starting at location UITW in MAGNET low core.

- Sets inter-control point word (ICAW) of the control point area as shown in figure 13-1.



rcall	Length of RCAL
rcal	RESEX request block buffer
pbufl	Length of PBUF
pbuf	FWA of preview buffer (read by DSD to build Preview display)

Figure 13-1. ICAW Word

- Sets 760000 event to indicate that MAGNET is available.

- Drops PPU. MAGNET is now in control.

For a level 3 recovery, 3MA only resets the control statements and reloads the 1MT function table and DSD error messages into the low core of MAGNET.

MAGNET INITIALIZATION

After MAGNET is loaded, execution begins at the preset routine PRS. PRS clears the interface area from UITW+1 through TPRO, followed by location 1 through UITW-1, and finally UITW. At this point MAGNET waits until 1MT is finished building the equipment definition words starting with UINT. 1MT signals MAGNET that it is finished by setting UITW nonzero. Next, PEQ is called to build the UDTs (refer to figure 13-2), one for each unit as sensed by 1MT. REL is called to perform instruction modification in the main routine where the operation definitions were used. The UDTs start at TDTAB and overlay the preset code. A maximum of 16D (MUNIT in COMSMTX) UDTs are established in MAGNET. PEQ also sets up a pointer word in RA+3 called UBUF, that points to the list of UDT entries.

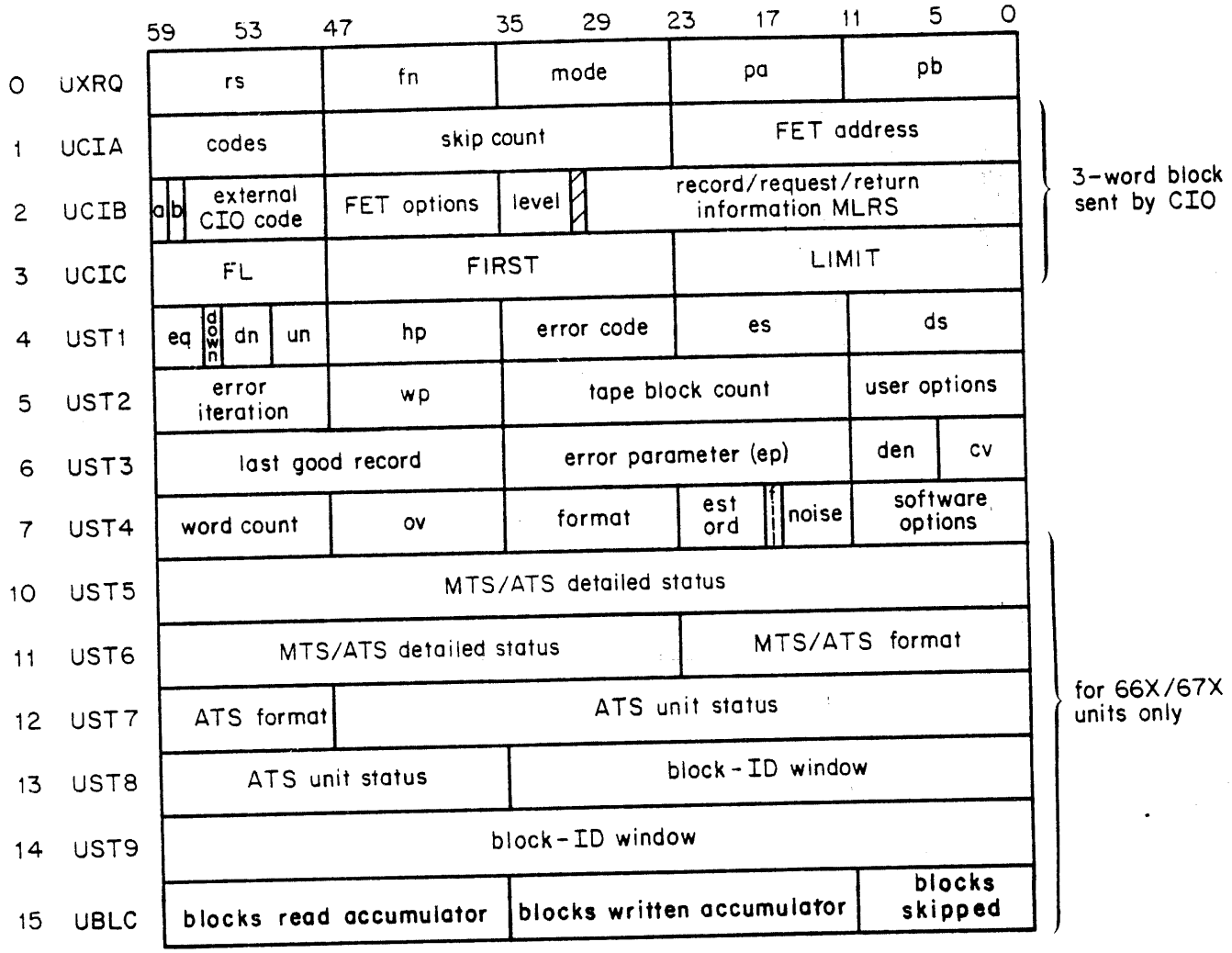


Figure 13-2. Unit Descriptor Table Format

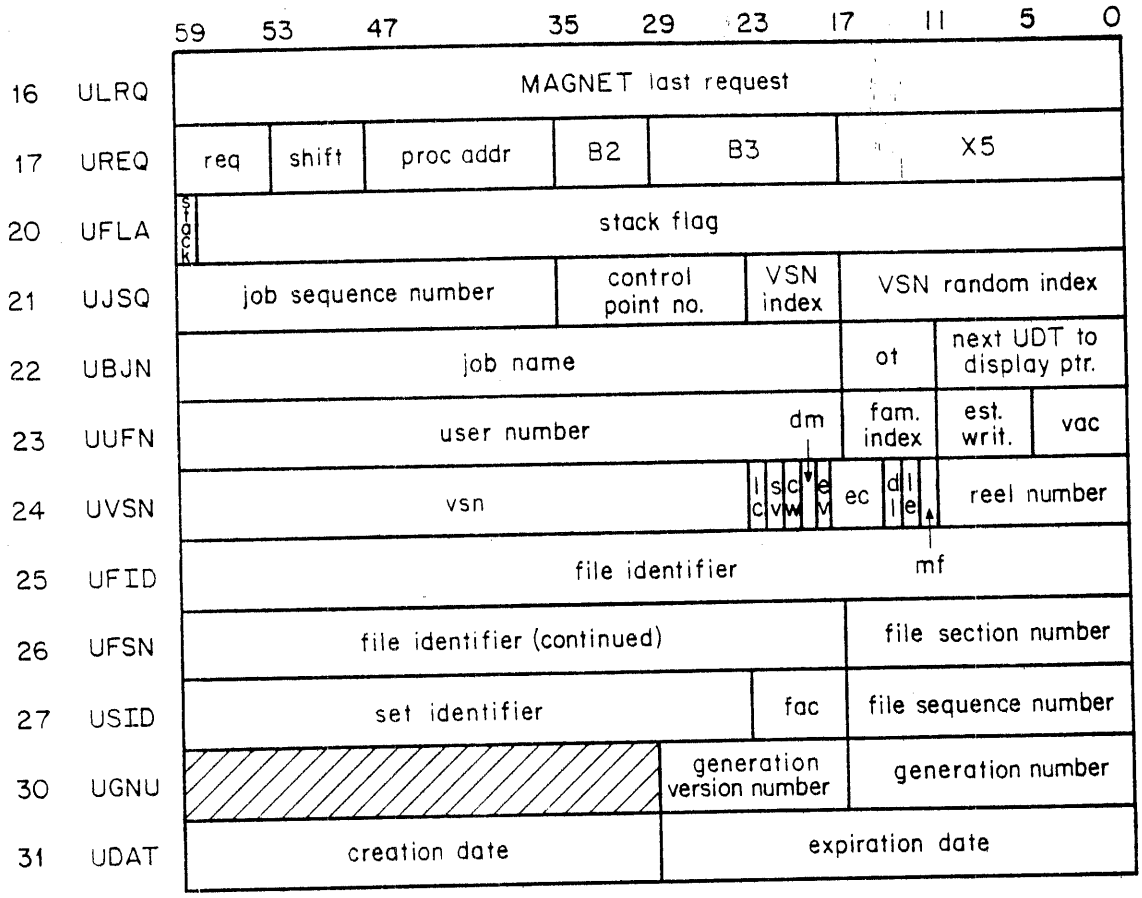


Figure 13-2. Unit Descriptor Table Format (Continued)

The fields of figure 13-2 are defined as follows.

rs — return status (RS):

RIP	(1)	Request in progress
NCP	(2)	Normal completion
REQ	(3)	Requeue with delay
ERR	(4)	Error return

fn — function number (FN):

SED	(1)	Set equipment definition
CUF	(2)	Complete user FET
MAB	(3)	Issue message and abort job
FNH	(4)	Process function
SKP	(5)	Skip
OPF	(6)	Open function
RDF	(7)	Read data
RLA	(10B)	Read label
WTF	(11B)	Write data
WLA	(12B)	Write label

mode — Mode (MD):

Bit	Description
11	Reverse (read data only)
10	Set IN=OUT=FIRST; reverse (read labels only)
9	First pass flag for reverse skip
8	EOR/EOF written this operation
7	Not used
6	Coded
5	200/204 control word
4	260/264 control word
2-3	0- PRU operation 1- EOR operation 2- EOF operation 3- EOI operation
1	Not used
0	Not used

pa — Parameter A (see individual functions for allowable values)

pb — Parameter B (see individual functions for allowable values)

codes Internal CIO codes:

0000B	Read
0002B	Write
0104B	Skip forward
4204B	Skip backward
4300B	Backspace PRU
6400B	Rewind
0500B	Open
4500B	Open rewind
0600B	Close
4600B	Close rewind
1000B	Evict

Note

Bits 9-6 of internal code are used as an index into table TPRO by MAGNET.

a Set if autorecall

b Set if data in buffer

external CIO code User supplied CIO request code

level Level number (74B-SKIPF, 0-SKIPEI)

eq Equipment number (used to connect equipment)

down Unit down flag

dn Channel designator (bit 4=first tape channel, bit 5=second tape channel, etc.)

un Unit number

hp Hardware options (HP)

Bit	Description
11	Last operation write
10	Last block EOR/EOF
9	Blank tape
8	Not Used
7	End of set
6	Not used
5	MTS controller
4	ATS controller
3-2	Unit speed for ATS/MTS units (0=100 ips; 1=150 ips; 2=200 ips)
1	GCR tape unit
0	9-track unit

es — Extended status (ES) (status 2 for 65X)

ds — Device status (DS) (general status for MTS/ATS, and status 1 for MMTC converted to MTS/ATS format)

wp — Block - ID window pointer (points to most recent entry)

user options — User processing options (UP)

Bit	Description
11	Tape labeled
10	Nonstandard label
9-8	Label type (0=ANSI; 1-3=unused)
7	Assigned message issued for this unit
6	Next VSN message issued for current reel swap
5-1	Not used
0	Coded

ep — Error parameters (EP and EP+1)

Read error recovery:

EP,11	Reverse direction flag
EP,10	Opposite parity mode
EP,9	Not used
EP,8-6	Clipping level
EP,5-0	Re-entry code
EP+1,11-9	Clipping level being tried
EP+1,8-6	Retry count
EP+1,5-3	Normal parity reread count
EP+1,2-0	Opposite parity reread count

Write error recovery:

EP,11	Verify in progress
EP,10	Erase error has occurred
EP,9-0	Not used
EP+1,11-6	Not used
EP+1,5-0	Number of erases

den — Density:

D02 (1)	200 bpi
D05 (2)	556 bpi
D08 (3)	800 bpi
D16 (4)	1600 cpi
D62 (5)	6250 cpi

cv — Conversion mode:
- BCD (1) BCD conversion (7-track)
- ANS (2) ANSI conversion (9-track)
- EBC (3) EBCDIC conversion (9-track)

word count — Word count (WC), $0 \leq WC \leq 512$ (for L-format tapes, WC is number of words in last partial chunk)

ov — Overflow (OV); for F and L format, the number of chunks (600B words each)

format — Format (FM):
- TFI (0) Internal binary
- TFSI (1) System internal binary
- TFF (2) Foreign format
- TFS (3) Stranger binary/coded
- TFL (4) Stranger long blocks binary/coded

est ord — EST ordinal of equipment

fill — Fill status (set if fill ok)

noise — Noise block size in bytes

software options — Software option (SP):

Bits	Description
11-10	End of reel processing: 0 Tape mark and trailer sequence, option 3, P0=S 1 EOT (accept PRU), option 2, P0=P 2 EOT (discard PRU), option 1, P0=I
9-8	Not used
7	Issue all messages to user dayfile setting, P0=M
6	Disable error correction on GCR write, P0=G
5	Inhibit unload, P0=U
4	Ring in required, P0=W
3	Ring out required, P0=R
2	Inhibit error processing, P0=E
1	Accept data on RPE/WPE without ep set, P0=N
0	Abort RPE/WPE with ep set, P0=A

req — 0 or 1, 3, 5, 7 if the request is stored in the QUEUE table

shift — Shift count (12, 24, 36, 48) used to shift a byte of a processor string entry to the low 12-bit position

proc addr	Address of processor string entry
stack	If set, there are entries in the QUEUE table for this unit
vsn index	VSN index (pointer to word in VSN entry containing the VSN)
ot	Job origin type
est written	EST order of unit that the tape was written on
vac	Volume accessibility character
vsn	VSN (6 character VSN)
lc	Label checking in progress (bit 23)
sv	Scratch VSN (bit 22)
ow	Open performed (bit 21)
dm	Density mismatch detected (bit 20)
ev	Equivalenced VSNs for this reel (bit 19)
ec	Error code for DSD display (bits 18-15):

1	EQxx needs label
2	EQxx cannot access data
3	EQxx ring conflict
4	EQxx wrong VSN
5	EQxx drive conflict
6	EQxx density mismatch
17-7	Reserved

dl	Default label (bit 14)
le	Label expired (bit 13)
mf	Mount flag (bit 12)
fac	File accessibility character

Each UDT entry is UNITL (31B) words in length. PEQ sets the SED function (set equipment definition) in each UDT entry; therefore 1MT will be called to determine the type of each unit.

PEQ sets up another low core pointer, UQUE. UQUE specifies the first word address of the queue table which follows the UDT list and is initialized with 10B empty entries. The queue table is terminated by two words with all bits set. Figure 13-3 shows the memory map of MAGNET after initialization; figure 13-4 illustrates a detailed map of MAGNET low core. figures 13-5 through 13-14 detail portions of MAGNET low core.

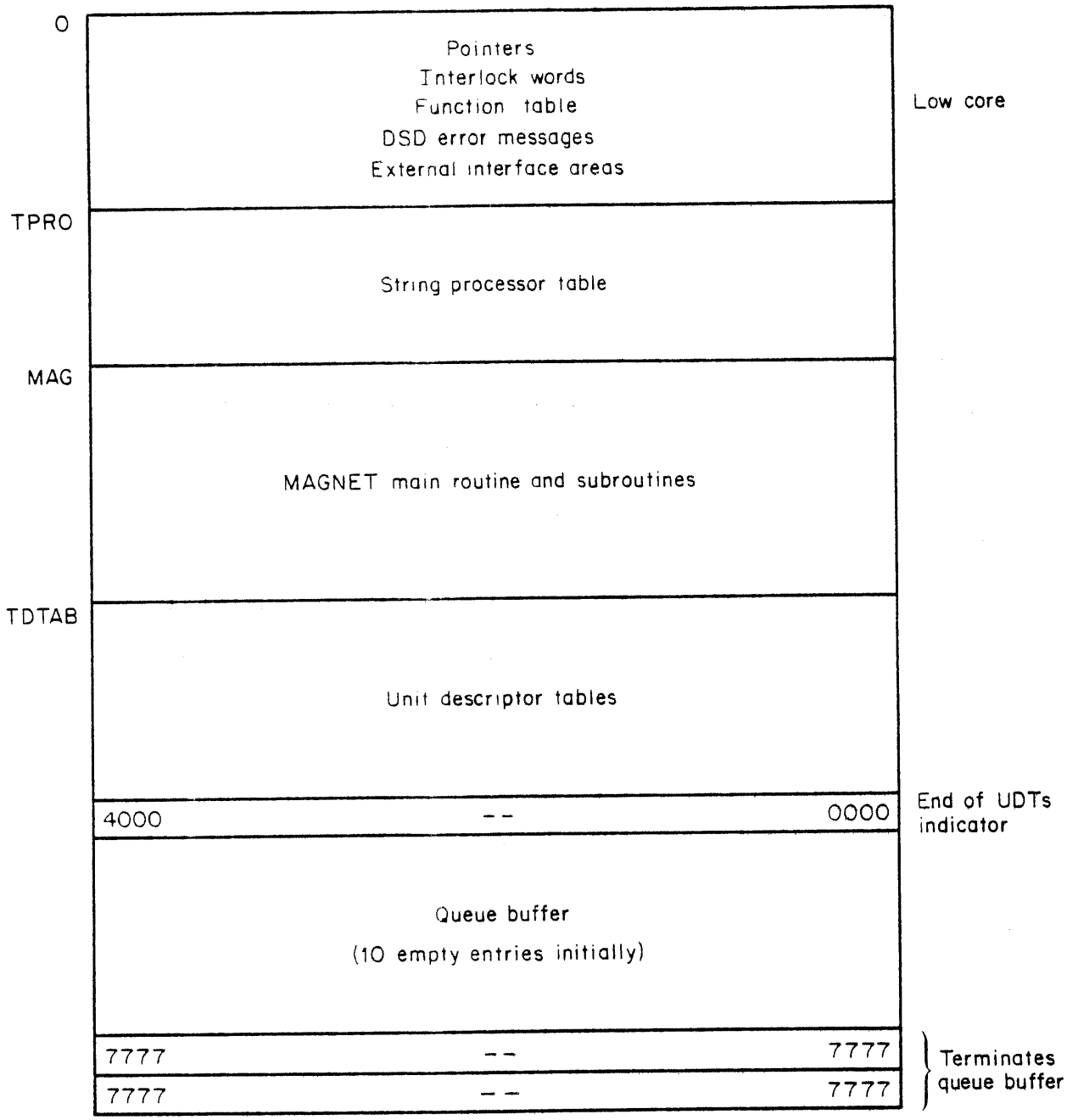


Figure 13-3. Overview of MAGNET After Initialization

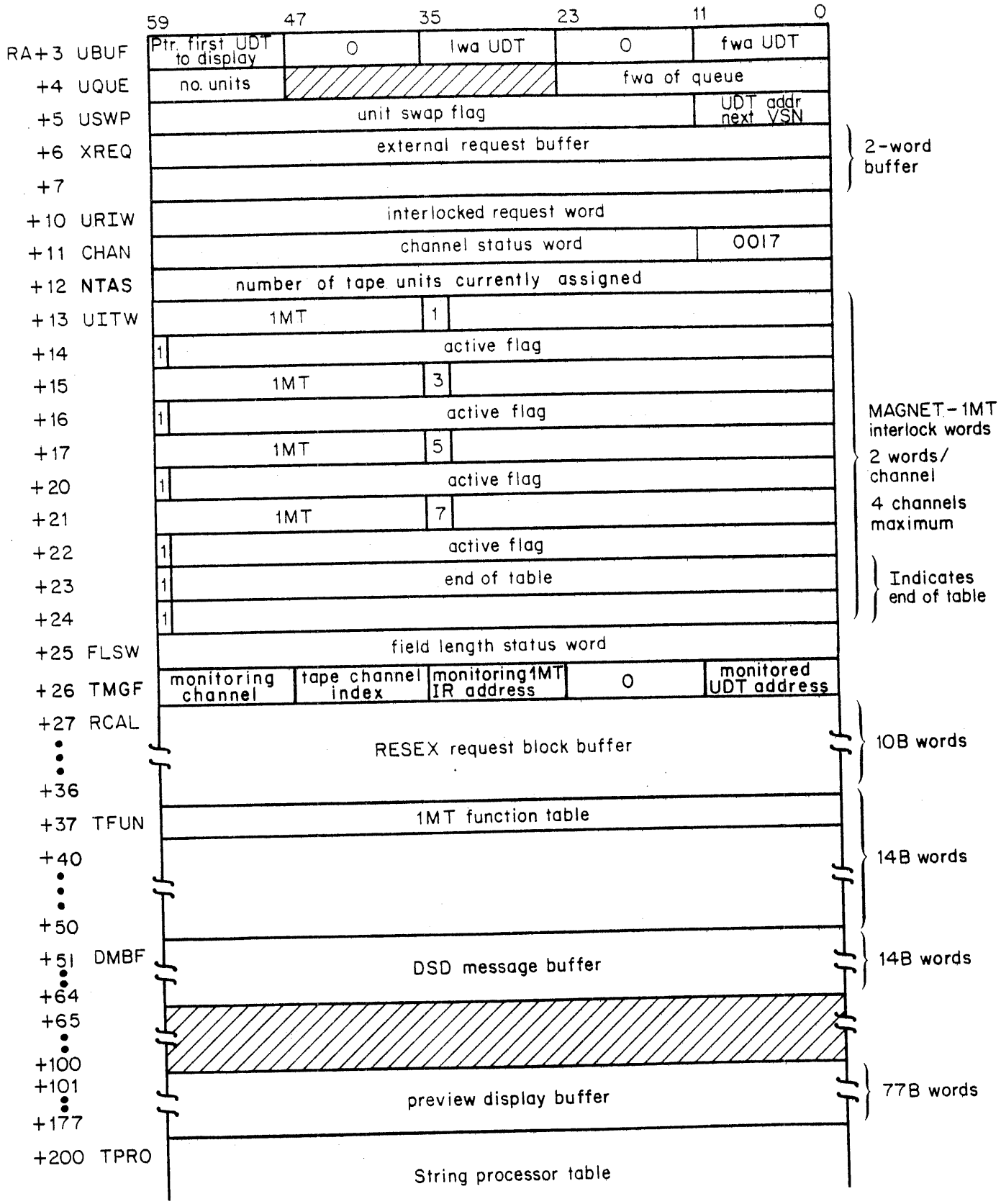
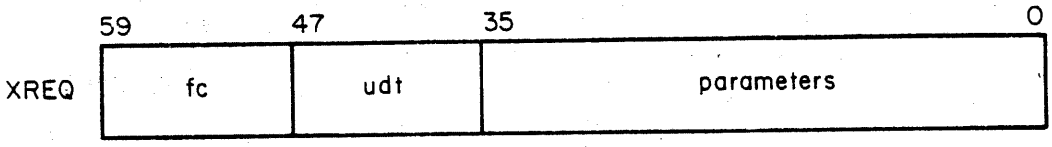


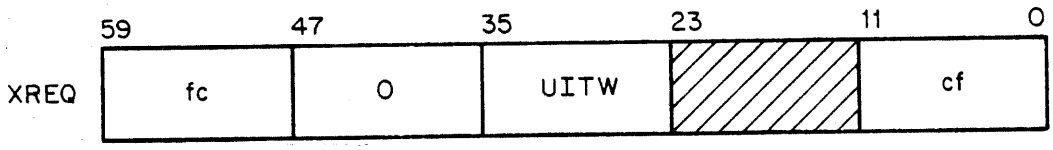
Figure 13-4. Detailed Map of MAGNET Low Core



fc Function code:

0	Return unit
1	Enter VSN for unit (parameter = VSN number)
2	Unload unit
3	Scratch VSN
5	ON/OFF unit (parameter = 0 if ON unit, 0 if OFF unit)

UDT UDT address



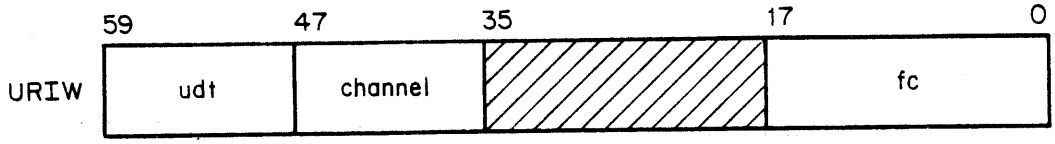
fc Function code:

4	UP/DOWN channel

UITW Channel status word index (1, 3, 5, 7)

cf Channel function (0 = UP, 1 = DOWN)

Figure 13-5. XREQ Format



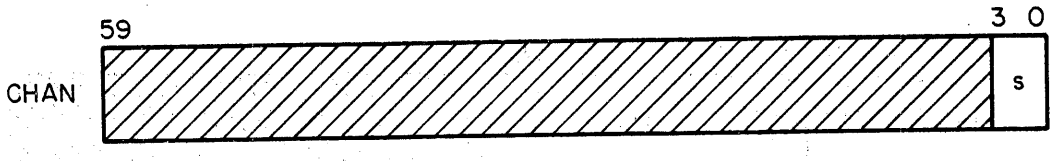
udt UDT Address

channel Channel status word index (1,3,5,7)

fc Function code:

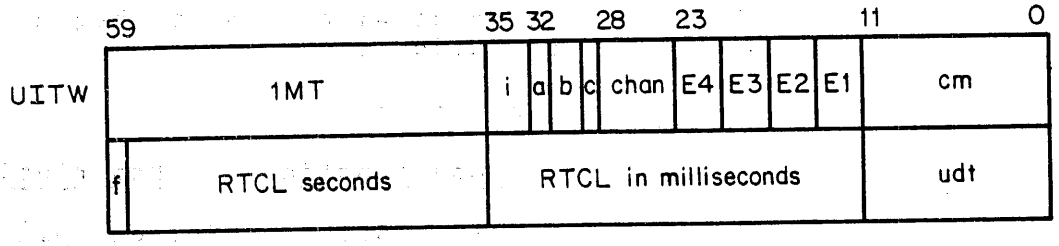
0	Interlock and reserve a unit
1	Down a unit

Figure 13-6. Interlock Request Word



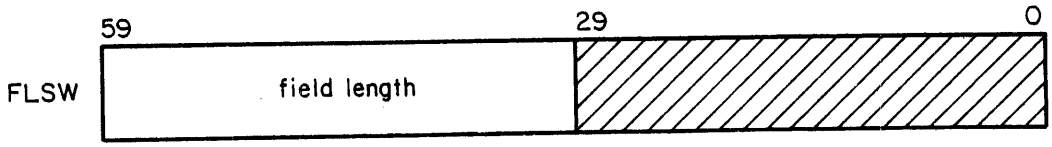
s Channel status; one bit for each tape channel - 4 maximum (0 = channel down, 1 = channel up)

Figure 13-7. Channel Status Word



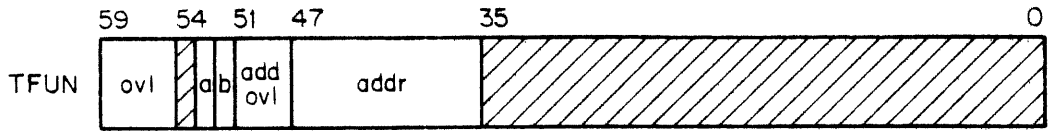
i	Index (1,3,5,7) (bits 35-33)
a	Channel down flag (bit 32)
b	Controller type (bits 31-30) 0 MMTC 1 ATS 2 MTS 3 MTS without BID
c	Dedicated channel flag (bit 29)
chan	Channel number (bits 28-24)
E4-E1	Controller/equipment numbers
cm	Conversion memory (1=BCD, 2=ANSI, 3=EBCDIC) if any required (MMTC only)
f	Active/inactive flag (1=active)
udt	Pointer to last UDT processed

Figure 13-8. MAGNET-1MT Interlock Words



field length Current field length requested

Figure 13-9. Field Length Status Word

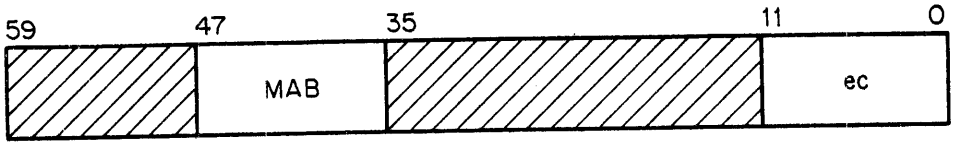


ovl	Overlay name (1 character)
a	Function requires ready and not busy status
b	Do at user's control point
add ovl	Additional overlay to be loaded first
addr	Address of function processor (address within ovl to begin execution at)

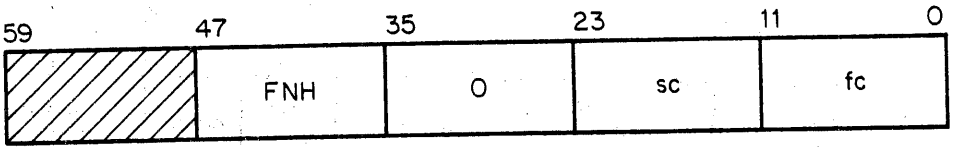
Functions:

Code	Function	Description
1	SED	Set equipment definition
2	CUF	Complete user's FET
3	MAB	Issue message(s) to user and abort job
4	FNH	Process function
5	SKP	Skip
6	OPF	Open
7	RDF	Read data
10	RLA	Read label
11	WTF	Write data
12	WLA	Write label

Figure 13-10. 1MT Function Table Entries



ec Error code. If EC=0, then check if any error flags set by system and return ef value to MAGNET in bits 23-12.



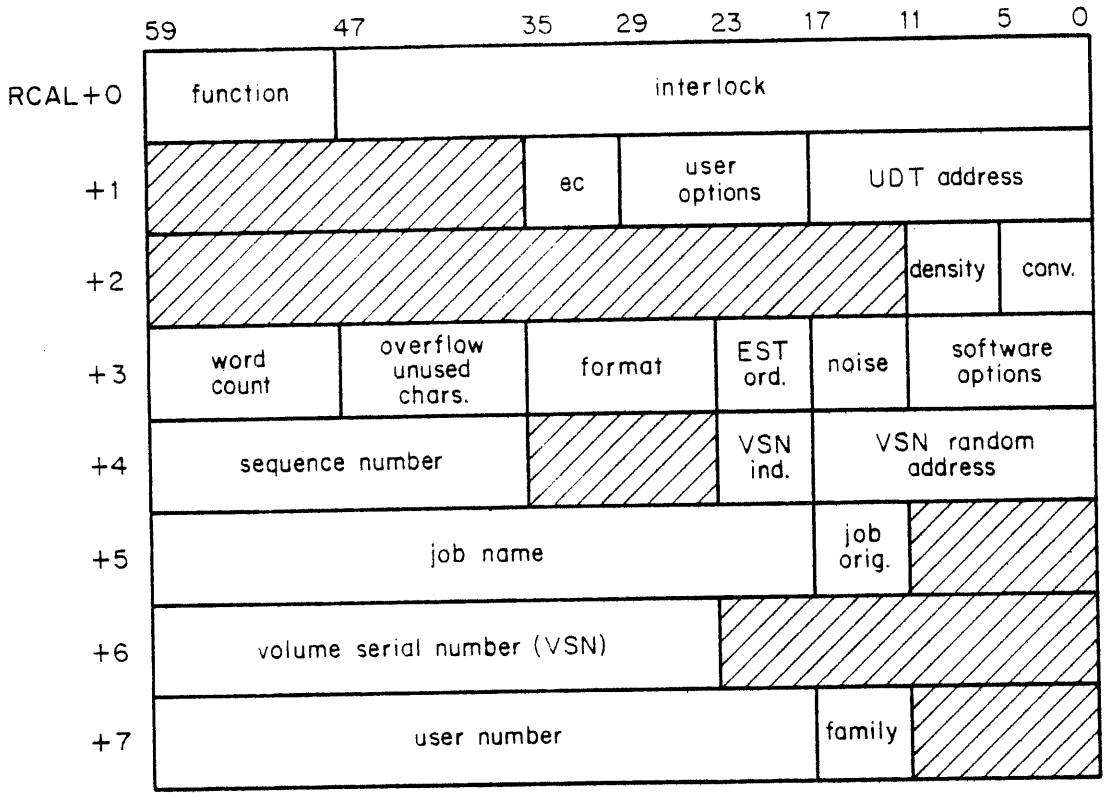
sc Subfunction code: (for fc=6)

0	Issue PRUs transferred message
1	Issue tape unit assigned message
2	Issue tape unit returned message

fc Function code:

0	Rewind tape
1	Unload tape
2	Select density
3	Clear job assignment
4	Read VSN from VSN file
5	Perform unit swap
6	Issue message to account file
7	Issue next VSN message for multi-reel
10	Post message at MAGNET control point

Figure 13-11. MAB and FNH Function Requests

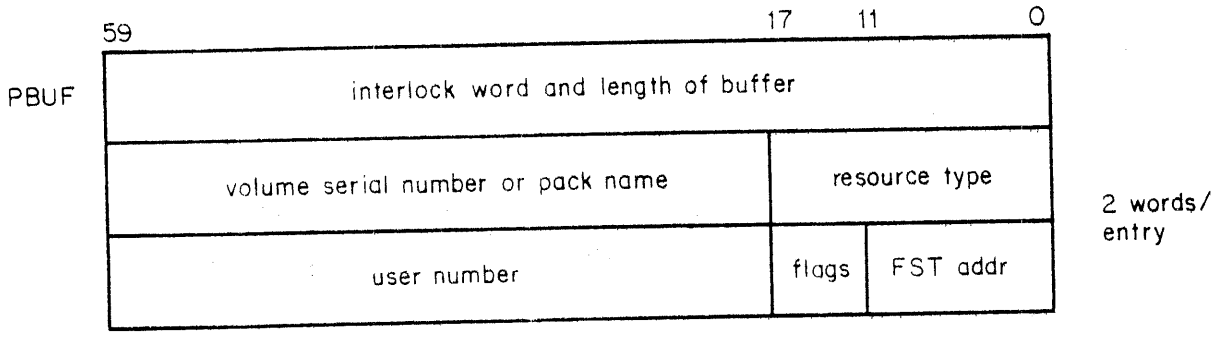


function Function code:

0 Assign unit
1 Set error code in UVSN word of UDT

ec Error code for function 1

Figure 13-12. RESEX-MAGNET Call Block



resource type	MT, NT, PE, HD, GE, DI-1 through DI-8, DJ-1 through DJ-8, DL-1 through DL-8, DK-1 through DK-8
flags	Ring in (bit 12), ring out (bit 13), labeled (bit 14) (for tapes only)

Figure 13-13. Preview Display Buffer

	59	47	35	23	11	0	
TPRO+0 (PWDA)	LAB1	WDA	CWC+ 6000B	WDA1	0		Write
+1	PSKP	0	0	0	0		Skip forward
+2	PSKP	0	0	0	0		Skip backward
+3	PSKP	0	0	0	0		Backspace PRU
+4	FET1	REW	0	0	0		Rewind
+5	OPE	POLA	0	0	0		Open
+6	CLO	0	0	0	0		Close
(PRDA) +7	LAB	RDA	CRK+ 6000B	0	0		Rewind (special case)
+10	PMAB	0	0	0	0		Evict
(POLA) +11	OPE	4000B	4000B	4000B	CLM+ 6000B		
+12	FNH	RLA	4100B	4400B	4000B		
+13	OPE1	FET1	0	0	0		

First entry group contains the first nine entries in the table. Each entry in this group can have no more than four processors (limited to one central memory word) because these entries are indexed by a portion of the internal CIO code. Read data is given special treatment. The second entry group contains the remaining entries in the table. There is no restriction on the length of these entries.

Figure 13-14. Table of Processor Strings

1MT INITIALIZATION

The initialization of 1MT each time it is loaded consists of the following steps.

1. If an initialize call or level 3 recovery call, execute overlay 3MA (described earlier in this section).
2. Set up the number of control points.
3. Set FWA, LWA+1 and length of UDT.
4. Set processor number (one of 4 values, 1, 3, 5, 7).
5. Set channel, enable/disable channel modification.
6. Determine if conversion memory load needed (MMTC only).
7. If MMTC controller, modify certain instructions.
8. If ATS controller, execute controller diagnostic.
9. Load conversion memory for 65X controller if needed.
10. Preset controller options and exit to main loop.

MAGNET RUN-TIME EXECUTIVE

One of the main components of this module is the string processor table TPRO (refer to figure 13-14). Each entry is generated at assembly time by the PROC macro which results in a string of processor entry point addresses (routines within MAGNET), addresses of other strings, and/or functions to be processed for a particular request. The request here refers to a CIO request. The first group of entries in the TPRO table are indexed by the internal CIO function codes defined in common deck COMSCIO (refer to figures 13-2 and 13-14 for more information). If any changes are made to COMSCIO, it may be necessary to make changes to the TPRO table.

The first group of entries is a maximum of one word in length due to the indexing explained previously. The second group of entries have no restrictions for length except that they must terminate with a zero byte. In addition to containing entry point addresses, addresses of other strings or function codes, each entry may also contain parameters which are associated with the function codes.

Up to three 12-bit parameters can be imbedded in a string per function code, but if less than three are given, the rest are assumed to be zero. A parameter is differentiated from an address or function code by the setting of bit 11 of a 12-bit byte. The three parameters, if specified, are referred to as MD, PB, and PA respectively. These parameters are stored along with the function code in word UXRQ of the UDT when making a request of 1MT.

If a byte within a string entry has bits 10 and 11 both set, it means that an error occurred in processing the last function requests. The processor defined within this byte should be executed and the rest of the string should be skipped. Otherwise, the byte is ignored if no error occurred. This byte contains either an address of a processor routine within MAGNET or an address of another string.

MAGNET's main loop consists of the following: update internal time counters and if necessary reduce field length, and execute major subroutines CUT, CXR, ASU, IUN, and PPU. Routine CUT checks all UDT entries for outstanding requests from CIO. The queue table is also searched for any outstanding requests, and if any are found, they are processed. CSR is called to process any requests from DSD/1DS or ODF/ORF. Since certain requests cannot be processed completely by CXR, it makes queue entries to complete the external request. ASU is called to perform unit assignment as requested by RESEX (RCAL contains the request block). IUN processes requests to interlock and reserve a unit. This is used by 1MT when a CONNECT operation fails. Finally, routine PPU is called to activate a copy of 1MT if necessary via a TLX call.

Routine CUT causes nearly all secondary request processing routines to be executed. These secondary routines return to CUT via one of the following exit points when completed.

Exit Point	Description
EXIX	Process next unit (next UDT entry)
EXIT	Process next request for this unit, if any
EXI1	Clear current request (UREQ=0) and enter new request
EXI2	Make request to 1MT (set function code and values for parameters MD, PA, PB in UXRQ)
EXI3	Empty request queue and make new request
EXI4	Queue new request; store in UREQ and if there is an outstanding request already in UREQ, move to the queue table
EXI5	Requeue operation; store the original request back in UREQ

The following are string processing subroutines.

Routine	Description
GPI	Get parameter item if next in the string. The next byte in the string is picked up and returned if bit 11 is set which indicates it is a parameter.

Routine	Description
GNR	Get next request. If request is already present in UREQ, it is returned. Otherwise a search of the request queue table is made for the next request for this unit.
MQE	Make queue entry. lEnters new request string in UREQ and stores the original request from UREQ (if one is present) in the request queue table in LIFO order.

Refer to figures 13-4 through 13-14 for formats of MAGNET's low core; and figures 13-2, 13-15, 13-16, and 13-17 for detailed descriptions of the UDT and other tables used by MAGNET and 1MT. Table 13-1 describes MAGNET processing options.

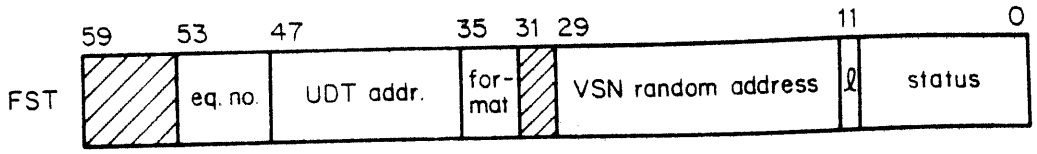
ROUTINE 1MT

MAGNET calls 1MT to perform certain tape operations that were initiated via a CIO call, DSD/1DS request, or a RESEX request and periodically calls 1MT to perform initial label check. In general, 1MT searches through the entire UDT for requests to process. The first word of each UDT entry contains the request if there is one. The function code in this request is used as an index into the TFUN table stored in MAGNET low core to get the proper overlay to load and starting address for execution. As requests are completed, a return code is placed into the first word of the UDT entry (byte 0).

If there is only one tape channel, then there can be only one copy of 1MT active at any one time. If two channels, then two copies of 1MT can be active at any one time. There can be a maximum of four tape channels. If there is one channel, then the one copy of 1MT processes all the UDTs. If there is more than one channel, then each copy of 1MT processes only the UDT entries for a particular channel.

TAPE MONITORING

MAGNET and 1MT have capabilities for monitoring tape controller dialogue, for use in conjunction with an internal tape testing tool consisting of CPU program and a PP program. Word TMGF of MAGNET low core indicates to 1MT that monitoring is in effect for the tape unit specified by the UDT address. When processing this unit, 1MT modifies its channel instructions to use the monitoring channel so that all tape controller functions for this unit are sent to the tape test PP program for logging before being relayed to the ATS or MTS tape controller. This tool is especially useful for analyzing tape error recovery processing.



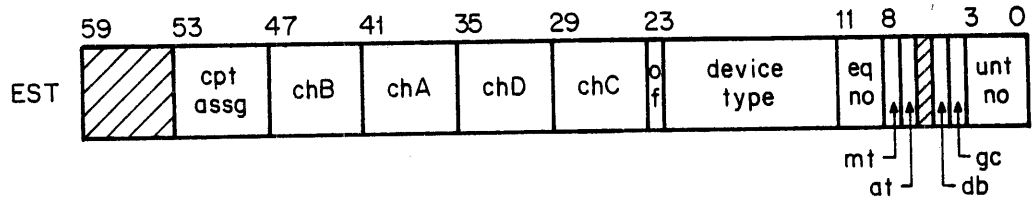
format — Format of tape:
- 0 I format (internal binary)
- 1 SI format (System internal binary)
- 2 F format (foreign)
- 3 S format (stranger binary/coded)
- 4 L format (stranger long block binary/coded)

l — Label flag:
- 0 Tape unlabeled
- 1 Tape labeled

status — Status of file:

Bits	Description
10-1	Not used (information is kept in UDT
0	Set if function complete; not set=busy status

Figure 13-15. FST Entry for Tapes



of	ON/OFF flag (set if access not allowed)
mt	MTS tape unit (bit 8)
at	ATS tape unit (bit 7)
db	Disable block-ID for MTS unit (bit 5)
gc	GCR tape unit (bit 4)

Figure 13-16. EST Entry for Magnetic Tapes

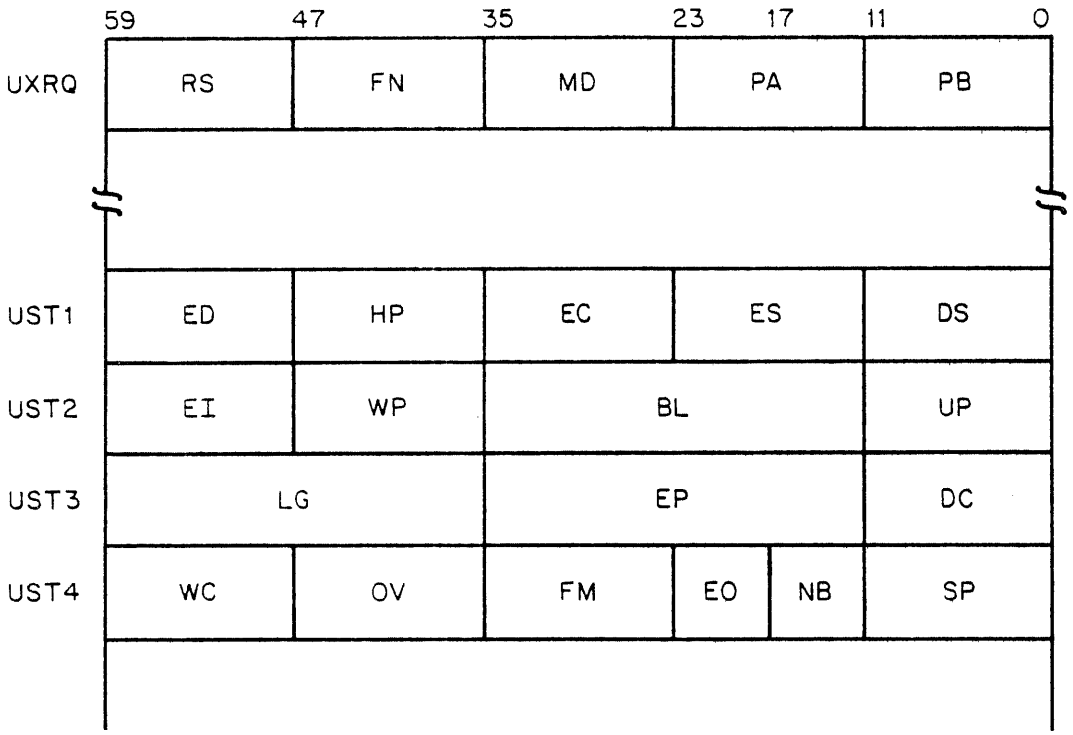


Figure 13-17. 1MT Direct Cell Allocation

TABLE 13-1. MAGNET PROCESSING OPTIONS

Byte Definition		Mode(MD)	User Options (UP)	Software Options (SP)
Dec.	Oct.			
0	0	Not used	Coded	Abort RPE/WPE with ep set, P0=A
1	1	Not used	Not used	Accept data on RPE/WPE without ep set, P0=N
2	2	Code | Read | Write 0 | PRU | PRU 1 | EOR | Buffer 2 | EOF | EOR 3 | EOI | EOF	Not used	Inhibit error processing, P0=E
3	3		Not used	Ring out required, P0=R
4	4	260/264 mode	Not used	Ring in required, P0=W
5	5	200/204 mode	Not used	Inhibit unload, P0=U
6	6	Coded	Next VSN message issued for current reel swap	Disable GCR hardware write correction
7	7	Not used	Message assignment flag	Issue all messages to user dayfile
8	10	EOR/EOF written this operation	Not used	Not used
9	11	Not used	Not used	Not used

TABLE 13-1. MAGNET PROCESSING OPTIONS (CONTINUED)

Byte Definition		Mode(MD)	User Options (UP)	Software Options (SP)
Dec.	Oct.			
10	12	Set IN = OUT = FIRST (reverse read label only)	Nonstandard label	End of reel 0-tape mark P0=S 1-EOT (accept PRU) P0=P 2-EOT (discard PRU) P0=I
11	13	Reverse (read data)	Labeled	

TABLE 13-1. MAGNET PROCESSING OPTIONS (CONTINUED)

Byte Definition		Hardware Options (HP)	Device Status (DS) 65X	Extended Status (ES) 65X
Dec.	Oct.			
0	0	9-track	Tape ready	Vertical and/or longitudinal parity error
1	1	GCR tape unit	Read/write control and/or tape unit busy	Memory parity error
2	2	Unit speed for ATS/MTS units	Write enable	Flag bit error
3	3	0 - 100 ips 1 - 150 ips 2 - 200 ips	File mark/tape mark detected	CRC error
4	4	ATS controller	Load point	Multi-track phase error or uncorrectable CRC error
5	5	MTS controller	End-of-tape	Character fill
6	6	Not used	Density 0-200 bpi 1-556 bpi 2-800 bpi/cpi 3-1600 cpi	Character crowding or dropout and false end-of-operation(NRZI)
7	7	End of set encountered		Phase error correction
8	10	Not used	Lost data	False postamble detected

TABLE 13-1. MAGNET PROCESSING OPTIONS (CONTINUED)

Byte Definition		Hardware Options (HP)	Device Status (DS) 65X	Extended Status (ES) 65X
Dec.	Oct.			
9	11	Blank tape	End-of-operation	End-of-operation
10	12	Last block EOR/EOF	Alert	Alert
11	13	Last operation write	Tape unit reserved	Cold start

TABLE 13-1. MAGNET PROCESSING OPTIONS (CONTINUED)

Byte Definition		General Status 66X/67X		
Dec.	Oct.			
0	0	Unit ready		
1	1	Unit busy		
2	2	Load point		
3	3	End-of-tape		
4	4	Tape mark detected		
5	5	Odd count		
6	6	Unit type 0 = 7-track, 1 = 9-track		
7	7	Write ring		
8	10	Noise detected		
9	11	No unit		
10	12	Coupler status (not used lower CYBER)		
11	13	Alert write	reserved	

RESIDENCY OF 1MT

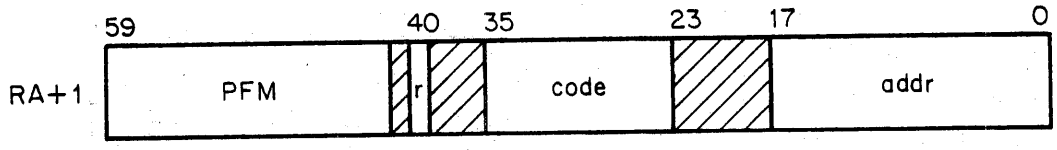
The following is a suggested order of priority for making routines CM or alternate library residence.

1. Function reject processor (3MB)
2. Control point/coded preset (3MH)
3. Read function processor (3ML)
4. Write function processor (3MT)
5. Drop PPU processor (3MG)
6. 1MT itself
7. Complete user FET (3MI)
8. Read label processor (3MN)
9. Read/write error recovery overlays, 3MR and 3MW, for 65x units and/or 3MS and 3MX for 66x/67x units

Permanent files are controlled by PP routine permanent file manager (PFM). All requests for permanent file action are accompanied by a specific user number. User numbers are established by the installation and entered into the system validation file VALIDUs. Each user number then maps to a user index. Only validated users may request permanent file action.

PFM COMMUNICATION

PFM is called using the following RA+1 request.



r Auto recall (bit 40, must be set)

code Command code (request):

Symbol	Octal Value	Command
CCSV	01	SAVE
CCGT	02	GET
CCPG	03	PURGE
CCCT	04	CATLIST
CCPM	05	PERMIT
CCRP	06	REPLACE
CCAP	07	APPEND
CCDF	10	DEFINE
CCAT	11	ATTACH
CCCG	12	CHANGE

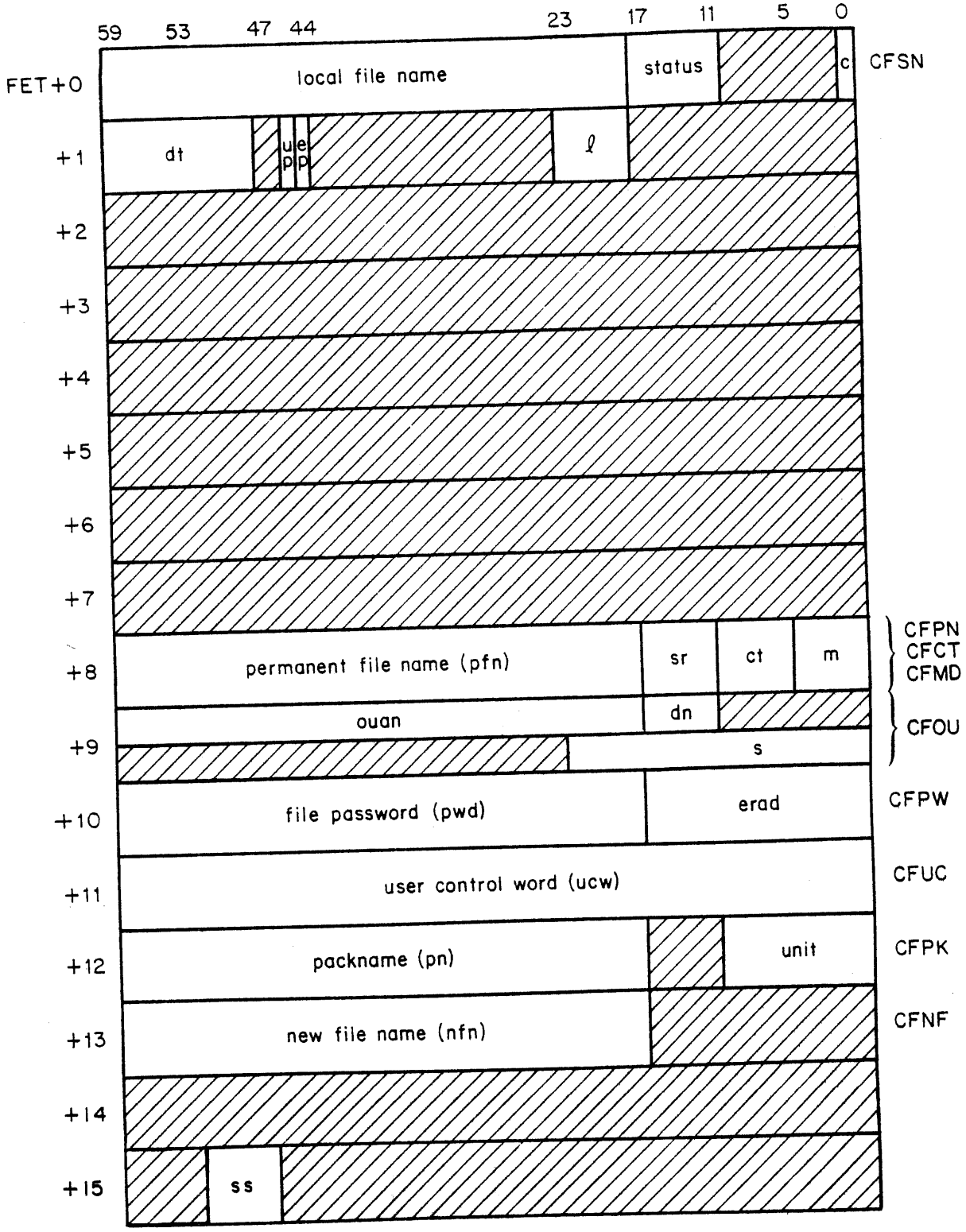
addr Address of the call block or FET

The code parameter is biased by 2000B if the default pack name is to be ignored or by 1000B if the default family is to be used.

The RA+1 call is either generated directly by the user or is generated by a PFM macro or control statement. Macros that use PFM are described in volume 2 of the NOS Reference Manual. PFM control statements are described in volume 1 of the NOS Reference Manual.

A PFM control statement causes CPU routine PFILES to be loaded in the user's FL. PFILES issues the PFM macro calls which result in an RA+1 request for PFM.

The FET used by all PFM requests is of the following form.


59
53
47
44
23
17
11
5
0
FET+0
+1
+2
+3
+4
+5
+6
+7
+8
+9
+10
+11
+12
+13
+14
+15
local file name
status
c
CFSN
dt
u
p
e
p
ℓ
permanent file name (pfn)
sr
ct
m
CFPN
CFCT
CFMD
ouan
dn
CFOU
s
file password (pwd)
erad
CFPW
user control word (ucw)
CFUC
packname (pn)
unit
CFPK
new file name (nfn)
CFNF
ss

status — Error codes are returned in bits 17-10.

c — Bit 0 is set to one upon completion of the request.

dt — Device type of file residence. If not specified, system default is used (DI for auxiliary device requests; any available device for others).

up — User processing bit (bit 45). If set, control is returned on device unavailable errors.

ep — Error processing bit (bit 44). If set, control is returned to the user on errors.

l — FET length minus 5.

pfn — Permanent file name. If zero, lfn is used.

sr — Special request:*

Symbol	Description
SRFA	Fast attach
SRDN	CATLIST of specific device
SRCE	Clear error status
SRNF	Force non-fast attach

ct — File category:*

Symbol	Description
FCPR	Private
FCSP	Semiprivate
FCPB	Public

m — File access mode:*

Symbol	Description
PTWR	Write
PTRD	Read
PTAP	Append
PTEX	Execute only

* Symbols defined in common deck COMSPFM.

Symbol*	Description
PTNU	Null
PTMD	Modify
PTRM	Read/modify
PTRA	Read/append
PTNM	Nonroll modify (IAUM only)

ouan** Alternate user number.

dn Device number for CATLIST function.

s** Number of PRUs required for the direct access permanent file being defined. This ensures that s PRUs are available when the file is defined, but does not necessarily ensure they will be available to write (that is, they are not preallocated).

pwd Optional file password.

erad Address where error messages are returned. The message may be up to three words long and is stored at the given address only if ep is set.

ucw Program control word. If bit 59 is set, whatever the user stores in this word is stored in the catalog entry when a permanent file is created, or a CHANGE, REPLACE, or APPEND is performed on it. This word is read from the catalog entry and stored in CFUC when the file is attached. (Bits 14 through 12 contain the time-sharing subsystem code.)

pn Name of the auxiliary device to be used in satisfying the permanent file request.

unit The number of units of the type specified by dt. For example, if the device type is DI4, the dt field contains DI and the unit field contains 4. If dn is nonzero and unit equals zero, then a default of 1 is used.

nfn New file name used with the CHANGE command.

ss Subsystem

---

* Symbols defined in common deck COMSPFM.
** Mutually exclusive fields; FET may contain either but not both fields.

PERMANENT FILE TYPES

There are two types of permanent files available to users of NOS: direct and indirect access files.

- A direct access permanent file is read and written by user I/O requests just as any local file would be read or written. Large data files occupy large amounts of mass storage and are normally created as direct access files. (Direct access files are allocated by logical tracks.)

- An indirect access permanent file is accessed by using a working copy of the file rather than the file itself. The working copy is attached as a local file to the user job. Thus, modifying the working copy does not alter the actual permanent file. Indirect access files are allocated in sectors and are generally used for small permanent files.

A direct access permanent file is normally declared by the user prior to writing the file by using the DEFINE control statement or macro. The DEFINE may be issued after the file has been written but, in this case, the file must reside on a device available to the user for permanent files. Indirect access permanent files are declared by the SAVE or REPLACE control statements or macros after the file has been created.

Whenever a permanent file is declared, the user index is mapped into a catalog track where permanent file names and statistics for that user are maintained. There is one catalog entry for every permanent file known to the system. A catalog track normally contains entries for several different users. A description of this mapping is provided in the NOS System Maintenance Reference Manual.

A family consists of 1 to 63 mass storage devices. Within a family, each user has a master device that contains his permanent file catalogs, all of his indirect access files, and some or all of his direct access files. Again, the mapping of a user index into a master user within the family is shown in the NOS Installation Handbook.

If more than one family is available in the system, the user must specify which family via the USER control statement. Although a family may have up to 63 devices, the number of mass storage devices in the configuration is still limited by installation parameter NMSD.

A user may allow other users to access his permanent files with the PERMIT control statement or macro, which results in adding an entry to the permit chain.

Direct access permanent files have the following characteristics.

- Allocated by logical tracks and, therefore, normally used for large files.
- When accessed, all users interact with the same (only) copy of the file.
- Write interlock.
- Multiread capability (multiple FNT/FST)
- Created/accessed with DEFINE and ATTACH macros or control statements.
- Fast attach capability.

Indirect access files have the following characteristics.

- Allocated by sectors and, therefore, normally used for small files.
- When accessed, each user interacts with his own copy of the file.
- Created via SAVE or REPLACE macro or control statement. Accessed by OLD control statement or GET macro or control statement. Information can be added to an indirect access file with the APPEND control statement.

For direct and indirect access files the user can specify one of the following permission modes for other users who access the files.

- Read
- Execute only
- Write
- Append
- Modify
- Null
- Read/modify
- Read/append

In addition, there are two classes of permission: implicit and explicit. Explicit permission is granted to the file by the owner of the file via a PERMIT control statement or macro. Implicit permission is automatically granted for semiprivate files. Refer to volume 1 of the NOS Reference Manual for a complete discussion of permission for permanent files.

USER NUMBERS CONTAINING ASTERISKS

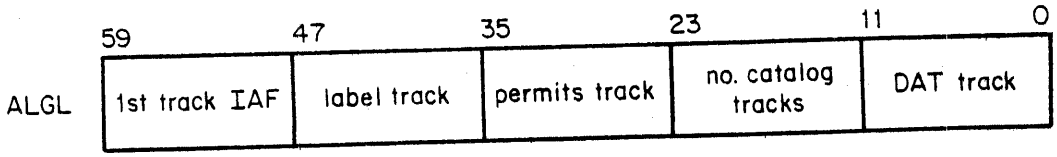
User numbers that contain asterisks represent users with automatic read-only permission to files in other user catalogs. The user number must match the alternate user number in all characters not containing asterisks.

For example, user number ABC* can access all the permanent files of ABC1, ABC2, ABCS, and any 4-character user number whose first three characters are ABC.

MASTER DEVICES

When a user requests permanent file activity, PFM uses the user index of the running job to map to that particular user's master device and catalog track.The algorithm for this mapping is in the NOS System Maintenance Reference Manual.

Each master device has a predetermined number of catalog tracks. The MST word ALGL contains permanent file information in the following format.



Byte 3 contains the actual number of tracks that are catalog tracks. (The catalog tracks are usually contiguous tracks.) Byte 1 is the first track of the label track chain. This track chain consists of the label and all the catalog tracks. Normally the label track is track 0; however, if track 0 is flawed, then the first available track is used. The preserved bit is set in the TRT so that this track chain is preserved across deadstarts. Normally track 1 is the first track of the catalog tracks. Only master devices have catalog tracks, so for a nonmaster device the label track chain consists of only one track.

If all the catalog tracks cannot be contiguous, then the next available tracks are used for the catalog, and in this case bit 17 is not set in MST word PUGL. The first catalog track is pointed to by the TRT link from the label track position.

Since each user is mapped to a particular catalog track, and these tracks are contiguous, the link byte in the last sector does not link to the next track in the chain. If this track becomes full, catalogs cannot overflow to the next contiguous track (other users are mapped to that track). A new track is linked into the chain via the TRT table, and the last sector link byte points to this new track. PFM then has increased the length of this catalog track. This slows PFM when it has to search more than one catalog track for any one user. So bit 16 is set in PUGL and an 0 is displayed in the E,M display to indicate catalog overflow.

Many users can be mapped to any specific catalog track, but no user can be mapped to more than one catalog track (in case of overflow, it is considered a very long track).

As a user creates permanent files, the entry shown in figure 14-1 is placed in the appropriate catalog track and the file is processed by PFM.

DIRECT ACCESS FILE PROCESSING

Direct access files are processed as follows. If the file resides on a device in the user's family which can contain permanent files, the entry is created, the first track is recorded, and the first sector entry is set to 4000B denoting a direct access permanent file. Since this is a regular file, sector 0 contains the system sector and sector 1 is the first sector of data. PFM issues the STBM function to set the preserved bit in the TRT. If the file does not reside on such a device, the job is aborted unless error processing was desired. In order to avoid this possibility, the user should define the file prior to writing on it.

If the file does not exist, PFM creates the file on a device in the user's family that can contain permanent files. PFM also creates a catalog entry and builds the appropriate system sector.

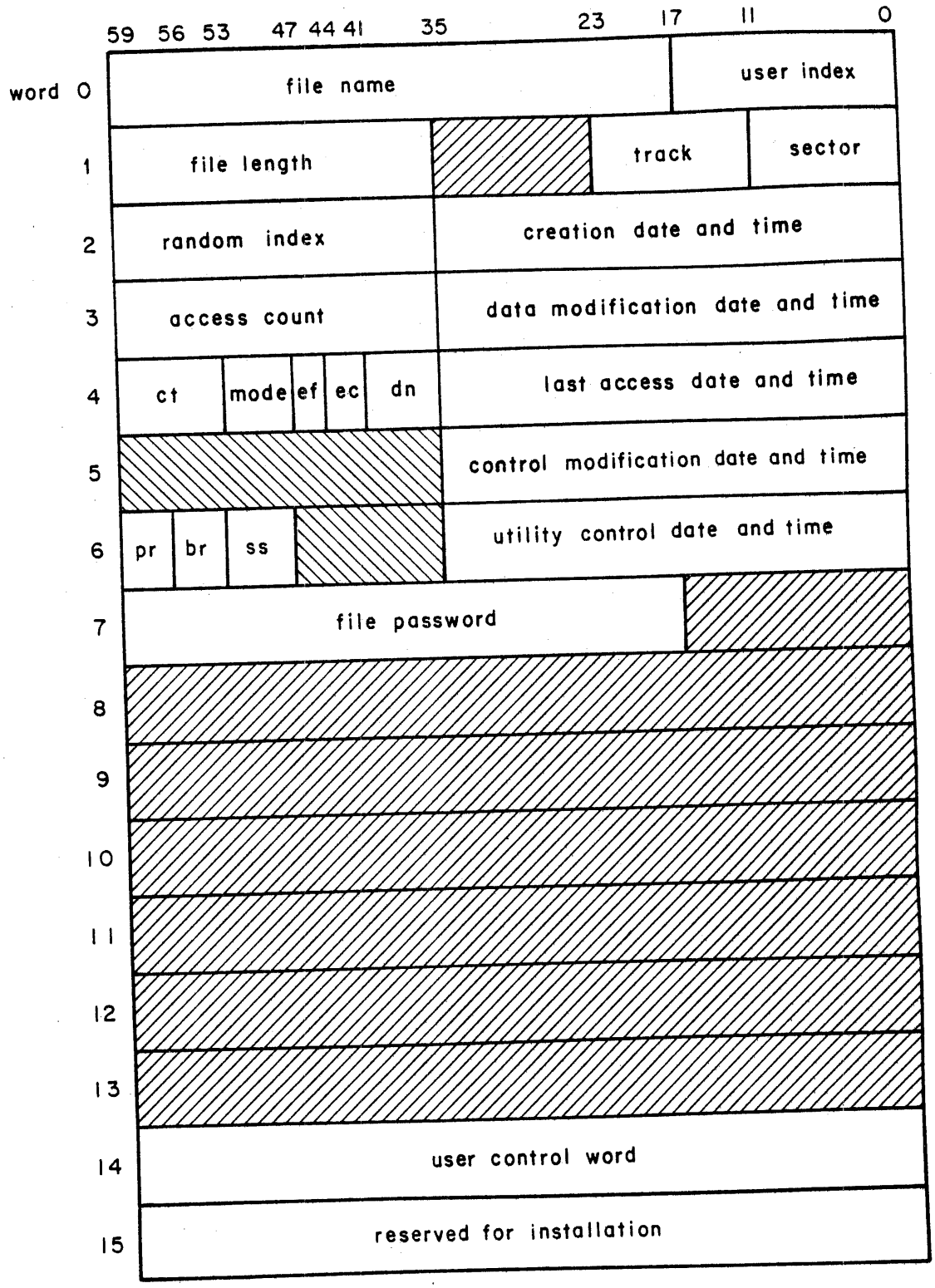


Figure 14-1. Permanent File Catalog Entry

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
creation date and time	Date and time (yymmddhhmmss in octal) when this file was first entered on the permanent file system. The year (yy) is biased by 70.
access count	Count of accesses to file
data modification date and time	Date and time (yymmddhhmmss in octal) when the data in this file was last modified. The year (yy) is biased by 70. For direct access files this field is updated only when the file is attached in a modifiable mode.
ct	File category:*

Symbol	Description
FCPR	Private
FCSP	Semiprivate
FCPB	Public

mode — Mode of access for semi-private and public files:*

Symbol	Description
PTWR	Write
PTRD	Read and/or execute
PTAP	Append
PTEX	Execute
PTNU	Negate previous permission
PTMD	Modify

Figure 14-1. Permanent File Catalog Entry (Continued)

* Symbols defined in common deck COMSPFM.

	PTRM	Read, allow modify
	PTRA	Read, allow append
ef	Error flags:	
	40	File error when purged (error code in ec field)
ec	Error code:	
	0	No error
	1	Error in file data
	2	Error in permit data
	3	Data/permit errors
	4	EOI changed by recovery
	5-7	Reserved
dn	Device number (0 through 77B); each device within a family of permanent file devices is identified by a device number	
last access date and time	Date and time (yymmddhhmmss in octal) when this file was last accessed. The year (yy) is biased by 70.	
control modification date and time	Date and time (yymmddhhmmss in octal) when the control information (catalog entry and permit record data) for this file was last updated. The year (yy) is biased by 70.	
pr	Preferred residence.	
br	Backup residence.	
ss	Subsystem code for this file:	
	0	NULL subsystem
	1	BASIC subsystem
	2	FORTRAN subsystem
	3	FTNTS subsystem
	4	EXECUTE subsystem
	5	BATCH subsystem

Figure 14-1. Permanent File Catalog Entry (Continued)

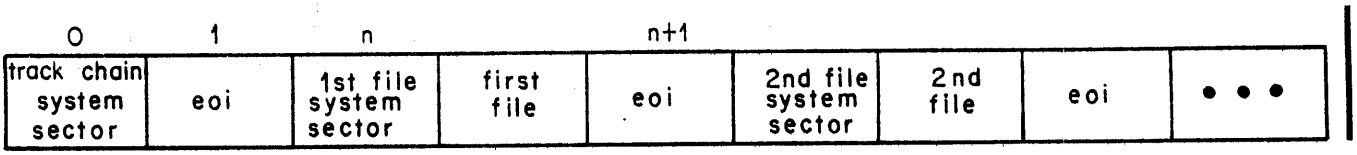
utility control date and time	Date and time (yymmddhhmmss in octal) used to determine this file's candidacy for being dumped by permanent file utilities. The year (yy) is biased by 70.
file password	Optional password
user control word	User control information (FET+11)

Figure 14-1. Permanent File Catalog Entry (Continued)

INDIRECT ACCESS FILE PROCESSING

If the permanent file is an indirect access file, then the entry is copied from the file that is to be made permanent; that is, indirect access files are allocated by PFM and the system does not keep track of them. The user must create the file first, and then issue the SAVE or REPLACE control statement.

PFM keeps an indirect access file track chain. This chain is reserved from the system as a normal file chain and the preserved bit is set to preserve it over deadstarts. Word ALGL byte 0 points to the first track of this chain. The chain is kept to a minimum length when possible, and is expanded (RTCM) and contracted (DTKM or DLKM) as necessary. However, the indirect access file track chain must completely reside on its master device since evey user mapped to the master device must have all of his indirect access files on this device. The format for the file is as follows. Sector 0 is the system sector of the indirect access file chain and sector 1 is an EOI. Sector 2 contains the system sector for the first indirect access file. Sector 3 contains data for the first indirect access file.



FILE CREATION, DELETION

As each SAVE or REPLACE function is processed, the PFM indirect access file track chain gets n contiguous sectors (the length of the user's file including the system sector) on the indirect access file chain. It copies the user's file including the system sector and the EOI. The number of sectors copied, not counting the EOI sector, is saved in the catalog entry as well as the first track and sector number of the file. Sector 4000B does not exist so there is no confusion between direct access and indirect access file entries in the catalog entries.

As more files are saved and defined, the catalog entries grow and could cause overflow as described earlier. However, available slots in the catalog entries are created by purges of permanent files. These available slots are known as holes.

When a direct access file is purged, the user index is set to zero, and all the tracks in that file chain are released to the system. These tracks can be used by the system for any purpose.

When an indirect access file is purged, its user index is set to zero; however, the sector count field is left intact. The sectors are not released physically unless the file was so large it spanned one or more whole tracks. In which case the tracks are returned to the system (DLKM) and the sector count field is set to the remaining sectors. These sectors can only be used by new indirect access files.

In the case of the REPLACE command, the following steps occur:

1. If the new file is the same size as the existing file, the new file is copied over the old indirect access file.

2. If the new file is smaller than the existing file, the new file is copied over the old one, the sector count field is modified, and a new permanent file catalog entry is built. This entry has a user index of zero, sector count field set to the remaining sectors, and first track and sector pointing to the remainder of the old file (that is, a hole). This is not done if the hole size equals two sectors.

3. If the new file is larger than the old file, the current entry becomes a hole (user index zero). A new hole is found if one big enough exists, or the new file is placed at the end of the indirect access files and a new catalog entry is used.

Hole searching is accomplished the same way for SAVE and REPLACE commands. Only the catalog track (plus overflow tracks, if any) mapped to by the user index of the user are searched. All catalog tracks are never completely searched. The search proceeds as follows:

1. If a hole with the exact number of sectors available is found, it is used.

2. If not 1, then the largest hole larger than the file is used.

3. If not 1 or 2, then the file is put at the end of the indirect access files and a new entry is used.

Eventually many holes result and a PFDUMP, INITIALIZE, and PFLOAD are required to close the holes. PFLOAD will recreate the catalog entries and indirect access files with no holes.

ACCESSING FILES

When a GET function is issued, PFM finds the entry, copies the file from the indirect access file to a local file (LOFT), and creates an FNT/FST entry for this local file with the proper permission bits set. PFM counts the sectors copied (exclusive of the EOI) and compares them with the sector count field, and, if they do not agree, it issues a file length error. The same procedure is used for the OLD command, except the local file is of type PTFT.

When an ATTACH function is issued, PFM finds the catalog entry and creates an FNT/FST pointing to this file. The file type is PMFT and the permission bits are set accordingly.

When an ATTACH function is issued by an SSJ= job, PFM first checks to see if a fast attach of the file can be performed. If the SRNF special request (force non-fast attach) is not specified, PFM searches the system FNT/FST for an FAFT entry with the proper file name and family. If such a file is found, PFM bypasses the catalog search and permits checking and creates an FNT/FST (type PMFT) pointing to the file.

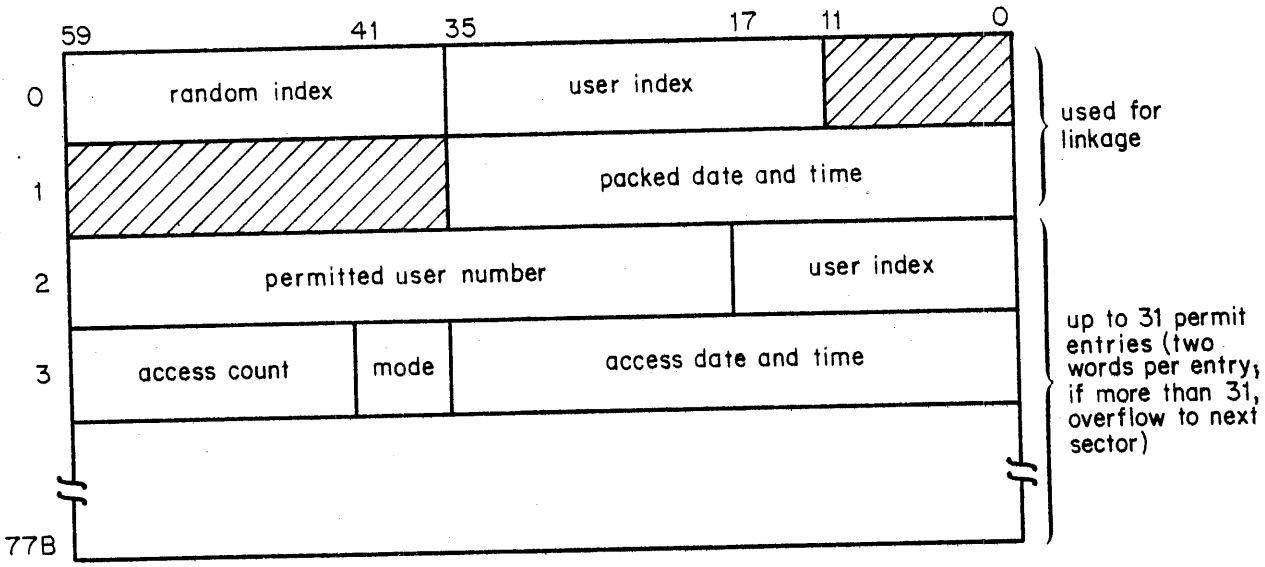
CATALOG/PERMIT ENTRIES

On all non-fast attach functions, PFM maps to the proper catalog track, finds the catalog entry for the file, and ensures that the user has permission to use the file. Explicit permission is granted by the owner of a private or semiprivate file through a PERMIT function. Implicit permission is available for accessing semi-private or public files from all alternate user numbers. The original owner always has permission to use the file.

For a private or semiprivate file, PFM first checks for an explicit permit. Byte 2 of word ALGL points to the first track of the permits track chain. The catalog entry for the file points to the first sector within the chain which contains permit entries for the file. Each of these entries indicates the permission mode available to a single user number. If an explicit permit for the user is found, permission is granted according to the mode specified in the permit entry.

For a public file, or a semiprivate file for which no appropriate explicit permit is found, permission is granted according to the mode specified in the catalog entry for the file.

The format of the permit entry is as follows.



random index	Disk address of next PERMIT buffer for this user index. Zero indicates end of chain.
user index	User index who created this sector.
access count	The number of times the permitted user has accessed the file.
mode	Mode of permitted access (defined in COMSPFM; refer to catalog track entry described earlier in this section). If bit 41 is set, indicates accounting permit entry; if bit 41 is zero, indicates explicit permit entry.
access date and time	Time and date of last access by permitted user.

Table 14-1 illustrates the relationship between the requested access mode and the mode specified in the catalog or permit entry. A y entry indicates permission is allowed; n indicates permission is not allowed.

TABLE 14-1. MODE RELATIONSHIPS

Mode Requested	MODE IN CATALOG/PERMIT							
	PTWR	PTRD	PTAP	PTEX	PTNU	PTMD	PTRM	PTRA
PTWR	y	n	n	n	n	n	n	n
PTRD	y	y	n	n	n	y	y	y
PTAP	y	n	y	n	n	y	n	n
PTEX	y	y	n	n	n	y	y	y
PTNU	n	n	n	y	n	n	n	n
PTMD	y	n	n	n	n	y	n	n
PTRM	y	n	n	n	n	y	y	n
PTRA	y	n	n	n	n	y	y	y

Table 14-2 illustrates the operations involved for each PFM function.

TABLE 14-2. PFM FUNCTIONS AND PROCESSES

Operation	PFM Function				
	* SAVE	* GET	PURGE	CATLIST	PERMIT
Catalog search for hit	X	X	X	X	X
Update PFC entry		X	X		0
Catalog search for ID hole	X				
Update hole	X				
Catalog search for DA hole					
Permit search for hit		0	0	0	0
Update permit hit		0	0		0
Permit search for hole		0			0
Update hole		0			0
System sector update					#
EOI update					
Catalog search to end	X			X	
Permit search to end					0
BOI/EOI verify					

* Operations restricted to indirect access files
X Always
0 Not always
Direct access files only

TABLE 14-2. PFM FUNCTIONS AND PROCESSES (CONTINUED)

Operation	PFM Function				
	* REPLACE	* APPEND	DEFINE	ATTACH	CHANGE
Catalog search for hit	X	X	X	X	X
Update PFC entry	X	X			X
Catalog search for ID hole	X	X			
Update hole	X	X			
Catalog search for DA hole			X		
Permit search for hit	0	0		0	
Update permit hit	0	0			
Permit search for hole	0	0		0	
Update hole	0	0		0	
System sector update			X	X	#
EOI update			0		
Catalog search to end	X	X			X
Permit search to end					
BOI/EOI verify			0	0	

* Operations restricted to indirect access files
X Always
0 Not always
Direct access files only

PFM STRUCTURE

Routines called by PFM include the following:

OAV	User verification
OBF	Begin file
ODF	Drop file
ORF	Update resource files

PFM consists of a few resident subroutines and the following overlays:

Overlay	Description
3PA	Command processor
3PB	SAVE and REPLACE command processing

Overlay	Description
3PC	APPEND command processing
3PD	ATTACH processing
3PE	Catalog list routines
3PF	DEFINE processing
3PG	PERMIT/PURGE processing
3PH	Error processing
3PI	Auxiliary routines
3PJ	CHANGE processing
3PK	Device-to-device transfer
3PL	APPEND, original file transfer
3PM	DEFINE auxiliary routines

Figure 14-1 shows the allocation of PPU memory for PFM overlays. There are six load addresses used for 3Px overlays.

Load Address	Definition
OVLA	End of PFM resident. Used as load address for overlays not requiring common catalog/ permit manipulation routines in 3PA.
OVLC	End of general command processing subroutines in 3PA. Used as load address for command level overlays that require catalog/permit routines.
BFMS	SYSTEXT symbol used for auxiliary overlay load address and mass storage I/O.
LOCG	End of 3PA resident. Load address for overlays requiring termination processing in 3PA, but not catalog/permit routines.
LOCI	Maximum of BFMS and LOCF + ZDFL where LOCF is the address to load zero-level overlays in 3PA. Load address for auxiliary overlays that also need to call zero-level overlays at LOCF.
BUF	End of 3PK. Start of storage buffer for device-to-device transfer.

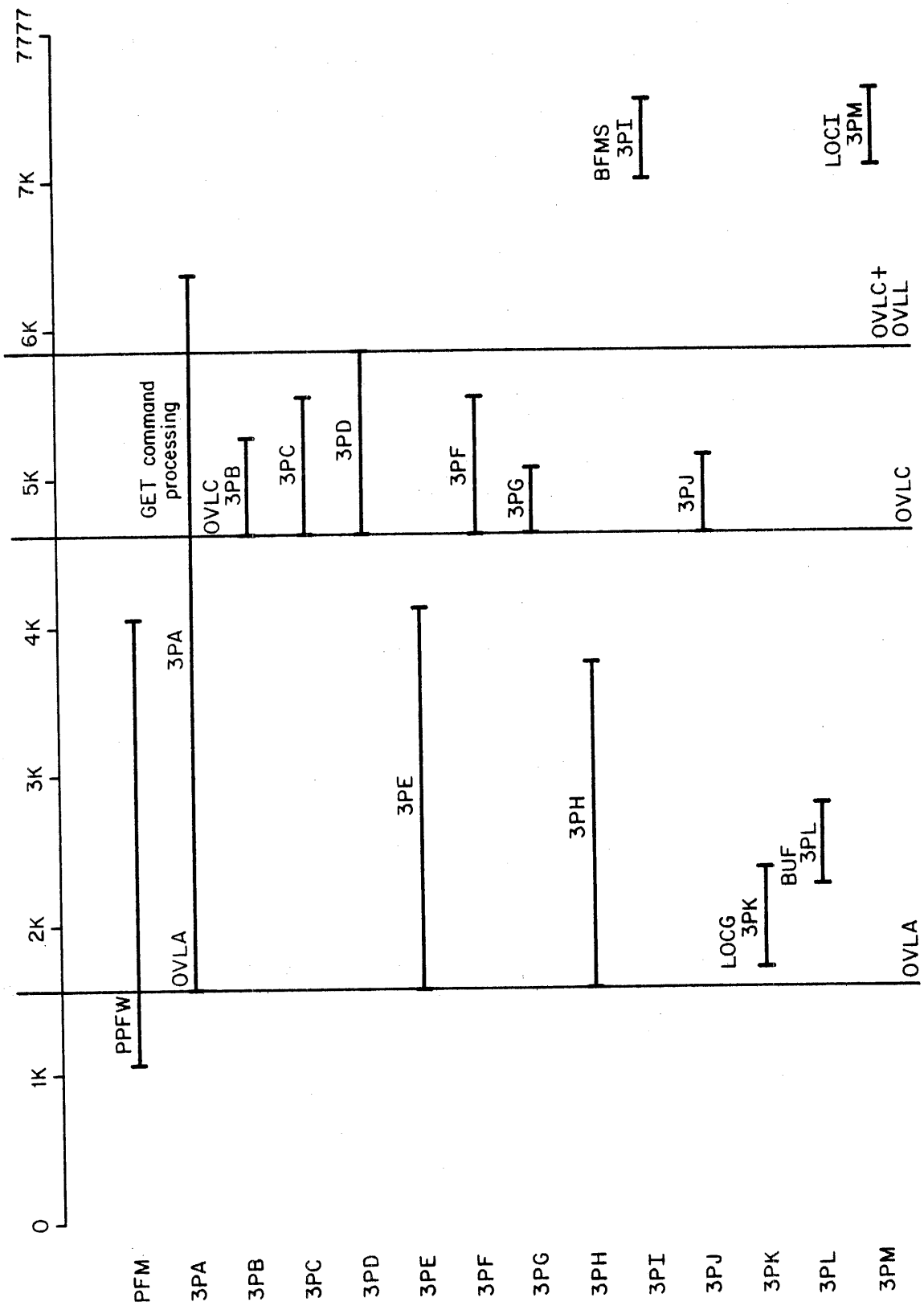


Figure 14-1. PFM Overlay Load Map

ROUTINE PFM

PFM provides storage for the call block and other temporaries. Its resident subroutines are as follows.

Subroutine	Description
SFA	Set FET address
CCI	Clear catalog interlock
DPP	Drop PPU
ERR	Process error
SFN	Set file name in CM
MSR	Mass storage read error processor

PFM presets processing routines to:

- Verify FET parameters
- Verify user validation allowances
- Place request in recall if catalog is interlocked
- Issue accounting messages
- Load proper function processor overlay (3PA or 3PE)
- Call RESEX if pack is unavailable

3PA - MAIN COMMAND PROCESSING

Routine 3PA performs all processing required to perform the GET function. It performs preliminary processing for the following commands:

SAVE APPEND

ATTACH	DEFINE
GET	CATLIST
PURGE	REPLACE
PERMIT	CHANGE

3PA also contains:

- Catalog processing routines
- PERMIT processing routines (some)
- File allocation routines
- General subroutines

The following outline describes 3PA subroutines and buffers in more detail. Many of the 3PA subroutines are called from other 3Px overlays. Those called from another overlay are labeled with an asterisk. Overlay 3PA resident routines include *TRP (terminate program). Resident common decks include *COMPSNT (set next track) and *COMPRNS (read next sector). Next, *LOCG, device-to-device transfer buffer, overlays subsequent subroutines.

PERMIT subroutines include the following.

Subroutine	Description
CPE	Create PERMIT entry
*CPI	Check permission information
*UPI	Update permission information
CSA	Compute sector address
SPI	Search permission information
PPE	Process PERMIT read error
WNP	Write new PERMIT buffer
FPE	Form PERMIT entry in buffer

Catalog processing subroutines include the following.

Subroutine	Description
*CCS	Create catalog sector
*DCE	Delete catalog entry
FCE	Form catalog entry
FHE	Form hole entry
*UCE	Update catalog entry
*SSC	Select catalog entry
*SCH	Search catalog
PCE	Catalog READ error processor
CCD	Check catalog data

Allocation subroutines include:

Subroutine	Description
AFS	Allocate file space for indirect file
ACS	Allocate catalog space
APS	Allocate PERMIT space
CML	Check mass storage limits against validation limits

General subroutines include:

Subroutine	Description
*DTK	Drop tracks
*ITC	Interlock track chain
*ITF	Interlock file
*CFA	Check fast attach file
*RTK	Request linked track
*WBI	Write buffer in place

Common decks include the following.

Common Decks	Description
COMPCRA	Convert random address
*COMPSEI	Search for end-of-information
*COMPCTI	Clear track interlock
*COMPSTI	Set track interlock
*COMPCKP	Set checkpoint bit in EST entry

OVLC command processing overlays are loaded next and destroy following subroutines. These overlays must not exceed OVLL. OVLL is defined to be OVLC plus one full PRU plus one short PRU. GET and ATTACH processing routine and the command processing initialization (SET) follow.

Catalog search initialization subroutines include the following.

Subroutine	Description
*ISP	Initialize search
*SPN	Set permanent file name
*COMPIRA	Initialize random access processors
*COMPFAT	Search for fast attach file
*COMPSAF	Search for assigned file
*COMPSFB	Set file busy

3PA calls many of the other 3Px overlays. Those called are shown in table 14-3.

TABLE 14-3. OVERLAYS 3Px CALLED BY 3PA

Overlay Name	Load Address	3PA Subroutine Called From	Command Processed
3PI	BFMS	ISP	
3PB	OVLC	SET	SAVE/REPLACE
3PJ	OVLC	SET	CHANGE
3PF	OVLC	SET	DEFINE
3PG	OVLC	SET	PURGE/PERMIT
3PD	OVLC	SET	ATTACH
3PC	OVLC	SET	APPEND
3PK	LOCG	GET	GET

3PB - SAVE/REPLACE PROCESSING

The 3PB overlay contains subroutines for processing the SAVE and REPLACE commands. It also contains the following subroutines.

Subroutine	Description
REP	Process REPLACE command
SAV	Process SAVE command
CUC	Check user controls
PFR	Process file replacement
SSP	Set statistical parameters
CFS	Check file size

Overlays 3PK and 3PI are called from 3PB.

3PC - APPEND PROCESSOR

The 3PC overlay contains the following subroutines for processing the APPEND command.

Subroutine	Description
APP	Process APPEND command
AFS	Allocate file space
CUC	Check user controls
SSP	Set statistical parameters
CFS	Check file size

Overlays 3PK, 3PI, and 3PL are called from 3PC.

3PD - ATTACH PROCESSOR

Overlay 3PD is called from subroutine GET in overlay 3PA to process the request to attach a direct access file to a job. 3PD consists of the following subroutines.

Subroutine	Description
ATT	Process ATTACH command main program
CFM	Check file mode
CLU	Clear local user table
CSL	Check size limits
DLT	Determine local user table to update
IMS	Initialize mass storage
MSS	Read system sector error processor

Common decks in 3PD include COMPRSS for reading system sectors and COMPWSS for writing system sectors.

3PE - CATALOG LIST ROUTINES

Overlay 3PE is called from the preset subroutine, PRS, in the main program, PFM. Routine 3PE is loaded at OVLA and is called to read permanent file catalogs for a central processor program. Data is returned to the CM buffer specified by the FET pointers: FIRST, IN, OUT, and LIMIT. Refer to the CATLIST macro in volume 2 of the NOS Reference Manual for a complete description of the use of the FET pointers for this function.

Overlay 3PE consists of the following subroutines.

Subroutine	Description
CAT	Main program
NCS	Normal catalog search (mode=0)
ACS	Alternate catalog search
PDS	PERMIT data search
SBS	Set status of FET (update IN)
RBS	Read buffer for search
MSR	Mass storage read error processor
SHB	Search catalog buffer
WDB	Write buffer
CCP	Check catalog permission (clear password)
DFS	Determine file size (store in catalog entry)
SPB	PERMIT buffer search
CSA	Compute sector address
CSU	Check for special user
ISP	Initialize search of catalog with common deck COMPSCA (set catalog address)

Common decks include:

Common Deck	Description
COMPCRA	Convert random address
COMPSRA	Set random address
COMPSEI	Search for end-of-information
COMPRNS	Read next sector
COMPSDN	Search for device number
COMPSCA	Set catalog address

3PF - DEFINE PROCESSOR

Overlay 3PF is called to create a direct access permanent file. The file exists prior to the DEFINE command, or the file may be created after the DEFINE command. File residency is determined in 3PF for the two situations as follows:

- Local file exists. The local file is made permanent if the local file resides on a permanent file device to which the user has access (as determined by the secondary mask for the device); otherwise, the request is aborted. The dt field of the call block is ignored. If the local file resides on a removable device, that device's pack name must be the same as the pack name specified in the call block.

- No local file. If the dt field is zero, the file is placed on the device with the most available space. If dt is specified, the file is placed (started) on the device of that type with the most available space. If np (number of PRUs) is specified, the file is placed on the device (type dt, if specified) with the most available space, provided that np PRUs are available. If np PRUs are not available, the request is aborted with the message PRUS REQUESTED UNAVAILABLE.

Routine 3PF writes a system sector for the file to reflect the permanent file status of the file. The catalog entry is stored in the system sector as indicated in the format below. However, note that byte 2 of word 1 is updated to indicate the current access modes. The current access mode values are as follows:

7 Write

3 Modify

1 Append

0 Read

The format of the system sector is shown in section 2.

Overlay 3PF consists of the following subroutines:

Subroutine	Description
DEF	Main routine to build catalog entry and write system sector
CUC	Check for maximum number of files reached
PFE	Process file error
MSS	System sector error processing
DFR	Determine file residency
CPR	Check for proper family or pack name residency
DDN	Determine device name from MST entry
SFT	Set FNT/FST information
SUC	Set user counts in system sector

Common decks include COMPRSS, COMPWSS and COMPWEI.

3PG - PERMIT/PURGE PROCESSOR

Overlay 3PG contains routines to process PERMIT and PURGE requests. 3PG consists of the following subroutines.

Subroutine	Description
PUR	Process PURGE request
PER	Process PERMIT request

Subroutine	Description
DDF	Delete direct access file
UEP	Update existing permit entry
USS	Update system sector
MSS	Read system sector error processing

Common decks include COMPRSS (read system sector) and COMPWSS (write system sector).

3PH - ERROR PROCESSING ROUTINES

Overlay 3PH contains the error processing routines for all other overlays. It performs the following:

- Sends the indicated error message to the dayfile.
- Sets the FST entry not busy or deletes the FNT/FST entry if created by PFM.
- Terminates the calling program if user error processing is not specified.
- Drops the PPU.

Overlay 3PH contains a list of error messages issued by PFM. These messages are listed in the NOS Reference Manual, volume 1. Some messages are sent to the control point dayfile while others are sent to the error log.

3PI - AUXILIARY ROUTINES

Overlay 3PI contains auxiliary routines used by many of the other 3Px overlays. These auxiliary routines can be overlayed after execution by any process that uses BFMS since 3PI is loaded at BFMS. Overlay 3PI contains subroutine search for system file (SSF). Currently, 3PI contains two common decks:

COMPSCA	Set catalog address
COMPSDN	Search for device number.

3PI must not overflow into the error processor and therefore must be no more than one PRU in length.

3PJ - CHANGE PROCESSOR

Overlay 3PJ processes the CHANGE command by changing and replacing the catalog entry for a file. 3PJ contains the following subroutines.

Subroutine	Description
CHG	Process CHANGE request
CCE	Change catalog entry
CFN	Check file names
PCE	Catalog read error processor
SCT	Search catalog
SFL	Set file names
USS	Update system sector
MSS	Read system sector error processor

3PK - DEVICE-TO-DEVICE TRANSFER

Overlay 3PK processes device-to-device transfers for those PFM operations that deal with indirect access permanent files. Overlay 3PK contains the following subroutines.

Subroutine	Description
DTD	Device-to-device transfer
DPC	Decrement PRU counter
PTE	Process transfer error
IBA	Increment buffer address
SDP	Swap disk parameters
WES	Write EOI sector
WNS	Write next sector
CEI	Create EOI sector

3PL - APPEND - ORIGINAL FILE TRANSFER

Overlay 3PL accomplishes the actual writing of the permanent file device with the appended permanent file. The order of transfer is old permanent to new permanent file, then the system (local) file to new permanent file.

Overlay 3PL contains the subroutines APPEND disk transfer (ADT) and process APPEND error (PAE).

The following common decks are included in 3PL.

Common Deck	Description
COMPCTI	Clear track interlock
COMPSTI	Set track interlock
COMPCKP	Set checkpoint bit in EST

Overlay 3PL is called only by overlay 3PC.

3PM - DEFINE AUXILIARY ROUTINE

Overlay 3PM contains subroutine DRF - determine file residency - which is used by overlay 3PF when processing the DEFINE command. Overlay ODF is executed by 3PM.

INTRODUCTION

TELEX is a subsystem that provides support for interactive processing from remote terminals such as TTYs (teletypewriter terminals) and CDC 713s. The subsystem consists of the following CPU and PP programs.

Program	Description
TELEX	Time-sharing executive initialization routine. This routine is loaded at 40000B relative to control point 1 when the operator types TELEX. It initializes tables and pointers and loads TELEX1.
TELEX1	Time-sharing executive processor. This is the main routine that processes I/O for the TTYs. It cracks and processes commands, and makes requests to dump source input to disk and refill output buffers from disk. It communicates with the transaction facility (at another control point) to support transaction terminals.
TELEX2	Time-sharing executive termination routine. This routine is executed after an abnormal condition is detected or when the operator terminates TELEX with 1.STOP.
1TA	TELEX auxiliary function processor. This routine processes functions for TELEX which require PP action.
1TD	Terminal communications driver low-speed interactive (600 baud or less). It performs communications between TELEX and terminals (accessed via the 6671 and 6676 multiplexers). It also communicates between TELEX and the NOS stimulator (checkout/test).
1TO	Terminal input/output. Called by TELEX to perform terminal I/O requiring disk accesses.

The relationship between the various system routines and subsystem routines is shown in figure 15-1.

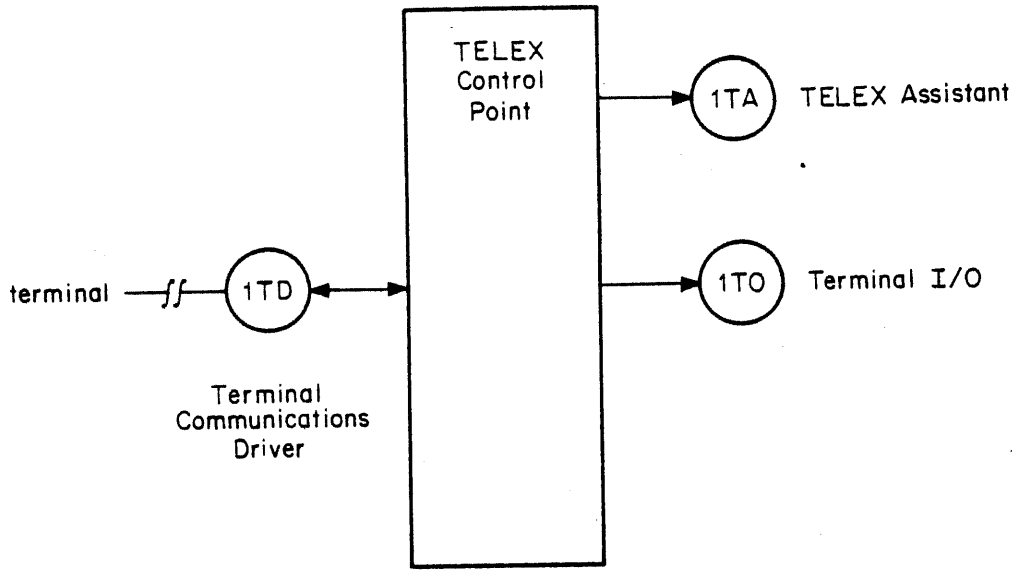


Figure 15-1. TELEX Interactive Subsystem

TERMINAL OPERATION

The flow of data to or from a terminal and a mass storage device is shown in figure 15-2. The terminal user enters source statements. These statements are built character by character and stored in pots (a pot is an eight-word buffer) by 1TD. Whenever 1TD has filled VIPL pots (defined in common deck COMSREM) it issues a dump pot request. TELEX initiates the routine DMP (local to TELEX) which calls 1TO. In the interim 1TD may have filled another pot. Routine 1TO dumps the accumulated pots onto one sector on mass storage. Thus, currently, during this phase 20 or 30 words are written per sector.

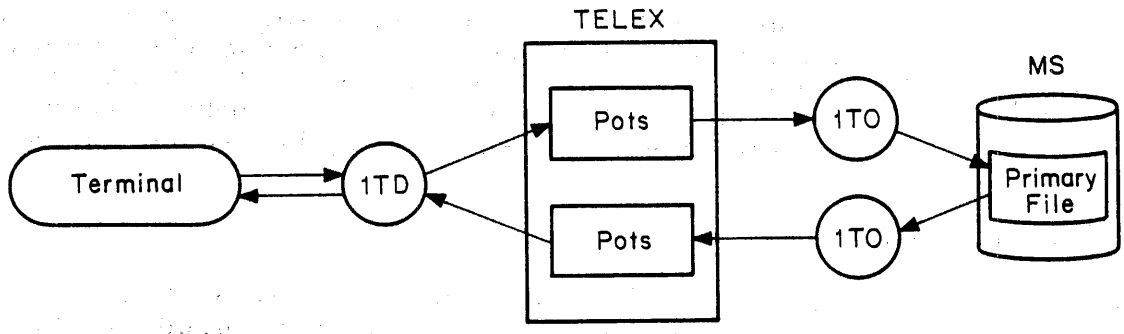


Figure 15-2. Terminal Mass Storage Data Flow

This continues until the user enters a command that forces a sort such as RUN or LIST. If the unsorted file is too large, then the message FILE TOO LONG TO SORT is issued. In this case, the user must issue the SORT command.

If, however, the file is not too long, then the terminal is placed in sort mode. An MTOT job called MSORT is scheduled and all users in sort mode are sorted at once. These users are queued up until a specified time interval has expired, then the MSORT job is run. All the files are given to MSORT in file size order, largest first.

MSORT is an in-memory shell sort. It is started at a control point with the FL necessary to sort the largest file. It sorts the file and rewrites the file in packed format (that is, 100 words per sector). When MSORT has finished a sort, it releases FL down to the necessary size for the next file and then sorts it. This continues until all the files are sorted. Routine 1RO sets all he terminals whose files were sorted to active mode and TELEX then processes the command that indirectly caused the sort.

TERMINAL JOB INITIATION

Refer to figure 15-3 for this discussion. Assuming that a user's primary file has been sorted and RUN is entered at the terminal, the following events occur.

1. TELEX builds a control statement ($LDC or compiler control statement) in a pot and calls 1TA.

2. 1TA builds a rollin queue entry in the system FNT/FST area. The FNT entry points to the user's rollout file (refer to figure 15-21).

3. The scheduler, 1SJ, determines that this is the best job to initiate, so it assigns a control point and calls 1RI to roll in the job.

4. 1RI reads the rollout file to build system FNT entries as specified, builds an FNT entry for the primary file (input to the compiler), and completes the initialization of the control point.

5. 1RI then calls 1AJ to advance the job which detects the $LDC or compiler control statement and loads the compiler with sufficient field length to compile the source statements. ($LDC is used to load the BASIC compiler.) After compiling, the program is executed. As the job executes it may interact with the terminal by issuing output and receiving input.

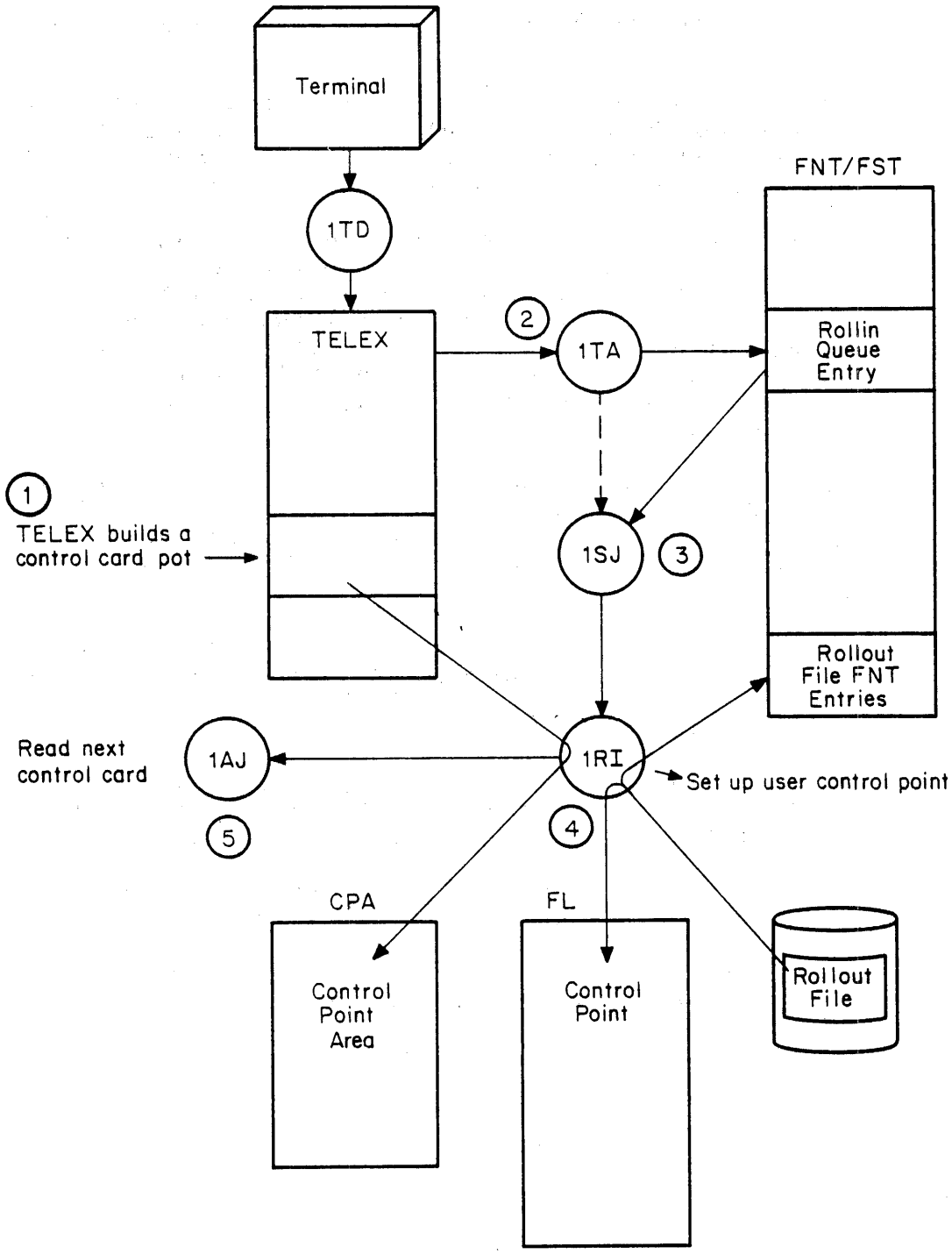


Figure 15-3. Terminal Job Initiation

TERMINAL JOB INTERACTION - OUTPUT

Refer to figure 15-4 for this discussion. When a terminal job writes information to the OUTPUT file, the following events occur.

1. CIO is called when the interactive program issues a write request to the OUTPUT file. CIO senses that this is a time-sharing job (TXOT) and issues monitor function ROCM to roll out the control point.

2. 1RO initiates the rollout and copies the entire field length (including output data) to the rollout file. In addition, all FNT entries associated with this control point are removed from the system FNT area and stored on the rollout file. Prior to calling 1TO, 1RO reads the first sector of output into 1RO's PP memory where it can be picked up by 1TO without additional disk input/output.

3. 1TO is loaded into the same PP as 1RO. The monitor function TGPM assigns 1TO pots into which it writes the output data. 1TO then informs TELEX that output is available for the terminal by issuing monitor function TSEM.

4. TELEX assigns the data pots to 1TD for transmission to the terminal. 1TD continues to ask TELEX for additional output and TELEX in turn calls 1TO until all output has been transferred.

5. After all output is transferred, TELEX calls 1TA to reinitiate the time-sharing job. 1TA builds the rollin file entry in the system FNT area.

6. Scheduler 1SJ selects this queue entry as the best job, assigns a control point, and calls 1RI.

7. 1RI rolls the job into the control point and the time-sharing job continues to execute.

TERMINAL JOB INTERACTION - INPUT

Refer to figure 15-5 for this discussion. Assuming that the time-sharing job is to receive data (input) from the terminal, the system performs the following functions.

1. The job issues a read request on the INPUT file which calls CIO. CIO stores the FET address in control point area word TINW and issues monitor function ROCM to roll out the job.

2. 1RO is loaded to perform the rollout operation. 1RO flags the request in terminal table word VROT and then calls 1TO to complete the process.

3. 1TO issues any available output and issues monitor function TSEM to inform TELEX of the completion of its processing.

4. TELEX calls 1TD to send any output and/or issue the input prompt character (?).

5. 1TD stores characters in pots as they are received from the terminal.

6. When the terminal carriage return is sensed, TELEX calls 1TA to reinitiate the time-sharing job. 1TA builds a rollin queue entry.

7. 1SJ selects the queue entry as the best job, assigns a control point, and calls 1RI.

8. 1RI rolls the job into the control point and transfers the input data from the pots to the job's circular buffer. The job is then activated (given the CPU) and continues to execute.

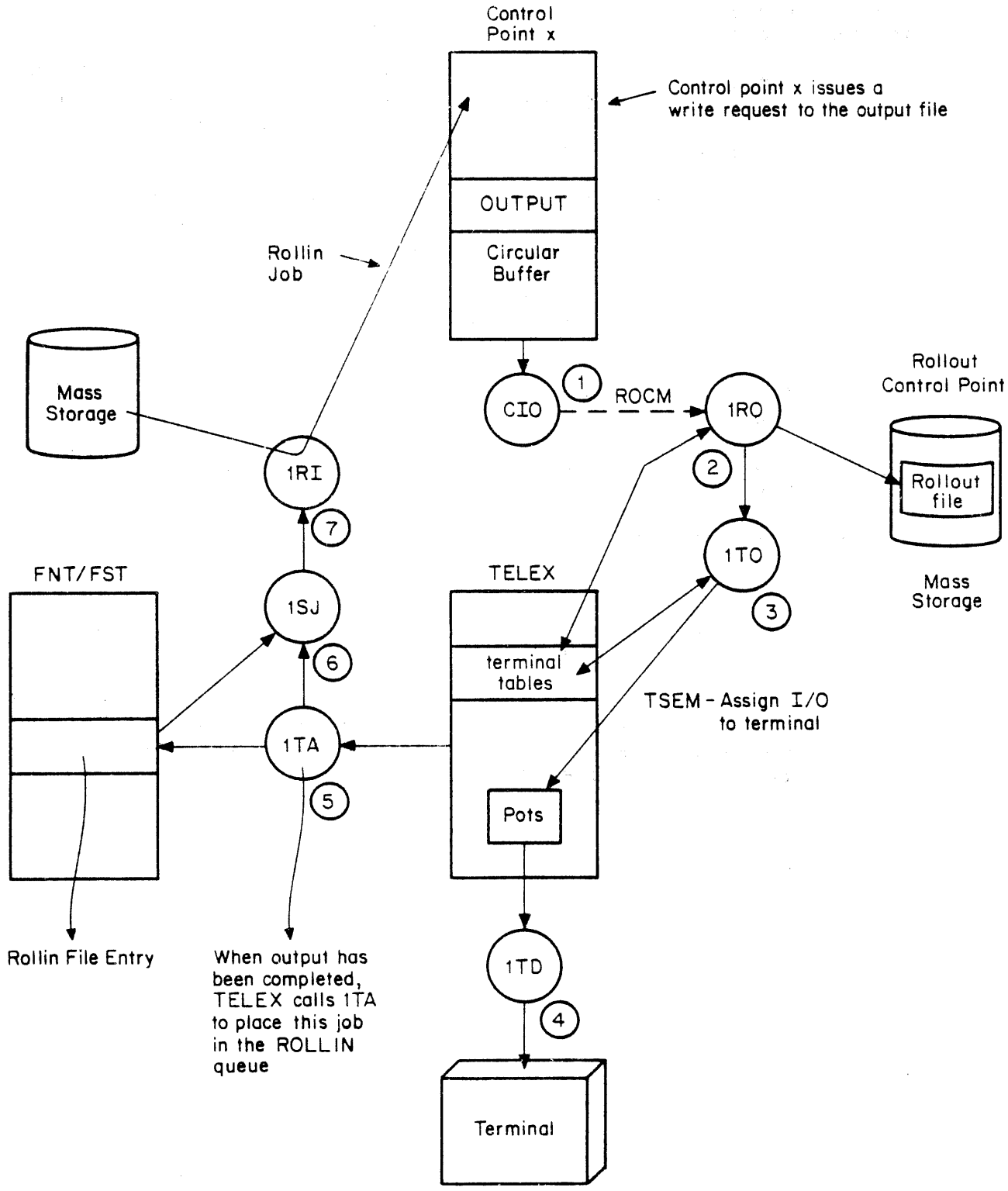


Figure 15-4. Terminal Job Interaction (Output)

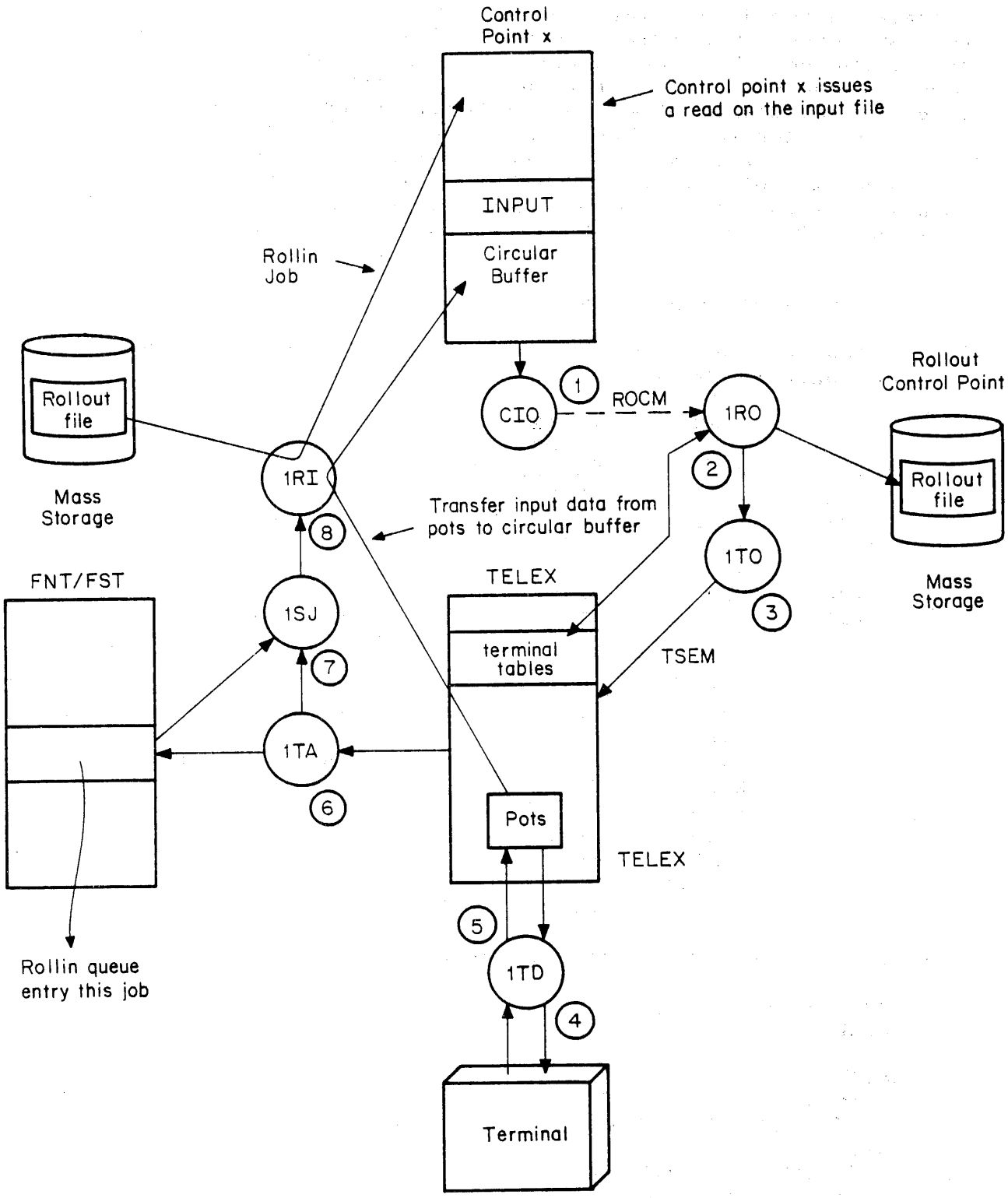


Figure 15-5. Terminal Job Interaction (INPUT)

TELEX INTERACTIVE JOB NAMES

Whenever a job is initiated at a control point, 1TA generates a job name based on terminal number and user index of the user. The common deck COMPGJN (generate job name) is used for this task. Whenever a job is rolled back to TELEX by 1RO, the job name must be decoded back to the terminal number. Routine 1RO uses the common deck COMPGTN (generate terminal number) for this task. In this way, 1RO knows the terminal table in which to indicate the rollout back to TELEX. The terminal number is coded into the fifth through seventh characters of the job name. The user index is coded into the first thru fourth characters.

INTERACTIVE COMPASS PROGRAM EXAMPLE

The following program demonstrates how an interactive COMPASS program could be structured.

```
            IDENT     INTER
            ENTRY     START
OUTPUT      FILEC     OUTBUF,101B,FET=6
OUTBUF      BSS       101B
INPUT       FILEC     INBUF,101B,FET=6
INBUF       BSS       101B
IN          BSS       16
SETUP       VFD       42/0LOUTPUT,18/OUTPUT
START       SA1       SETUP       SET FET POINTER FOR BUFFER
                                  FLUSHING
            BX6       X1
            SA6       2
            BX6       X6-X6       TERMINATE FILE LIST
            SA6       3
            WRITEC    OUTPUT,(=C* THIS PROGRAM INTERACTS.*)
            READ      INPUT
            READC     INPUT,IN
            WRITEC    OUTPUT,IN
            WRITER    OUTPUT
            ENDRUN
            END       START
```

The following demonstrates how the program is executed.

```
old, interf
READY.
batch
$RFL,0.
/compass,i=interf,l=0
    1.008  CPU SECONDS  ASSEMBLY TIME.
/lgo
 THIS PROGRAM INTERACTS.
? please repeat after me...
PLEASE REPEAT AFTER ME...
LGO.
```

TELEX INITIALIZATION

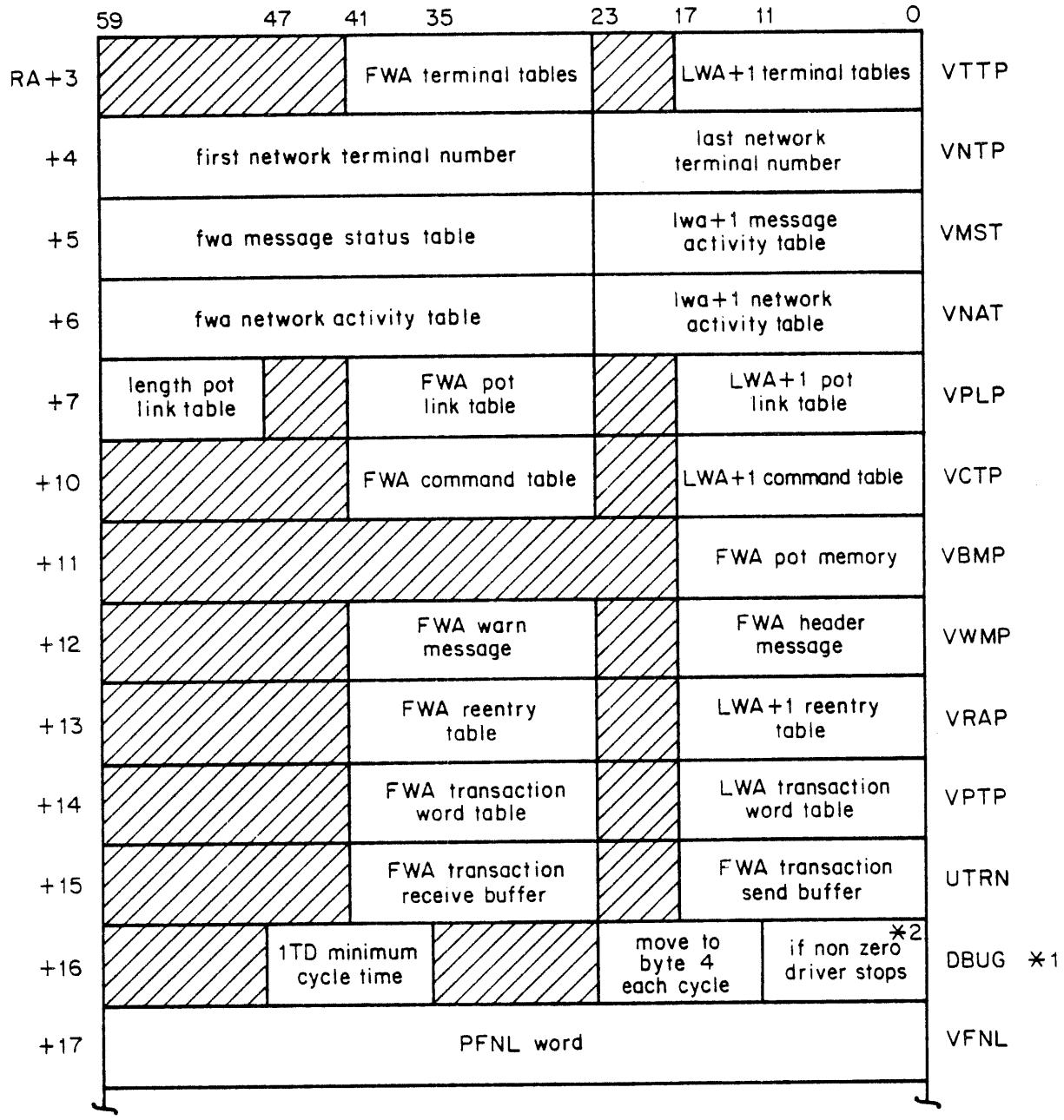
Basically, TELEX initializes tables and pointers, then loads and starts TELEX1, the main routine. PP programs called during initialization include the following.

Program	Description
CIO	Combined input/output
CPM	Control point manager
LDR	Load overlay
PFM	Permanent file manager
1MA	Issue dayfile message
1TA	Auxiliary function processor
1TD	Terminal multiplexer driver

When the operator types TELEX., DSD calls 1DS which generates an input FNT/FST entry of a special type. Routine 1SJ recognizes this entry during its scheduling process, assigns the entry to the proper control point, and calls 1SI into a PP. Routine 1SI calls a procedure file named TELEX, an indirect access file found under the system user index. The recommended contents of this file are as follows.

```
*RETURN PROCEDURE FILE TELEX
RETURN,TELEX.
WHILE,TRUE,LOOP.
TELEX.
TELEX2.
SKIP,LOOP.
EXIT.
TELEX2.
ENDIF,LOOP.
ENDW,LOOP.
```

Initialization consists of allocating tables, establishing the pointers shown in figure 15-6 and the constants shown in table 15-1.



*1 Driver debug word.
*2 Useful in debugging 1TD.

Figure 15-6. Pointer Addresses

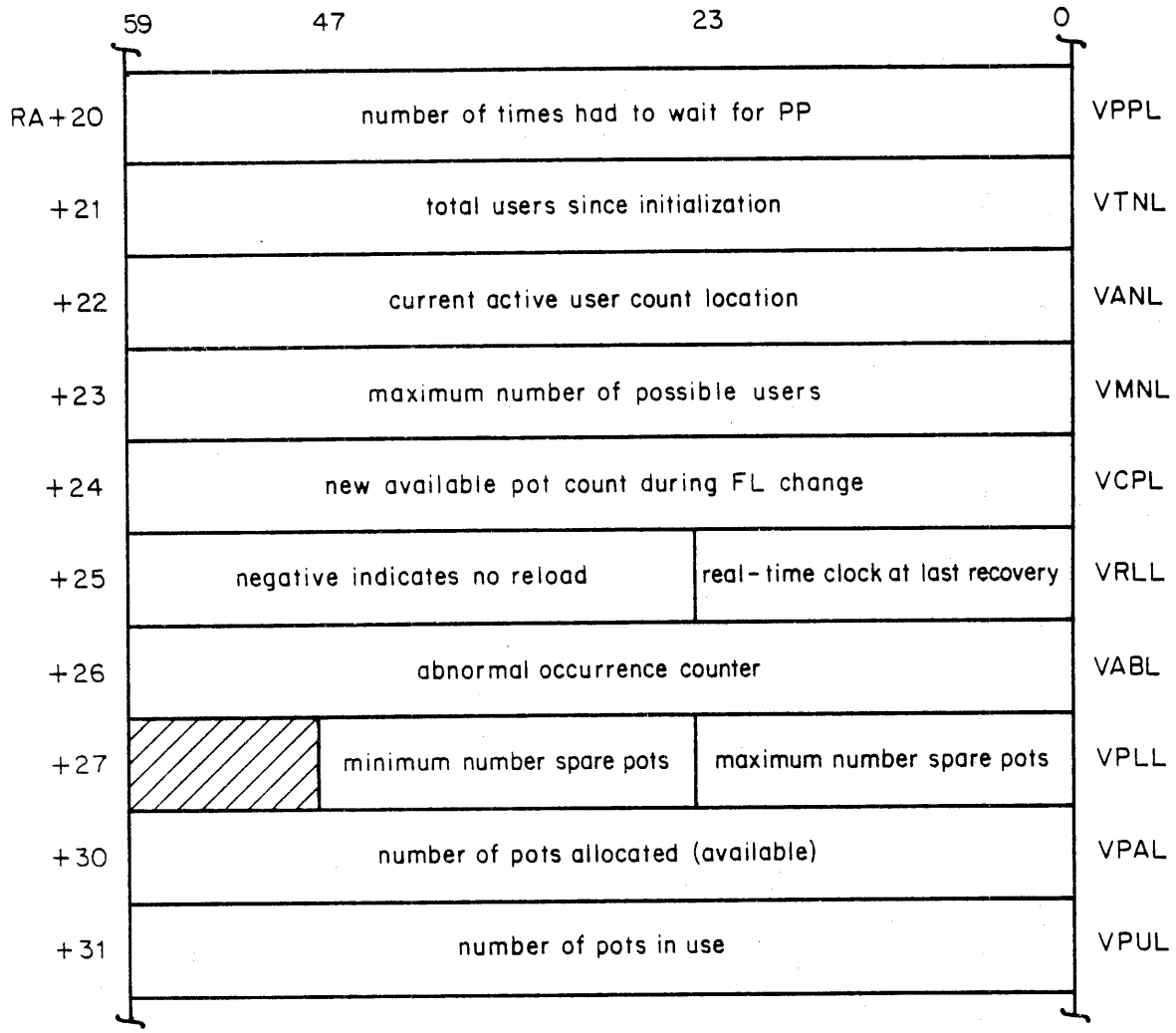


Figure 15-6. Pointer Addresses (Continued)

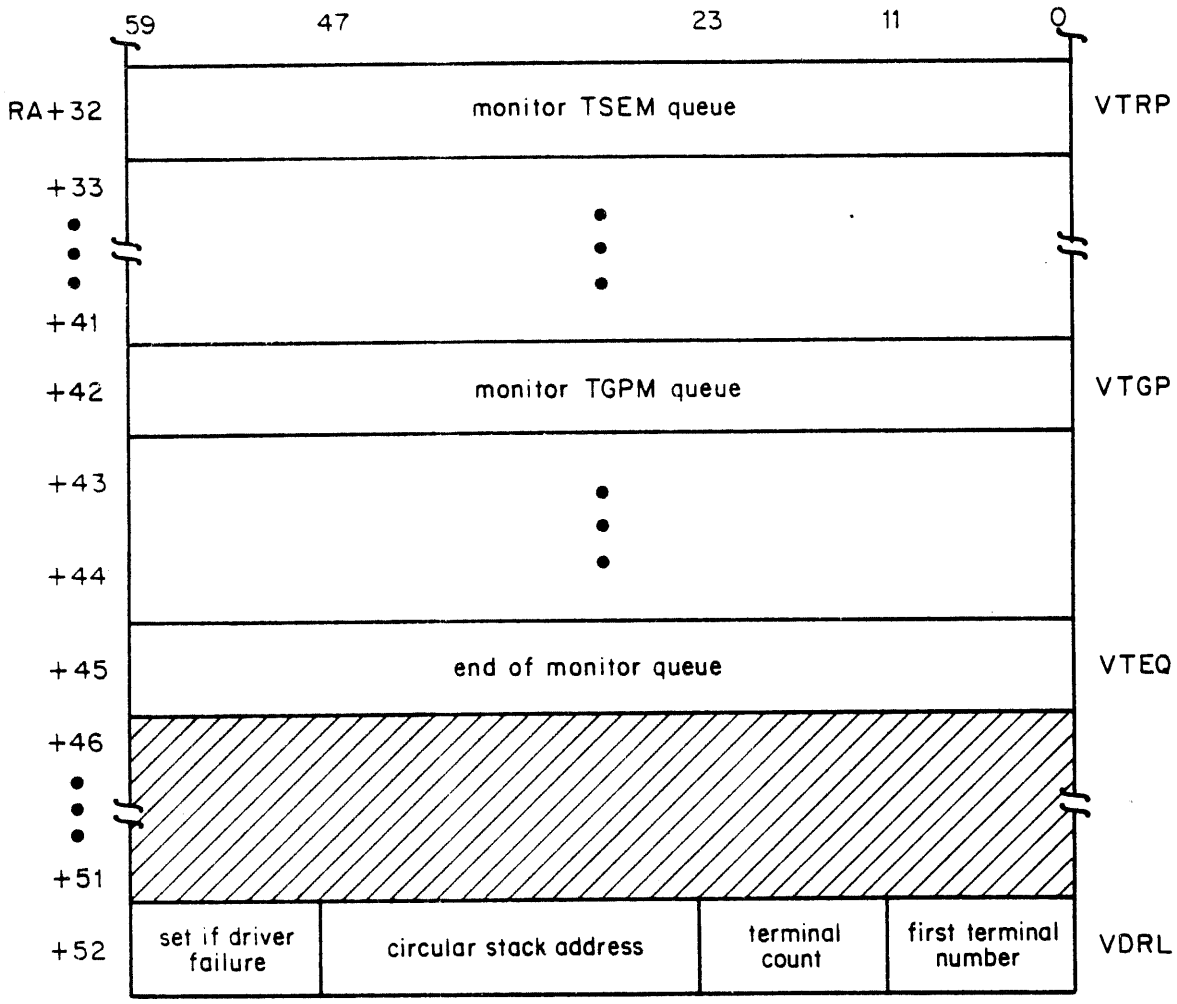


Figure 15-6. Pointer Addresses (Continued)

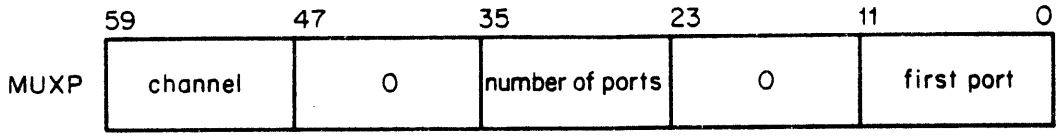
TABLE 15-1. TELEX CONSTANTS

Constant	Value	Description
MSORFL	4	MSORT base FL
VBFL	0	Default BATCH subsystem FL
VBPL	2	Maximum ABL for low-speed lines
VBPT	3	Additional PLT words per high-speed line
VBTL	12B	1TO call time delay for high-speed lines (one second)
VCPC	10	Number of words per pot
VCPT	1	TELEX control point number
VDSL	100	Length of driver circular queue
VIPL	2	Number of allowable source pots before dump
VMPL	40	Maximum number of spare pots per 64 users
VMTO	10	Multiplexer terminal SALVARE file time-out value (minutes)
VNPL	4	Minimum number of pots for network
VNTL	13B	Default 1TO call time delay (two seconds)
VNTO	20	Network terminal SALVARE file time-out value (minutes)
VOPL	4	Number of pots issued for multiplexer
VRBL	110/VCPC	Transaction receive buffer length in pots
VSBL	100/VCPC	Transaction send buffer length in pots
VSPL	20	Minimum number of spare pots per 64 users
VTGL	VTEQ-VTGP	Length of monitor TGPM queue
VTRL	VTGP-VTRP	Length of monitor TSEM queue
VXPL	20B	Maximum number of pots for network
UTIS	10	Default user time limit/10

TABLE 15-1. TELEX CONSTANTS
(Continued)

Constant	Value	Description
The following are VROT status bits used with 1RO		
VJIR	1S1	Job in system
VRIR	1S2	Job to be rolled in again
VIPR	1S3	Input requested
VOPR	1S4	Output available
VECS	1S10	Job uses ECS
MAXTT	1024	Maximum number of terminals
MPLT	120B	Number of PLT words per 64 users
WCQT	100	Wait completion queue delay time (msec.)
LIAA	4	Login attempts allowed
CBASE	0	Default base for command parameter (octal)
LISDL	2	List delay time
COMDL	6	Compile delay time
EXEDL	5	Execute delay time
CATDL	5	CATLIST delay time
SORDL	2	Sort delay time
BATDL	4	Batch delay time
RESDL	4	Resequence delay time
SWPDL	0	Swap-in delay time
NULDI	10	Null input response delay time
BASDI	4	BASIC input response delay time
FTNDI	4	FTNTS input response delay time
TRADI	4	Transaction input response delay time
EXEDI	4	Execute input response delay time
BATDI	5	Batch input response delay time
ACCDI	10	Access input response delay time
SYSDI	3	System processed commands
SALTO	3	SALVARE file time check (mins.)

The following illustrates the multiplexer table. An entry exists for each time-sharing multiplexer in the EST which is on.



After initializing the tables, TELEX modifies addresses in TELEX1 code which use the increment instruction operation definitions. Next, each terminal table entry is set to complete status by setting VROT equal to 3 in each entry. Next, the warning message address VWMP is set to the normal header. Next,

TELEX calls 1TA to search for time-sharing jobs in the system. The jobs searched for are TXOT and MTOT type. The count of such jobs is returned in a pseudo terminal table for TELEX. If the count is nonzero, TELEX aborts with the message: TELEX INITIALIZATION ABORT. Next, each driver queue is initialized by setting FIRST, IN, OUT, and LIMIT. The driver queues are used like circular buffers. Finally, after starting the drivers and verifying the recovery file (SALVxx, where xx is the machine ID), TELEX is complete and control is transferred to TELEX1.

TELEX1 - MAIN PROGRAM

TELEX1 is the main program that controls and coordinates the time-sharing subsystem. This program is driven by the following queues.

- Request entering TELEX:

Queue	Description
Driver request	Requests from 1TD
Monitor request	Requests from other PPs
Monitor pot request	Requests from other PPs for pots

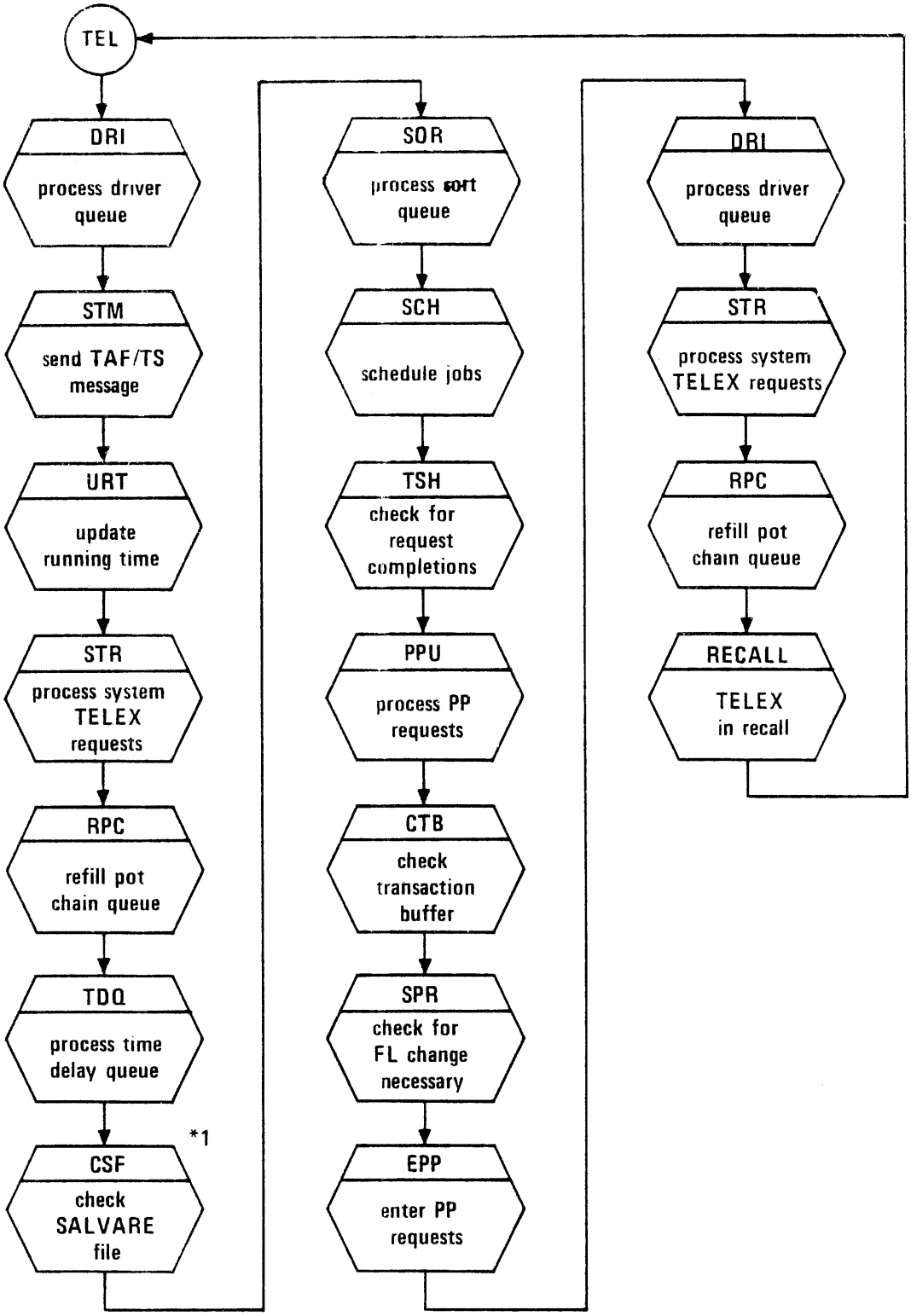
- Internal control:

Queue	Description
Wait completion	Wait for completion of a process
Time delay	Wait for time to elapse
Job	Wait to do all job scheduling at one time
Sort	Wait to do all sort scheduling at one time

- Requests sent by TELEX:

Queue	Description
1TA	Send all 1TA requests at one time
1TO	Send all 1TO requests at one time

These queues are scanned by the TELEX1 control loop which is defined in the flow chart of figure 15-7.



*1 The SALVARE file contains a two-word entry for each user in recovery state.

Figure 15-7. TELEX1 Control Loop

Every 3 minutes the SALVARE file entries are checked and if the time is over 10 minutes old, the entry is removed, the rollout file and all nonpermanent local files are dropped, and the terminal logged off. The users must recover within 10 minutes of system recovery or the SALVARE file entry will be eliminated. The SALVARE file is discussed further under SALVARE - TELEX Recovery File in this section.

The relationship between processing modules of TELEX1 is shown in figure 15-8.

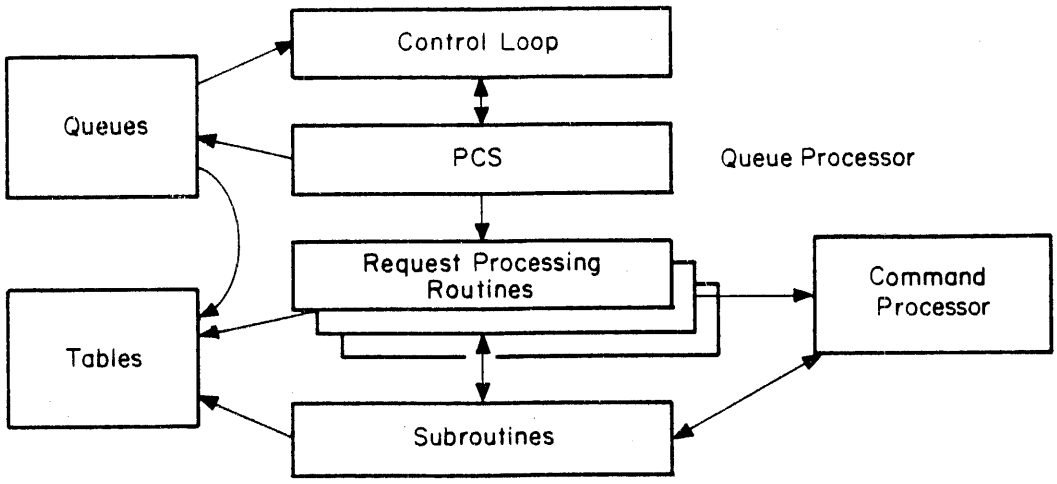


Figure 15-8. TELEX1 Processing Modules

In general, all tables in TELEX are dynamic in length at initialization time. The lengths of the various tables and queues are determined by the maximum number of terminals to be serviced. Thus, it is necessary for all routines at initialization time to determine the values of table pointers, and so on. Once TELEX is initialized, the lengths of tables do not change. Thus, pointers such as FIRST and LIMIT could be read and saved by programs that are time critical. These pointers could also be saved as abolute addresses because TELEX will never be moved. Thus, no SYSEDITs which change the size of CMR can be run while TELEX is running. The TELEX1 memory layout is shown in figure 15-9.

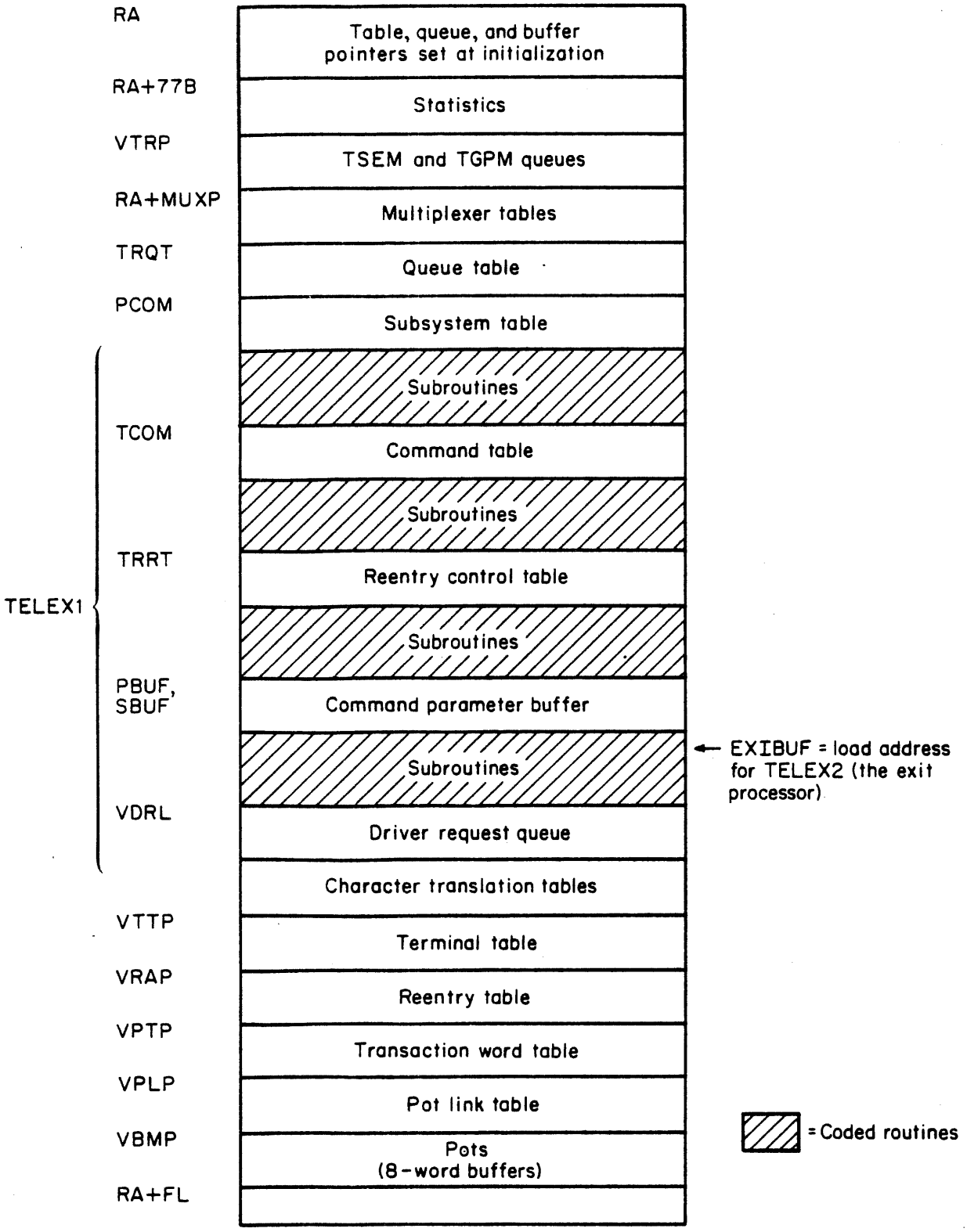


Figure 15-9. TELEX1 Memory Map

DRIVER REQUEST QUEUE(S)

Driver (1TD) requests are passed to TELEX1 via the driver request queue which are circular stacks as shown in figure 15-10.

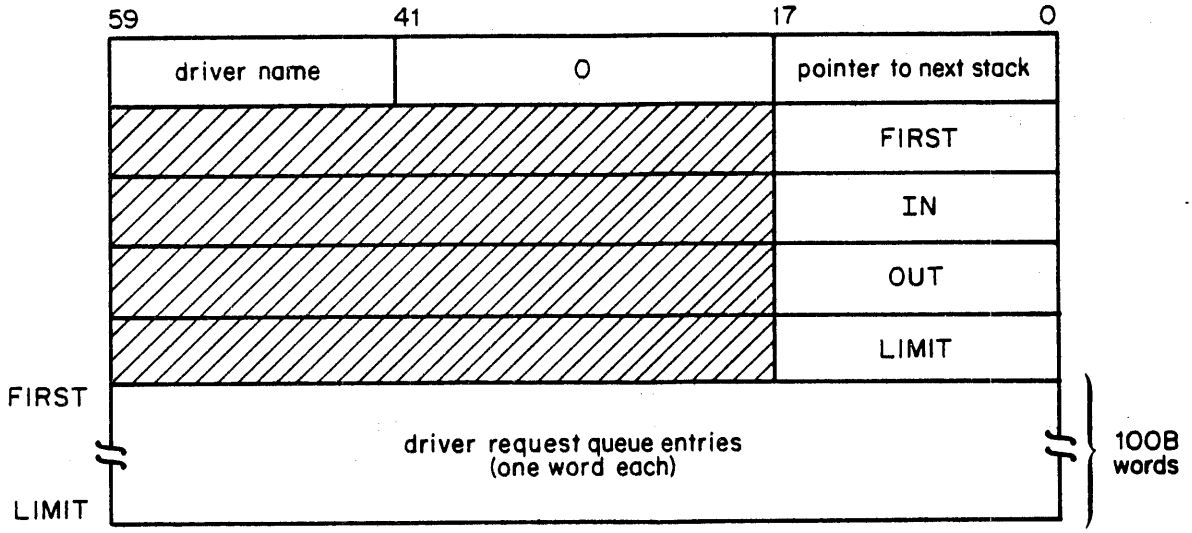
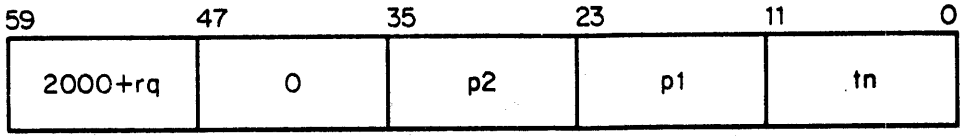


Figure 15-10. Driver Request Queue Stack

Driver request queue entries are placed in a circular stack by 1TD. The IN pointer is updated by 1TD when an entry is placed into the queue. TELEX1 updates the OUT pointer as the driver requests are completed. The driver name is stored in word 1 with a pointer to the next stack. A zero pointer indicates the last stack. Each stack is 105B words in length (100B words for entries plus five header words). A maximum of four stacks may exist; one for each driver (1TD). The entries are one word as follows:



rq Request number
p2 Parameter 2
p1 Parameter 1
tn Terminal number

The request number is always biased by 2000B so that a jump table index can be stored in a B register with use of the unpack instruction. For example, if the above word is in X2, consider the following instruction.

```
UX1,B7  X2
```

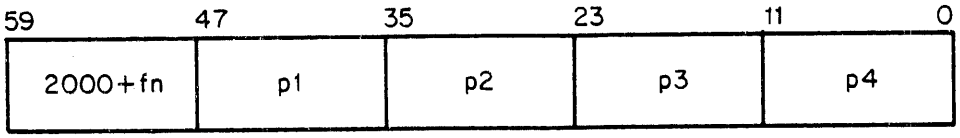
The result is that B7 contains the request number and X1 contains the parameters and terminal number (that is, the lower 48 bits). A list of request numbers (request codes) is maintained in common deck COMSTDR and are listed in table 15-2.

TABLE 15-2. DRIVER REQUEST NUMBERS (ISSUED TO TELEX)

Request Code	Symbol	Description
0	AOD	Increment retry count
1	CSC	Circuit scan complete
3	CLI	Command line input, P1=first pot, P2= word in pot
4	DLO	Data lost, P2=type
5	DRP	Drop pot
6	DRT	Drop pot chain, P1=first pot
7	HUP	User hung up phone
10	IAM	Issue accounting message, P2=type
11	LOF	Log off user
12	LPT	Request additional pot, P1=current pot
13	MAL	Set transaction terminal malfunction, P1=status (1=malfunction, 0=none)
14	MTN	Terminate monitor mode (monitor teletypewriter)
15	RES	Request more output, P1=current pot
16	RIN	Release source line, P1=first pot
17	SAI	Set autoinput mode
20	SKY	Interrupt from terminal, P2=interrupt level
21	SPT	Set transaction output pot, P1=pot
22	SSC	Set transaction sequence code, P1=code
23	TTI	Transaction terminal input
24	EMO	Completed marked transaction output

MONITOR REQUEST QUEUE(S)

PP requests for TELEX processing are handled via the PP monitor function TSEM. The message buffer is set up by the requesting PP according to the following format.



p1 - p4 Parameters depending on the function.

fn Function code. These function codes are defined in packed format in common deck COMSREM. They are listed in table 15-3.

TABLE 15-3. TSEM MONITOR REQUEST FUNCTIONS

Value	Name	Description
2000	VDPO	Drop pots
2001	VASO	Assign output
2002	VMSG	Terminal message
2003	VSDT	Set terminal table bit (VSTT only)
2004	VCDT	Clear terminal table bit (VSTT only)
2005	VSCS	Set character set mode
2006	VPTY	Set parity
2007	VSBS	Set subsystem

PP monitor picks up the above request and stores it in a free slot in the TELEX monitor queue for TSEM functions. This queue is located at VTRP in TELEX and is 10B words in length. If no slot is free in this queue, monitor (MTR) keeps trying until TELEX honors an existing request and clears a slot.

In general, TELEX drops any unused pots in the chain. If the last pot is not completely filled by the routine issuing output, the routine must put in a terminator byte (0014) at the end of the output data.

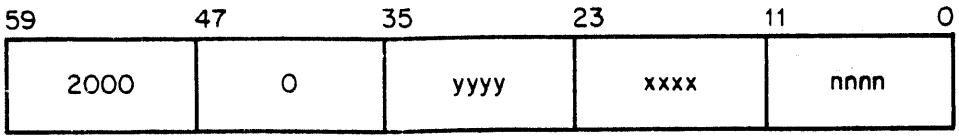
NOTE

When issuing a 2001, terminal status must have bit 4 set in VROT.

Pots for output are obtained by issuing the monitor function TGPM. These requests are handled by TELEX in a three-word queue similar to TSEM requests.

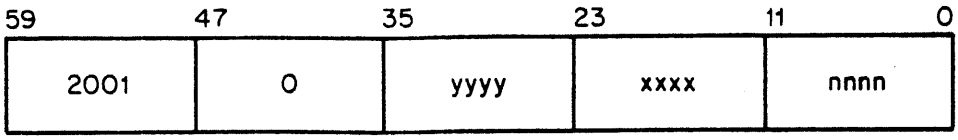
The parameters for the various functions are defined as follows.

VDPO - Drop Pots (TELEX Routine DRT)



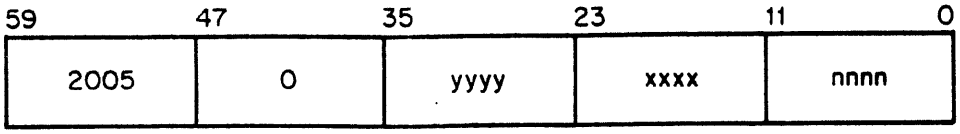
yyyy	Last pot to be dropped
xxxx	First pot to be dropped
nnnn	Terminal number

VASO - Assign Output (TELEX Routine ASO)



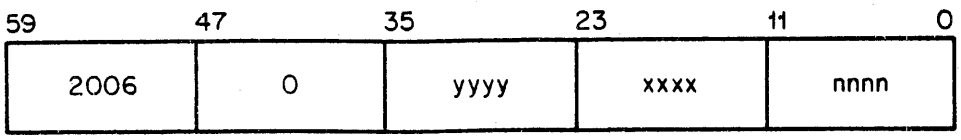
yyyy	Last pot of output
xxxx	First pot of output
nnnn	Terminal number

VSCS - Set Character Set Mode (TELEX Routine SCS)



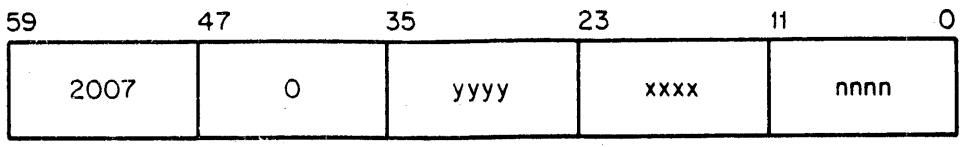
yyyy	Last pot containing mode
xxxx	First pot containing mode
nnnn	Terminal number

VPTY - Set Parity (TELEX Routine PTY)



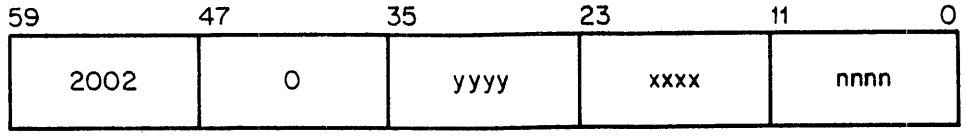
yyyy	Last pot containing parity
xxxx	First pot containing parity
nnnn	Terminal number

VSBS - Set Subsystem (TELEX Routine SBS)



yyyy	Last pot containing subsystem
xxxx	First pot containing subsystem
nnnn	Terminal number

VMSG - Assign Message (TELEX Routine DSD)



yyyy	Last pot of message
xxxx	First pot of message
nnnn	Terminal number; if below maximum number of pseudo terminals, then this is a warning message sent to all terminals

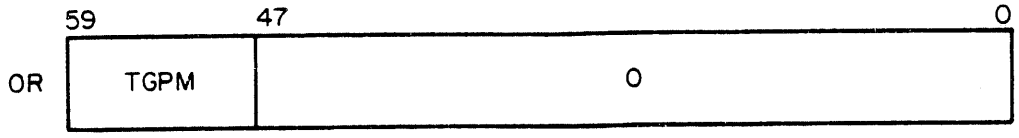
VMSG is used by DSD to process the DIAL and WARN operator commands.

VSDT and VCDT TSEM Requests

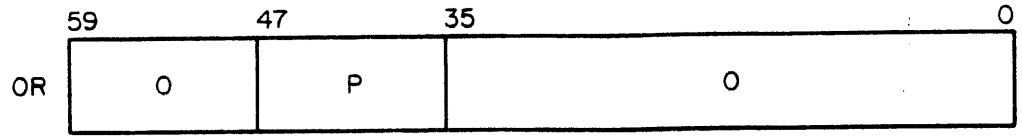
When a terminal user initiates a CPU program, he may terminate that program with the S or STOP entry. If the user wishes to disable/enable this function he can use the DISTC macro described in section 12 of the NOS Reference Manual, volume 2. This macro generates an RA+1 call to the PP routine TLX. TLX issues the appropriate TSEM request (function 2003 or 2004), which sets the terminal interrupt address in TIAW. The disable function ignores this field and sets the disable bit in the terminal table VSTT. The enable function sets this field to the address relative to RA specified in the call and clear the disable bit in the terminal table VSTT. Refer to volume 2 of the reference manual for a complete description of DISTC.

TGPM Request

Pots for output are obtained by issuing the monitor function TGPM The requests are handled by TELEX in a 3-word queue similar to TSEM requests. The call to TGPM is as follows.



Upon return, the OR is as follows.



p Pot pointer (0 if no pots available)

If p=0, the PP should reissue the request.

Whenever a PP needs a pot chain it issues the TGPM MTR request. MTR searches the TELEX TGPM queue for a nonzero entry. If MTR finds one, it will be the first pot of a pot chain. The chain size is an assembly constant and is currently fixed at 4 pots. This pot chain is assigned to the calling PP and the queue entry is zeroed. If the queue is empty, MTR issues an RCLM on TELEX.

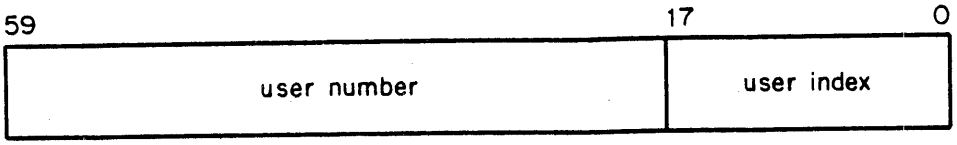
During TELEX's main loop it checks the TGPM queue and if it finds any empty entries, it generates a pot chain and places the first pot number in the queue.

The TGPM is used by 1TO, which requests pots for flushing a TXOT type jobs OUTPUT file. Another user is DSD, who must get a pot chain for the WARN and DIAL messages.

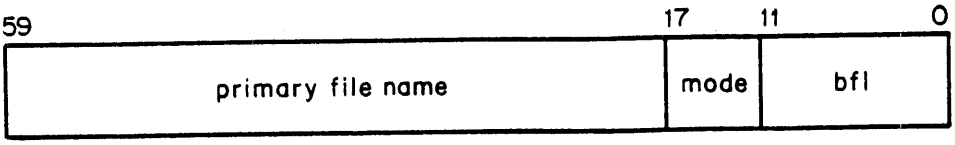
TERMINAL TABLE

The terminal table contains an eight-word entry for each possible active user. Each entry contains the current status of each port on each multiplexer. These eight-word entries are structured in such a way so as to minimize interlocks between TELEX1 and the various PP routines which read and write them.

Word 0 (VUIT) is written by TELEX and 1TA and read by TELEX and 1TA. Its format is as follows.

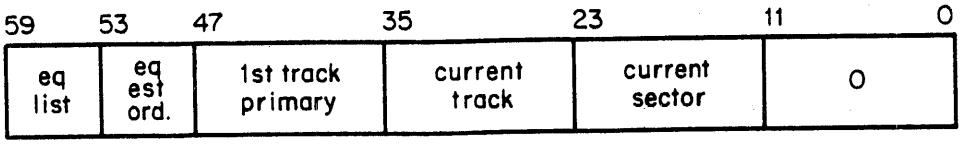


Word 1 (VFNT) IS WRITTEN BY TELEX, 1RO, and 1TA and read by 1RI, 1TA, TELEX, 1RO, and 1TD. Its format is as follows.

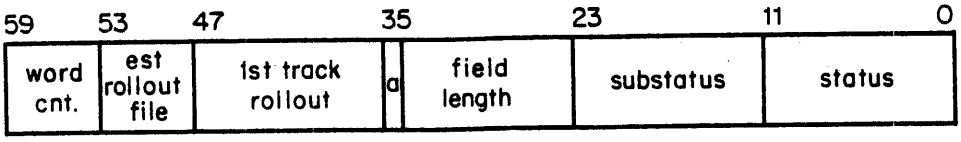


mode	Write lockout if bit 0 set; execute only if bit 2 set
bfl	RFL value for batch subsystem; sector count for 1TA on RUN, I=lfn

Word 2 (VFST) is written by TELEX, 1RO, 1TA, 1RI, and 1TO and is read by TELEX, 1RO, 1TA, 1RI, and 1TO. Its format is as follows:



Word 3 (VROT) is written by TELEX, 1RO, 1TO, 1RI, and 1TA and is read by TELEX, 1RO, 1RI, 1TA, 1TO, and 1TD for the rollout file. Its format is as follows:



a Absolute FL flag; if not set, then FL is in units of 100B

substatus:

File list if 0 (list with EOR and EOF if 1)

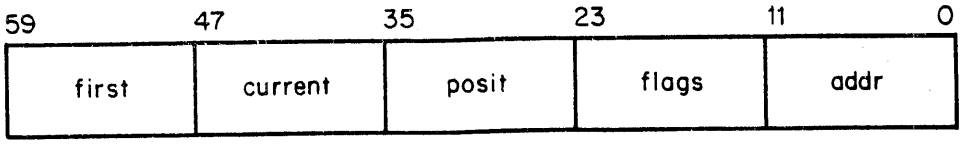
Job status:

Meaning	Bit
Level number	23-20
SRU limit	19
Time limit	18
Terminate special job with FL	17
Terminate special job	16
Interrupt	15
Input status	
EOI	14
EOF	13
EOR	12

status:	Bit	Value
TELEX in control	0	1
System in control	0	0
Job in system	1	0
Job to be rolled in	2	1
Job awaiting input	3	1
Output available	4	1
TAFTS output bit	4	1
Special system job	5	1
List	6	1
Multi-terminal	7	1
Suspended	9	1
Not used	10	
Error on last operation	11	1

Words, VDPT, VCHT, and VDCT are used by 1TD to maintain current information for the terminal. The main loop of 1TD reads these words into PP memory at direct cells DP, CH, and DC corresponding to VDPT, VCHT, and VDCT. When the main loop jumps to the appropriate routines, they use these direct cells instead of reading from CM. When control is returned to the main loop, it writes these direct cells back to CM if necessary. VDCT is mainly used for communication with TELEX and is interlocked by TELEX. If byte 4 is not clear, then this terminal is being processed by 1TD. When byte 4 clears, then 1TD is done and TELEX can use the information to continue activity for this terminal.

Word 4 (VDPT) is written by 1TD and read by TELEX1 and 1TD. Its format is as follows:



first First pot of input line
current Current pot of line being processed
posit Position within pot as follows:

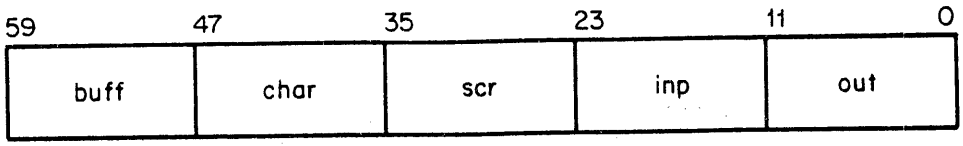
Bits	Meaning
11-9	First word in first pot of input line
8	Input initiated
7	Next input pot requested
6-4	Current word in current pot
3-0	Character number in current word

flags Control flags as follows:

Bits	Meaning
11-7	Translation table index
6	Full duplex
5	Polled terminal error flag Correspondence upper case flag
4	Monitor mode
3	Binary transmission
2	Transparent input
1	Extended mode transmission
0	Odd parity

addr The address of the PP driver subroutine which is currently processing the terminal

Word 5 (VCHT) is read and written by 1TD. Its format is as follows:



buff During input, buffer holds the upper (even) character of byte until the next character is received at which time both characters (one byte) are stored into a pot; during output, buffer contains the driver subroutine address

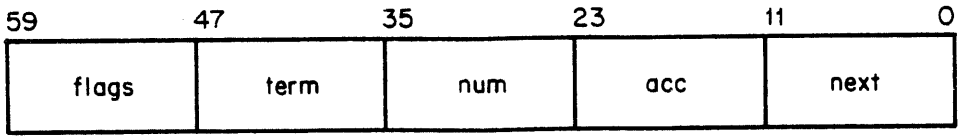
char Total character count of line being processed

scr Scratch and reentry address for polled terminals (TAF/TS type); it most often contains the current input or output character for non-polled terminals (may also hold control byte during output)

inp Total number of characters received from terminal

out Total number of characters transmitted to terminal

Word 6 (VDCT) is written and read by TELEX1 and 1TD. Its format is as follows.



flags Flags as follows:

Bit	Value	Meaning
0	0001	Tape mode
1	0002	Auto mode
2	0004	Text mode
3	0010	Extended mode
4	0020	Transaction mode
5	0040	Monitor mode
6	0100	Read data mode
7	0200	
8	0400	Input requested
9	1000	User logged in
10	2000	Interrupt complete
11	4000	Driver request from TELEX1 byte 4

term Terminal control information as follows:

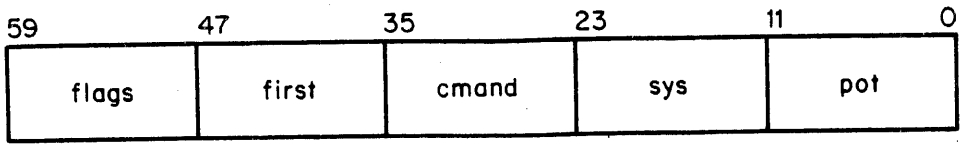
Bit	Value	Meaning
0-2	0-7	First word of output line in pot
7-3	0-37B	User defined carriage return delay
9-8	0-1	Line type 0-1 Terminal type 2 Hardwired line 3 Polled line
11-10		Not used

num In AUTO mode, the line number increment; in MONITOR mode, the terminal number of the terminal being monitored (that is, the monitoree)

acc Access control flags (lower 12 bits of access word defined in VALIDUs file for this user). Refer to the NOS System Maintenance Reference Manual for procedures to establish the access word. Refer to section 20 for a complete description.

next First pot of an output message assignment or driver request function code (byte 0, bit 59 flag). (Refer to BGI - STT Subroutines).

Word 7 (VSTT) is written by TELEX and is read by TELEX, 1TA, 1TD, 1TO, 1RI, 1RO, and DSD. Its format is as follows.



flags Flags as follows:

Bit	Value	Meaning
48	0001	Log-out in progress
49	0002	Unconditional abort flag
50	0004	Warning issued
51	0010	Run complete message
52	0020	Sort flag
53	0040	Not used
54	0100	Job complete flag
55	0200	Input lost or job not started
56	0400	Not used
57	1000	Charge number required
58	2000	Conditional abort flag
59	4000	Disable terminal control

first First pot of source line input. This byte, along with byte 2 (pot count), is used in subroutine DMP to dump pots to disk as input is received by calling 1TO.

cmand Pot count or index into command table, TCOM. The index is set by subroutine SCT. Also may be used as DSD command pointer.

sys Bits 23-15 nonzero if files lost on RECOVER command. Bits 14-12 are current system in control:

0	Null	3	FTNTS	6	Access
1	BASIC	4	Execute	7	Transaction
		5	Batch		

pot Pot pointer to a queued output message. That is, if a message is already in VDCT and not yet processed, the next message is queued by using byte 4 of VSTT. If another message must be assigned, it will be lost. Refer to subroutine ASM. Normally, this byte is zero.

Table 15-4 is a summary of the terminal table entry.

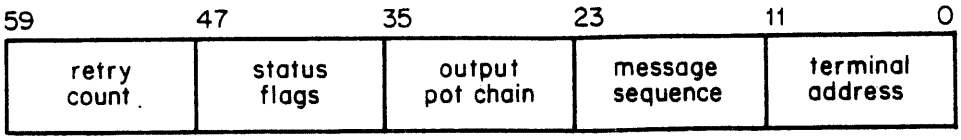
TABLE 15-4. TERMINAL TABLE ENTRY SUMMARY

Name	Word	Written by	Read by
VUIT	0	TELEX, 1TA	TELEX, 1TA
VFNT	1	TELEX, 1RO, 1TA	TELEX, 1RI, 1RO, 1TA, 1TD
VFST	2	TELEX, 1RI, 1RO, 1TA, 1TO	TELEX, 1RI, 1RO, 1TO, 1TA
VROT	3	TELEX, 1RI, 1RO, 1TO, 1TA	TELEX, 1RI, 1RO, 1TO, 1TA, 1TD
VDPT	4	1TD	TELEX, 1TD
VCHT	5	1TD	1TD
VDCT	6	TELEX, 1TD	TELEX, 1TD
VSTT	7	TELEX	TELEX, 1RI, 1RO, 1TA, 1TD, 1TO, DSD

In table 15-4, the name TELEX refers to any of the three overlays comprising TELEX. Any routine which writes a word also is assumed to read that word.

TRANSACTION WORD TABLE

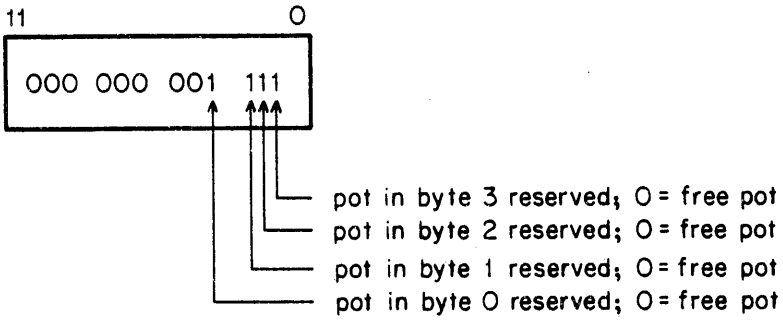
The transaction word table provides TELEX and TAF/TS communication and is pointed to by VPTP and contains the following one-word entry for each transaction terminal.



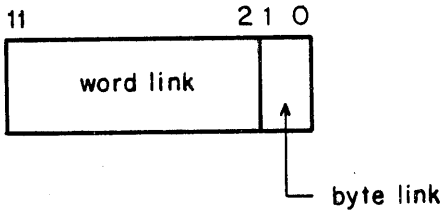
status flags:

Symbol	Bit	Meaning
UTOB	47	Terminal off
UTMB	46	Terminal malfunction
UAMB	45	Terminal waiting for message
UWOB	44	Terminal waiting for output

Byte 4 contains reservation flags in the following format.



Each byte (0-3) represents a pot, an 8-word CM buffer starting at VBMP. Bytes 0-3 contain a link to the next pot in the chain. The last pot in the chain is indicated by a zero byte. Pot zero is always reserved and links to 7777. Each PLT byte has the following format.



These words are written by TELEX transaction routines and read by the driver, 1TD.

POT LINK TABLE

The pot link table (PLT) controls the use of pots (8-word buffers). Its layout is as follows.

	byte 0	byte 1	byte 2	byte 3	byte 4
VPLP+0	7777 0	0002 1	0003 2	0004 3	0017
+1	0005 4	0000 5	0007 6	0000 7	0017
+2	0000 10	0012 11	0013 12	0014 13	0007

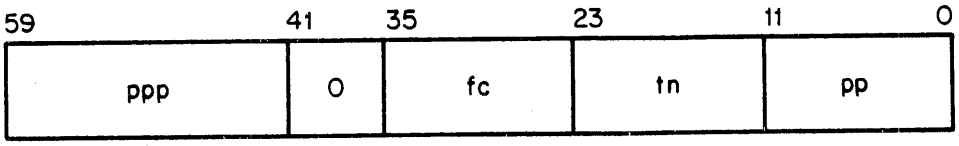
In the preceding table, pots 1-5 are reserved and comprise one chain. Pots 6 and 7 comprise another chain. Pot 10 is free. Pot 11 is the start of another chain.

INTERNAL QUEUES (TRQT)

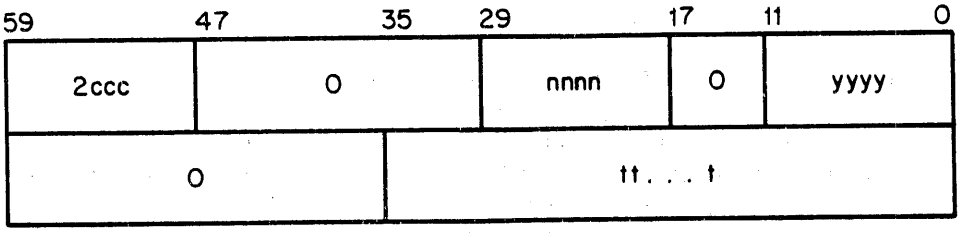
All internal queues are built at assembly time in a table of queues. This table consists of all the queues that may have requests in the reentry table. The following is a list of valid queue names in the table of queues.

Name	Description
WCMQ	Wait completion queue
TIMQ	Time delay queue
JOBQ	Job queue
SORQ	Sort queue
ITAQ	1TA queue (PP request queue)
ITOQ	1TO queue (PP request queue)

The PP request queues are one-word entries in the table of queues, while the other 4 are two-word entries. The format of the entries is as follows.



ppp 1TA, 1TO
fc Function code
tn Terminal number
pp Pot pointer



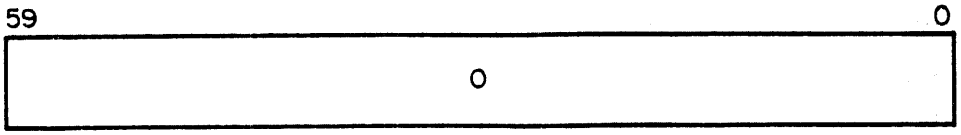
ccc Number of entries (packed format)
nnnn First terminal entry (index into reentry table)
yyyy Last terminal entry (index into reentry table)
tt...t Resource control count

NOTE

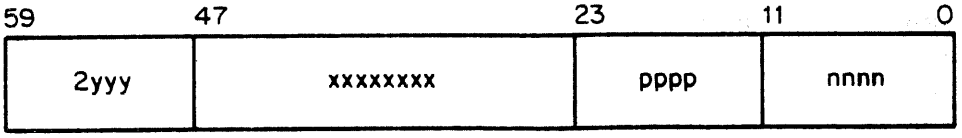
Each queue has an associated string of entries in the reentry table.

REENTRY TABLE (VRAP)

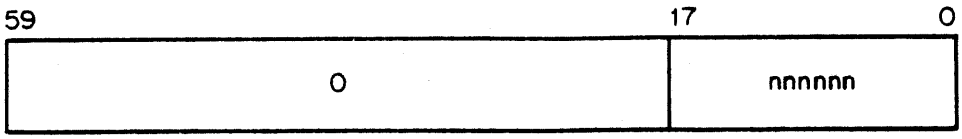
The TELEX subroutines use the reentry table to have control returned or functions performed for them when a set of conditions are met. The table consists of one word for each terminal with one of the following formats.



No reentry conditions



yyy	Index to TRRT (table of reentry processors)
xxxxxxxx	Anything
pppp	Pot pointer for further parameters
nnnn	Link to next entry in the queue of this type (see TSR)

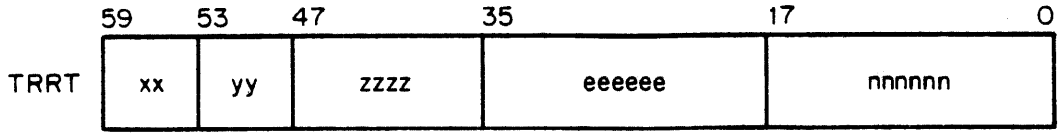


nnnnnn	Pot address of stacked entries

Each entry in the reentry table contains an index to the table of reentry routine parameters (TRRT).

TABLE OF REENTRY ROUTINE PARAMETERS (TRRT)

This table is built at assembly time. It consists of entries that direct further processing based on entries from the reentry table and on completion of certain sections. Entries are added to the table by use of the COMMND macro. Entries are one word, according to the following format.



xx	Index to TRQT (queue table); if xx=0, no resources are required except for a peripheral processor, possibly
yy	Function code for called program
zzzz	Function processing address relative to TSRPROC
eeeeee	Error return address
nnnnnn	Normal return address

The COMMND macro format is as follows.

LOCATION	OPERATION	VARIABLE SUBFIELDS
	COMMND	proc,sysr,npro,erra,func

proc — Entry point of routine to process this command (zzzz)
sysr — The queue that the request is to be placed in: WCMQ, TIMQ, JOBQ, SORQ, ITAQ, or ITOQ (xx)
npro — Normal return address (nnnnnn)
erra — Error return address (eeeeee)
func — Function code to be passed to the called program (yy)

The following example uses the COMMND macro to generate a queue entry.

```
            COMMND INP6,WCMQ,INP6,INP6
INP6$       EQU    *       (This is generated by the COMMND
                            macro).
```

INP6$ is the symbol for this word in the table of reentry routines.

Now to make the WCMQ queue entry:

```
            .          .
            .          .
            .          .
            SX5        INP6$  SPECIFY COMMAND TABLE ENTRY
            EQ         PCS4   MAKE QUEUE ENTRY
            .          .
            .          .
            .          .
INP6        BSS        0      NORMAL AND ERROR RETURN ADDRESS
            .          .
            .          .
            .          .
```

In general, queue entries are made in this manner throughout TELEX.

Figure 15-11 shows the relationship between the table of queues, the reentry table, and the table of reentry routine parameters. There is one queue entry per terminal.

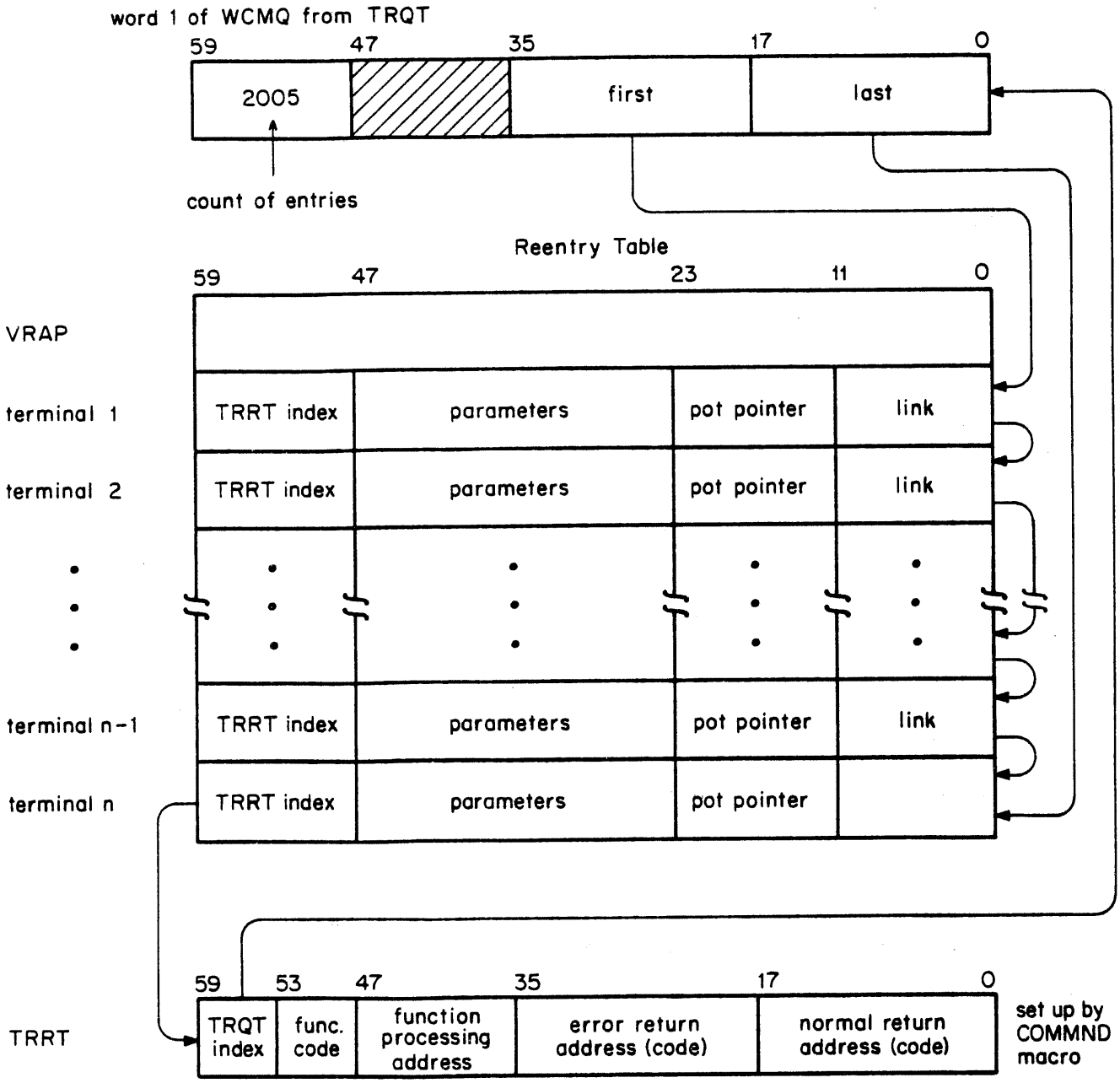


Figure 15-11. Table Relationships

QUEUE PROCESSING

Processing of queue entries is done by the PCS subroutine. As entries are completed, PCS extracts the normal or error return address and jumps to it. Making queue entries is done by a jump to PCS4 or PCS6. Before returning to a routine, PCS calls SSP which sets up the following registers (from the queue entry, bits as shown).

Register	Description
A0	FWA of user's terminal table entry
B2	Terminal number (bits 11-0)
B3	Pot pointer (extracted from byte 3 of entry in reentry table) (bits 47-24)
B4	FWA of pot pointed to by B3 (B3*10+VBMP)
X7	Bits 47-24 of reentry table entry

These A and B registers are generally not changed within the various subroutines of TELEX.

TELEX ROUTINES

The following is an outline of the subroutines comprising TELEX.

- MUXP - Multiplexer table (RA + 101B)
- TRQT - Table of queues:

WCMQ	ITAQ
TIMQ	ITOQ
JOBQ	PFMQ
SORQ	

- TEL - Control loop; calls the following:

CSF	PPU	SPR	TSR
CTB	RPC	STM	URT
DRI	SCH	STR	
EPP	SOR	TDQ	

- STR - Process requests to handle output to terminal by calling the following subroutines:

ASO	DSD	SCS
CDT	PTY	SDT
DRT	SBS	

- CSF - Checks SALVARE file user time out
- DRI - Process driver (1TD) requests by calling the following subroutines:

AOD	DLO	HUP	MAL	SAI	TTI
CSC	DRP	IAM	MTN	SKY	
CLI	DRT	LOF	RES	SPT	
DIN	EMO	LPT	RIN	SSC	

- PCM - Process terminal commands (called from CLI, AUT); calls following subroutines:

ACC	DIA	LIS	REP	SUB
ASC	EDI	MTR	PER	TAP
ATT	FDP	NOR	ROT	TER
AUT	GET	NOD	RUN	TXT
BAT	HEL	NOS	SAV	UNS
BIN	HDP	PAC	SOF	UNU
BYE	LAN	PAR	STA	XEQ
CLR	LEN	PFC	STO	

- Reentrant command processing routines:

BJB	IEX	IUA	IAF	PUR
BJS	INJ	PBS	PFF	RDY
EJB	IPF	PSS	PFM	
IDT	IPL	DAF	PFP	

- PCS - Process queue entries
- PPU - Process PP requests

- RPC - Refill pot chains
- SCH - Build job queue entry for scheduling a job
- SOR - Set up for scheduling SORT job
- SPR - Call 1TA to adjust field length
- TDQ - Process time - delay queue
- TSR - Process WCMQ; reenter the following:

DCR	ITA	MJE	SRE
HNG	ITO	MTO	SSO
ICH	JOB	REC	
INP	LIN	SEN	

- General subroutines including:

ABT	CPF	GPL	MQE	SFL
BRQ	DAP	GQE	MVA	SLF
CCM	DMP	GRT	O6S	SRC
CFL	DPT	GTA	PCB	SRR
CJT	ENP	GZP	RPL	SSP
CLE	GEM	ISH	RPT	TPF
COI	GFN	LTT	SAF	UPF
COP	GFS	MDA	SCT	UQS

- Transaction routines including:

  Transaction executive driver routines
  Transaction executive interface routine
  General subroutines

TELEX2 - TERMINATION OVERLAY

TELEX2 performs exit processing for the TELEX subsystem. It is loaded whenever an abnormal condition is detected or when the operator types 1.STOP to drop the subsystem.

When an abnormal condition is detected within TELEX1 processing, a jump to the abort subroutine (ABT) is executed. ABT issues the message

TELEX ABNORMAL - xxx.

where xxx is the name of the subroutine calling ABT.

After issuing this message, if sense switch 3 is on, the ABORT macro is used to abort the control point. Routine 1AJ senses the EXIT control statement, the next control statement (TELEX2) is found, and 1AJ has the termination routine loaded. TELEX2 is loaded in a buffer which dumps the FL if requested and loads TELEX3 which leaves the tables and queues untouched. Basically, TELEX2 logs out all active users so that there will not be any time-sharing jobs left in the system. After issuing system statistics, 1TD is called to restart the time-sharing subsystem depending upon sense switch settings. (There are 6 sense switch options for TELEX. Refer to the NOS Operator's Guide for more information.)

MULTIPLEXER DRIVER

Routine 1TD performs communication between TELEX and terminals (accessed via the 6671 and 6676 multiplexers) and the NOS stimulator. It has the capability to communicate with most ASCII compatible terminals and correspondence code compatible terminals such as the IBM 2741, NOVAR 541, and CDC 713 terminals, if the multiplexer has the required options installed.

Routine 1TD processes up to 512 (10 character per second) terminals. The number of terminals for which performance can be guaranteed will decrease as the terminal speed is increased. In any event, the total driver capability is 5120 characters per second. The maximum terminal speed which may be accommodated is 60 characters per second.

Terminal communication is processed in a half-duplex mode. A line is generally the unit of transmission in each direction. Interruption of continuous output is provided along with an input line and character deletion facility.

Communication between 1TD and TELEX is accomplished by means of a circular request queue provided by TELEX. Routine 1TD inserts a request in the queue and TELEX removes the request as it is processed.

Terminal control operations for ASCII terminals include the following.

- Complete an input line by pressing the return key. A line feed is not needed, since the driver issues one to the terminal.
- Delete or ignore an input line by pressing the ESC key.
- Delete a previously entered character by pressing the underline (back arrow on some Model 33 teletypewriters, backspace on the CDC 713).
- Terminate output by pressing the BREAK key, or the S key.
- Interrupt output by pressing the I key. Output may be resumed by pressing P followed by return.

Terminal control operations for correspondence code terminals include:

- Complete an input line by pressing the return key.
- Delete or ignore an input line by pressing the ATTN key
- Delete the previously entered character by pressing back space.
- Terminate output by pressing the ATTN key.

Routine 1TD consists of routines 1TD and 2TD. Routine 1TD is the initialization (and termination) routine that loads the 2TD overlay. The 2TD overlay is normally loaded and executing in the PP while the TELEX subsystem is servicing terminals. Several other overlays are assembled with 1TD. These are the translation tables for the various terminals listed in table 15-5.

TABLE 15-5. TRANSLATION TABLES OVERLAYS

Overlay	Terminal Type
9JA	ASCII terminal, 64 CS
9JC	Correspondence/text, 64 CS
9JE	Correspondence/APL, 64 CS
9JG	Memorex 1240/APL, 64 CS
9JI	ASCII block edit terminal, 64 CS

Figure 15-12 shows the multiplexer servicing concept as being similar to the hardware slot and barrel concept for peripheral processors. Up to eight multiplexers are serviced by the driver and each port is allotted a time slice in which the driver peforms I/O and required overhead.

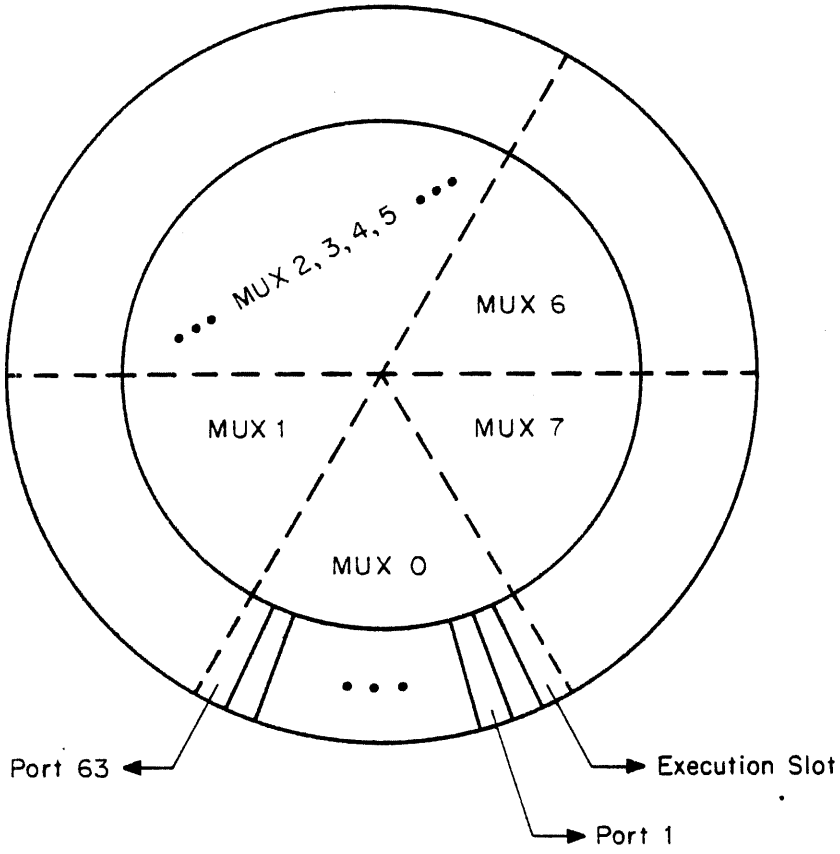


Figure 15-12. Multiplexer Servicing Concept

DRIVER INITIALIZATION (1TD)

The multiplexer driver is initialized by the overlay 1TD. This overlay consists of the following USE blocks.

Block	Description
MAIN	Initialize TELEX control point
PRESET	Load 2TD
RESIDENT	Code resident during execution

The lengths of these blocks are determined by the difference between their last word address and their first word address as shown in table 15-6.

TABLE 15-6. USE BLOCK LENGTHS

Last	First	Description
MANE	MANF	Length of MAIN
RESE	RESF	Length of RESIDENT
PRSE	PRSF	Length of PRESET

These three lengths are added and the sum is subtracted from 4096 to establish the origination (ORG) address. The multiplexer input buffer (IBUF) is defined in PRESET and must follow the PP resident translation tables. A check for this overflow condition is made at the end of the 2TD overlay.

Overlay 1TD is loaded when the operator types TELEX to start the time-sharing executive and during termination to perform certain post processing operations. Routine 1TD is called by 2TD from the DRP subroutine. Since 1TD is loaded above the translation tables, much of 2TD is overlayed when it calls 1TD. Routines overlayed include some write mode processing (WTM), all polled line processing routines, and all of the utility subroutines. In addition, the translation tables and the multiplexer input buffer are overlayed. Figure 15-13 shows the relative load addresses of the USE blocks comprising 1TD, as well as the 2TD overlay while executing.

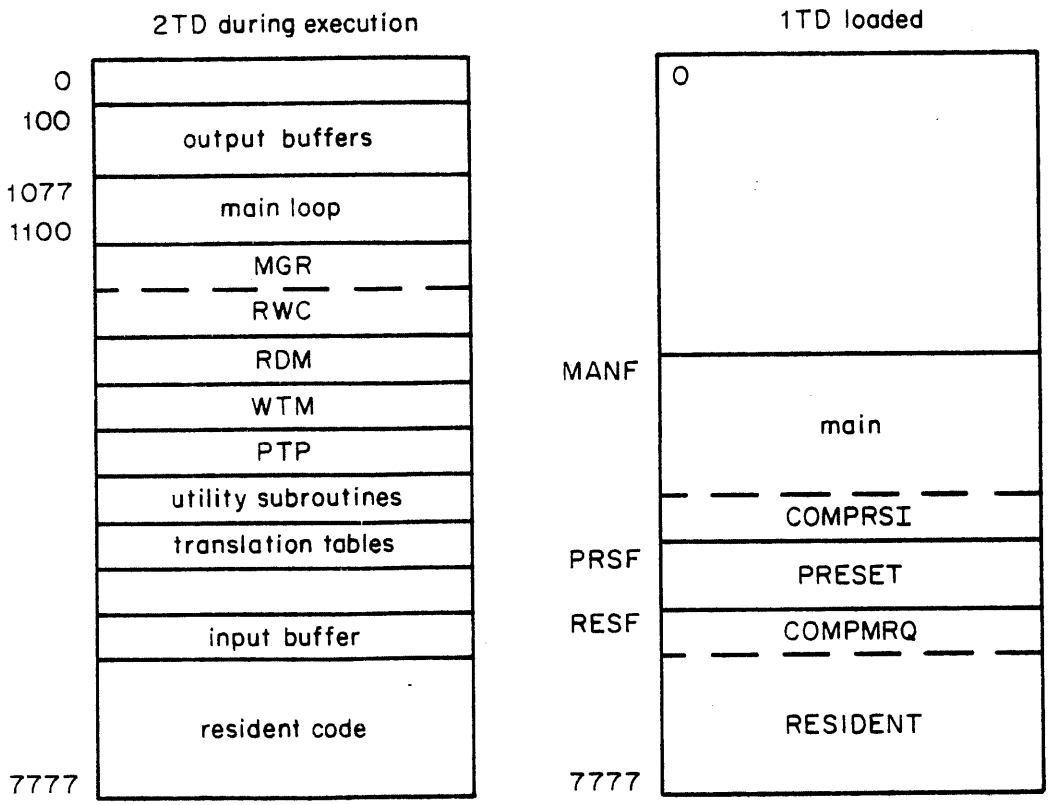


Figure 15-13. 1TD/2TD Memory Maps

Figure 15-14 provides an overview of the initialization processes in blocks MAIN and PRESET. Resident code is used by 2TD during termination processing.

Data in the multiplexer input and output buffers within 2TD consists of an 8-bit character per port along with control bits as shown in figure 15-15.

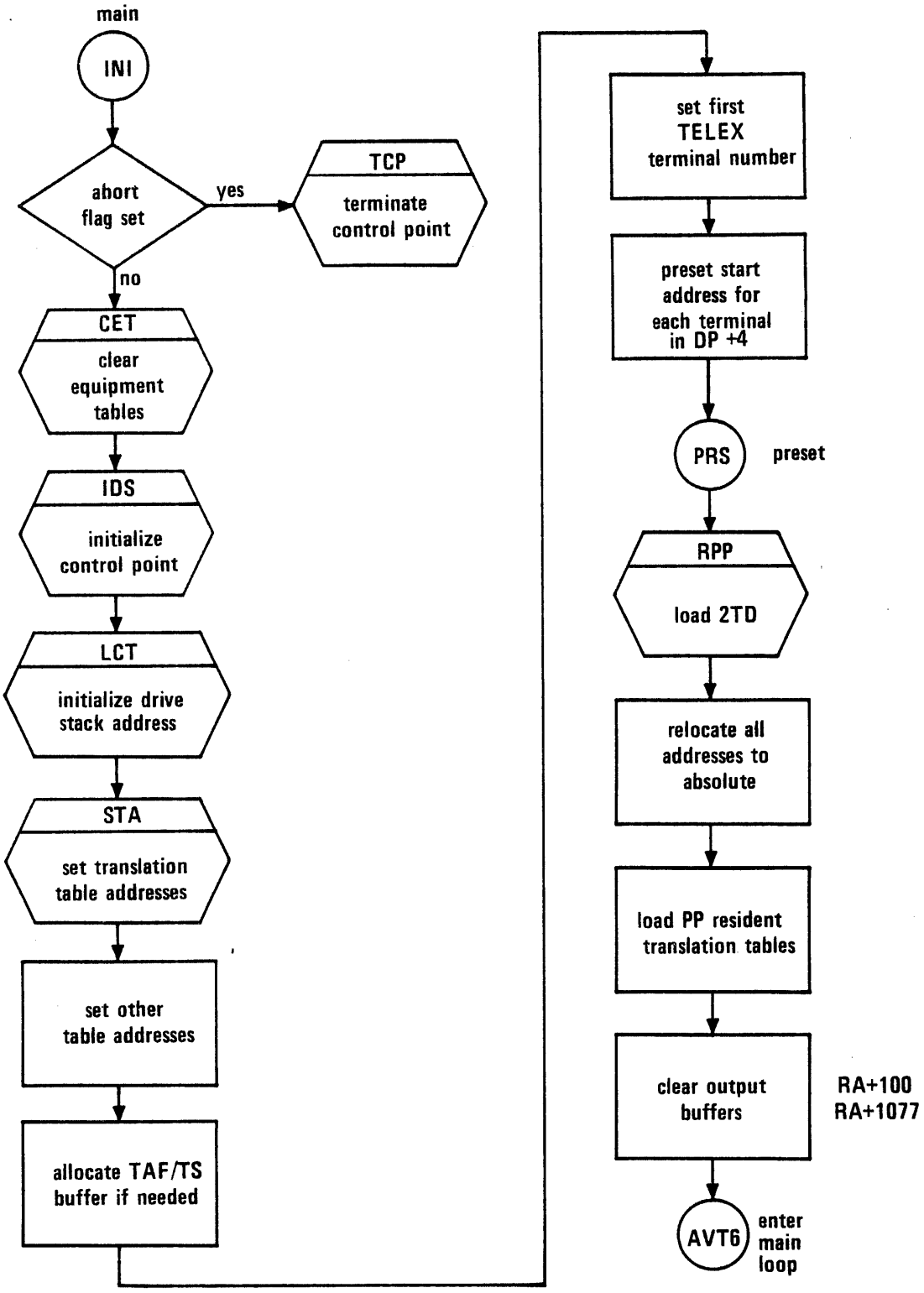


Figure 15-14. MAIN and PRESET Overview

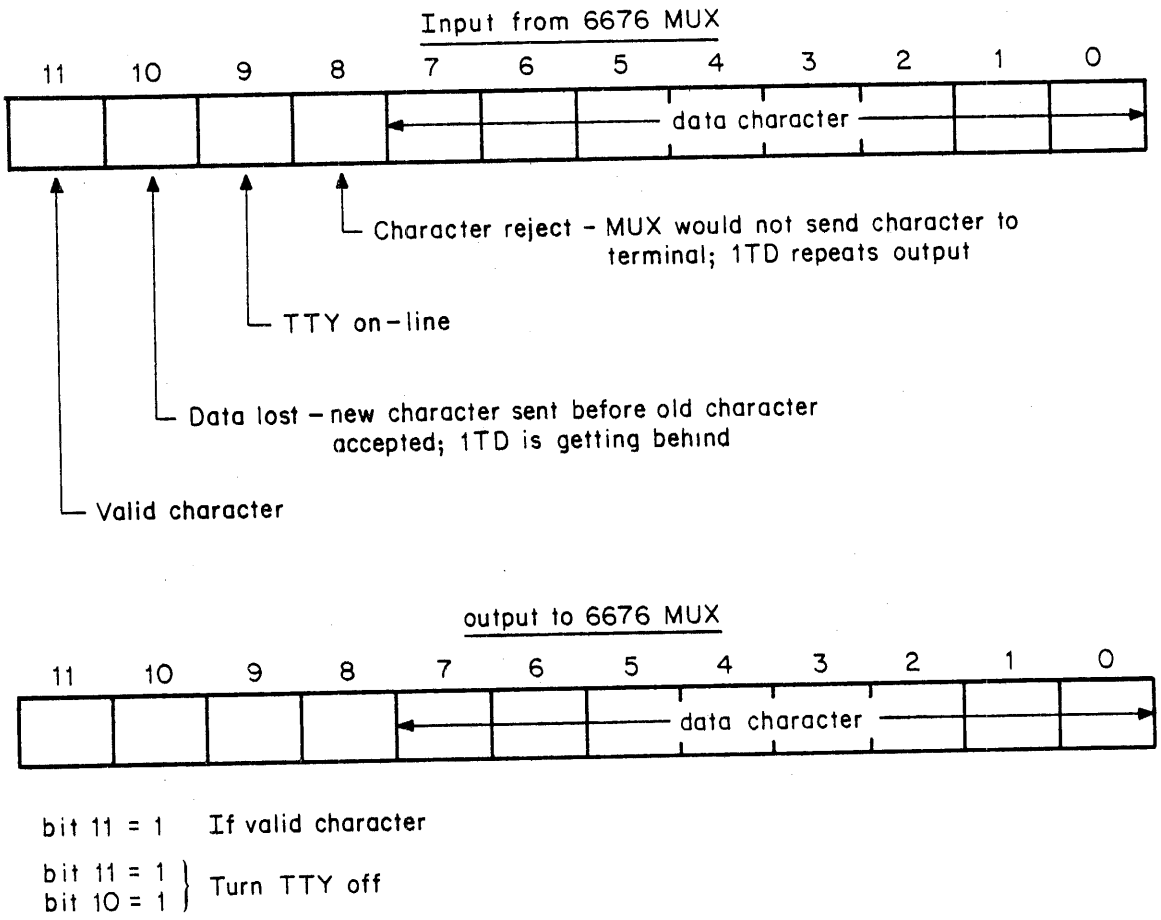


Figure 15-15. Input/Output Buffers

Figure 15-16 describes the logical breakdown of the 2TD driver while executing.

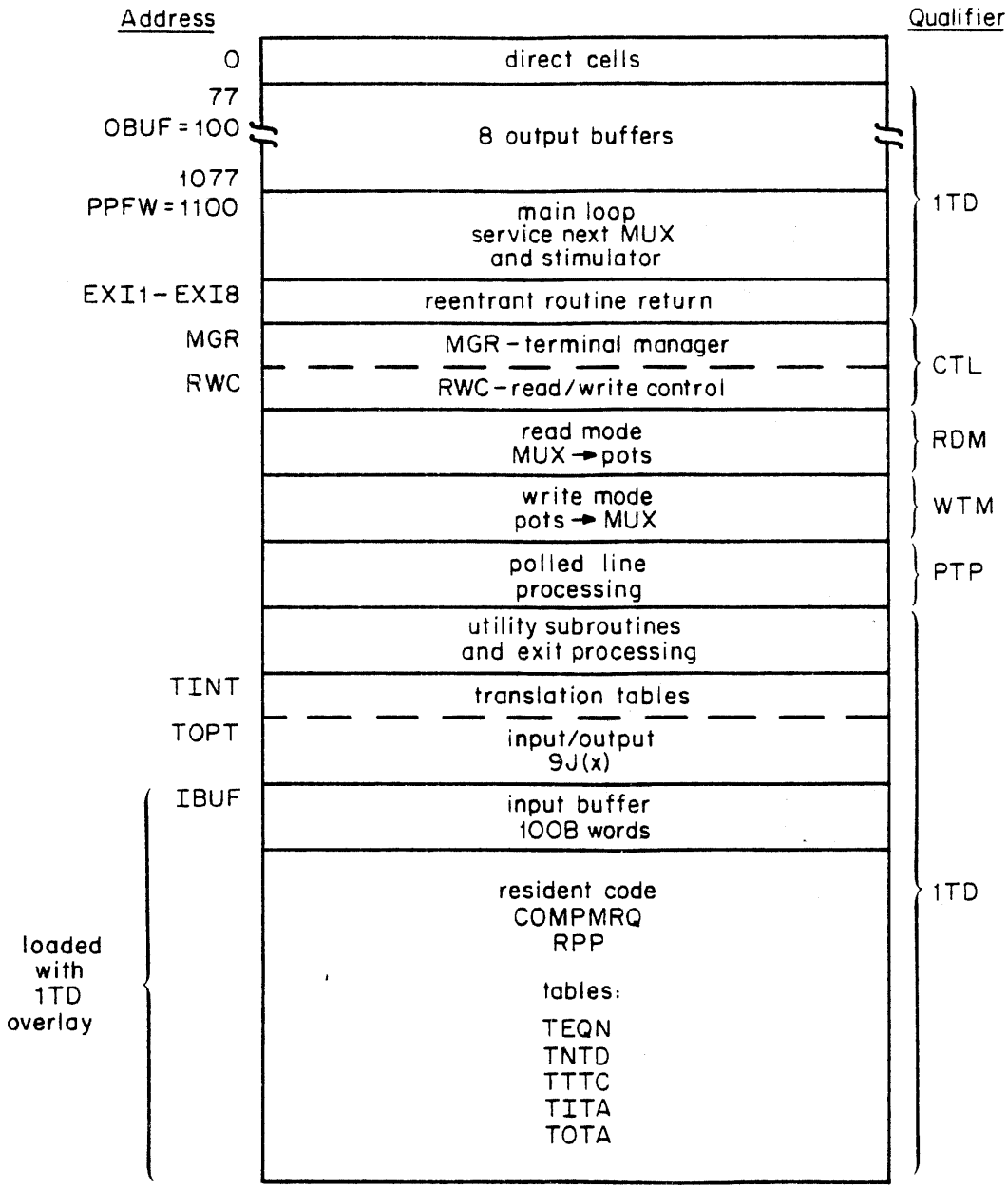


Figure 15-16. 2TD Memory Map

REENTRANT ROUTINE RETURNS

The reenrant routine returns are eight entry points which are jumped to by any subroutine that cannot complete its function in a single time slice. The RETURN macro is the most common method used throughout the routine for the purpose of setting a return address. Control is returned to the next instruction or to another specified routine address. For example,

```
RETURN EXI7
```

enables control to be returned at the next instruction, while

```
RETURN EXI3,LIN
```

causes control to be set to the LIN subroutine for the next time slice for this port. In any case, the EXI(x) specifies a reentrant return address. If x is odd, the reentry address is in the A register and stored in DP+4 (that is, VDPT, byte 4). If x is even, no return address is given, and control is returned to the previous return address in DP+4. The reentrant return addresses and terminal table words updated are shown in table 15-7.

TABLE 15-7. ADDRESSES AND WORDS

Reentrant Return Address	Terminal Table Word(s) Written
EXI1, EXI2	VDPT
EXI3, EXI4	VDPT, VCHT
EXI5, EXI6	VDPT, VCHT, VDCT
EXI7, EXI8	VDPT, clear byte in output buffer

Direct cell assignments are explained in the program. During execution VDPT, VCHT, and VDCT are available in direct cells. VDCT is read and updated only when necessary to minimize CM reads and writes.

The main loop controls the advancement to the next multiplexer, performs multiplexer I/O, checks for stimulator processing, and enters the manager (MGR subroutine).

PROCESS SUBROUTINES

The MGR subroutine processes individual ports and satisfies requests from TELEX. A flowchart of MGR is shown in figure 15-17. The symbol qualifier CTL contains the following routines.

Routine	Description	
MGR	Terminal manager	
CIS	Check interrupt status	
INT	Process interrupt	
CTO	Check time out	
DIN	Dial-in processing	
HUP	Hang up phone	*1
OFL	Process user off line	
RWC	Read/Write control	
DTT	Determine terminal type	
LIN	Process login	
RAB	Read answerback drum	
1TD	Function codes for the processor TFR	*2
TFR	Process TELEX functions with the following subroutines:	

Routine	Description
BGI	Begin input
CFD	Clear full duplex flag in VDPT 1TD function values for byte 4 of VDCT.
HUP	Hang up phone
IIP	Issue input prompt (question mark)
LGI	Process login
SNM	Set normal modes
SOP	Set odd parity
SEP	Set even parity
SFD	Set full duplex flag
STT	Set terminal type

---

*1 VDPT word, DP+4 gets one of these addresses.
*2 TELEX requests 1TD to perform certain functions by setting bit 11 of byte 0 of VDPT and the function code in byte 4.

The symbol qualifier RDM contains the following read mode subroutines:

Routine	Description
BRD	Binary read
CRD	Correspondence read (APL type, NOVAR)
ARD	ASCII read

These routines call RTC which translates the input character and stores it in a pot.

If the input character is a special character, one of the following subroutines is called.

Routine	Description
CES	Check escape status
CRT	Process carriage return
DLN	Line delete
DPC	Delete previous character
NLI	Null input
CSF	Case shift
NWL	New line
EOT	End of transmission
BRK	Break
ECI	Escape character input
CAN	Cancel line block
EOL	End of line (block edit)
ETX	End of text

CRT, BRK, CAN, and NWL call EIL for end of line processing which calls CLI for command line input and SLI for source line input.

CLI calls ACL for ASCII end of command line or CCL for correspondence end of command line. SLI calls ASL for ASCII end of source line or CSL for correspondence end of source line.

General subroutines used by RDM are as follows.

Routine	Description
ITM	Issue terminal message
NIP	No input pot available
DLO	Process lost data
TIC	Translate input character
WIC	Write input character
NLI	Process null input
NWL	New line (unit for EOT from terminal)

Normal read mode processing starts with the RDM subroutine which sets the return address in DP+4 to BRD, CRD, or ARD. As characters are received from the multiplexer, they are processed by RTC which calls TIC to translate them and then calls WIC to write them in pots. The normal exit is to EXI4. Figure 15-18 shows the general relationship of the read mode processing subroutines.

The symbol qualifier WTM contains the subroutines used for write processing. These subroutines are structured similar to RDM subroutines and include the following.

Routine	Description
BWT	Binary write
CWT	Correspondence write
AWT	ASCII write

These subroutines call WTC to write the terminal character by using the following subroutines.

Routine	Description
ROC	Read output character from pot
TOC	Translate character

A special character is processed by one of the following routines.

Routine	Description
NLO	Null output
ANL	ASCII terminal new line
ACR	ASCII terminal carriage return
CNL	Correspondence end of line
CCR	Correspondence carriage return
CLF	Correspondence line feed
CBS	Correspondence backspace

Other write mode general subroutines include the following.

Routine	Description
CMM	Process monitor mode
SOC	Set output control
SRC	Send repeated character

SOC restarts a job to get more output and processes output control bytes by jumping to one of the subroutines listed in table 15-8.

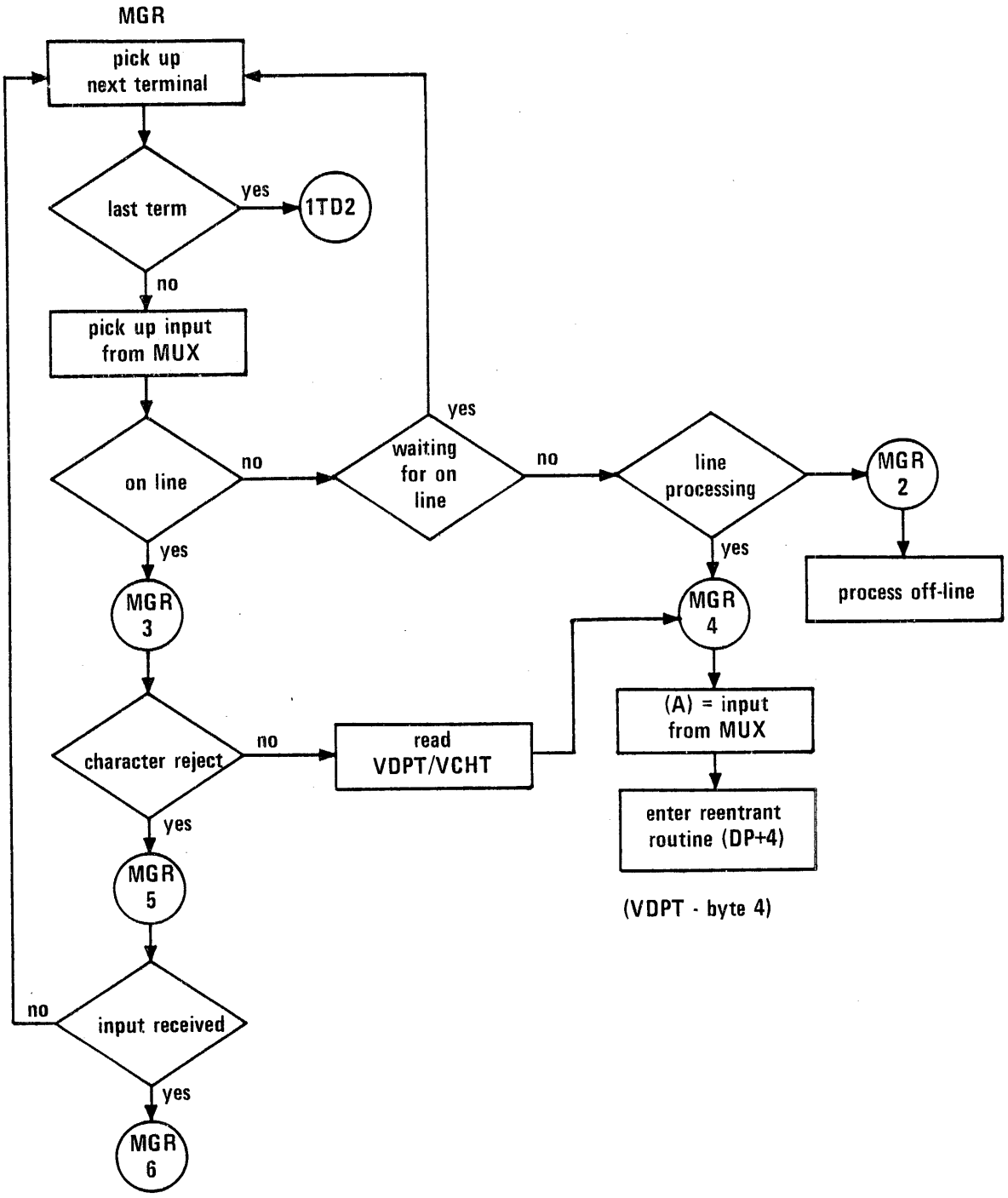


Figure 15-17. MGR Flowchart

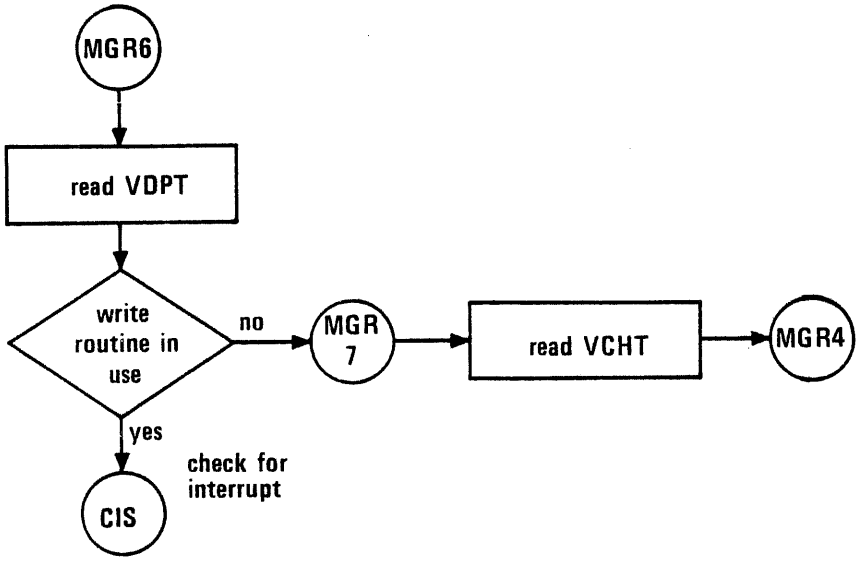
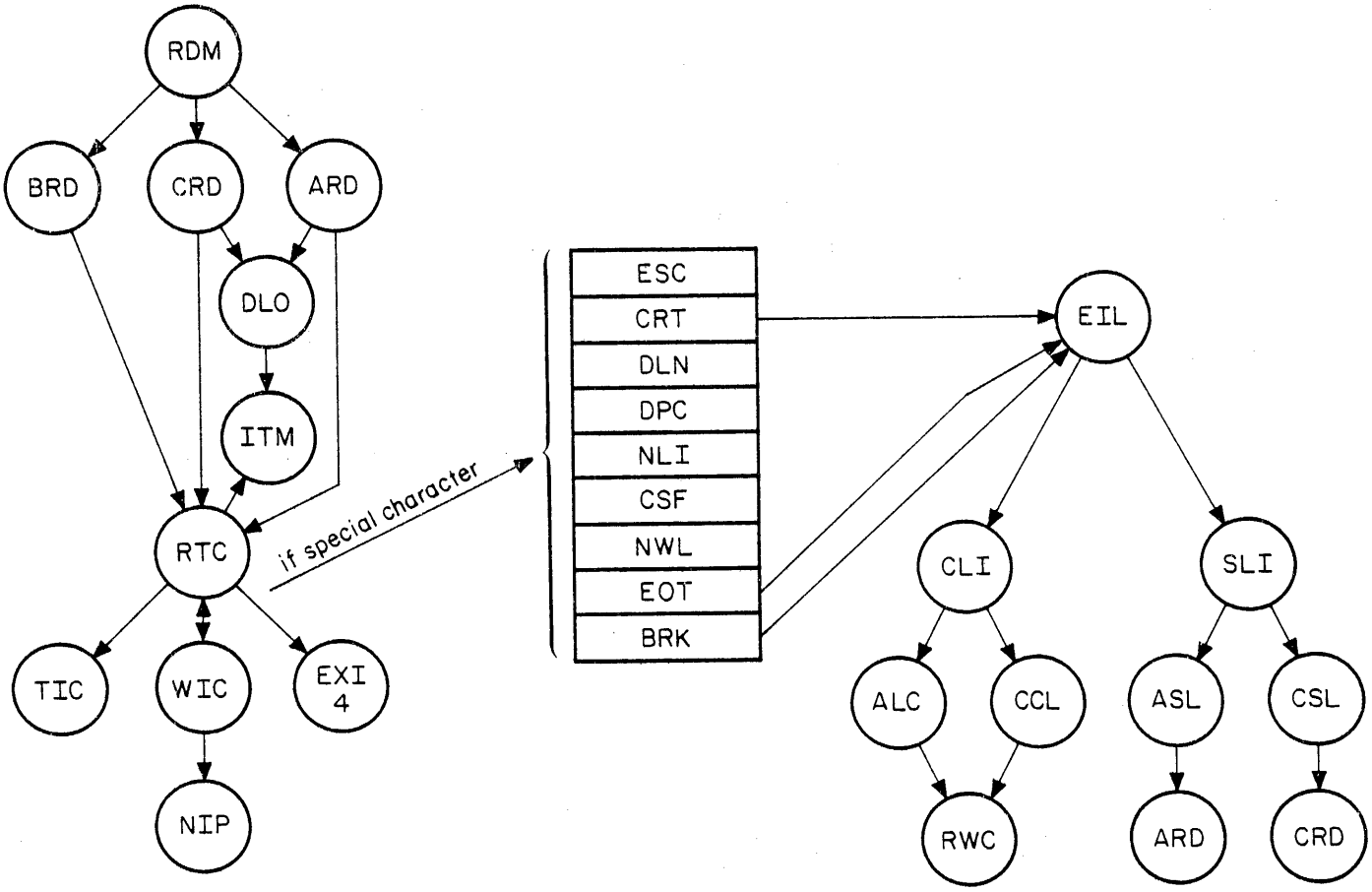


Figure 15-17. MGR Flowchart (Continued)



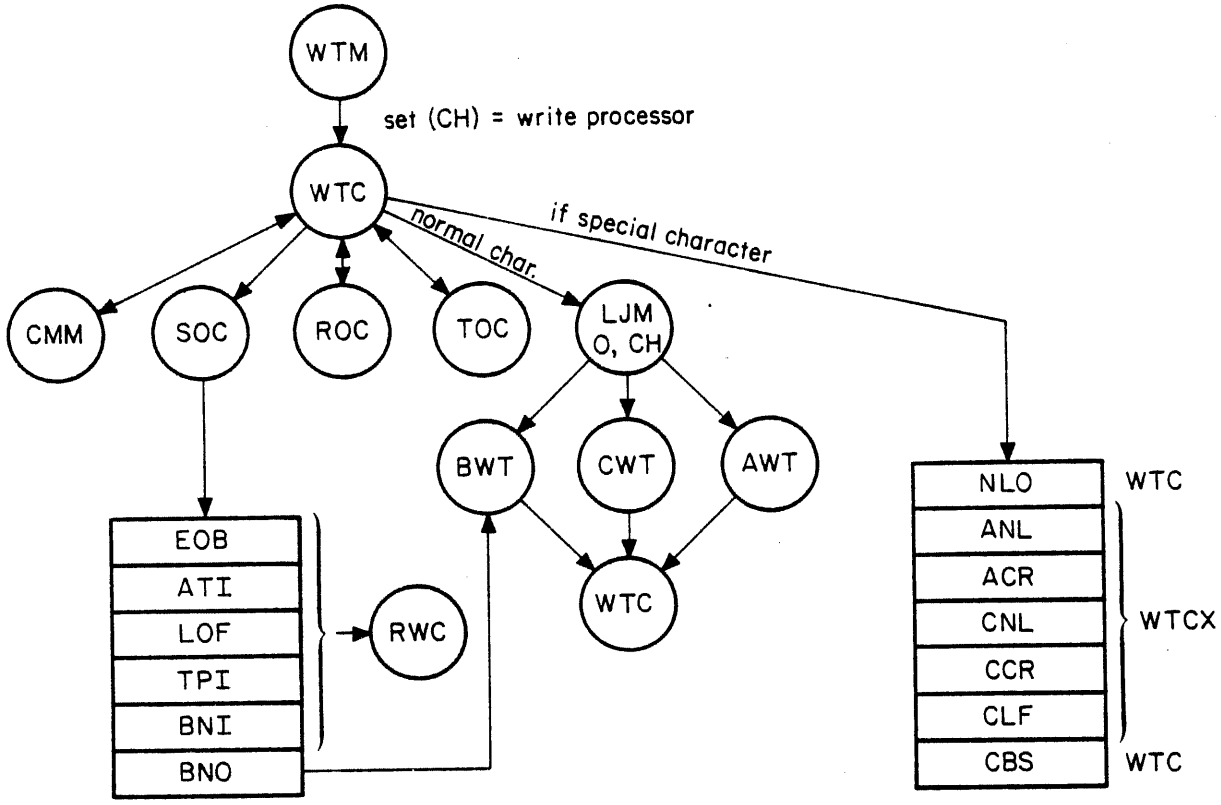
*1 A line with two headed arrows represents a subroutine entered via RJM instruction

Figure 15-18. Read Mode Processing Subroutines

TABLE 15-8. CONTROL SUBROUTINES

Control Byte	Subroutine Name	Function
0000	EOL	End of line
0001	EOB	End of block
0002	ECB	End of correspondence block
0003	ATI	AUTO input
0004	LOF	Log off user
0005	TPI	Set transparent input (allows all characters to be transmitted to the CPU program)
0006	BNI	Set binary input
0007	BNO	Begin binary output
0010	ETM	End of transaction message
0011	BEO	Begin extended output
0012	MTO	End o marked transaction output
0013	EOS	End of string

The relationship between the write mode subroutine is shown in figure 15-19.



*1 A line with two arrows indicate a return jump.

Figure 15-19. Write Mode Processing Subroutines

The symbol qualifier PTP contains the routines used to process polled lines. These include the following.

Routine	Description
SPL	Sense polled lines
RPR	Read poll response
PTR	Process terminal response
SSC	Set sequence count

Utility subroutines are under the symbol qualifier 1TD and are general subroutines used by the other routines described previously. The utility subroutines are as follows.

Routine	Description
BUP	Back up pointers
CUT	Clean up terminal tables
ERQ	Put entry in TELEX's request queue
RDC	Read VDCT entry
RLT	Read link table to get next pot in chain
RPC	Read previous character in pot
SCA	Set control address (for instance, RDM uses this to set + read routine BRD, CRD, or ARD depending upon translation table)
WTO	Wait time out
SWA	Set address of current word of current pot
TCH	Translate characters

Exit processing routines include MXE to process multiplexer errors and DRP to process driver exit (call resident code set up by 1TD at initialization time).

1TA TELEX AUXILIARY ROUTINE

Routine 1TA processes functions for TELEX which require PP action. The functions allowed are listed in table 15-9.

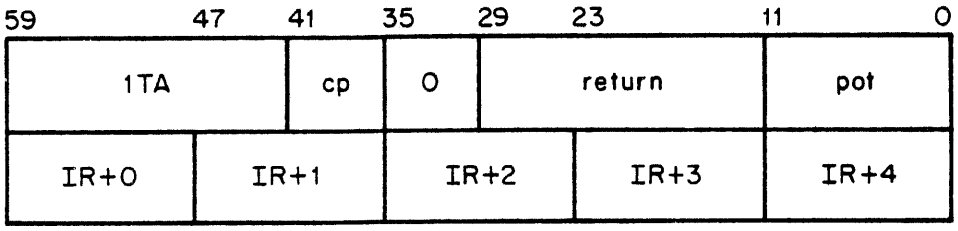
TABLE 15-9. PROCESS FUNCTIONS

Overlay Name	Function Code	Routine Name	Description
3TA	1	TFL	Adjust TELEX field length
3TB	2	RTJ	Return terminal job
3TC	3	CRF	Create rollout file
3TD	4	TLP	Terminal logout processor
3TE	5	DAM	Display accounting message
3TF	6	TRP	Terminal recovery processor
3TG	7	IRL	Increment resource limit
3TH	10	RFP	Recovery file processor
3TI	11	SJS	Sort and job scheduler
3TJ	12	GST	Gather statistics
3TK	13	CUS	Clear up SALVARE file

TELEX calls 1TA in one of two ways.

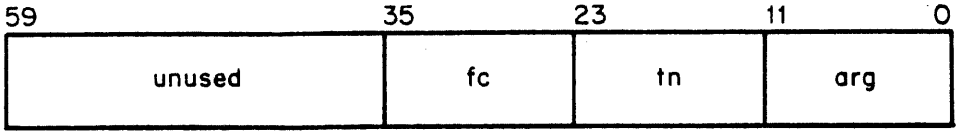
GROUP REQUEST

A group of requests are stored in pots. The input register format is as follows.



return	Upper 24 bits of the word specified are set to zero upon completion of all requests.
cp	Control point number
pot	Pot containing the list of requests

The requests are one word each with the following format:

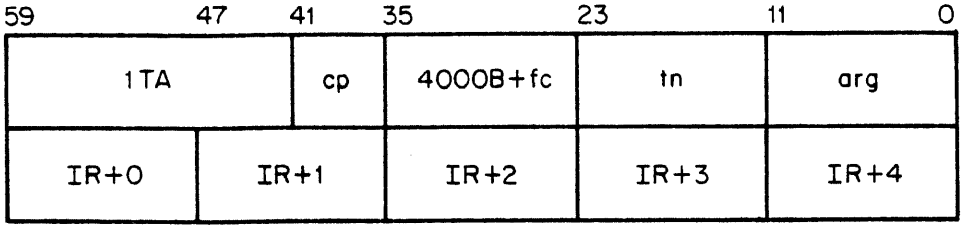


fc	Function code
tn	Terminal number
arg	Pot pointer or request type

The list of requests is terminated with a zero word.

SINGLE REQUEST

A single request is denoted by setting bit 35 in the input register which is formatted as follows.



cp	Control point number
fc	Function code
tn	Terminal number
arg	Pot pointer or parameter (depending on function)

Routine 1TA uses several bits in VROT of the terminal table. These bits are:

Bit	Description
0	Completion status bit
4	Set to indicate recall function by TELEX
10	Purge rollout FNT's
11	Error return

Figure 15-20 is the flowchart of the initialization, execution, and termination of the control loop for 1TA.

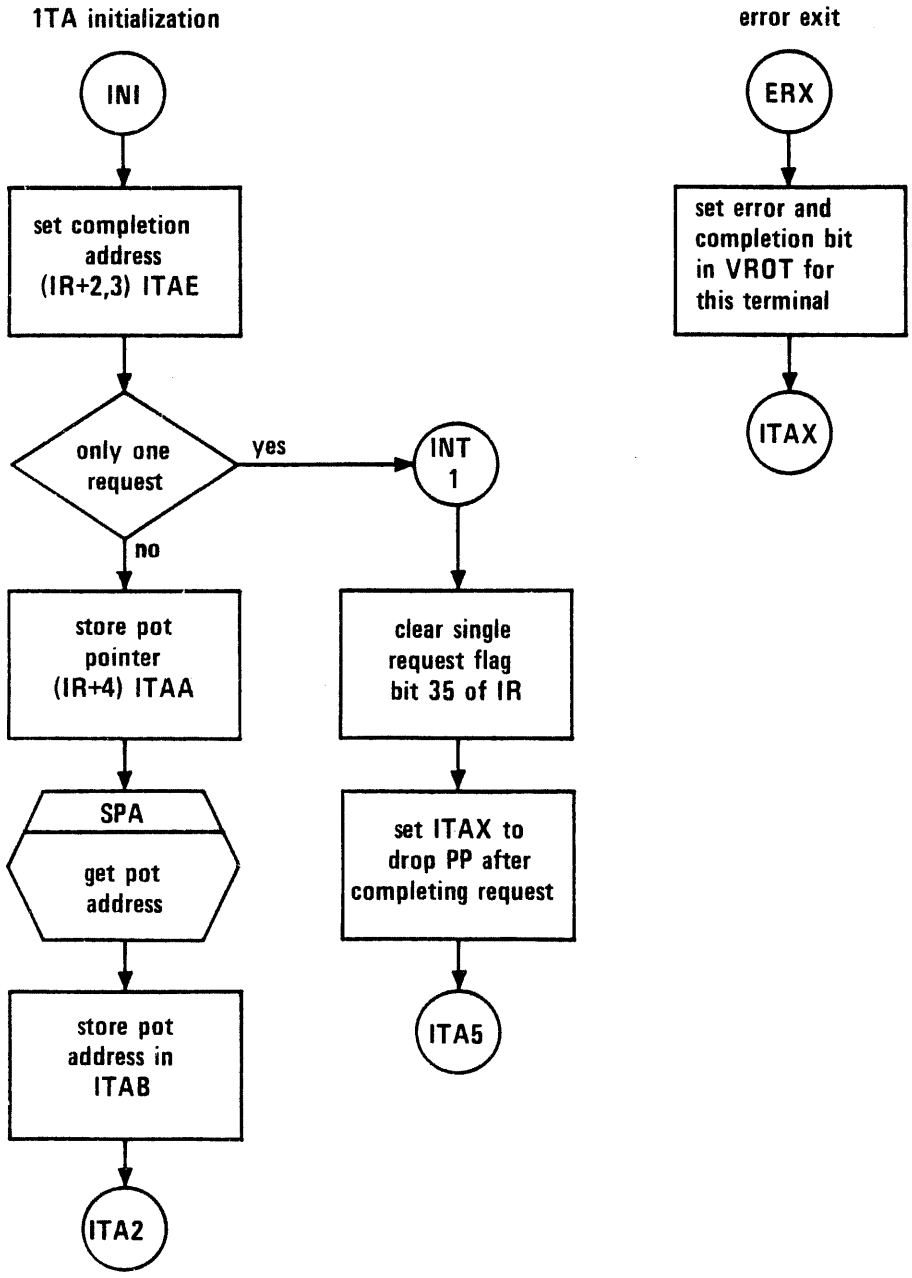


Figure 15-20. 1TA Control Loop

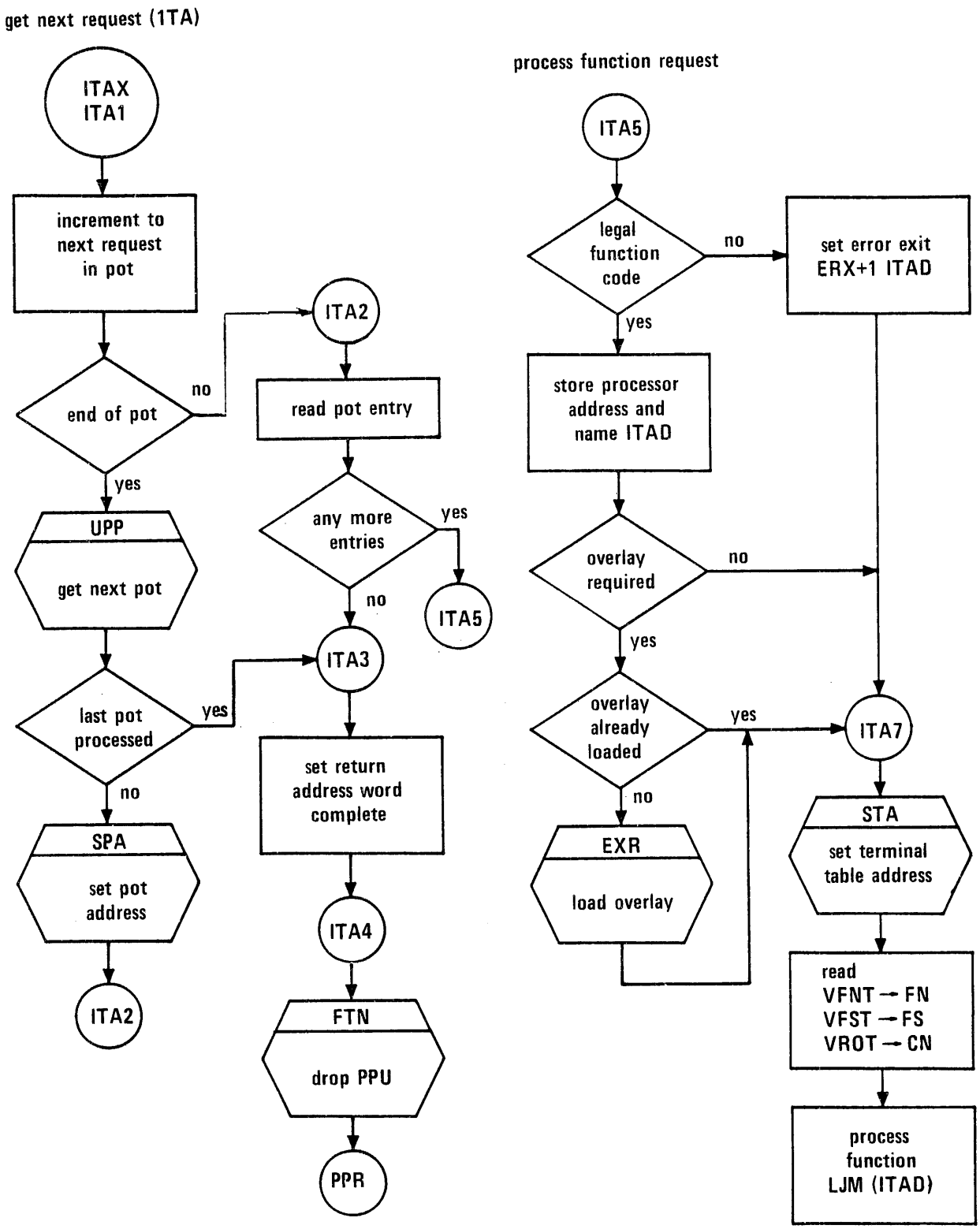


Figure 15-20. 1TA Control Loop (Continued)

Function 5 is used to create a rollout file for a time-sharing job. The format of the rollout file is given in figure 15-21.

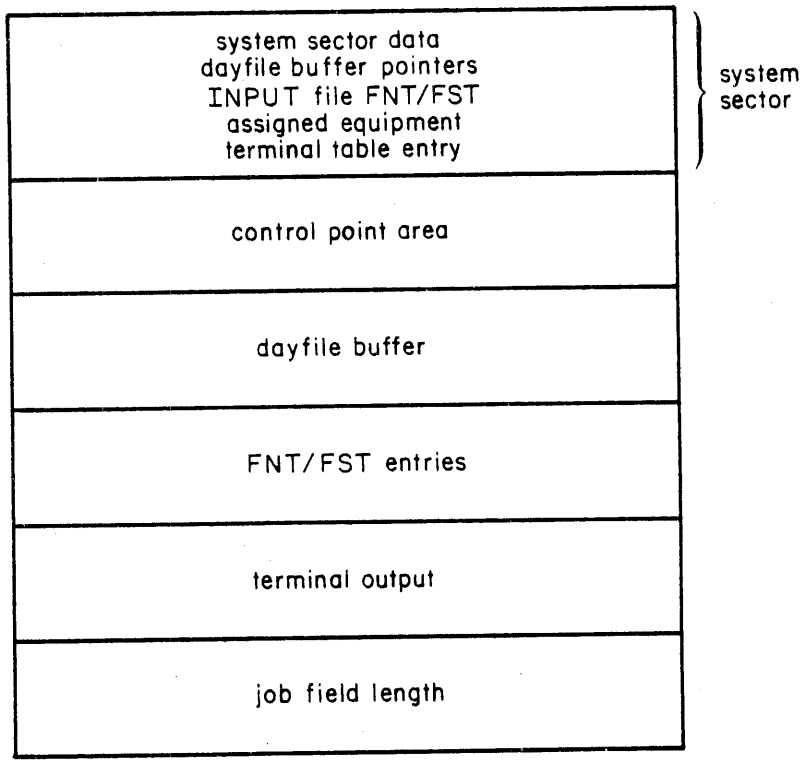


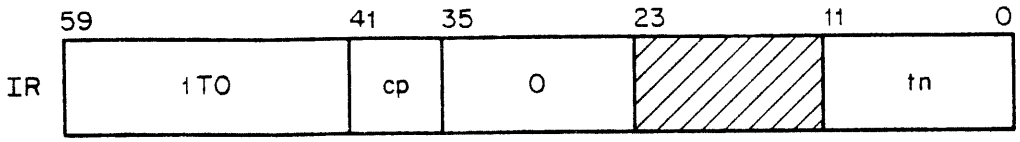
Figure 15-21. Time-sharing Job Rollout File

1TO - TTY INPUT/OUTPUT ROUTINE

Routine 1TO is called by TELEX to process a queue of requests for terminal input and output which require disk accesses. The queue resides in pots within the TELEX field length. The queue has been sorted by TELEX in order of equipment and disk addresses so as to minimize disk time. If there are requests for more than one mass storage device, the entries are processed for the first device available.

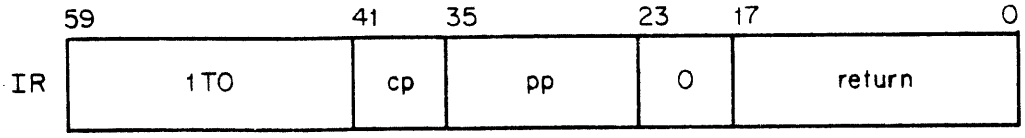
Routine 1TO is also called by 1RO to handle the first section of data on a rollout file. This data is passed to 1TO in a PP buffer. Routine 1TO dumps the PP buffer into pots and makes a VASO request to TELEX for that terminal.

The input register format when 1TO is called by 1RO as follows.



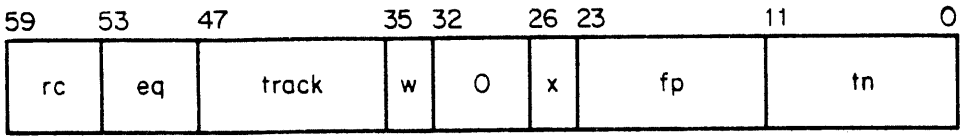
cp	TELEX control point number
tn	Terminal number

The input register when called by TELEX is as follows.



cp	TELEX control point
pp	POT pointer to first POT of requests
return	Location of completion status word

The request in POTs are one word entries with the following format.



rc	Request code 0 Correction dump 1 Output data
eq	Equipment number
track	First track of file if rc = 0; current track if rc = 1
w	Number of words in last pot (0 means 10); w is meaningful when rc = 0
x	Number of pots to dump; rc = 0
fp	First pot of source or output
tn	Terminal number

As a group of requests is completed, the above entries are updated by setting byte 2 to the last pot to be dropped or assigned. These requests are then written back in the same pot from which they came.

The flowcharts of 1T0 (figure 15-22) shows that it is broken down logicaly into the following sections.

- Preset or initialization
- Main loop (get next request)
- ICH subroutines (correction handler if rc = 0)
- PR0 subroutines to process output if rc = 1 (that is data flow is disk to pots to terminal)

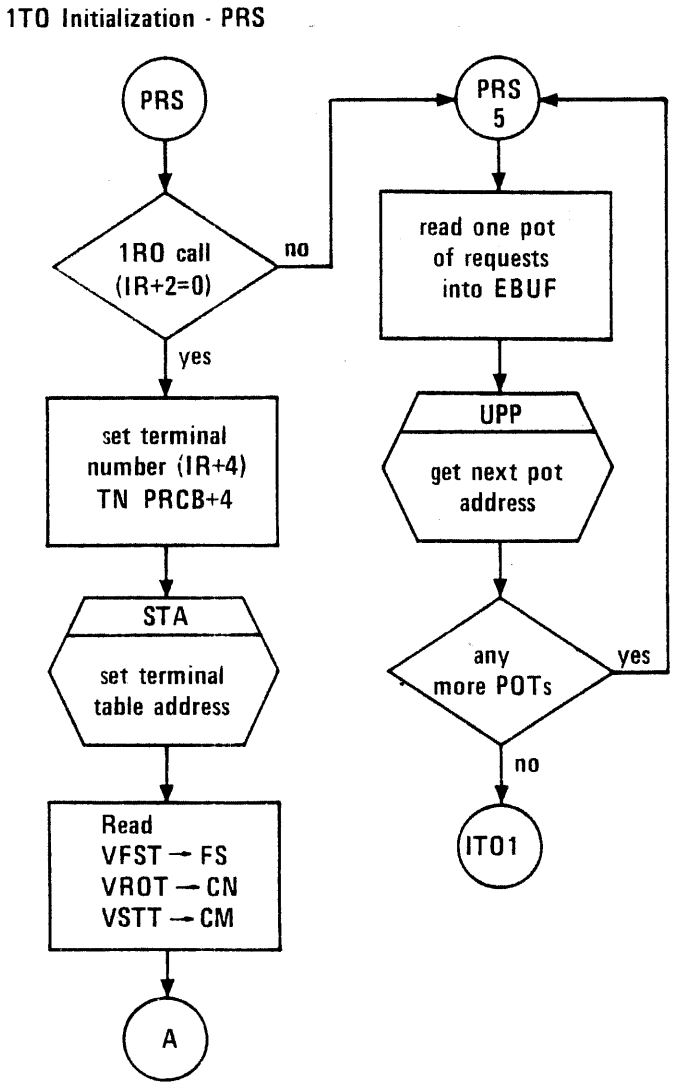


Figure 15-22. 1TO I/O Routine

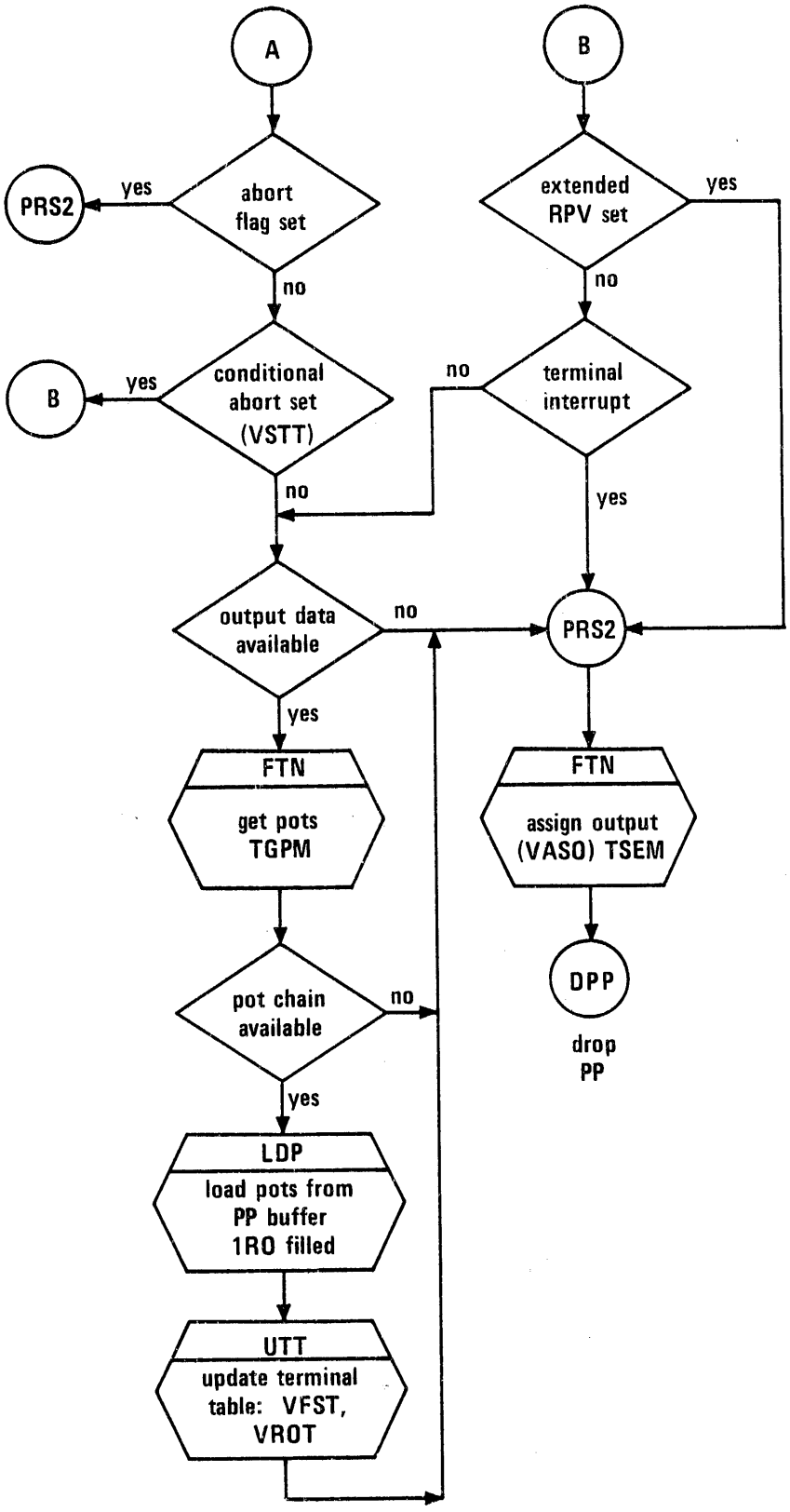


Figure 15-22. 1TO I/O Routine (Continued)

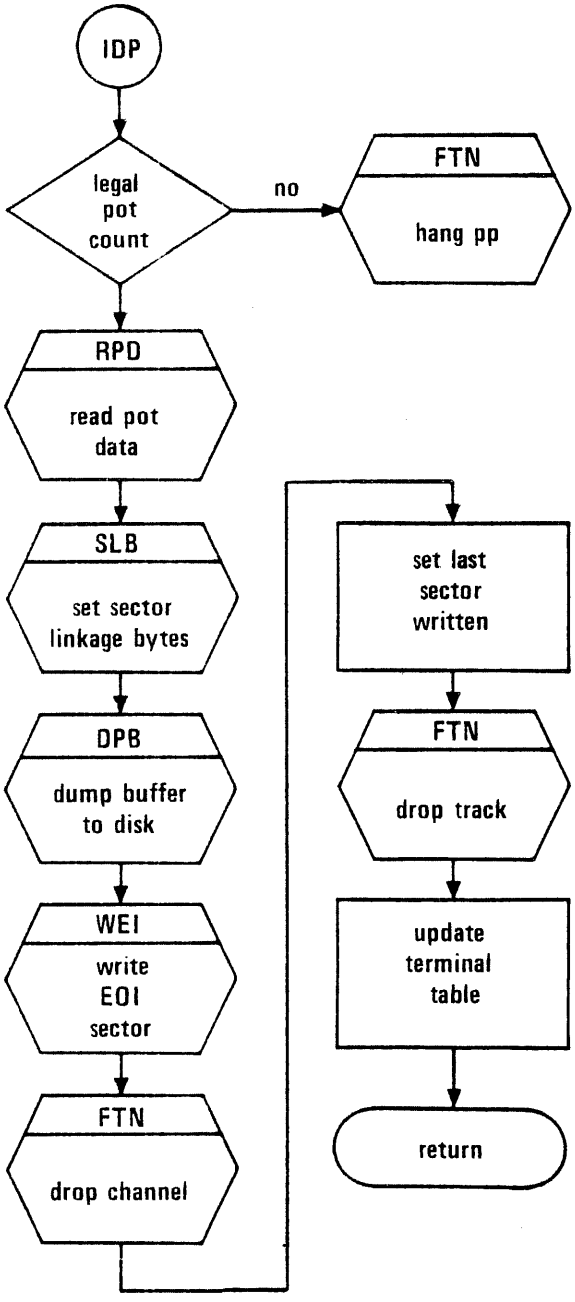


Figure 15-22. 1TO I/O Routine (Continued)

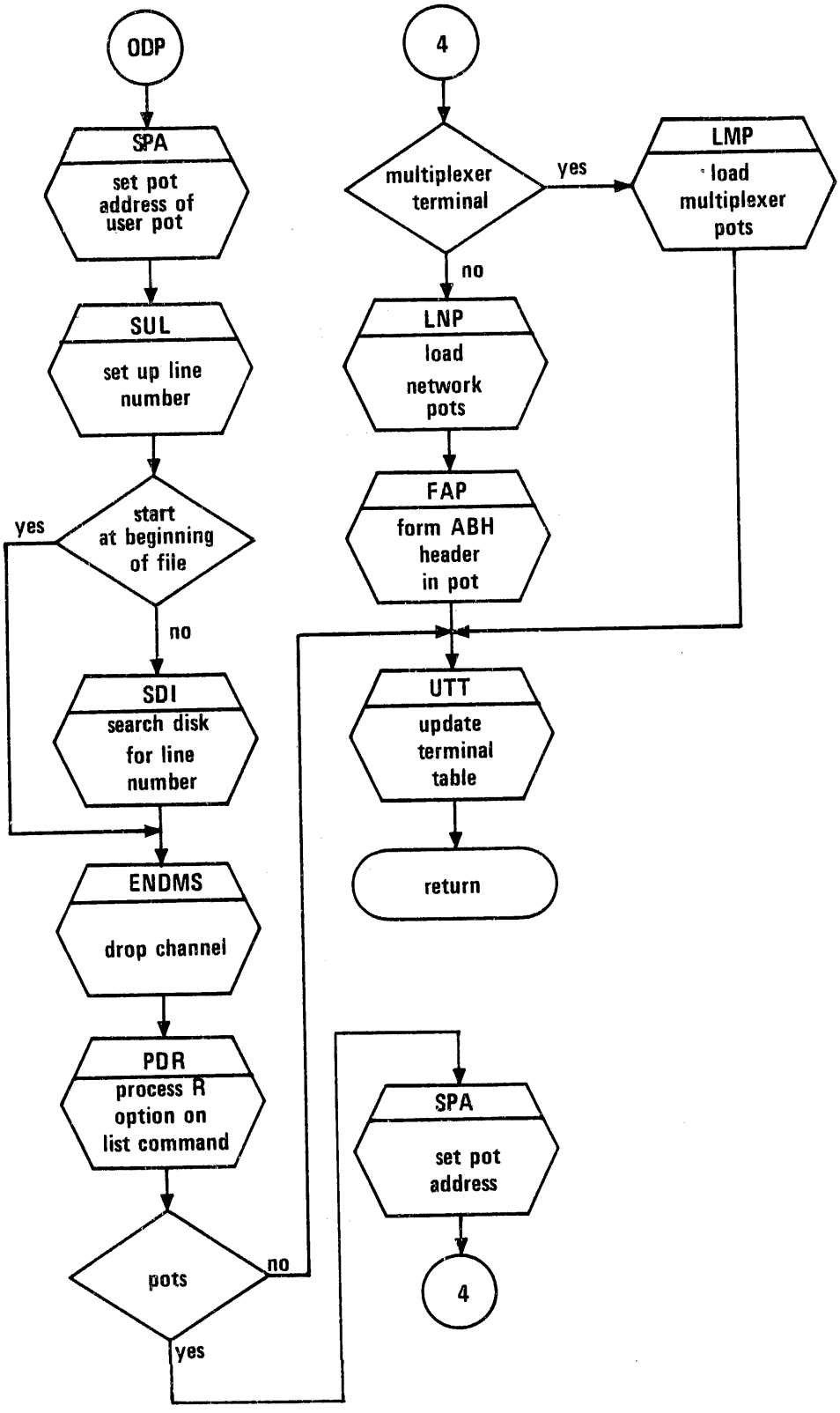


Figure 15-22. 1TO I/O Routine (Continued)

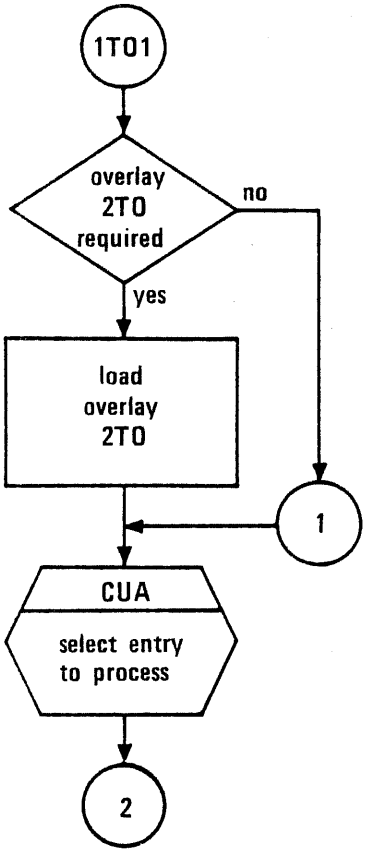


Figure 15-22. 1TO I/O Routine (Continued)

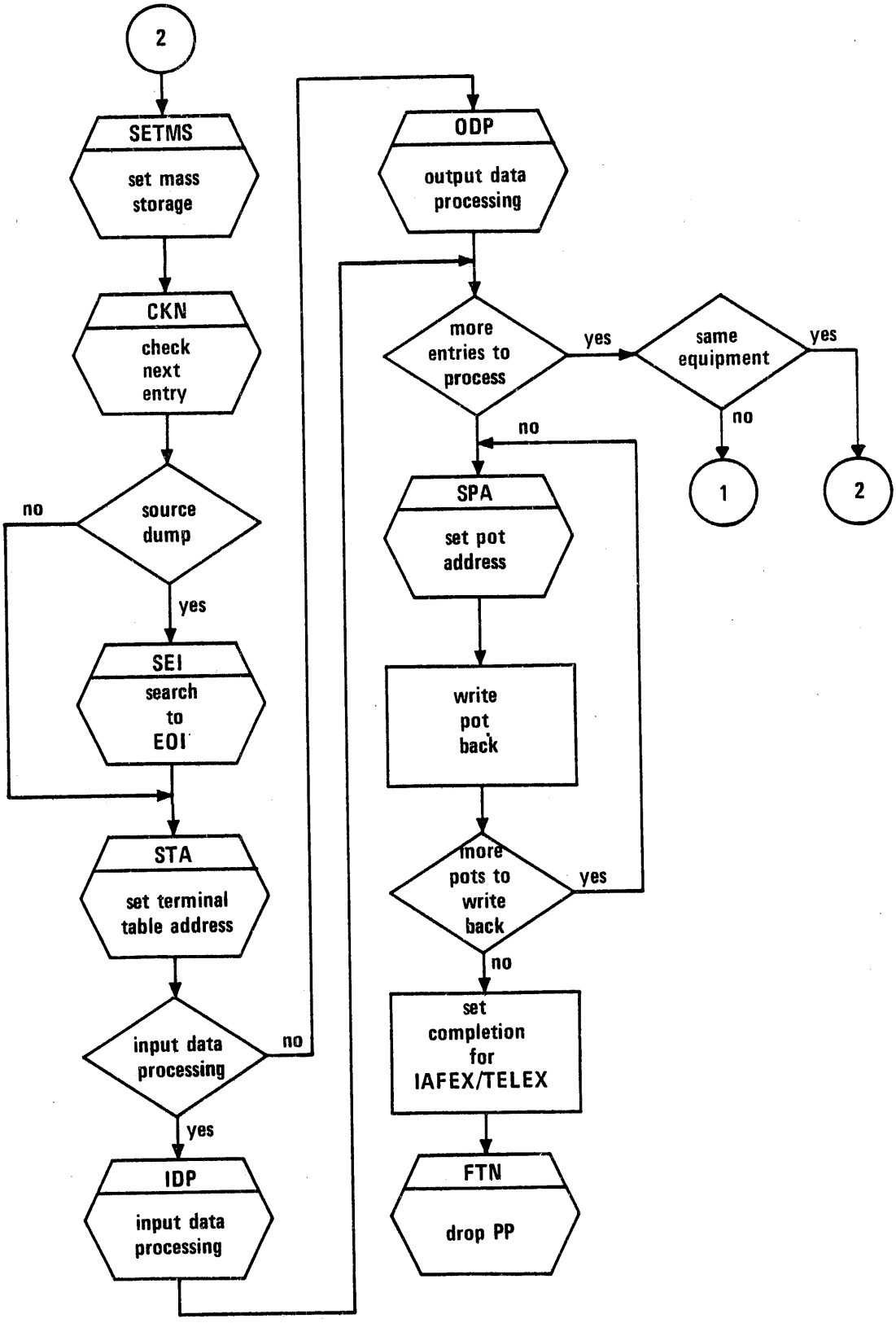


Figure 15-22. 1TO I/O Routine (Continued)

ADDITIONAL CONSIDERATIONS

The NETWORK or SIMFILE file specifies all the known terminals. Routine 1TD assumes all terminals are 110 baud (10 cps) time-sharing terminal type during login.

The $LDC issued by TELEX is a compiler call statement issued in response to terminal user typing RUN or some similar call in the BASIC subsystem.

The TT entry in VALIDUs is used for validation. If the entry is set, the user must be on that type of device to be validated. The TERMINAL table defines what type of terminal is calling.

SALVARE-TELEX RECOVERY FILE

The SALVARE file is a fast attach permanent file built by ISF during an initial deadstart or recovered during a recovery deadstart. The size of the file is determined by ISF and its length is not altered by TELEX. ISF assures that the file length does not exceed one logical track on the residence device.

During initialization, TELEX reads the SALVARE file in subroutine URT to update the recovery times. This is done to assure that a user is given the full time to recover, no matter how long a system recovery has taken.

During operation in TELEX1, the main loop calls CSF. CSF issues a 1TA queue call to check the SALVARE file in 1TA routine CUS function 20. CUS clears all entries in the SALVARE file and logs off users over 10 minutes old. This call is made about evey 3 minutes.

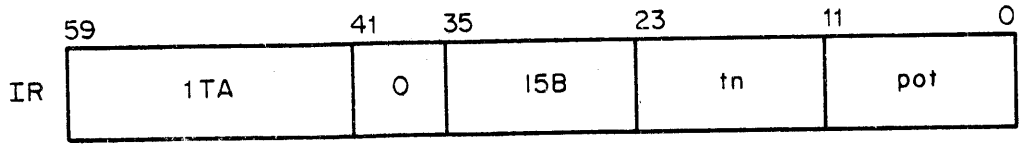
Routine 1TA is a combination of functions to perform for TELEX. The important functions associated with the SALVARE are as follows.

Function	Description
CUS	Clean up file
TLP	Terminal log out processor
TRP	Terminal recovery processor; this overlay contains the SALVARE format documentation
RFP	Recovery file processor

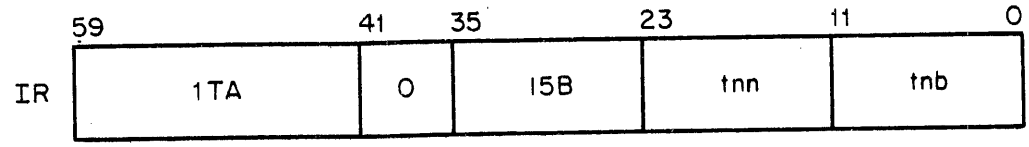
Since the SALVARE file is checked about every 3 minutes and entries more than 10 minutes old are eliminated, then:

- A user that wishes to be recovered after losing contact must attempt recovery within 10 minutes.
- Entries in the SALVARE file are updated upon system recovery so that a user is assured of the full time-out period after system recovery.

Recovery is accomplished in the recovery file processor routine, RFP. The call to overlay RFP is as follows.



Upon entry, IR+4 contains the parameter pot number. The pot contains the terminal table. IR+4 is set to the previous terminal number, which is recovered from parameter pot.

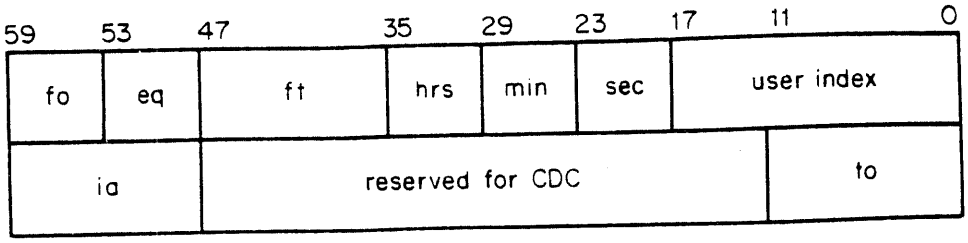


tnn	Terminal number now
tnb	Terminal number before

To recover a user, the entry on the SALVARE file is found and the information is returned to the terminal table. The entry in the SALVARE file is cleared and the current rollout file is released. A dayfile message is issued indicating the user recovered.

A completion logout is done for all entries that have been there longer than 10 minutes. At that time the files are released and subsequent dayfile messages issued. The beginning and EOI sectors for each file are validated to see if the user's files are all there. The status at the time the user was recovery processed is returned in VFST+4. The contents of VROT+4 is returned as 0003.

The SALVARE file is always at FNT ordinal 1. If 1TA finds the file active or destroyed (unrecognizable at recovery time) it hangs with the MXFN monitor function. The format for the file is as follows.



fo	Family equipment ordinal
eq	EST ordinal of rollout file
ft	First track of rollout file
hrs.min.sec	Last entry time in compressed format
ia	Installation reserved area
to	Terminal table ordinal

TAF OVERVIEW

The transaction subsystem allows the high speed handling of repetitive executions of a relatively small number of jobs called tasks. That is, the same set of tasks run many times by different people to perform the same function. The advantages of using a transaction system instead of the time-sharing executive functions are as follows.

- More important tasks may be given higher scheduling priority.
- Tasks may be kept CM resident within the executive to eliminate loading time and overhead.
- I/O requests may be managed by the executive to provide orderly data base access where many users are updating the same data base. I/O requests are overlapped.
- Data flow may be optimized by allowing only pre-specified file structures known to the executive; therefore, user jobs need not be concerned with complex file structures.

TAF may communicate to terminals through the time-sharing executive or through Network Access Methods (NAM). NAM uses the 2550 Host Communications Processor. NAM provides the following advantages to the transaction product.

- A Network Definition Language (NDL).
- Buffering and queuing of data for regulation of data flow.
- Support of a wide variety of terminals through normalization of data formats (code conversion), as well as ability to handle transparent data. Synchronous and asynchronous terminals may use NAM.
- Dynamic establishment, maintenance, and termination of data paths between terminals and TAF.

The transaction application user is assumed to be familiar with COMPASS, the TAF reference manuals and the TAF Data Manager reference manuals. For information related to the implementation of a transaction application, refer to the list of manuals in the preface.

The transaction executive is composed of two major functional pieces. One is directed to task execution management, the other to data base management. Three data managers are available for use with the subsystem, the TAF data manager, the TOTAL data manager, and the TAF CYBER Record Manager (CRM) data manager.

A transaction application is implemented by creating a data base, defining a terminal network and by writing tasks to perform the transaction process. Some of the characteristics of tasks are as follows.

- They must be (00) level overlays placed on a transaction task library by LIBTASK.
- The only RA+1 requests they may make are those processed by the transactions executive.
- They are loaded and executed at subcontrol points within the tranasction subsystem field length.

These tasks read and update information on the user's data base and generate output to the transaction terminals.

The subcontrol point feature allows the transaction executive to maintain complete control over each task. Some of the advantages associated with subcontrol points are:

- Isolation of one subcontrol point from other subcontrol points and the transaction executive, guaranteeing system security.
- Blocking of RA+1 direct requests. No PP requests or I/O actions are allowed directly from a subcontrol point. Any such requests are intercepted by the system monitor which returns control to the transaction executive. The transaction executive manages resources and provides synchronization.
- Freedom to move, load, and overlay areas within the subsystem field length. Since each subcontrol point has a relative origin of zero, absolute overlays all originating at a given address (for example, 111B) can be loaded in any order and at any place within the subsystem field length.

An installation parameter sets the number of subcontrol points that the transaction executive initializes. When the transaction executive is loaded, the operator may select a number of subcontrol points other than this default value. The number of subcontrol points must not be less than two or greater than 31. Once the transaction executive in initialized, no change in the number of subcontrol points is allowed.

Each subcontrol point requires eight words of table space within the transaction executive. No space, other than a table entry, is allocated for a subcontol point unless it is active. The optimum number of subcontrol points is selected by the site.

A data manager controls the structure of user data. In order to control this data, the data manager must be supplied information about a user, his application area, and installation. This information is provided by the user at data base definition time. A data base can be defined as this control information together with transaction data supplied by the user.

The transaction data base consists of related data files. Data files have specific names that provide a common point of reference between user programs and the data manager. Data files are structured into groups of information called records. All records in a file must have the same structure. Records may be subdivided into elements. One or more elements may serve as a key or identifier for a record.

At data base definition time, the user supplies a desription of all data elements and data files to be contained in the data base.

The information provided to the data manager consists of parameters that describe the physical allocation of the data, parameters that describe the element characteristics and security, and parameters that describe the file organization.

TAF INITIALIZATION

TAF uses a procedure file for initialization. For TAF using NAM for terminal communications the file is TAFNAM. For TAF using multiplexers for terminal communications the file is TAFTS. Both indirect access files are under the system's user index. The applications analyst may modify these files for special transaction subsystem initialization or termination. The general format of these files is as follows.

TAFNAM	TAFTS	Purpose
CALL,SETUP.	CALL,SETUP.	Optional. Installation may use procedures to set up transaction environment.
RFL,70000.	RFL,60000.	Required.
1,TAFNAM1.	1,TAFTS1.	Required for initialization.
TAFNAM2.	TAFTS2.	Required for termination.
EXIT.	EXIT.	Required.

TAFNAM	TAFTS	Purpose
DMD. DMD,0,37777.	DMD. DMD,0,377777.	Optional. May be used to get dum to investigate problems.
TAFNAM2.	TAFTS2.	Required for termination.

IF(SW4)GOTO,1.
IF(SW5)GOTO,2.
RETURN,OUTPUT.
2,EXIT.
	IF(SW4)GOTO,1.
	IF(SW5)GOTO,2.
	RETURN,OUTPUT.
	2,EXIT.
		Optional. Dump is printed if sense switch 5 is set.

CALL,CLEANUP.	CALL,CLEANUP.	Optional. Installation may use special procedures for cleanup.

The routine 1SI calls the proper file depending on the DSD command used to initiate the transaction subsystem, TAFNAM or TAFTS.

The transaction subsystem is initiated via a DSD console entry and is scheduled by the operating system to run until operator termination, at control point 2. To use TAF using terminal communication via the time-sharing executive type the DSD command TAFTS. To use NAM for terminal communication under TAF type in the DSD command TAFNAM.

Initialization consists of attaching files required for operation and dynamically establishing internal tables and management areas as a result of installation parameters and operator K display commands. Parameters dynamically set include the number of subcontrol points, number of communication blocks, maximum CM and ECS field length, and the number of data manager I/O buffers. Files attached include the task libraries and data bases.

Table 16-1 lists the table and buffer pointers used by the transaction executive.

TABLE 16-1. TABLE AND BUFFER POINTERS

Word	Name	Meaning
10	VNSCP	Number of subcontrol points
11	VNCMB	Number of communication blocks
12	VTST	Start of terminal status table
13	VNTST	Number of terminals
14	VMDM	Multiple for data manager buffers
15	VLSP	Address of last subcontrol point
16	VATL	Address of ATL
17	VFSCP	Start of subcontrol point allocatable storage
20	VCBRT	Start of communication block storage allocation bit maps
21	VCBSA	Start of communication blocks
22	VTLD	Start of task library directory
23	VEDT	Base address of element descriptor tables
24	VPOTT	Start of data manager buffer area
25	VMFL	Maximum field length for transaction subsystem
26-31	VSDB	Specified data base(s)
32	VTFL	Task library file name (must follow VSDB)
33	VREC	Recovery flag
34	VCRAT	Start of copied record address table

TABLE 16-1. TABLE AND BUFFER POINTERS (CONTINUED)

Name	Meaning
VECS	ECS field length
VECSC	Current next available ECS address
VCRS	CRAS terminal name
VROLT	Rollout control table
VRLAT	Rollout file allocation map
VCPA	Address of first subcontrol point table
VTOT	Total data manager initialization flag
VDBA	Transaction subsystem data manager initialization flag
VAAM	TAF CRM data manager initialization flag and FWA of routine to process CRM requests
VAAQ	FWA of FETs for TAF CRM input and output queues
VAMB	FWA of AAM record buffer
VINT	Initialization complete flag
VNACP	Address of pointer to free subcontrol point
VOEP	Overlay entry point name list
VOREL	Overlay relocation list
VDBAA	Transaction subsystem data manager status word
VSTAT1	Statistics area
VRTLW	Requested task list
VBSTR	Transaction subsystem data manager BST reservation map
VNCT	NAM communication table
VNON	NETON status (zero if NAM is running)
VSND	Application block number for NAM

The flowchart in figure 16-1 outlines the routine INIT which performs the initialization for the executive and for the data manager. The flowchart shows an overview of the transaction executive initialization process. The tables and buffers are set up in subroutine SETL. Following the flowchart, table 16-2 provides an overview of transaction subsystem memory, showing the order of the tables and buffers established during initialization.

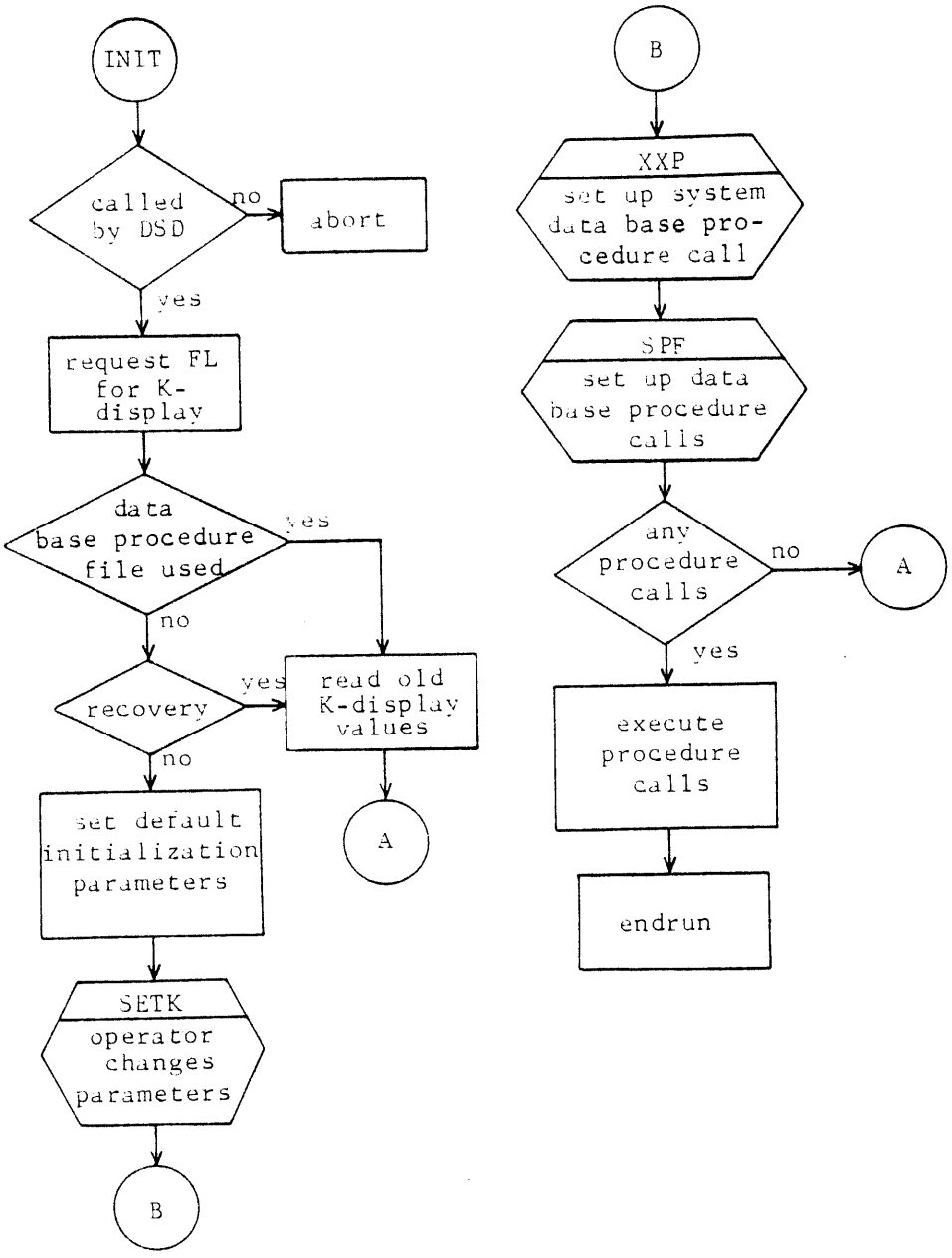


Figure 16-1. INIT - Initialize Transaction Executive

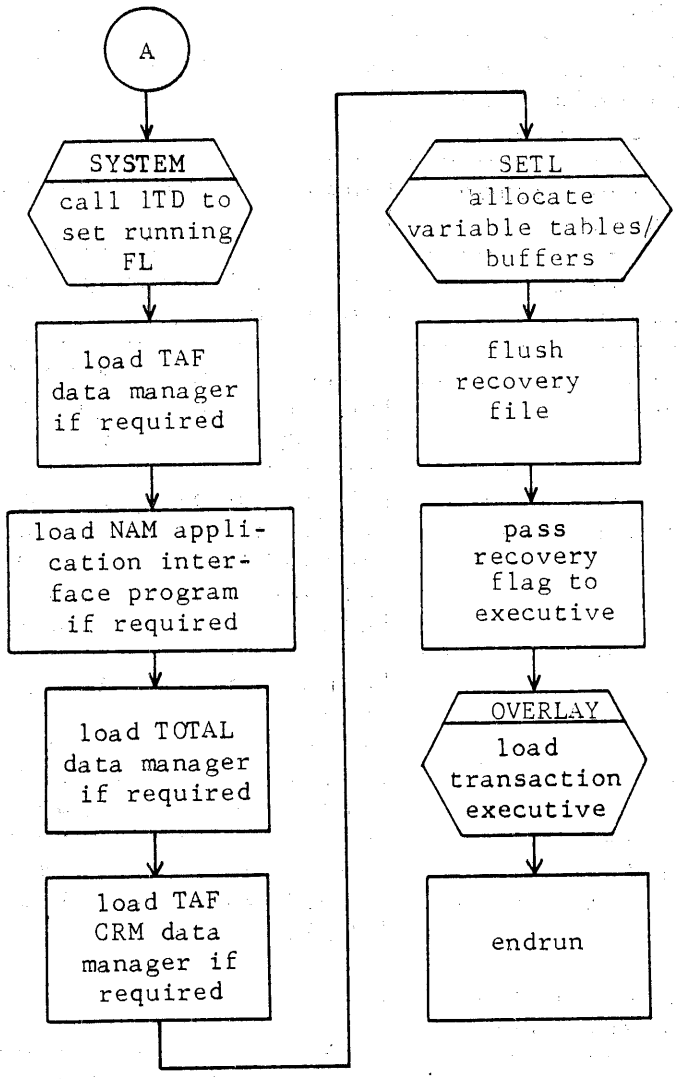


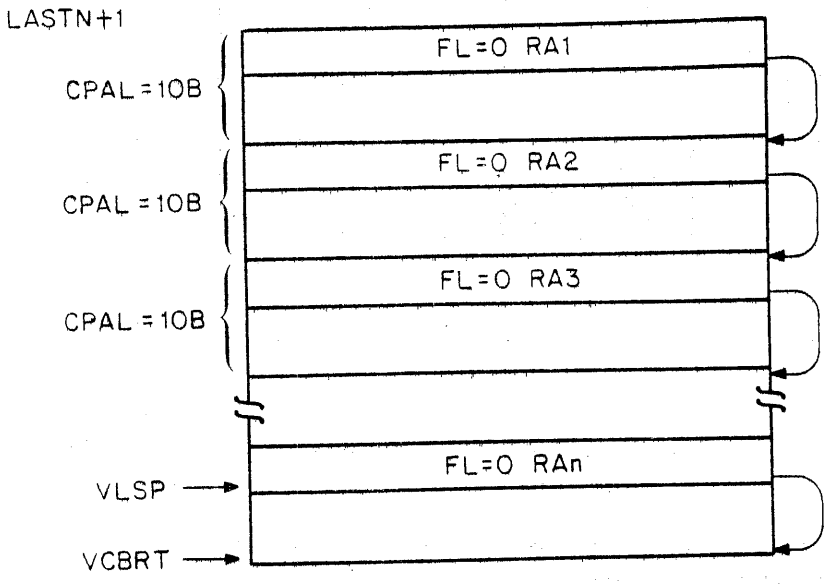
Figure 16-1. INIT - Initialize Transaction Executive (Continued)

TABLE 16-2. BUFFERS AND TABLES

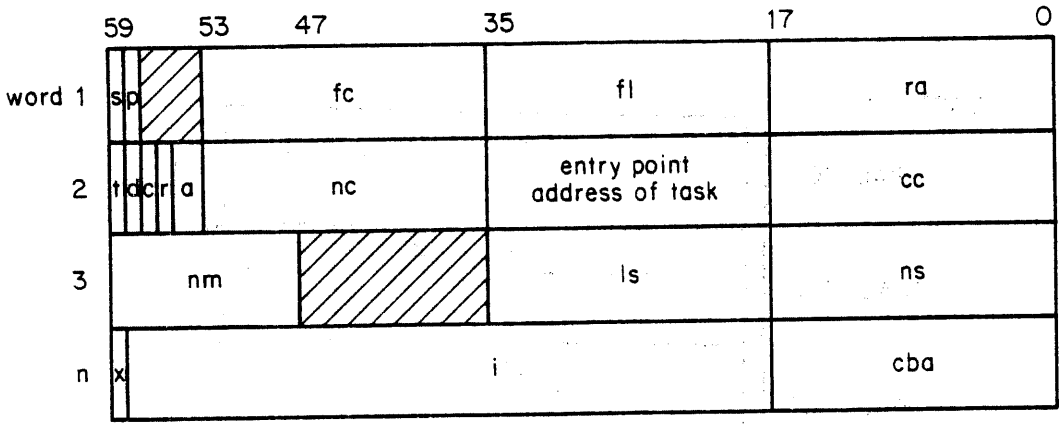
Address	Contents	Notes
LAST	Transaction executive	
	Data manager	
LASTN+1	Subcontrol point table	Number defined in VNSCP
VCBRT	Bit maps for communication blocks (CB)	Number defined in VNCMB
VCBSA	Communication blocks	Number defined in VNCMB
VATL	Active transaction list	One word per CB; each entry points to a CB
TST	Terminal status table	Built from NETWORK file or SIMFILE. Entries sorted on MUX channel, equip, port key.
VNCT	Network Communication table	Three words per connection. The first word points to the TST, the second contains the supervisory message, the third contains the application block header.
VEDT	EDT	Contains FETs for journal files, etc.
VCRAT	CRAT	Records from XX data base ERPF error recovery pool file
	Buffers for journal files	Pointed to by FETs in EDT above. 2002B words for MT, 402B words for disk.
VTLD	Task library directories	
VPOTT	Data manager buffers	Space allocated by transaction subsystem initialization but no FET poiners set.
VFSCP	Subcontrol points	
RA+FL		

SUBCONTROL POINT TABLE

The structure of the subcontrol point table as established by SETL is as follows.



The format of the subcontrol point table entry is as follows.



s Do not storage move this subcontrol point (bit 59)
p Can release this subcontrol point if core is needed (bit 58)
fc Available free core after subcontrol point
fl Subcontrol point FL
ra Subcontrol point RA

t If set, task is a system task and gets entire communication block (bit 59)
d If set, task code is reusable (bit 58)
c If set, task is CM resident (bit 57)
r Recall status bit (bit 56)
a If set, task is to be aborted (bits 55 through 54)
nc Number of communication blocks at subcontrol point
cc Address of status word for communication block now in execution

nm Task directory index
ls Last subcontrol point
ns Next subcontrol point

x Communication stack is present at subcontrol point (bit 59)
i Initial communication block load
cba Communication block FWA address

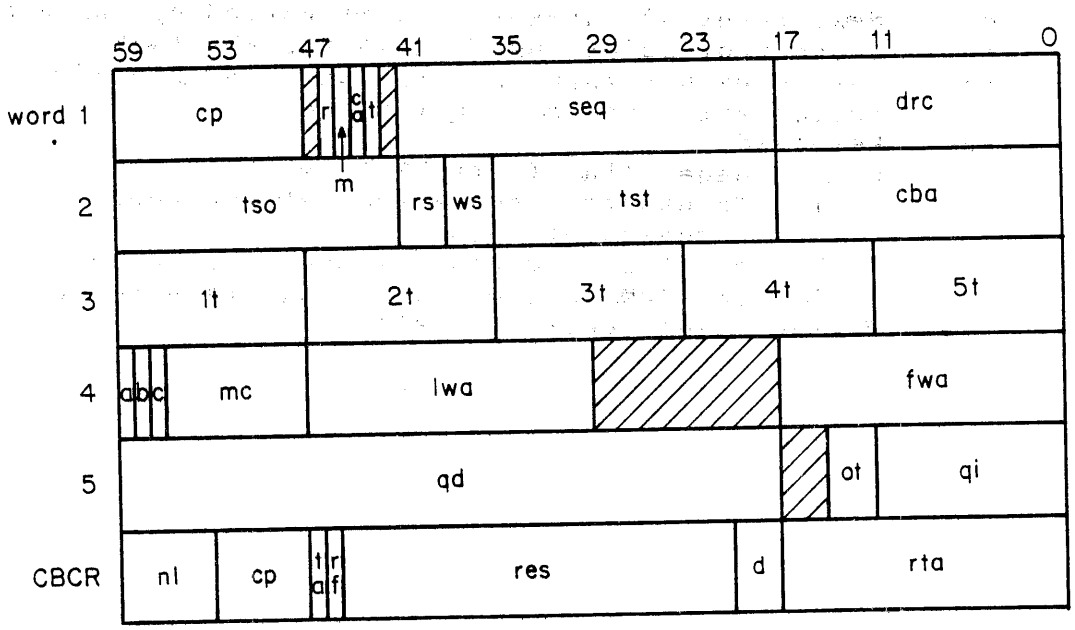
The length of the subcontrol point entry is 10B words. Thus, there are five words of type N. The length of an entry is defined by CPAL and must be a multiple of 10B.

COMMUNICATION BLOCKS

The communication block is used to pass input received by the transaction executive from a terminal to a task. It may also be used by tasks to pass along information to another task. It is sometimes used by the transaction executive to pass system information to a transaction system task.

Each new transaction message received by the transaction executive is sent, in a communication block, to ITASK. ITASK then selects the appropriate task(s) to call to process the input.

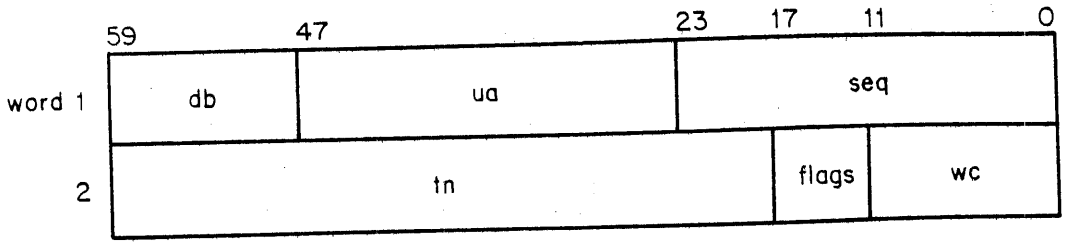
Communication blocks are set up by SETL by reserving CMBL times NCMB words. CMBL is the length of one entry and is equal to 75 words (a 6-word system header and 69 words of data). NCMB is the number of communication blocks (10 by default). Although not written during initialization, the format of the 6-word system header is formatted as follows (not accessible to nonsystem task).



cp	CPU priority
r	Recall on all outstanding data manager requests (bit 46)
m	At least one message was sent to terminal (bit 45)
ca	Transaction chain has aborted (bit 44)
t	Task initiated by a CALLRIN (bit 43)
seq	Primary sequence communication block address
drc	Data manager requests currently outstanding

tso	Terminal ordinal
rs	Terminal data base read security level (bits 41 through 39)
ws	Terminal data base write security level (bits 38 through 36)
tst	Address in TST for terminal
cba	Communication block address
1t	Next task schedule
2t	2nd task in chain to schedule
3t	3rd task in chain to schedule
4t	4th task in chain to schedule
5t	5th task in chain to schedule
a	Valid DSDUMP request (a=1) (bit 59)
b	Dump exchange package (b=1) (bit 58)
c	Dump data base buffers (c=1) (bit 57)
mc	Count of multiple communication blocks used for input (bits 56 through 48)
lwa	First word address of task dump
fwa	First word address of task dump
qd	Queue designator (see K.DSDUMP)
ot	Origin type value of queue destination (bits 14 through 12)
qi	Queue destination indicator
nl	Next level of current task if called by CALLRTN
cp	Subcontrol point number last CALLRTN task
ta	Called with return task has aborted (bit 47)
rf	Called with return flag (bit 46)
res	Reserved
d	Data manager flag (bits 21 through 18): 1 Transaction subsystem data manager requests allowed. 2 Total data manager requests allowed. 4 TAF CRM data manager requests allowed.
rta	Rollout table entry address

The communication block user header is formatted as follows (accessible to all user tasks).



db Data base that the terminal is validated to use
ua User area
seq Transaction sequence number

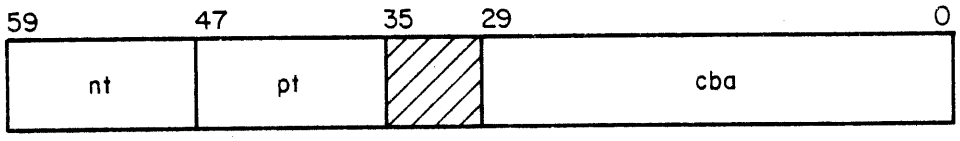
tn Terminal name
flags Each bit defined as follows:

Bit	Description
17	System origin transaction if set
16	Parity error occurred on terminal input if set
15	Transaction input came from batch rather than terminal if set
14-13	Unused
12	Multiple communication block if set

wc Word count of terminal input data for this communication block

ACTIVE TRANSACTION LIST

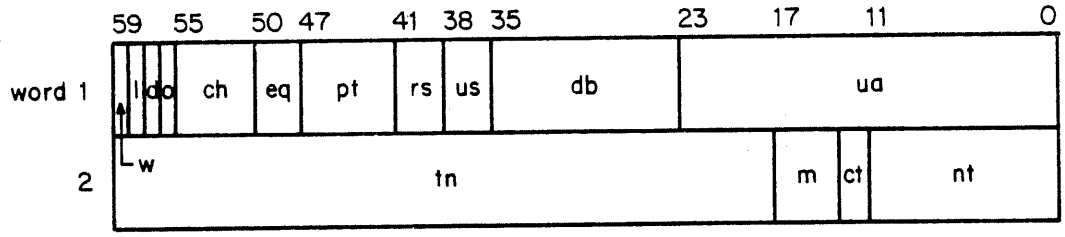
The active transaction list (ATL) as established by SETL contains a one-word entry for each communication block. Each ATL entry contains a pointer to a communication block. The format of the ATL entry is as follows.



nt	Next task in queue chain (biased by +1)
pt	Previous task in queue chain (biased by +1)
cba	Address of communication block

TERMINAL STATUS TABLE

The terminal status table (TST) contains a two-word entry for each terminal described in the network file or simfile stimulation. The list of entries is sorted according to multiplexor channel, equipment, and port key. For a description of the network file, refer to part IV, section 3 of the NOS Installation Handbook. The format of the TST entry is as follows.



w	Interactive task waiting for input
l	Terminal user active
d	Terminal down
o	Terminal on/off
ch	Multiplexor channel *
eq	Multiplexor equipment *
pt	Multiplexor port *
rs	Data base read security level
us	Data base update security level
db	Data bsse terminal is validated to use
ua	User area
tn	Terminal name
m	Multiple block input being sent
ct	Character type for terminal
nt	Number of transactions received from terminal

---

* For TAFNAM this area is used as an ordinal into the network communication table (NCT).

After SETL completes the initialization of the tables mentioned previously, routine IDM initializes the transaction subsystem data manager using the remaining field length. IDM attaches the data base identification file (DBID). The contents of the file are read into memory (error if not enough room) and written to the recovery file. The entries in the file contain data base names for the element descriptor table (EDT) files which must be attached. However, before proceeding, subroutine SDT is executed to allow the operator (via the K display) to specify up to three data base names. The procedure is outlined in the following steps for data base initialization:

1. Attach xxJ file for this data base (xx is the data base name specified in the DBID file). This file provides the user number and password which are stored in the EDT header.

2. Call subroutine INT in common deck COMBINT. This routine attaches the journal files described in the xxJ file attached in step 1. Trace and pool files (xxTFIL and xxERPF) are assigned FETs. The EDT header is initialized followed by the EDT entries for this data base as specified in file xx (xx=two-character data base name). The EDT header is described later.

3. A call to subroutine ATT attaches the pool and trace files (xxERPF and xxTFIL).

4. Call xxJ again. xxJ establishes FETs for the journal files (maximum of 3), and calls ATT to attach them.

The preceding process continues for all known data bases. Next, the copied record address table (CRAT) is initialized by subroutine ICRT. The subroutine reads the error recovery pool files (xxERPF) for the various data bases. Any records found in these files are placed into the CRAT. The CRAT is defined to be CRATL words long (currently CRATL equals 100B). If more records exist than will fit into the 100B word table, transaction initialization aborts.

After initializing the CRAT, IDM calls subroutine ABJ to allocate circular buffers for journal files. Tape and disk buffer sizes are defined by symbols TAPL and DSKL, respectively.

Currently, TAPL equals 2002B, and DSKL equals 402B.

Next, the last subroutine, LTL, is called. This subroutine loads the system task library directory, TASKLIB, and xxTASKL (xx is data base name), the directory for each data base. These directories have been created by subsystem utility, LIBTASK, and occupy the last file of the task library.

Finally, IDM determines the amount of buffer space required based on the number of subcontrol points, and sets the starting address for the subcontrol points. The recovery file is written and the main program is loaded. The loader performs the loading of the transaction executive and begins execution at the preset routine PRE.

TOTAL DATA MANAGER INITIALIZATION

The steps for TOTAL initialization are as follows:

1. Attach TOTAL binaries
2. Get data bases from TDBID file
3. Attach TOTAL data base files
4. Pass names of TOTAL data base files to TOTAL via loader request.
5. Use CYBER Loader to load TOTAL
6. Call INTOT. to initialize TOTAL. INTOT. is a TOTAL routine.
7. Save entry points of TOTAL in VTOT. Routine PRESET will set up calls to TOTAL.
8. Process xxJ files for TOTAL. The xxJ files give the name and residency of TOTAL data base files.

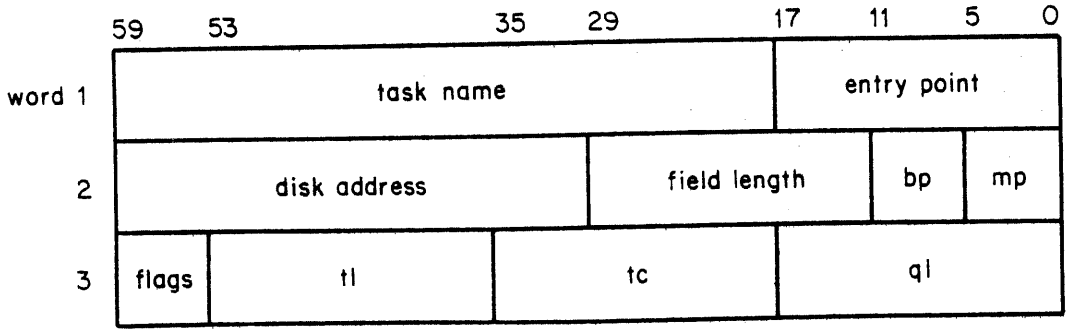
TAF CRM DATA MANAGER INITIALIZATION

The steps for initializing the TAF CRM data manager are as follows:

1. Use CYBER Loader to load CRM and Common Memory Manager (CMM).
2. Get data bases from CDBID file.
3. Process xxJ (xx equals data base) to attach CRM files and allocate space for CRM file environment tables (FETs/FITs) and record lock tables. A buffer is also allocated for the largest record.
4. Link calls in TAF executive to entry points in TAF CRM.

TASK LIBRARY DIRECTORY

The task library directory header is formatted as follows.



bp Base priority
mp Maximum priority (future use)

flags

Bit	Description
59	System task
58	Nondestructive code
57	CM resident
56	Library copy in ECS
55	Task turned off by operator
54	Task deleted
53	Solicit input

tl Number of times task was loaded
tc Number of times task was called
ql Queue length limit

FILES USED BY THE TRANSACTION SUBSYSTEM

When the transaction subsystem is initialized, the following files must have been set up.

NETWORK File

This file contains the description of each terminal (that is, the data base it has access to) which may communicate with the transaction subsystem (refer to the NOS Installation Handbook).* This file is used to build the terminal table which may be displayed with the 0,TR display.

DBID/TDBID/CDBID Files

The files that tell which data managers and data bases to initialized.*

Procedure Files SYPR, xxPR

Initialization procedure files, where xx is the data base name of a data manager data base.*

xxJ File

The file which identifies the xx data base user number and index, the journal files, the residency of data base files, and the data base task library.

---

*Refer to the TAF reference manuals or the TOTAL Reference Manual

EDT/DPMOD Files

The data manager data base descriptor files; EDT is for the Data Manager data base and DBMOD is for Total data base.*

TASKLIB/xxTASKL Libraries

The transaction subsystem system task library, TASKLIB, and the data base task libraries, xxTASKL.*

Journal Files

JOUR0 is the transaction subsystem journal file. Each data base may have up to three additional data base journal files xxJOR1, xxJOR2 and xxJOR3.*

ERPF File

The error pool file used by the TAF Data Manager to record file error recovery data.*

Trace Files

The Data Manager data base, xxTFIL, journal, log or trace of activity.*

xxTLOG File

The Total data manager data base journal, log, or trace of activity.*

SPECIAL RESERVED FILES

The following files are reserved by TAF while executing:

File Name	Description
KTSROLL	Task rollout file used to roll out tasks as needed.
OUTPUT	Output file.
RECOVR	Contains a copy of the low core pointers of TAF.
SCRA, SCR4	Used by TAF for tape journal files.
SCR5	Scratch file.
TROB	Used by TAF to roll out part of its own field length.

---

*Refer to the TAF reference manuals or the TOTAL Reference Manual.

The use of these file names by an application at the TAF control point or attached in a procedure file called by TAF may result in an error.

TRANSACTION EXECUTIVE

The transaction executive is loaded at TFWA by the loader and begins execution at the preset routine, PRE. In general, PRE completes the initialization started by transaction initialization. The preset routine performs the following steps.

1. Call subroutine SETA to modify the 30-bit increment instructions to eliminate the need for reading up pointer words (V-words) when referencing tables.

2. Call PVV to set variable values, such as maximum field length (MFL), current field length (CURFL), and available central memory (AVAILCM).

3. Call LIT to load the initial task from system task library to subcontrol point one. Initial task remains at subcontrol point one.

4. Call LCT to read task library directories and load CM-resident tasks at subcontrol points. If more tasks than subcontrol points available, abort.

5. Call IJF to position each journal file to EOI and write a label containing the current date.

6. Call SIC to initialize intercontrol point transfers. Terminal input, batch input, and LIBTASK status messages are done via intercontrol point transfers.

7. If NAM is to be used for terminal communications do a NETON.

8. Jump to TMDC to begin main processing.

A memory map of the transaction subsystem is shown in figure 16-2 for TAFTS and figure 16-3 for TAFNAM. The purpose of the three SEG instructions is to allow COMPASS to write partial binaries during assembly. Thus, less memory is required to perform the assembly.

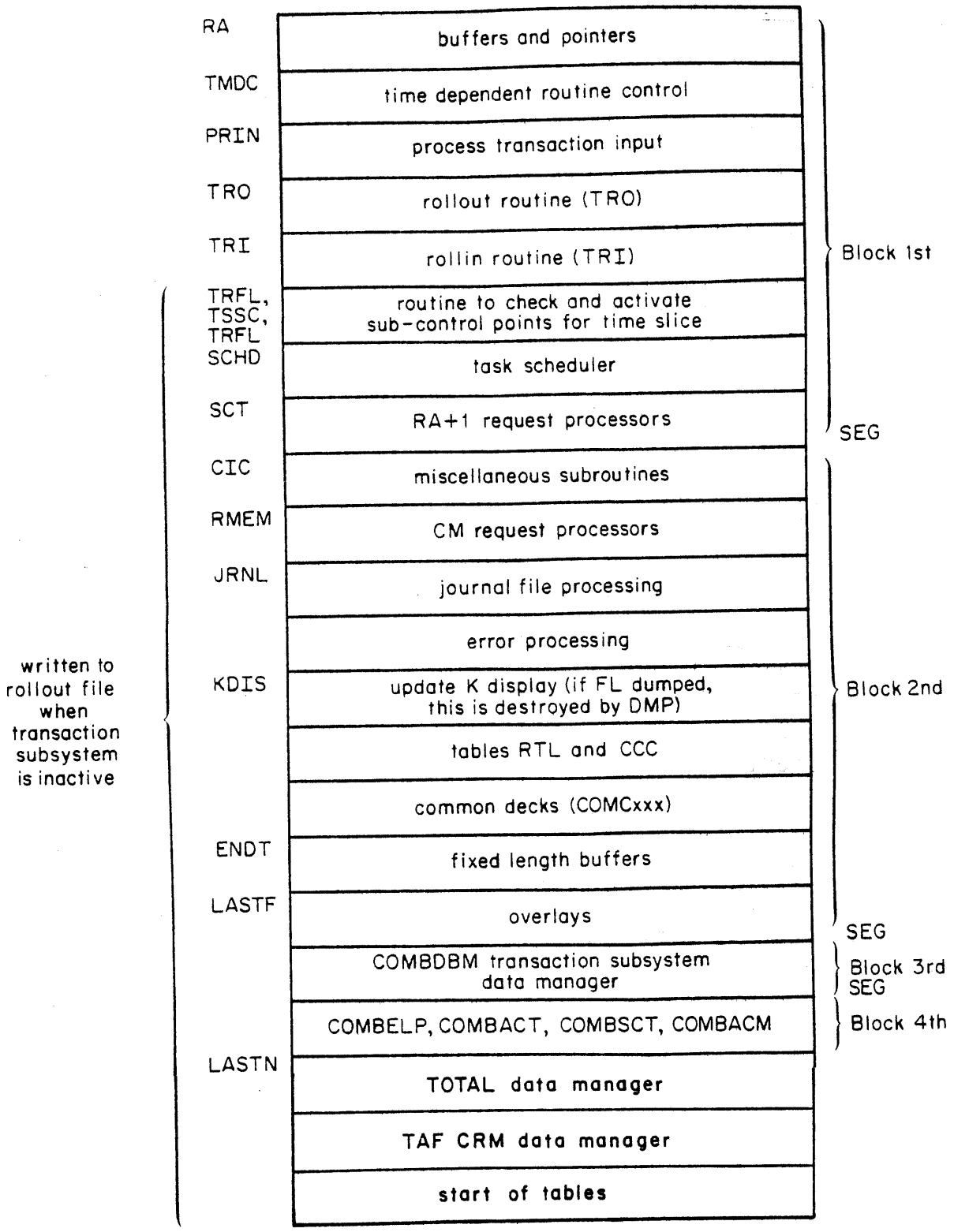


Figure 16-2. Transaction Subsystem Memory Map - TAFTS

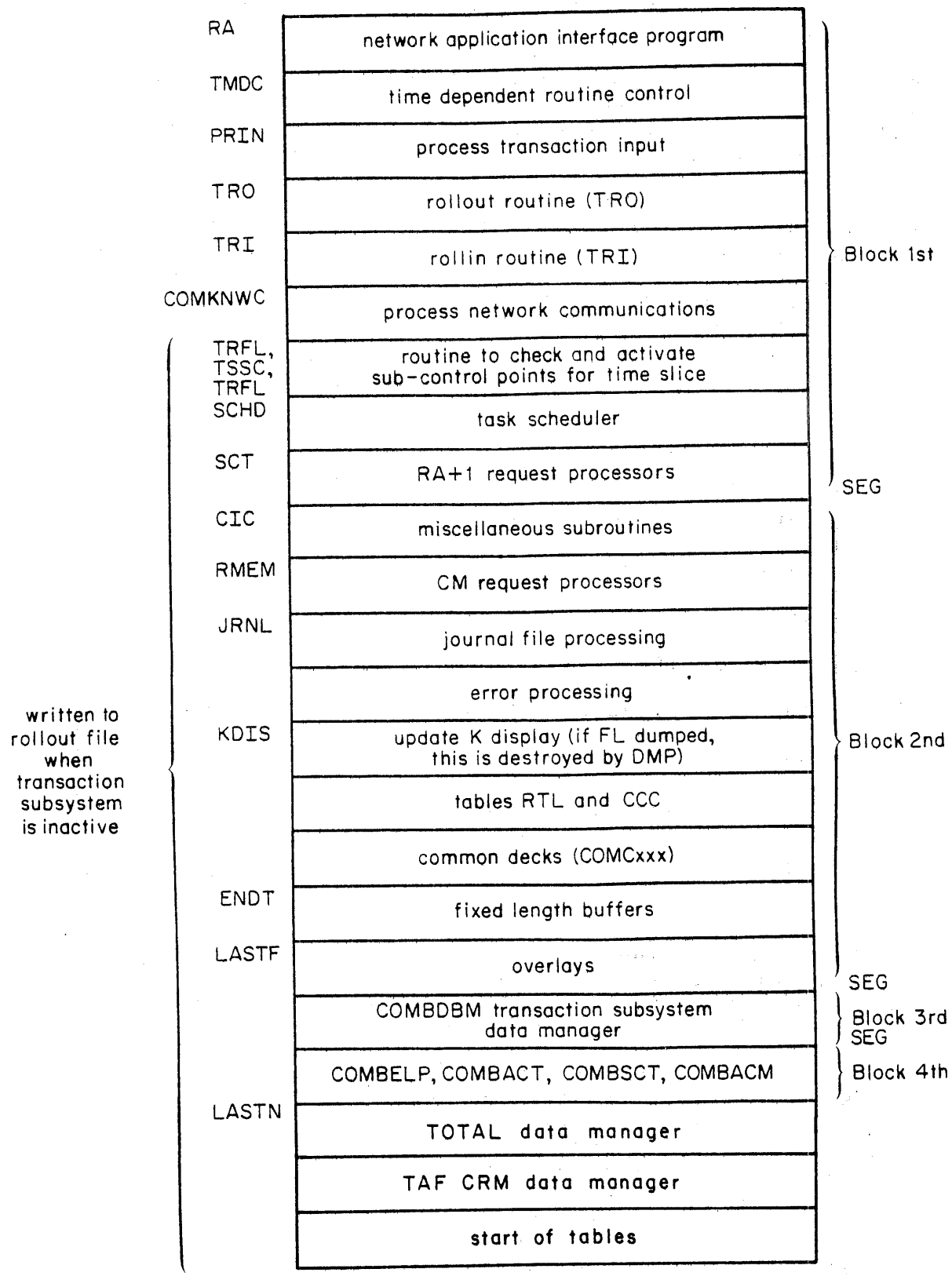


Figure 16-3. Transaction Subsystem Memory Map - TAFNAM

The symbol TRFL is defined at the end of subroutine TRI and is rounded up to the nearest 100B. The field length from this point to the end is written to a rollout file by subroutine TRO when no transaction activity is occurring.

Subroutine TRI will read the file, thus rolling the field length back into the control point. This occurs when transaction input is received by subroutine PRIN or when the rollout time slice (TROTL) has elapsed (to ensure time-originated tasks are activated).

Location ENDT marks the end of the run time code and the beginning of the fixed length buffers. Location LASTN marks the end of the buffers and the beginning of the tables and buffers set up by initialization. The fixed length buffers and their sizes are listed in table 16-3.

TABLE 16-3. BUFFERS AND LENGTH

Buffer	Length
JBUFO - Journal File	1201B
DIBF - Data Manager Input	20B
DOBF - Data Manager Output	30B
TDIBFL - TOTAL Data Manager Input	12B
TDOBFL - TOTAL Data Manager Output	30B
OBUF - Output Buffer	401B
SBUF - Scratch Buffer	100B
PBUF - Internal Trace Buffer	170B

Time dependent routine control consists of one routine named TMDC. TMDC calculates elapsed time for various subroutine calls. If the time limit for a particular routine has been exceeded, that routine is called. Subroutines called by TMDC include:

NGL	Get terminal input from NAM
PRIN	Process transaction input
SCHD	Schedule tasks
DCPT	Drop CPU for a task
KDIS	Update K display
CORU	Check core usage
TRN	Check transaction activity
JSTS	Write statistics to journal file
TSSC	Activate subcontrol points

The data manager (DM) is called by subroutine TSSC only. The data manager returns control to TSSC at a location defined by symbol TSSCO. The batch TAF data manager interface routine, BDMI, also adheres to this convention.

DM requests are queued and several issued at one time.

Figure 16-4 illustrates the flowchart for the transaction subsystem main loop. The TSSC routine is flowcharted in figure 16-5.

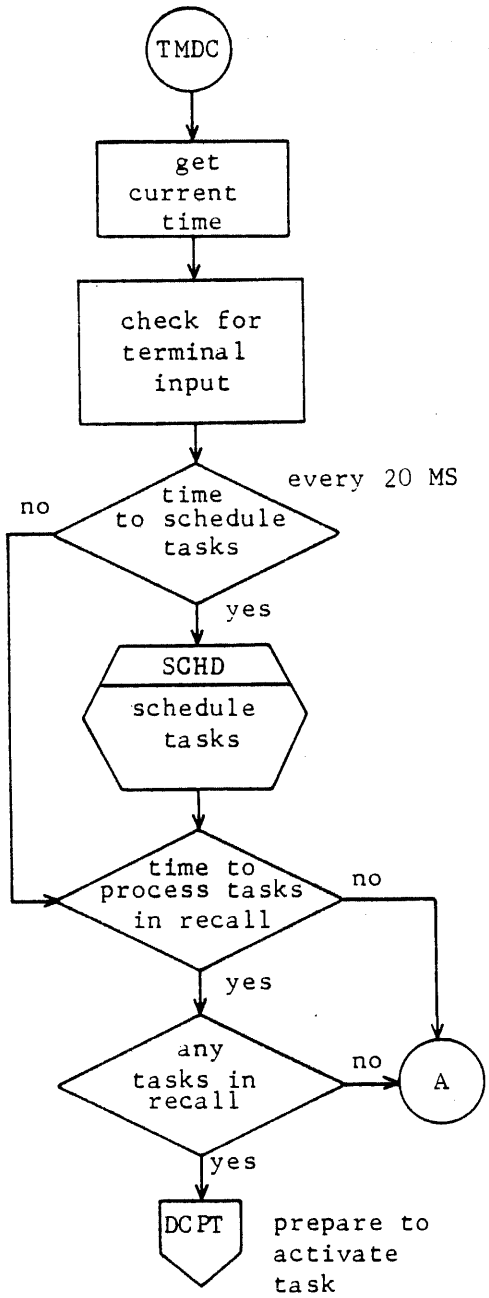
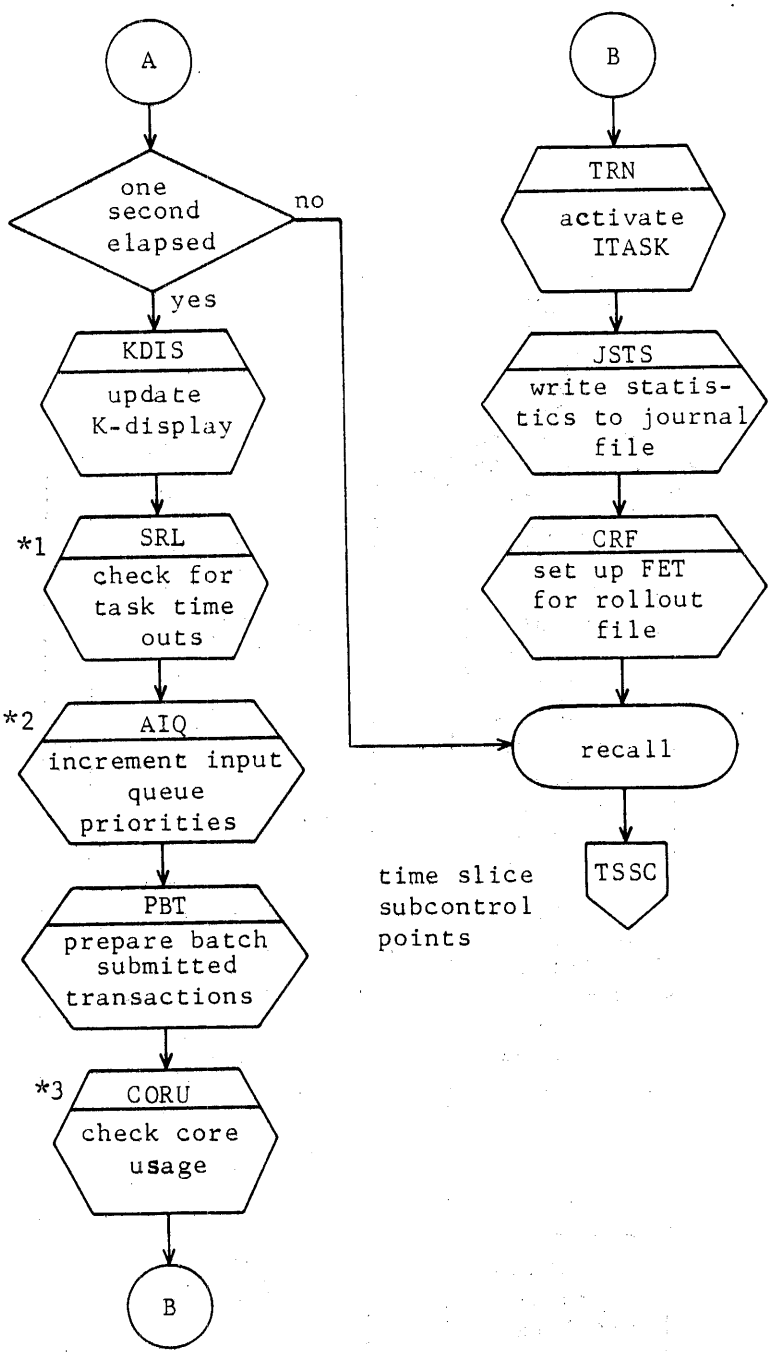


Figure 16-4. Transaction Main Loop



*1 Time to start tasks which have been rolled out for n seconds.
*2 Ensures new tasks are not started before starting a waiting task.
*3 Check for memory increase or decrease.

Figure 16-4. Transaction Main Loop (Continued)

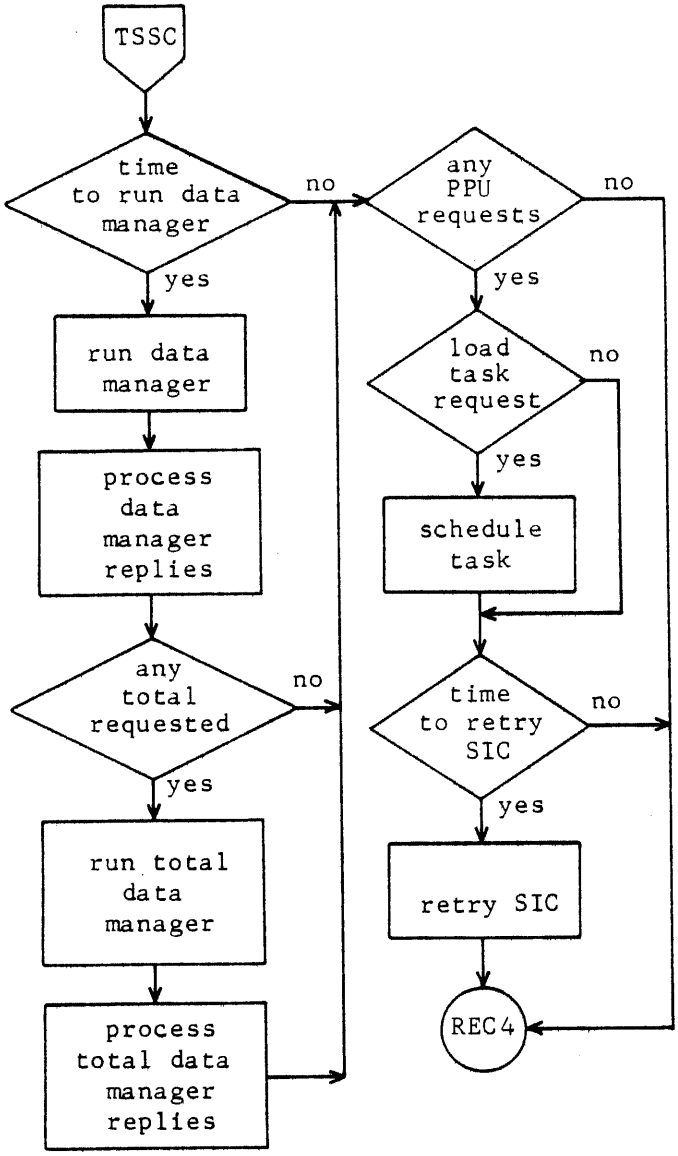


Figure 16-5. TSSC Loop - Task Slicing

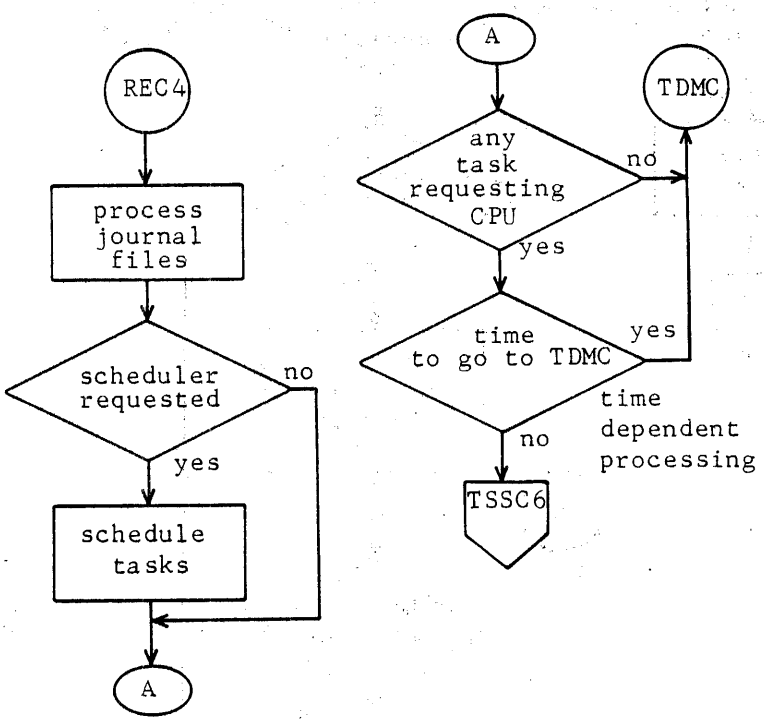


Figure 16-5. TSSC Loop - Task Slicing (Continued)

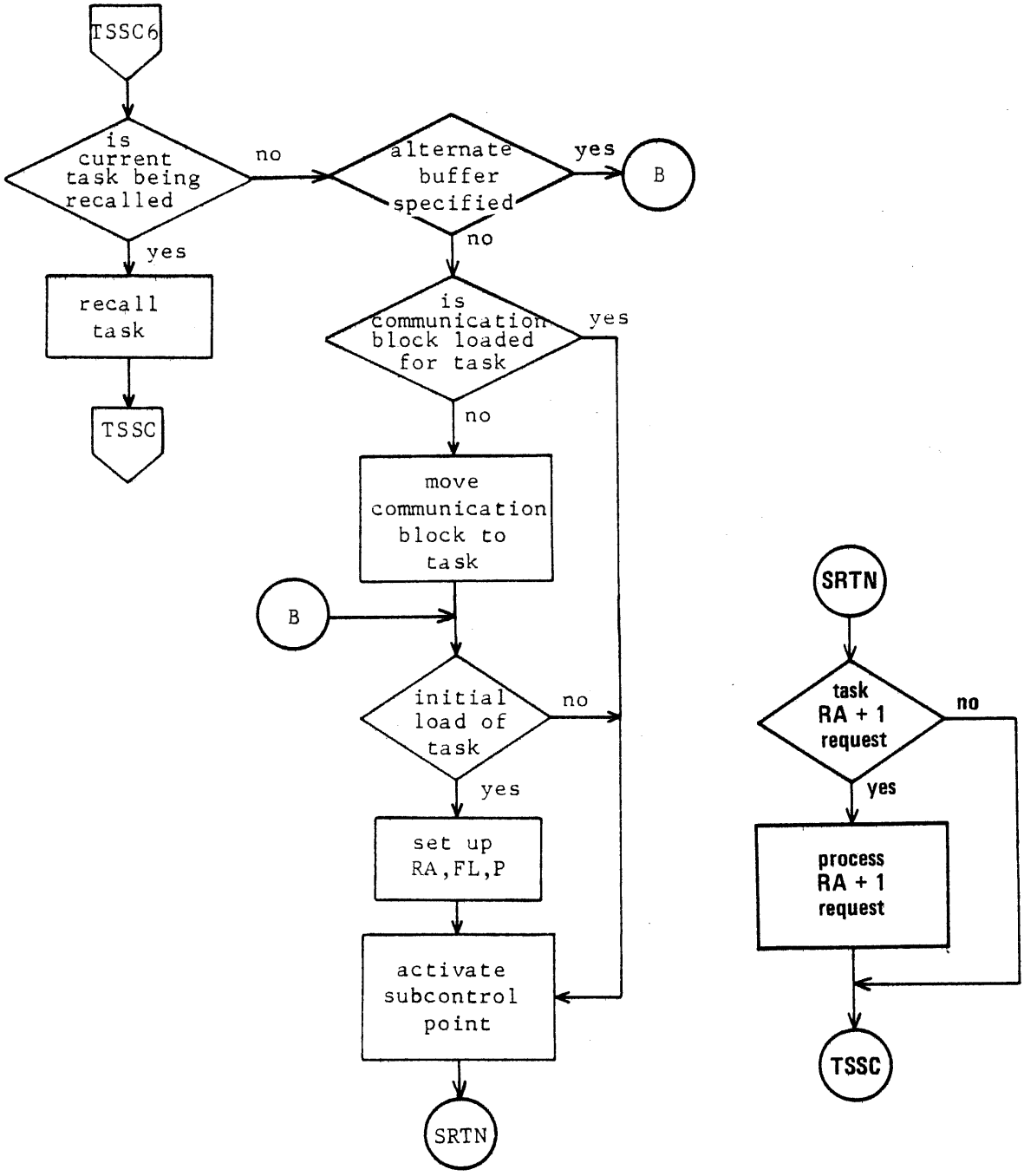
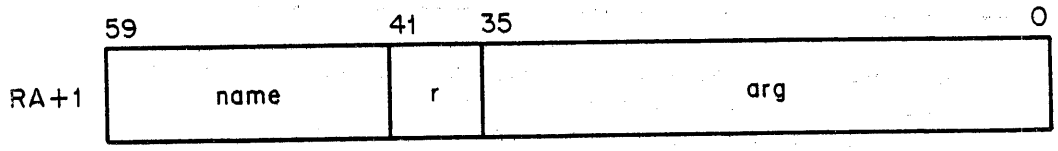


Figure 16-5. TSSC Loop - Task Slicing (Continued)

SUBCONTROL POINT PROGRAM REQUESTS

Subcontrol point reuests are passed to the transaction subsystem through RA+1. The format is as follows.



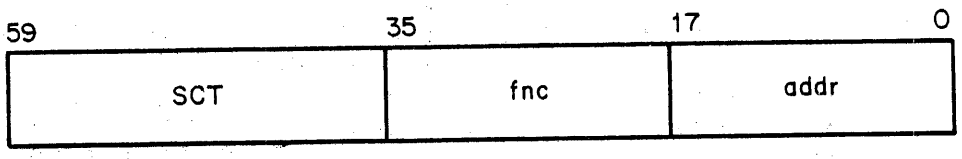
name Request name
r 20B if auto recall desired
arg Arguments

The recall parameter is of use only on data manager requests, as all other requests are answered immediately by the transaction executive.

If the request is not of the above format, the task is aborted.

The following paragraphs illustrate the individual request formats.

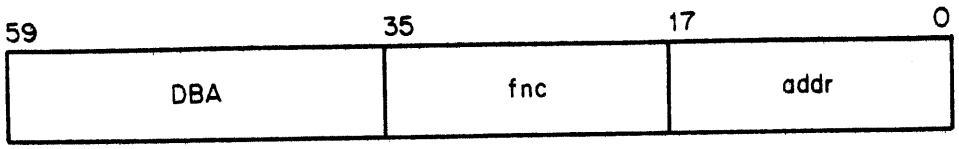
SCT - SCHEDULE TASK



fnc Schedule function code:

0 Task cease - end current task
1 NEWTRAN - start a new transaction
2 Call task with cease
3 Call task without cease - start an asynchronous task chain
4 Call task with return
5 Wait for terminal input
6 Timed rollout
7 CHKON - set Total interlock flag
8 CHKOFF - clear Total interlock flag
9 BWAITINP - return terminal input to a specified buffer, not restricted to the traditional communication block location (FWA of load)

addr Parameter address



fnc Data manager function code (0 through 177 is handled by the data manager with no special processing in TAF; 200 indicates recall on all outstanding data manager requests):

Code	Name	Description
1	GETT	Get elements
2	GETL	Get elements and lock
3	GETN	Get next record's elements (forward)
4	GETNL	Get next record's elements and lock (forward)
5	GETB	Get elements from record before this one (backward)
6	GETBL	Get elements from record before this one and lock (backward)
7	GETR	Get record
8	GETRL	Get record and lock
9	GETNR	Get next record
10	GETNRL	Get next record and lock
11	GETRB	Get record before this one (backward)
12	GETRBL	Get record before this one and lock (backward)
13	PUT	Put elements
14	PUTF	Put elements and force write
15	PUTI	Put elements in current record
16	PUTIF	Put elements in current record and force write
17	PUTR	Put record
18	PUTRF	Put record and force write
19	PUTRI	Put record in buffer replacing current
20	PUTRIF	Put record in buffer replacing current and force write
21	ADDR	Add record
22	UNLOK	Unlock record
23	UNLOKAL	Unlock all records
24	REPOS	Reposition
25	PURGER	Purge a record
26	RECHAIN	Rechain record
27	RELES	Replace buffer space
28	RELESAL	Replace all buffers held by this user
29	Cease	TAF issues this request to the data manager
30	BLKGET	Block get of records
31	BLKPUT	Block put of records
32	LOCKF	Lock file
33	UNLOCKF	Unlock file
34	OFFTRCE	Turn off trace
35	ONTRCE	Turn on trace

Code	Name	Description
36	DMSTAT	Return data manager buffer status
37	CLEAR	Clear
38	READEDT	Read element descriptors from the EDT

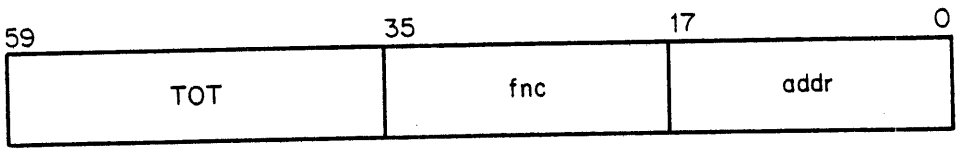
NOTE

The READEDT function is not assembled into the data manager by default. It is assembled when the COMBDBM symbol CREDT is equated to a nonzero value.

addr Start of data manager parameter area

A more detailed description of the data manager commands can be found in the TAF Data Manager reference manuals (refer to preface).

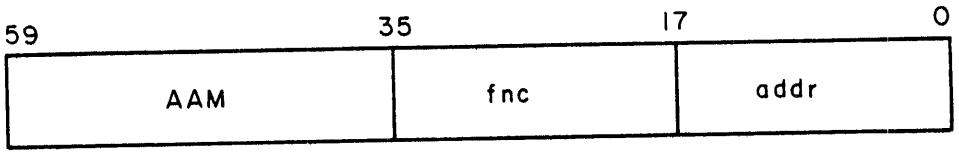
TOT - ENTER REQUEST INTO TOTAL DATA MANAGER QUEUE



fnc Total data manager function code

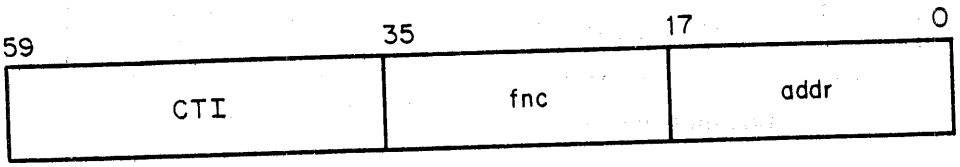
addr Parameter list address

AAM - ENTER REQUEST INTO TAF CRM AAM QUEUE



fnc Advanced Access Methods (AAM) function code

addr Parameter list address

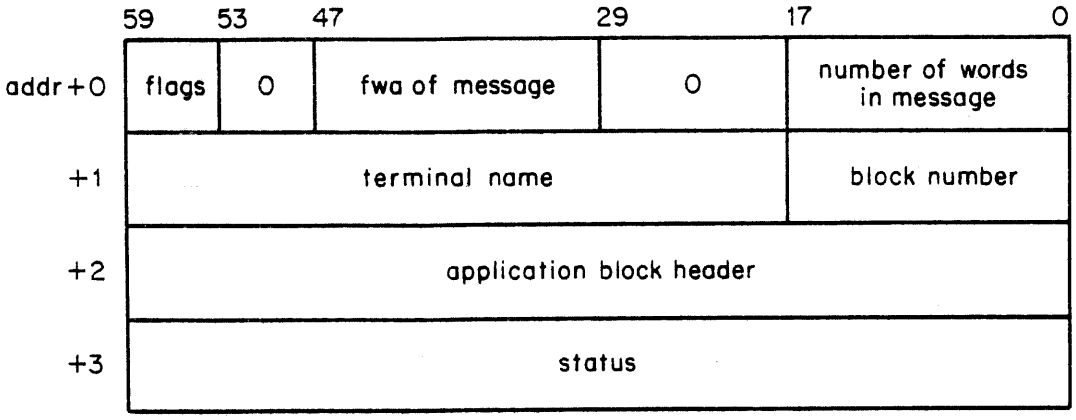


fnc Request code:

0	Send a message to a transaction terminal
1	Make a journal file entry
2	Check for a specific transaction still active
3	Process terminal argument operation
4	CMDUMP request
5	DSDUMP request
6	Return terminal status
7	K-display command
8	Use task data field for K display
9	Reserved
10	Submit batch job
11	Increase time limit
12	Increase I/O limit
13	Log out dial-in transaction terminal
14	Read multiple communication block input
15	Release extra multiple input communication blocks
16	Set terminal character type
17	Define terminal type
18	Get terminal application block header
19	Abort task; task request argument error
20	Return active teleprocessor code
21	Return communication block to specified location in the task's memory

addr Address of parameter list

SEND TERMINAL OUTPUT

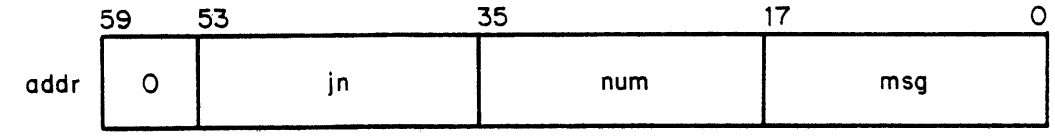


flags Each bit is defined as follows:

Bit	Description
59	Send to terminal specified in addr+1 if bit set; otherwise, send to originating terminal
58	Task cease request if set
57	Output flag; if set, more sends are to follow
56	Return application block number if set; applies to TAF using NAM
55	If set, task must wait for block to be delivered to terminal
54	If set, user supplied application block header at addr+2

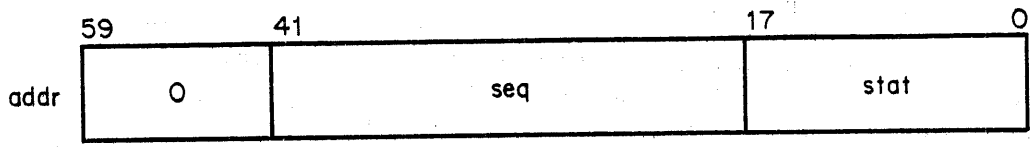
status NAM supervisory message returned if task send waited for block to be delivered to terminal

TASK JOURNAL REQUEST



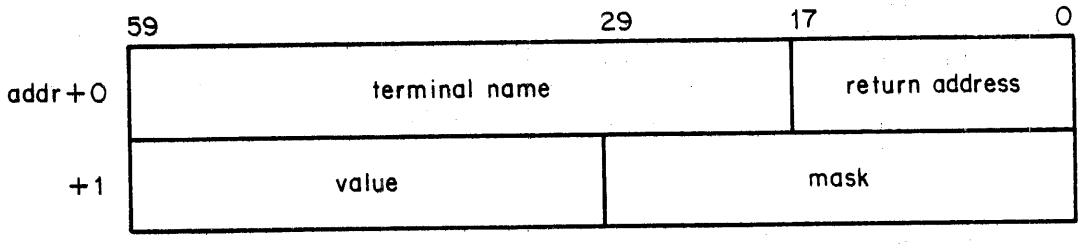
msg FWA of block to be journalized
num Number of words to write to journal file
jn Journal file number

CHECK FOR TASK CHAIN IN SYSTEM



seq Sequence number of transaction
stat Set stat to zero if transaction not in system

REQUEST CODE 3 - TERMINAL ARGUMENT OPERATION



Terminal name identifies terminal to be operated upon. If zero, originating terminal is assumed.

Return address is location to place result of operation (in addition to terminal table). Zero if no return desired.

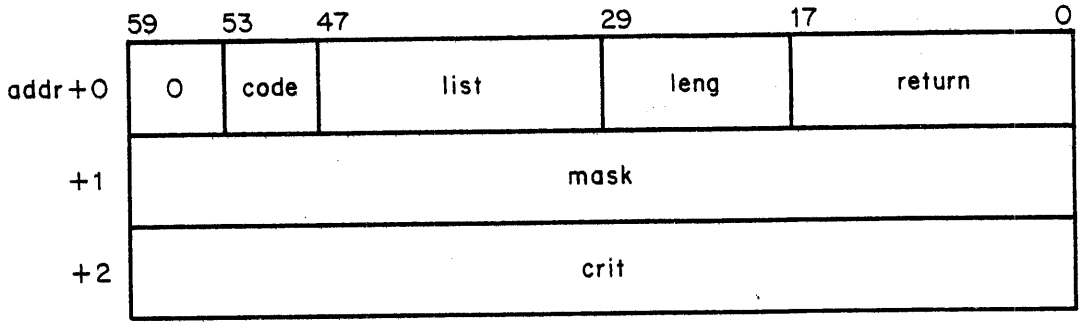
Value is a value to be used to alter terminal arguments, and mask is a 24 bit mask.

The user argument area (24 bits in each terminal table entry) is operated upon as follows:

Return = USER ARG=(USER ARG.AND.MASK).XOR.VALUE

Non-system tasks may alter terminal arguments only for those terminals that share the originating terminal data base.

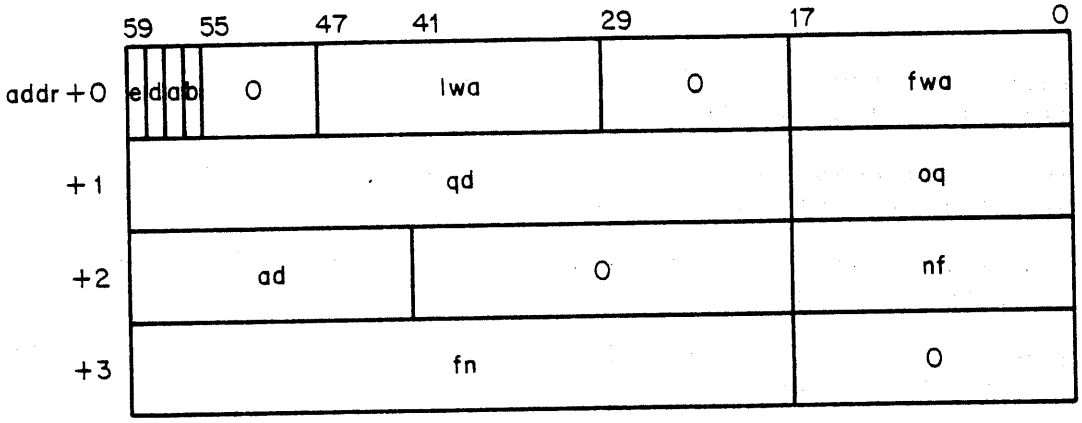
REQUEST CODE 6 - RETURN TERMINAL STATUS



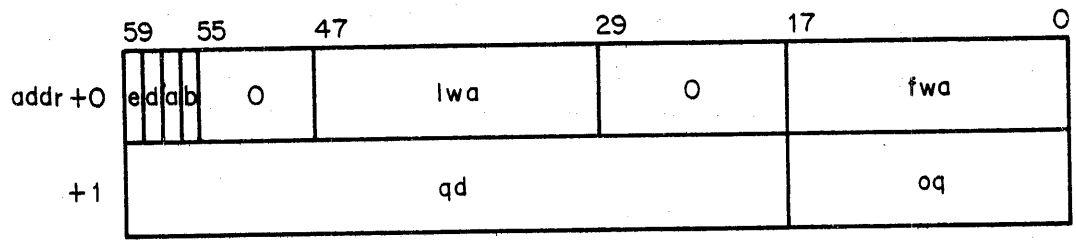
code	0 IF data base name field is to be searched 1 IF user argument field is to be searched 2 IF communication line field is to be searched 3 IF terminal name field is to be searched
crit	Criterion value for search
leng	Number of words that list can hold
list	FWA of list of returned terminal entries; if zero, no list is returned, but the number of found entries will be returned as specified below
mask	A value taken as a binary mask
return	Address in which to place the number of entries found

The field specified by code is examined in each terminal table entry by taking the logical product of the field and mask and then taking the logical difference of this product and grit. If this result is zero, the terminal entry is placed into list and the number of found entries is incremented.

CMDUMP

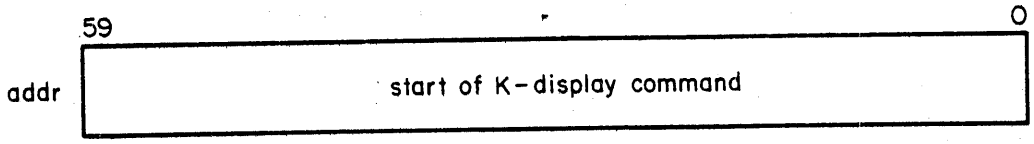


e	Dump exchange package
d	Dump data manager buffers
a	Use default exchange package parameter
b	Use default data manager parameter
lwa	Last word address of task to dump
fwa	First word address of task to dump
oq	Output queue
qd	Queue destination
ad	Address user called from
nf	Number of specified files
fn	Specified file name

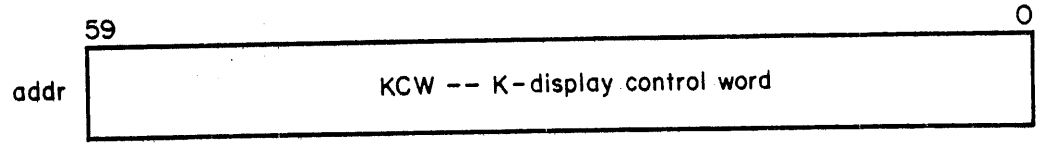


e	Dump exchange package
d	Dump data manager buffers
a	Use default exchange package parameter
b	Use default data manager parameter
lwa	Last word address of task to dump
fwa	First word address of task to dump
oq	Output queue
qd	Queue destination

KPOINT - TERMINAL K-DISPLAY COMMAND

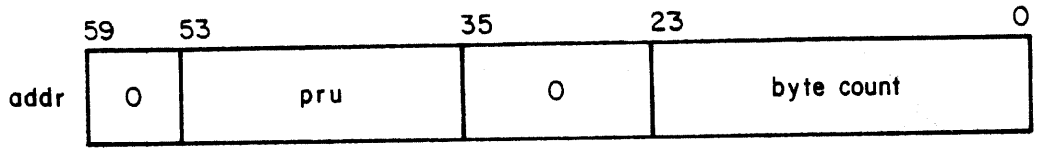


SET K-DISPLAY TO RUN FROM TASK



KCW K-display control word

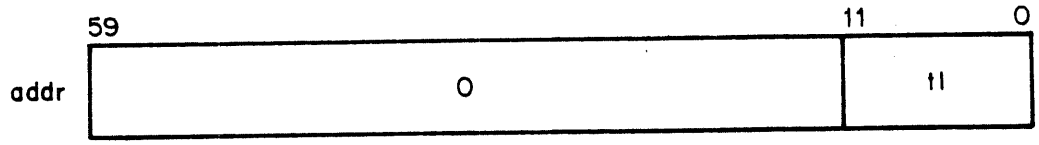
SUBMIT JOB TO BATCH



Contents of addr is the first control word for the output job data.

PRU is number of 60-bit words in each PRU on device. Byte count is number of bytes in this PRU. The block of information starting at addr is set up in control word format.

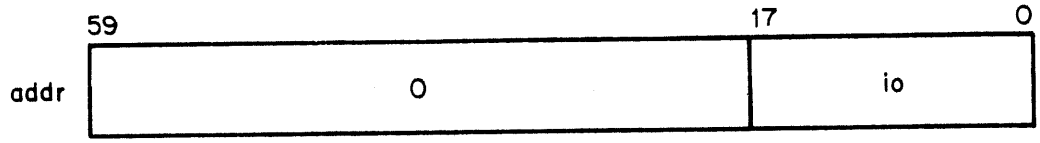
ITL - INCREASE TIME LIMIT



tl New time limit in XJ time units

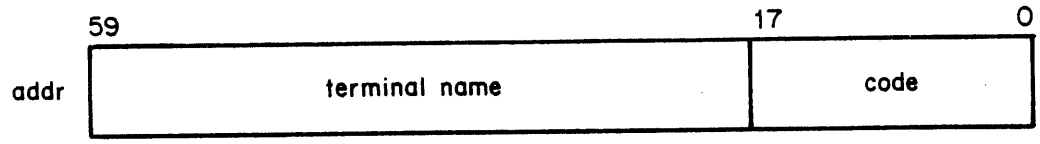
Each call to this function decrements the CPU priority of the task until zero is reached. Subsequent calls do not affect the CPU priority.

IIO - INCREASE I/O LIMIT

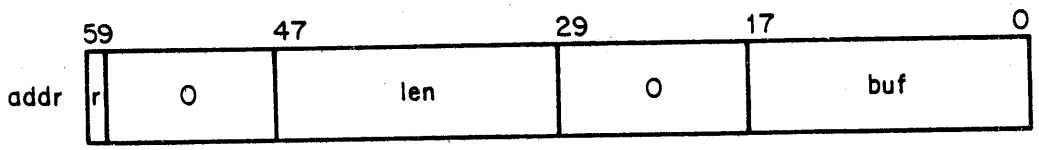


io New I/O limit in RA + 1 calls

SEND TERMINAL STATUS FUNCTION TO COMMUNICATION EXECUTIVE

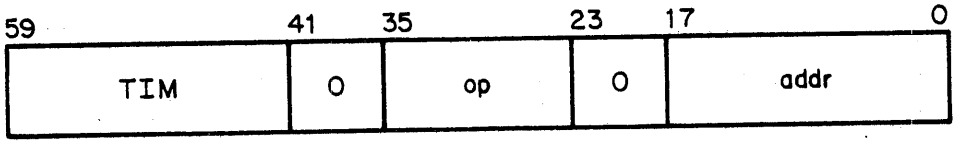


LOADCB - READ MULTIPLE COMMUNICATION BLOCK INPUT



r	Release extra communication block(s) after transfer
len	Length of buffer in task to receive data
buf	FWA of buffer in task to receive data

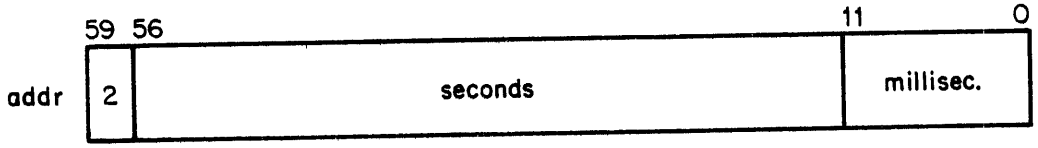
TIM - REQUEST SYSTEM TIME



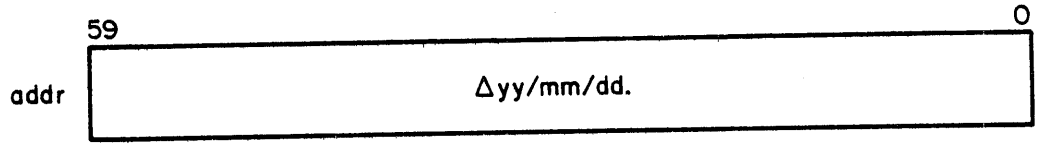
op	Time option
addr	Address for response

The contents of addr depend on which of the following time options is selected.

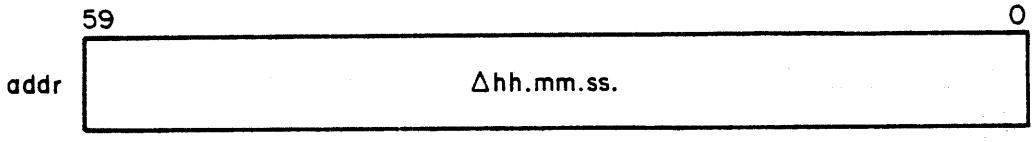
- Seconds (op=0)



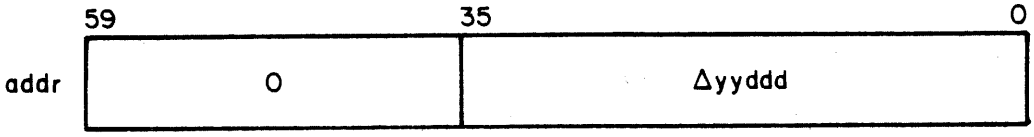
- Date (op = 1)



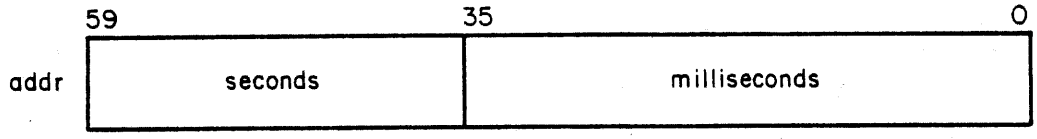
- Clock (op = 2)



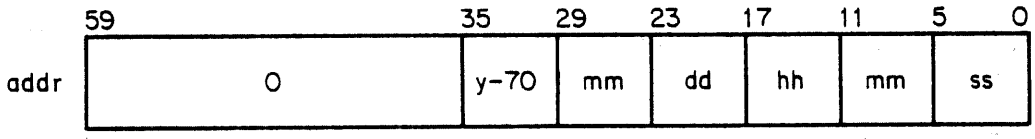
- Julian Date (op = 3)



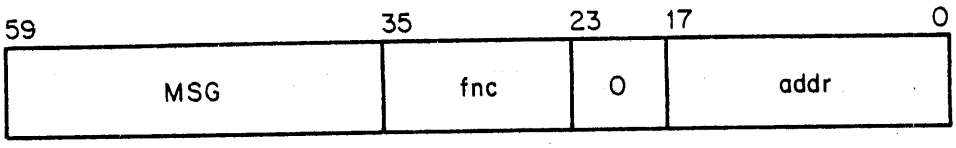
- Real Time (op = 5)



- Packed Date/time (op = 6)



MSG - PLACE MESSAGE ON LINE ONE



fnc Function code
addr Address of message to be displayed

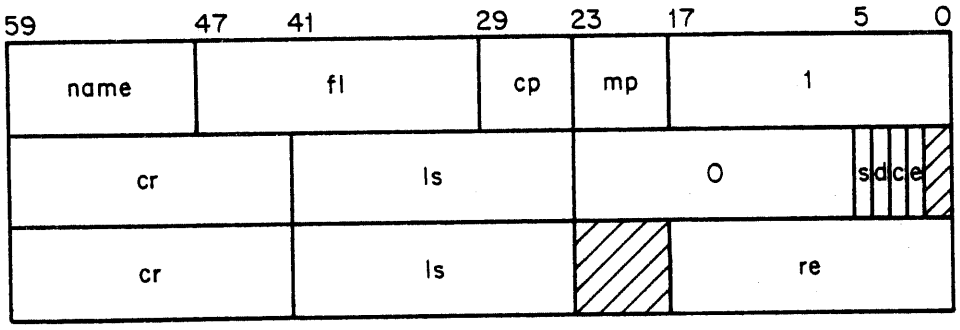
RA+1 REQUEST PROCESSING

Most of the RA+1 request processing routines described previously enter the TSSC subroutine upon completion of the request. TSSC is the subcontrol point supervisor which activates a subcontrol point via the XCHNGE macro. TSSC determines which subcontrol points are requesting the CPU and SRTN determines what servicing to schedule upon return of the CPU from a task. If there are any outstanding data manager requests, TSSC branches to the data manager(s) before activating a subcontrol point. TSSC also monitors PP completion statuses and reinitiates routines when their PP call is complete. For example, TSSC restarts task loading after a PP has performed the load, or TSSC restarts nonbuffered journal file processing as PP completion is sensed. Finally, at absolute time intervals, the system monitor drops the CPU from a subcontrol point so that control can be returned to the main loop for time-dependent processing. This time interval is defined by symbol TSKTL and is currently 120 milliseconds. Control returns at SRTN which checks error exit flags and RA+1 requests from the subcontrol point program. Under certain conditions, SRTN calls TXT, the internal task XJP trace processing subroutine (for more information concerning the trace, refer to Internal Task XJP Trace in this section). If an RA+1 request is present, one of the processors described previously is executed.

TASK SCHEDULING

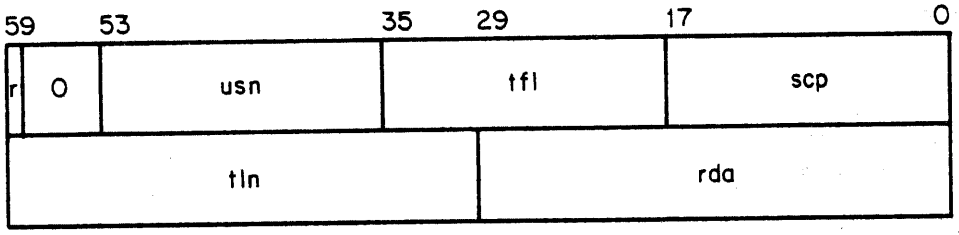
TMDC calls the task scheduler (SCHD) every SCHTL milliseconds (currently SCHTL=60). SCHD searches the requested task list (RTL) for the highest priority task, requests enough memory to run the task (via subroutine RMEM), and (if memory is available) initiates loading of the task. The RTL is one of two tables assembled and not set up dynamically by initialization. The other table is a task load request stack with the name CCC. The RTL consists of two-word entries and is currently 120B words long; while CCC consists of three two-word entries with a zero-word terminator. The format of these two internal tables is as follows.

RTL - REQUESTED TASK LIST



name	Task directory index
fl	Field length
cp	Current priority
mp	Maximum priority (future use)
l	Queue length limit
cr	Current ATL entry
ls	First ATL entry
s	System task
d	Non destructive code
c	CM resident
e	ECS resident
re	Rollout table entry address of task to roll in (if for a task roll in request).

CCC - TASK LOAD REQUEST STACK



r	Task roll in flag
usn	Address (-2) of user number for task library
tfl	Task field length
scp	Start of subcontrol point FL
tln	Address of task library name
rda	Random disk address of task

TRANSACTION EXECUTIVE RECOVERY/TERMINATION

In general, recovery/termination involves the following operations:

- Flush buffered journal files
- Flush TOTAL data manager buffers
- Close CRM files
- Issue statistics to the dayfile
- Restart the subsystem

Figure 16-6 is a flowchart of REC, the main control portion of recovery/termination.

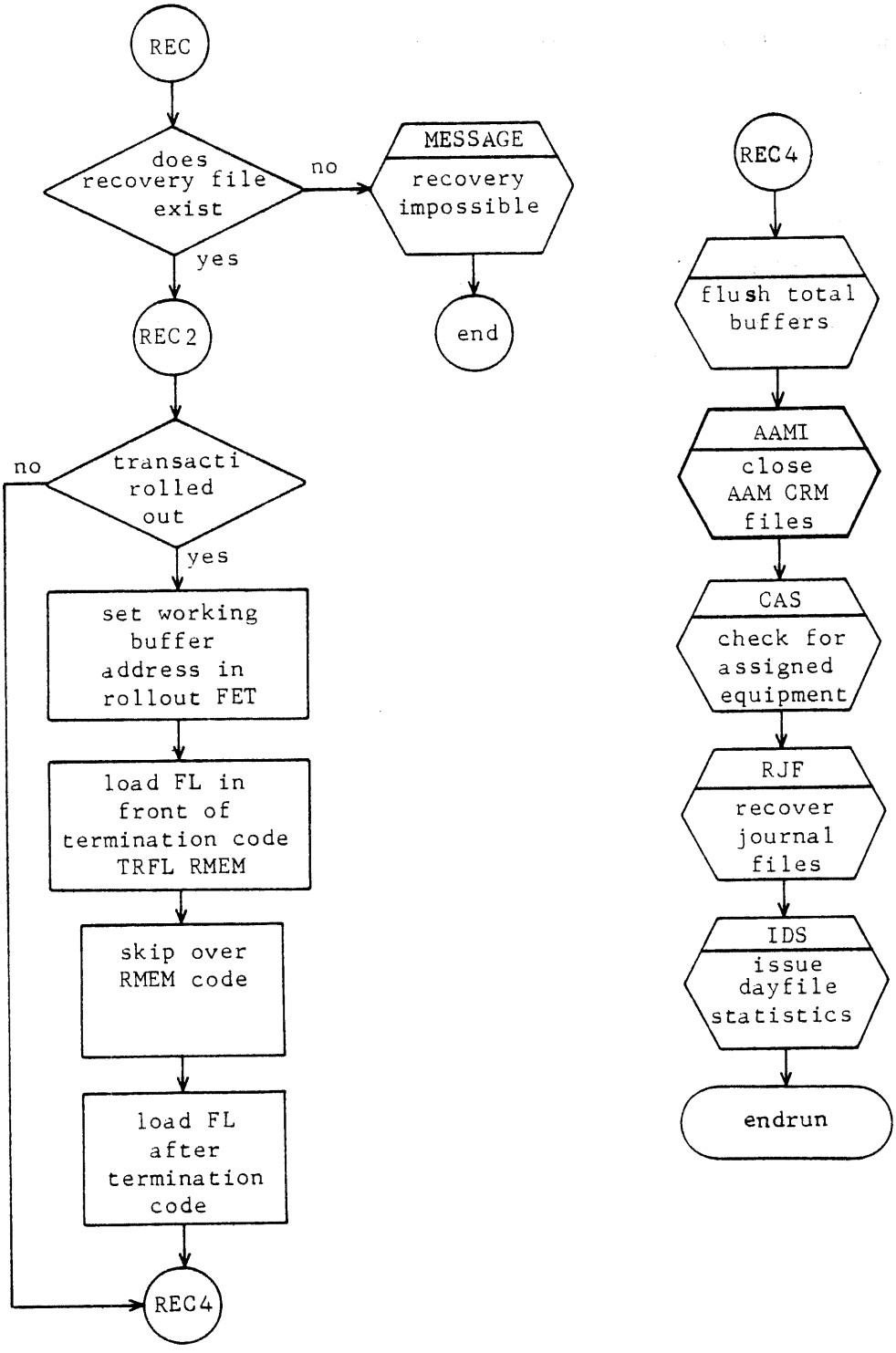


Figure 16-6. REC - Recovery/Termination

TRANSACTION SUBSYSTEM CONTROL POINT

Figure 16-7 shows the breakdown of the TAFTS control point.
Figure 16-8 shows the breakdown of the TAFNAM control point.

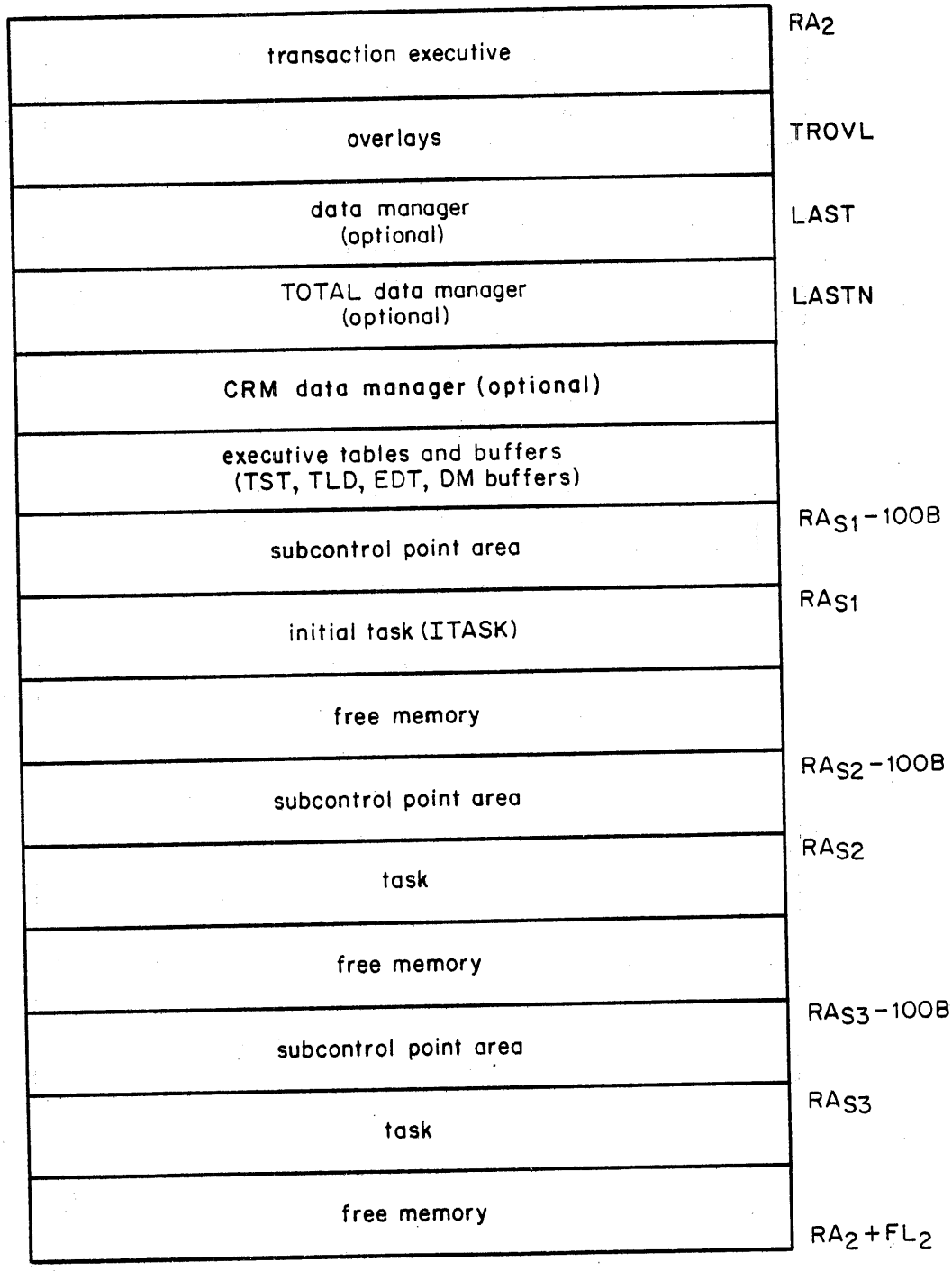


Figure 16-7. TAFTS Control Point

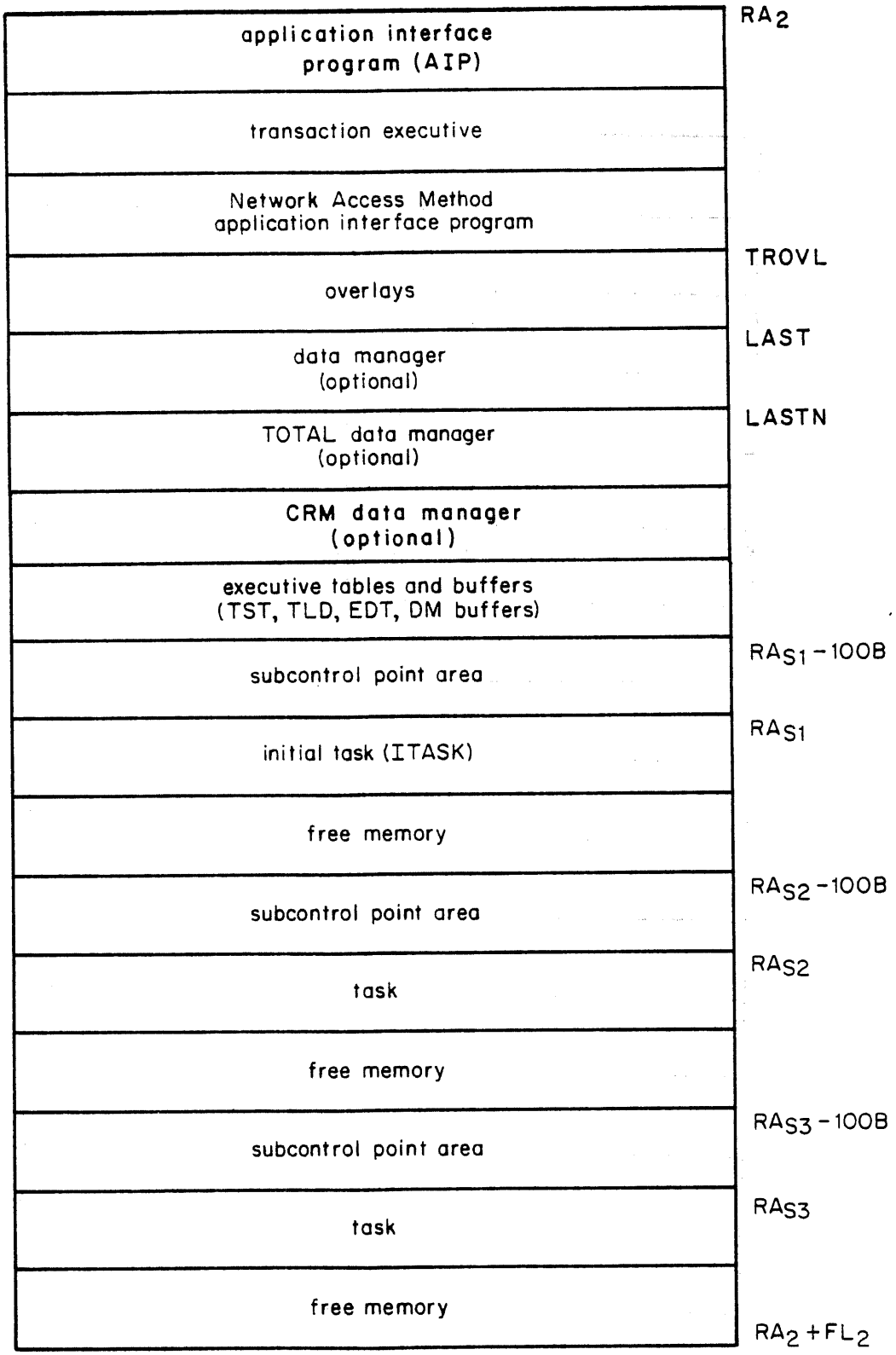


Figure 16-8. TAFNAM Control Point

TAFTS/TIME-SHARING EXECUTIVE INTERFACE

The relationship between TAFTS and the multiplexer time-sharing executive is shown in figure 16-9. The time-sharing executive runs at control point 1, while the transaction executive runs at control point 2. This avoids a storage move of the two executives which would be necessary if they resided at other control points. Transactions are passed between the transaction executive and the time-sharing executive via inter-control point communication. That is, the CPUMTR function, SIC, is used to transfer data between control points.

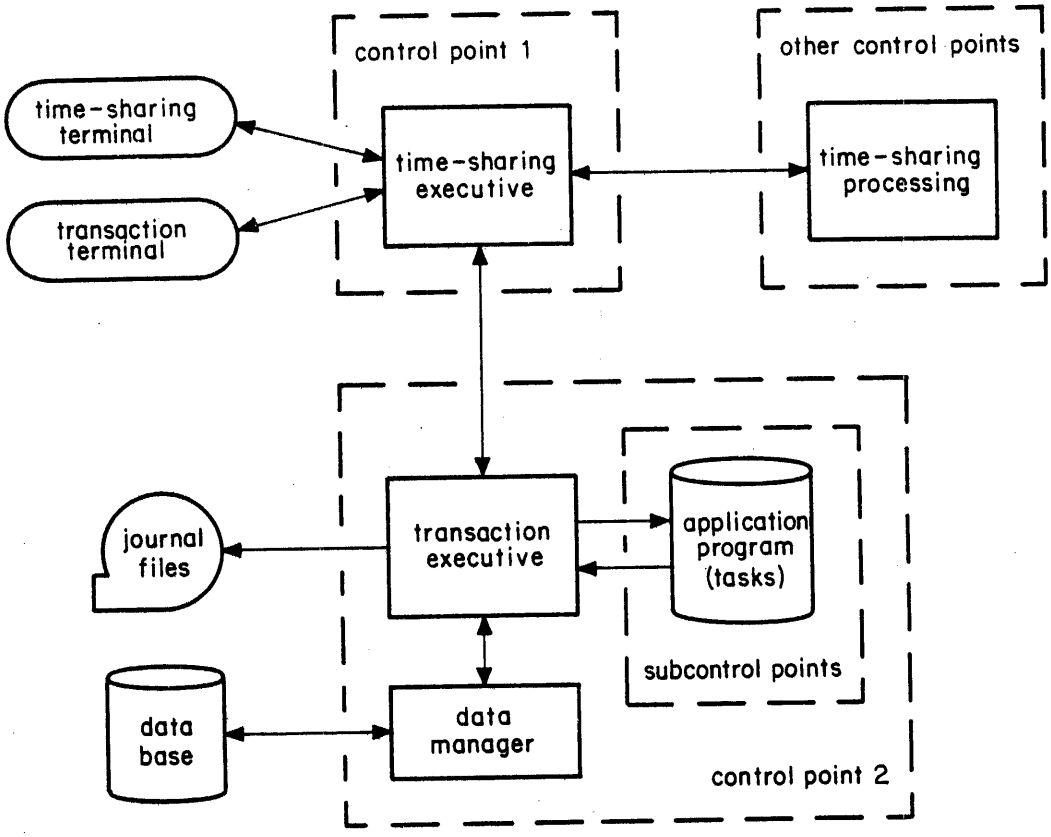


Figure 16-9. TAFTS/Time-Sharing Executive Relationship

TRANSACTION SUBSYSTEM/NAM INTERFACE

The relationship between TAFNAM (using TAF Network Access Method) and NAM is shown in figure 16-10. NAM may run at any control point. Routine NGL formats the input from NAM to look like time-sharing executive input. Processing of terminal input then is done exactly as for the time-sharing executive.

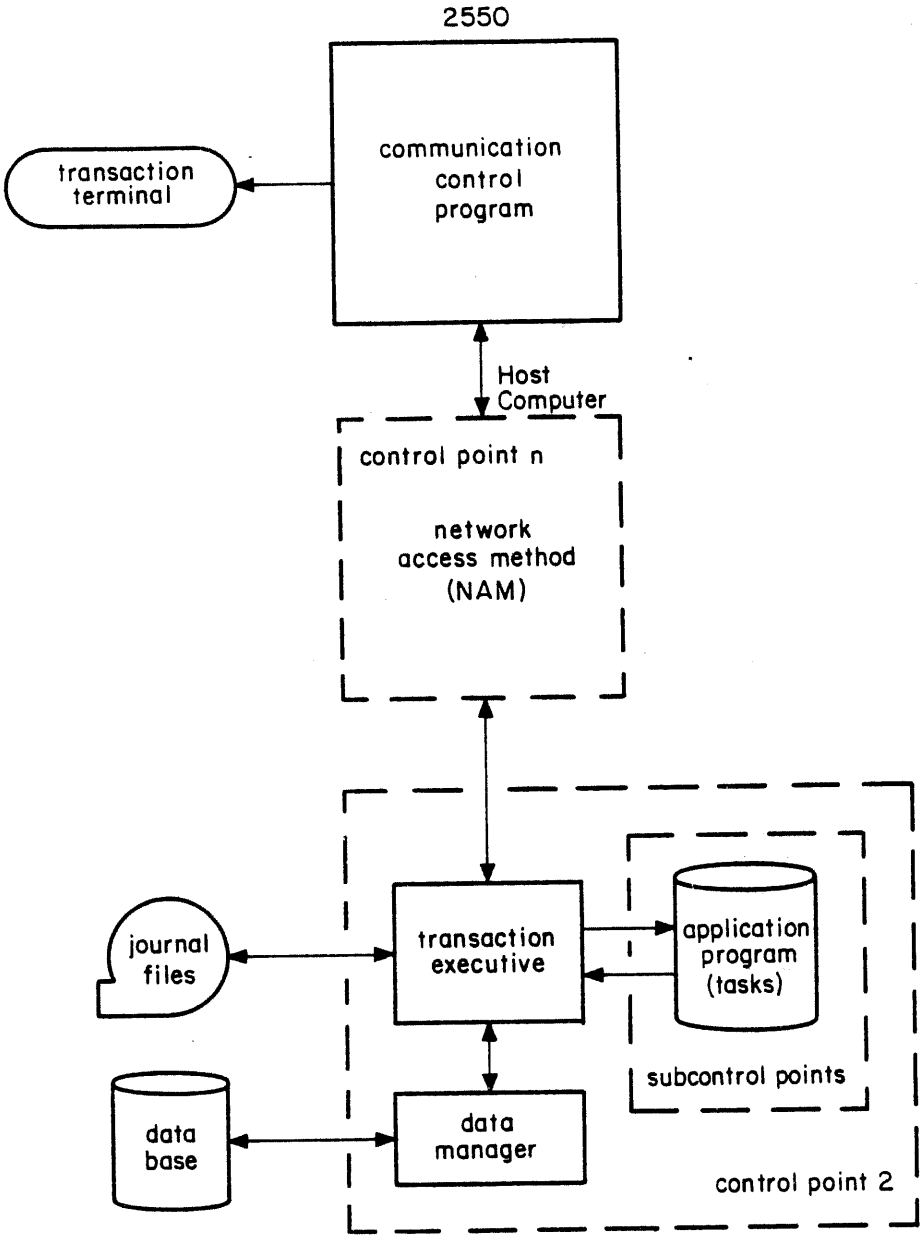


Figure 16-10. Transaction Executive Using Network Access Method

TRANSACTION COMMUNICATION FLOW

The following paragraphs describe the flow of communications for terminal login, terminal input, task scheduling, and terminal output.

TERMINAL CONNECTION TO TRANSACTION SUBSYSTEM

Polled terminals on dedicated lines that are in the network file are defined as always connected to the transaction subsystem.

Terminals under the time-sharing executive that are in the network file become connected to transaction subsystem by issuing the time- sharing command TRAN (refer to the NOS Time-Sharing Reference Manual).

ITASK is notified through a special communication block that a new terminal has been connected to transaction subsystem (refer to the TAF reference manuals).

When TAF uses NAM for terminal communications, terminal connections are controlled according to the NAM network configuration file (refer to the Network Definition Language Reference Manual). Terminal connection may occur:

1. By terminal operator typing TAF in response to the Network Validation Facility (NVF) application prompt.

2. By terminal operator doing a login. Upon login, terminal is automatically connected according to NAM network file.

3. By placing terminal in operating condition. Login and application connection are done according to NAM network file. Terminal may be dedicated or dial-in.

TIME-SHARING EXECUTIVE TO TAF LOGIN

The login procedure to the time-sharing executive is as follows:

1. The time-sharing executive does an intercontrol point transfer to the transaction executive with a function code of three (packed exponent) in the first word or the terminal name in the second word.

2. Either the inner loop, TSSC, or the outer loop, TMDC, calls routine PRIN to process input in the intercontrol point buffer.

3. PRIN does the following:

   a. Checks if previous communication block was processed. If last input in communication block not processed, try to schedule ITASK; otherwise continue at next step.

   b. If the transaction executive is not rolled in, call TRI to roll it in.

c. Checks the data manager activity limit (MDM). If at limit, then exit; else, continue.

d. Calls RDCB.

4. RDCB checks for command/status message and branches to CSM.

5. CSM (command/status message processor) does the following:

a. Branches to terminal login on function code 3.

b. Calls STST (search terminal status table) to find entry corresponding to the terminal name.

c. Checks for valid terminal. If illegal terminal, set error code.

d. Does an intercontrol point transfer to the time-sharing executive with a function code 2012B, the terminal ordinal, the terminal name and the error code.

e. If an error, clear intercontrol point buffer and exit to TSSC.

f. Sets the login flag in the TST.

g. Builds a new message in the intercontrol point buffer which is a system origin transaction, code CILOG, to inform ITASK of the login. This is processed as a normal message input.

h. Return to the inner loop, TSSC.

NAM TO TAF LOGIN

The login procedure to NAM is as follows.

1. The transaction subsystem checks for supervisory messages from NAM in the TAFNAM communication routine NGL.

2. When NGL detects any supervisory message (a flag is set in NSUP by AIP) it gives control to the TAF supervisory message processor (SMP).

3. When SMP receives a connection request (CON/REQ) supervisory message it calls CRE to process the following steps.

a. Match the terminal name specified by NAM to a name in the terminal status table (TST). If no TST entry is found with a matching terminal name, reject the connection because of illegal terminal name.

b. Check login flag in TST; if terminal already logged in, reject the connection. TAF allows only one connection per user name.

c. Put TST address and output block limit in NCT entry word one.

d. Save block size, hard wired line, device type, page width, and page length at terminal for ITASK.

e. Send a connection accepted supervisory to NAM.

4. When SMP receives a connection initialized (FC/INIT) it calls PCI to process the following steps.

a. Set flag in TST to indicate terminal is logged in.

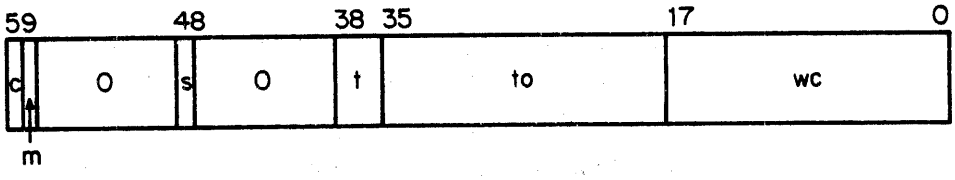
b. Send a normal response supervisory message to NAM.

c. Call routine PRIN to schedule ITASK with login function code and terminal information saved in step 3.d. If ITASK is scheduled, it sends a READY. to the terminal. Otherwise, the supervisory message (FC/INIT) is queued and executed later by this routine unless the connection is broken or a network drop supervisory is received.

INPUT MESSAGE SEQUENCE FOR TIME-SHARING EXECUTIVE TO TAFTS COMMUNICATIONS

The time-sharing executive transfers messages to the transaction executive using SIC RA+1 request. Up to 64 words may be transferred in to a buffer (referred to as the input buffer) at a time.

The input buffer status word (word one of the input buffer) when input is a message is:



c Set to indicate message input. Clear indicates command status message.
m Set indicates more input buffers (block) to follow for this message. Clear indicates last buffer (block) of this message.
s Set indicates a system origin transaction (originated by the transaction executive). Clear indicates a terminal origin message.
t Application character type. Used only for NAM.
to Originating terminal ordinal.
wc Message word count including status word.

When a message is too large to transfer to the transaction executive in one input buffer (block), it will be sent in multiple buffers. The m bit is used to flag this situation. The number of pots per input buffer and maximum number of buffers are parameters in the time-sharing executive. The transaction executive will assign a communication block to each input buffer and assign an input buffer (block) number to each

successive buffer received. The task will receive the data from the first input buffer in its communication block and can read the rest with the LOADCB macro.

Routine PRIN processes the input buffer.

PRIN processes input and queues for ITASK. The following steps are performed.

1. If routine PRIN is called by NGL, go to step 9.

2. If communication blocks are available, continue; else, return to caller.

3. If first word of input buffer is zero, exit.

4. If TAF is rolled out, call TRI to roll in.

5. Check MDM for zero (maximum number of data manager request in progress). If zero, exit.

6. Call RDCB (read communication block) to transfer a transaction input from the input buffer to a communication block.

   a. If bit 59 (command status flag) of first word of input buffer is set, continue; else, branch to CSM to process command status message.

   b. Record time of this input.

   c. Call FFCB to get a communication block.

   d. If no communication block available, exit.

   e. Validate terminal ordinal from input buffer. If terminal ordinal is zero, negative or beyond the table length, then increment counter of number of times input has been thrown away (this is a system bug), clear input buffer, release the communication block, and exit; else, continue.

   f. If bit 48 of input buffer word 1 set (system origin transaction) or bit 17 of TST entry word 2 clear (multiple input bit which when clear means first input block of a transaction input), continue else search communication blocks to determine the next block number for this partial transaction input.

   g. If last input block (bit 58 of input buffer word 1 clear), increment transaction count in TST and continue, else set multiple input block flag in TST (bit 17 of TST word 2) and if first multi-block input. Set block number to one.

   h. Call routine FCB to do the following.

- If first input block, increment transaction sequence count and use new sequence number.

- Format the communication block.

  I. User header contains:

  - Data base name
  - User area
  - Transaction sequence number
  - Terminal name
  - System transaction indicator
  - Multiple block indicator
  - Batch input indicator
  - Parity error indicator
  - Word count of input area

  II. System header contains:

  - CPU priority = 3 (packed format)
  - Terminal read and update securities
  - Communications block address
  - Terminal ordinal
  - TST address
  - Input block count

  III. The following words are cleared

  - Task list (word 3)
  - Dump parameters (word 4)
  - Call with return (word 6)

i. Move message from input buffer to communications block message area with zero fill.

j. If system origin transaction, exit.

k. If terminal not in wait for input state (bit 59 of TST word 1 clear) or not last block of multiple input, then continue, request call routine RTK to rollin of task waiting for input.

l. Put packed date and time in communication block.

m. Call routine JOL to journal transaction. Code 3 if wait for input; code 5 if multi-block input; code 1 if last of multi-block input or first of single block input.

n. If first block of input, exit; else, replace communication block address of current block with address of first block and exit.

7. If input buffer was not last of multi-block input or input was in response to a wait for input, exit.

8. If no communication block was available, set overflow flag and exit.

9. Search for an open entry in ITASK (subcontrol point 1) subcontrol point table (word 4-CPACL) if found, continue; else, put communication block address in overflow flag and exit.

10. Make entry in subcontrol point table status word with communications block address, initial load bit set and waiting for CPU bit set.

11. Increment the communication block count by one in subcontrol point table.

12. If ITASK is active (communication block in execution, word 2 of SCP table bits 0-17 nonzero) exit.

13. Call RCPT to request CPU for task.

    a. Set request CPU bit in CR for the subcontrol point.

    b. If someone has CPU (B2=0) or current transaction is higher priority, exit; else, branch to BNT. BNT assigns CPU to a different subcontrol point by setting B2 to the start of the subcontrol point area and B7 to the address of the subcontrol point table and exits.

14. PRIN exits.

INPUT MESSAGE SEQUENCE FOR NAM TO TAF COMMUNICATIONS

TAF uses the application interface program (AIP) to communicate with NAM. Currently, TAF uses parallel mode which enables it to continue execution after making the network request. Routine NGL periodically checks for the completion of a request and then processes the next statement of the request. The following are the possible sequences of input from the network.

1. Supervisory message. The supervisory message processor (SMP) is called to perform required actions.

2. Small message (MSG) input block. Routine NIT requests a communication block and performs a NETGETL to place the message into the communication block. Routine PLB is called to format the communication block. A task waiting for input or ITASK is scheduled.

3. Large MSG input block. Routine NIT gets an undeliverable block and calls routine PBU. PBU requests NCBN-1 communication blocks, chains them together, and performs a NETGETF request. PLB is called as in step 2. If the communication block is not available, the terminal is put in TEMP OFF state until communication blocks are available.

4. First small BLK input block. Since the message size cannot be determined, PFB reserves NCBN-1 communication blocks and chains them together. Routine PSI is called to process and pack all inputs in communication blocks until the last MSG block. Then PSI calls PLB to process all inputs as in step 2. If not enough communication blocks are available the terminal is placed in a TEMP OFF state and the first communication block is pointed to by the network communication table (NCT) terminal entry until communication blocks are available.

5. Subsequent BLK input blocks. NCBN communication blocks must be reserved and chained together. All inputs are packed into communication blocks until the last MSG block. PLB then is called to process all inputs as in step 2.

TASK EXECUTION FOR INPUT MESSAGE

Either TSSC is returned to or is eventually branched to. The assumption is that a return is eventually made to TSSC after the CPU is assigned to ITASK.

1. Abort flag will not be set.

2. Not a start after recall.

3. No communication block has been loaded for task.

4. Find status word in subcontrol point table for communications block waiting for CPU.

5. Set up control point area.

   a. (CB1C) = word 1 of communication block system header.
   b. (CB2C) = word 2 of communication block system header.

6. Put status word address (in subcontrol point table) in word 2 of subcontrol point table.

7. This is system task so put communication block system header words 1 and 2 in task field length address 111B and 112B.

8. Move user communication block to task field length starting at 111B.

9. Since this is an initial load

   a. Set up exchange package using information in subcontrol point table.

   b. Clear initial load bit in subcontrol point table status word.

   c. Set time slice and RA+1 request limits to default.

d. Clear the folowing:

- Terminal output count
- Rollout threshold terminal word count
- Task RA, RA+1, and RA+2 branch count
- Outstanding data manager request count
- Total data manager request count

e. Set data base name task is validated for.

10. Exchange the subcontrol point.

At this point, the communication block is in ITASK's FL at 111B and ITASK is ready to start execution.

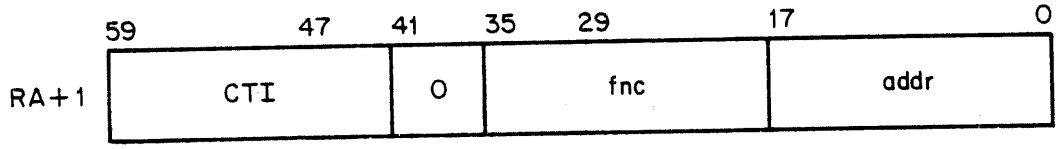
DOWNLINE MESSAGE PROCESSING

A task may send messages to the terminal which originated the transaction (the originating terminal) or another terminal validated for the same data base as the originating terminal using a subcontrol point RA+1 request (CIT, function code 0). A FORTRAN or COBOL task may use the SEND interface deck to format the RA+1 request.

The SEND interface deck is loaded and linked with the application task. It is entered with a return jump and the following parameters.

- Message address
- Message length (embedded in COBOL parameter)
- Terminal (optional)
- Cease flag (optional)
- Sequence flag (optional)
- Block return flag; applies only when NAM is used
- Status return flag; applies only when NAM is used

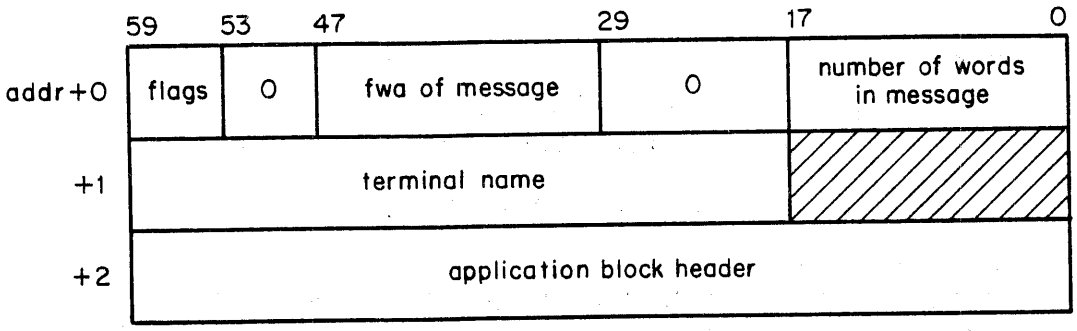
An RA+1 request is formatted as follows.



fnc Function code (SEND=0)

addr Parameter list address

The parameter list at location addr is formatted as follows.



flags Each bit defined as follows:

Bit	Description
59	If set, sent to terminal name in addr+1; else send to originating terminal
58	Task cease requested, if set
57	Output flag; if set, more sends are to follow
56	Return application block number
55	If set, send must wait for block to be delivered to terminal
54	If set, user supplies application block header

The trailing characters not specified in the character count of the message are cleared before the RA+1 request and restored after the request. Also, a zero word is added at the end of the message if the above process does not guarantee 12 bits of zero. It is restored after the request is processed.

The task RA+1 request gives the transaction executive control in SRTN after the XCHNGE macro. The following steps are performed.

1. If exit flag set, check error conditions and process them, else continue.
2. If RA+1 request present, continue else go to TSSC.
3. Clear RA+1 , increment RA+1 count and if limit exceeded, go to error processor.
4. Branch to CTI processor (for SEND).

CTI will process the RA+1 request using the following procedure.

1. If illegal function code or parameter address out of bounds, go to error processor, else continue.

2. Branch to SEND code (based on function code in RA+1).

3. Next.

4. If message is to originating terminal, set message sent flag in communication block system header.

5. If terminal is not originating terminal:

   Find the terminal in terminal status table.
   Validate that task data base is the same as the terminals or that the task is validated for all data bases. If not, go to error processor.

6. If no terminal exists, go to error processor.

7. If terminal not logged in go to error processor.

8. If word count is negative, zero or greater than or equal to MAXWS (64), go to error processor.

9. Add current message word count (from subcontrol point area) to tasks output to date word count. If greater than MAXTO, go to error processor.

10. If message is ahead (at a lower address) of the communication block, go to error processor.

11. If NAM is used for terminal communications go to step 41.

12. The time-sharing exeutive is used for communications to terminals.

13. Get rollout word count (ROWC) from subcontrol point area.

14. If not first send and rollout word count plus current word count are greater than rollout threshold (ROLTO), then set the rollout threshold bit in the SIC buffer header word (bit 58).

15. Update rollout word count (ROWC) in subcontrol point area to include current count.

16. Save word in task ahead of the message and replace it with the SIC buffer header word.

17. A SIC request is formatted (see CPUMTR listing for details on how to use SIC requests). A copy of the RA+1 request is stored in the task's RA+1 without the buffer address being biased by the task's RA.

18. If other task(s) are waiting for SIC (TPLW nonzero), go to 20.

19. Issue SIC request and wait for completion.

20. Restore word used by header of buffer.

21. Check SIC status. If not successful or if other task(s) are waiting, go to 26.

22. Clear task RA+1.

23. If this was a retry, remove next waiting task subcontrol point table address from RSIC word is subcontrol point area and place it in TPLW word. TPLW is checked in TSSC and if enough time has elapsed, it will reenter the SEND routine at step 33.

24. If CEASE requested, go to CEASE entry point in SCT.

24.1 If rollout threshold bit is set, go to 30, else continue.

25. Go to task slicing loop TSSC.

26. Save SIC buffer header word in subcontrol point area word RCLA and CEASE flag in RCL.

    Set time since last rejected SIC to current time.

27. Link tasks waiting for SIC request via TPLW and task word RSIC. Go to routine DCPT to drop CPU for task.

28. Routine DCPT clears control point bit in CPU switching word CR.

29. If CR zero (CPU idle), clear B2 and B7 and exit to inner loop, TSSC, else continue.

30. Using mask in word after CR (set at the end of TMDC), search

    A. Lower control points

    B. Then check the lower control points for highest CPU priority task. Note that the mask in the word after CR ensures that tasks of equal priorities are time sliced one after another.

31. The highest priority task is then activated by setting (B2) to subcontrol point area address, (B7) to subcontrol point table address, and exiting to TSSC. This point is reentered from TSSC after a specified time since last SIC attempt. TPLW contains the address of the tasks subcontrol point table.

32. Set the recall bit in word 2 of the subcontrol point table. Put step 33 in recall address. Request CPU for whichever subcontrol point adds request bit in CR and assign CPU to task if no one has CPU or this task is greater or equal priority. The return is to TSSC. This point is entered by recall processing in TMDC.

33. Reinitialize to try SIC request again

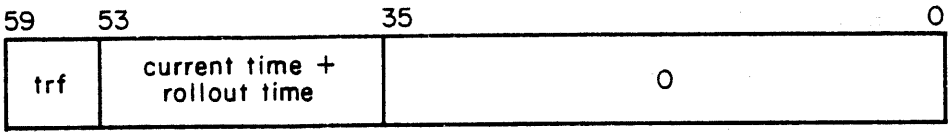
    - o Set cease flag.
    - o Load SIC request from task RA+1
    - o Put SIC buffer header in buffer
    - o Set retry flag
    - o Go to step 19

34. If data manager requests are outstanding, jump to recall routine with step 34 as reentry.

35. If no rollout table entry is available or the prior rollout is not complete, jump to recall routine with step 36 as reentry.

36. Compute rollout time based on line speed and amount of output.

    Build event descriptor word. EVTO is terminal rollout event (20020--0). Build rollout parameters as follows.



    trf Timed rollin flag

    Set time to leave task in core to zero.

37. Clear rollout word count in ROWC in subcontrol point area with sign bit set.

38. Enter event roll routine to rollout task.

39. The time-sharing executive will issue a SIC status command message when the end of the previous output to the terminal is about to be reached. This is processed by CSM and a task with event EVTO is made a candidate for rollin.

40. The task is rolled in by the scheduler and started up to continue processing.

41. The following steps process the terminal output to NAM.

    a. If terminal logged in, go to step c.

    b. If send with recall, return terminal not logged in status to task and go to task switch routine TSSC; else, abort the task.

    c. If terminal does not have a down line stop status, go to step e.

d. If task send with recall, return down-line stop to task and go to task switch routine; else, abort the task.

e. If terminal has send with recall active or the previous network request is not yet complete, go to step z.

f. If outstanding output block is zero, go to j.

g. If task call send with recall, go to step z.

h. If there is no block limit, go to step k.

i. If output does not exceed block limit, go to k; else, go to step f.

j. Set recall bit (TNSR) in NCT, if needed.

k. Format application block header (ABH); if task does not supply ABH, use default values for ABH. Put ABH in NCT.

l. Update outstanding output block counter (TNBO) in NCT.

l.a. If break flag is set, send a reset supervisory message to NAM.

m. Do a NETPUT. Call routine PPM to check network status. If the request is complete, go to step n; otherwise TAF returns to step n later.

n. If send with cease, exit to cease routine SCT2.

o. If send with no recall, exit to TSSC.

p. Compute time that block must be delivered by using current time plus installation assembly wait time and put time in task system area.

q. Set task return to step x upon rescheduling.

r. If message is less than installation defined limit, go to step aa.

s. If task has data manager activity or rollout table is unavailable, go to step aa.

t. Set rollout bit TNSE in NCT.

u. Use ACN and ABN to identify the event and exit to rollout routine.

v. If there is a supervisory message in NCT, copy the supervisory message to the task status address, clear the recall field in the NCT, clear the supervisory message from the NCT, and exit to the task switching routine.

w. If the time for the block delivered supervisory message has passed, return a block not delivered supervisory message to the task status address, clear the recall field in the NCT, zero number of outstanding output blocks in the NCT, and exit to the task switching routine.

x. Task must continue to wait; set task return address.

y. Prepare entry registers for SND and go to step a.

z. Let return address be step y upon task rescheduled.

aa. Put task in recall and exit to executive recall routine.

DATA MANAGER COMMUNICATION

TAF communicates to all data managers (TAF, TOTAL, and TAF CRM) in the same manner. The general procedure is as follows (refer to figure 16-11).

1. A task makes a data manager RA request.

2. CPU monitor detects the request and gives control to TAF.

3. TAF determines the type of RA request and jumps to the appropriate data manager processor (TAF, TOTAL, or TAF CRM).

4. The data manager RA processor enters the task request into the appropriate data manager input queue (TAF, TOTAL, or TAF CRM).

5. Periodically TAF initiates the data manager which removes entries from the input queue and performs the requested function. The TAF CRM interface routine performs all the necessary record and file locking. The TAF CRM request is mapped to the corresponding CRM request. The CRM data manager accesses the data base and transfers blocks to/from the buffer pool. The user data is passed from/to these buffers to/from the working storage areas at the task subcontrol point.

6. When the data manager completes the request, the user function is entered in the output queue.

7. Periodically TAF examines the data manager output queues. The data manager output queue processor removes the entry from the queue and schedules the task for execution.

TAF DATA MANAGER

The general steps for processing a TAF data manager request are as follows.

1. The task begins running and makes a data manager RA+1 request.

2. CPU monitor gives request to TAF.

3. The TAF executive puts data manager request into data manager queue.

4. The data manager is called periodically if activity has occurred.

5. The data manager converts key to PRU address.

6. The records in central memory are checked to see if any records contain desired key. Assume record not in central memory. Records held by task are paged out of memory.

7. A read is initiated for the record. An entry for the request is made in the input/output active data manager queue.

   Upon next entrance to the data manager, a check for I/O complete is made.

8. Assume I/O is complete and no errors have occurred. An entry is put in the active reentrant request data manager queue.

9. Upon next entrance to data manager, a check is made on reentrant request data manager queue. A jump is made to the appropriate processor.

10. The element is converted from format of record to the format of user task area and is moved to the task.

11. A data manager complete entry is made in the data manager output queue.

12. When the data manager has completed one pass of the data manager input queue, control returns to the TAF executive.

13. The TAF executive processes the data manager output queue. The CPU is requested for the task.

14. The TAF executive does an exchange with the task. The task has the element.

The steps for processing a TAF CRM data manager request are as follows.

1. The TAF executive in routine AAM places the task request in the AAMIQ input queue.

2. TAF CRM is called periodically if any activity has occurred.

3. TAF CRM tries to process any requests with active input/output. If the data record has not been retrieved another seek is done and the next request is processed. If the data record has been retrieved, the record is moved from the TAF buffer to the task buffer.

4. An entry is made in the AAMOQ output queue. The TAF executive examines this queue after TAF CRM returns to the executive and starts up the task.

5. After processing all active requests, the AAMIQ input queue is processed.

6. If the requst needs to lock a record, the record is locked. If a lock cannot be obtained, all locked records for a transaction are released.

7. The CRM request corresponding to the TAF function is done. If the request does not require input/output, an entry is made in the AAMOQ output queue; otherwise the first SEEK request is initiated.

8. Control is returned to the TAF executive after one pass is made of the input queue.

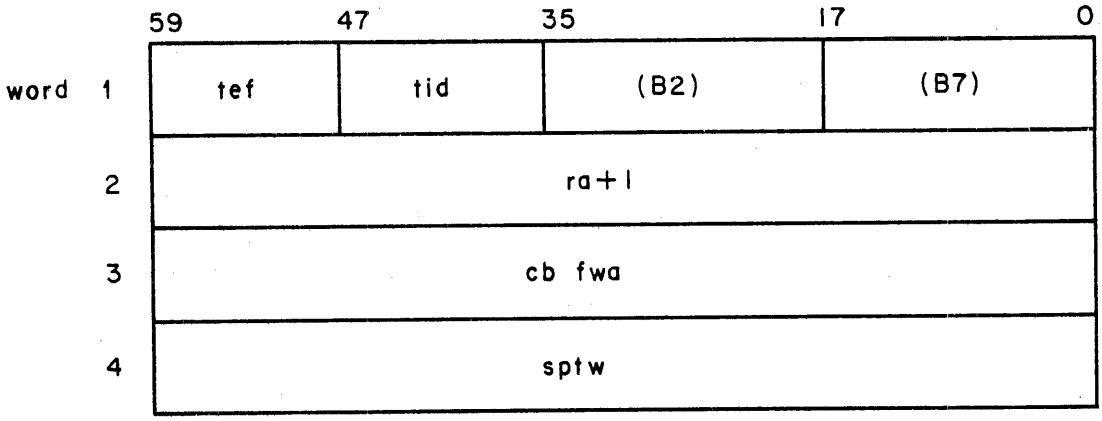
INTERNAL TASK XJP TRACE

As an aid for locating a problem if TAF aborts as a result of a task RA+1 call, the internal task XJP trace is provided. Information related to the task being processed is stored in a circular buffer within the field length of TAF by the trace. If TAF aborts, the contents of this buffer can be examined if a dump of TAF has been taken.

Upon return from subcontrol point activation in SRTN, a call is made to the internal task XJP trace processing subroutine TXT under one of the following conditions.

- If an RA+1 request from the task is present
- If an error flag is set and the error is not time-slice exceeded or the error is time-slice exceeded and the total number of time-slices allowed has been exceeded

Each time it is called, TXT sets up and stores one packet of task-related information into the circular buffer PBUF. The length of a trace packet is defined by the symbol ITTPL as being four words. The format of each trace packet is as follows.



tef	2000B plus the error flag returned from subcontrol point activation
tid	Task trace packet identifier (set to zero)
(B2)	Start of the system area of the task currently selected for CPU assignment
(B7)	Start of the subcontrol point area of the task currently selected for CPU assignment
ra+1	Contents of RA+1 of the task field length
cb fwa	First word of the communications block kept in the system area preceding the RA of the task
sptw	Third word of the subcontrol point table for the task

The FET for the internal trace buffer PBUF is found at the symbol INTRACE. The length of the buffer is defined as 30*ITTPL or 170B words. It contains enough space to store the last 30 packets of four words each which were produced. The information is not written to any external file.

INSTALLATION MODIFICATION OF INTERNAL TRACE

The following are provided as guidelines should an installation wish to either make modifications to certain aspects of the internal XJP trace or write its own internal trace code to trace other events in TAF. The internal XJP trace processing subroutine TXT is based upon these guidelines and not following them may cause TXT to work improperly.

1. Since the code for the internal trace is executed frequently, it should be simple and fast, essentially straightline.

2. PBUF is intended to be the circular buffer used for storing all internal trace packets; those generated by TXT and also those generated by any installation code. PBUFL, the length of the buffer, is defined as the total number of packets that can be stored in the buffer at one time multiplied by ITTPL, the packet size (four words at present). For example, by default PBUFL is defined as 30*ITTPL. This provides enough space to store up to 30 trace packets of four words each.

3. All trace packets, from both TXT and installation code, are the same length. This length is defined by ITTPL as four words. If the length is changed, ITTPL must be redefined. Only lengths of four or more words are acceptable.

   Although all packets are of the same length, the number of words of trace information stored in each may not be equal to ITTPL. This situation can occur if, for example, an installation routine stores trace packets of five words each and TXT stores four. ITTPL will have been redefined as five. The last words of the packets stored by TXT should then be ignored. This unused word occurs because of the way TXT updates the IN pointer (refer to guideline 4).

4. The IN pointer of the INTRACE FET is updated so that the following are assured:

   - All packets are of length ITTPL.
   - The circularity of the buffer is maintained. That is, when the IN pointer equals LIMIT, set the IN pointer equal to the beginning of the buffer.

   It is strongly suggested that an installation use the code in TXT as a guide for updating the IN pointer. TXT calculates the value of the IN pointer for storing the next packet by taking the value of the IN pointer for storing the first word of the current packet and adding ITTPL to it.

5. In order to identify the type of event being traced, a unique task trace packet identifier TID should be assigned for each event and be stored into bits 47 through 36 of the first word of each packet. This field is set to zero to identify those packets stored for the task XJP trace.

   The TID can also serve as an indicator to the number of words in the packet that contain trace information. (This pertains to the situation where some packets contain less than ITTPL words of trace information.) Refer to figure 16-11 for more information.

TAF TROUBLE-SHOOTING

The following steps provide an orderly process for trouble-shooting when errors occur in TAF.

1. Examine the TAF dayfile. Setting sense switch 6 assures that the job dayfile is printed upon termination.

2. Insert DMP statements after the EXIT statement in the TAF procedure file.

3. Examine the exchange package. Register B2 usually points to the context block (task system area) for the currently executing subcontrol point. Register B7 usually points to the subcontrol point table for the currently executing task.

4. Examine word CR where set bits indicate which subcontrol points are candidates for execution. Bit 46 corresponds to subcontrol point 1 (ITASK), bit 45 corresponds to subcontrol point 2, and so on.

5. Examine word RCR where set bits indicate which subcontrol points are in recall.

6. Examine word VCPA to find the location of the subcontrol point tables. Examine the subcontrol point tables indicated by register B7, word CR, and word RCR.

7. Check the communication blocks indicated by subcontrol point tables to examine terminal input.

8. Check other communication blocks that are being used. Word VCBRT gives the first word address of a map for the communication blocks. Zeros in bits 47 through 0 indicate communication blocks in use. Word VCBSA gives the first word address of the start of the communication blocks.

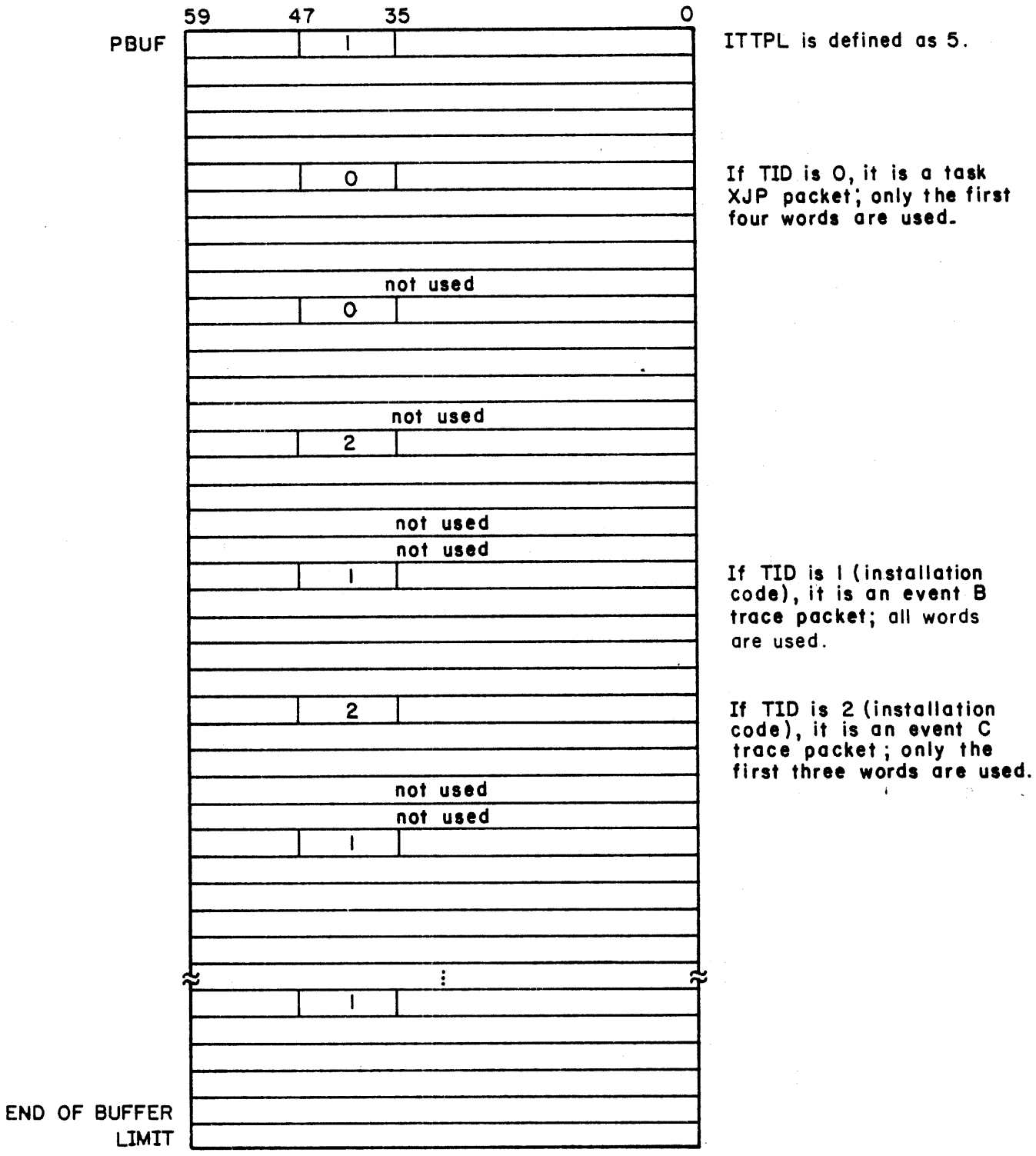


Figure 16-11. Trace Buffer Layout

9. Check the task system area (context block) of subcontrol points that are active or are in recall. The task system area is at RA minus 100B from the RA given by the subcontrol point table.

10. The following words are of special interest in the task system area.

Word	Description
XJPC	Task exchange package
LRA1	Last RA+1 call
RCL	Address for recall
RCLA	Parameters for recall
ERRC	Task error code (zero if a valid error code)
DMEC	Data manager error code

11. Examine the rollout table. Word VROLT gives the location of the rollout table allocation map. Zeros in the map in bits 47 through 0 indicate active rollout table entries. The rollout table follows the mapping words.

12. Examine terminal input as follows.

    - For communications with the time-sharing executive examine word INRB
    - For communications with NAM examine file ZZZZZDN, the AIP debug file. This file is produced by using library NETIOD to obtain the AIP.

13. Examine the journal files using TDUMP. Useful information may also be found in the JOUR0 FET located at symbol PJRNL.

14. Examine the data manager input/output queues DMIQ/DMOQ. The FETs for these queues are at symbols DI and DO.

15. Examine the data manager buffer status table (BST) to find current records. Active BST entries are given by a map pointed to by the word VBSTR.

16. Examine the internal trace information in the buffer PBUF. The FET for this buffer is found at symbol INTRACE.

LIBTASK UTILITY

The LIBTASK utility is used to create a new library, add new tasks to an existing library, replace tasks on an existing library, and compact an existing library by removing inactive records. Tasks may be added or replaced while the transaction subsystem is running, as tasks are added as a new file at the end of the library. The library is multifile in format. The transaction executive attempts to read a new library directory if the LIBTASK control statement contains a TT parameter.

If the library is modified while the trasaction subsystem is running and the user wants the new directory used, the user number and password of the LIBTASK user are passed to the transaction subsystem for validation. This is done by intercontrol point communication, which is also used to check if the transaction subsystem is running if the CR or PR option is selected. Therefore, the first USER statement in the job must have permission to perform intercontrol point communication. The LIBTASK run issues an error message and does not update the library if the user is not validated (refer to section 20 for information concerning user validation). The task binaries to be added or replaced must be in absolute overlay format and greater than 64 words in length. The first word address must be between 77B and 1000B. The format is that produced by the CYBER Loader.

The main flow of LIBTASK is shown in figure 16-12.

PRS-PRESET ROUTINE

PRS cracks the control statement and sets up for the main program. Error checking is performed on control statement parameters. If any error occurs, LIBTASK aborts with an error message. All files are initialized and input directives are processed by LIBTASK (default value is used if an error occurs in a directive). Routine PRS is flowcharted in figure 16-13.

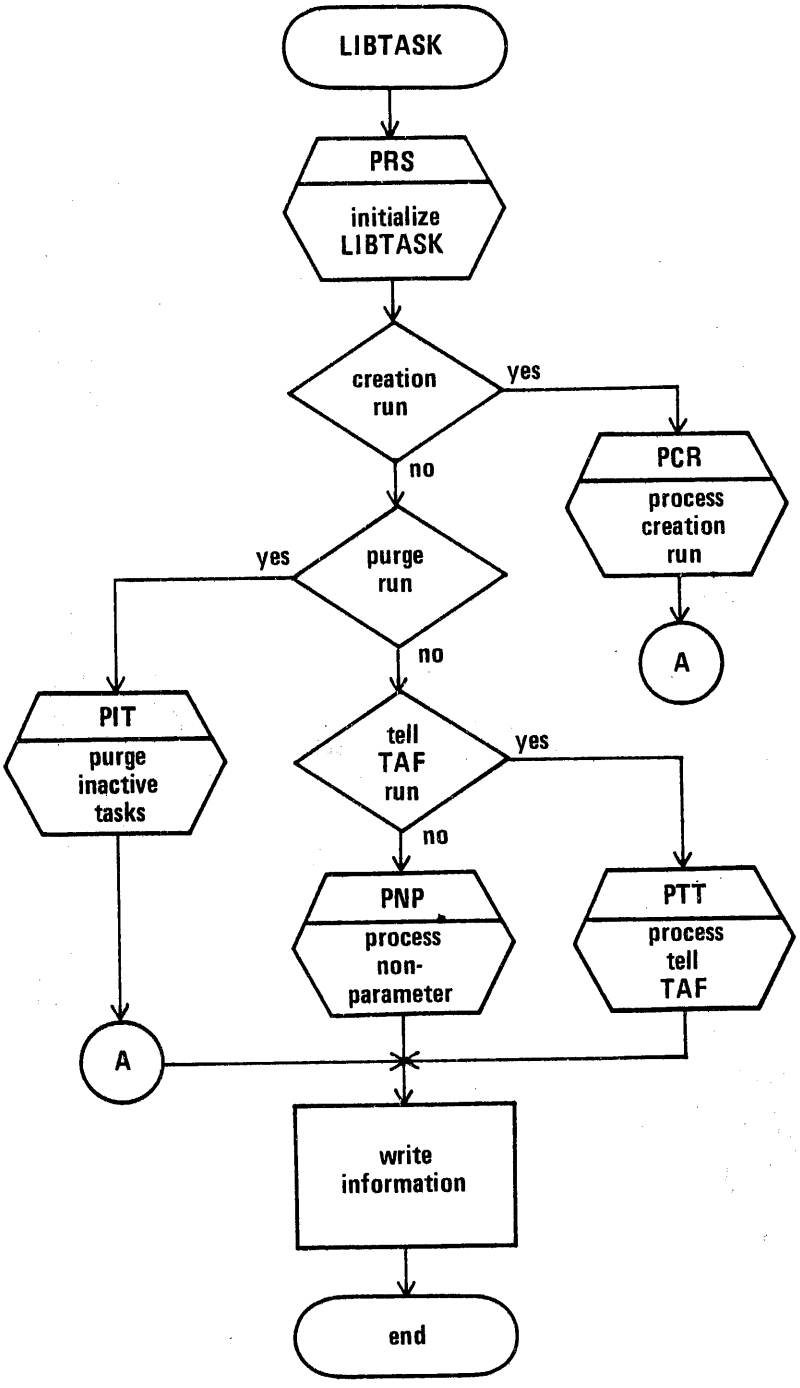


Figure 16-12. LIBTASK Main Flow

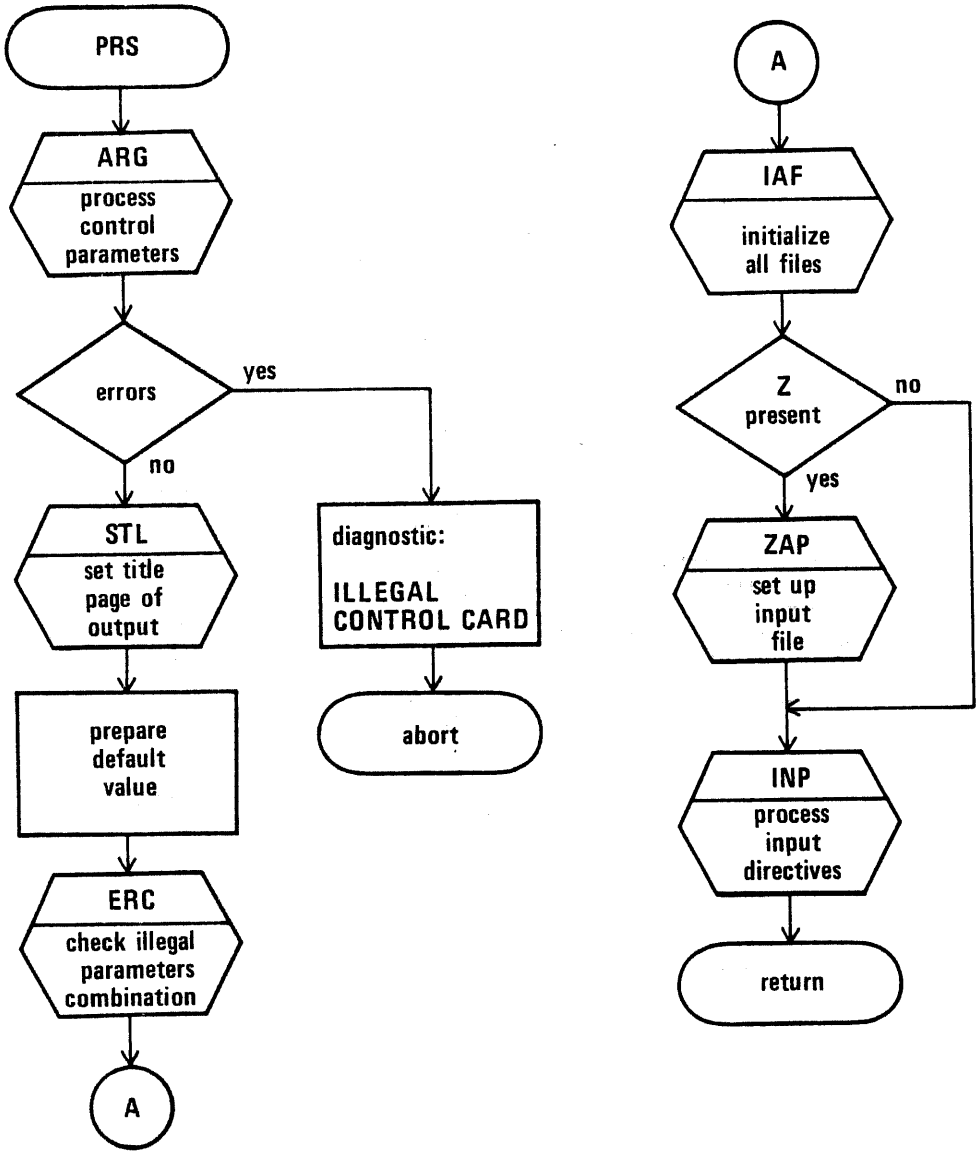
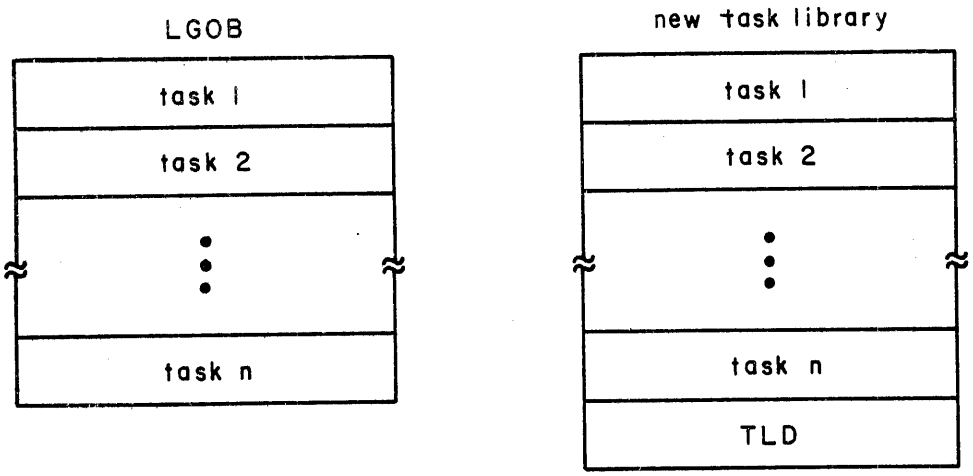


Figure 16-13. PRS - Preset Routine

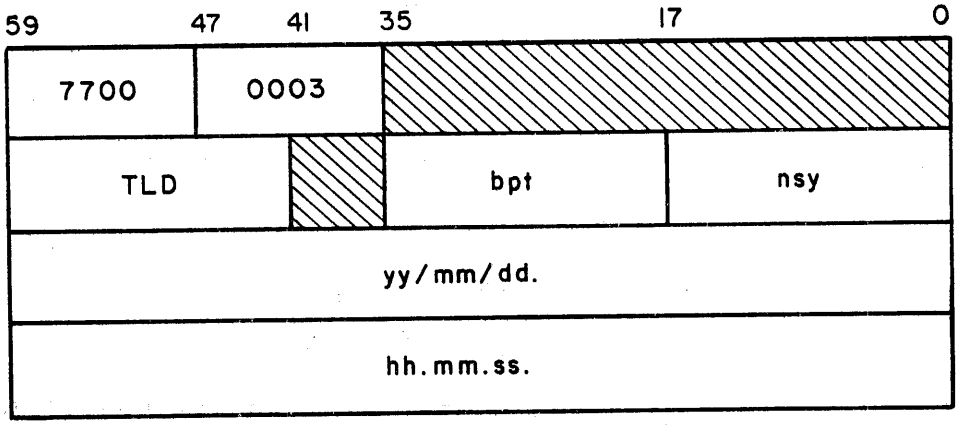
LIBTASK creates a task library from the specified binary file and adds the task library directory (TLD) to the end of the task library as the following illustrates:



Task Library Directory

The TLD consists of a four-word header and a three-word entry for each task in the library, both of which are constructed automatically by LIBTASK. The directory is the last record in the task library.

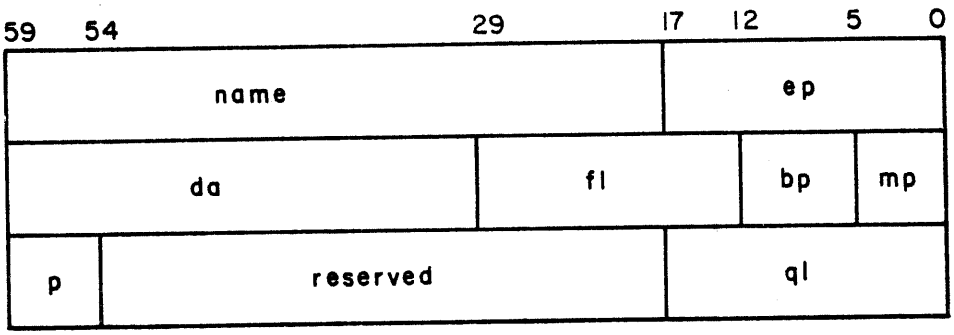
The format of the header which includes the date and time of the last modification of the library is as follows.



bpt Relative word in directory where tasks are added after creation of library.

nsy Number of system origin tasks.

The format of a directory entry is as follows.



name Task name, left-justified, zero filled
ep Entry point address
da Mass storage index
fl Field length required for task
bp Beginning priority
mp Maximum priority
p Each bit defined as follows:

Bit	Description
59	System task (special system privileges) if set
58	Destructive code (reload binaries after use) if zero
57	Memory-resident task if set
56	ECS-resident library copy if set
55	Task ON if zero
54	Task active if zero
53	Solicit input

ql Maximum number of transactions to queue for this task

The default values for the directory are as follows.

- Beginning priority (bp) is 20B
- Maximum priority (mp) is 40B
- Queue limit (ql) is 3
- Task is mass storage resident
- Task is not a system task (p bit 59 zero)
- Task code is nondestructive (p bit 58 set)
- Task is not a memory resident task (p bit 57 zero)
- Task is not an ECS resident task (p bit 56 zero)
- Task is ON (p bit 55 zero)
- Task is not logically deleted (inactive) (p bit 54 zero)
- Task does not solicit input (p bit 53 zero)

Figure 16-14 illustrates the flow for routine PCR.

PTT - PROCESS TELL TAF OPTION

The task library can be updated while the library is attached by TAF. The transaction executive attempts to read the new library directory if the LIBTASK control statement contains a TT parameter. The format of the task library is shown in figure 16-15.

Figure 16-16 illustrates the flow for routine PTT.

PIT - PURGE INACTIVE TASKS

Inactive tasks can be removed from the task library when the library is not attachd by TAF. LIBTASK also removes inactive tasks when the purge option (PR) is specified on the LIBTASK control statement. The format of the task library is shown in figure 16-17.

Figure 16-18 illustrates the flow for routine PIT.

PNP - PROCESS NO PARAMETERS

The user does not have to specify the CR, PR, or TT parameters. If P=0, LIBTASK performs a creation run. If the file specified by N is attached by TAF, LIBTASK processing is the same as the TT option with sorted directory at the end of the library file. When P is nonzero and N file is not attached by TAF, the PR option is assumed.

Figure 16-19 illustrates the flow for routine PNP.

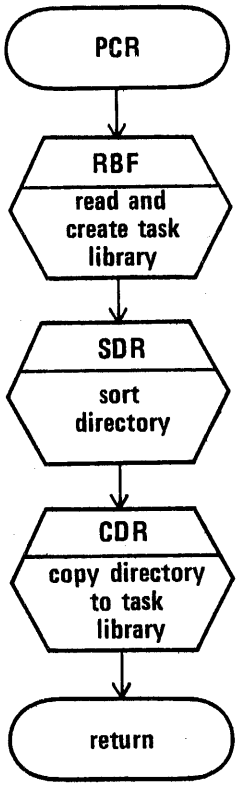
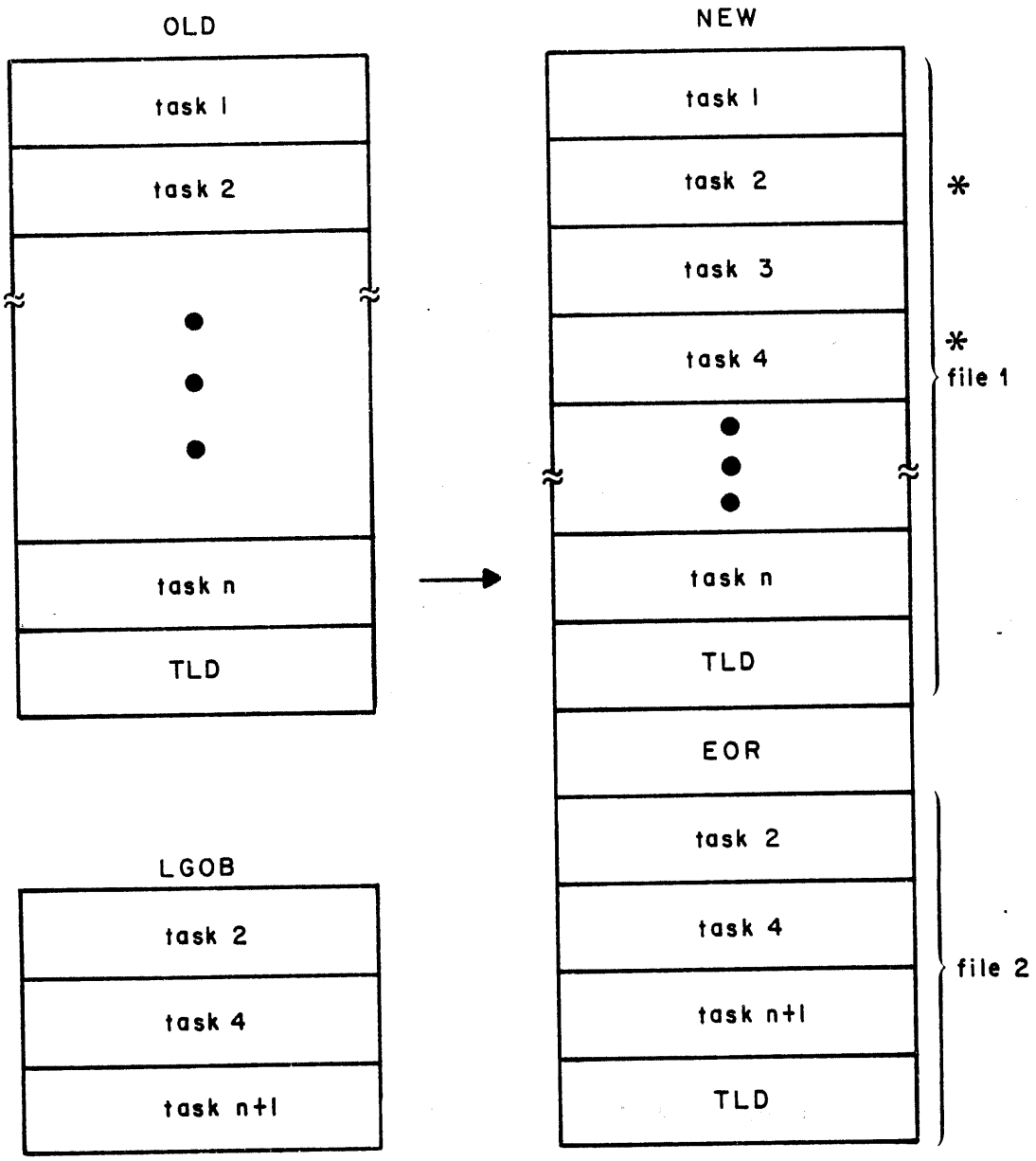


Figure 16-14. PCR - Process Create Option



* Task 2 and task 4 in file 1 will not be used by TAF.

Figure 16-15. Library Format

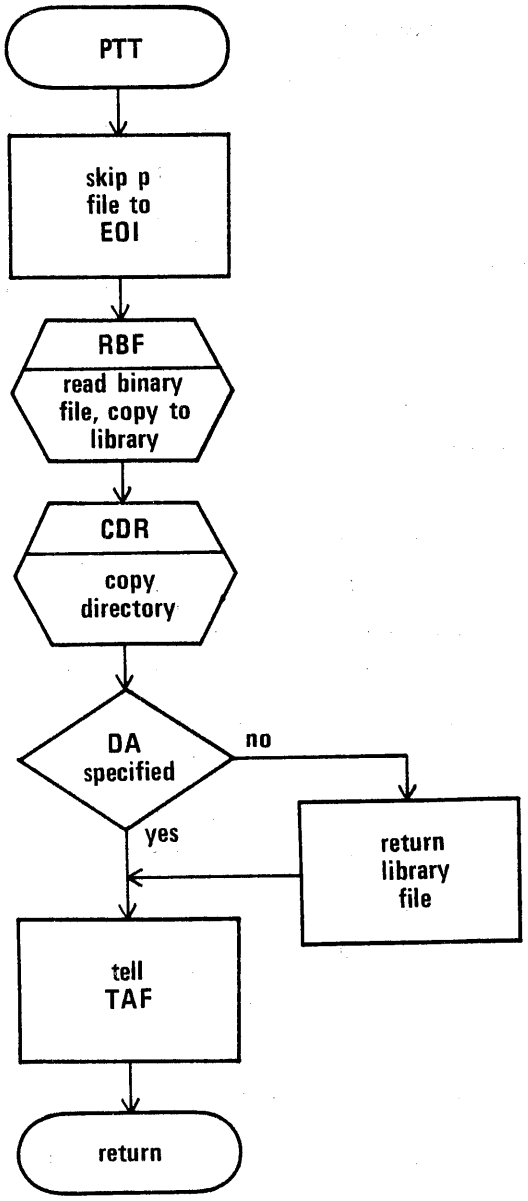


Figure 16-16. PTT - Process Tell TAF Option

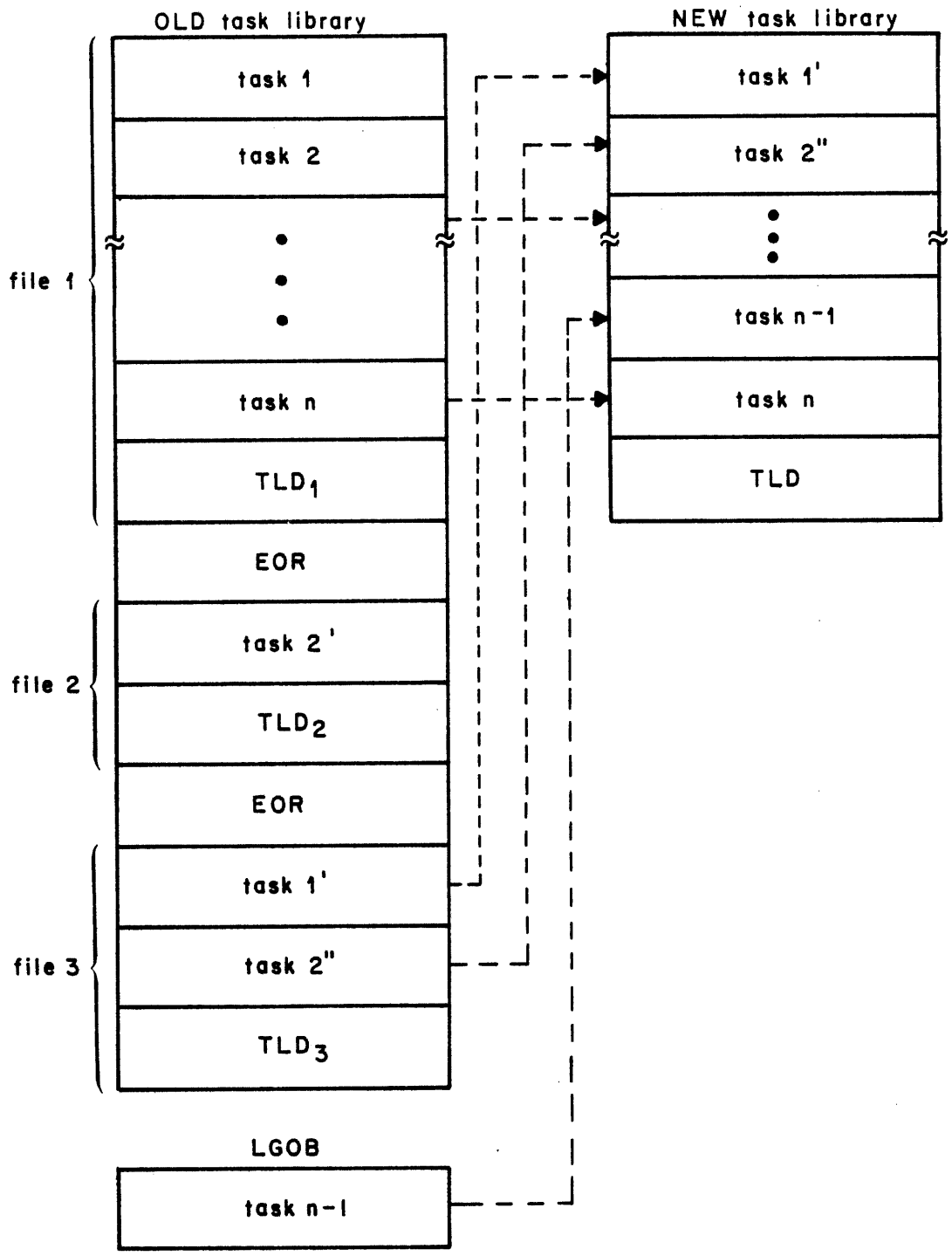


Figure 16-17. Task Library Format

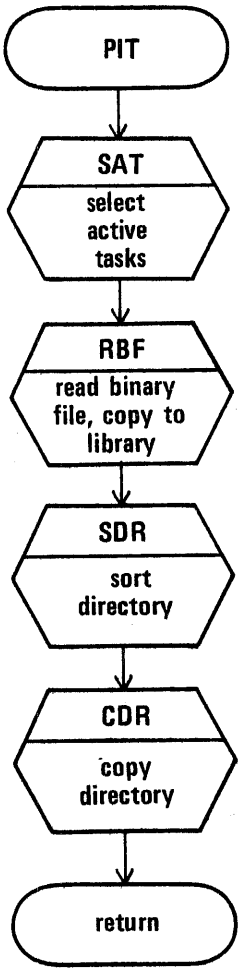


Figure 16-18. PIT - Purge Inactive Tasks

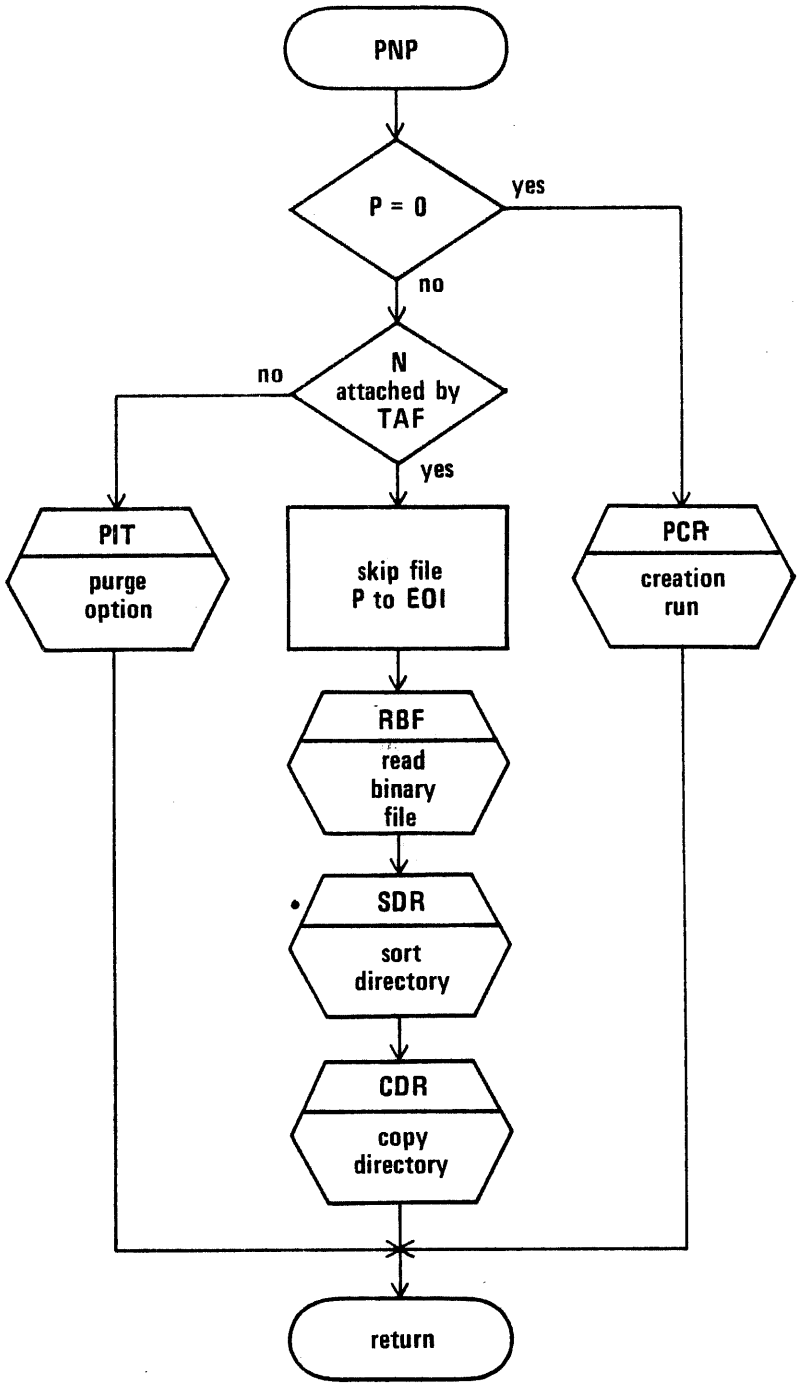


Figure 16-19. PNP - Process No Parameters

PRODUCT SET SUPPORT MONITOR REQUESTS

A subset of the standard NOS monitor functions is supported by TAF in order to allow product set binaries to function in the transaction environment. The monitor functions are processed with recall, even if not specified by the task.

SFP D00 REQUEST

The purpose of the SFP D00 function is to retrieve message texts that are stored on an STEXT record residing on a system file. Many of the product sets install in such a manner that the error texts which the object libraries use to issue dayfile messages at abnormal termination are resident on a system file and must be retrieved by the D00 processor. This technique allows the field lengths of the various products to be reduced by the sizes of the individual texts. The format of the D00 call is described in section 30.

Since the transaction environment does not support the concept of dayfiles for tasks, the option to transfer the error message that might be generated by a task to the transaction executive dayfile is not supported. Tasks with this option are aborted by the executive if they issue a D00 request.

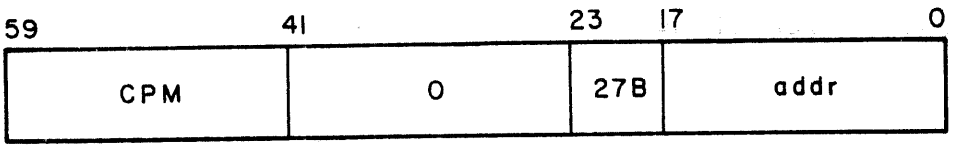
Excessive use of the D00 request by tasks can result in a performance degradation of the TAF subsystem. D00 is implemented as an overlay to SFP which requires that the call be made with autorecall. Thus, the task request is actually made by the TAF executive, resulting in the inoperativeness of the entire TAF subsystem for the duration of the call. Therefore, a site should not use this function to reduce the general size of tasks issuing error messages by installing an error text on the system file and then accessing it through D00.

CPM (27B) - GET JOB ORIGIN

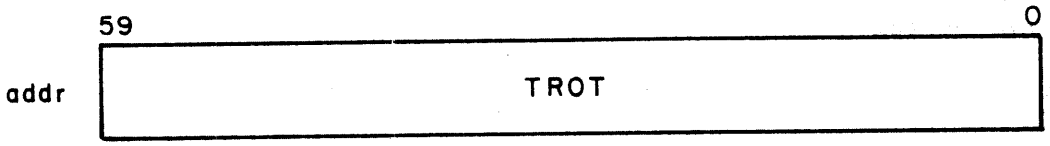
Product sets can determine whether or not they are running in the transaction environment with the get job origin CPM function. A transaction origin job returns a job origin type of TROT.

TROT is a pseudo-origin in that the operating system is not involved in TAF internal scheduling or processes. TROT is useful for differentiating task and TAF executive CM and CPU usage and in determining execution-time environments.

The format of the call is as follows.

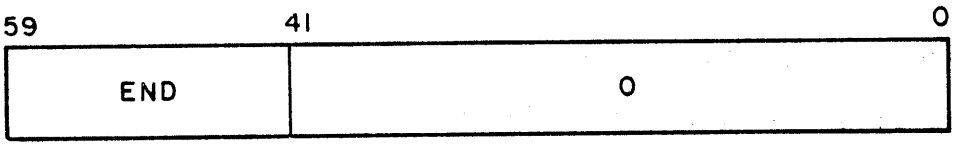


Upon completion, location addr has the following format.



END - END CPU PROGRAM

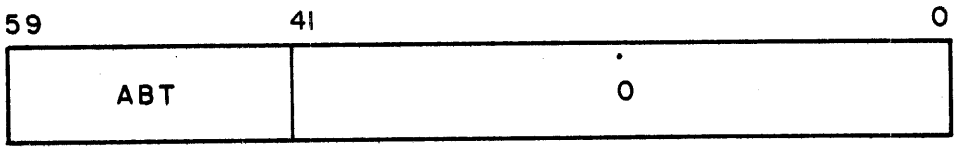
The END request (ENDRUN macro) allows a termination path through the current object libraries without explicitly issuing a TAF CEASE function. This function is mapped into a CEASE in the function handler. The format of the END call is as follows.



There is no response to the END function, since the CPU cannot be returned without a job step advance or task switch.

ABT - ABORT CPU PROGRAM

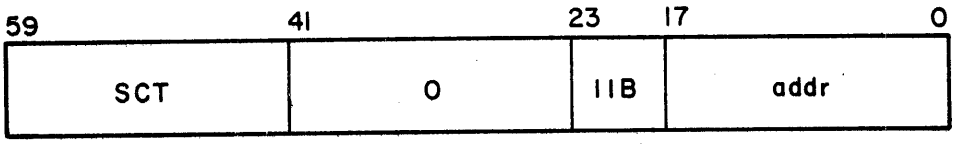
The ABT request (ABORT macro) maps into a CEASE with abort by the function handler. The format of the ABT request is as follows.



There is no response to the ABT function.

SCT - BUFFER WAITINP

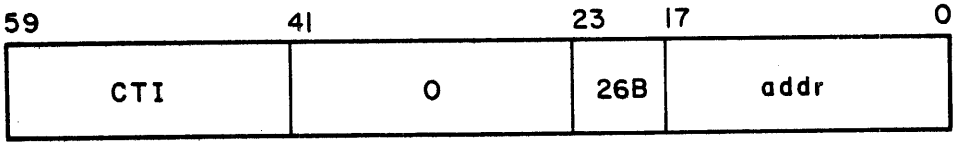
The ACCEPT COBOL verb is supported by this function, rather than by CIO. The format of the SCT call is as follows.



addr FWA of buffer to pass the terminal input to. The usual FWA of load is not the default residence for the buffer which is the argument to this function.

CTI - TPSTATUS

The terminal control attributes of the two support teleprocessors use the CTI monitor function which allows a task to determine which it is running under in order to issue the appropriate SEND requests. The format of the CTI call is as follows.

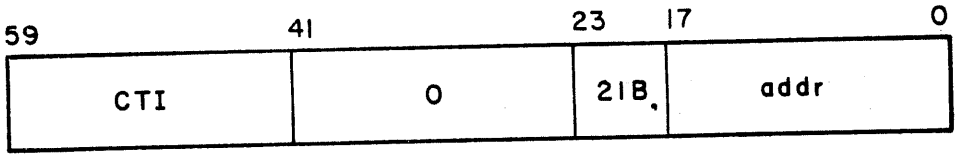


addr Address to receive the active teleprocessor code

0 TLXTP
1 NAMTP

CTI - BEGIN

The CTI BEGIN function returns the communication block which contains the primary terminal transmission. The user may specify the location of the buffer with this request and the area where the normal communication block resides is not overwritten unless that location is specified. This request supports higher level languages which cannot reserve the FWA of the task load for user declared common storage. The format of the call is as follows.



addr FWA of buffer to receive the communication block.

Tasks that use this feature must declare the tasks to be solicited communication block requests with the *SC LIBTASK request. Also, if an *SC task is invoked through the CALLRTN request, a BEGIN request must be issued by the calling task in order to obtain the communication block which is the usual data communication mechanism between two tasks.

INTRODUCTION

The BATCHIO subsystem processes data transfers between the unit record equipment (card reader, CR; card punch, CP; and printer, LP) and the operating system. BATCHIO basically performs the following functions.

- Reads cards from the card reader, creates the input file, and enters the job into the input queue.
- Locates files in the print queue, locates a free printer, and prints the file on the printer.
- Locates files in the punch queue, locates a free card punch, and punches the file on the card punch.
- Processes the DSD operator commands issued for the specified file currently being operated on at a given buffer point. Each device is entered into the available equipment table, TAEQ. The index to each entry is the buffer point number. That is, the first entry is buffer point 1, the second entry is buffer point 2, the last device is n. All information passed to and from BATCHIO for DSD/1DS is done using the logical equipment number.

The subsystem consists of three PP programs and one control point (figure 17-1).

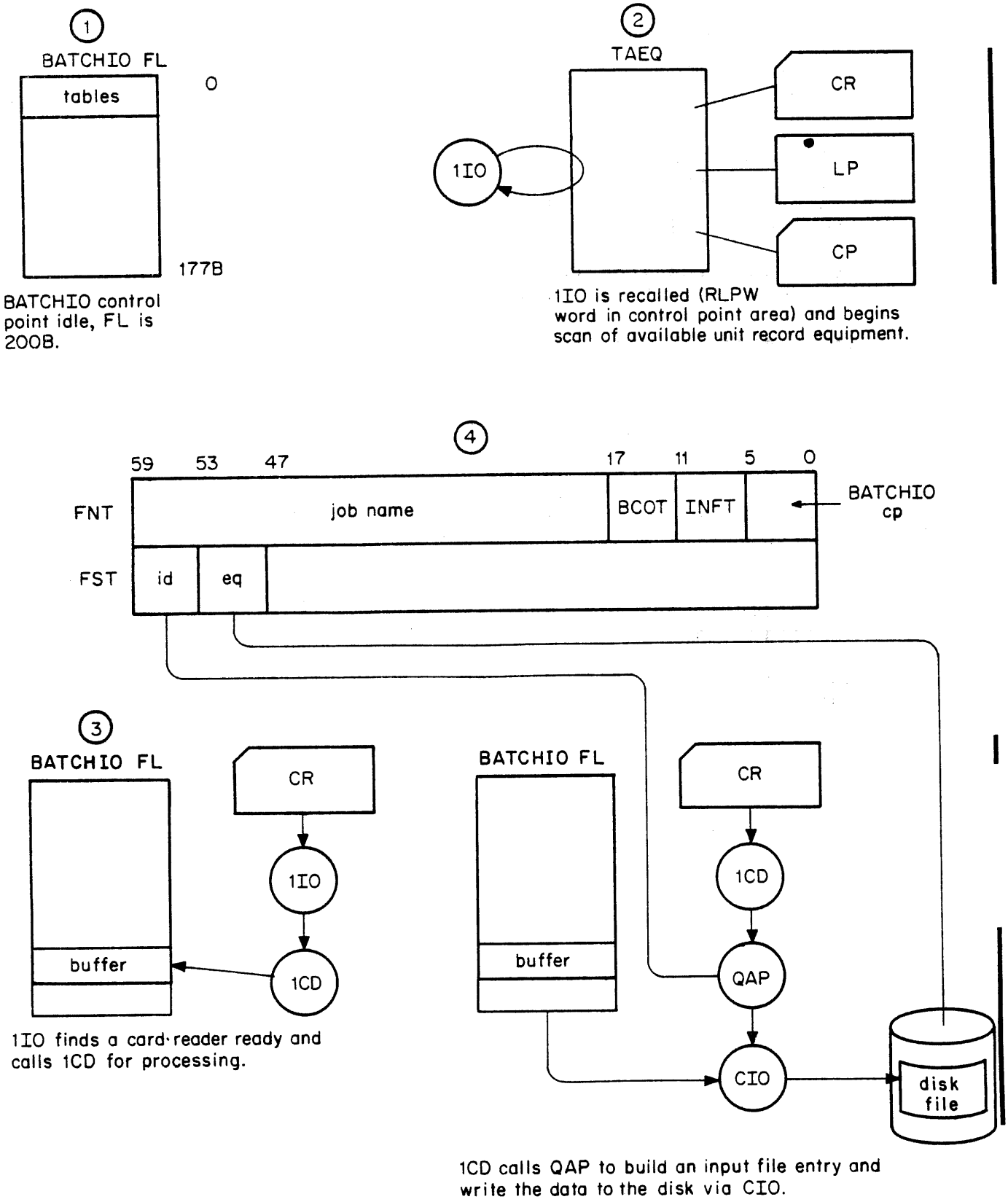
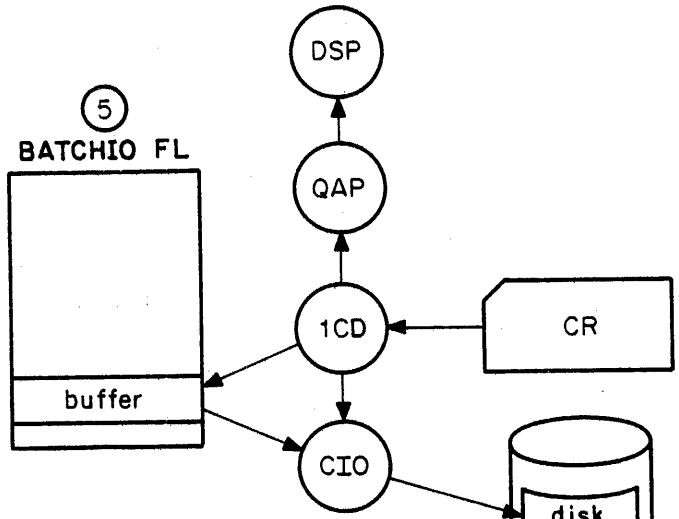


Figure 17-1. BATCHIO Overview



1CD reads the card reader, writes the data into a CM buffer, and calls CIO to write the CM buffer to the disk. After the file has been completely read, 1CD calls QAP who in turn calls DSP to put the file into the input queue.

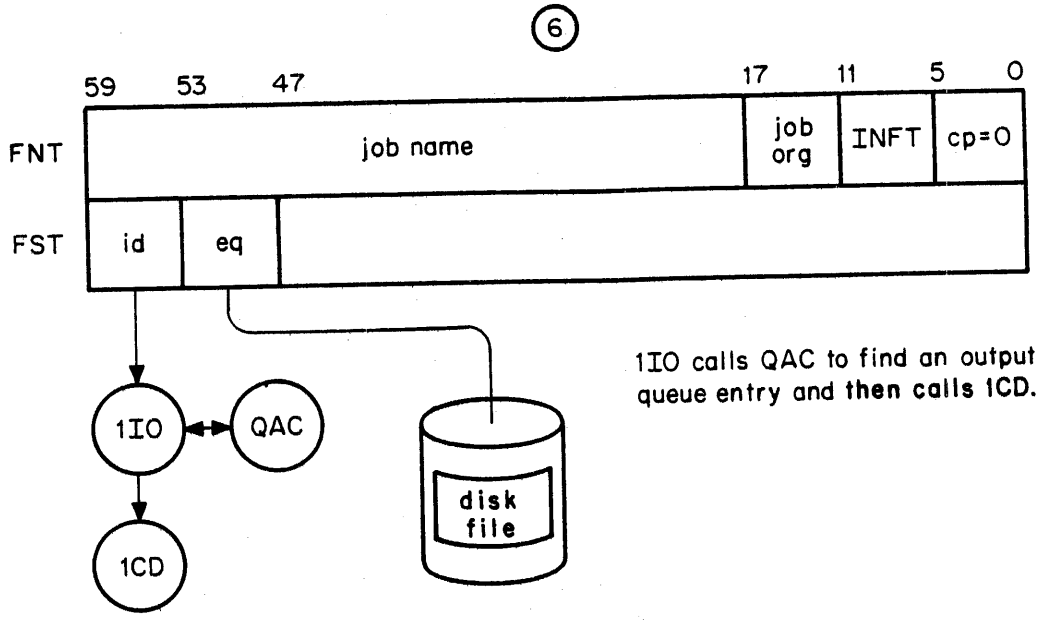
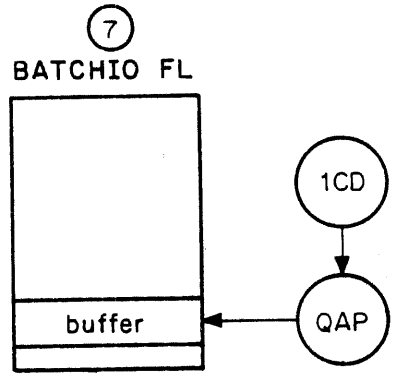
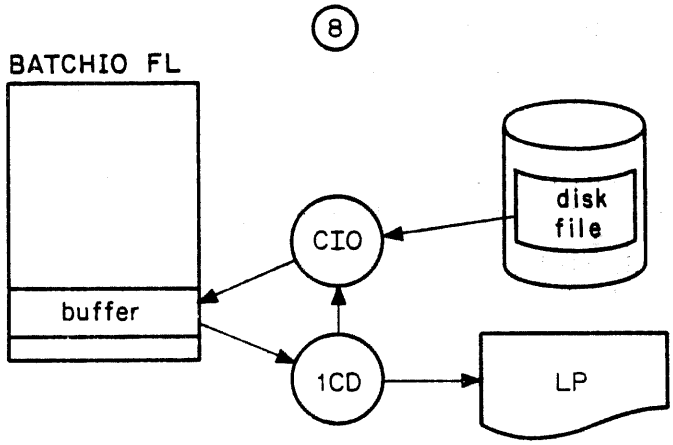


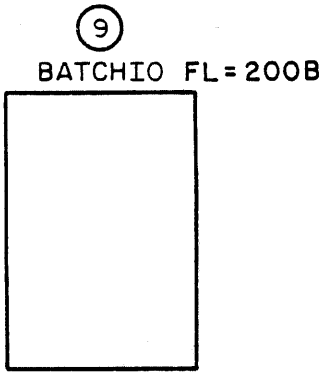
Figure 17-1. BATCHIO Overview (Continued)



1CD calls QAP to create a banner page (QAP calls OBP) in the CM buffer.



1CD reads the rest of the file to the CM buffer via CIO and prints the data on the printer.



1CD completes and drops; BATCHIO is now idle.

Figure 17-1. BATCHIO Overview (Continued)

BATCHIO CONTROL POINT

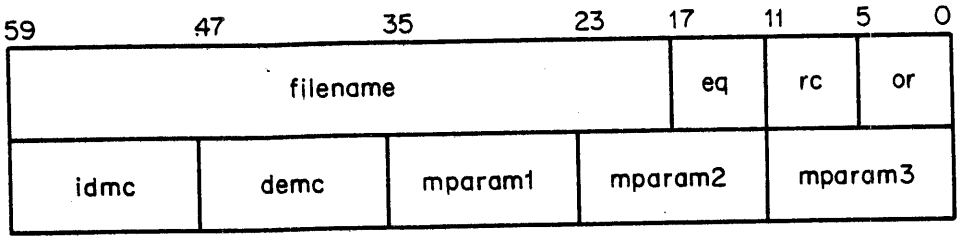
The BATCHIO control point field length contains no executable code; it is used entirely for communication, conversion tables, FETs, and buffers. The field length is expanded and contracted as devices need servicing or complete their operations. Each active device is assigned one FET and one buffer. A FET of 14 words is used and the buffer size ranges from 420B words for punch files up to 2020B words for 12-bit ASCII code print files.

BATCHIO COMMUNICATION

Much of the intercommunication among 1IO, 1CD, and QAP occurs in two areas of central memory:

- The BATCHIO control point area
- The BATCHIO field length

In the BATCHIO control point area, the locations normally used for the control statement buffer (CSBW) are not needed for that purpose by BATCHIO. Hence, these locations are used to store buffer information (BFCW=CSBW) and are referred to as the buffer point area. Each line printer, card punch, and card reader use two locations in the buffer point area in the following format.



filename	Logical file name
eq	Equipment number (EST ordinal)
rc	One of the following: • Repeat count, if no operator request • Request parameter, if operator request
or	Operator request: 0 None 1 End 2 Repeat 3 Suppress

4 Rerun

5 Hold

6 Continue

7 BKSP PRU

10 BKSP record

11 BKSP file

12 Skip PRU

13 Skip record

14 Skip file

idmc I display message code

demc Dayfile/error log message code

mparam1 Message parameter:

- Repeat count (LP and CP only)
- End count (LP and CP only)
- Channel number

mparam2 Message parameter:

- Driver address
- Printer error count (LP only)

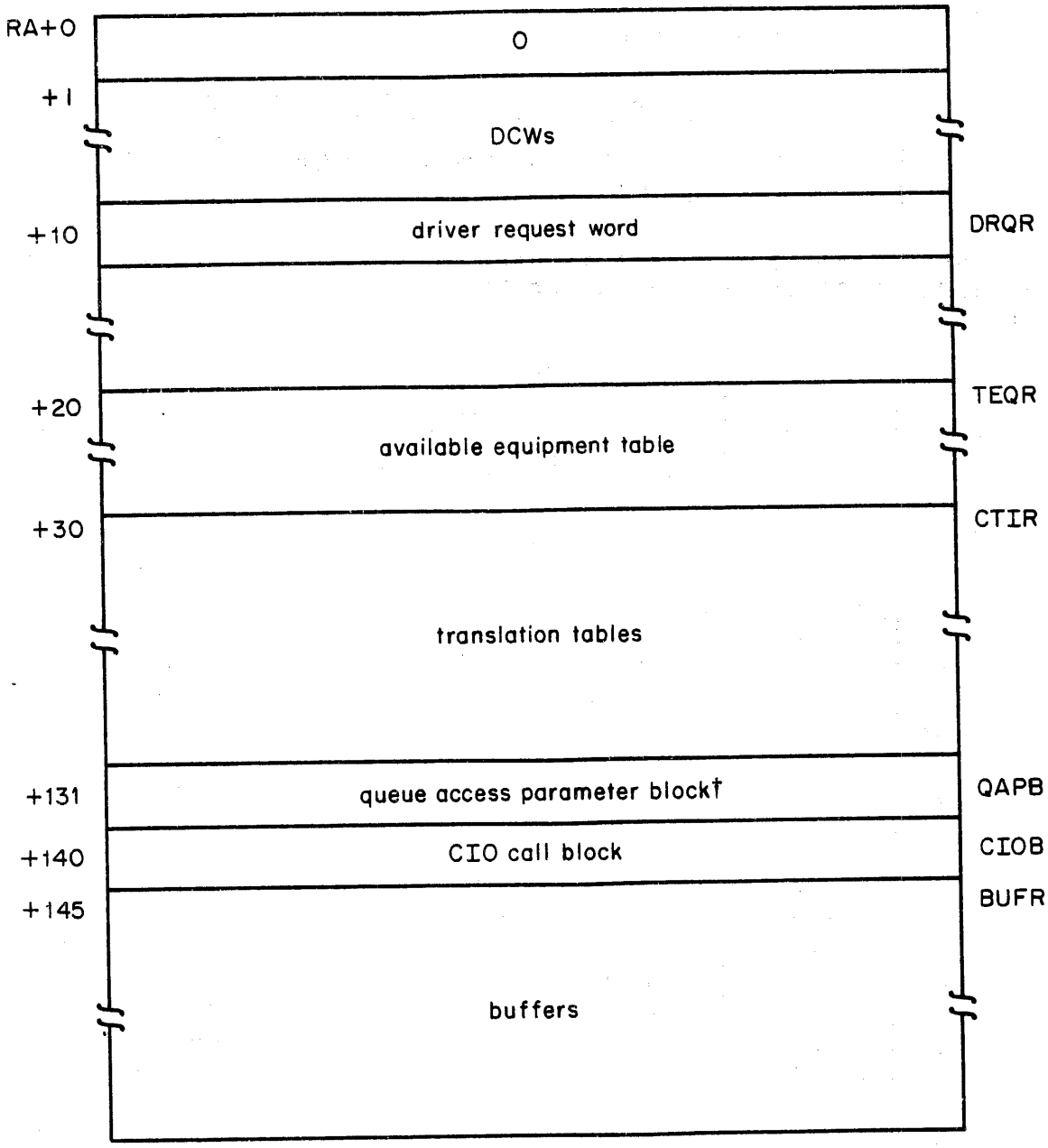
mparam3 One of the following

- Function code
- Message parameter:
  - Function code
  - Bytes remaining after an incomplete data transfer

The buffer point area is referenced by the logical equipment number of the device from the I display when communicating with DSD.

The BATCHIO field length also provides an area for communication. Each active device has a corresponding FET and buffer in the BATCHIO field length as follows.

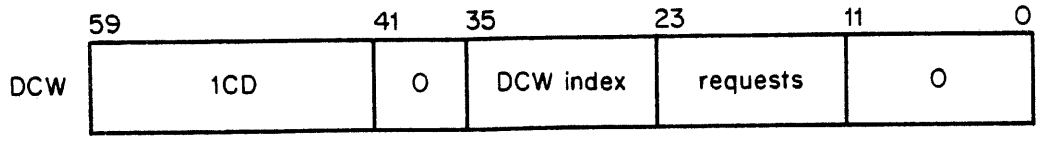
The core layout of the BATCHIO control point is shown in Figure 17-2



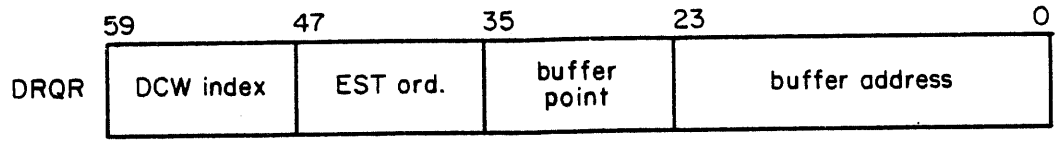
† Built by 1IO for QAC call to locate an unavailable qualifying file.

Figure 17-2. BATCHIO Central Memory Layout

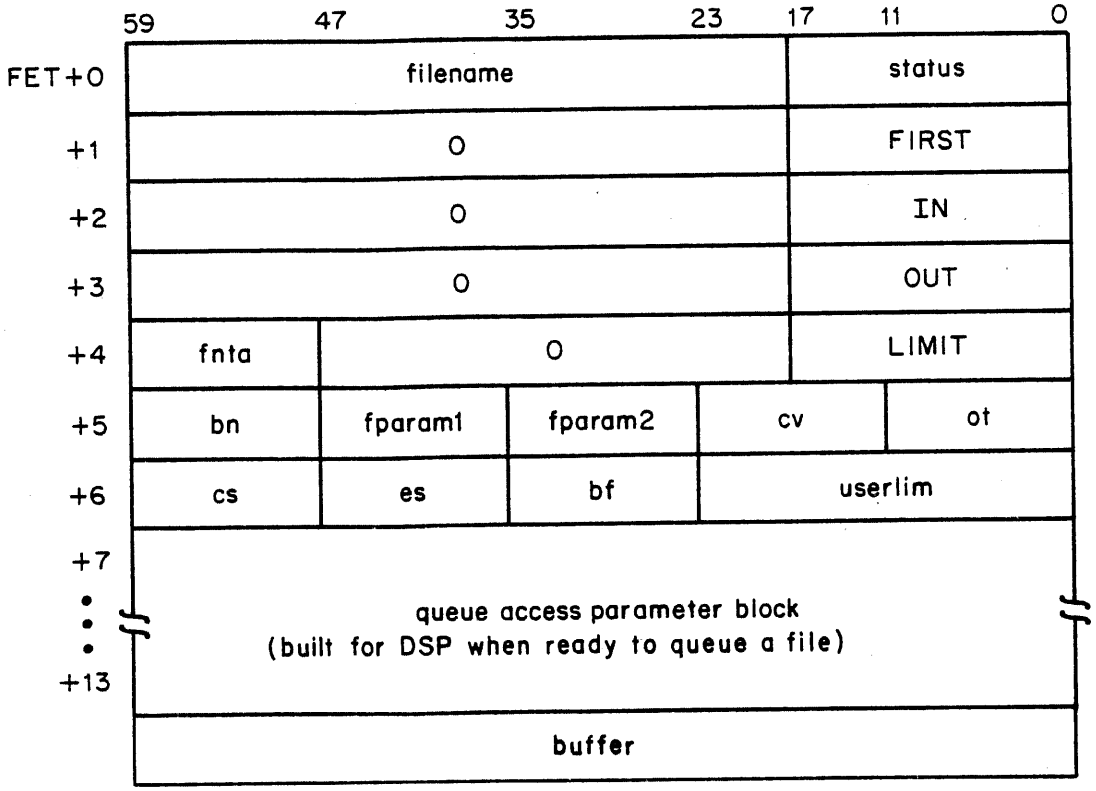
A driver control word (DCW) is used by 1IO to determine how many copies of 1CD are active and how many requests (equipments) each one is currently processing. The DCW (one for each copy of 1CD that is active) are stored in reverse order beginning at RA+7 continuing through RA+2. Each DCW can support up to MEQD (10B) equipments. The DCW is reference by a DCW index which is the DCW position relative to RA; that is, the DCW at RA+7 has index 7. The DCW has the following format.



Word DRQR (driver request) is used to communicate between 1CD and 1IO. DRQR is located at RA+10B and has the following format.



Buffers



filename	Logical file name
status	File status
FIRST	Address of first word of buffer (relative to RA)
IN	Next available address for entering data in buffer (relative to RA)
OUT	Next available address for removing data from buffer (relative to RA)
fnta	FNT address
LIMIT	Last word address plus 1 of buffer (first word address of next FET and buffer, relative to RA)
bn	Buffer point number
fparam1	File parameter: • First track of dayfile (LP only) • First card number of read sequence error (CR only)
fparam2	File parameter: • First sector of dayfile (LP only) • Punch format (CP only) • Record number of read sequence error (CR only)
cv	Count of V characters used (LP/PFC only)
ot	Origin type
cs	Converter (6681) status
es	Equipment status
bf	BATCHIO flags
userlim	Maximum number of possible lines printed or cards punched (LP and CP only)

Each buffer immediately follows its corresponding FET; that is, the first word address of the buffer is the last word plus 1 of the FET. Also, the next FET and buffer immediately follow the current buffer; that is, the first word address of the next FET is RA+LIMIT of the current FET.

Routine 1IO is a transient PP routine. It recalls itself by always copying its input register (IR) into the PP recall word (RLPW) before dropping or relinquishing the PP to another PP routine.

If there are no outstanding requests, 1IO reads and scans the available equipment table for ready devices (subroutine REQ). When the scan has been completed, 1IO recalls itself, if no equipment has been processed.

If a card reader in ready status is found, a buffer is assigned for the equipment and a DRQR request for 1CD is initiated in BATCHIO's field length to process the card reader. If 1CD is active and is not already processing MEQD equipments, the active request count in the DCW is advanced by one. Otherwise, a DCW is set up to start up a new copy of 1CD (subroutines 3IA/ABF and 3IA/ADR).

If an unassigned card punch or line printer is in ready status and is ON, a QAC call is made to identify a queue file having the same external characteristics as the printer or punch. Routine 1IO recalls itself with the file requested flag set in the RLPW. QAC is loaded into this PP (subroutine SFF).

When recalled, 1IO detects the file requested bit and checks to see if QAC has completed and indicated a queue file to be disposed to the printer or punch (subroutine CFF). If a file is found, a buffer is assigned for the equipment and DRQR request initiated as is done for the card reader (subroutines 3IA/ABF and 3IA/ADR). If QAC has not completed, 1IO recalls itself. If QAC completes but no file is found or an error occurs, the scan of available equipments starts again.

Routine 1CD responds to requests that appear in DRQR and in the buffer point words. Routine 1CD contains the drivers for all unit record equipments. If a card reader request is detected, 1CD reads one full buffer of cards (1020B words) or until EOR/EOF is detected and calls QAP to process the input file subroutine IIF). Routine QAP calls overlay OVJ to process the job and user cards, which must be the first two cards in the job deck. An FNT entry is made using the job name returned by OVJ and mass storage is assigned using MSAL if an input equipment was specified. CIO is then called to begin writing the file. Routine 1CD continues to read cards and transfers their images to the mass storage file (input file) via CIO. When the file has been completely read, QAP is called again to perform accounting of the cards read and to enter the file into the input queue via a DSP call (QAP function ACTF, subroutine ACT).

If the request was for a printer or punch 1CD calls QAP to build the banner page (function GBPF, subroutine LPR) or lace card (function GLCF, subroutine LPH) and places it into the BATCHIO buffer. Routine 1CD then transfers data from the file into the BATCHIO buffer using CIO and prints or punches the file in the proper format. When the file has been completely printed or punched, QAP is called again to perform accounting of the lines printed or cards

punched and calls DSP to decrement the files repeat count if any (QAP function ACTF, subroutine ACT).

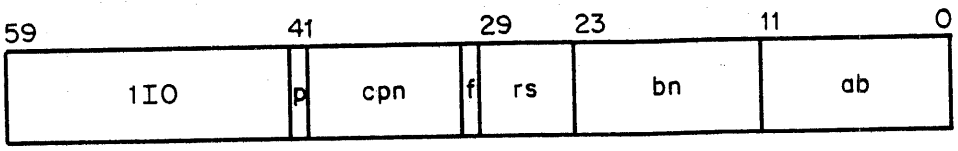
After the equipment has completed its operation, 1IO releases the BATCHIO buffer and may return the storage it required.

BATCHIO MANAGER - 1IO

Routine 1IO is the executive routine for the BATCHIO subsystem and performs scheduling of all processes operating at the BATCHIO control point. These include the following.

- Searching for the highest priority OUTPUT and PUNCH files (via QAC)
- Checking for a ready status on any card readers, line printers, or card punches
- Managing of buffer storage and allocating and deallocating central memory for the BATCHIO control point
- Posting of error condition messages for any of the above

The 1IO call format is as follows.



p	Preset performed (bit 41=1 once preset is performed)
cpn	Control point number
f	File previously requested flag (bit 30)
rs	Release storage repeat count
bn	Buffer point number currently under consideration
ab	Active buffer count

As 1IO operates, it stores values in cells IR+2, 3, and 4, and when recalled IR+2, 3, and 4 are reset. On recall these cells contain the following.

IR+2	File previously requested and storage release repeat count
IR+3	Buffer point number
IR+4	Number of buffers allocated (number of requests currently performing)

When the operator command n.IO, AUTO., or MAINTENANCE. is sensed, DSD calls 1DS to initiate the BATCHIO subsystem (see subsystem initiation in Section 5). Routine 1IO is called and checks the

p bit in the input register and if not set, calls the preset overlay 3ID.

Routine 1IO first checks DRQR. If DRQR is nonzero (that is, some copy of 1CD has not yet responded to the last request), 1IO processes error messages and checks central memory allocation. Eventually, DRQR becomes zero. The frequency of requests, versus the speed of unit record equipment, versus the speed of 1CD responding, does not necessitate more than a one-word request stack. The time lost while 1IO waits for DRQR to clear is negligible.

If DRQR is zero, 1IO checks the input register for pending file requests. If a file request is pending, 1IO checks to see if a file has been identified for disposal. If none is found, the file requested flag is cleared and 1IO proceeds with the TAEQ scan. If a file is found, 1IO requests a buffer for the file and requests 1CD. Line printer image memory is loaded by 1IO if the equipment is a printer. If the file search done by QAC is not complete, 1IO recalls itself.

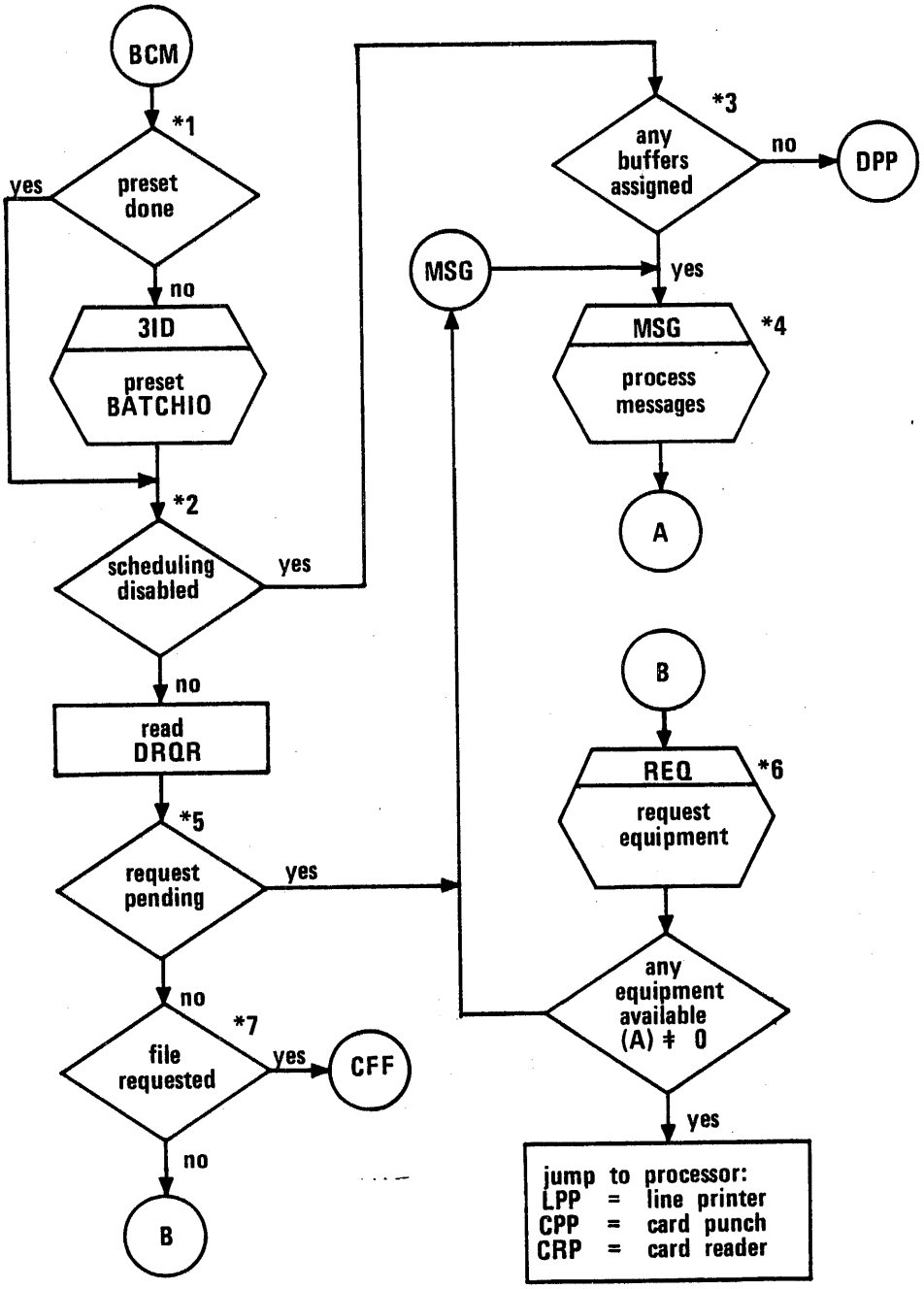
Routine 1IO scans the TAEQ tables for ready and unassigned unit record equipment when no other requests are pending. If an equipment is found, it is processed according to its type. For card readers, 1IO assigns a buffer and calls 1CD through DRQR. For line printers and card punches, 1IO formats a QAC call to search the queue for the highest priority file having the external characteristics, forms code, and ID of the available equipment. The file requested flag is set to indicate the QAC call is being made and 1IO recalls itself relinquishing its PP to QAC.

If a 1CD request needs to be made, 1IO checks the DCW words for an active 1CD. If one is found and byte 3 (number of current requests active) is less than MEQD, 1IO sets up the request in DRQR. Up to six copies of 1CD could be active at one time, depending on MXEQ, which is an assembled constant in COMSBIO and states the maximum number of equipments that can be active at once. Currently MXEQ is 24B, so the maximum 1CD copies is three. MXEQ is determined by the size of the control statement buffer. The buffer size is 50B words, and with two words per buffer point, this translates to a maximum of 24B equipments.

If there are no copies of 1CD currently active, or the copies which are active have the maximum current requests MEQD, and this request brings the total current request to less than MXEQ, then 1IO sets up the next DCW word, sets up the DRQR word, recalls itself, and calls 1CD into this PP.

When 1IO is recalled, it checks if scheduling has been disabled. If this disable bit has been set (bit 13 of INWL in CMR), 1IO does not schedule any new 1CD requests. It waits until all pending requests are complete, processes any error messages as they occur, releases buffers and all of central memory assigned to the control point, releases the control point, and then drops from the PP. If the disable bit is not set, 1IO continues performing as above.

Figure 17-3 is a flowchart of the main loop routine 1IO.



*1 IR+1 = display code for 1IO, P, CP number.
*2 CMR cell INWL (bit 13)
*3 Is IR+4 nonzero?
*4 Messages are IDLE and xx BUFFERS ACTIVE.
*5 Check RA+DRQR.
*6 RA+TEQR read EST and check each equipment for status in TEQR table; if some equipment needs to be processed, then (A) nonzero, (EQ)=equipment number, (ES-ES+4)=EST entry, and (IR+3)=equipment index.
*7 IR+2, bit 6 set if file previously requested.

Figure 17-3. 1IO - BATCHIO Main Loop

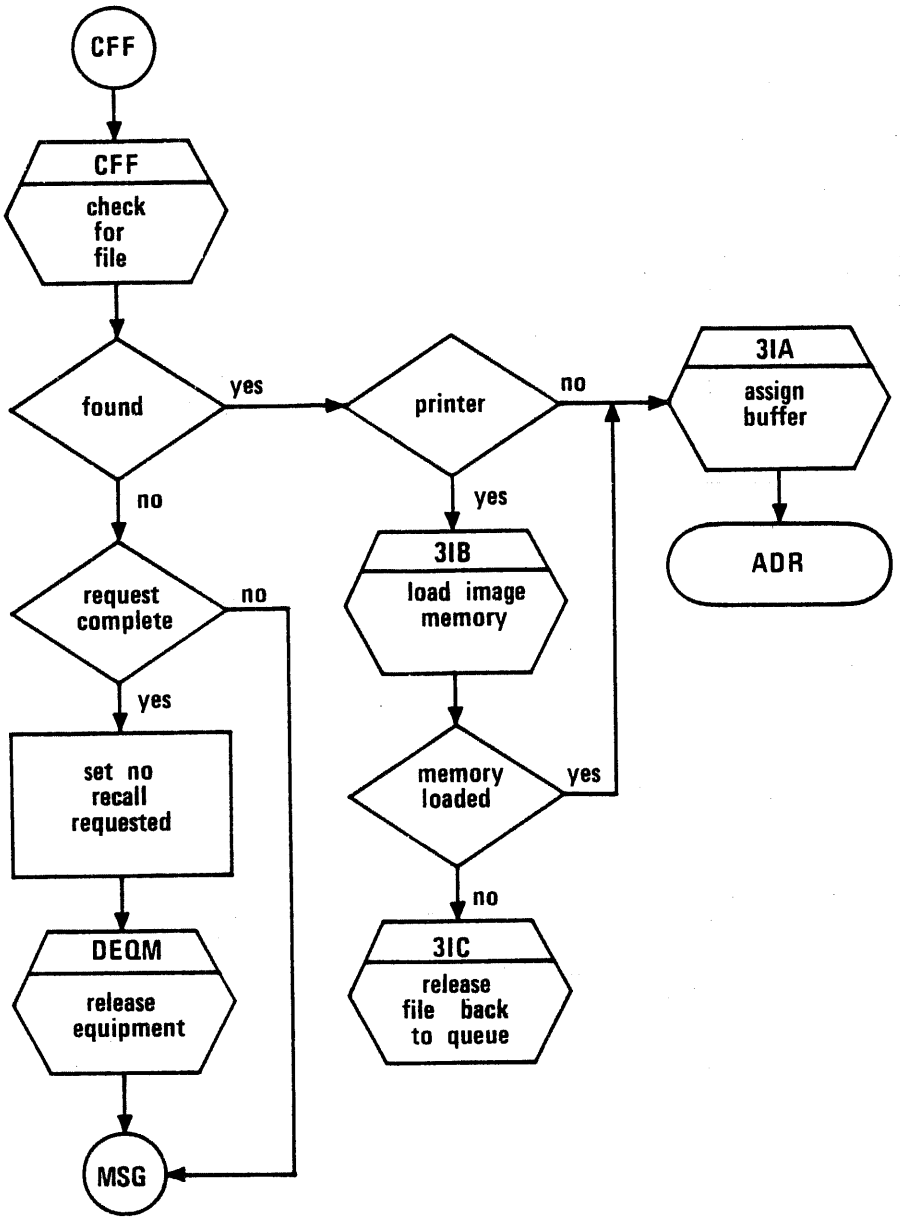


Figure 17-3. 1I0 - BATCHIO Main Loop (Continued)

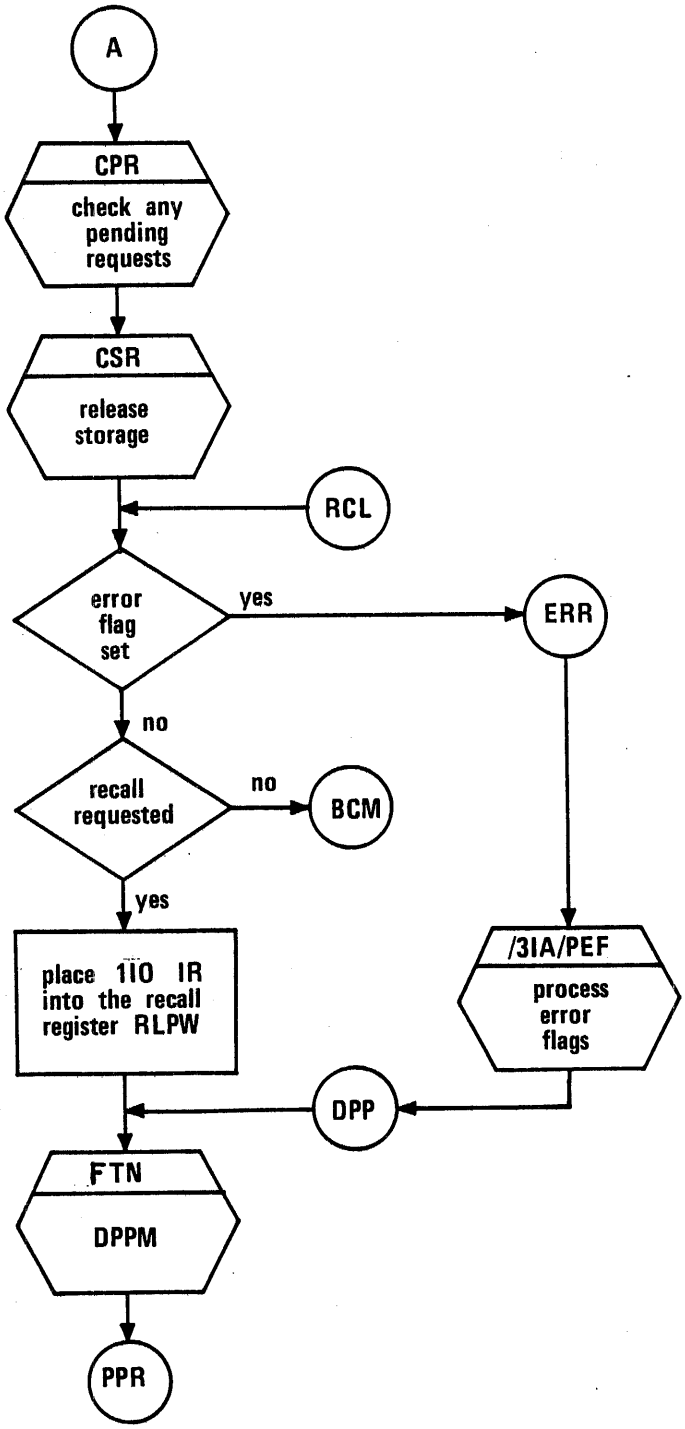


Figure 17-3. 1IO - BATCHIO Main Loop (Continued)

The following paragraphs describe 1I0 subroutines and overlays.

CFF - CHECK FOR FILE

CFF reads the buffer point table (TAEQ) and the EST entry for the buffer point in question (from IR+3). CFF then reads the QAC parameter block from BATCHIO's field length. This block is found at location QAPB (RA+131B) and is seven words in length. If the request has completed, (bit 0 of QAPB is 1), the file requested flag is cleared from the input register (bit 30; IR+2, bit 6). If the request completes with errors, the error code is retrieved from bits 17 through 12 of QAPB and is returned to the caller. If the request completes successfully, the file's FST is read and CFF returns to its caller. If the request is not complete, this status is returned to the caller.

CPR - CHECK PENDING REQUEST

CPR reads DRQR and if a request is present, the control word for the driver is read. If 1CD is not active, then 3IA is loaded and the driver is assigned.

CSR - CHECK FOR STORAGE RELEASE

CSR releases BATCHIO field length not needed due to a buffer point completing its activity. Field length management is only done every five (5) calls of CSR. The number of CSR calls is kept in the lower 6 bits of IR+2. Storage space is released from the last buffer in use to the current FL; that is, if there are any free buffers beyond the last buffer in use, their space is returned. The new buffer count is set into IR+4, which includes all active and inactive buffers up to the last buffer in use.

MSG - PROCESS CONTROL POINT MESSAGE

MSG issues messages to MS1W of the control point area. Two messages are issued: IDLE if no buffers are active; or nn BUFFERS ACTIVE where nn is the number of buffers defined. Some of the nn buffers may not be in use, but cannot be released since a buffer beyond them is still active.

REQ - REQUEST EQUIPMENT

REQ scans the buffer point table (TAEQ) checking the status of the unassigned unit record equipment listed. If the equipment is off, the OFF message pointer is set for display on the I display. If on, the equipment is requested (REQM) and statused. If it is not ready, it is released (DEQM) and the NOT READY message pointer is set for the I display. The starting position of the TAEQ search is saved in IR+3 so that the search moves circularly through TAEQ as ready equipment are processed.

SFF - SEARCH FOR FILE

SFF formats a call to PP routine QAC to search the FNT for a queue file having a given set of characteristics. The QAC call block is written into BATCHIO's field length at location QAPB, 1I0 is put into recall by writing the input register into RLPW with the file requested flag set (IR+2, bit 6), and the QAC call is written into this PP's input register. QAC is then loaded through a call to PPR.

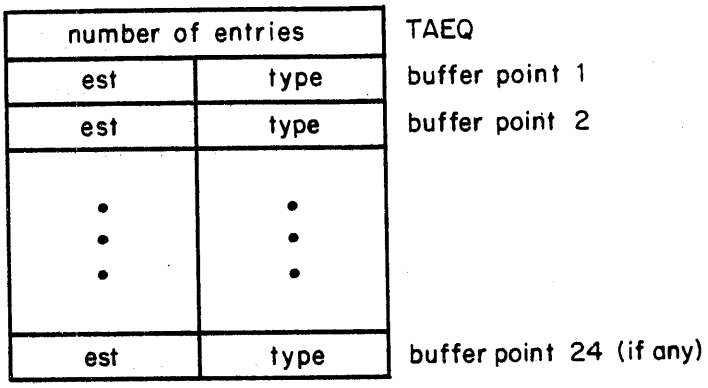
3ID - 1I0 PRESET BATCHIO

Overlay 3ID writes the jobname BATCHIO into control point area word JNMW and sets the CPU priority to 1 and queue priority to BIPS via the RPRM monitor function. The initial field length of 200B words is requested using the RSTM function.

The EST is scanned for valid unit record equipment, namely card readers (CR), card punches (CP), and line printers (LP, LR, LS, and LT). If a valid unit record peripheral is found, a buffer point entry is made in BFCW and in the TAEQ table. If no equipment is found, the message NO EQUIPMENT AVAILABLE is issued and BATCHIO terminates. If more than 24B peripherals (defined by symbol MXEQ) exist, the diagnostic NOT ALL EQUIPMENT SERVICEABLE is issued and only the first 24 equipments are used. The input register has bit 43 set to indicate preset has been performed.

The 64 character set is used as the default character set. If 63 character set is used (IPRDECK parameter CSM=63), the 63 character set conversion is entered in conversion tables COMT6DP, COMT9DP, COMTDA8, COMTDP6, and COMTDP9 such that a colon is changed from display code 00 to 63, 00 becomes a space, and percent is dropped. The conversion tables are then written into BATCHIO's field length beginning at word CTIR and 3ID returns to the main program.

The TAEQ (table of available equipment) has a one to one correspondence with the buffer point table entries and has the following format.



type 0 Printer (LP, LR, LS, LT)
      1 Punch (CP)
      2 Printer (CR)

est EST ordinal of equipment

3IA - 1IO AUXILIARY SUBROUTINES

Overlay 3IA contains various subroutines used by 1TO to perform buffer point management.

ABF - Assign Buffer

ABF assigns a buffer to a particular equipment. Subroutines EFT, EFP, EBP, and FFB are called to accomplish the assignment.

ADR - Assign Driver

ADR calls the combined driver 1CD into an available PP or this PP if none is available as necessary. If 1CD is already active and is not full (driving less than 8 equipments), the equipment being assigned is added to the current DRQR requests.

ANB - Add New Buffer

ANB sets the FIRST and LIMIT values for a newly established buffer.

EBP - Enter Buffer Point Information

EBP sets the initial repeat count, job name, and equipment number in the buffer point word for display by DSD on the I display.

EFP - Enter File Parameters

For output files, EFP extracts parameters such as user limits, dayfile address (if any), and other file characteristics from the QAC parameter block. The control values contained in FET+5 and FET+6 are initialized at this time; these values are the buffer number, dayfile track (if any), punch format, job origin, and limit values.

EFT - Enter FET Information

EFT initializes the buffer FET. FET+0 receives the file name (two blanks if input file) and completion status, and IN and OUT are set equal to FIRST. The address of the FNT is set in the LIMIT word.

FFB - Find Free Buffer

FFB scans the buffer points for an available CM buffer of the size required for the equipment making the request. If a free buffer of the correct size is found, it is used. Otherwise, a storage increase of the CM buffer size required is made. If the storage is made available, the new CM buffer is added (by ANB) at the end of the current CM buffers.

The CM buffer space required is 1020B to 2020B words for a line printer, 1020B for a card reader, and 420B words for a card punch.

3IB - LOAD IMAGE MEMORY

Routine 3IB contains the code required to load print train image memory for defined line printers.

The print train image memories supported in NOS are the 596-1, 596-4, 596-5, and 596-6. The image memories are contained in overlays named 5Ix where x is a character that defines the type of train. The data in the image memory overlay is specified by a macro that defines the actual print slug. Macros SLUG and ESLUG are defined in 3IB and specify the number of characters on each slug and the characters themselves. The following image memories are defined.

Image Memory	Print Train
5IE	596-1
5IG	596-4, 596-5
5IH	596-6

The print train image memories apply to all 580 printers.

Extended array mode is set at the time image memory is loaded. Therefore, all data sent to the line printer is interpretted by the controller as one 8-bit character per byte.

3IC - ERROR PROCESSOR

Overlay 3IC contains subroutines to process hardware status errors and to requeue files attached to BATCHIO.

BATCHIO COMBINED DRIVER - 1CD

The BATCHIO driver, 1CD, can drive up to eight devices of three types (any combination). These three types of devices are:

580	Printer
3446/415	Card Punch
3447/405	Card Reader

Some functions, such as accounting, are performed by calling QAP. Mass storage transfers are performed by CIO.

Each of the drivers use the conversion tables that reside in the BATCHIO field length. The conversion tables required by the devices are loaded into 1CD from CTIR as needed.

PRINTER DRIVER CHARACTERISTICS

Line spacing is normally done in the AUTO EJECT mode. That is, creases in the paper are skipped via the 3555 or 580 automatic line spacing. Thus, it is necessary for AUTO EJECT to be deselected to use format channels to advance from prior to bottom of form to beyond top of form. An example of this is the typical NOS format tape which has only one hole in channel 6, thus providing an eject of up to two pages in order to ensure all banner pages are printed correctly. Deselection of auto eject mode on a 580 results in deselection of eight lines/inch if previously selected.

The first character of the print line controls the optional formats. The print line, therefore, consists of up to 136 characters. The first character is recognized as a format control character and is never printed. If the first character is not a valid format control character, it is ignored and treated as if it were a blank.

The format control characters, their functions, and the number of lines charged are listed in table 17-1.

Any format control other than Q, R, S, and T are processed once for the line printed.

TABLE 17-1. FORMAT CONTROL CHARACTERS

Character	Function	Lines Charged
C	Skip to format channel 6 after print	4
D	Skip to format channel 5 after print	3
E	Skip to format channel 4 after print	3
F	Skip to format channel 3 after print	2
G	Skip to format channel 2 after print	2
H	Skip to format channel 1 after print	1
Q*	Suppress auto eject	0
R*	Set auto eject	0
S*	Clear 8 lines/inch	0
T*	Set 8 lines/inch	0
V	Eject page and change PFC image	PL6L/ PL8L
0	Space 1 line before print	2
1	Eject page before print	PL6L or PL8L
2	Advance to last line of form before print	PL6L/2
3	Skip to format channel 6 before print	4
4	Skip to format channel 5 before print	3
5	Skip to format channel 4 before print	3
6	Skip to format channel 3 before print	2
7	Skip to format channel 2 before print	2
8	Skip to format channel 1 before print	1
+	Suppress space before print	1
-	Space 2 lines before print	3
/	Suppress space after print	1

*Q, R, S, and T ignore the remainder of the line.

TABLE 17-1. FORMAT CONTROL CHARACTERS (CONTINUED)

Character	Function	Lines Charged
Space	No line control	1
Other	No line control - character printed	1

CARD PUNCH DRIVER CHARACTERISTICS

Hollerith cards are punched from a line consisting of up to 90 characters. However, only the first 80 characters of the line are actually punched. The display code to 026/029 conversion is accomplished by a display code to binary column image conversion in the driver. These column images are then punched in binary mode.

Binary data are punched in the following format.

Column	Description
1	Word count and binary card indicator (79)
2	Binary data checksum modulo 4096
3 - 77	15 central memory words of data
78	Blank
79 - 80	24-bit binary card sequence number

Absolute binary data are punched 16 central memory words per card with no special punches.

End-of-record cards contain a 7/8/9 multi-punch in columns 1 and 80, and the remainder of the card is blank.

End-of-file cards contain a 6/7/9 punch in columns 1 and 80, and the remainder of the card is blank.

End-of-information cards contain a 6/7/8/9 multi-punch in columns 1 and 80, and the remainder of the card is blank. Cards offset are as follows.

- The end-of-information card
- All end-of-record cards
- A card on which a compare error is detected and the following card (these two cards are repunched until no error is detected)

CARD READER DRIVER CHARACTERISTICS

All cards are read in binary mode. Coded cards are converted to display code by the driver. Up to 80 characters may be transferred to the CM buffer. The mode of the coded conversion can be changed as follows.

Card	Mode Change Indicator
Job	26 or 29 punched in columns 79 and 80
7/8/9	26 or 29 punched in columns 79 and 80
6/7/9	26 or 29 punched in columns 79 and 80
5/7/9	No punch in column 2 indicates 026 mode; 9 punch in column 2 indicates 029 mode.

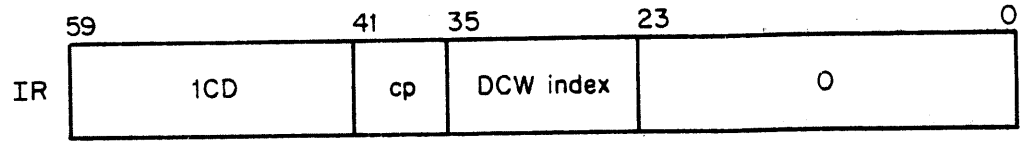
A mode change is in effect until changed. Default keypunch mode for a job is defined as an installation parameter (KEYPM).

For the 5/7/9 card, the following are valid conversion mode punches in column 2:

Punch	Description
Blank	026
9	029
4/5/6/7/8/9	Literal input; cards are read in binary format with no conversion or checking until a card which is identical in all 80 columns is read

Binary cards must conform to the above specification for punched binary data. An end-of-record consists of a card with 7/8/9 multipunched in column 1. An end-of-file consists of a card with 6/7/9 multipunched in column 1. An end-of-information consists of a card with 6/7/8/9 multipunched in column 1. The 7/8/9 card and the 6/7/9 card signal input keypunch mode conversion. A 26 punched in columns 79 and 80 of either of these cards (or on a job card) indicates that the following cards are in 026 mode. A 29 in columns 79 and 80 indicate a change of 029 mode. Anything else in columns 79 and 80 do not affect the input mode and are ignored, as are columns 2 through 78.

The call to 1CD is as follows.



cp Control point number

1CD - BATCHIO PERIPHERAL DRIVER

Routine 1CD does not use any mass storage drivers. This allows it to use the area of PP resident reserved for the mass storage driver and high core reserved for driver recovery processing for translation tables and data. The translation tables are loaded when needed beginning at location MSFW.

The preset routine moves subroutines WNB, CEP, RBB, WBB, and the channel table TCHS to PP resident following the translation table area, sets the default keypunch mode from IPRL, and sets the control point number for CIO and QAP service calls.

After preset (and as long as 1CD is at this PP), the main loop (manager) checks DRQR for nonzero. If it is nonzero, the DRQR DCW offset is compared to the 1CD DCW index in IR+2. If they do not match or DRQR is zero, 1CD assesses the status of its active requests. As a request goes inactive, 1CD decrements the active request count in its appropriate DCW word. If the active request count goes to zero, 1CD cleans its respective DCW word and drops.

If the DRQR DCW offset is equal to this 1CD, 1CD starts up the proper driver and increments its active request count in its approrpiate DCW word. The manager is flowcharted in figure 17-4.

Routine 1CD can service up to eight devices by each device sequentially gaining control of 1CD to perform processing. The relative location buffer in 1CD associated with each device and the CM buffer in the BATCHIO FL (which includes a FET and a QAPB) associated with each device, preserve the required information for 1CD to process this device.

After each call to CIO or QAP, 1CD jumps to its main loop. The drivers can be processing many requests simultaneously and are honored in sequence by 1CD. The limit of MEQD (currently 10B) is all one copy of 1CD can process and still continue to drive each device at top speed.

The layout of 1CD is shown in figure 17-3.1.

A relative location buffer is reserved for all eight possible devices. The data buffers are each 141 PP words in length and are shared by all devices serviced by that copy of 1CD. The number (n in figure 17-3.1) of such data buffers is determined by the space available in 1CD (n is currently 4).

A data buffer is assigned to a particular device by subroutine ADB (assign data buffer) and released by subroutine RDB (release data buffer). These data buffers are reserved for a minimum period of time. A relative location buffer is associated with each device, while a data buffer is assigned (ADB) and released (RDB) by the reader, punch, or printer driver one or more times during the I/O process.

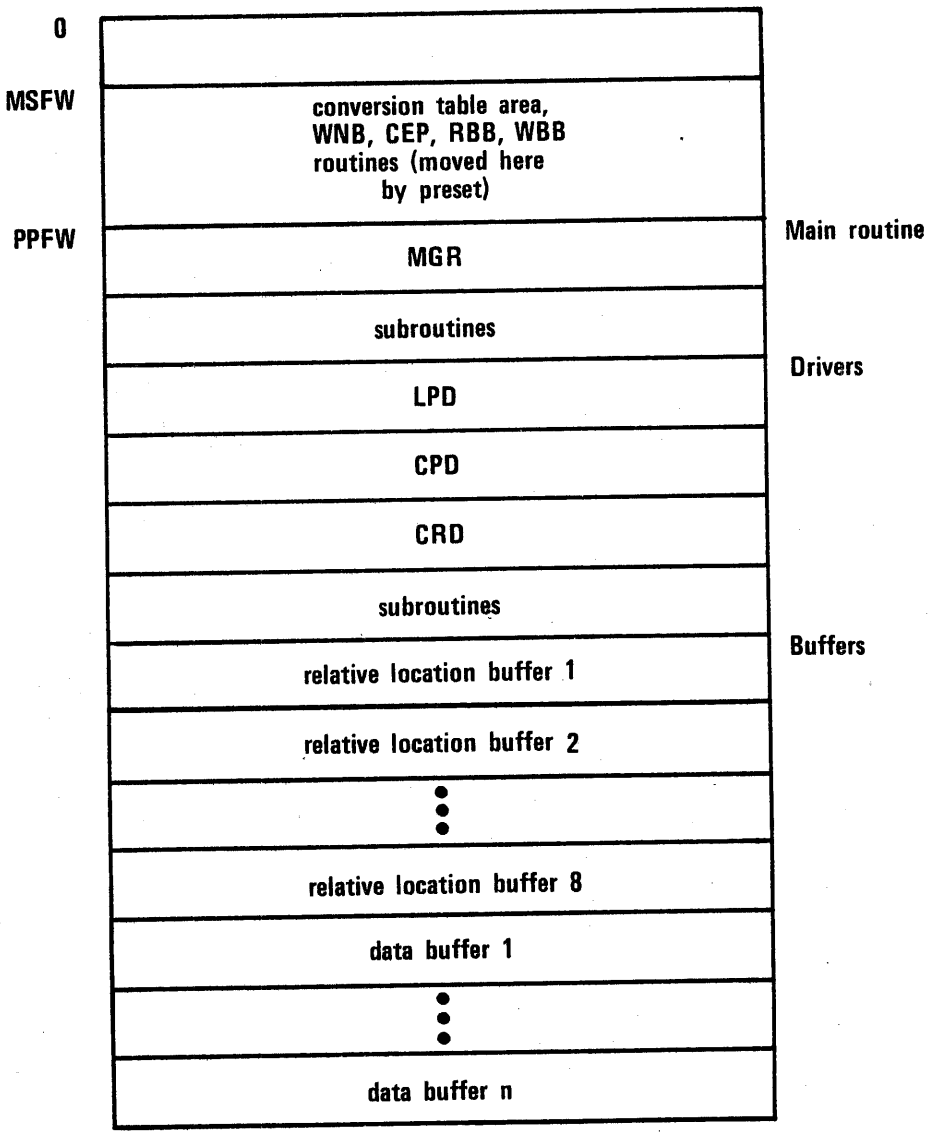


Figure 17-3.1. 1CD Layout

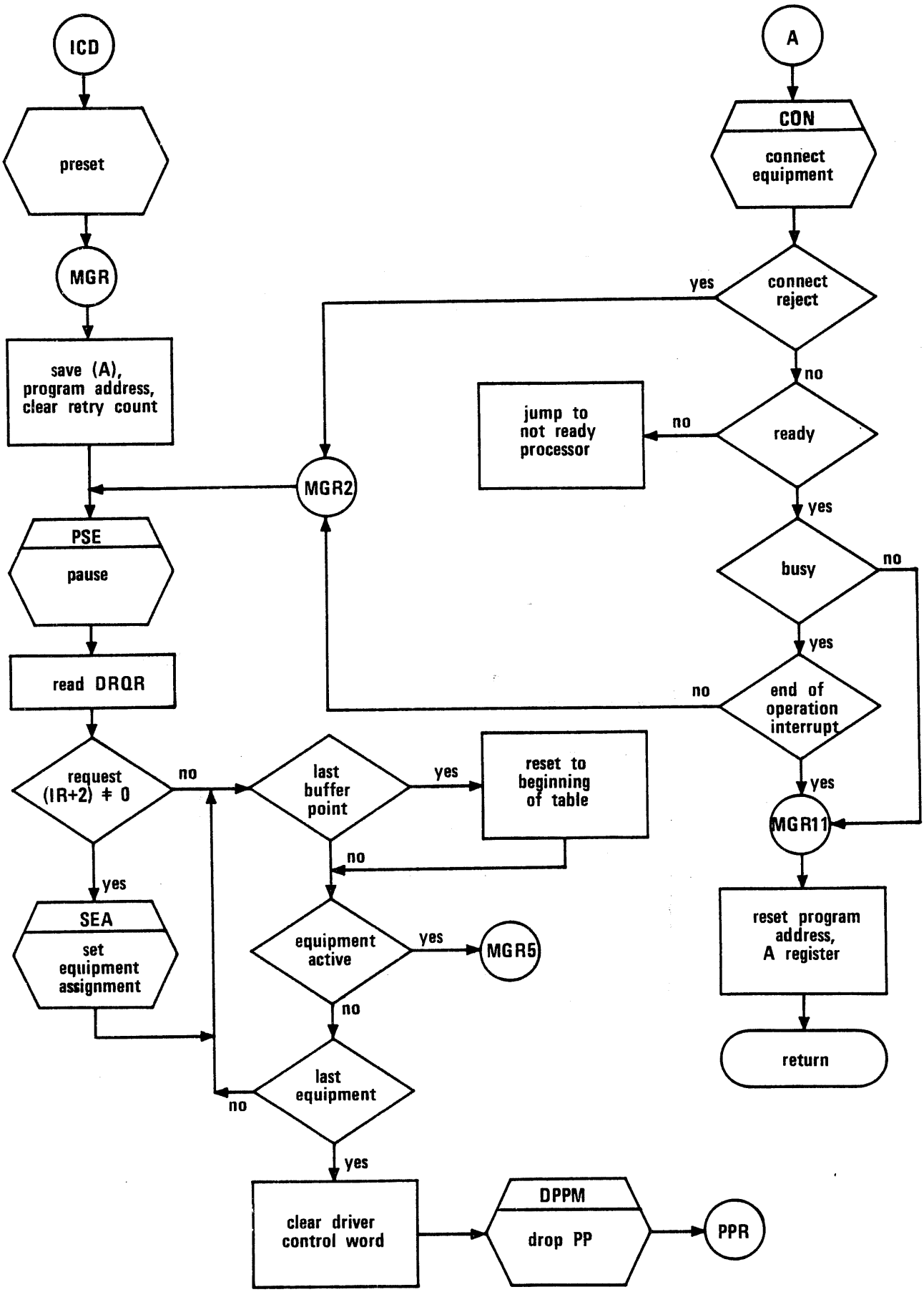


Figure 17-4. ICD Manager

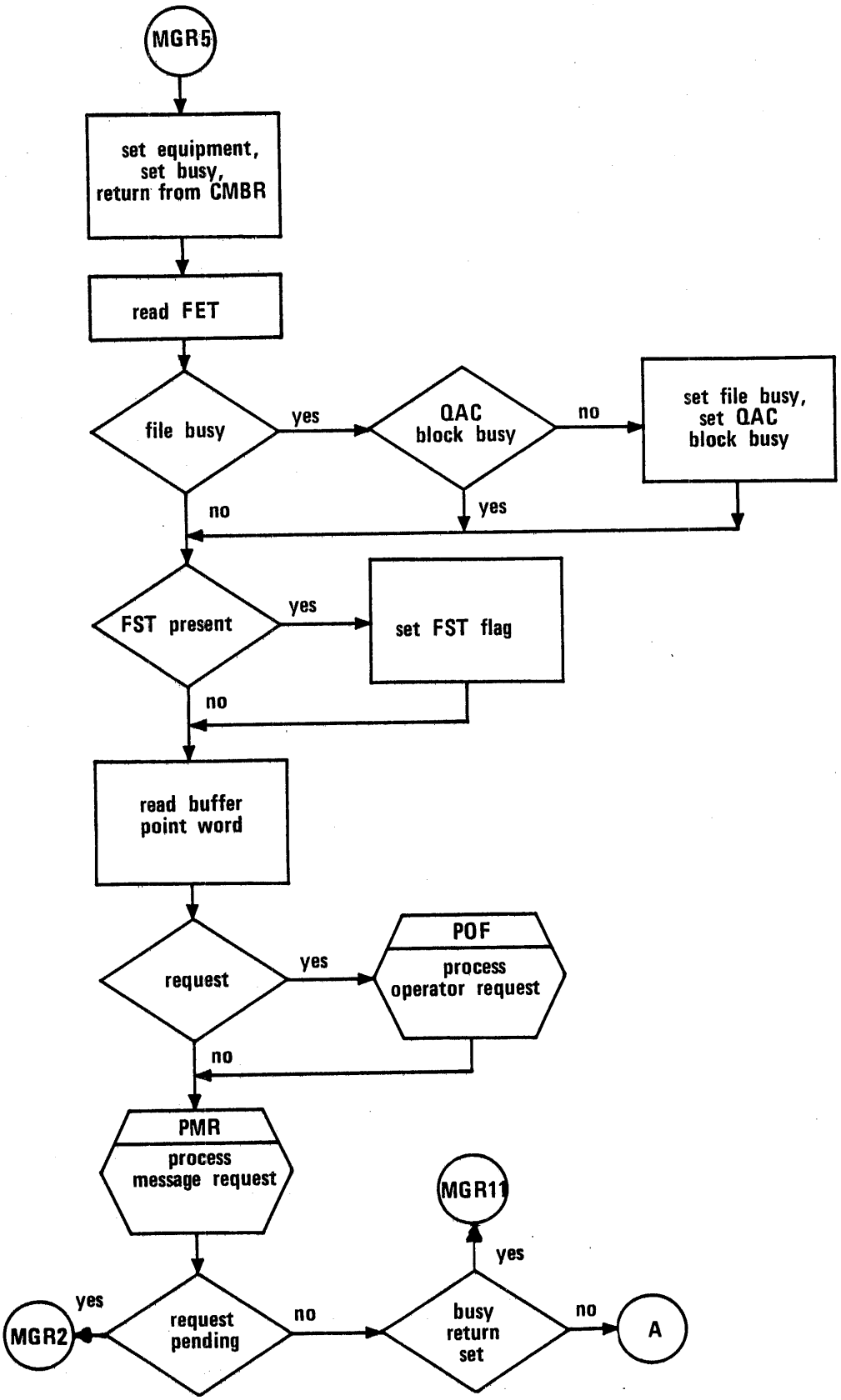


Figure 17-4. 1CD Manager (Continued)

DSD OPERATOR REQUEST

When DSD senses a BATCHIO operator request, it places the request in the buffer point word for the equipment through a call to 1DS. (If BATCHIO is not active, these commands are illegal.)

Routine MGR in 1CD checks the buffer point area words of the equipments it is currently processing. If a request is set and the buffer is not busy, it calls the processor for the request. If the buffer is busy, the request is ignored until the buffer becomes available.

The DSD buffer point commands are processed by 1CD as follows:

Command	Significance
END.	Set EOI status in buffer and set buffer empty. Print files with dayfiles present have their dayfile printed. If there are no repeats specified, QAP is called to perform the accounting for the operation.
REPEAT.	Advances repeat count by the number specified in the request. Repetition on card readers is ignored.
SUPPRESS.	Suppress toggles the suppress flag and is only processed for line printers. Suppression means that carriage control is ignored.
RERUN.	For line printers and card punches, a QAP call is issued (function RQFF) to requeue the file. For card readers, RERUN is ignored.
STOP.	The hold flag is set for line printer; it is ignored for card equipment. Stop suspends printing.
CONTINUE.	The stop flag is cleared and printing is allowed to resume.
BKSP. BKSPF. BKSPRU.	Backspaces print file the specified number of records (BKSP), files (BKSPF), or PRUS (BKSPRU). A QAP call is issued (function PORF) to position the print file.
SKIP. SKIPF. SKIPRU.	Skips forward on the print file a specified number of records (SKIP), files (SKIPF), or PRUs (SKIPRU). A QAP call is issued (function PORF) to position the print file.

A message is issued to the system and control point dayfile showing the parameters in the buffer point word. This is done by QAP (function PDMF).

The following paragraphs describe 1CD subroutines.

SEA - SET EQUIPMENT ASSIGNMENT

SEA is called on each DRQR request to initialize the parameters used in processing that request. SEA sets the equipment number, buffer point address, buffer address, program address, conversion mode, equipment type, channel, and connect code; initializes the buffer status; clears the message request, controller busy retry count, data transfer error retry count, function reject retry count, busy return, and end/repeat flags; updates the equipment count in the input register; and clears the driver request.

POF - PROCESS OPERATOR FLAG

POF interrogates the operator request in the buffer point word and jumps to the processor for the particular operator function.

LPD - LINE PRINTER DRIVER

LPD is the driver for the line printer. The driver passes through the following phases.

- Initialization
- Load buffer
- Process format control
- Print line
- Post print

In the initialization phase, various flags and counters are initialized and QAP is requested to format the banner page (function GBPF).

In the load buffer phase, if the hold flag is set, the busy return status is set and the manager is called. If there is no hold condition, the line to be printed is read from the buffer in central memory to the data buffer in 1CD. As the line is read into the data buffer in 1CD, it is converted to 12-bit ASCII code if the print file is not already 12-bit ASCII code. If the print file is already 12-bit ASCII code, no conversion is necessary. A call to ADB was made previously to assign a data buffer.

In the format control phase, if the supress flag is set or the format control character (first character of the line is blank), the line is printed. If the control character is nonblank, subroutine SPC (process space control) is called to perform the proper spacing and to charge for it. If the control character is Q, R, S, or T, no line is printed and phase 1 is reentered.

In the print line phase, the line is printed without printing the format control character. A call to RDB is made here to release the data buffer.

In the post print phase, the line count is advanced and a check is made to determine if the user limit for the number of lines allowable has been exceeded.

CPD - CARD PUNCH DRIVER

The card punch driver includes the following phases.

- Initialization
- Load buffer
- Punch card
- Check previous card

In the initialization phase, the proper 026 or 029 conversion mode is selected and QAP is called to format the lace card (function GLCF). The remaining phases are duplicated for binary or coded cards. A call to ADB is made here to assign a data buffer.

In the load buffer phase, the EOR, EOF, and EOI status of the buffer is detected to determine if the special ending cards need to be punched. The card buffer is filled from the buffer in central memory by binary or coded reads. If a checksum for the binary card is required, it is generated at this time.

In the punch card phase, the card is punched in binary format from the card buffer. The previous phase has prepared the data to be punched in binary mode (for the controller) so that the card punched has the proper system coded or binary specification.

In the check previous card phase, the punch status is interrogated to verify that the data has been punched correctly. If an error has occurred, a COMPARE ERROR message is posted and the card repunched with offset.

CRD - CARD READER DRIVER

The card reader driver has the following phases.

- Initialization
- Check mode of card
- Process card
- Dump buffer
- Store IN and update record count

In the initialization phase, record and card counts are initialized and default conversion mode is selected.

The next phase checks the mode of the card. A call to ADB is made here to assign a data buffer. All cards are read in binary mode. If cards are being read in literal input mode, processing is transferred to the literal input processor. If not literal mode, the first column of the card determines whether the processing continues with the binary or coded processors.

For coded card processing, the process card phase converts from binary to display code using the appropriate conversion table. Once the card has been converted, the dump buffer phase is entered and the data is copied to the central memory buffer.

For binary card processing, the process card phase checks for special format (EOR/EOF with conversion mode specified), or literal input. The checksum for the binary card is generated. If the checksum is correct, the dump buffer phase is entered and the data copied to the central memory buffer. If the checksum does not agree, the operator is directed to reread 3 cards. If the cards are out of sequence, status is recorded in the FST so that the job is aborted when initiated by 1AJ. An appropriate diagnostic is issued under these conditions.

For literal input, the data is taken as is and is sent to the central memory buffer. An EOI terminates literal input.

The final phase advances the IN pointer, advances the card count and record count (if needed), and returns to the check mode of card phase. A call to RDB is made to release the data buffer.

ACT - PROCESS ACCOUNTING INFORMATION

ACT requests that QAP format an accounting message for the unit record operation performed (function ACTF).

CIB - CHECK INPUT BUFFER

CIB determines whether data has filled the buffer for a card reader. If the buffer is being written for the first time, a QAP call is made to initiate the input file. Otherwise, a CIO call is made.

COB - CHECK OUTPUT BUFFER

COB determines whether there is data in the buffer available to be written to the line printer or card punch. COB transfers control to the EOR and EOF processors if that status was encountered. If the buffer is empty, it is filled by a CIO call (function 600 - READEI for line printers or function 10 - READ for card punches).

CPS - CALL PP SERVICE PROGRAM

CPS formats calls to QAP and CIO to request the services the PP routines provide for 1CD. The function is set in the first word of the buffer FET. A QAP or CIO call is then formatted for the buffer. A PP is requested via the RPPM monitor function. If no PP is assigned, the previous status is reentered into the FET. Otherwise, the FET remains active and the function is completed by QAP or CIO.

CUL - CHECK USER LIMIT REACHED

CUL compares the current lines printed or cards punched values against the user limit values that were passed in the FET when the file was initiated by QAP. These limit values are obtained from the file's system sector wherein they are stored when the file is disposed to the queue. A call is made to QAP (PLEF) to terminate the buffer with the LINE LIMIT EXCEEDED (for printers) or LIMIT (for punched) messages to notify the user that the limit has been reached.

PMR - PROCESS MESSAGE REQUEST

PMR makes a QAP call (function PDMF) to process a message associated with a particular driver operation.

RCB - READ CODED BUFFER

RCB translates coded data from the central memory buffer into the proper character representation for printing or punching.

TOF - TERMINATE OUTPUT FILE

TOF terminates active output. If there are no repeats specified, control is transferred to subroutine TOP. If repetition is specified, the repeat count is decremented in the bufferpoint word. Accounting for the printing or punching are performed prior to the call to TOF; hence, the file is rewound and driver initialization phase reentered if repeat counts are present.

TOP - TERMINATE OPERATION

TOP causes the file to be dropped by a CIO close unload function (170). The buffer point word is reset for the equipment, the first word of the buffer FET is cleared, the number of equipment being processed by this copy of 1CD decremented, and the equipment released.

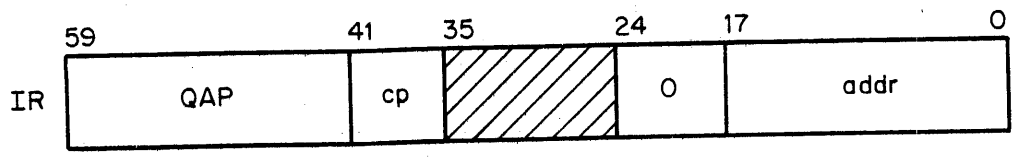
QAP - BATCHIO AUXILIARY PROCESSOR

Routine QAP is called by 1CD to process several functions. The function codes (defined in COMSBIO) and their processors are:

Symbol	Code	Processor	Description
PDMF	400	PDF	Process dayfile message
WTIF	410	IIF	Initiate input with write
WRIF	420	IIF	Initiate input with write EOR
WFIF	430	IIF	Initiate input with write EOF
CECF	440	CEC	Channel error cleanup
PORF	450	POR	Process operator request
RQFF	460	CEC	Requeue file
PFCF	470	LPR1	Load user image to 580 PFC
GBPF	500	LPR	Generate banner page
GLCF	510	LPH	Generate lace card
PLEF	520	PLE	Process user limit exceeded
ACTF	530	ACT	Process accounting

Processors IIF, LPR, PLE, and PDF are contained within QAP, while the majority of the other processors call QAP overlays LPH (3BC), ACT (3BA), POR (3BD), CEC (3BE), LPM (3BB), or PFC (3BF).

The call to QAP has the following format.



cp Control point number
addr Address of the FET

The function code is contained in byte 4 of the first word of the FET.

The following paragraphs describe QAP processors and subroutines.

IIF - INITIATE INPUT FILE (WTIF, WRIF, WFIF)

IIF builds a skeleton system sector and then calls OVJ to examine the first two control cards read by 1CD. Routine 1CD has read the first block of cards into the buffer, but a file has not yet been initiated to hold them. Routine OVJ establishes the name of the input file and OBF is called to enter the name of the input file into the FNT, but not assign mass storage. IIF assigns the mass storage equipment to the device specified in MSAL or chooses any mass storage device using the AMSM function. The FNT entry is then transferred to the system sector, along with the file ID (DISS through DISS+1), machine ID (MISS), and creation date and time (CDSS, read from PDTL). CIO is then called into this PP to begin writing the cards read to mass storage. The system sector built by QAP and OVJ is written by CIO.

LPR - LOAD PRINT DATA (GBPF, PFCF)

LPR checks the equipment type for a PFC printer. If this is the case, overlay 3BB is executed to load the PFC memory. Otherwise, overlay OBP is called to format the banner page. Routine OBP builds the banner page from information contained in the system sector and stores it in the buffer starting at FIRST. The IN pointer is set to the end of the banner page. After the banner page has been put in the buffer, LPR returns and drops QAP.

Overlay 3BB is called to process a load of the 580 PFC memory. If the request in the FET is PFCF, overlay 3BF is executed to load a user supplied PFC image. Otherwise, a space code parameter is extracted from the FET. This value maps to a system defined PFC image contained in overlay 5BA or 5BB. When the proper PFC image overlay has been loaded, it is transmitted to the 580. A check is then made to determine the single line space count to a 6/8 line coincident point. If the space count is not equal to one, the paper alignment is in question. The operator is notified of this condition via the B display (CHECK*I* DISPLAY) and the I display message (CHECK PAPER ALIGNMENT) and the printer is placed in hold status. The operator must enter CONTINUE,xx. (xx is the equipment number) to begin printing. Control is returned to LPR to generate the banner page.

Overlay 3BF is called to process a user supplied PFC image as defined by the current print line (S) in the buffer, prefixed with the V carriage control character. IF the user is not validated to use this feature, or if an error is detected in the PFC image, the print job is aborted. If the priner is not a PFC, the OUT pointer in the FET is advanced over the line (S) containing the PFC image and control is returned to 1CD. Otherwise, the user supplied PFC array is loaded to the printer using subroutines defined in overlay 3BB. The OUT pointer is advanced and control returns to 1CD via BCAX.

TPF - TERMINATE PRINT FILE

TPF terminates a print file when its line limit has been reached. The message LINE LIMIT EXCEEDED is printed and then the dayfile, if one exists, is printed.

PDF - PROCESS DAYFILE MESSAGES (PDMF)

PDF formats and issues to the dayfile various messages for 1CD. These messages include hardware error messages and messages associated with DSD operator requests.

PLE - PROCESS LIMIT EXCEEDED

PLE reads the FNT to determine the file type. If it is a print file, TPF is called. If the file is a punch file, LPH is called.

ACT - ACCOUNTING (ACTF)

Overlay 3BA is called to do summary accounting of unit record activities. The accounting messages are formatted and issued to the account dayfile for output (print, punch, and plot) files. DSP is called to enter an input file into the queue. For input files, the DSP call parameters select queuing (even if job card error), routing to the input queue, returning an error code, and routing to the central site.

For print and punch files, the repeat count is set, error codes are returned, the deferred route bit is set, and FNT address are specified in the DSP call.

PHD - GENERATE LACE CARD (GLCF)

Overlay 3BC is called to load the initial punch data. If the user limit on the number of cards punched has occurred (function PLEF), the card LIMIT is written to the buffer with an EOI (1030) status. If limit has not occurred, the banner card of the job name is formatted into the buffer beginning at IN. The IN pointer is set to the next position and LPH returns and drops QAP.

POR - PROCESS OPERATOR REQUESTS (PORF)

POR loads overlay 3BD to process directives for BATCHIO entered through the operator's console. The following requests are processed.

Function	Description	CIO Function Used
BKP0	Backspace PRUs	44
BKR0	Backspace records	640
BKF0	Backspace files	640 level 17
SKP0	Skip PRUs	600
SKR0	Skip records	240
SKF0	Skip files	240 level 17

Operator requests that do not require file manipulation are handled within the driver (1CD).

Overlay 3BD contains subroutines that perform skipping and backspace CIO functions on the buffer as necessary to properly position the file. The proper function code, level number, and number of records or files are set into the input register and CIO is called into this PP.

CEC - CHANNEL ERROR CLEANUP (CECF)

CEC loads overlay 3BE to cleanup after repeated channel errors. Input files are dropped using a CIO close unload function (170). Output files are requeued through DSP calls. Faulty equipment is turned off with the appropriate error message (EQUIPMENT TURNED OFF) issued.

BCAX - EXIT

BCAX is the exit address for BCA. The function is completed, buffer status updated, and PP dropped.

ERROR PROCESSING

The majority of the error processing done by BATCHIO routines involves hardware malfunction. Hardware malfunction includes channel error processing and bad system sector processing.

The following channel errors are detected.

- Connect reject
- Function reject
- Transmission parity errors
- Incomplete data transfer
- 6681 function timeout
- Equipment function timeout

When these errors are detected, the output file (if any) is requeued, the tracks for the input file (if any) are dropped, the faulty equipment is turned off, and error log messages are issued. The connect reject, function reject, transmission parity errors, and incomplete data transfer errors are retried ERRL (3) times while the 6681 and equipment function timeout conditions are not retried.

If an error is encountered while reading the system sector, the file is released from BATCHIO with the following conditions.

- The name of the file is changed to **BAD 0
- The file type is changed to common (CMFT)
- The write lockout bit is selected for the file; this will leave the file in the system for inspection by an analyst, but cause it to be ignored in further queue processing.

The error messages issued by BATCHIO routines are listed in the NOS Reference Manual, Volume 1, appendix B.

If BATCHIO is aborted or stopped, all active files are rerun by passing the rerun operator flag (RRNM) to 1CD through the buffer control word for each active equipment. This operation is performed by 1IO overlay 3IC.

INTRODUCTION

A system control point (SCP) provides a centralized facility for a module or group of modules to be used by one or more jobs residing at other control points. This centralization provides the following benefits.

- A centralized control over items such as interlocks, resource allocation, and resource access.
- Reduction in the overall usage of central memory. Instead of several jobs having this set of modules in their field length, only one set of these modules needs to occupy central memory.
- The ability of the Network Access Method (NAM) to use the SCP facility to provide an interface between an application program and the network. This facility then provides the controlling element for communications to the network.

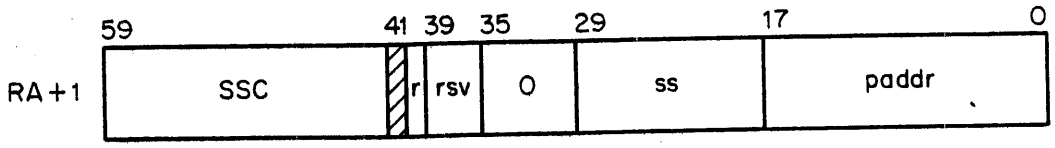
The system control point facility provides a method of linking a user to a subsystem by means of connection indicators which reside with the control point that issues the call to the subsystem. This method is used to handle normal end, abort, and hostile users in such a way as to eliminate various undesirable situations which might otherwise occur in using the SCP facility.

CALLSS MACRO

The CALLSS macro provides two-way communication between a subsystem residing at a particular control point and one or more user jobs residing at user control points (UCP), or from one subsystem to another.

The CALLSS macro is issued by a UCP to request a particular function from a subsystem. A UCP may call more than one subsystem, either serially or in parallel. Also, a UCP may make more than one call to an individual subsystem.

The CALLSS macro can be issued from a control point and generates the following RA+1 call (processed by CPUMTR).



r Autorecall bit (bit 40)

rsv Reserved

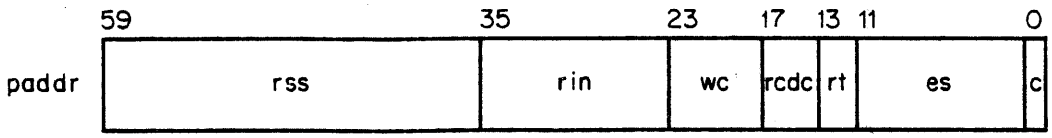
ss Subsystem queue priority

paddr The address within the calling control point's field length of the parameter list that defines the function to be performed; bit zero of the first word of the parameter list or block must be zero

PARAMETER BLOCK

The parameter block pointed to by parameter address paddr is used by the UCP to pass parameter information to the SCP. The parameter block must be at least one word in length.

The first word of the parameter block is used for calling and status information. The second and subsequent words of the parameter block are used for data which is passed to the SCP by the operating system. The format of the first word is as follows.



rss Reserved for subsystem

rin Reserved for installation

wc Word count; length of parameter list minus one that is passed with the request (maximum of 77B)

rcdc Reserved for operating system

rt Return control field set by the user prior to making the SSC or CALLSS call as follows:

Bit	Description
12	Set to 1 indicates to the system that it is to return control to the user if the subsystem is present but unable to accept the user's call (for example, buffer full); bit 2 in the es field is set by the system in such an event If set to 0 and the subsystem is present, the system will hold the request until the subsystem is able to accept it; the c bit is not set until the request is accepted and the processing of the request is finished

13 Set to 1 indicates that the system should:

- Set the es field to indicate which error condition occurred
- Return control to the user on nonfatal error conditions (fatal errors cause a UCP abort)

If set to 0:

- The es field is not set (except bit 2)
- All errors cause the UCP to abort

es Error and status bits; set to indicate error or unavailable system conditions; this field is not set if the rt field is zero:

Bits	Value	Description
1	0	Subsystem present
	1	Subsystem not present
2	0	Subsystem not busy
	1	Subsystem busy
3	0	Subsystem defined
	1	Subsystem not defined as SCP
5-4	-	Reserved
11-6	-	Error condition other than those already described in this field; may assume the following octal values:

00	No error
01-17	Reserved for system errors
20-67	Reserved for subsystem errors
70-77	Reserved for installations

c Completion bit; must be 0 prior to function or macro call; set to 1 when request is completed

MACRO FORMAT

The CALLSS macro is available on systems text SSYTEXT and has the following format.

LOCATION	OPERATION	VARIABLE SUBFIELDS
	CALLSS	ss,paddr,r

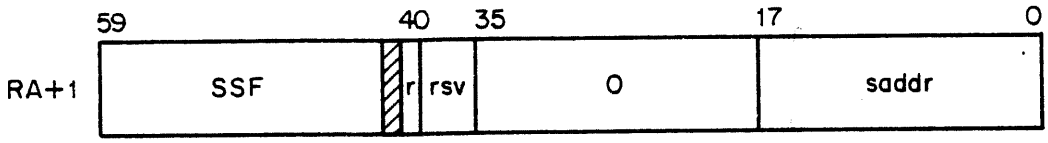
ss Subsystem queue priority

paddr Parameter address as previously described

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

SFCALL MACRO

The SFCALL macro is issued only from a subsystem. It generates the following RA+1 call (processed by CPUMTR).



r Autorecall bit

rsv Reserved

saddr The address (ap+0) within the subsystem of a two- or three-word parameter block that defines the function to be performed

MACRO FORMAT

The SFCALL macro is available on systems text SSYTEXT and has the following format.

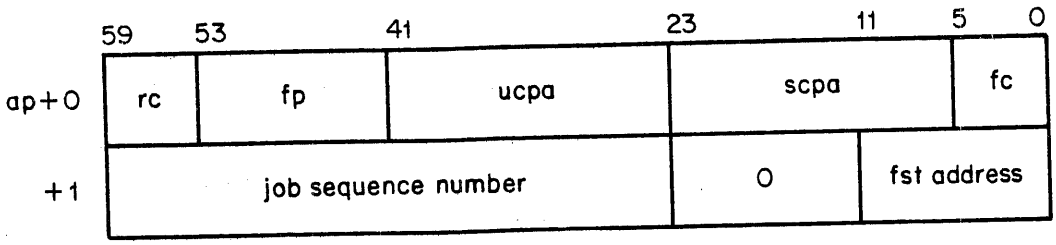
LOCATION	OPERATION	VARIABLE SUBFIELDS
	SFCALL	saddr,r

saddr Parameter address as previously described

r Autorecall

PARAMETER BLOCK

The parameter block for the SFCALL consists of two or three words, depending upon the function code. The first two words are defined as follows.



rc Reply code:

0	No error encountered
34-37	Reserved for installations
40	At least one error detected in list
41	Job identifier is invalid
42	SCP address not within SCP CM or ECS field length
43	UCP address not within UCP CM or ECS field length
44	User job is swapped out
45	User job is not in system
46-55	Reserved for system
56	Unrecoverable ECS abort or parity error occurred during ECS data transfer
57	Connection previously established
60	Connection rejected
61	Connection not previously established
62	Word transfer too long
63	UCP not established with subsystem
64	Subsystem established with receiver
65	Attempt to set illegal error flag
66	Illegal dayfile processing flag
74-77	Reserved for installations

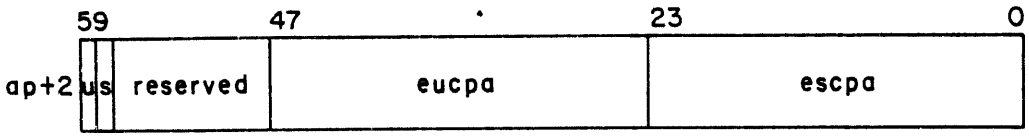
fp Function parameter for particular function code

ucpa Relative CM address within UCP

scpa Relative CM address within the subsystem

fc Function code

The extended read and extended write SFCALL functions (refer to function codes SF.XRED and SF.XWRT) have three-word parameter blocks, with the third word defined as follows.



u If bit 59 is set, eucpa is the UCP ECS address; otherwise eucpa is the UCP CM address

s If bit 58 is set, escpa is the SCP ECS address; otherwise escpa is the SCP CM address

eucpa Extended read or write UCP CM or ECS address

escpa Extended read or write SCP CM or ECS address

SFCALL FUNCTION CODES

The following is a list of function codes (fc) possible on SFCALL macro calls. The fc is always an even number that is incremented by one when the function has been completed.

Symbol	Value	Description
SF.REGR	02	Place message into UCP's dayfile and/or abort UCP
SF.ENDT	06	Indicate end of task to UCP
SF.READ	10	Read UCP CM FL
SF.STAT	12	Request the status of a UCP
SF.WRIT	14	Write to UCP CM FL
SF.EXIT	16	Exit from subsystem status
SF.SWPO	24	Indicate UCP is swapout candidate
SF.SWPI	26	Request system to swap in UCP
SF.SLTC	30	Set long-term connection
SF.CLTC	32	Clear long-term connection
SL.LIST	34	Multiple request capability to process a list of SF.xxxx functions
SF.XRED	40	Extended read of UCP CM or ECS field length
SF.XLST	42	Extended list processing
SF.XWRT	44	Extended write of UCP CM or ECS field length

Symbol	Value	Description
SF.INS1	70	Reserved for installations
SF.INS2	72	Reserved for installations
SF.INS3	74	Reserved for installations
SF.INS4	76	Reserved for installations

CALLSS PROCESSING

The UCP uses the CALLSS macro to communicate with a particular subsystem via the SCP facility. The UCP must have permission for communication with a subsystem via the SCP. The permission required is the setting of the CUCP bit (bit 12) of the access word (AW). If the validation is improper, the UCP is aborted immediately by the system* without allowing EREXIT, REPRIEVE, or EXIT processing.

The amount of data transferred to the SCP (placed at the address defined by ap in the subsystem's RA.SSC word) depends upon whether the subsystem has requested a fixed amount of data (v bit in RA.SSC cleared) or a variable amount of data (v bit set). If data is fixed length, the total amount of data moved to the subsystem is lp words (taken from the RA.SSC word). This amount includes the two-word header at ap+0 and ap+1.

For variable length data, the system transfers wc+1 (taken from the CALLSS parameter block) words in addition to the two-word header. The subsystem determines how much data was actually moved by extracting the wc value from the data word at ap+2. This word corresponds to the first word of the UCP parameter block on the CALLSS.

Following the processing of the CALLSS macro (checking parameter validity and moving the UCP parameter block to the subsystem FL), the operating system causes the subsystem to be assigned a CPU if it is of higher priority than what is currently running. The subsystem then executes as a normal job. The subsystem receives the remainder of the UCP's CPU slice if the CALLSS was issued with or without autorecall. This is done to ensure that the system is not saturated with CALLSS requests that cannot be satisfied.

SUBSYSTEM/UCP COMMUNICATIONS PATH

There must be information available to allow the operating system to properly manage intercontrol point communications. This is provided by connection indicators residing at the control point that issues the CALLSS to a subsystem.

Four bits are assigned to each possible subsystem. One of these bits indicates that the UCP has been granted access to the

---

* SYET error flag is set on unauthorized UCP.

subsystem and the others indicate the number of requests awaiting a response from the subsystem.

The bit indicating that access has been granted is set on request of the receiving subsystem through the SFCALL macro. The function which sets this bit is SF.SLTC (set long-term connection). This bit remains set until cleared either by n.OVERRIDE or on request of the appropriate subsystem.

The field indicating that the UCP is awaiting response from the subsystem is incremented by the operating system whenever a CALLSS is issued.

The long-term connection can be set only when the wait response field is nonzero. This is necessary to protect the UCP from an unsolicited connection being established arbitrarily by a subsystem.

This 3-bit wait response field causes the UCP to be locked-in (not a normal candidate for rollout). This field is zero when no outstanding requests remain for the appropriate subsystem.

Whenever the wait response field is nonzero, the operating system inhibits rollout of the UCP. This is accomplished by giving the UCP an installation specified number times its normal CM slice whenever the UCP has outstanding wait response indicators set without corresponding swapout allowable indicators set. Then after the CM slice has expired, the UCP is considered a normal candidate for rollout.

Before the wait response field is incremented, a check is made to see if the maximum is met (seven outstanding requests). When the maximum is attained, the next request forces the UCP into recall, if recall was selected on the call, until at least one outstanding request has been satisfied through a subsystem response that decrements the count in the wait response field.

(The SF.ENDT function issued by the subsystem causes the wait response field to be decremented.)

If recall is not selected on the call, RA+1 is cleared and the subsystem busy status is returned to the user. The maximum number of outstanding requests in the wait response field is an installation parameter initially set to 7 (maximum field size).

CONNECTION STATE TABLE

Although one subsystem is shown in table 18-1, the same conditions exist independently for all possible subsystems.

TABLE 18-1. CONNECTION STATE TABLE

User Control Point State	A	B	C	D
Long-Term Connection	N	N	Y	Y
Wait Response Nonzero	N	Y	N	Y
No Connection	X			
Awaiting Response		X		X
Locked In*		X		X
Candidate for Rollout	X		X	
SF.SLTC Allowed		X		
SF.SWPO Allowed		X		X
SF.SWPI Allowed			X	
SF.CLTC Allowed			X	X

The following steps refer to table 18-1.

1. Initially, any control point is in state A.

2. When a control point in state A issues a CALLSS macro (except for the special initialization call to the operating system; refer to SSF Call Processing), it is placed in state B by the operating system if the subsystem exists. This locks the UCP until the subsystem replies.

3. The subsystem then must reply to the request which places the requestor in state A or C.

4. In any state except state A, the subsystem may respond with the hostile user SFCALL that eliminates the caller. Refer to Hostile User.

5. Refer to End Processing for proper action by a UCP or a subsystem on normal termination.

6. Refer to Abort Processing for information about what happens when either a UCP or a subsystem aborts.

END PROCESSING

It is necessary to inform the subsystem(s) when a control point established in communication ends. The procedure described here is for normal end cases.

---

* Locked in implies the UCP is given an installation specified number of times is CM slice before rollout, occurs.

End UCP

It is possible for any UCP to inform its subsystem(s) that it is ending. That is, every subsystem should have a provision to have its connection disestablished on request.

In the event it does not, the operating system uses the following procedure.

If the UCP ends in any state other than A (refer to table 18-1), the operating system initiates termination processing identical to abort processing except that the status is end rather than abort. The user job remains in the system, in either case, until the subsystem(s) clears all of the outstanding connection indicators or operator intervention occurs (n.OVERRIDE.).

End Subsystem

A subsystem is in a different position than a UCP since requests come to it unpredictably. In order to establish a logical termination procedure, the subsystem must first disallow the receipt of further requests (clear RA.SSID). It should next answer all outstanding requests (UCPs in state B, C, or D) with a message indicating the subsystem is terminating, and then it should end.

When 1AJ detects that a subsystem has ended, reprieve processing is checked and allowed for the subsystem. The subsystem should have an established cleanup procedure to inform UCPs of the end. After the reprieve processing (or if no reprieve processing was established), 1AJ checks the outstanding connection count (word SSOW) in the subsystem's control point area. If the count is nonzero, 1AJ attempts to set the subsystem end/abort interlock bit (INWL word, bit 0) via the UADM monitor function. If unable to do so, 1AJ is placed in recall. It then performs an FNT search and depending upon the file type performs the following steps.

1. If at a control point, 1AJ issues an SFIM monitor function* to attempt to interlock that particular FNT entry. If unsuccessful, 1AJ pauses and reissues the SFIM. When the FNT interlock is set, a CEFM monitor function sets the SSET (subsystem aborted) error flag on the specified UCP. The FNT interlock is then cleared via another SFIM call and the FNT search continues. Upon setting the FNT interlock, the job must be reexamined to determine whether the job state has changed. If the state has changed, the UCP is handled according to its current state.

2. If the job is in the rollout queue (ROFT) or timed event rollout queue (TEFT), the FNT interlock is used (SFIM)

---

* The SFIM monitor function controls the transition states (state of rolling in or out) of jobs in the system.

and if successful bit 5 of the FNT is examined. If set, the system sector is read to determine whether the UCP had connections with the aborting subsystem. If bit 5 of the FNT is not set or if no connections were established, the FNT interlock is cleared. If connections are established, the FNT pseudo channel (FNCT) is used as an interlock to allow 1AJ to change the error priority to SSPS. File types TEFT are changed to ROFT. The FNCT channel and SFIM interlocks are then released and the search continued.

A UCP in state A (that is, without long-term or wait response status) at the time of subsystem end is not informed of the ending condition. Subsystems must place all UCPs which require notification in the long-term state.

After going through the preceding procedure, all jobs which were in state B, C, or D with the subsystem are informed of the end condition. Other UCPs (those which were in state A at the time of the subsystem end) receive a subsystem not present status from the operating system if they attempt a CALLSS after the subsystem is actually dropped. If these jobs attempt a CALLSS during the end process, they also receive the subsystem not present status.

Subsystems which deal with UCPs on a short-term basis only (that is, no long-term connection) must be designed to allow for proper resynchronization with the UCP if the subsystem ends.

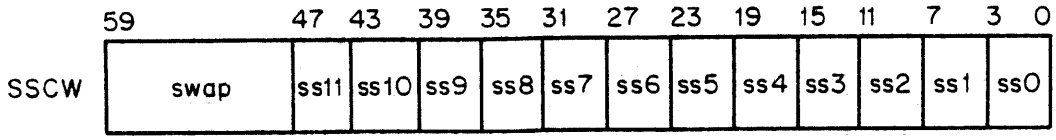
ABORT PROCESSING

It is necessary to inform the subsystem when a control point established in communication aborts.

Whenever a UCP tries to terminate in any state other than A, the subsystem is informed via a message generated by the operating system and sent to the subsystem. It is the responsibility of the individual subsystems to clear any outstanding connections with the UCP. This may be done by individual SF.CLTC and SF.ENDT requests or by a special SF.ENDT request (UCPA = -1) which clears all outstanding connections for this subsystem. The UCP remains in the system until all subsystem connections are cleared.

This method of informing the subsystem of the UCP abort is handled by 1AJ through the monitor function TDAM. All wait response and long-term connection bits are checked. If any are set, the appropriate subsystem queue priority is obtained from the connection indicators position (subsystem index). This queue priority is then used by the TDAM function to inform the appropriate subsystem. The connection indicators remain set until cleared by the SCP or an n.OVERRIDE command is issued. Thus the UCP remains in the system and no job advance occurs until these connection indicators are cleared.

The format of SSCW is as follows.



swap — Swapout allowable indicator for subsystem indexes 11 through 0.

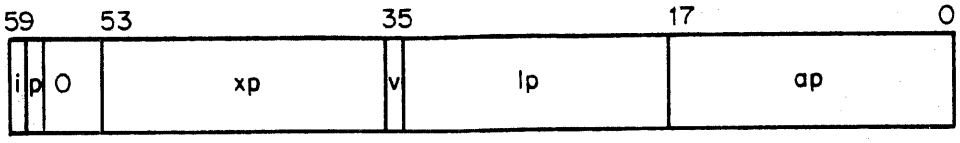
ss(i) — 4-bit subsystem indicator. The upper bit indicates long-term connection, if set, for subsystem index i. The lower 3 bits are the wait response indicator count (number of outstanding requests to a maximum of 7) for subsystem index i.

In the TDAM monitor function, the job sequence number, FST address, and error flag are then sent to all the appropriate subsystems. If the call is not accepted, a check is made to see if the subsystem is inactive (a reply code of 4 from TDAM). If the subsystem is active but the call was not accepted, the message is reissued. If the subsystem is inactive* or the call was accepted, the message is issued to the next appropriate subsystem.**

NOTE

1AJ issues a special TDAM request which CPUMTR traps to send to the receiving buffer pointed to by RA.SSC.

The format of RA.SSC (RA+51) is as follows.



i — Set by CPUMTR when a request has been placed in the parameter area. It is cleared by the subsystem to acknowledge the request has been received. When the bit has been cleared, ap should point to a parameter area which the subsystem is prepared to receive on the next request. After clearing this bit, the subsystem should not attempt to rewrite RA.SSC until the next request is received.

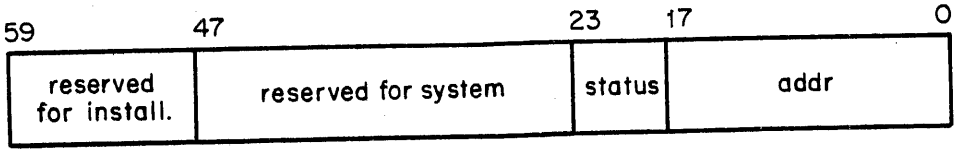
---

* If the subsystem is inactive the connection indicators for this subsystem are cleared at this time.

** 1AJ is placed in PP recall if any subsystems are busy.

p	If bit 58 is zero, the system does not impose any additional restrictions on access by UCPs for this subsystem (no additional validation is required besides CUCP bit set in access word). If set, the system allows access to the SCP (CALLSS macro request transfer) only by UCPs that are privileged programs. A privileged program is one with an SSJ= entry point or a queue priority greater than MXPS (subsystem). These restrictions also imply that the UCP program was loaded from the system library. Unauthorized UCPs are aborted immediately by the system* without allowing EREXIT, REPRIEVE, or EXIT processing, and no data is transferred to the SCP's CALLSS receiving buffer.
xp	Address within the subsystem for the UCP exchange package.
v	If set, indicates that wc words (refer to CALLSS Macro) are transferred from UCP rather than lp. This allows the UCP to transfer a variable number of words ($\leq$ lp) to the subsystem.
lp	Length of the request parameter area.
ap	Address within the subsystem for UCP parameters.

The format of location ap is as follows.



status	This field may assume the octal values 0 through 77: 00 Normal UCP messages (addr not equal to 0) 01 System message (UCP ended) 02 System message (UCP aborted) 03 Connections forcibly broken (n.OVERRIDE.)

---

* SYET error flag is set on unauthorized UCP.

04 SCP aborted (this status is returned to a UCP that is also an SCP)

addr — Address of a parameter block for this request (same as parameter address on SSC call; refer to CALLSS macro). If this field is set to 0, the system has generated this message with status bits set indicating end or abort conditions of the UCP.

Processing for end is the same except that the end status is set rather than abort. When the subsystem aborts, this procedure should be essentially identical with end processing.

HOSTILE USER

The hostile user flag is used by subsystems to protect themselves from UCPs that are possibly endangering the system. The subsystem, through this flag, blocks normal exit processing so that the offending UCP is forced from the system.*

This is a powerful feature that requires some limitations in order to protect users from themselves. Sufficient protection should be provided to ensure the following.

- The request to set the hostile user flag is issued by a subsystem.
- The control point receiving the hostile user flag is in state B, C, or D with the issuing subsystem. This assures that the flag is being set judiciously by a solicited subsystem.
- The subsystem issuing the request is not in state B, C, or D with the receiver. Two subsystems may be connected to one another with the hostile subsystem attempting to set the hostile user flag on the nonhostile subsystem.

These checks minimize the UCP's ability to jeopardize the performance of the SCP facility.

COMMUNICATION ENDS AND ABORTS

If the UCP is in a state other than A, an error flag SSET is set on the UCP.

---

*The SYET error flag is used to identify the hostile user. SYET is handled in 1AJ so as not to allow exit or reprieve processing.

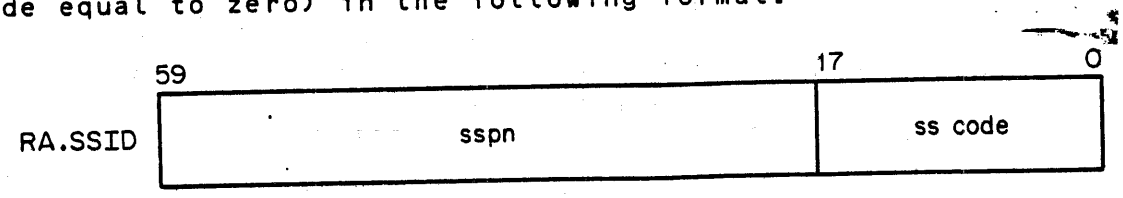
The operating system places information in bits 23 through 18 of the ap word (refer to Abort Processing) which indicates end or abort conditions of the UCP.

In order to communicate with subsystems of all types, system messages are two words in length. These messages consist of word ap + 0 and ap + 1 only. This is necessary in order to observe the stated rule that the minimum value of lp is two.

Also, these two words are sufficient to identify the UCP of interest. The system generates the message with addr equal to zero and a normal word at ap + 1.

CPUMTR PROCESSING OF SSC CALLS

SSC is defined as a special RA+1 call that is processed by CPUMTR instead of being passed to the PP program SSC. If the subsystem code (ss code in call below) is equal to zero, this call is treated as a null request. This is done to achieve a level of compatibility with NOS/BE which treats this special case as a request to attain subsystem status. NOS/BE checks to see if the subsystem is already at a control point. This special case situation is not needed in NOS because of the queue priority hierarchy established for subsystems. The NOS SCP facility does not automaticaly initialize or bring up subsystems as in the NOS/BE implementation. These subsystems must be initialized via appropriate DSD console commands prior to any UCP requesting service from that particular subsystem. Otherwise a subsystem not present status is returned. Thus, for NOS it is imperative that subsystems set up RA.SSID prior to any future SCP issuing this special CALLSS macro call (subsystem code equal to zero) in the following format.



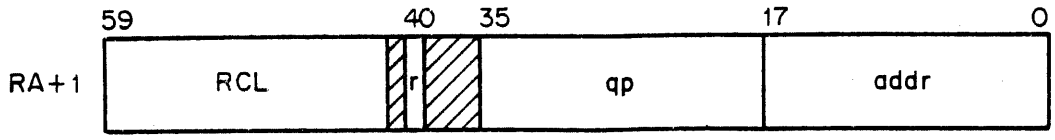
sspn	Subsystem program name
ss code	Subsystem queue priority

RA.SSC (refer to Abort Processing) is a pointer that tells CPUMTR where to put incoming requests from a UCP. The parameter area can receive a request when bit 59 is zero.

CPUMTR sets bit 59 when the transfer is complete. It is cleared by the subsystem whenever the buffer pointed to by ap is ready for more data. The subsystem should never update RA.SSC unless bit 59 is set.

Because subsytems in general using the SCP facility have a high CPU priority, the subsytem should acknowledge receipt of the request as soon as it is able to do so.

Whenever an SSC call is made with recall, CPUMTR sets RA+1 of the UCP to the following.



r	Autorecall bit
qp	Queue priority of subsystem CALLSS issued to
addr	Address of the parameter block

CPUMTR checks the UCP control point area (refer to Abort Processing) to ensure that the UCP has seven or fewer (installation defined) requests outstanding. If not, the UCP is placed in recall. If recall was not selected on the call, RA+1 is cleared and the subsystem busy status is returned. If the subsystem code is nonzero, CPUMTR checks its validity and finds the subsystem control point address. Table 18-2 summarizes the other checks made at this time (refer to CALLSS Macro, parameter block description).

TABLE 18-2. UCP/SUBSYSTEM CHECKS

Subsystem State	Return (rt) Code Field	Action
Subsystem not present, or ap + lp >FL	Bit 13=0	Abort UCP and set error flag.
	Bit 13=1	Clear RA+1 and set es bit 1.
Subsystem busy * (bit 59 of RA.SCC is set)	Bit 12=0	Place UCP in recall.
	Bit 12=1	Clear RA+1 and set es bit 2.
xp not 0	NA	Move UCP exchange package to xp.
xp = 0	NA	Do not move exchange package.

After completing the validity checks, CPUMTR moves the job sequence number and FST address to ap+1. Then the parameter block is moved from the UCP to the subsystem by either a variable length move of data if v (bit 35) of RA.SSC is set or else a fixed length move of data if v is cleared (refer to SFCALL macro) At this point, CPUMTR increments the wait response indicator to .

---

* Subsystem is being storage moved, advanced, or user control point has reached the maximum allowable number of outstanding requests.

inhibit subsequent rollouts. The wait response indicators being nonzero allow the UCP to get an installation specified number times its CM slice. (Refer to SF.SWPO for overriding the wait response indicators.)

SSF CALL PROCESSING

SSF is also a special RA+1 call processed by CPUMTR. This call is only made by a subsystem. Thus, the queue priority is read (JCIW) and if the caller is not a subsystem, the caller is aborted. The job identifier (job sequence number and FST address) is read from the second word of the SSF parameter block (refer to SFCALL Macro). Then the first word of the parameter pair is read and the function code is checked for validity. If valid, the appropriate SSF function is called (processed by CPUMTR). For functions requiring UCP/SCP addresses or other parameters, the corresponding fields are also checked for validity. Table 18-3 identifies the checks performed on the UCP before transferring to the appropriate function processor. The error status, if applicable, is returned in RC to the subsystem and the completion bit is set.

TABLE 18-3. CHECK USER JOB TABLE

UCP State	Error Status
Job identifier is invalid	41B
UCP address is out of range (ucpa or eucpa + fc if applicable)	43B
UCP is swapped.	44B
Job identifier on SSF call does not match that of UCP's control point area. User job not in system.	45B

SF.ENDT (06)

This function informs the operating system and the UCP that the subsystem task has been completed.

If none of the errors in table 18-3 were encountered, a check is made on the wait response indicator field to see if the field is zero. If it is, error status 63B (UCP not established with subsystem), is returned. If the field is nonzero, it is decremented from the subsystem control word (SSCW).

If the count then equals zero, the UCP is scheduled normally, according to the queue priority present in the control point

area. If the ucpa field is equal to a minus one, all outstanding connections are cleared from the UCP control point area for this subsystem. If the UCPA field is nonzero and positive, this field is used as the address of the CALLSS complete bit (bit 0) and the bit is set (corresponds to address addr supplied with CALLSS). If bit 42 of the fp field is set, the CPU is switched immediately from the SCP to the UCP if the UCP was in recall on this CALLSS.

SF.READ (10), SF.WRIT (14)

SF.READ and SF.WRIT transfer data (up to 100B words) from the UCP's CM field length to the SCP's CM field length or vice versa.

All transfers of data are done by the compare/move unit (CMU), if available and not a dual CPU configuration, or by the generalized 8-word move loop.

CPUMTR first checks for the fatal error scpa+fp out of range and issues the error status 42B (SCP address is not within the subsystem CM FL) if necessary. The user job is then checked as in table 18-3. If the wait response field is zero, error status 63B (UCP not established with subsystem) is returned. If fp exceeds the maximum transfer size (currently 100B), error status 62B (word transfer too long) is returned.

If no errors occur, fp words are then either moved from ucpa to scpa (SF.READ), or moved from scpa to ucpa (SF.WRIT). Completion bits are then set in the SSF call and RA+1 is cleared.

SF.XRED (40), SF.XWRT (44)

These functions allow the transfer of data (length limited only by 12-bit fp field size*) from the UCP's CM or ECS field length to the SCP's CM field length and from the UCP's CM to the SCP's ECS field length (SF.XRED) or from the SCP's CM to the UCP's ECS field length (SF.XWRT).**

The u and s fields in the third word of the parameter block (ap+2) indicate whether the eucpa and escpa fields are CM or ECS addresses in the UCP and SCP. The ucpa and scpa fields in word one of the parameter block (ap+0) are not used for these functions.

CPUMTR first checks for the fatal error escpa+fp out of range and issues the error status 42B (SCP address is not within the subsystem CM/ECS field length) if necessary. The user job is then checked as in table 18-3. Reply code 43B is returned if

---

* Although large transfer lengths are allowed, the system overhead to move large amounts of CM or ECS in monitor mode is extensive and overuse is a misuse of the SCP facility.

** Direct transfers from UCP ECS to/from SCP ECS is not supported; SCP is aborted with monitor call error if such a transfer is requested.

eucpa+fp is not within the UCP's appropriate CM or ECS field length. If the wait response field is zero, error status 63B (UCP not established with subsystem) is returned.

If no errors occur, as much of the transfer is processed as monitor mode time for one RA+1 request allows.* To prevent performance degradation, CPUMTR prematurely terminates CM and ECS transfers requested by SF.XRED and SF.XWRT functions when time allowable in monitor mode is exhausted, and then decrements wc to the remaining word count, increments escpa and eucpa addresses by the transferred data length, sets the complete bit, and returns control to the SCP. To complete the read/write, the SCP must reissue the SF.XRED/SF.XWRT SSF RA+1 call until wc is zero.

An SCP requesting a data transfer of less than or equal to 100B words via SF.XRED or SF.XWRT does not have to check the word count to guarantee the data transfer is complete, since transfers of less than or equal to 100B words are not fragmented (compatible with SF.READ and SF.WRIT functions).

To allow use of SF.XRED and SF.XWRT in a list, the extended list function SF.XLST, with two words per list entry, must be used. The second word of the SF.XLST list entry for an SF.XRED or SF.XWRT function corresponds to the third word of the nonlist SF.XRED/SF.XWRT parameter block. Large data transfers requested by SF.XRED or SF.XWRT within an SF.XLST are prematurely terminated when monitor mode time is exhausted; the amount of data transferred depends on the number and complexity of prior functions in the list that have already been processed during this RA+1 request. The SF.XRED/SF.XWRT wc field is decremented to the remaining word count and the escpa and eucpa addresses are incremented by the transferred data length. The complete bit is not set on the SF.XRED/SF.XWRT function in the list, and the list address (scpa field in first word of SF.XLST parameter block) remains pointing to the SF.XRED/SF.XWRT function until all data has been transferred (wc=0), so that subsequent reissue of the SF.XLST SFCALL by the SCP continues the data transfer where it left off.

The inclusion of an SF.XRED or SF.XWRT function in an SF.LIST list (one word per entry) causes the SCP to abort with a monitor call error.

SF.EXIT (16)

With this function the subsystem is removed from SCP status.**

---

* The amount of data that can be transferred at one time depends upon the CM and ECS transfer rates estimated for the machine. The CM transfer rate is varied depending upon whether the machine has a CMU or is a stack machine. The ECS transfer rate is varied depending upon the ECS size, and whether ECS is shared in a multimainframe environment.

** RA.SSID is cleared (the subsystem remains a subsystem but no longer acts as a SCP).

When CPUMTR processes this request, it determines the UCP's associated with this subsystem to inform them that the subsystem is ending its SCP status (refer to End Processing for a further discussion of this case).

SF.SLTC (30), SF.CLTC (32)

These functions are used to control the state of the long-term connection bit belonging to the UCP. The operating system defines a bit position belonging to each subsystem so that it is only necessary to ask for the bit to be set or cleared at the proper time. (CALLSS processing discusses the various states possible when using these functions in relation to the wait response field previously discussed.)

SF.SLTC - Set Long-Term Connection

Possible status returns are the following.

00	Connection established
57	Connection previously established

Existing status returns 41, 44, 45, and 63 may also occur. Any error will cause the long-term connection to remain in its original state.

SF.CLTC - Clear Long-Term Connection

Possible Status Returns are the following.

00	Long-term connection disestablished
61	Connection not previously established

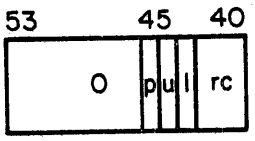
As in SF.SLTC, status returns 41, 44, and 45, may occur. Any error will cause the long-term connection to remain in its original state.

SF.STAT (12)

The subsystem requests the current status of the user job. The rc field is used for reply.

rc	Description
0	The user's job is in a normal state of execution
41	Job identifier invalid
44	The user job is swapped or swapping
45	The user job is not in system

The status of the connection indicator is returned in fp when rc is 0 in the following format.



p If zero, UCP is not a privileged program. If set, UCP is a privileged program. A privileged program is one with an SSJ= entry point or a queue priority greater than MXPS (subsystem).

u If zero, UCP is not a privileged user. If set, UCP is a privileged user. A privileged user is a system origin job or a user with system origin privileges (CSOJ bit set in access word) when the system is in DEBUG mode.

l Long-term connection

rc Request count

The ucpa and scpa fields are not used on this call, but should be zero in case of future expansion.

SF.SWPO (24)

This function indicates that the UCP is a candidate for rollout. CPUMTR checks the user job for error status returns 41, 44, 45, and 63. If there are no errors the swapout allowable field is set in the subsystem control word SSCW and the UCP is checked for the conditions necessary to swap out the UCP as described under CPUMTR Processing of SSC Calls. For each set of wait response indicators set, there must be a corresponding swapout allowable indicator set. If these conditions are met, CPUMTR calls 1MA to the UCP control point to set the forced rollout priority (FRPS) on the UCP and to set the SF.SWPO completion bit. If these conditions are not met, the completion bit is set and the function is completed.

Because the wait response field remains unchanged after this function, 1SP checks the swapout allowable field before checking the wait response field. If the swapout allowable field is set, 1SP does not check the wait response field for this subsystem. Thus, whenever the UCP has either the wait response field set to zero, or the swapout allowable field set to one for all subsystems it is communicating with, the operating system no longer assigns an installation specified number of times its CM slice and normal scheduling occur (refer to SSCW word).

NOTE

The following special subsystem requests are accomplished by use of the monitor auxiliary

processor (1MA) because of the overhead involved in doing the request in monitor mode. Routine 1MA must be CM resident in order for data to be passed by CPUMTR to 1MA through the message buffer.

SF.REGR (02)

This function sends a dayfile message and/or aborts the UCP. Several checks are made by CPUMTR prior to calling 1MA for assistance. CPUMTR and 1MA checks determine the current state of the UCP with the subsystem (refer to CPUMTR Processing of SSF Call) and issue the following error codes if appropriate.

63 UCP not established with this subsystem. (Not in state B, C, or D)

64 Issuing subsystem is established with the receiver (UCP is a subsystem)

65 Attempt to set an illegal error flag

66 Illegal dayfile processing flag *
(The subsystem wants to abort the UCP, but has issued an improper error flag.) **

Error status returns 41, 42, 44, and 45, are also possible.

If no errors occur, a value of zero in the ucpa field means the error flag at the UCP remains unchanged. Otherwise, the error flag is set to ucpa. If there are other subsystems involved with this UCP, NOS performs abort processing (refer to Abort Processing). Finally, if scpa is nonzero, 1MA issues the message beginning at scpa to the UCP dayfile. Currently the maximum message length is 4 words (40 characters). This maximum is imposed because CPUMTR prestores the message in the 1MA message buffer and then calls 1MA to the UCP control point. Routine 1MA upon completing this function sets the complete bit by means of the TDAM monitor function due to the fact that 1MA is assigned to the UCP.

SF.LIST (34), SF.XLST (42)

Multiple special subsystem requests to be made with a single SFCALL request. The format of the SF.LIST/SF.XLST function parameter word pair must be as shown for the SFCALL macro.

---

* fp is an invalid dayfile index. Valid range of indices is from 0 to 7.

** Presently the only legal error flags are:
SF.SEXX (1) - set normal subsystem (FSET) error flag
SF.SEHX (2) - set hostile user (SYET) error flag

The scpa field in the SF.LIST/SF.XLST parameter block points to a list of SFCALL functions to be processed for one UCP (identified by job id in second word of SF.LIST/SF.XLST parameter block). The wc field contains the number of SFCALLs in the list.

The SF.LIST list consists of one-word entries corresponding to the first word of the nonlist SFCALL function parameter block (SF.READ, SF.WRIT, SF.REGR, SF.STAT, etc.). The SF.XLST list consists of two-word entries where the first word corresponds to the first word of a nonlist SFCALL function parameter block (ap+0) and the second word corresponds to the third word of the nonlist SFCALL function parameter block (ap+2) for the SF.XRED and SF.XWRT functions and is reserved for all other functions. SF.XRED and SF.XWRT functions cannot be specified in an SF.LIST list, since there is no room for the extended address word (SCP aborts with monitor call error in such a case). Refer to SF.XRED and SF.XWRT functions for a description of extended transfer length processing within an SF.XLST list.

SFCALL functions in a list are in the same format as for nonlist SFCALL functions, except that the job id is not duplicated throughout the list. The system processes as much of the list at a time as monitor mode time allows. After a function in the list is processed, its complete bit (and error reply code if error has occurred) is set, the list address is advanced, and the number of functions in the list remaining to be processed (wc field in SF.LIST/SF.XLST parameter block) is decremented. After the list processing monitor mode time is exhausted, or when an error status return is required, or after a special 1MA function is processed, the complete bit is set on the SF.LIST/SF.XLST function and control is returned to the SCP. The SCP must clear the complete bit in the SF.LIST/SF.XLST parameter block and reissue the SFCALL until all functions in the list have been processed (wc = 0).

Before processing any functions in the list, the SF.LIST/SF.XLST function itself is validated and the user job checks defined by table 18-3 are performed. If all these checks pass without error, processing of individual entries within the list is initiated. If the UCP is swapped out, an SF.LIST is rejected with RC44 even though an SF.SWPI may be the first function requested within the list.

When the SF.LIST/SF.XLST parameter word pair's fc field is set complete by the operating system, the rc and fp fields must be examined for proper action. The order of examining these fields must be determined by the subsystem design.

The SF.LIST/SF.XLST call is processed partially by CPUMTR and 1MA. The functions which CPUMTR processes are done in monitor mode; as such, these functions are time critical. Therefore, the number of functions processed by CPUMTR may not exhaust the list in order to avoid serious system degradation.

Also, various other functions are considered time consuming in themselves. These functions are processed by 1MA. Thus, it is quite likely when processing a large list of functions that the

entire list will not be exhausted in one pass. The following paragraphs discuss what happens when this occurs.

- If fp is 0, the entire list has been processed by the operating system. If fp is nonzero, it indicates that processing of the list was abandoned by the operating system. The value of fp is set to the number of entries remaining is the list and scpa is set to the address of the first entry in the remaining list. Essentially, the subsystem may reissue the SF.LIST/SF.XLST call by resetting the fc field and executing the SFCALL macro until fp is 0.

- It is also important to check the rc field to determine if any errors occurred while processing the list. The operating system sets the field only if an error is detected, so it is the responsibility of the subsystem to initialize this field prior to the SFCALL. Since the operating system does not check the rc field, but only sets it on errors, multiple issues of the same SF.LIST/SF.XLST request (until fc is set complete and fp is zero) accumulate error returns whether or not the entire list is processed on one SFCALL.

- The operating system only aborts the SCP during SF.LIST/SF.XLST processing if the scpa is not within the subsystem's FL (42B error) or if any fatal errors are encountered during list processing. Error 42 implies that none of the list has been processed. This check is made prior to initiating the list process on the first and only word pair. This address must point to the list of contiguous function words (one word per function) to be performed for this particular user. Subsequent fatal errors are returned in the function's reply code with a 40B error set in the first word pair.

- The detailed error conditions must be determined by examining the individual list entries whenever the SF.LIST/SF.XLST fc field equals 40B.

- List entries are processed sequentially by the operating system and those entries detected as erroneous for any reason are considered completed. It is expected that in most cases the entire list will be processed on one SFCALL. The option of abandoning the list is provided to allow the operating system to take corrective action if it decides that either the length of the list, the complexity of the processing, or other reasons have caused an excessively long uninterruptable interval.

- If the SCP is aborted due to an error in one of the list entries other than the SF.LIST/SF.XLST, rc = 40B, scpa and fp are updated, and fc is set complete. The proper return statuses are also placed in the offending list entry.

SF.SWPI (26B)

The user job is checked to see if it is at a control point. If it is at a control point*, CPUMTR sets the completion bit and clears the swapout allowable indicator in the subsystem control word. If the user job cannot be found, a user job not in the system error (45B) is returned and the completion bit is set. If the user job is in the rollout queue, CPUMTR calls 1MA and processes this function. If the UCP has been swapped out, 1MA checks the control point assignment in the UCP's FNT. If equal to 37B, the user job swapped out status is returned.

This condition only occurs when two subsystems issue this function on the same UCP when the UCP is in the rollout queue. The error occurs because the other subsystem has already begun the swapin procedure, and the queue priority must identify the subsystem.

Routine 1MA sets the FNT interlock (SFIM) and checks the special subsystem range of queue priorities beneath MNPS. If the queue priority is not in the range, it is set for the subsystem which issued the SF.SWPI, the control point assignment is set to 37B, and the FNT interlock is released. Then the subsystem queue priority and the address of the SF.SWPI completion bit is stored in the rollout file system sector at location SWSS and the control point assignment removed. The control point assignment is used as an interlock on the system sector write. Later, 1SJ checks for this subsystem related queue priority range beneath MNPS and forces the maximum queue priority for this origin type to assure rollin of the UCP.

Then when 1RI is called, assuming the UCP was selected, 1RI checks the same subsystem related queue priority range (this new queue priority assigned by 1SJ is only done internally; therefore the FST still contains this special subsystem related priority). Routine 1RI obtains the subsystem queue priority and address of the completion bit address for the SF.SWPI call from the rollout file system sector. The control point of the subsystem is found** and the completion bit address is checked for out of range. If out of range, a system dayfile message is issued and the subsystem is aborted. Finally, if the address was not out of range the completion bit is set for the SF.SWPI call. This bit is only set when the UCP has been swapped and the swapout allowable field is cleared for this subsystem. It is the responsibility of the subsystem to check this completion bit before attempting to communicate with the UCP.

---

* The flag for SF.SWPO in progress (bit 12 of word SS0W in the control point area) is examined. If set, a status of the user job swapped out is returned because the previous SF.SWPO has not completed. This is necessary to eliminate a potential timing problem which could result in the user control point being rolled out indefinitely.

** If the subsystem is not found, the completion bit and wait response indicator are not set. Abort processing of the subsystem handles this case if the UCP was established with the subsystem.

I/O queue protection and dayfile recovery allows the system to recover queues and dayfiles across all levels of deadstart. It offers the ability to define the residence of the system dayfiles, and to terminate a dayfile and start a new one while the system is running.

For queue recovery, a catalog of the recovered queues is built at deadstart (level 0). This feature makes it possible to remove files from this catalog and add them to the active queues, or to remove files from the active queues and place them in the list of inactive queues.

Utility programs are provided to dump and load I/O queues, back up queued files, take queues out of the system temporarily, and move queued files from one device to another. There are also list utilities that list the files on the dump tape (LDLIST), list selected inactive queues (QLIST), and list the permanent files that were created by DFTERM (DFLIST).

PRESERVED FILES

A preserved file is one that is retained through a level 0 deadstart. Prior to the I/O queue protect feature, permanent files (direct or indirect access) were the only preserved files. With this feature, I/O queue files and system dayfiles (system, account, and error log) are also preserved files.

QUEUED FILES

When a file is placed in the input queue and I/O queue protect is enabled, it is placed on a temporary device with an input queued file type, and the track linkage for the file is labeled preserved in the TRT. Similarly with output files.

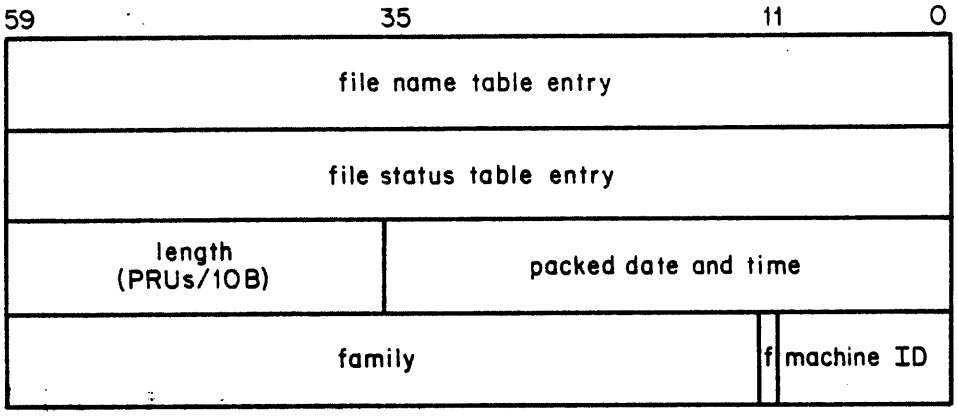
Active queued files (files with an FNT/FST entry and ready for processing) can be made inactive by use of the QREC or QMOVE utilities. This procedure releases the FNT/FST entry and makes an entry in the inactive queue file table (IQFT) on the disk that the file resides. This process is referred to as a dequeue.

The operator can make inactive files active and available for scheduling (input files) or printing/punching (output files). This procedure makes an FNT/FST entry for the file and deletes the IQFT entry on the disk. This process is referred to as a requeue.

Queued files are placed on any temporary mass storage device. The device mask and secondary mask are not considered during their allocation.

IQFT ENTRY

The IQFT is a pseudo catalog file containing a four-word entry for each queued file with all of the information necessary for re-entry into the FNT/FST at a later time. The format of the IQFT entry is as follows.



f System sector format flag

0 If system sector not formatted

1 If system sector formatted

QUEUED FILE ENTRANCE

The entrance of a queued file into the queue (input or output) is preceded by the following.

- The FST for the file is written into the system sector at FSSS (the FST as it exists when the file is initially queued). FNSS in the system sector contains the FNT (except for the control point number). The family name is written into location FMSS of the system sector.

- The preserved file bit (the bit set in the TRT to indicate this file is preserved over all levels of deadstart) for the first track of the file is set in the TRT for the device and the checkpoint request is set (bypassed if I/O queue protect is disabled).

QUEUED FILE REMOVAL

The following steps are taken when a queued file is removed from the queue.

- The file's tracks are dropped and, if I/O queue protect is enabled, the checkpoint request is set.
- A message is issued to the account file identifying the job.

INPUT files are not considered removed from the queue until completion of the job.

System origin jobs are not protected in the input queue, but are protected in the output queue.

QUEUED FILE RECOVERY

The actual recovery of queues begins with routine REC scanning all available devices and reading system sectors for all preserved files.

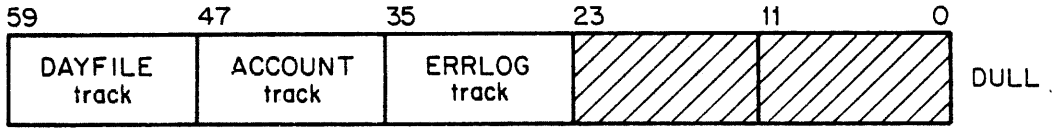
REC creates a variable length file on the device it is recovering containing a list of all queues recovered. Each file recovered occupies a 4-word entry in this file (refer to IQFT Entry). This file is known as the IQFT and resides on the device only if there are inactive queues on the device.

The first track of the IQFT is kept in location ACGL of the MST (refer to section 2), if one exists for the respective device. If an IQFT exists, it is rebuilt on each level 0 deadstart. When all files on a device have been requeued, the IQFT is released. Only when there are inactive files left on a device because of the queue selection does the IQFT file exist.

As REC recovers each device, messages are issued for devices found with queues. The requeueing of queues is performed by routines QREC and QFM.

DAYFILE RECOVERY

A dayfile may be present on a device in one of three states: active, inactive, or permanent file. An active dayfile contains a first track pointer in the MST (location DULL) for recovery purposes. An inactive dayfile is a dayfile recovered on a device but not made active because a different dayfile was also recovered. Word DULL of the MST is formatted as follows.



Active and inactive dayfiles have first track pointers set in the MST which serve the same function for device recovery as the preserved file bit. A device may contain only one active or inactive dayfile of each type.

A utility program may be used to terminate a dayfile. The termination of an active dayfile makes the dayfile a permanent file and starts a new active dayfile. Termination of an inactive dayfile results in clearing the MST pointers and making the dayfile a permanent file.

RECOVERY PROCESSING

The first dayfile, account, and error log track assigned by REC is held in that device's DULL MST word. The EOI sector written by 1DD contains the first track to assure a correct BOI/EOI pair for validation by RMS on dayfile recoveries. SET passes residence information via the dayfile buffer pointers to RMS for processing the preserved dayfiles.

If an active dayfile exists on a device, it is preserved unless the device is initialized. Either an INITIALIZE, DF or full initialize prevents preservation. Once all dayfiles are preserved, the one with the latest date is made the active dayfile. If no dayfiles are preserved, the first system device is selected for residence unless directives to SET specify otherwise.

RMS completes the disk chain by reading the dayfile starting at the TRT EOI and continuing to the disk file EOI. This is done if the file is to be preserved in order to set the correct last sector written. If in the process of computing the TRT linkage a mismatch between the dayfile and TRT reservations is discovered, the recovery of the dayfile is discontinued and the file space beyond this point is lost.

REC updates the FST for the active system, account, and error log dayfiles by updating the current track and sector if the file is preserved. If the file is initialized, the track chain is dropped by RMS. If the file is preserved, the system sector is rewritten to indicate last recovery date and time, together with the file size.

The file length is utilized for dayfile dumping to tape options. The file length can be used by dump utilities as an indication of where to resume dumping on subsequent calls. Dayfiles not preserved as active dayfiles remain protected until an initialization of that device is performed.

EQUIPMENT SECTION

If no dayfiles are recovered, the following CMRDECK entries can be entered to select the equipment residence of the dayfiles. This overrides the default of the first system device.

```
DAYFILE = eq,bl.
```

ACCOUNT = eq,bl.

ERRLOG = eq,bl.

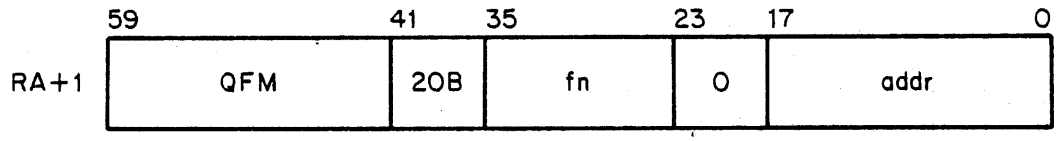
In the preceding entries, eq is the equipment number and bl is the CM buffer length.

This allows a site to spread dayfiles across several devices. The only restriction to dayfile residence is that it must be a mass storage, nonremovable device. The eq entry is ignored if a preserved dayfile is detected during recovery of the system, but the bl entry is always used.

QUEUE FILE MANAGER (QFM)

QFM is a PP program which performs many different tasks for queue management programs such as QREC and QMOVE. QFM is the function processor for all requeueing options, loading and dumping queues, and initialization of queues. As QFM reloads queues, it issues identifying messages to the account dayfile.

The format of the RA+1 call to QFM is as follows.



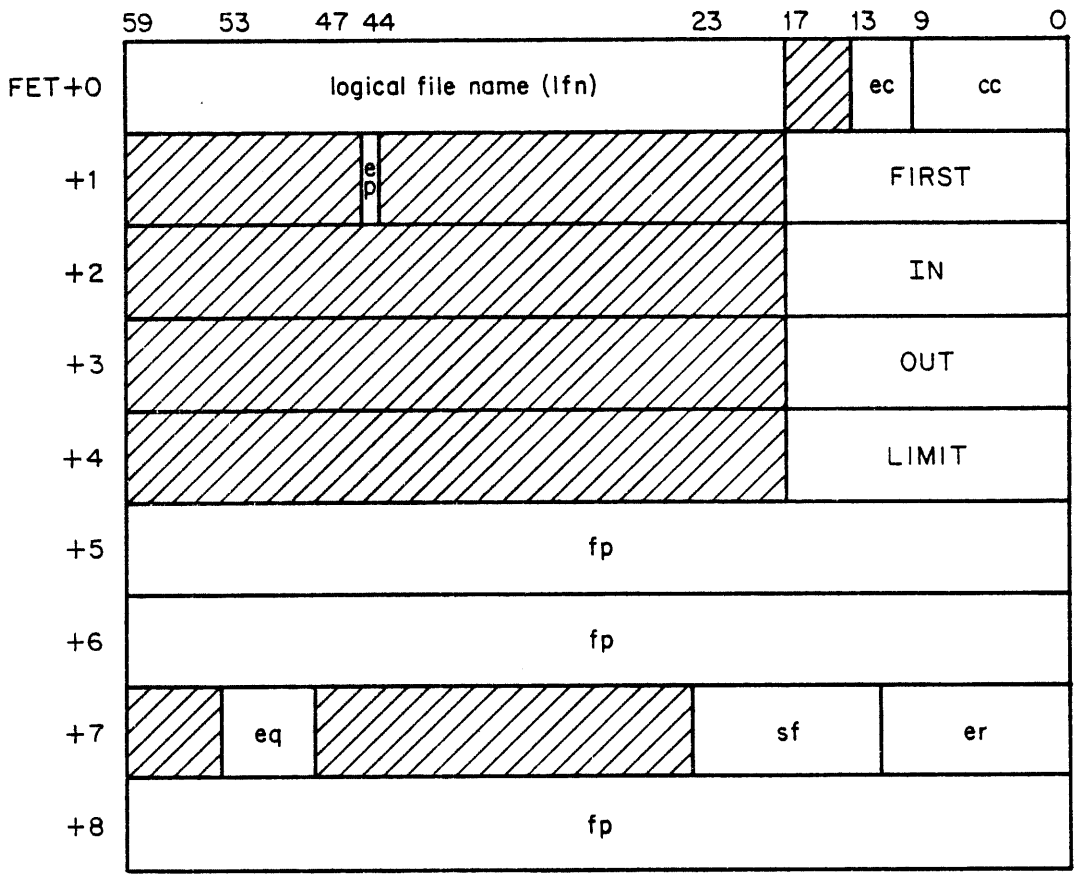
fn Function code:

Code	Symbol	Description
1	ATQF	Attach preserved file (SYOT only)
2	DTQF	Detach preserved file (SYOT only)

3	PGQF	Purge preserved file (SYOT only)
4	STQF	Set IQFT file (SSY= required)
5	INQF	Initialize IQFT file (SSJ= required)
6	RQLF	Requeue FNT/FST list (SSJ= required)
7	RLLF	Release FNT/FST list (SSJ= required)
10	DEQF	Dequeue FNT/FST (SSJ= required)
11	AQFF	Attach queued file (SSJ= required)
12	QRSF	Read system sector (SSJ= required)
13	AIQF	Attach inactive queued file (SSJ= required)
14	RIQF	Requeue inactive queued file (SSJ= required)
15	SRRF	Set rerun protect bit (not TXOT)
16	CRRF	Clear rerun (not TXOT)
17	RIFF	Release file to input queue
20	ASFF	Assign file to queue device

addr Address of the FET

The FET for QFM is formatted as follows.



lfn Logical file name.

ec Error code return (bits 13 through 10):

Code	Symbol	Description
1	FNFE	File not found.
2	FAIE	File already interlocked.
3	TASE	IQFT track already assigned.
4	FTHE	FNT threshold reached.
5	INSE	Invalid system sector.
6	RMSE	RMS error.
7	RRAE	File already protected.

10	NRAE	File already unprotected.
11	INFE	No input file.
12	RDVE	Removable device ignored (QREC).
13	SDVE	Shared device ignored (QREC).

cc Completion code (bits 9 through 0).

ep Error processing bit (bit 44). If set, error processing is selected.

fp Function parameters. (Refer to entry conditions for particular function for required format).

eq Equipment.

sf Subfunction:

Code	Symbol	Description
1	SDAY	System dayfile byte
2	ACCF	Account dayfile byte
3	ERLF	Error log dayfile byte
4	IQFT	IQFT byte

er Mass storage error code.

QFM is organized as follows.

- QFM main program, resident subroutines, common decks, resident function processors
- Overlay 3QA - QFM error processor
- Overlay 3QB - Initialize IQFT file

- Overlay 3QC - Requeue/Release FNT list
- Overlay 3QD - Dequeue/Attach processors

QUEUE FILE SUPERVISOR (QFSP)

The queue file supervisor (CPU program) is the initial processor for the queue/dayfile manipulation utilities. It provides the operator with the list of options which are available for queue/dayfile processing. The input to QFSP may be either through the K display, control statements, or directive input file (refer to the NOS System Maintenance Reference Manual for more information). All utilities are, however, not callable via the K display.

The following utilities are called via QFSP.

Utility	Description
QDUMP	Dump I/O queue files
LDLIST	List the queued files on the dump
QLOAD	Load I/O queue files
QMOVE	Move I/O queues from one mass storage device to another
QREC	Deactivate or activate selected I/O queue files
QLIST	List selected inactive I/O queue files
DFTERM	Terminate an active or inactive dayfile and retain it as a direct access permanent file
DFLIST	List all dayfiles which have been made permanent by DFTERM
QALTER	Alter or purge selected queued files
FNTLIST	List selected active queued files

NOTE

In the following discussions of queue utilities, refer to the NOS System Maintenance Reference Manual for a complete description of parameters and available options.

The following tables are built by QFSP.

- TARG - Table of processed arguments.
- TEQP - Table of mass storage equipment.
- TSDM - Table of secondary device masks.

The following words are set by QFSP.

- FNTA - FNT pointer.
- DFPA - Dayfile pointer.

Upon exiting from QFSP utility overlays, the following registers contain the indicated information.

(X2) = 0, set primary right screen K display

(X2) < 0, do not change right screen K display

(X2) > 0, FWA of new right screen K display

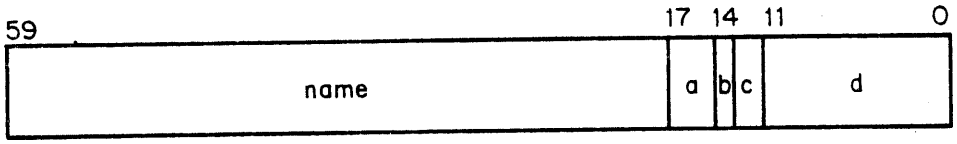
(X5) = FWA of K-display message (message must be 4 words long)

(X5) = 0, no K-display message

(X5) < 0, complement of FWA of K display (do not change left screen K display)

QDUMP/QLOAD UTILITY CONTROL WORDS

Each queued file and the associated data for the file resides on the dump file as a logical record. Each dump session resides on the dump file as a logical file, allowing multiple dumps on the same dump file. Each block of data is preceded by a control word in the following format.



name — Name of queued file if data, or dump file if block is dump header (7 characters maximum).

a — Logical content of block:

- 0 — Data
- 1 — Route block (5 words plus checksum in sixth word which is the arithmetic sum of the population counts of the 5 words in the route block)
- 2 — Dayfile (pre-dayfile or regular output dayfile)
- 3 — NOS dependent data
- 4 — NOS/BE dependent data
- 5 — SCOPE 2 dependent data
- 6 — Dump header (4 words, words 1 and 2 containing an alphanumeric description of the dump, word 3 the date, and word 4 the time)
- 7 — Errors (d field is number of errors encountered)

b — Operating system dependent data (bit 14). For NOS (a=3):

- 0 — System sector
- 1 — End of volume

c Structure of data (bits 13 through 12):

0 Full block of data

1 EOR (word count is data words plus one word which contains the level number)

2 EOF

d Number of words in block.

QFSP is organized as follows.

- QFSP main area (tables, directive input processor, directive processor routines, command processor, subroutines, default values for parameters, and K-display area.
- QFSP overlay area
  - QDUMP overlay
  - QLOAD/LDLIST overlay
  - QMOVE overlay
  - QREC/QLIST overlay
  - DFTERM/DFLIST overlay

QUEUE RECOVERY (QREC) UTILITY

The QREC utility provides the capability to deactivate or activate selected I/O queue files. QREC deactivates an I/O queue file by removing its entry from the FNT and creating a corresponding entry in the IQFT file. An IQFT file exists on each mass storage device containing inactive I/O queue files. I/O queue files recovered across a level 0 deadstart are also inactive and are not processed by the system until activated by QREC. Typically, the IPRDECK contains a call to QREC and queue files recovered across level 0 deadstart are activated automatically without operator intervention. An inactive queue

file is activated or requeued by removing its entry from the IQFT file and making a corresponding entry in the FNT. QREC also provides the capability to purge selected inactive queue files.

This utility is callable either via the queued file supervisor (QFSP) or control statement.

QREC options include the following.

- Requeue specified files and purge others. The IQFT is searched for the specified files, the files are requeued, and the remainder of the files in the IQFT are purged.
- Requeue specified files and ignore others. The IQFT is searched for the specified files, the files are requeued, and the remainder of the files in the IQFT are not affected.
- Purge specified files and ignore others. The IQFT is searched for the specified files, the files are purged from the system, and the remainder of the files in the IQFT are not affected.
- Dequeue specified files and ignore others. The FNT/FST is searched for the specified files. The files are dequeued and added to the IQFT and the remainder of the files in the FNT/FST are not affected.

Entry conditions for QREC include:

- TARA contains FWA of the parameter table.
- FNT contains FNTP word from central memory resident.
- TEQA contains mass storage equipment table.

QLIST UTILITY

The QLIST utility provides a listing of selected I/O queue files. This list may include all inactive queue files in the system or a selected subset based on options specified when calling the utility.

This utility is part of the QREC overlay. It is called by control statement only.

QLIST accepts all of the QREC parameters except the OP parameter. The entry conditions are the same as those for QREC.

QMOVE UTILITY

QMOVE is a utility program that moves queue files from one mass storage device to another and provides a list of all files moved with information relative to each. QMOVE may be called either via the console (QFSP) or control statement.

QMOVE options include:

- Leave queue files as active files.
- Leave queue files as inactive files.

The entry conditions for QMOVE are the same as those for QREC.

QLOAD UTILITY

QLOAD processes the dump tapes generated by QDUMP or other utilities using the same format. QLOAD can selectively load the I/O queues on these dump files. It creates the file on the specified device and writes the system sector. QLOAD may be called either via QFSP or by control statement.

QLOAD provides the following options.

- Load and inactivate.
- Load and activate.

The entry conditions for QLOAD are the same as QREC.

LDLIST UTILITY

LDLIST provides a list of the queued files on the dump file. It can be called only by control statement. The LDLIST options are the same as for QLOAD and entry conditions the same as QREC.

QDUMP UTILITY

The QDUMP utility dumps selected I/O queue files from a single device, a family of devices, or all devices on the system. These queue files can be dumped either to tape or mass storage. When active queue files are dumped, the FNT is searched to obtain the proper file. The IQFT is searched when inactive queues are dumped. QDUMP also provides a listing of all files dumped. This listing includes information about each file. QDUMP may be called either via QFSP or by control statement.

QDUMP options include:

- Dump only active files.
- Dump only inactive files.
- Dump all I/O files.

Entry conditions for QDUMP are the same as those for QREC.

DFTERM UTILITY

The DFTERM utility terminates an active or inactive dayfile and retains it as a direct access permanent file for later interrogation or processing. When an active dayfile (the current system, account, or error log dayfile) is terminated, information in the central memory buffer for that dayfile is written to mass storage to be included with the permanent file and a new active dayfile is started. The new dayfile may reside on the same device or a new device may be specified.

Terminating an inactive dayfile has no effect on the currently active dayfiles. Inactive dayfiles are not used by the system. Furthermore, the presence of an inactive dayfile in the system is possible only under unusual conditions. For example, assume that the system is deadstarted and the device which previously contained the account dayfile is not in the system (OFF). The new account dayfile is then started on another device. Two devices now contain account dayfiles. If both devices are turned on when the system is next deadstarted, two account dayfiles are recovered. The most recent account dayfile is made active and is used by the system. The remaining account dayfile is made inactive.

DFTERM may be called either via QFSP or control statement.

The DFTERM options include:

- Terminate active dayfile.
- Terminate inactive dayfile.

Entry conditions for DFTERM are as follows.

- TARA contains FWA of the parameter table.
- TEQA contains FWA of the mass storage equipment table.
- TSDA contains FWA of the secondary device mask table.

DFLIST UTILITY

The DFLIST utility provides a printer listing of all dayfiles which have been made permanent by the DFTERM utility. DFLIST is callable only via control statement. There are no DFLIST options. No parameters are permitted on the control statement. DFLIST entry conditions are the same as for DFTERM.

FNTLIST UTILITY

The FNTLIST utility provides a list of selected active I/O queues. These lists may include all active queued files or a selected subset based on options specified on the control statement, K display, or directive input file.

QALTER UTILITY

The QALTER utility provides the capability to change routing information associated with active queued files. QALTER may also purge active queued files. QALTER generates a list of files that were processed. This utility is called either via the console (QFSP) or control statement.

The entry conditions for QALTER are the same as for QREC. This utility is part of the FNTLIST overlay.

This section describes the accounting and validation facilities provided by NOS including the following.

- Account dayfile
- SRU algorithm
- Validation files

The external characteristics of the accounting and validation facilities, and the utilities that interface with them, are described in the NOS System Maintenance Reference Manual.

ACCOUNT DAYFILE

The account dayfile provides a history of system and resource usage. This dayfile serves the following two purposes.

- Provides the information necessary to properly bill the users of the system.
- Provides the installation with the information necessary to analyze the use of the system or any specific part of it (for example, magnetic tape usage).

A standardized message format is provided to aid account dayfile analysis. All account dayfile messages have the following general format.

hh.mm.ss. jobnameo, geac, info.

hh.mm.ss.	Current time, beginning in column 2. The system appends this field to the beginning of any message issued to the account dayfile.
jobname	The name of the job causing the entry of this message into the account dayfile. The system appends this field to the beginning of the message along with the time.
o	A single character in column 18 which describes the origin type of the job. This field is automatically appended to the jobname and followed with a period.

geac — A unique four character message identifier which defines the particular activity indentified. This field begins in column 21 and ends with a comma followed by a space. The g identifies the information group, the e identifies the event that caused the message to be entered into the account dayfile, and ac identifies the activity being recorded.

info — Information that gives further detail to the activity identified by geac. This field begins in column 27 and ends with a period. If a field is not used, it appears as a comma and space unless it is the last field in the message; then it does not appear.

The individual message groups are described in the NOS System Maintenance Reference Manual. Each message is contained in the program that issues it to the account dayfile; NOS does not have a common routine for issuing all accounting messages. However, the system symbols ACFN and AJNN are used in conjunction with calls to PP resident routine DFM to issue the message to the account dayfile. Thus, through KRONREF, the routines issuing accounting messages may be identified.

SRU ALGORITHM

The basic accounting unit of NOS is the system resource unit (SRU). The SRU is a measurement of the resources used by a job or terminal session. The SRU algorithm combines measurements of the following resources into a single unit.

- Central memory field length
- ECS field length
- CPU time
- Mass storage usage
- Magnetic tape usage
- Permanent file usage

The SRU calculation is dynamic; that is, each time additional amounts of the above resources are utilized by the job or session, the SRU value is updated.

The SRU algorithm is:

$$SRU = (M1(CP + M2 * IO) + M3(CP + IO)CM + M4(CP + IO)EC) + AD$$

Each parameter is defined as follows.

Parameter	Description
CP	Central processor usage in milliunits as determined by the formula: CP = S0 * CP0 + S1 * CP1 CP0 — CPU 0 accumulated time in milliseconds CP1 — CPU 1 accumulated time in milliseconds S0,S1 — Multipliers used to normalize CP time when the system is running in a dual CPU machine or the system is used on different mainframes at the same site.
IO	A measure of the accumulated input/output system activity for a user. This parameter, expressed in milliunits, is defined by the formula: IO = S2 * MS + S3 * MT + S4 *PF MS — Mass Storage Activity MT — Magnetic tape activity PF — Permanent file activity S2, S3, S4 — Multipliers used to weight MS, MT and PF activity against each other
CM	Central memory field length in words/100B
EC	ECS field length expressed in tracks
M1	Multiplier to scale overall SRU value
M2	Multiplier used to weight I/O activity against CP time, CM field length and ECS field length.
M3	Multiplier used to weight CM field length against CP time, ECS field length and I/O activity.
M4	Multiplier used to weight ECS field length against CP time, I/O activity and CM field length.
AD	Adder applied to the SRU value as an accumulation of individual unit charges (such as utilized at account block initiation and SRU accumulation enabling/disabling).

SRU multiplier value ranges and default values are defined in common deck COMSSRU. The M1 through M4 multipliers and the initial AD adder may vary according to charge and project numbers and may be changed by a CHARGE statement during the job or session. The S0 and S1 multiplier defaults may be overridden by

the IPRDECK installation parameter CPM. The S2, S3, and S4 multipliers, however, are fixed at assembly time in COMSSRU, but they may be modified to fix the needs of the installation. The NOS System Maintenance Reference Manual contains the information necessary to adjust SRU multipliers and how to specify them for individual charge and project numbers.

The account block defines the accumulation of SRUs and other resource usage from CHARGE statement to CHARGE statement or end of job or session.

The SRU algorithm is contained within CPUMTR and the SRU multipliers and accumulators are contained in the job's control point area. CPUMTR updates these accumulators and detects accumulator overflow conditions. The following CPUMTR subroutines perform some portion of the SRU algorithm.

AAD ROUTINE

AAD applies the adder increment to SRU accumulator by the formula:

```
SRU = SRU + AD
```

AD is supplied through the UADM monitor function.

AIO ROUTINE

AIO applies the IO increment to SRU accumulator, by the formula:

```
IO = S2*MS + S3*MT + S4*PF
SRU = SRU + IOM*IO
```

IOM is determined by CPUMTR routine SRU and S2, S3, and S4 are obtained from common deck COMSSRU. AIO is called as the result of UADM and TIOM monitor function.

CPT ROUTINE

CPT adds the CP time increment to the SRU accumulator by the formula:

```
CP = S0*CP0 + S1*CP1
SRU = SRU + CPM*CP
```

where CPM is determined by CPUMTR routine SRU and S0 and S1 are in common deck COMSSRU or overridden by installation parameter CPM. CPT is called whenever the control point is given or relinquishes the CPU or when the user program asks for the CP time through the TIM RA+1 monitor call.

SRU ROUTINE

SRU calculates the SRU multipliers CPM and IOM that are used by routines AIO and CPT. The following formulas show the derivation of these two multipliers.

```
SRU = M1(CP+M2*IO+M3(CP+IO)CM + M4(CP+IO)EC + AD
    = M1(1+M3*CM+M4*EC)CP + M1(M2+M3*CM+M4*EC)IO + AD
    = (M1+M1*M3*CM+M1*M4*EC)CP +
      (M1*M2+M1*M3*CM+M1*M4*EC)IO + AD
    = CPM*CP + IOM*CP + AD
```

thus

```
CPM = M1 + M1*M3*CM + M1*M4*EC
IOM = M1*M2 + M1*M3*CM + M1*M4*EC
```

M1, M3, M3, and M4 are supplied through the user's charge and project number, unless defaults are being used. SRU is called whenever a change in the job's central memory or ECS field length occurs or when new M1 through M4 values are specified.

ACCOUNTING CPUMTR FUNCTIONS

The following paragraphs describe the CPUMTR functions that are used for accounting purposes. The subfunctions of these monitor functions are defined in common deck COMSCPS.

ACTM - ACCOUNTING FUNCTIONS

ACTM performs the following accounting activities.

ABBF (1) Function

This function begins an account block by inserting new multipliers in the control point area and applying the initial adder AD to the SRU accumulator. The multipliers CPM and IOM are calculated.

This function is used by CPM to begin the account block on the first CHARGE statement and by 1AJ to initialize the job's control point area with default multiplier values.

ABSF (2) Function

The multipliers CPM and IOM are calculated in this function using the current CM and ECS field lengths.

Routine 1RI uses this function prior to restarting the job.

ABCF (3) Function

This function changes or ends the account block by clearing the SRU accumulator, replacing the multipliers in the control point area, and applying the initial adder AD to the SRU accumulator. The multipliers CPM and IOM are calculated.

CPM uses this function to set SRU multipliers for secondary CHARGE statements.

ABEF (4) Function

Elapsed SRUs (new-old) are calculated and program mode is entered with this function for the conversion by RDC. If elapsed SRUs are less than MDSR (defined in COMSSRU), zero is returned.

Routine 1RO uses this function when completing a time-sharing job step.

ABVF (5) Function

With this function the accumulators are unpacked and stored one per word in the message buffer (MB) and program mode is entered for conversion by RDC. If the total SRUs is less than or equal to MCSR (defined in COMSSRU), MCSR is returned as the SRU accumulator.

This function is used by CPM, 1CJ and 1TA to output the accumulators at the end of an account block whether it is caused by an end of job/session or new CHARGE entry.

ABIF (6) Function

The SRU accumulator value is first converted to an integer number and then integer addition or subtraction is performed to increment or decrement the SRU accumulator with this function. If the converted accumulator value is less than 1, 1 is used.

This function is used by 1AJ in processing SRU limit errors and OAU to increment SRU accumulators when updating the PROFILa file.

RLMM - REQUEST LIMIT

RLMM sets SRU and time limits for individual job steps. RLMM is also used to clear accumulator overflow flags. The functions performed by RLMM are as follows:

Code	Function	Description
0	RLCO	Clear overflow flags
1	RLIT	Increment job step time limit
2	RLIs	Increment job step SRU limit
3	RLJS	Start job step
4	RLTL	Set time limit
5	RLSL	Set SRU limit

TIOM - TAPE I/O PROCESSOR

TIOM updates the accounting accumulators due to tape I/O activity. An increment is specified in the TIOM call to be added to the MT accumulator. This increment is applied to the SRU accumulator through a call to subroutine AIO. 1MT is the only routine in the system that issues TIOM calls.

UADM - UPDATE CONTROL POINT AREA

The UADM updates various fields in the control point area including the ACLW (counting limits) and the IO and SRU accumulators. UADM performs the following functions.

Code	Function	Description
0	LICS	Increment low core field
2	LIOS	Increment low core field by one
4	LDOS	Decrement low core field by one
6	LDCS	Decrement low core field
10	CICS	Increment control point field
12	CIOS	Increment control point field by one
14	CDOS	Decrement control point field by one
16	CDCS	Decrement control point field
20	AISS	Increment MS or PF accumulators; calls AIO to apply the MS or PF usage increment to the SRU accumulator.
30	AIAD	Increment AD accumulator; calls AAD to apply the adder increment to the SRU accumulator.

Functions 0, 2, 4, 6, 10, 12, 14, and 16 are used by PP routines CIO, DSP, LFM, OUT, PFM, ODF, 1AJ, and 1MA to adjust the counting limits maintained in ACLW.

Function 20 is used by 1AJ to charge for program loading, CIO to charge for I/O activity, PFM to charge for its activity, and to charge for I/O associated with message handling.

Function 30 is used by CPM (function 53) when enabling SRU accumulation; an incremental SRU charge may be specified.

VALIDATION FILES

The system validation and project profile files are used to validate user access to the system. Validation defines and controls the following:

- Who can use the system
- What resources can be used (hardware and software)
- To what extent these resources may be used.

Every user of the system must have a valid user number (if VALIDATION is enabled). In a batch environment this means that the statement following the job statement must be a USER statement. This causes the routine ACCFAM to be loaded to process the USER statement, verifying that the user number exists. If the user number is valid, ACCFAM sets up the validation information for entry into control point area words UIDW, ALMW, ACLW, and AACW. Subsequent job operations are restricted by these validation limits.

If the user is required to be further validated for accounting purposes, a CHARGE statement must be the next statement in the job. The CHARGE statement causes the CHARGE routine to be loaded to validate the charge and project numbers to which the user is charging his activities.

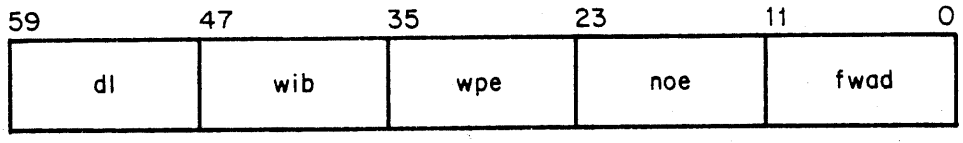
Thus, the validation procedure allows the system to:

- Determine if a user is allowed to use the system.
- Charge the user for his resource usage.
- Restrict the user to certain resource usage, including denying access to the system when the project's SRU limit has been reached.
- Maintain permanent files for the user and control access and security of them by mapping the user's user number into a specific user index.

TREE-STRUCTURE FILES

The validation file, VALIDUs, and the project profile file, PROFILa, are both tree-structured files. System routine SFS (special system file supervisor) and its common deck COMSSFS (special system file macros and equivalences) provide a function processor for common routines that perform basic table management, data manipulation and I/O processing for tree-structured special system files. SFS is a (01,00) overlay that is loaded by the MODVAL and PROFILE utilities that manipulate the validation and project profile files. SFS is designed to process tree-structured files of a given format. The functions provided by SFS are designed to process any number of levels in the tree-structure; however, table space is only allocated for a three-level tree-structured file, three directory levels and one data level).

The first word of each record on the file is a control word containing sufficient information to describe the data within the block. The following shows the format of the control word used in the directory level blocks. The second word normally contains information used by the processing programs, usually the creation and last modification dates. The third word contains linkage (random address) to the next logical block on that level, if one is present. The remaining words in the block are directory entries for directory level records.



dl	Data level
wib	Words in block
wpe	Words per entry
noe	Number of entries in block
fwad	First word address of data

A total of 63 words (60 words of entries plus three control words) can be used in each block in the directory levels. For the data level, the control word should be compatible with the control words for the directory levels, but this is not mandatory. The remainder of the block can be any length and format desired. Because of this flexible format, the processor program must perform its own I/O for the data-level block. However, if the data level is constructed similar to the directory level records, SFS functions can be used to perform the I/O. The information in all levels is maintained in a collated sequence.

The 0 and 1 directory levels correspond to the primary level of the tree. The entries in the 0-level consist of the first entry (and corresponding random address) of each level-1 block. All primary entries can be found in the level-1 directory. This method enables a quick access to a given primary entry. The first sector in the file is defined to be the first level-0 directory block which is linked to the next level-0 block. Except for the primary level, there exists one directory level for each tree level terminating with the data level.

COMSSFS

COMSSFS provides the communication between SFS and the processor program, SFS processes all directory level blocks but the data blocks must be handled by the processor program.

The following functions are defined in COMSSFS for use with SFS.

Code	Function	Description
		Input processing functions
0	ASCT	Assemble characters
1	SCIT	Scan for code identifier
		File read functions
2	ANBT	Add next block
3	CCWT	Crack control word
4	SBTT	Set block in table
5	SPBT	Set primary block
6	PNAT	Pick next address
7	PNET	Pick next entry
		File manipulation functions
10	DZET	Delete zero entries
11	MWST	Multiple word search
12	SDFT	Set data in field
13	SFTT	Space fill table
14	STBT	Sort table
		File write functions
15	BLDT	Build directory
16	RBAT	Reset block address
17	UDDT	Update directory
20	WTBT	Write table
21	MAXT	Maximum function number

MODVAL AND VALIDATION FILES

MODVAL is a system utility that is used to create and maintain the system validation files VALIDUs and VALINDs. These two files are both direct access permanent files catalogued under the system user index, 377777B. VALIDUs is accessed as a fast attach file by most of its users while VALINDs is accessed by MODVAL as a normal direct access file.

MODVAL creates or updates the validation files either by reading a file of input data or by accepting commands directly from the operator's console via the K display. When MODVAL is operating via the K display, the update (OP=U) mode is assumed. Changes made are available to the system for the next USER statement as soon as the operations performed on a user number are ended via the K-display typein K.END, since update mode works with the VALIDUs and VALINDs files directly. Refer to the NOS System Maintenance Reference Manual for MODVAL options, parameters, and examples of usage.

VALINDs FILE

VALINDs contains a record of which user indices have been assigned. It consists of 4210B central memory (60-bit) words, each bit representing one of the 377700B (AUIMX) user indices available in the system. If the bit is set to 0, then the user index is available for assignment to a user number. The value AUIMX is the maximum user index for legal login and USER statements and is defined in common deck COMSACC. User indices greater than AUIMX represent special user numbers and as such are not automaticaly assigned to user numbers by MODVAL; therefore, user indices above AUIMX are not represented in the VALINDs file.

VALIDUs FILE

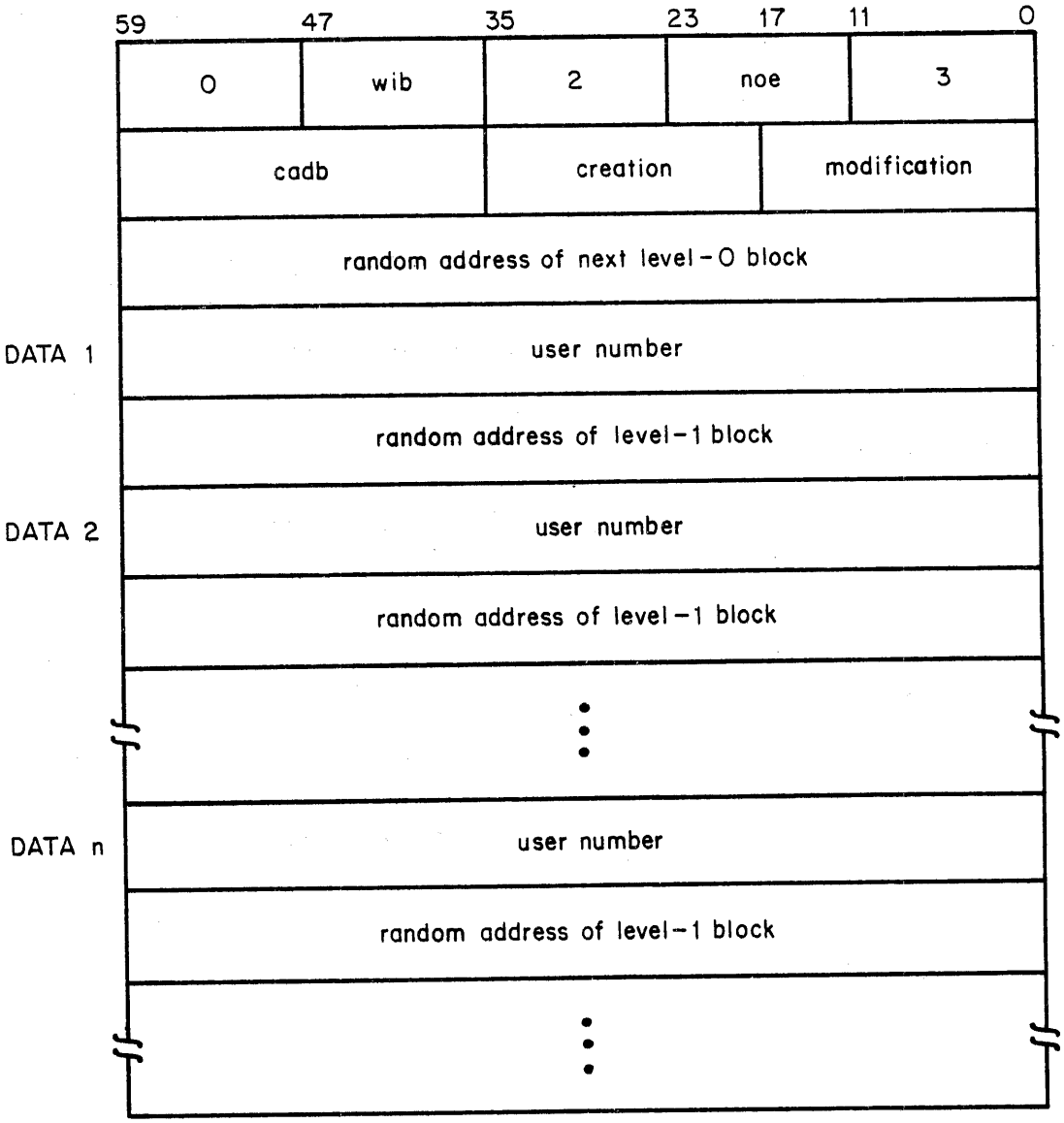
The validation file, VALIDUs, contains the user validation information which, when referenced through a USER statement, specifies the user's access permissions and limits.

VALIDUs is a tree-structure file indexed by user numbers. The data in each level is arranged in alphabetical order with the lowest item first.

The zero-level contains a fixed amount of data concerning the history of the file, and the first user number (and corresponding random address) in each primary level-1 block.

The next level (level-1 or primary level) of the tree contains all validated user numbers with corresponding random addresses pointing to the level-2 blocks. All level-1 blocks are less than one PRU (64 words) in length and may be linked to other level-1 blocks if the data that should be contained in one block exceeds one PRU.

The next level (level-2) of the tree contains all the user validation information associated with the particular user number. Level-2 blocks are one PRU (64 words) in length and are constructed such that four user validation blocks plus block header information are contained in one level-2 block. Figure 20-1 illustrates the VALIDUs level-0 data block, figure 20-2 shows VALIDUs level-1 data block and figure 20-3 is the VALIDUs level-2 data block. This structure is also defined in common deck COMSACC.



cadb	Random address of available level-2 block, if nonzero
creation	Creation date
modification	Last modification data

Figure 20-1. VALIDUS Level-0 Block

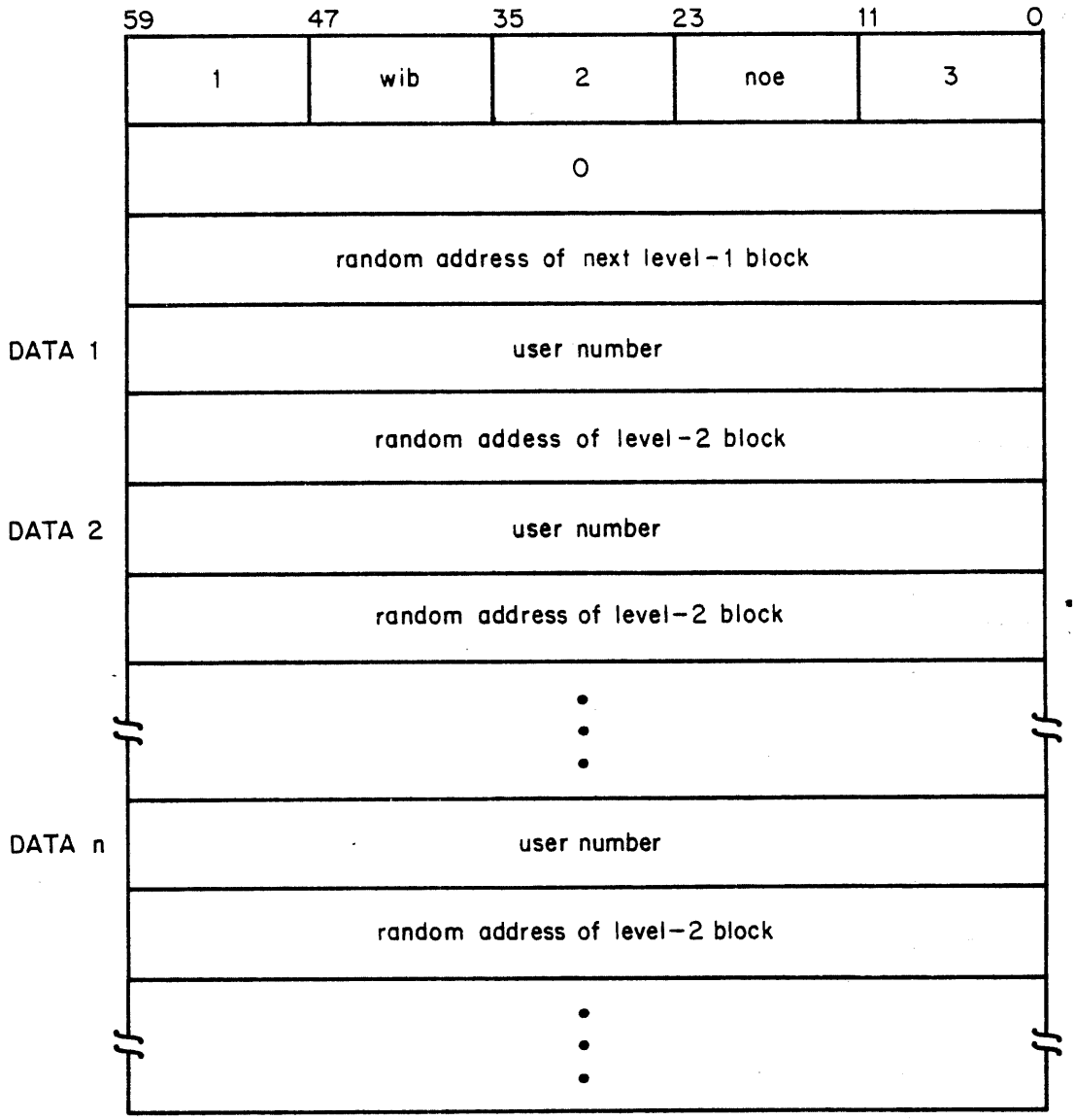


Figure 20-2. VALIDUS Level-1 Block

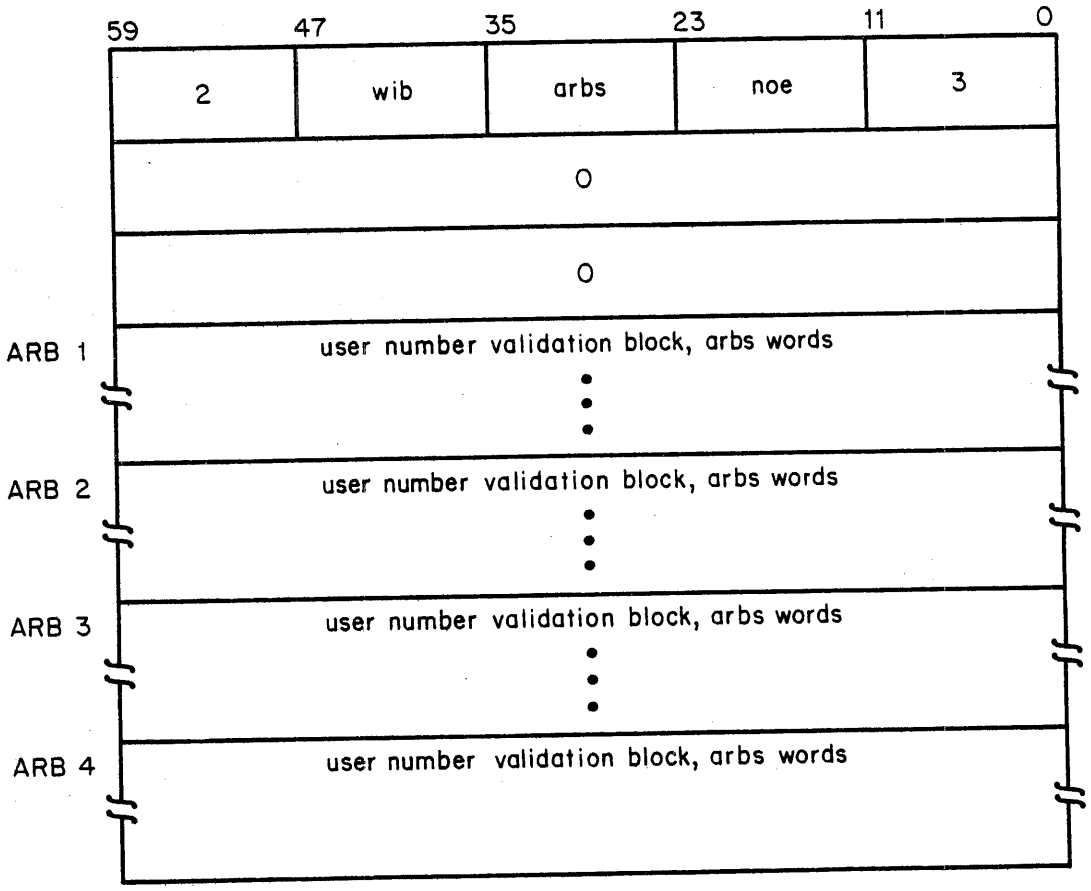
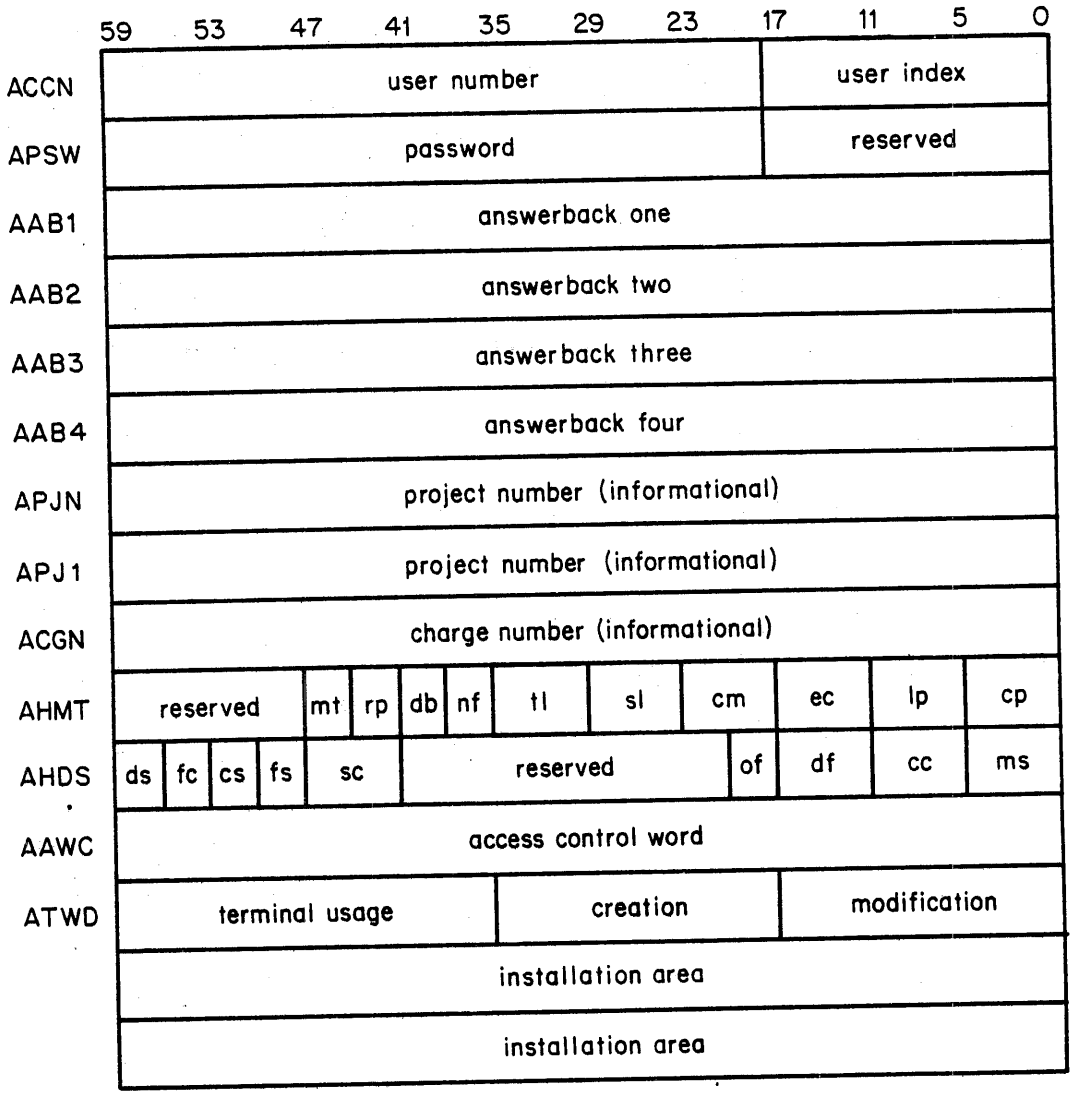


Figure 20-3. VALIDUS Level-2 Data Block

Level-0 blocks are created only during a create (OP=C) or restructure (OP=R) MODVAL run. During updates (OP=U), changes are made only to level-1 and level-2 blocks. If too many user numbers are added to one level-1 block, the information overflows into a new level-1 block. When this happens MODVAL issues the diagnostic LEVEL-1 INDEX BLOCKS LINKED. The validation file should then be restructured so as to eliminate the level-1 block linkage, thus improving the search time for user numbers residing in the linked blocks or for nonexistent user numbers which would have resided in the linked blocks.

USER NUMBER VALIDATION BLOCK

The user number validation block is ARBS words in length. Four of these blocks may be contained in a level-2 block. The format of the user number validation block is shown in figure 20-4.



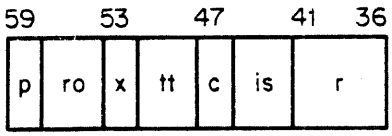
mt Maximum magnetic tapes (3 bits)
rp maximum removable packs (3 bits)
db Deferred batch index (3 bits)
nf Number of local files index (3 bits)
tl Time limit index (6 bits)
sl SRU limit index (6 bits)
cm Central memory FL index (6 bits)
ec ECS FL index (6 bits)
lp Lines printed index (6 bits)
cp Cards punched index (6 bits)

Figure 20-4. User Number Validation Block

The symbol definitions used in AHDS are:

Symbol	Definition
ds	Maximum direct access file size index (3 bits)
fc	Number of indirect access files index (3 bits)
cs	Cumulative indirect access file size index (3 bits)
fs	Maximum indirect access file size index (3 bits)
sc	Security count (6 bits)
of	Disposed output index (3 bits)
df	Dayfile message index (6 bits)
cc	Control statement index (6 bits)
ms	Mass storage PRUs index (6 bits)

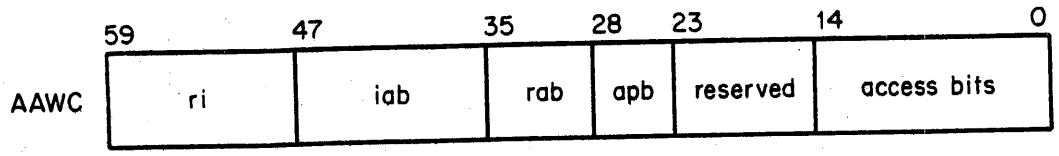
The terminal usage portion of ATWD is defined in as follows.



p	Terminal parity (bit 59)*
ro	Number of rubouts (bits 58 through 54)*
x	Transmission mode (bit 53)
tt	Terminal type (bits 52 through 48)
c	Terminal character set (bit 47)
is	Terminal initial subsystem (bits 46 through 42)
r	Reserved (bits 41 through 36)

*Not applicable to network terminals.

The access control word (AAWC) is defined as follows.



ri Reserved for installation
iab Installation application permission bits
rab Reserved for future applications
apb Application permission bits as follows:

Bit	Application	Definition
24	IAF	Interactive Facility
25	RBF	Remote Batch Facility
26	TAF	Transaction Facility
27	MCS	Message Control System
28	TVF	Terminal Verification Facility

access bits Each bit defined as follows:

Bit	Keyword	Definition
0	CPWC	User can change his password
1	CTPC	User can use ACCESS time-sharing commands
2	CLPF	User can create direct access permanent files
3	CSPF	User can create indirect access permanent files
4	CSOJ	User can have system origin privileges
5	CASF	User can access library type (LIFT) files
6	CAND	User can use nonallocatable equipment
7	CCNR	User can run with CHARGE required
8	CSRP	User can use auxiliary device requests
9	CSTP	User has special transaction privileges
10	CTIM	User desires no timeout at terminal
11	CUCP	User can use system control point (SCP)
12	CSAP	User has special accounting privileges
13	CBIO	BATCHIO subsystem privileges
14	CPRT	Protect ECS privileges

DELETED USER NUMBERS

If a user number is deleted from the system and the user index is returned to the available user indices pool, the permanent files associated with that user index are not automatically purged. These permanent files will become available to a new user who may be assigned to the user index. The permanent files are also still available to those jobs that can specify the user index via the SUI statement.

Normally, new users are assigned user indices sequentially so user index holes would not be assigned unless the particular user index is specified. If a user number is going to be deleted from the system, it is wise for the site analyst to also purge all the permanent files catalogued under the deleted user index; this can easily be accomplished by using the PURGALL command under this user index.

Once the VALINDs bit for a user index becomes set, it remains set even if the corresponding user index is deleted. Only during a MODVAL restructure (OP=R) are the VALINDs bits reset to zero for deleted user indices. If there are several user numbers with the same user index (this is accomplished by the use of the FUI command), the VALINDs bit for the user index remains set during a restructure even if some of the user numbers are deleted. By not resetting VALINDs bits for deleted users, MODVAL can guarantee that no new user number will be assigned a previously assigned (and then deleted) user index, until a MODVAL restructure (OP=R) has been performed. This serves to protect a user's permanent files from being purged should the user accidently be deleted.

When a user number is deleted, all the pointers for that user in level-0 and level-1 blocks are purged. The level-2 entry for this deleted user remains in the level-2 block even though there are no pointers to it. During a MODVAL restructure (OP=R), these level-2 blocks are read and those blocks without level-1 pointers are eliminated, and if no other user number has that corresponding user index, all permanent files catalogued under that user index are purged. No permanent files are automatically purged until a restructure of the validation files is done. Only at that time does MODVAL purge permanent files for those user indices specified as unused (deleted) in the VALIDUs file.

ACCFAM PROGRAM

ACCFAM is the CP program that processes the USER and FAMILY control statement. ACCFAM cracks the USER statement and makes a call to CPM to validate and return the validation limits for the specified user number. CPM validates the user number and obtains the user's validation limits by calling OAV. The validation limits returned are entered into the UIDW, ALMW, ACLW, and AACW words of the control point area by the SSJ= special entry point mechanism when ACCFAM completes.

ACCFAM also sets the name of the family selected by the system origin user via the FAMILY control statement. A CPM function is used to enter the family information into the job's control point area.

ROUTINE OAV

Routine OAV is used by PP routine to locate the user number validation block of a user number.

Routine OAV first finds the validation file for the desired family, using the default family if no family is specified.

The level-0 blocks contained in this fast attach permanent file are searched until the user number in question is greater than a user number found in the level-0 block. When this condition is met, the random address portion of the user number entry in the block is the location of the level-1 block needed. The entries in the level-0 block are structured such that only the first data user number entry is each block need be examined if the searching criterion is not met by the other entries in the block. Thus, once the level-0 search criterion has been met, it is not necessary to search more than one level-1 block (unless linked) and one level-2 block.

The level-1 block is then read and searched until the desired user number is matched with a user number in the level-1 block (or a linked level-1 block). When this match occurs, the random address portion of the level-1 entry is the location of the level-2 block that contains the validation information for this user number.

The level-2 block is then read and searched until the user number matches with an entry in the level-2 block. The address of the validation information is then made available to the caller of OAV along with the user index assigned to the user number. If, during the search, no match is found or one of the random addresses is fraudulent, the user index returned is set to zero to indicate that the user number could not be found. If the user number is found, but the password was not correct or some other error was detected (such as secondary user statements not enabled and not first user statement for non-SYOT jobs), the security count field in the validation file entry is decremented and this level-2 block is rewritten to the validation file.

Routine OAV is called by PP routines CPM, DSP, LFM, PFM, QAC, QFM, VEJ, XSP, OVJ, and 1TA.

Figure 20-5 contains a flow chart of OAV.

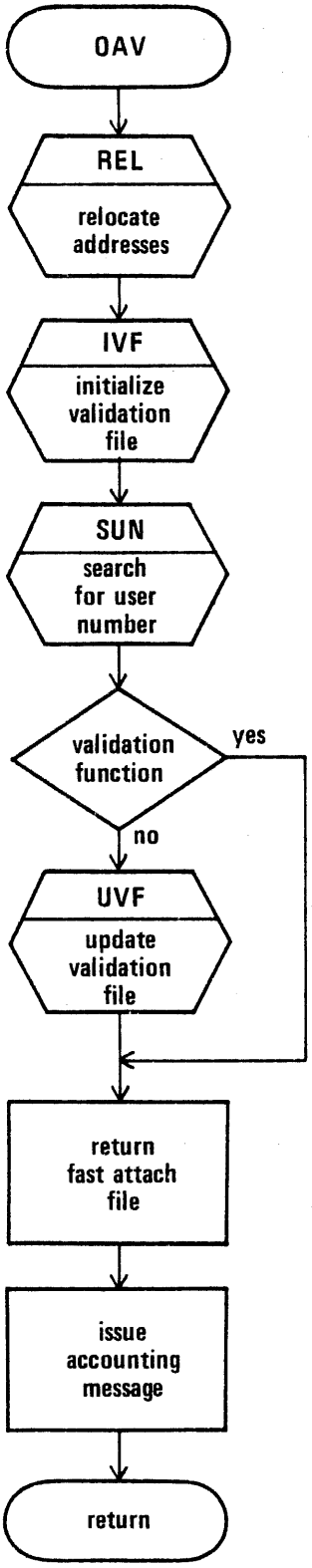


Figure 20-5. Routine OAV

The following paragraphs described the main routines in OAV.

SUN - Search for User Number

Routine SUN is responsible for reading the VALIDUs file searching for a specified user number. Routine SIB (search index block) is called to search the level 0 and level 1 blocks for the user number. SUN either returns the user block or returns a user block not found status.

UVF - Update Validation File

Routine UVF is responsible for decrementing the security count in the user block found by routine UVF. The user block is then rewritten to the VALIDUs file.

IVF - Initialize Validation File

Routine IVF is responsible for finding the correct validation file and interlocking it in the correct file mode. The correct mass storage driver is loaded.

VALIDATION LIMITS

Common decks COMCCVI and COMPCVI convert index values into limit values using concepts called index limit and index table.

The index limit concept determines an upper limit value from an index value by using the index value in a formula. The formula usually has the form:

limit = index * multiplier + default constant

This technique allows many values to be saved in a minimal amount of space. The following resources are limited by the index limit concept:

Routine	Resource
TLI	Time Limit
SLI	SRU limit
LPI	Lines printed
CPI	Cards punched
NFI	Local files
CMI	Central memory field length
ECI	ECS field length
DBI	Number of deferred jobs

The index table concept uses an index value to point to an entry in a table which contains the limit value. The following permanent file controls are managed in the index table manner:

Routine	Resource
FCI	Number of permanent files
DSI	Length of individual direct access file
FSI	Length of individual indirect access file
CSI	Total length of indirect access files

A third method of limiting resource usage is the counting limit method. In this method, a limit value is established and is decremented until it reaches zero, at which point the limit has been reached. This limit value may also be incremented as charged units of the resource are released. The counting limit values are adjusted by the monitor function UADM and are contained in word ACLW of the control point area. The limit value used is determined by the index limit mechanism. The counting limit method is used to control the following:

Routine	Resource
NSU	Mass storage PRUs
CCI	Control statements processed
DFI	Dayfile messages issued
OFI	Output files disposed

The limit values used for magnetic tape and removable pack

resources is the actual number entered; there is no formula or table derivation of these two validation limits.

The limit value derivations contained in the two common decks may be selectively assembled by including an appropriate DEF pseudo instruction prior to the common deck call. The definition of the routine name with a $ appended causes that derivation routine to be assembled. For example, the sequence:

```
CMI$  DEF        1
*CALL  COMCCVI
```

causes only the CMI (central memory field length limit) routine to be assembled. This technique allows all the formulas and tables to be maintained in one common deck of each type without excessive memory requirements in the programs using the deck.

PROFILE AND PROJECT PROFILE FILES

PROFILE is a system utility that is used to create and maintain the system project profile file PROFILa. PROFILa is a private direct access permanent file catalogued under the system user index (377777B) that is normally accessed as a fast attach file.

PROFILa contains the information required to control a user's accounting and access to the system as defined by charge and project numbers, with additional limits on time-of-day and accumulated resource usage. The user is required to supply correct charge and project numbers if the CHARGE not required (CCNR) bit is the user's access word (AACW) is clear. PROFILE also allows the specificaton of a master user for a charge number. This master user is validated to add or delete project numbers, user numbers, and user access information for the specified charge number.

ACCESS TO PROFILA

There are three classifications of access and modification to PROFILa:

- System origin jobs
- Special accounting users (CSAP bit set in AACW)
- Master users

In each case, the former level possesses more capability than the latter.

System origin jobs have complete access to PROFILa, with no restrictions regarding PROFILE options and directives.

Special accounting users from non-system origin jobs have full capabilities on update (OP=U) and inquiry (OP=I) PROFILE runs,

but may not perform create (OP=C), reformat (OP=R), list (OP=L), or source (OP=S) operations.

Master users from non-system origin jobs may only alter values pertaining to charge numbers for which they are the defined master user, such as entering users numbers under the number. Master users may not change any of the charge installation related project number parameters, such as installation accumulators, nor may they change any charge number parameters, such as the SRU multipliers. Refer to the NOS System Maintenance Reference Manual for PROFILE options, parameters, and examples of usage.

PROFILa FILE

The PROFILa file is a 3-level tree-structured file and is manipulated in the same manner as the validation file VALIDUs, Levels 0, 1 and 2 are directory levels and level 3 is the data level. The data in each level is arranged in alphabetical order with the lowest item first.

The level-0 contains a fixed amount of data concerning the history of the file, and the first charge number (and corresponding random address) in each primary level-1 block.

The next level (level-1 or primary level) of the tree contains all validated charge numbers and their master users, with corresponding random addresses pointing to the level-2 blocks. Information pertaining to all projects associated with the charge number, such as SRU multiplier indices, are also contained in the level-1 block.

The level-2 block of the tree contains all valid project numbers for the corresponding charge number. Along with each project number is a random address pointing to the level-3 block.

The level-3 block contains all the project profile information associated with this particular charge number and project number. Two entries are contained in one level-3 block. If more than thirteen user numbers are specified for a given project number, an overflow level-3 block containing only the excess user numbers is linked to the original level-3 block. Figures 20-6 through 20-10 show the structure of the PROFILa file and the contents of each level block; this structure is also defined in the common deck COMSPRO.

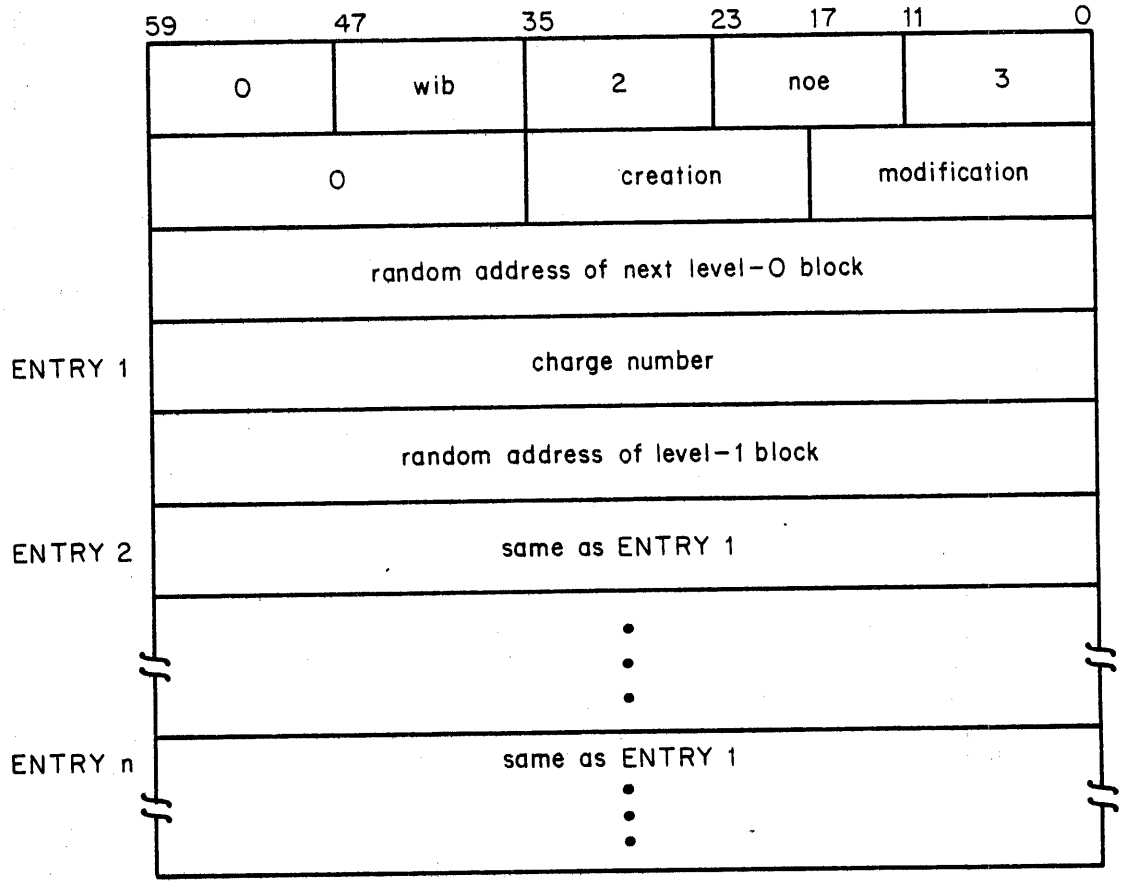
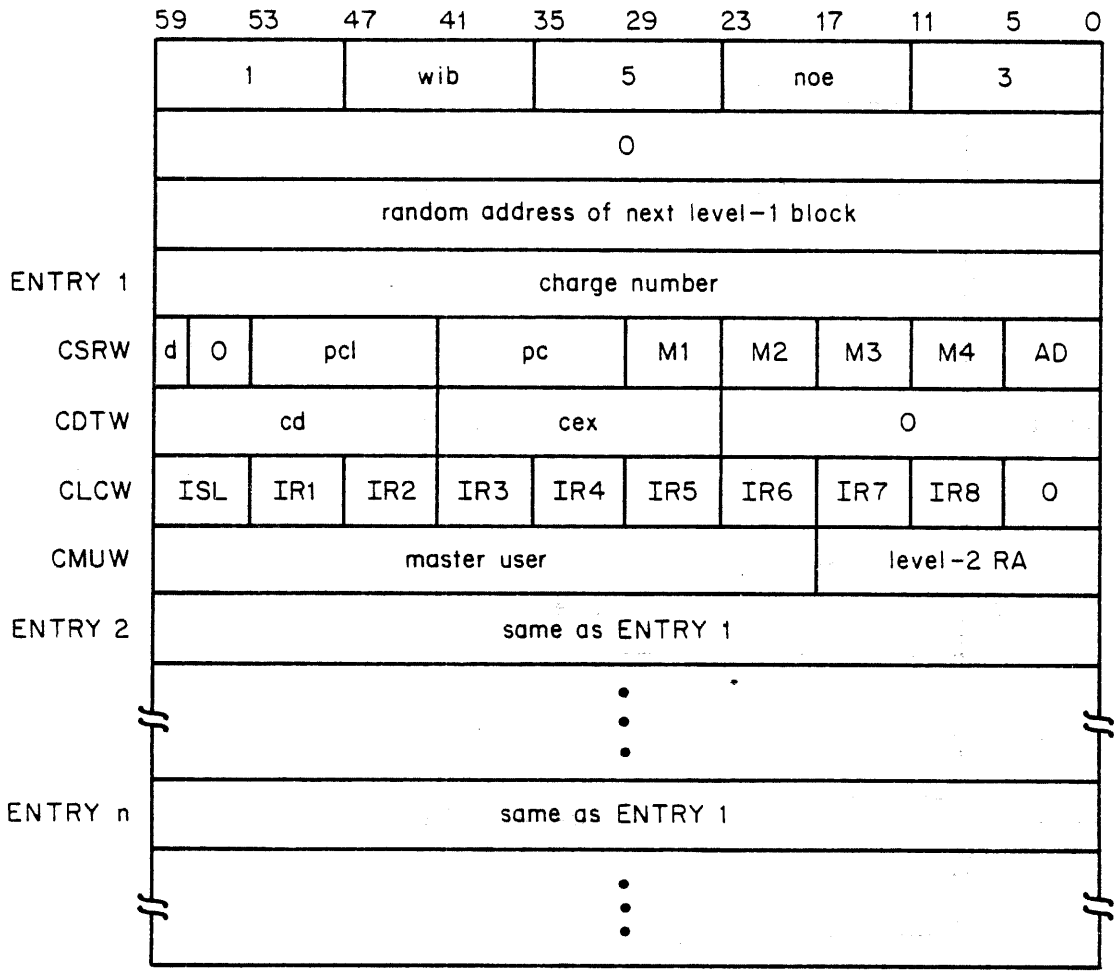
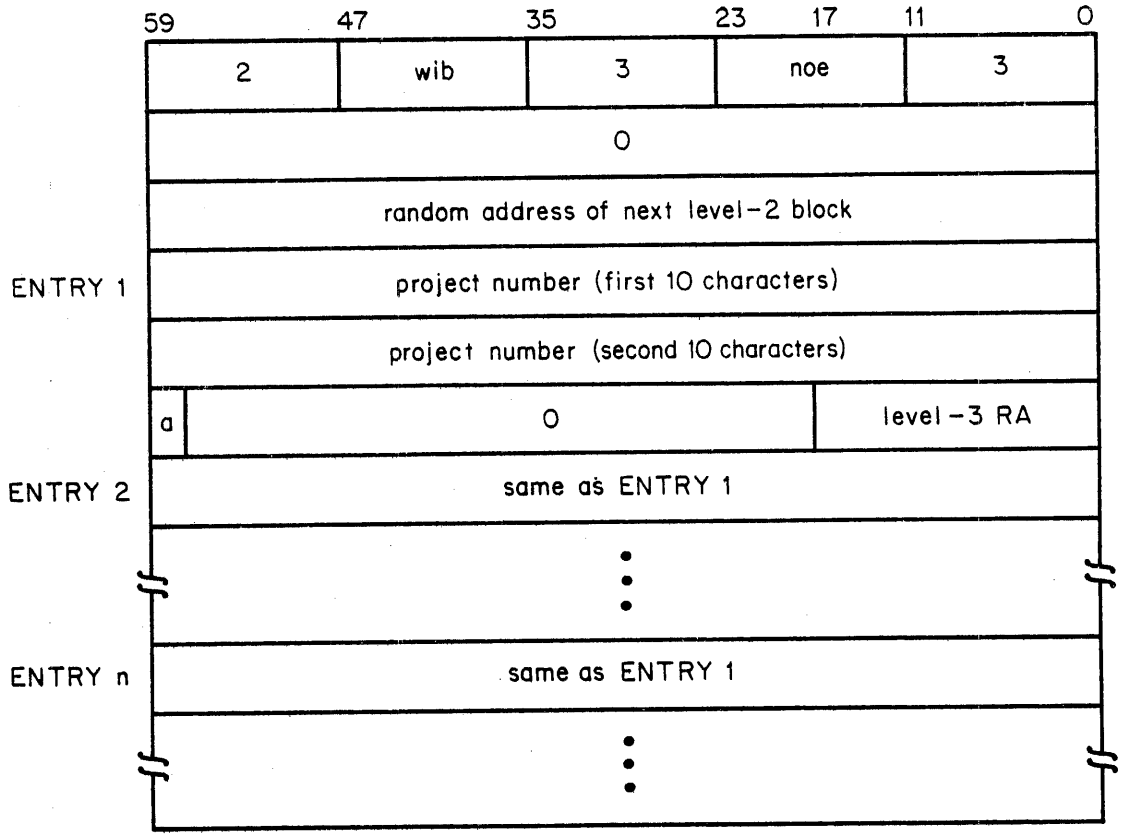


Figure 20-6. PROFILa Level-0 Block Format



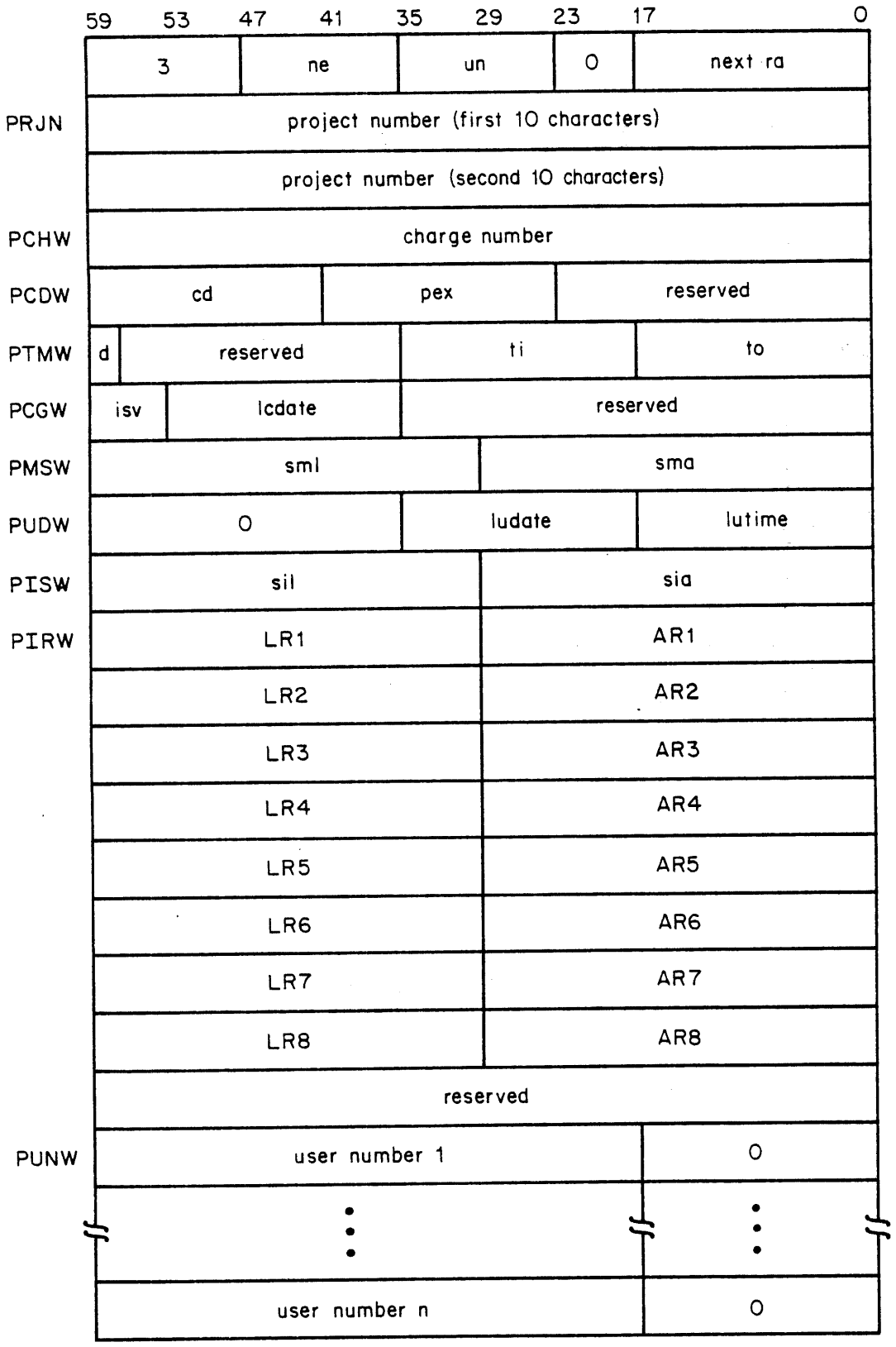
d	Charge deactivate flag
pcl	Project count limit
pc	Project count
Mi-AD	SRU multiplier indices
cd	Charge creation date
cex	Charge expiration date
ISL	Installation SRU limit index
IR1-IR8	Installation limit indices

Figure 20-7. PROFILa Level-1 Block Format



a = 0 signifies that the level-3 block has 1 entry and a = 1 signifies that the level-3 block has 2 entries.

Figure 20-8. PROFILa Level-2 Block Format



The above format is repeated if there is a second entry in the level-3 block.

Figure 20-9. PROFILa Level-3 Block Format

ne	Pointer to next entry in the block
un	Number of users in the project
ra	Random address of first overflow block
cd	project creation date
pex	Project expiration date
d	project deactivate flag
ti	Time in
to	Time off
isv	SRU validation limit index
lcdate	Last change data by PROFILE update run
sml	SRU master user limit
sma	SRU installation limit
ludate	Last update date by OAU
lutime	Last update time by OAU
sil	SRU installation accumulator (updated by OAU)
LR1-LR8	Installation limit registers
AR1-AR8	Installation limit accumulators

Figure 20-9. PROFILa Level-3 Block Format (Continued)

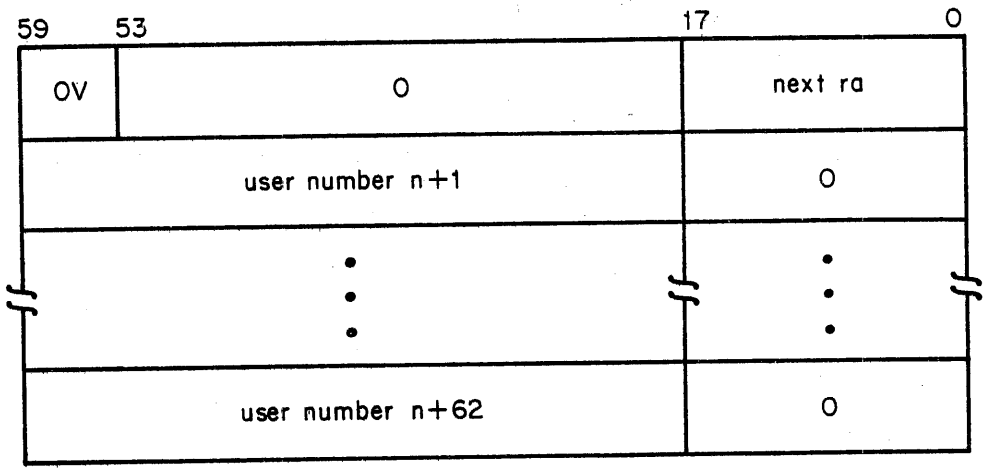


Figure 20-10. PROFILa Level-3 Overflow Block format

DELETED CHARGE AND PROJECT NUMBERS

Charge, project, and user numbers may be deleted from the PROFILa file; however, only the user numbers are physicaly removed from the file when deleted. A bit is set in a deleted entry to signify that the entry is no longer active. When a reformat PROFILE (OP=R) operation is performed, the entries that were marked as deleted are excluded in the restructuring.

CHARGE ROUTINE

The system routine CHARGE provides validation of a user's charge and project numbers for access to defined amounts of system resource usage measured in SRUs. A call to CHARGE is required for all users if the CCNR bit is not set in the user's access word (AACW).

If the validation of the user's charge is not successful, the job is aborted with an appropriate diagnostic. The validation is unsuccessful if the charge or project numbers are not found, the user is not validated to use the charge or project number, or the charge/project has reached its SRU limit.

If the validation is successful, then

1. Appropriate information is written to the account dayfile (ACCOUNT) indicating the user's charge and project numbers.

2. Accounting parameters (SRU multipliers and the SRU validation limit for the account block associated with the user's charge and project number) are entered into the job's control point area via a CPM function for usage in the accounting formula. These values are used until the end of job or session or until another CHARGE statement is issued.

3. The accumulated SRUs (or the minimum installation charge, if larger) are entered into the accounting dayfile and OAU called to update the SRU accumulator for this charge/project/user in the PROFILa file. This occurs only on the second and subsequent CHARGE statements.

4. The SRU accumulator in the control point area is cleared but the AD, CP, MS, MT, and PF accumulators are not affected.

ROUTINE OAU

Routine OAU updates the level-3 block for the charge/project number when the job's SRU accumulator overflows or at the end of an account block. This mechanism allows the information contained in the level-3 block for the charge/project number to be as up-to-date as possible. The equipment, track, and sector of the level-3 block for the project number is kept with the accounting words in the control point area; this allows fast access to the PROFILa file by OAU.

Routine OAU is called by the PP routines CPM, 1AJ, 1CJ, 1RO, 1SP, and 1TA. Routine OAU is flowcharted in figure 20-11.

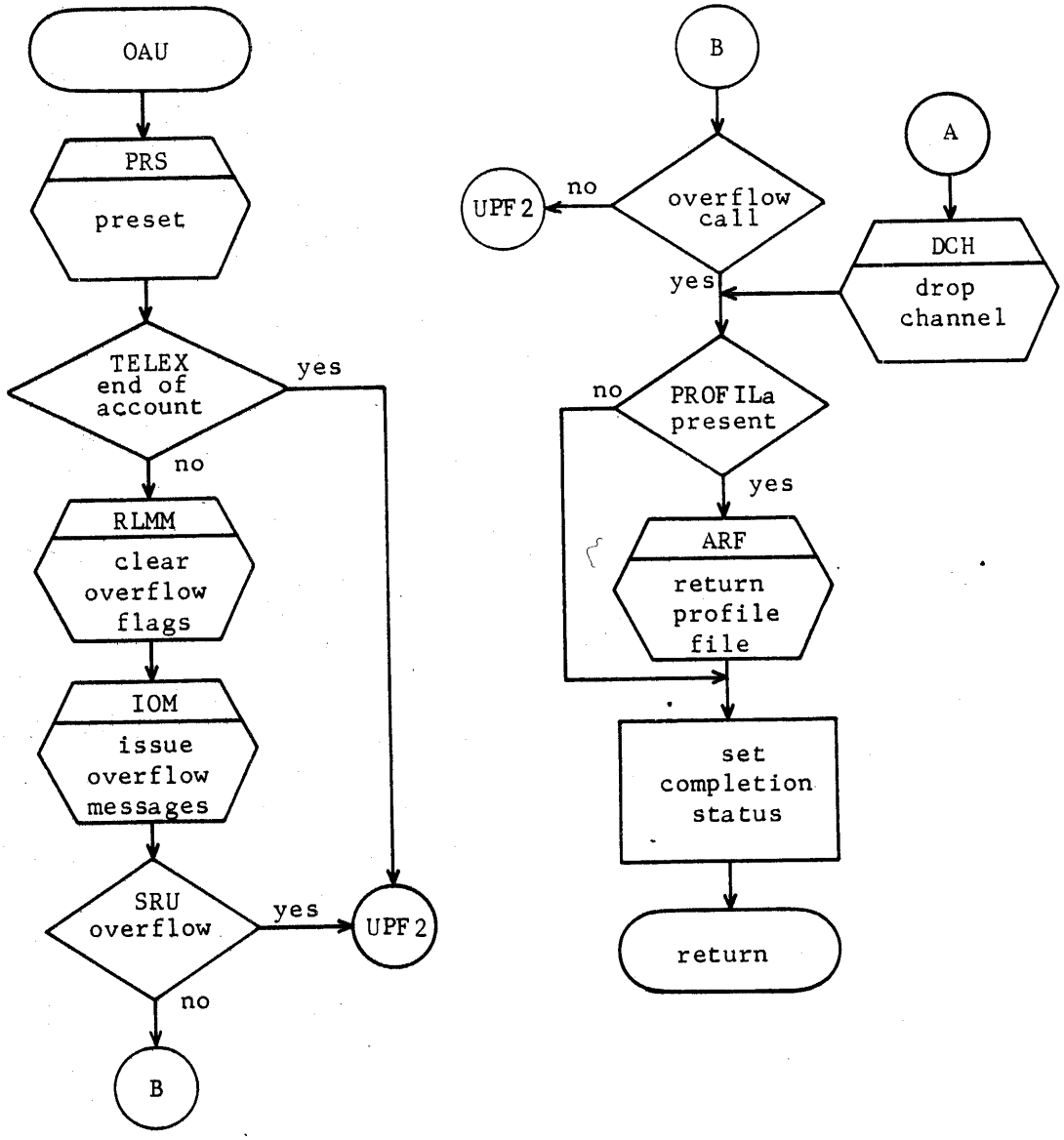


Figure 20-11. Routine OAU

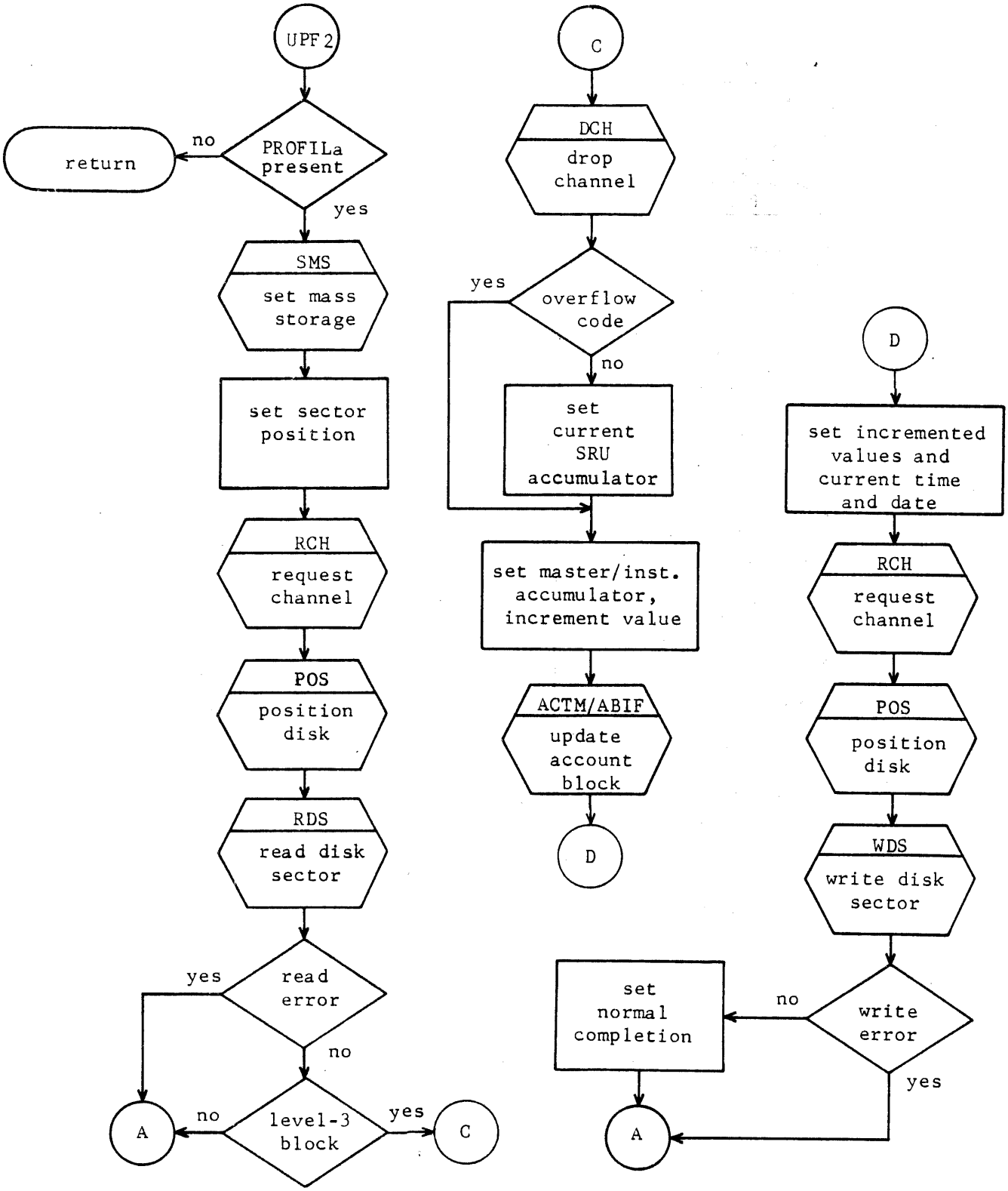


Figure 20-11. Routine OAU (Continued)

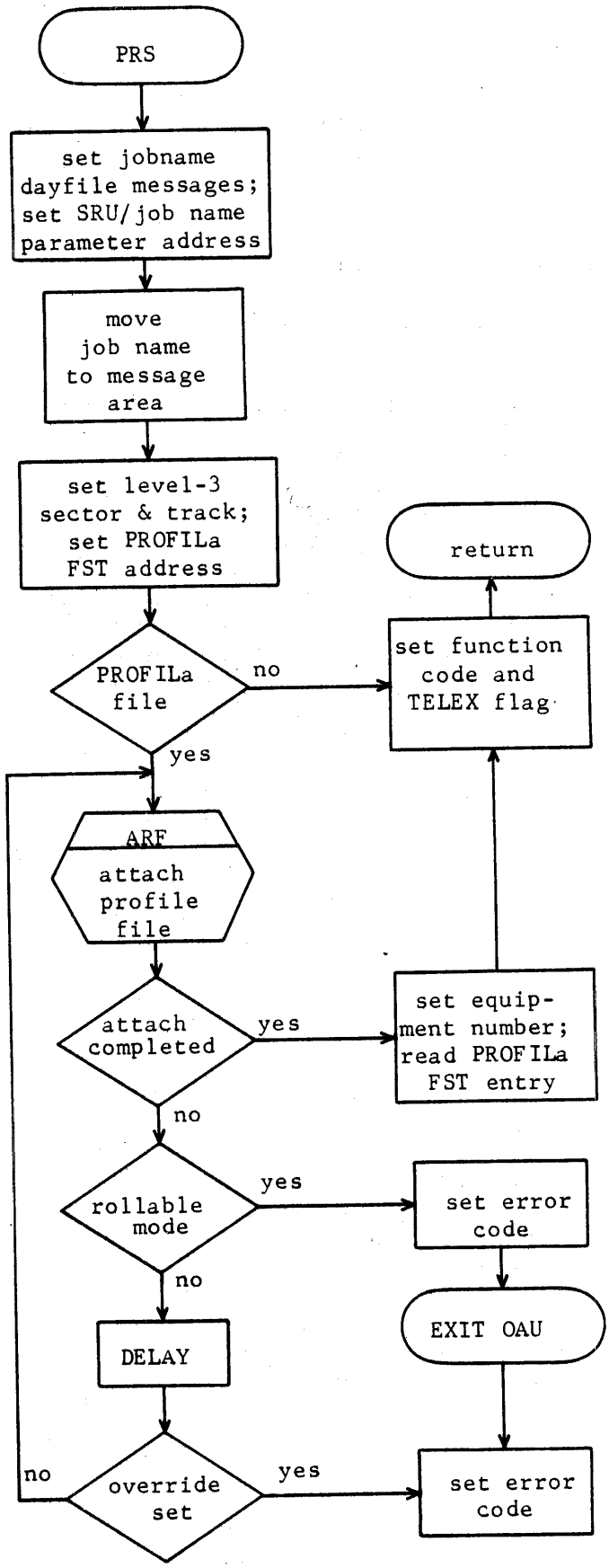


Figure 20-11. Routine OAU (Continued)

DATA BASE ERRORS FROM PROFILE

PROFILE issues a diagnostic message in a variety of conditions that normally should not occur. This diagnostic has the form:

```
DATA BASE ERROR nn - NOTIFY ANALYST.
```

The following errors will prompt this message to be issued.

nn	Description (routine)
1	Bad table 2 pointer (PDE)
2	Bad overflow block pointer (PRF)
3	Negative project count (UPC)
4	Bad queue pointer (MQE)
5	Bad level 0 or 1 block (RCE)
6	Existing charge number not found (RCE)
7	Existing charge number not found (CNP)
8	Existing project number not found (PNP)
9	Illegal K display update (CKU)
10	Bad level 0 or 1 block (CNP)
11	Existing charge number not found (CNP
13	Existing project number not found (PNP)
14	Bad level-3 block (PDE)
15	Bad level-3 block (PDE)
16	Bad table 3 length (WOB)
17	Bad table 3 pointer (WOB)
18	Bad table 3 pointer (WOB)
19	Bad level-3 block (ADB)
20	Bad level-3 block (ADB)
21	Bad level-3 block (ADB)
22	Bad level-3 block (CEP)
23	Bad random address (RDB)
24	Illegal field size (GFV)
25	Bad level-3 block (RDB)
26	Bad queue pointer (DQP)
27	Negative user number count (DUN)
30	Badl level-0 block (PRF)
31	Bad user number pointer (NUE)
32	Bad user number pointer (NUE)
33	Bad overflow block (NUE)
34	Bad Level 0 or 1 block (PIO)
38	Bad Level 0 or 1 block (CND)
39	Bad Level-2 block (PEI)
40	Bad Level-2 block (PCS)
41	Bad Level-0 block (PLO)
42	Bad Level 3 length (FUH)
43	Bad Level 3 pointer (FHP)
45	Bad Level 0 or 1 block (DQP)

MMF OVERVIEW

Multimainframe (MMF) enables up to four computers to access shared mass storage devices. This allows the mainframes to share preserved files residing on such devices. Each mainframe in the complex can operate in multimainframe mode or in stand-alone mode; however, two machines cannot access the same device unless both are in multimainframe mode. A device is considered shared if it can be accessed by more than one of the mainframes. It need not be accessible to all the mainframes in the complex.

ECS is used as the means and media for controlling shared mass storage and inter-mainframe communication. Each mainframe has a CPU port into ECS which is utilized by CPUMTR to control system activity. The ECS flag register is used as the first level of interlock among the machines accessing tables in ECS. These tables, such as the TRTs for all shared mass storage devices, contain lower levels of interlocks in order to provide the same controls that exist in a standalone environment.

In order to control shared mass storage devices, several ECS resident tables are required. A device access table (DAT) contains the logical description (family name/pack name and device number) of each mass storage device (shared and non-shared) that is accessible by any machine in the complex. For each device in the DAT that is to be accessed by more than one machine, a corresponding mass storage table (copy of central memory resident MST), and TRT, also reside in ECS. In addition, a machine recovery table (MRT) exists in ECS for each machine and device. That is, there are as many MRTs for each shared device as there are mainframes in the complex. Section 2 contains a detailed description of these tables.

The interlock for a device is a word in the ECS resident MST. When a machine is going to update an MST/TRT, it must place its machine mask in the interlock word of the MST. The writing of this interlock word is controlled by the ECS flag register. Only the machine which has a bit set in the ECS flag register may interlock a device. CPUMTR controls all interlocks.

Recovery allows for a machine to either join other machines operational in a multimainframe environment, or to come up in a standalone mode. The standalone system will not be allowed to use the same mass storage devices as other machines. Every effort is made to provide automatic detection of this situation; however, it is not possible to provide a 100 per cent guarantee that a system cannot be activated, and access a device currently accessed by other machines. The existing levels of recovery (1, 2, or 3) are available in a multimachine environment. Methods are available in the event of a system interruption to operate the remaining machines in a multimainframe environment.

MMF ENVIRONMENT

Any combination of from two to four 6000, lower CYBER 70, or CYBER 170 mainframes may comprise an MMF environment. ECS is required, with one CPU port for each mainframe. The presence of a DDP on a CPU port decreases by one the total number of mainframes that may run together. The 841, 844, and ECS devices are the only devices that are supported as shared devices.

If ECS is being placed in maintenance mode (MA parameter set in EQ=entry), a full level 0 deadstart must be performed. The size of ECS is automatically reduced by half at deadstart. The remaining half of ECS is available for ECS on-line diagnostics.

SYSTEM FLOW

Automatic detection of ECS is not provided because it is not possible to determine its absence and continue to run on all machine types. The 6600 CPU will hang if an attempt is made to execute an ECS instruction without any ECS. ECS status is checked by CMR (SET) when called upon to process an ECS entry in the CMRDECK. If the CMRDECK entries are present which indicate a multimainframe environment, a check is made to insure that either a DE or DP equipment entry is also present. If none is found, an error indication is given to the operator that no link device has been defined. The link device is automatically designated as a shared device.

DEADSTART

At deadstart time each machine must be given a two-character machine identifier (MID). This CMRDECK entry must be unique for each machine in the complex. The purpose of this identification is to associate a specific machine with its access to a shared device. There are no external characteristics associated with this identification.

When a mainframe joins a complex it is necessary to associate it with an identification that it can utilize as long as it is up and running, but which is also independent of the MID it has chosen for itself. During the deadstart process a machine investigates the MMF tables residing on the link device to find a slot (of which there are four) that currently is not being used by another machine. Once an empty slot is found, the MID is placed into it. Associated with each slot is a unique machine index and a unique machine mask which the machine uses to index itself into various MMF tables or to identify itself in these tables. The indices are 1, 2, 3, and 4; whereas the masks are 1, 2, 4, and 8.

In the deadstart process, if level 0 has been selected, it is necessary to determine if this is a global deadstart (first machine up in an MMF complex). The machine performing a global deadstart presets certain tables in ECS, and in so doing assures that no other machine has arrived at the same point in the deadstart sequence and is attempting the same thing. These operations are performed by the PRESET segment of CPUMTR.

One of the functions of a machine operating in a multimainframe complex is to periodically copy its system time into the MMF environment table. This mechanism provides a real time determination of how many systems are working together. Determination of which machines are running in the MMF environment can therefore be accomplished by having CPUMTR read the ECS table of system clocks, and then detect a change in the information written. Whenever CPUMTR goes through its advance running time routine (once per second), it also interrogates the flag register and updates its ECS system clock. Every second time, CPUMTR check the clocks of the other machines and sets an indication in MMFL (refer to section 2) as to which machines are currently active, and which machines have been active but are not currently. Once an inactive machine becomes active again, the MMFL status is reset to indicate that that machine is active.

Preset of ECS involves the initialization of the DAT and the MMF environment tables (refer to section 2). It is imperative that the link device (ECS) be initialized at deadstart time if it has not been previously used as a link device. When initializing the DAT, enough tracks are allocated to handle the maximum number of shared devices that can be supported in the complex.

The first machine deadstarted in a complex must have the PRESET multimainframe command entered when deadstarting. This machine clears the flag register and sets the interlock in the flag register indicating preset in progress (ECS resident is being preset). All other machines deadstarted in the complex should not have the PRESET multimainframe command entered at deadstart time. They interrogate the flag register for an interlock bit, which, among other things, indicates deadstart in progress. If they find it, a message to this effect is displayed until the interlock is cleared from the flag register by the presetting machine and then this machine proceeds with its deadstart process.

A machine that does not preset ECS cannot determine if it has already been preset by another machine. Therefore, the operator must insure that ECS has been preset by a prior deadstart before deadstarting a particular machine without presetting ECS.

A check is also made to ensure that the MID is not the same as any machine already running in the complex. If a machine has already specified the same MID, the deadstart process on the second machine halts and displays a message.

Any errors encountered before DSD is running are passed to RMS as error codes. RMS translates these error codes to messages and displays them.

SHARED MASS STORAGE

For purposes of device usage determination, tables are maintained in ECS that identify the status of all devices in the multimainframe complex. This includes shared and nonshared devices for all machines. These tables are called the device access tables (DAT). The format of the DAT is illustrated in section 2.

In order to minimize configuration problems, all shared removable equipments should be configured the same on all machines in the complex. For example, if one system defines three shared units as three DI-1s and another system defines the same units as a DI-3, the first system can accommodate a DI-2 on these units whereas the second system would find it as an error situation. Unless the configurations are the same on all machines, any devices mounted on those drives may not necessarily be recoverable on all machines.

RESEX considers only the configuration of the machine it is executing on in its overcommitment algorithm.

MASS STORAGE RECOVERY TABLES

One problem that is created by having more than one machine sharing a mass storage device is that of recovering a machine's mass storage space and interlocks should it require recovery processing. This problem is resolved by having a table that provides the information needed to recover a machine's mass storage space and by having a utility (MREC) that does the recovery. This table is called the machine recovery table (MRT). There is one MRT for each mainframe per device. It describes which tracks are interlocked and which tracks are first tracks of files local to a particular machine. The MRT is utilized in two ways: either during deadstart recovery on an interrupted machine, or by MREC which is run on another machine to recover the mass storage space of the interrupted machine. Each MRT is a maximum of 100B words with the low order 32 bits in a word being used. One bit in the MRT represents one logical track. Bits 10 through 5 of the logical track number denote the word number in the MRT and bits 4 through 0 are the bit number within the word.

The meaning of the MRT bit, if set, depends upon the state of the track interlock bit for the same track in the TRT.

Track interlock bit	MRT bit	Description
0	0	Track not interlocked or it is local to another machine
0	1	Indicate first track of a file local to this machine
1	0	Track is interlocked by another machine
1	1	Track is interlocked by this machine

Track interlocks are set and cleared in the following manner:

1. When the MRT bit denotes the first track of a local file:

   a. MRT bit is set for the first track when tracks are initially requested for a file.

   b. MRT bit is cleared when the preserved file bit is set.

   c. MRT bit is cleared when the first track of a track chain is dropped.

2. When the MRT denotes that a track (preserved file only) is interlocked:

   a. MRT bit is set when track interlock bit is set.

   b. MRT bit is cleared when track interlock bit is cleared.

   c. MRT bit is cleared if the track is the first track of a track chain being dropped.

The purpose of the MRT is the recovery by the remaining machines of a down machine's mass storage space and interlocks. This MRT processing is one of several things that must be performed to recover shared devices from an interrupted machine. (The whole area of machine recovery is discussed in more detail later in this section.) Briefly, the steps required to process an interrupted machine's MRT are as follows:

1. Secure the device interlock to insure that another machine is not attempting the same recovery.

2. Read in MRT for device and machine being recovered.

3. For each MRT bit that is set, the following is done depending on the state of the corresponding track interlock bit. If the track interlock bit is set, this track was interlocked by the down machine. Clear the track interlock.

4. Rewrite the MRT with bits cleared for those tracks whose interlocks were released. Clear the device interlock (ACGL).

5. If MREC is then called, it secures the recovery interlock (DATI) to insure that another MREC is not attempting the same recovery.

6. Read in the MRT for the device and machine being recovered.

7. Then, for each MRT bit that is set, do the following. If the track interlock bit is not set, this track was the first track of a file local to the down machine. The track chain is dropped. A verification is performed to insure that the chain is indeed local and reserved.

TRT INTERLOCKING

CPUMTR interlocks and updates its copy of the MST/TRT for DLKM, DTKM, STBM, and RTCM (AMSM) monitor functions. There is an option on the STBM function to indicate that the MST/TRT should be updated. The way these functions interlock an MST/TRT is as follows.

1. The device is determined to be shared (indicated by the presence of an ECS address in the MST).

2. CPUMTR attempts to secure the flag register interlock (TRTI).

3. If CPUMTR successfully sets the flag register interlock, it then attempts to interlock any MST/TRT. This is done by placing its machine mask in the SDGL location in the MST. If this location already contains a machine mask, CPUMTR clears the flag register interlock, rejects the request, and exits to handle other requests.

4. If CPUMTR can set the MST/TRT interlock it then rewrites the first three words of the MST out to ECS with its machine mask in SDGL. The TRT flag register interlock is then released and CPUMTR is ready to read in the TRT and global MST.

5. A check is made to see if the MST/TRT has been updated since this machine last obtained an up-to-date copy. If it was not, the TRT need not be read in. At this point the MST/TRT is interlocked with the current copy in CM and it can continue processing the monitor function. When the monitor function is completed the TRT is updated. A field in SDGL is incremented indicating that this was the last machine to update the MST/TRT, thus making this device available for others to allocate on. The first three words of the MST are then written out to ECS (with the MST/TRT interlocked cleared) making this device available for others to allocate on. The interlock word is the last of any global words written to ECS.

6. Separate steps must be taken if CPUMTR cannot secure the required interlocks.

DEVICE INITIALIZATION

In order to initialize a shared device, it is necessary to prevent any new activity from starting on the device. The next step is to wait until all current activity has completed. Once it is detected that all activity has subsided on the device, the device is interlocked and initialization proceeds. The following steps must be taken to accomplish this set of tasks.

First, the INITIALIZE command should be entered on the machine from which the initialization is to take place. All other machines sharing the device need to unload the device (enter the UNLOAD command) in order to prevent any new activity. The INITIALIZE command sets the initialize bit in this machine's local portion of the MST. If when attempting to set this bit it is found that another machine already has its initialize bit set, the operation will not be performed; instead, an error is displayed at the system control point.

Next, IMS, on the machine with the initialize set, monitors the status of the other machines that are sharing the device. Once each of them has their unload status set and all user counts are zero, initialization proceeds.

When the device is free for initialization, the device is interlocked and the current TRT is obtained from ECS. After initialization is complete, the MST/TRT is updated in ECS and the checkpoint request bit is set in the MST. The initialization bit is then cleared. Routine 1CK is then called to checkpoint the device. The initialization process includes updating of the DAT entry (if necessary).

In order to activate the device on other machines, the operator must mount the device. The MOUNT command clears the unload status. If initialization is still in process on another machine when a MOUNT is attempted, the MOUNT process is terminated with an error displayed at the system control point. (The recovery of the device that occurs at this point is the same as items 3 and 11 in the recovery matrix in table 21-2.)

DEVICE UNLOAD

UNLOAD from any machine sets the local unload bit in the MST for this machine only. This is an indication to this machine that no new accesses should be initiated. If the intent of the unload is to eventually physicaly unload the device, then an unload must be entered on each machine sharing the device. When CMS on any machine checks a device and finds that all local unloads are set and all user counts are zero, it then sets the global unload only if the device is a removable device. The global unload is displayed to the operators on all machines indicating that there is no activity on the device from any machine and that the device may be physically unloaded.

When a CMS detects that it can set the global unload and does so, it also clears the DAT entry in ECS.

The operator can then switch packs and enter the MOUNT command at the console to initiate recovery of the device. A MOUNT command clears the global unload, if set, and the local unload for this machine indicating that this machine is now accessing the device. All other machines continue to ignore the device until the operator enters the MOUNT command on those machines. The MOUNT command does nothing if the local unload is not set.

RMS and CMS use similar logic in recovering mass storage devices. When a device is recovered, the DAT is interlocked while a check is made to see if an entry exists for this device. The presence of an entry indicates that another machine is also accessing the device. If an entry is found, and the machine recovering the device has not been instructed to share it, an error is indicated and recovery halts with an appropriate message displayed. If the machine already accessing the device is not allowing it to be shared, (TRT is not ECS resident), the same error condition is detected. These situations are illustrated in table 21-1.

The statuses across the top indicate in which mode the device is being utilized. The statuses down the left side indicate in which mode a machine coming up wants to utilize the device.

TABLE 21-1. DEVICE ACCESS STATUS

	Device used in nonshared mode	Device used in shared mode
Use device in nonshared mode	ERROR 2 machines want to use the same device in nonshared mode.	ERROR Machine coming up wants to use a device in nonshared mode that other machines are sharing.
Use device in shared mode	ERROR Machine coming up wants to use a device in shared mode that another machine is using in nonshared mode.	Add accessing status to DAT

When a machine recovers a device, it adds an indication to the DAT entry, if the indication does not already exist, that this machine has accessed (recovered) this device.

If the device is shared and another machine has it interlocked, a bit in the DAT is checked to determine if a level 0 type recovery is in progress on the device. Once the recovery is completed, recovery on this machine proceeds as indicated below. It is not allowable to attempt a non-level 0 recovery on an interrupted machine once MREC is run on another machine to recover the mass storage space of the interrupted machine. When RMS recovers a device on a non-level 0 deadstart, the DAT indicates that this machine has accessed the device previously. This status is cleared by the machine recovery utility.

It is the responsibility of each machine to recover its own local MST area from the device. The manner in which the local information is maintained on the device is detailed under Device Checkpoint. A bit in the global portion of the MST indicates if the sector of local information exists. In any event, if the local area that exists in the label sector matches the MID of the recovering machine, that local area is assumed to be the most up-to-date regardless if information also exists in the sector of local areas.

A level 3 deadstart assumes that the ECS MMF tables are intact as well as CMR when running in a multimainframe complex.

If no entry for the device to be recovered exists in the DAT, an entry is made by RMS. A flag register interlock is set to prevent other machines from attempting the same. Once recovery is completed, the flag register inerlock is cleared by REC.

Table 21-2 shows the steps involved for mass storage device recovery during the various levels of deadstart. When a device is not shared with any other mainframe, it is termed a stand-alone device. If the device is shared, the DAT is interrogated and recovery proceeds differently depending on whether it is active (in the DAT) or not active (not in the DAT). Another criterion that denotes which steps are taken is the machine mask field in the DAT which indicates either that the device has been accessed previously by this machine or that the device has not been accessed previously by this machine. Removable devices recovered on-line are handled the same as devices on a level 0 deadstart.

TABLE 21-2. MASS STORAGE DEVICE RECOVERY

Level of Deadstart \ Device Type	Stand Alone Device	Shared-not active (not in DAT)	Shared-active (in DAT)	
0	2,4,6,7,8,14	1,4,6,7,8,9,10,14	3,11	-device not accessed previously
	n.a.	n.a.	11,12,13	-device accessed previously
1 and 2	2,4,7	1,4,5,7,9	3,11	-device not accessed previously
	4,7	n.a.	11,13	-device accessed previously
3	error	error	error	-device not accessed previously
	4,7	n.a.	11,13	-device accessed previously

1. DAT entry not found; make DAT entry which indicates that this machine only is currently accessing the device.

2. DAT entry not found; make DAT entry which shows that this machine is accessing the device but has no MST pointer (not shared).

3. Add indication to existing DAT entry which shows that this machine is accessing the device.

4. Retrieve MST (all local and global portions) from the device and, if shared device, preset into ECS. Retrieve TRT from device.

5. Set MRTs from device into ECS.

6. Edit TRT (release all track chains except the preserved file chains).

7. Clear track interlocks for all machines.

8. Clean up system sectors (interlocks and user counts) for all machines.

9. Set TRT from device into ECS.

10. Clear MRTs for all machines.

11. Retrieve TRT and global MST from ECS. Get local MST from device. Clean up local MST (clear interlocks, reservations and request statuses).

12. Process MRT for this machine and drop local tracks.

13. Process MRT for this machine and clear track interlocks.

14. Build file of inactive queued files.

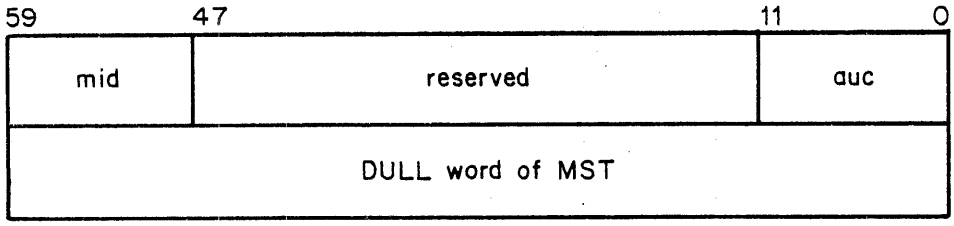
DEVICE CHECKPOINT

Routine 1CK checks for checkpoint requests in the local portion of the MST. If a device is not shared, the checkpoint proceeds as for standalone systems, if the checkpoint request bit is set. If a device is shared, and the request bit in the local area of another machine is set, no action is taken. (This situation arises when another machine accomplishes the requested checkpoint because the checkpoint bit is set in its local area.)

If the checkpoint request bit is set, it is cleared and the local TRT and MST updated from ECS (if necessary). The global portion and local portion of the MST as well as the TRT are checkpointed to the device.

The local MST information for all machines is maintained on the device. All of this information is kept in one sector on the label track following the TRT sectors. This requires a read and a write to update the sector which means updates (checkpoints) must be minimized and therefore performed separate from normal TRT checkpoints.

The format of the two-word entries in this sector is as follows.



mid	Machine identification
auc	Active user count

A maximum of 37B entries can exist.

Whenever a field in the local area of the MST is updated which requires checkpointing (currently only those fields of word DULL), those programs that perform the update make a call to 1CK. This 1CK function reads up the sector, searches for a MID match, and then updates the existing entry or adds one if none exists, and finally rewrites the sector. Bit 6 of word ACGL of the MST indicates whether or not the sector exists yet.

FAST ATTACH FILES

The concept of fast attach files is to move the access control and interlock mechanisms for highly accessed, system, direct access permanent files, from the mass storage device into the FNT. This provides a much quicker access to the information on these files primarily by eliminating several disk accesses (six for the normal ATTACH/RETURN sequence) each time the file is accessed. To provide accessibility to all machines, this information must either be maintained on the mass storage device (shared) or in ECS (link device). Putting the information back on the device eliminates the benefits of fast attach and is therefore undesirable. By placing the control information into ECS some overhead is added to the file access sequence but it is minimal compared to having it on the device itself.

Eight-word tables are maintained on the same track in ECS as the DAT for each fast attach file. Within this eight word table are the file name, family name, and five words of access and control information that are laid out identically to words 20B through 24B of the permanent file system sector. Each machine keeps its access and control information separate but looks at all words to determine the current state of the file. There is a limit of 77B fast attach entries.

The monitor function (IAUM) provides the means for interlocking the fast attach file when operating in standalone mode. When operating shared mode this interface extends to accessing and updating the information in ECS.

PERMANENT FILE UTILITIES

The PFNL word in CMR is used to keep track of permanent file activity and also to control that activity. A dynamic count is maintained in this word to account for the number of PFMs/PFDUMPs that are currently accessing permanent file devices.

For this purpose, PFM and PFDUMP utilize this word in the same manner. PFM and PFDUMP determine from this word if they can execute. If so, they increment the count and, when finished, decrement the count.

PFLOAD needs to interlock a whole device so that it is assured no one else (namely PFM) is changing the device (such as catalog entries or track chains) as it is loading the device. Therefore, since PFLOAD only knows if there is activity on some device and not necessarily if it is the device it wants to load, PFLOAD sets a total interlock in PFNL which prevents any new activity on any device. Once all activity has subsided, PFLOAD sets the MST interlock (permanent file utility active) for the device it is interested in and drops the total interlock in PFNL.

Likewise, CMS sets the total interlock in order to prevent further activity. However, in the case of CMS, this is done to halt all activity while it changes device configurations.

The permanent file utilities are allowed to execute in a multi-mainframe environment in the same manner as in a standalone system. To accomplish this the PFNL word (110 in CM) is kept in the FAT table (entry 0) (refer to section 2) in ECS. Each PFNL word in ECS is maintained separately but a global word is also used to control total permanent file activity.

To accommodate the updating of PFNL, options are provided on the IAUM monitor function to do the following: secure/release total interlock; secure/release request for total interlock; and increment/decrement activity count.

I/O QUEUE PROTECT

Queue file protection when in MMF mode presents special problems when the following occur:

- A down machine does a level-0 deadstart.
- MREC is run on another machine to clean up the device for the down machine.

When a machine running with QPROTECT enabled goes down, it must decide what is done with the previously active queued files that reside on a shared device whose IQFT file is actively being used by another machine.

In this case, the queued files are retained since they have the preserved file bit set on them; however, they are inaccessible until all machines do a level 0 deadstart.

When a machine not running with QPROTECT enabled has dequeued file entries in the IQFT of a shared device and goes down, it must decide how to delete IQFT entries once the local tracks for such files are dropped.

Here, dequeueing is not allowed if the queued file is not preserved and the device is a shared device. In this case a dayfile message is issued indicating that the queued files were ignored on the shared device.

CPUMTR CONSIDERATIONS

A significant amount of code is required in CPUMTR to support a multimainframe environment which is not needed by sites not utilizing this feature. Since CPUMTR resides in central memory, it is desirable to provide a mechanism whereby code associated with a particular feature (in this case multimainframe) may be optionally loaded or discarded at system deadstart time.

SEGMENTATION

CPUMTR accommodates blocks of code that may be optionally loaded. These blocks of code are placed into labeled common by USE instructions. Each block of code usually has two segments of which one is always loaded. All block names are unique. For example, if ECS is the name of the block which is loaded when ECS processing is desired, then OECS is loaded in its place if ECS processing is not desired. The convention therefore is to place a zero in front of the block name for the option not present block. Any given feature may have as many blocks associated with it as is necessary with any number of them being loaded.

A CPU program, CPUMLD, loads the desired CPUMTR blocks. CPUMLD is a simple relocating loader which reads in and loads the segments required to utilize any optional feature selected during the prior deadstart process. This covers the case of wanting one set of code for environment A and another set for environment B. STL loads CPUMLD, and CPUMLD issues requests to STL to read in CPUMTR from the deadstart tape.

ECS INTERLOCKS

When operating in MMF mode most of the device related interlocks must be maintained in ECS in order to access them from all

machines. To provide control for the access of these interlocks, another level of interlocks has been defined in the ECS flag register which must first be secured before accessing and changing the respective ECS areas. The following ECS flag register interlocks have been defined for this purpose.

TRTI Interlock

Secured whenever updating an MST or TRT in ECS. Updating includes attempts to secure interlocks maintained in the MST itself. Because of the high frequency usage of this interlock, the machine mask identifying who has the interlock is kept in the flag register also instead of in ECS as for the other flag register interlocks. This reduces the number of ECS accesses and therefore reduces the interruptions of data transfers.

PRSI Interlock

Secured by a machine when assigning or updating the machine ID words in the environment table. This occurs on all levels of deadstart during CPUMTR PRESET when a machine is being recovered or assigned.

BTRI Interlock

Secured by a machine which wants to perform a block transfer via the ECS storage move area (sector 17B of the label track).

MRUI Interlock

Secured by the machine recovery utility (MREC/1MR) before attempting to recover ECS tables, interlocks and shared devices from a downed machine. This is to prevent another machine recovery utility from doing the same thing simultaneously.

CIRI Interlock

Secured by CPUMTR when cleaning up the flag register interlocks and device interlocks for a downed machine as soon as it senses that it is down. This is to prevent another CPUMTR from attempting the same thing simultaneously.

DATI Interlock

Secured by any program that is searching, adding entries to, or deleting entries from the FAT or the DAT. Also secured by any program that wants to change an existing DAT entry or update the DAET/FAET words in the environment table.

FATI/PFNI Interlocks

Secured by CPUMTR or MREC to update PFNL words or to update existing fast attach entries.

IFRI Interlock

Secured by CPUMTR when entering an inter-machine function request.

COMI Interlock

Set by CPUMTR when communication processing is requested of other machines.

CMR INTERLOCK TABLES

There are two types of CMR tables that reside in ECS that have interlocks associated with them. These are the MSTs and the PFNL (refer to section 2) which require the following interface.

PFNL Table

All PFNL updating for a machine must be performed via the IAUM function. This includes: request/release total interlock, request/release request for total interlock, increment/decrement activity count, and family count. CPUMTR must secure PFNI before performing the update.

Global area words TDGL, ACGL, SDGL and local area word STLL are only updated by CPUMTR. The remaining global areas are only updated by CMS and IMS. (CMS and IMS use ECSM to update these areas in ECS.) Initial set-up is performed by CMS/RMS which has unavailable set for interlock; the PFNL total interlock is secured when recovering a new device and when removing a device.

Local areas (currently only DULL) are updated by obtaining the local utility interlock (in STLL). This interlock is secured and released in place of the former device interlock (COMPSDI and COMPCDI) for those programs that still require it.

Local areas are written to ECS without any interlocks. Global areas written to ECS require an interlock.

TRTI (flag register), is used by CPUMTR when interlocking the device. A device must be interlocked when writing the TRT and words TDGL, ACGL and SDGL.

IMS has initialize set which acts as an interlock for the rest of the global area. CMS/RMS have unavailable set for initial setup.

All updates necessary in CPUMTR's words (TDGL, SDGL, ACGL, STLL) must be requested via the STBM function. Non-CPUMTR words must be updated in ECS (if shared device) by the respective PP program via ECSM.

INTERLOCK REJECT HANDLING

When CPUMTR attempts to secure one of the various interlocks it may find that it is interlocked by another machine. There are three things it can do with the function.

- Retry the request.
- Return an indication to the PP that the interlock could not be secured. PP resident would sense this situation and reissue the function following a delay.
- Return an indication to the PP that the interlock could not be secured. The PP program itself would decide what to do next.

Following steps describe how rejects of various interlocks are handled.

1. Any reject on ECSM/SFRS (set flag register bit) is communicated back to the PP program.
2. If CPUMTR cannot secure BTRI, it performs a regular storage move instead of through ECS.
3. For monitor functions which require securing of a flag register interlock (TRTI, FATI/PFNI) to complete, monitor attempts to secure the interlock n times. If still unsuccessful, the monitor function is requeued to PPR so that PPR can reissue it after a delay.

The flag register interlock may be secured; however, the next level of interlock (device interlock) may not be obtainable. The associated flag register interlock is released and the unsuccessful attempt is requeued for retry by PPR.

Likewise, the PFNL interlock may not be obtainable. Rejects of this type must be processed on an individual basis. In most cases it is not advantageous to have PPR continually retry the function. Whenever possible, the function should be rejected back to the PP program so that it can go on recall or perform other processing that minimizes resource monopolization.

Since most function requests continuously attempt to secure the desired interlock and not proceed until they have it, if a machine goes down and has various interlocks, it is important that those interlocks be cleared as quickly as possible to allow the function requests to complete. To accomplish this, as soon as CPUMTR detects that a machine has been down for four seconds, a routine (in CPUMTR) is executed to clear interlocks the downed machine may have in the following areas.

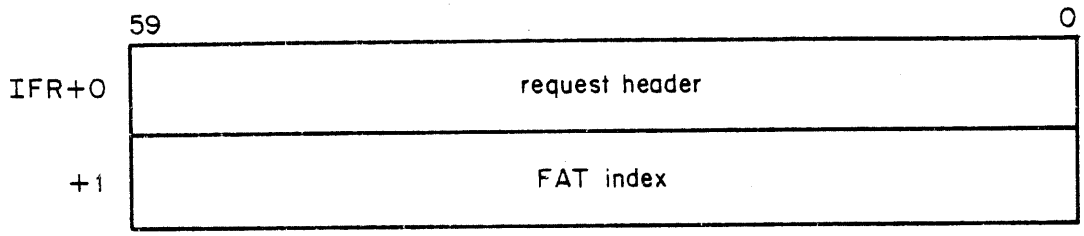
- Flag register
- Track interlocks
- Device interlock (MST/TRT)

All other interlocks and requests for same must wait until the operator initiates MREC. When interlocks of the latter type are encountered, a reject is returned to the PP program so that it can process it.

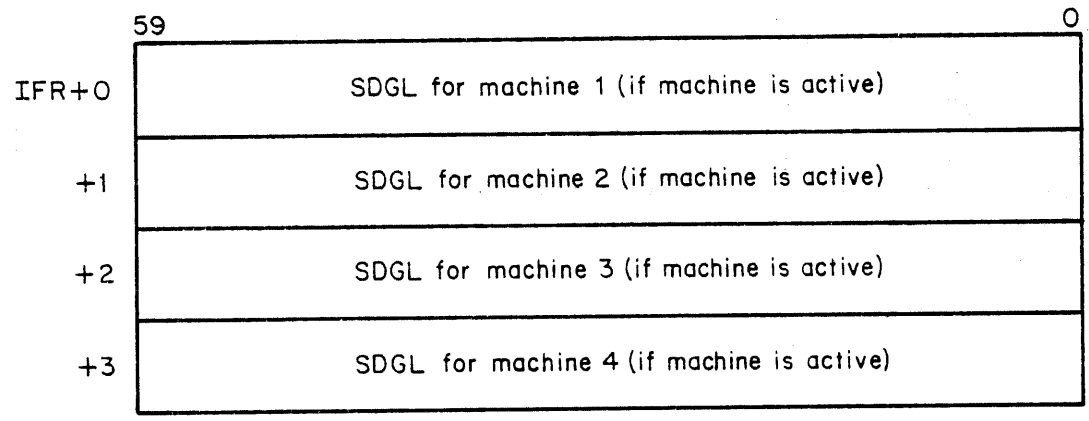
INTER-MAINFRAME FUNCTION REQUESTS

In the processing of ECS parity errors, it is sometimes necessary to have an ECS table restored by another machine. This is accomplished by an inter-mainframe function request (IFR). There are two ways it initiates an IFR. First, in the case of ECS parity error processing, the IFR is issued internally to CPUMTR. The second method is for PP programs to issue an IAUM function that enters the function request and its communication information in an ECS communication area. Following is the format of the IFR.

IFR Format for fast attach track parity error processing.



IFR Format for device parity error processing

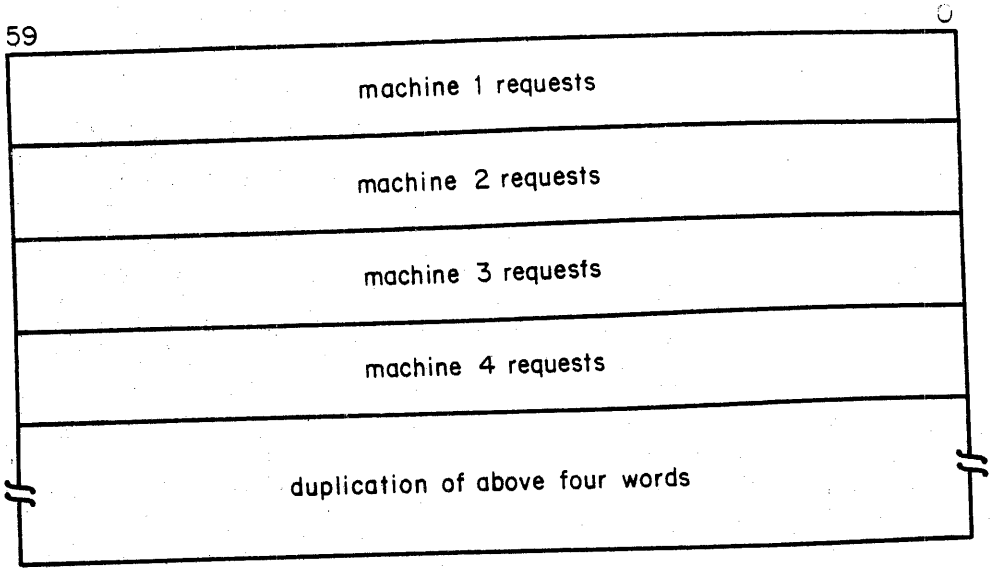


Communication is accomplished with a three-level information structure. The first level is the flag register bit COMI which tells when there is at least one function request present. CPUMTR checks this bit once every second when statusing the flag register. If the communication bit (COMI) is set, then CPUMTR attempts to interlock communication processing (IFRI flag bit) and read the function request words. One machine request word, the second level in the communication structure, is allocated for each mainframe. Each machine request word is a bit table describing which communication areas contain function requests for the particular machine. The communication area is the last level in the structure. It contains the function number and parameters to be processed.

The following flag register bits are used for inter-mainframe function processing.

Bit	Description
COMI	Set when at least one function request is present.
IFRI	Interlocks the updating of the machine request words and the communication areas.

The MMF environment table contains four machine request words (one for each mainframe). These describe which communication areas contain function requests for their associated machines. Its format is as follows.



Machine request word

Bit	Description
47	Communication area 0
46	Communication area 1
45	Communication area 2
44	Communication area 3
43	Communication area 4
42	Communication area 5
41	Communication area 6
40	Communication area 7
39	Communication area 8
23	Communication area 0 flawed
22	Communication area 1 flawed
21	Communication area 2 flawed
20	Communication area 3 flawed
19	Communication area 4 flawed
18	Communication area 5 flawed
17	Communication area 6 flawed
16	Communication area 7 flawed
15	Communication area 8 flawed

When entering an IFR in a communication area, a read is performed after the write to insure the words can be read. If an unrecoverable error occurs either on the read or write, the request bit is left set for that communication area and the next available one is tried. The flaw bit is set when an unrecoverable read error occurs while trying to process the request.

PARITY ERROR PROCESSING

The purpose of ECS parity error processing is to report and recover the following types of errors.

- Transient errors in a block read or write which can be recovered by retrying the instruction.
- Solid errors in a block read or write which can be recovered by single word transfers of the block.
- Solid errors on a read which can be corrected by rewriting the correct data. This type of error is recovered for MSTs/TRTs and fast attach track entries.

When an ECS error (half exit) occurs, the following steps are performed.

1. Increment the ECS error count in word DSLL of the MST for ECS.
2. Retry the block read/write one time to check for the first error type. If the error can be corrected by this method, reporting is done and normal processing continues.
3. Retry the transfer with single word read/writes. If the operation is a read, the data read with the single word transfer is compared with the data read on the block read. This allows the detecting of the good and bad data for the second error type.
4. If the error is recovered by single word transfers, it is reported and normal processing continues.
5. If the error is unrecovered it is reported and the appropriate error action taken.

NOTE

A single retry is the optimum number for either the single or block transfers. If failures are still encountered on the single retry, it is highly doubtful that the integrity of the data can be trusted even if future recovery attempts were successful.

A recovered error is reported and normal processing continues. Write errors are retried and reported where possible. If an unrecovered write error occurs, the following message is displayed at the system control point and central monitor will be stopped.

HUNG - ECS ERROR

The following is a list of error actions for unrecoverable errors.

1. MRT parity error processing.

   - The master copy of an MRT resides in CMR following the TRT.
   - If an error occurs in writing the MRT words to ECS, the error is reported and ignored.
   - If an error occurs in reading the MRT (this is done when clearing track interlocks for a down machine), the error is reported and no clearing of track interlocks is done for that device.

2. ECS clock error processing.

   - If an error occurs while writing the ECS clock for this machine or while reading all clocks to status machines, the error is reported and the statusing of the clocks is not done.

3. Flag register interlock words.

   - When setting or clearing a flag register bit, a word is written indicating which machine has the flag register bit interlocked.
   - If an error occurs while writing one of these words, the error is reported.
   - If an error occurs while reading these words, (during down machine processing to clear flag register interlocks), the error is reported and down machine processing is aborted.

4. MST errors in down machine processing.

   - The first three words of the MST are read to check if the device interlock is set by a down machine.
   - If an error occurs while reading these words, down machine processing is aborted.

5. ECSM errors.

   - If an error occurs in any ECS subfunction, the error is reported and the error address returned to the calling program.

6. Machine request word errors.

   - The machine request words are written in duplicate to ECS
   - If an error occurs while reading the first four request words, an attempt is made to read the duplicate words.
   - If these cannot be read, the IFRI interlock is left set and processing is terminated. The message SYSTEM ECS TABLE ERROR. is displayed at the system control point.

7. Device error parity error processing.

   - When an MST/TRT read error occurs, an IFR (refer to figure 21-2) is initiated to request the rewriting of the ECS tables for this device. This request is processed by all active machines including the machine initiating the request. The TRTI flag register interlock is left set until processing of the IFR is complete.
   - The processing of this request involves the following steps. First, search for the specified device by comparing ECS addresses of MST/TRT. If the device is not found, processing is cleared for this machine. If the device is found, the MRT is written to ECS. Next, set the device update count in this machine's slot in the IFR. Then check to see if this machine has the maximum device update count. If this machine does not have the maximum update count, processing is complete (but not cleared) for this machine. If this machine has the maximum update count, the ECS and CM copies of the MST/TRT are compared and all errors reported. Next, the MST/TRT is then written to ECS, a checkpoint request is set for the device, processing is cleared for all machines, and the TRTI flag fit is cleared.

8. Fast attach track parity error processing.

   When an ECS read error occurs while reading a fast attach track entry, an IFR (refer to figure 21-2) is initiated to request all active machines to process this FAT entry.

   In processing the IFR the following steps are done. First, the FAT entry is read. Next, search for the FAT entry in the FNT. Entry 0 equals PFNL is always present. If it is not found, set zero word for this machine's entry. If it is found, reconstruct the entry from the FST. Then compare the computed entry with the entry read from the FAT. If there is a difference, report the error and write the correct word to ECS. Finally, recompute total counts word, write to ECS and clear processing for this machine.

REPORTING OF ECS ERRORS

For each word that is in error, the following information must be passed to a PP program for reporting.

- Read/write indication
- CM address of transfer
- ECS address of transfer
- Word count of transfer
- ECS error count (DSLL word of ECS MST).
- Good/bad data present flag.
- Bad data
- Good data
- Retry count

There could be as many words in error as the size of a TRT. This represents a large amount of data to be stored for reporting. To avoid having a CM buffer for storing this data, a PP program 1MC is assigned when reporting ECS parity errors. This program buffers the three-word blocks of error information into PP memory. When all the data for this error is passed, 2MC is called in to process the data. 1MC must be assigned and loaded while CPUMTR is active (possibly in monitor mode). This means that the library search and load of 1MC must be done without any monitor functions. If no PP is available, the reporting is not done.

The format of the error log message is as follows.

```
ECxxxx,R/Wyy,R/U/S,zzzzzzz,Cxxxxxx,Wxxxxxx.
ECxxxx,R/Wyy,R/U,Gxxxxxxxxxxxxxxxxxxx.
ECxxxx,R/Wyy,R/U,Bxxxxxxxxxxxxxxxxxxx.
```

EC	Indicates ECS error through CPU port.
xxxx	ECS error incident number (word DSLL of ECS MST). xxxx=0 indicates reporting error from another machine.
R/W	Read or write error indication.
YY	Retry count.
R/U/S	R = recovered error. U = unrecovered error. S = error recovered by single word transfers.
zz....z	ECS address of transfer.
Cxx...x	CM address of transfer.
Wxx...x	Word count of transfer.
Bxx...x	Bad data.
Gxx...x	Good data.

OPERATOR INTERFACE - DSD

DSD and DIS displays are available (similar to CM displays) to display the contents of ECS. Console commands also allow the operator to enter changes in ECS.

The contents of the flag register are interrogated every second by CPUMTR and stored in central memory location EFRL. DSD displays the contents of EFRL on the ECS displays.

Two DSD commands manipulate the flag register bits. These are SFRn. and CFRn. to set flag register and clear flag register where n is the bit number to process.

MACHINE RECOVERY - MREC/1MR

MREC is a CPU program called by the operator which interfaces with the operator via a K display. Its main function is to clear interlocks held by an interrupted machine which have not been cleared by CPUMTR. It also recovers mass storage space on a shared device that is currently not accessible because one of the other machines in the complex has interrupted.

When initiated, MREC displays the status of the devices this machine is sharing and the other machines with which it is sharing them. The operator must indicate the MID of the machine for which the recovery is being attempted as well as the devices to recover. MREC then calls PP program 1MR, which performs the following steps to recover the interrupted machines shared devices.

- Clears any hardware unit reservations that may be preventing other machines from accessing the device.
- Clears device accessed bit for interrupted machine in the DAT to prevent non-level-0 recovery.
- Clears interlocks and user counts in system sectors for the down machine.
- Processes MRT to release local file space held by the downed machine.

It may be necessary to run the machine recovery utility on all machines still up. This is dependent upon which machines were sharing devices with the down machine.

When 1MR attempts to access a device, it may determine that it is unable to because the other machine has left its channel access reserved or the unit reserved.

In the case of reserved channel access, this status is relayed to the operator via the K display. The deadstart button must be pushed on the interrupted machine which will issue a master clear on the channel which in turn clears the reserve. (This handles any reservation situatios for the 841.)

If it is a case of unit reserved, 1MR determines if another machine can access the unit from the other side and notifies via the K display what machine that is. If so, it leaves it up to that machine to initiate its recovery utility (MREC) in order to clear the reserve. Otherwise, this machine issues the grenade function (7054/0011) which clears all unit reserves on the other side.

In the event there are three machines that can access a given unit and one of the machines is interrupted and leaves that unit reserved, it is not always possible for the other machines to determine if they can clear the reserve without affecting each other. If this situation exists, each machine informs the operator. The only case of concern is when both machines are on the opposite side of the unit (connected to a different controller) from the interrupted machine, in which case either machine could clear the reserve. Therefore, the operator must specifically direct the utility on one of the machines to clear the reserve.

If it is found that the interrupted machine had a checkpoint request pending, that request is transferred to this machine in order to have it satisfied. The ECS copy of the MST/TRT is checkpointed to the device.

When the machine recovery utility is required to perform functions that are otherwise illegal for this machine (such as drop tracks reserved to another machine), in order to recover that machine's mass storage, a special bit is required to be set on the pertinent monitor functions which indicates that this function should be allowed to execute whereas it would otherwise be illegal.

This section contains descriptions of several features of NOS that are specifically concerned with reliability, availability, and maintainability (RAM).

The following subjects are detailed.

- List hardware registers in deadstart dump
- S/C register error logging
- CYBER 170 power failure and environment bits
- Unhangable I/O channel code
- Output channel parity error detection/logging
- BATCHIO unit record equipment

LIST HARDWARE REGISTERS IN DEADSTART DUMP

By selecting the express deadstart dump (E) option, the user has access to the S/C register contents of a CYBER 170, and also the contents of the hardware registers at deadstart time, both of which may contain valuable information as to the source of a problem. This is done in the deadstart dump routine EDD.

ROUTINE EDD

EDD has two subroutines, DSC and DCP, to accomplish this. DSC reads up and writes to tape the contents of the S/C registers of a CYBER 170. A four-word header is written to the tape followed by one record for the contents of each S/C register. Each S/C register record has a one-word header identifying the channel of the register. DCP exchanges a package in CM to obtain the CPU hardware register contents at deadstart time. A four-word header is written to the tape followed by one record for the hardware register contents of each existing CPU. Each hardware register record has a one-word header identifying the CPU number. The header and records follow the CM dump record on the dump tape.

DSDI has two options, SC and XP, which utilize these records. The SC option checks for the S/C register label and, if there, prints out the contents of the registers. SC also prints out an informative message if the I deadstart option is chosen. The XP option prints out the CPU hardware register contents.

The S/C registers are written to the dump tape prior to the PP dump, or first on the tape (refer to figure 22-1). This is done in subroutine DSC. The dump includes a four-word header label (figure 22-1) followed by a record for each existing S/C register (figure 22-2). Each record must be a multiple of 4 CM words to be compatible with any type of tape unit. DSC first checks if the machine is a CYBER 170 by checking for channel 16 active. If not active, DSC exits without writing to the dump tape. If a CYBER 170, DSC reads the contents of the channel 16 S/C register and writes the contents to the dump tape. Next, DSC checks for 20 PPs by checking channel 20 active, and if they exist, reads up and writes to the tape the channel 36 S/C register contents.

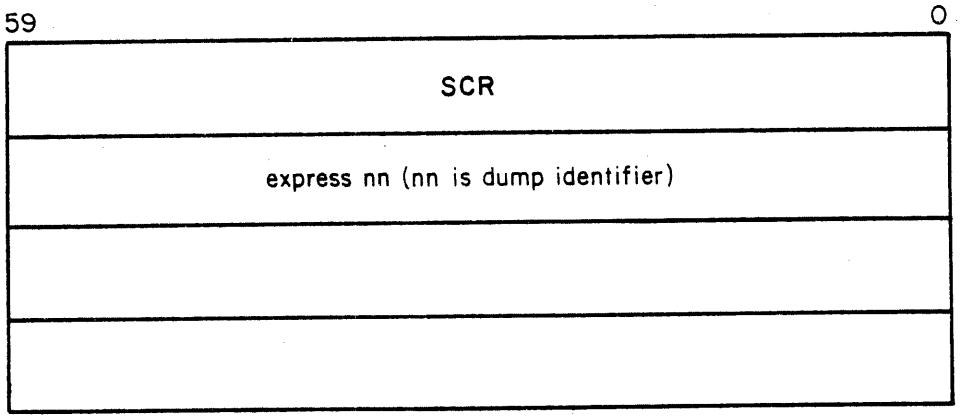


Figure 22-1. Dump Tape Header Label

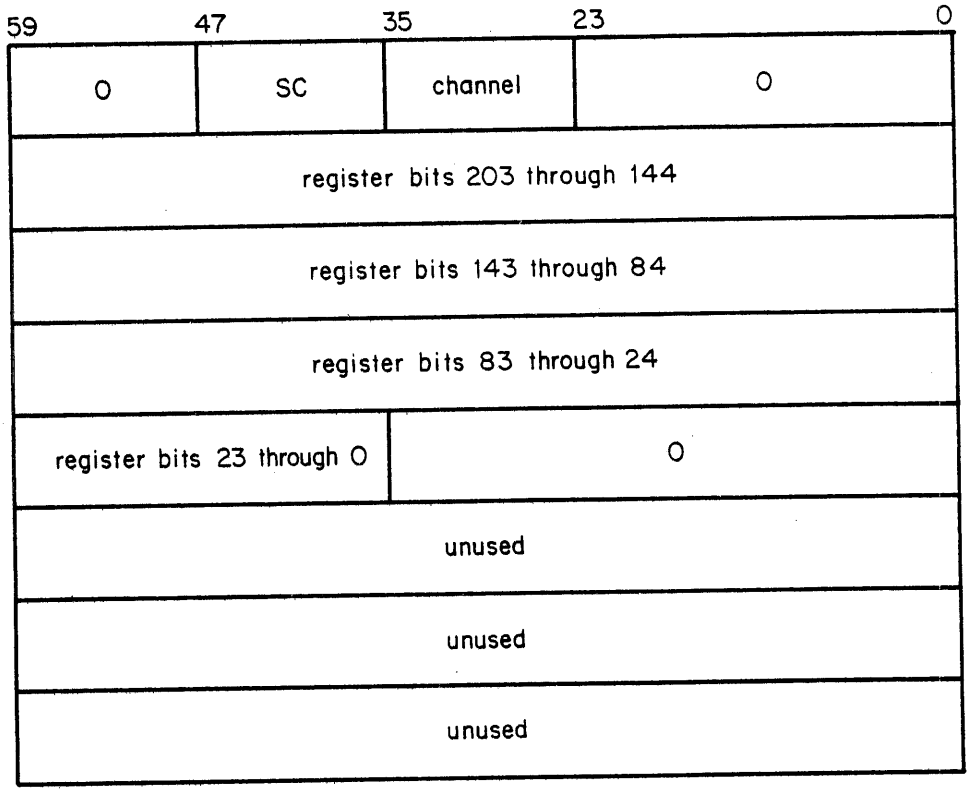


Figure 22-2. Dump Tape Record Format

The PP dump includes a four-word header label (figure 22-3) followed by several records of PP dumps formatted as in figure 22-4.

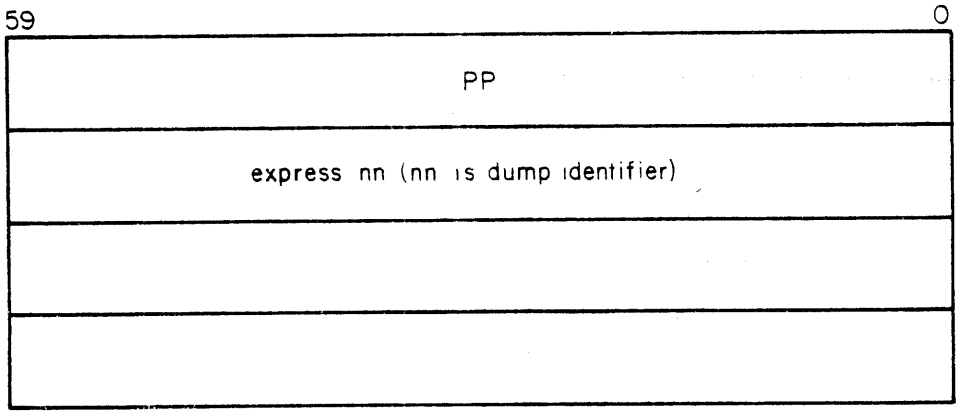


Figure 22-3. PP Dump Header Label

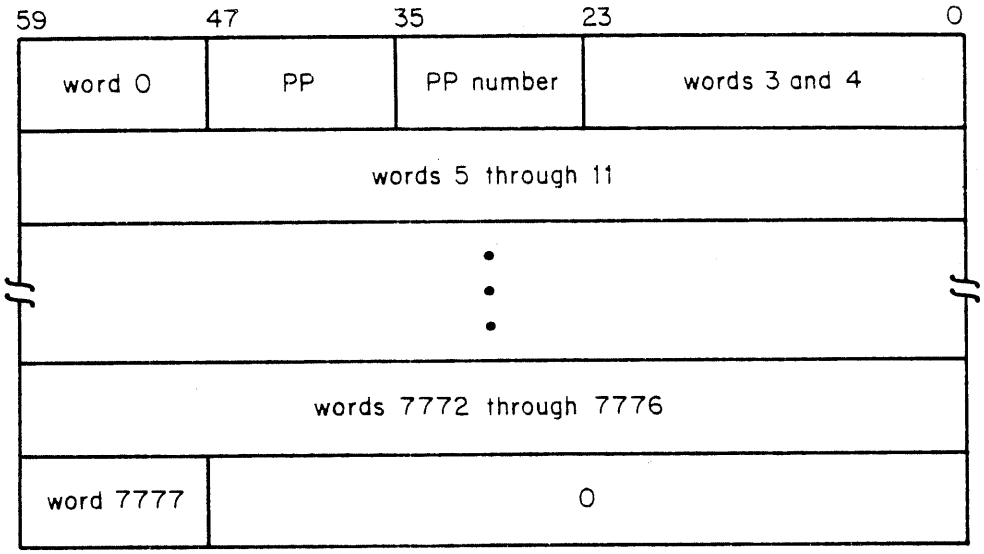


Figure 22-4. PP Dump Format

The CPU hardware register contents are written to the dump tape following the CM dump record (figures 22-5, 22-6, and 22-7). This is done in subroutine DCP. The dump includes a four-word header label followed by a record containing the register contents of CPU 0 and a second record for the register contents of CPU 1, if it exists. An exchange package is exchanged in the CPU to obtain the contents of the registers at deadstart time.

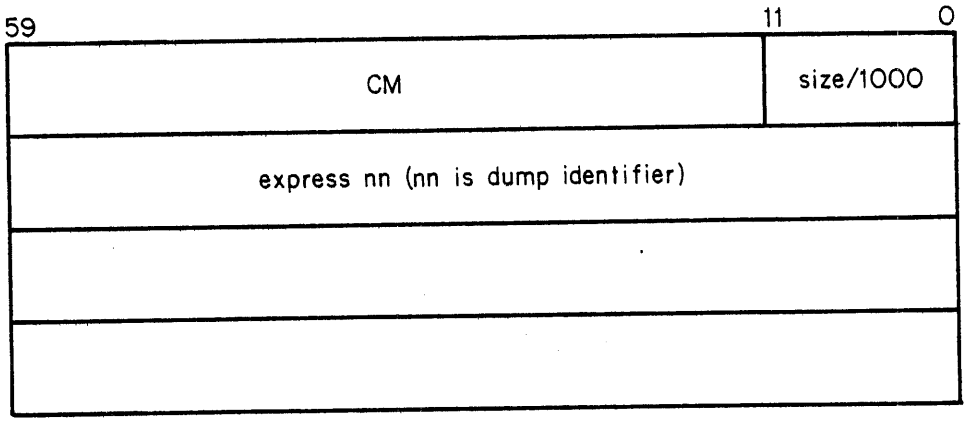


Figure 22-5. CM Dump Header Label

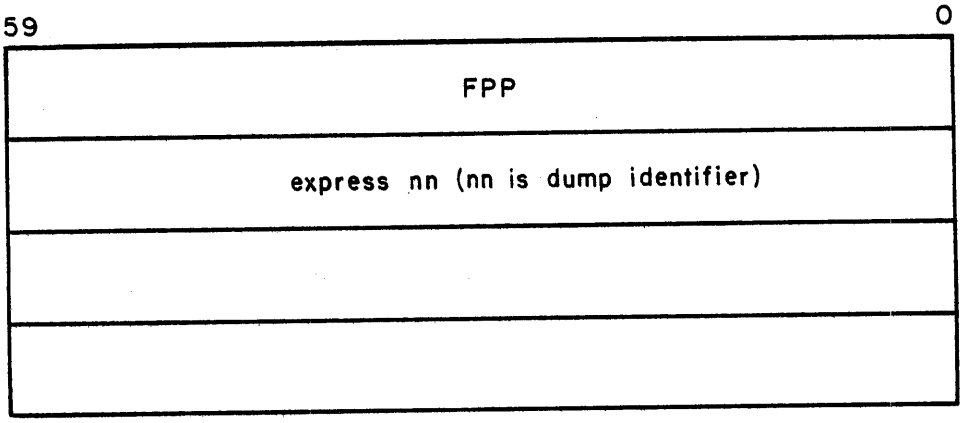


Figure 22-6. FLPP Dump Header Label

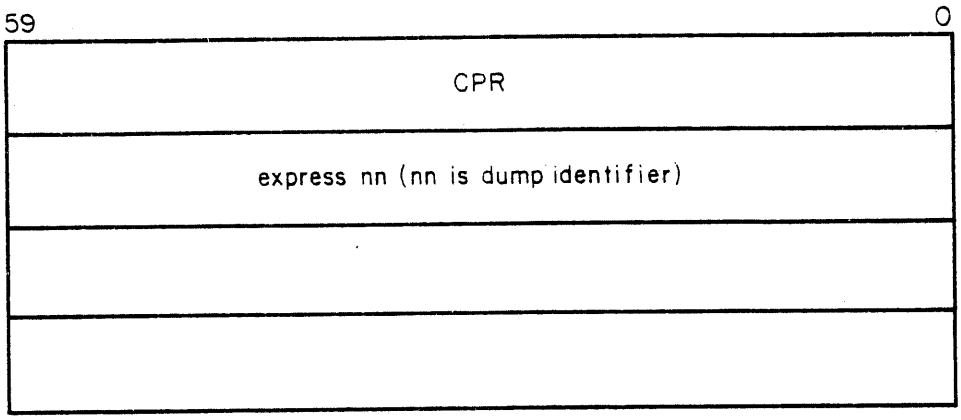
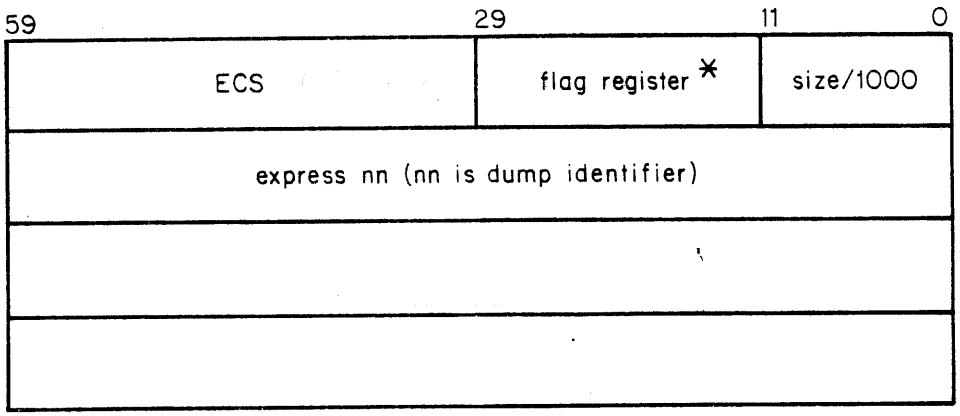


Figure 22-7. CPU Hardware Register Contents

Next, an ECS dump is provided, including a four-word header label (figure 22-8) and one record containing the dump of ECS.



*Not used on CYBER 170 Model 176.

Figure 22-8. ECS Header Label

Finally, a dump of peripheral controller (tape and disk) controlware is provided. This capability, if used, includes a four-word header label (refer to figure 22-9) and one or several records containing the peripheral controlware (figure 22-10).

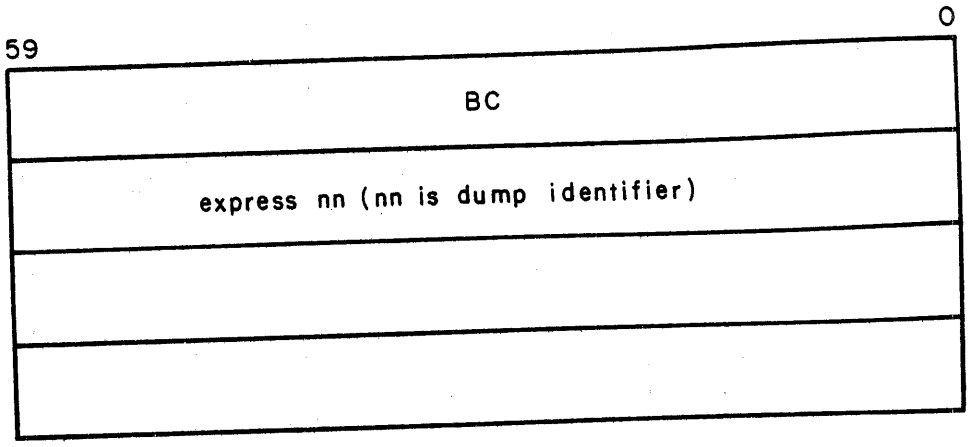


Figure 22-9. Controlware Header Label

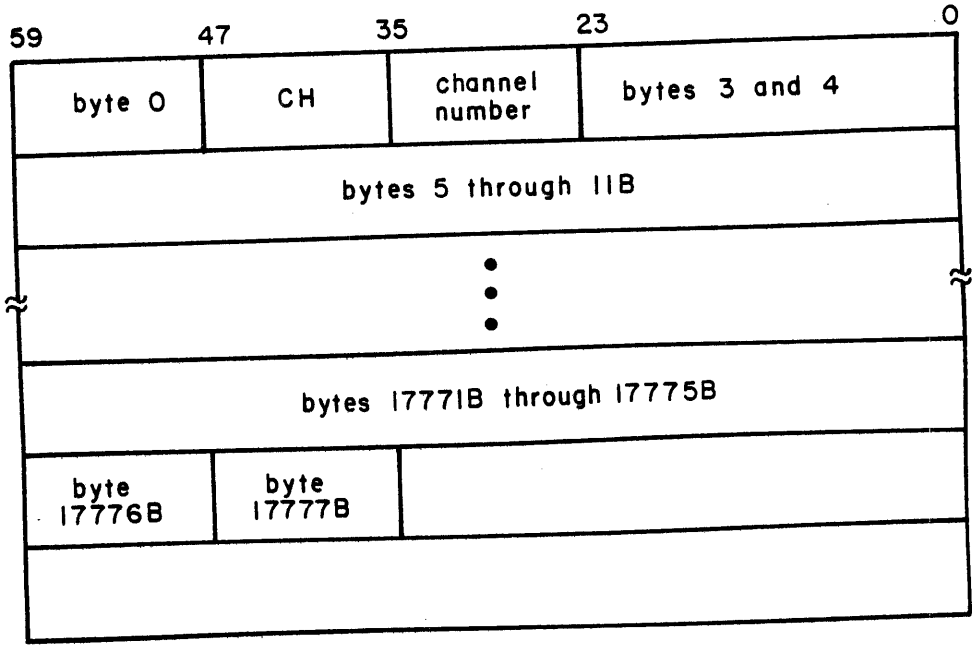


Figure 22-10. Controlware Dump Format

SC causes the S/C register contents to be printed, and XP prints the CPU hardware register contents.

When SC is specified, the dump tape label is checked for SCR. If the label does not contain SCR, an informative message is issued by DSDI. If the label contains SCR, a page header is printed and the label checked for the I option flag.

The next record is checked for a header word, aborting with an error message if it does not exist. If it does exist, the channel 16 S/C register contents is printed in octal format (figure 22-11). Again, the next record is checked for a header word. If one exists, the channel 36 S/C register contents is printed.

When XP. is specified, the dump tape is searched for the label containing CPR. If not found, an informative message is issued. If found, the record following the label is checked for a header word, aborting with an error message if it does not exist. If it does, the record is read, and the hardware register contents printed in the format shown in figure 22-11. Again, the next record is checked for a header word. If one exists, the CPU 1 hardware register contents are printed.

Refer to the NOS System Maintenance Reference Manual for the format of the DSDI call.

S/C Register

```
BITS  59 -   0 0000 0000 0000 0060 0001
BITS 119 -  60 0000 0000 0000 0000 0000
BITS 179 - 120 0000 0000 0000 0000 0000
BITS 203 - 180                0000 0000
```

Hardware Registers

```
P          0  A0     0  B0     0
RA    317700  A1     0  B1     0
FL     50000  A2     0  B2     0
EM*     7007  A3     0  B3     0
RAX        0  A4     0  B4     0
FLX        0  A5     0  B5     0
MA      2400  A6     0  B6     0
EEA** xxxxx  A7     0  B7     0
X0  0000 0000 0000 0000 0000
X1  0000 0000 0000 0000 0000
X2  0000 0000 0000 0000 0000
X3  0000 0000 0000 0000 0000
X4  0000 0000 0000 0000 0000
X5  0000 0000 0000 0000 0000
X6  0000 0000 0000 0000 0000
X7  0000 0000 0000 0000 0000
```

Figure 22-11. Dump Formats

---

* PSD for the CYBER 170 Model 176.

** CYBER 170 Model 176 only.

S/C REGISTER ERROR LOGGING

Routine 1MB, called when an SCR error bit is detected set by MTR, first checks the S/C register for fatal errors and then for bits 36 and/or 37 set. If either is set, the procedure described in the next subsections, CYBER 170 Fatal Mainframe Errors or CYBER 170 Power Failure and Environmental Bits, are followed.

If neither of these conditions are present, 1MB S/C register processing will, for each available S/C register, clear appropriate error bits, issue an error message to the error log, and place the contents of the S/C registers in an appropriate 4-word CM register buffer (S16L and S36L CM buffers). Routine 1MB looks at the bit table TOBR corresponding to the error bits checked by the test/error function (bits 0-39 of each S/C register). If a bit is set in the bit table, and also set in the S/C register, 1MB will inclusively OR the bit into the CM S/C register buffer.

For single error bit SECDED status, 1MB issues error log messages when a threshold value is reached. If a error is detected and the threshold value has not been reached before a time interval expires (time = approximately 1 hour), a message is sent to the error log stating the number of single bit errors that have occurred since the last single bit error entry in the error log. All detected double bit SECDED errors are logged.

When 1MB is finished, it clears bit 59 in CM location SCRL, set by MTR, to reenable S/C register error logging.

CYBER 170 FATAL MAINFRAME ERRORS

This feature minimizes the damage to the system and to user jobs and improves system recoverability where possible, upon the occurrence of fatal errors caused by problems in the CYBER 170 hardware. The term fatal errors refers to errors that are virtually certain to eventually cause uncontrolled system crashes unless they are detected early and guarded against. For purposes of discussion, fatal errors are divided into two groups as listed below, along with the status bits set in the SCR upon their occurrence.

Group 1	Status Bits
CSU address parity error	1 and 2
CSU faults	8 and 9
PP stop on CM read error	0, 3, 182, 183, 14-23
PP stop on PP parity error	14-23

Group 1	Status Bits
Double SECDED error	3 and 183
CMC input parity error	5, 54, 55 and 139

GROUP I ERRORS

Group I errors are normally detectable only because of being recorded in the SCR and only after an indeterminate amount of processing has continued. In addition, the errors are not usually confined to one job's field length, making segregation of the affected area less easy. A level 0 deadstart is normally required after the hardware is corrected. Therefore, improving recoverability will not likely be beneficial but taking immediate steps to minimize the effects of the error is essential. The following sequence is performed.

1. 1CK is called to attempt a checkpoint of mass storage.
2. The system is placed in step mode.
3. The following message is displayed at the system control point.

   FATAL MAINFRAME ERROR.

4. The fatal error bits set in the SCR are not cleared and the operator command UNSTEP is not allowed so that a deadstart is required.

5. SCR analyzes fatal error bits on deadstart and displays a more informative error message than that in step 2.

6. SCR allows deadstart to proceed only after all fatal error bits have been cleared.

GROUP II ERRORS

Group II errors cause an error exit upon occurrence, permitting CPUMTR to immediately take steps to localize the effects of the error. After this is done, efforts can be made to optimize recovery capabilities. The following actions are taken.

1. CPUMTR sets the PEET error flag for jobs getting a mode 20 or 40 error flag.

2. 1AJ checks the SCR for jobs with PEET error flag. If a mode 20 or 40 has occurred, the job remains at the control point while the rest of he system is checkpointed. If the job has faked a mode 20 or 40 error it is aborted with no exit processing and no dump.

3. MTR when moving a control point checks for the PEET error flag and if set validates through the SCR that a mode 20 or 40 error has occurred. If so, the mode request is rejected and 1MB is called.

4. The following message is displayed at the system control point.

   FATAL MAINFRAME ERROR.

5. 1MB initiates a system checkpoint.

6. The contents of the SCR is entered into the error log.

7. After the system checkpoint is complete, the system is placed in step mode.

8. The fatal error bits set in the SCR are not cleared and the operator command UNSTEP is not allowed so that a deadstart is required.

9. SCR analyzes fatal error bits on deadstart and displays a more informative error message than that in step 3.

10. SCR allows deadstart to proceed only after all fatal error bits have been cleared.

NOTE

If CPUMTR or IAF/TELEX itself hardware error exits, the procedure described for Group I errors is followed.

If, contrary to recommended procedure, the installation is operating in parity mode when a CM parity error is encountered, only steps 1 and 2 of the Group II procedure are performed. This is because the remaining steps are dependent on the SCR error bits for initiation and, in parity mode, there is no indication provided for CM parity error in the SCR, nor even any indication that the system is in parity mode.

CYBER 170 POWER FAILURE AND ENVIRONMENTAL BITS

Status and control register (SCR) bit 36 indicates main power failure. On the CYBER 70, bit zero of the interlock register (ILR) serves a similar function. SCR bit 37 indicates an environmental failure. One of the two sequences following is initiated upon deletion of the setting of either or both bits:

If (CYBER 170 only) SCR bit 37 (environmental) is set, but not bit 36, the following steps occur.

1. A message is displayed at the system control point indicating bit 36/37 conditions. The SCR contents are logged in the error log.

2. A system checkpoint is performed. All programs are automatically idled by this action. (During steps 1 and 2, the SCR is continuously monitored and the displayed updated. If mains failure becomes set, the second sequence is initiated immediately).

3. The system is placed in step mode.

4. The SCR continues to be monitored and the display updated. After (and not until) bit 37 clears, the operator may enter an UNSTEP command and normal processing may be restarted using existing facilities. After the UNSTEP, the time at which bit 37 last went clear is entered into the error log.

If SCR bit 36 (main failure) on a CYBER 170 or ILR bit 0 on a CYBER 70 is set, then the following steps occur.

1. A message is displayed at the system control point indicating SCR bit 36/37 or ILR bit 0 conditions.

2. The system is placed in step mode.

3. The SCR or ILR is monitored and the display updated. After (and not until) SCR bits 36 and 37 or IRL bit 0 are cleared, the operator may enter an UNSTEP command and normal processing may be resumed using existing facilities. After the UNSTEP, the time and contents of the SCR when bit 36 was set and the time and contents of the SCR when bit 36 last cleared are entered into the error log. The system is placed in step mode.

SYSTEM FLOW

MTR checks the SCR each time through its main loop and calls 1MB to process any errors detected. MTR checks the ILR bit 0 if running on a CYBER 70.

SCR Bit 37 Only Set

Upon detecting bit 37 and not bit 36 set (or not ILR bit 0), 1MB enters the time and the contents of the SCR in the error log. In addition, a message is placed in buffer MS2W of the system control point area for display by DSD on the B display. From this time until 1MB drops, the SCR (or ILR) is continually monitored. If SCR bit 36 (or ILR bit 0) becomes set, 1MB immediately starts the procedure described in the next subsection.

Routine 1MB next issues a RPPM monitor function requesting 1CK to checkpoint the system. If no PP is available, 1MB enters a five-second delay loop waiting for one to be freed. If a PP is assigned, 1MB monitors 1CK's input register. When this indicates 1CK is no longer active, or if no PP is available and the five-second delay has expired, 1MB sets bit 57 in the SCR parameter word (SCRL). Routine 1MB then becomes dedicated to monitoring the status of SCR bits 36 and 37 (or ILR bit 0) and keeping the system control point message up to date, until such time as both SCR bits 36 and 37 (or ILR bit 0) are cleared. Routine 1MB records internally the time of this occurrence and the contents of the SCR (or ILR). It then clears bit 57 of the SCRL and starts monitoring bit 1 of word INWL (STEP mode) which was set by MTR upon detection of the setting of SCRL bit 57. When INWL bit 1 is cleared, indicating an UNSTEP operator command has been executed, the time and contents of the SCR (or ILR) of the last clearing of SCR bits 36 and 37 (or ILR bit 0) is entered in the error log, and 1MB drops. If, prior to clearing of INWL bit 1, SCR bit 36 or 37 (or ILR bit 0) are again set, 1MB again sets bit 57 in word SCRL and resumes monitoring as above. Meanwhile, MTR, in main loop routine CSC after detecting 1MB already active (existing code), checks SCRL bit 57. If the bit is set, MTR sets STEP MODE (if not already set) and sets the STEP flag, bit 1 of INWL (central memory word 127). This flag, as well as serving as the UNSTEP communication of 1MB, permits DSD to display STEP mode in its normal manner.

The MTR STEP function processor sets bit 1 of INWL and the UNSTEP processor clears it. DSD will not process an UNSTEP function if bit 57 of word SCRL is set.

SCR Bit 36 Or ILR Bit 0 Set

When 1MB detects SCR bit 36 set (or ILR bit 0) indicating main power failure, processing is similar to the description in the previous subsection, except 1CK is not called, and no error logging is performed prior to setting SCRL bit 57. STEP mode is set by MTR immediately and 1MB internally records the time of detection of bit 36 set and the contents of the SCR. Upon the operator entering UNSTEP after bit 36 has cleared, both setting of the bit and clearing of the bit messages are sent to the error log.

UNHANGABLE I/O CHANNEL CODE

It is necessary to establish that regardless of the software technique employed by the drivers to support unhangable I/O channel code, the end result is something less than complete protection against PPs hanging as a result of I/O channel failures. This results because certain hardware failures cannot be compensated for by any form of unhangable channel protect code.

Past experience indicates that hardware problems resulting in I/O channel hangs have been most frequently associated with remote (as opposed to the display controller) peripheral controllers; more specifically, magnetic tape and rotating mass storage controllers. I/O channel hang problems directly relate to the complexity of the hardware required to position to the data, and, to read/write data on the peripheral device. Response timing between a magnetic tape drive and associated controller is more critical than similar response timing between a display console and its controller. This results from the unpredictable variations in timing as a consequence of mechanical motion directly associated with the read/write of data. An electronic peripheral device (that is, display controller) is inherently more reliable in I/O channel communication since the PP-to-controller interaction is dependent entirely on electronic circuit responses as opposed to the combination mechanical/electronic responses associated with other peripherals.

DRIVERS

In general, all drivers verify that an inactive signal was received in response to a function command. This will, in most instances, consist of an issue function subroutine that transmits the function code, checks for inactive on channel, and, if active, initiates a time-out loop while waiting for the channel to go inactive. A time-out results in logging the error in the error log file, deactivating the channel using DCN with bit 5 set and, if applicable, a retry after detection of an error condition. This technique is implemented provided the PP/controller interface timing for the function command sequence being checked allows the increased overhead without degrading I/O performance.

Implementation of additional unhangable I/O channel in other areas of the drivers is dependent upon identification of specific problem areas on the basis of current experience. Another consideration is the availability of space for the code required in the driver to support this concept versus the potential enhancement in reliability of the system. As an example, if additional overlays must be created to support this concept, the increased overhead in time and the increased probability of encountering an error because of loading more overlays may negate any potential enhancement to be derived in support of the unhangable I/O channel code concept.

Routine 1ED

A function issue subroutine is used by the driver. If an inactive signal is not received within 4 major cycles the driver aborts the subsystem after issuing an appropriate message to the error log. Bit 5 is used by all DCN instructions. Routine 1ED checks for complete transfers on I/O instructions.

Routine 1TD

Routine 1TD performs a function timeout on both input and output, with an inactive required within 4 major cycles. Bit 5 is used on all DCN instructions. A message is issued upon a function timeout. Any error aborts the subsystem.

Routines DSD, 1DL

In the operator-initiated channel command code (DSD), the assembly time channel numbers are set to 40 instead of 0 (**) for all IAN, OAN, ACN, DCN, FAN, and FNC instructions. This sets bit 5 in those instructions. Setting bit 5 in IAN, OAN, and DCN instructions in code for display operations and in DCN instructions in code for overlay loading is accomplished by replacing CH by CH+40 in the variable field of these instructions.

OUTPUT CHANNEL PARITY ERROR DETECTION/LOGGING

CYBER 70 systems have no hardware capability of detecting parity errors between the PP and the data channel converter (6681).

CYBER 170 systems can check for parity errors between the PP and the converter. If a parity error is detected by the converter, bit 2 and bit 11 are set in the converter's status word. Bit 11 is unique to CYBER 170 and signifies that channel parity error has occurred between the PP and the converter. Bit 2 set signifies an error between the converter and the equipment.

65x Equipment

When the data channel converter is statused, all 12 bits of status are saved in order to log both bits 11 and 2. Format of the first line of the 65x error message is

```
MT,Ccc-e-uu,vsnxxx,rw,eo,Sss,xxxx,yyyy.
```

In the message, ss is the DCC status. The first digit is 0 or 1, depending on if bit 11 is set. The second digit is bits 2 - 0 of status. Bit 11 is set only if bit 2 is set, indicating transmission parity between the PP and DCC. If only bit 2 is set, parity occurred between the DCC and equipment.

MTS Equipment

If channel parity occurs on a function issue or parameter output, the general status alert bit is set and error code of detailed status is set to the following.

55B Channel parity occurred when a function was transmitted

54B Channel parity occurred when data or parameters were transmitted

The message CHANNEL MALFUNCTION is then issued and the error code appears in the detailed status indicating that channel parity has occurred. If channel parity errors occur while transmitting status functions, no status information is returned from the tape controller.

If channel parity occurs on data output, the following occurs.

- Alert bit is set in general status.
- Channel parity error bit is cleared.
- Error code 54B is set in detailed status error code field.
- The message STATUS ERROR is issued.

Normal write error recovery takes place if channel parity on data output is detected.

BATCHIO - UNIT RECORD EQUIPMENT

CYBER 170 channel parity errors result in the blocking of response to both function and connect codes. Thus, if there is a transmission error between the PP and converter, the PP's channel is not set inactive and the PP may hang when another channel instruction is executed.

For function and connect codes to the peripheral equipment, the question of delay length has a simple solution. Built into the converter is a hardware feature that, after 100 microseconds, deactivates the channel if there is no respone from the peripheral equipment. In addition, bits 0 (reject) and 1 (internal reject) of the converter's status word are set. In order to use this existent hardware feature, timeout loops for function and connect codes to the peripheral unit record equipment are at least 100 microseconds. Again, an inactive channel state occurring before the total delay causes the loop to terminate. All delay loops are designed to accomodate 2X PPs.

BATCHIO also checks bit 2 of the converter status word. If a parity error is found the following ERRORLOG message is issued

```
eqxx,CHyy,Fzzzz REJ P0000,Caaaa,Ebbbb.
```

or (if there are parity errors in either of the status word functions)

```
eqxx,CHyy,Fzzzz FUNCTION TIMEOUT.
```

eq	Equipment type
xx	Equipment number
yy	Channel
zzzz	Function code
0000	Program address
aaaa	Converter status word
bbbb	Equipment status word

This section describes various NOS features and techniques that are available to the site to assist in protecting system operations and user data from curious or malicious users. Many sites make local modifications to many areas of the operating system. As a by-product of these modifications, system and user data, as well as access to them, may be left in an unsecure condition. This section details features and techniques that are used to make the standard NOS system secure.

Many of the security features of NOS are enforced through the job origin concept -- only jobs of a given origin type can perform certain operations. A system origin job (SYOT) is usually initiated by the site analyst from the operator console and may perform privileged tasks that a terminal or batch user may not perform. It is expected that the site exercise the constraints necessary to limit access to the operator console and the privileges associated with being system origin.

There will always be the case of the sophisticated system programmer who may be able to circumvent the security controls of any system. The security controls in the system are meant to protect against the batch or terminal user; the analyst who gains system origin access may still be able to violate security constraints.

SYSTEM ACCESS

This subject covers user access to the system and the accounting (for subsequent billing) of the activities performed by the user.

The control of user access (validation) and user accounting is based on two system permanent files: VALIDUs (validation file) and PROFILa (project profile file). These files identify who can use the system and to what projects system resources may be used and charged. These files can be manipulated through special system jobs that require system origin. The maintenance of these files is described for the site analyst in the NOS System Maintenance Reference Manual. Since these files contain user and project numbers, the knowledge of these values should be limited to the user identified by these values and the site personnel that maintain these files.

The system provides for the suppression and secure entry of passwords and charge and project numbers. The user can use system mechanisms so that these values do not appear in hard copy form in his job dayfile or hard copy terminal output. Thus, these values can remain known only to the user and those responsible for maintaining them.

For example, the user number and password is suppressed from the dayfile of batch job USER statements and, at the user's discretion, blacked-out during a terminal session log-in. Passwords are also suppressed from the dayfile for the PASSWOR control statement as are permanent file passwords contained in permanent file control statement. Both types of passwords may optionally be securely entered by having the passwords read from the INPUT file (at a terminal, the passwords are entered over a blacked-out sequence). Charge and project numbers normally appear in the dayfile, but may be securely entered by the reading of the charge and project numbers from the INPUT file (or entered over a blacked-out sequence at the terminal). Whenever these parameters are securely entered, they do not appear in the dayfile.

SECONDARY USER STATEMENTS

To protect against a user deliberately issuing USER statements until a valid statement is accepted by the system, the site may allow or disallow the processing of more than one USER statement in a job or terminal session by means of an installation parameter. This installation option may be selected through the use of the USERS IPRDECK entry or the SECONDARY USER CARDS DSD console entry. With this option enabled, the user may have more than one USER statement in his job.

SECURITY COUNT

To protect against a user who deliberately attempts to determine valid user numbers by attempting invalid USER statements when secondary USER statements are permitted or by submitting jobs with invalid USER statements, a security count is available in each user's validation file entry. The security count is decremented each time an invalid USER statement is detected. When the security count reaches zero, the user will no longer be granted access to the system. Only the resetting of the security count through MODVAL will give the user access to the system again.

OTHER USER NUMBER PROTECTIONS

The philosophy of user number security followed in NOS is that it should be impossible for a user to determine another user's user number and password. The security count is just one example of this. Other areas where user number validation is done also take steps to prevent the user from determining valid user numbers.

For example, if the user attempts to access permanent files using an alternate user number, a file not found condition is returned if the alternate user number is not valid. This error response does not indicate whether the file was not present under a valid user number or the user number was invalid. Thus the user attempting to violate user number security in this manner does not know whether or not the user number was valid. However, if the file does exist, then the violator knows he has detected a valid user number. This means that the accessor retrieved a file that was public and also correctly guessed the password for the file, if one was required.

SPECIAL USER NUMBERS

User numbers whose user indices are greater than value AUIMX (defined in common deck COMSACC) are called special user numbers. These user numbers cannot be used in log-in sequences or in USER statements. Thus, access to the system cannot be gained using these special user numbers, nor can access be gained to files cataloged under these user numbers (such as VALIDUs) unless the files have been made public or the user explicitly permitted to access them.

NOS uses two special user numbers, the system user number (SYSTEMX) and the library user number (LIBRARY). These user numbers have permanent files like any user number but the files must be cataloged under system origin after an SUI control statement has been entered to set the user index. Other users may access files under these user numbers on an alternate user access basis only.

The site may create its own special user numbers by a force user index (FUI) MODVAL directive to set the user index above AUIMX.

USER ACCESS PERMISSIONS

In the user validation file entry is a word containing certain access permissions. The setting of various bits in the AACW word allows the user to perform certain operations that the site may wish to control. Of particular note is the CSOJ (system origin privileges) permission. The site should exercise caution in selecting which users may have system origin privileges, as the presence of the CSOJ bit usually excuses the job from certain system controls if the system is running in DEBUG mode.

The site should take care in implementing their own access permissions so as not to remove the controls provided by NOS. A complete description of the AACW access word and associated permission bits is provided in section 20.

SPECIAL CONSOLE MODES

The system may be operated under special modes in which normally prohibited operations may be performed by certain users. If the system is placed in DEBUG mode, then many of the controls on user operations are relaxed if the user has system origin privileges (CSOJ validation). If the system is placed in UNLOCKED mode, then certain console commands, including the ability to alter memory, are not prohibited. These special console modes should not be used during normal system operations as their function is to permit software debugging (DEBUG) and hardware maintenance (ENGINEERING) as well as protecting against accidental operator entries that may impact system operation and performance (UNLOCK). The NOS Operator's Guide and NOS Installation Handbook detail the console commands and IPRDECK entries that enable/disable special console modes.

SPECIAL ENTRY POINTS

The system performs certain operations based on encountering special entry points when loading programs from the system library. The special entry points are detailed elsewhere in this manual, but those entry points dealing with security techniques are discussed here.

SSJ= ENTRY POINT

The SSJ= entry point indicates that the program being loaded has special privileges while executing. A variety of system functions are permitted only if the job step executing has the special privileges signaled by the presence of the SSJ= entry point. The operations permitted by jobs with SSJ= privileges include the following.

- Manipulating fast attach permanent files
- Increasing the job's CPU and queue priority
- Performing input/output operations on execute-only files
- Issuing RSB and SIC RA+1 calls
- Manipulating job queues
- Manipulating dayfiles
- Ignoring current family and pack name specification
- Overriding security count
- Accessing the control statement file
- Issuing messages to the account and error log dayfiles

Jobs with SSJ= privileges have the capability to access permanent files and areas of central memory not available to the normal user. Most of the routines that contain the SSJ= entry point are system maintenance utilities such as the permanent file, queue, dayfile, and validation utilities.

The SSJ= entry point also causes the secure system memory status to be set (refer to item on secure system memory in this section).

Another feature of the SSJ= entry point is that all files created by the job while under SSJ= status are created with a special file ID. These files are automatically returned by 1AJ at the completion of the SSJ= portion of the job step. Thus, even if an abnormal termination occurs, the files used by the job do not remain.

The validation file maintenance routine MODVAL is an example of SSJ= usage. MODVAL accesses the validation files, VALIDUs and VALINDs, which are fast attach permanent files under the system user index. Thus, SSJ= privileges are required so that these fast attach files may be accessed. All of the files created by MODVAL are created with the special system ID (SSID) so that they are returned automatically at job completion or termination. MODVAL sets the file ID to zero for all files that are the output of the MODVAL step, if they did not have an ID already specified, such as the output file or the source file generated during an OP=S operation. And since the files accessed by MODVAL are created with the special ID, they do not remain at the control point should MODVAL terminate abnormally.

SSM= ENTRY POINT

The SSM= entry point causes the secure system memory status to be set. The setting of the SSM flag prevents the dumping of any portion of the job's field length. Secure system memory is a NOS feature that is described later in this section.

SDM= ENTRY POINT

Programs that contain the SDM= entry point do not have their control statements issued to the dayfile when the program is loaded. This allows the processing program to suppress fields in the control statement that are privileged in nature. This technique is used by PFILES, for example, in processing permanent file control statements so that passwords may be suppressed from the control statement when it is issued to the dayfile.

VAL= ENTRY POINT

If user validation is enabled (that is, a USER statement must be used in every job), NOS allows only those programs containing a VAL= entry point to execute. The system uses this feature in the ACCFAM and CHARGE programs to ensure that the USER statement and CHARGE statement (if required) are properly processed.

When the validation required bit (bit 17 in word UIDW of the control point area) is set, the control statement being processed must be an entry point in a processor which has a VAL= entry point. If the processor does not have this special entry point, the job is aborted and the diagnostic IMPROPER VALIDATION issued. This mechanism forces the USER statement to be the first statement following the job statement if validation is required (bit 48 set in location SSTL). The processing of the USER statement by ACCFAM clears the validation required bit in UIDW if the user is not required to have a CHARGE statement (CCNR is set in the user's AACW word). When a required CHARGE statement has been processed, the validation required bit is cleared by the CHARGE processor.

SECURE SYSTEM MEMORY

The secure system memory (SSM) feature of NOS prohibits a user from accessing data in central memory after the program that brought the data into central memory has released storage, been storage moved, rolled out, completed, or aborted. Secure system memory involves prohibiting the dumping of privileged field length and clearing of memory when loading a new program or increasing the field length.

This feature prevents access to data that may be left as a residue from other jobs or job steps that have manipulated privileged data or files. For example, if the LIMITS command was aborted and the field length dumped, the dump might expose validation file data to unauthorized access if the secure system memory feature was not available.

PROHIBIT DUMPING

The SSM bit, when set, indicates that the data in the field length is of a privileged nature, and the program executing may not request system functions via RA+1 calls that are processed by programs containing a DMP= special entry point, unless that program has an SSJ= entry point, is system origin (SYOT), or the user has system origin privileges (CSOJ) and the system is in DEBUG mode. Control statements that call DMP= processors may not follow a job step for which the SSM status was set.

The RA+1 calls prohibited by the SSM setting include:

CKP	Checkpoint program
DMP	Dump field length
DMD	Dump field length with display code translation
REQ	Request tape equipment assigned
LFM	REQUEST and LABEL macro calls for tape assignment
PFM	Requests for removable auxiliary devices

The following control statements may not follow job steps with SSM set.

CKP	PBC
DMD	RBR
DMP	WBR
LBC	RESTART
LOC	

The following routines have an SSM= entry point in order to prohibit the next job step from dumping portions of the file's data that may be residue in a buffer used by these utilities.

CATALOG	LIBEDIT
COPYB	VERIFY
COPYC	VFYLIB
EDIT	

Other utilities that manipulate privileged system or user files are protected by their SSJ= entry point such as MODVAL and the permanent file utilities. The main reason these routines have the SSM= entry point is not to protect the data residue from being dumped, but rather to reduce system overhead as the field length does not have to be cleared for the next job step.

The SSM status bit is set by the system whenever a program containing an SSJ= or the SSM= special entry point is loaded or if the program issues a SETSSM macro. The SSM status may not be cleared by any program containing the SSM= entry point. Although the SSJ= entry point causes the SSM status to be set, programs that contain the SSJ= entry point may still make RA+1 calls to DMP= processors.

CLEARING MEMORY

Central memory storage assigned in response to an RFL increase request for all jobs that do not have an SSM= special entry point is cleared when the field length is given to the job. This keeps any data that may be privileged from being dumped. The presence of the SSJ= or SSM= entry points sets the SSM status prohibiting field length dumping. The portion of the field length not loaded when loading a program without SSM= is cleared if the previous job step had the SSM status set. This area is the field length between the last RFL and the field length required to run the current job step.

OTHER DATA PROTECTIONS

The philosophy of data security followed by NOS is that it should be impossible for a user to access the data that may be a residue in his own or another's field length. The secure system memory feature is just one example of this. Other areas where data protection is done is the area of permanent file length errors. If a permanent file length error is detected, the file being retrieved is not given to the user (unless the user is SYOT) as the data currently residing on the tracks specified for the file in error may contain someone else's data.

FILE ACCESS

The permanent file subsystem has a variety of mechanisms whereby a user may control the access to his permanent files. These mechanisms include the file category (public, private, and semi-private) and the file password. The system provides the secure entry and suppressing of permanent file passwords so that these values do not appear on hard-copy output. More detail on the access constraints available for user permanent files is found in the NOS Reference Manual, volume 1.

In addition to the mechanisms that control access to permanent files on the basis of ownership (user number), a method exists which enables a job step to prevent subsequent access to a file it creates. This feature is known as user file privacy and is intended especially for use by application programs. When selected (turned on), files created by the job step have a special ID assigned by routine OBF. Routine 1AJ returns such files prior to initiating the next control statement in the job stream. For information on the use of this feature, refer to the PROTECT macro in the NOS Reference Manual, volume 2.

SYSTEM FILE ACCESS

The SYSTEM file may be accessed in a variety of ways such that it is impossible to deny all users access to it. Therefore, programs and procedure files that are part of the SYSTEM file should not have user numbers, passwords, or charging information in them.

COMMENT SHEET

MANUAL TITLE: CDC NOS Version 1 Internal Maintenance Specification, Volume 2

PUBLICATION NO.: 60454300 REVISION: B

NAME: _____

COMPANY: _____

STREET ADDRESS: _____

CITY: _____ STATE: _____ ZIP CODE: _____

This form is not intended to be used as an order blank. Control Data Corporation welcomes your evaluation of this manual. Please indicate any errors, suggested additions or deletions, or general comments below (please include page number references).

CUT ALONG LINE

AA3419 REV. 4/79 PRINTED IN U.S.A.

NO POSTAGE STAMP NECESSARY IF MAILED IN U.S.A.

FOLD ON DOTTED LINES AND STAPLE

STAPLE

STAPLE

FOLD

FOLD

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 8241 MINNEAPOLIS, MINN.

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 O, MINNEAPOLIS, MINN. 55440
SALES OFFICES AND SERVICE CENTERS IN MAJOR CITIES THROUGHQUT THE WORLD

LITHO IN U.S.A.

CONTROL DATA CORPORATION