60454300

CONTROL DATA CORPORATION

NOS VERSION 1
INTERNAL
MAINTENANCE
SPECIFICATION

VOLUME 3 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

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)

TABLES

TABLES (Continued)

INTRODUCTION

A stimulator enters a hypothetical work load into the system to analyze the effects of such a work load on response time and system reliability. An internal stimulator allows the work load to be entered into the system without the use of any external communications equipment on the related terminals. An internal stimulator is part of the stimulated environment (that is, it runs on the same computer that it is stimulating). An external stimulator allows the work load to be entered into the system using external communications equipment, but without the use of the related terminals. An external stimulator may or may not run within the stimulated environment.

The NOS stimulator software consists of the following programs:

Program	Description
STIMULA	A CPU program that processes input from a session file and K display.
1TS	A PP program called by STIMULA to enter the work load into the system.
1TE	A PP program called by STIMULA to enter a work load into the system using external communications equipment.
DEMUX	A CPU program that processes the stimulator output.

Figure 24-1 illustrates the relationship of the various stimulator modules.

CALLING STIMULA

The type of stimulation to be run is determined by the control statement used in the STMxxxx procedure file. The control statements STIMULA, ASTIM, and NSTIM all initiate the CPU program STIMULA.

STIMULA CONTROL STATEMENT

The STIMULA control statement initiates an internal stimulation that enters a work load into the system through TELEX or IAF. When used with IAF, the work load is entered directly to IAF without going through NAM (refer to figure 24-2). STIMULA can be run as the only front end in the system or in conjunction with live terminals. Only interactive terminals can be stimulated. The format on the control statement is as follows:

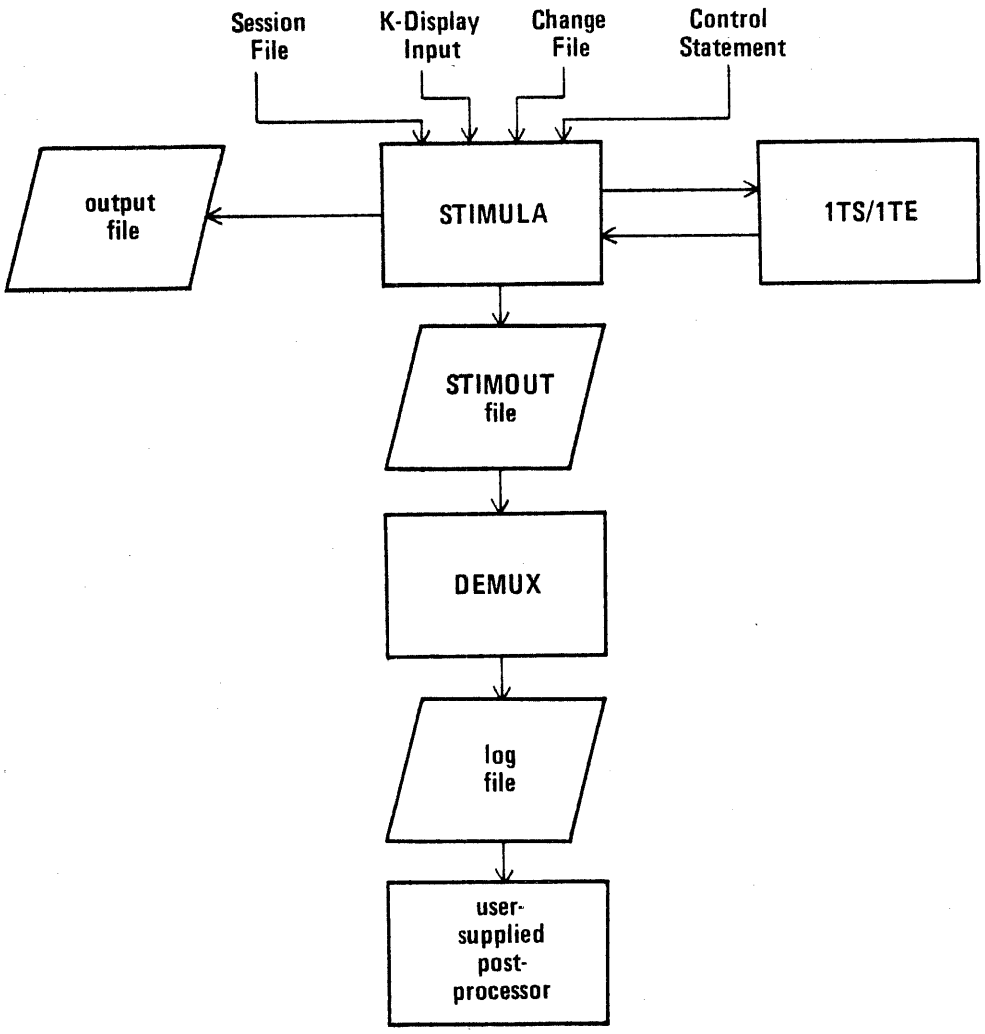


Figure 24-1. Relationship of Stimulator Modules

stimulated. The format on the control statement is as follows:

STIMULA(I=lfn)

lfn	Local file to be used as the session file. If not specified, the initial K display requests the session file name.

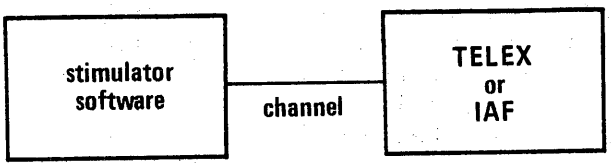


Figure 24-2. Hardware Configuration for STIMULA

ASTIM CONTROL STATEMENT

The ASTIM control statement initiates an external stimulation that enters a work load into the system through TELEX. The stimulation software communicates with a TS type 6676 or 2550-100 multiplexer while TELEX communicates with a TT type multiplexer. The ports of the TS and TT type equipment are hardwired together (refer to figure 24-3). ASTIM can be run as the only front end in the system or in conjunction with live terminals. Only interactive terminals can be stimulated. The format of the control statement is as follows:

ASTIM(I=lfn)

lfn	Local file to be used as the session file. If not specified, the initial K display requests the session file name.

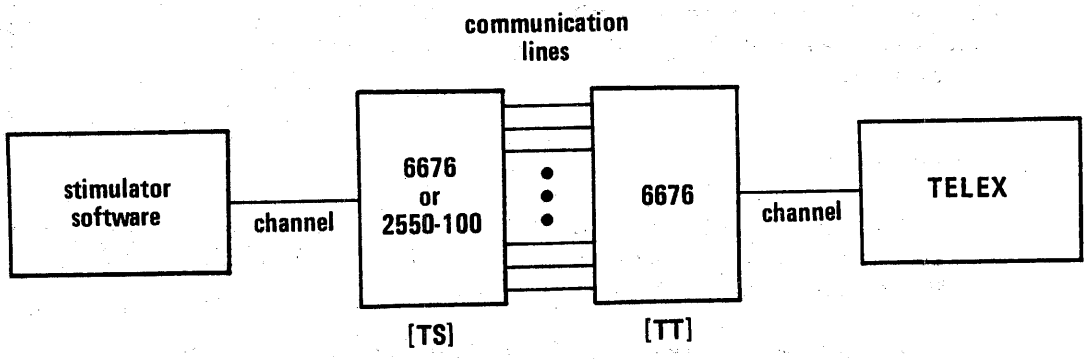


Figure 24-3. Hardware Configuration for ASTIM.

NSTIM CONTROL STATEMENT

The NSTIM control statement initiates an external stimulation that enters a work load into the system through Network Access Method (NAM) and its applications (in particular, IAF). The stimulator software communicates with a TS type 6676 or 2550-100 multiplexer while the NAM software communicates with an NP type communications processor (255x Host Communications Processor). The ports of the TS and NP type equipments are hardwired together (refer to figure 24-4). NSTIM can be run as the only front end or in conjunction with live terminals. Only interactive terminals can be stimulated. The format of the NSTIM control statement is as follows:

```
NSTIM(I=lfn)
```

lfn	Local file to be used as the session file. If not specified, the initial K display requests the session file name.

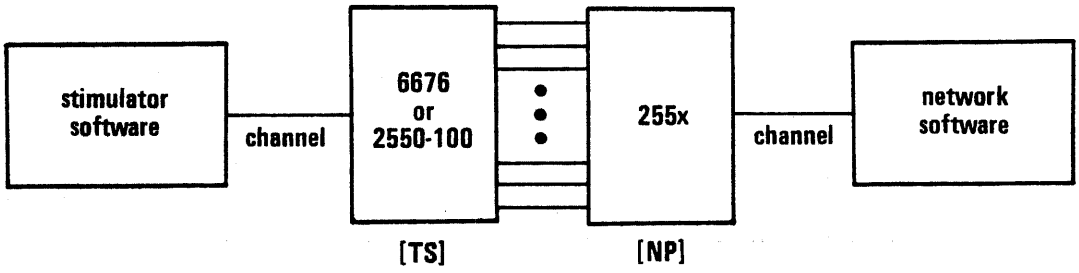


Figure 24-4. Hardware Configuration for NSTIM.

FUNCTIONAL OVERVIEW

The following paragraphs describe what functions STIMULA, 1TS/1TE, and DEMUX have with regards to the NOS stimulation software.

STIMULA

STIMULA is a CPU program that is initiated by the STIMULA, ASTIM, or NSTIM control statements. The major functions of STIMULA include the following.

- Access session file and convert session text into a format suitable for the stimulation.
- Process K-display input and directive file input that describes the terminal characteristics (such as line speed, think time, logout delay, and so on).

- Build tables that will be needed by the stimulator drivers 1TS or 1TE.

- Monitor stimulation driver status during the stimulation run. The output buffers will be flushed when required, the next best task computed, and the end of stimulation detected.

- Once stimulation is complete, the STIMOUT file is written.

1TS AND 1TE

Routines 1TS and 1TE are PP programs that are called by STIMULA to drive the stimulation. Routine 1TS is called if the STIMULA control statement is used. Routine 1TE is called if the ASTIM or NSTIM control statement is used. The major difference between 1TS and 1TE is that 1TS communicates directly with 1TD via a channel. Routine 1TE interfaces with a 6676 or 2550-100 multiplexer. Routine 1TE does not know if it is driving TELEX or Network Access Method applications.

The major functions of 1TS and 1TE include:

1. Initialize control table in STIMULA with maximum number of terminals that can be stimulated.

2. Activate terminals at a rate specified by the AC and AD K-display directives.

3. Transmit data to the host at a rate specified by the IS K-display parameter.

4. Receive data from the host at a rate specified by the LS K-display parameter.

5. Process think time delays as specified by the TT and TI K-display directives and optional think times as set in the session text.

6. Process logout delay as specified by LD K-display directive.

7. Process repeat count processing as specified by the RC and LF K-display directives.

8. Process task requests as set in session text.

9. Process dynamic login requests as set in the session text.

10. Return all data sent or received from the host to the output buffer as specified by the RO K-display directive.

In addition to the above functions, 1TE has the following additional functions.

1. Returns trace data to the output buffer as specified by the TE and TL K-display directives.

2. Processes source line input as specified in the session text.

3. Processes line regulation when encountered.

DEMUX

DEMUX is a CPU post processor program. It is run after the stimulation is complete. Its main functions include the following.

1. Sort terminal data by terminal number.

2. Sort trace data by terminal number.

3. Convert terminal data from ASCII to display code.

4. Convert trace data to readable output.

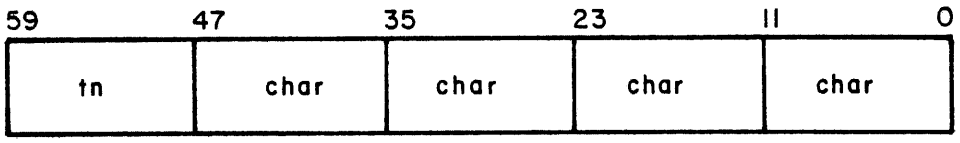
5. Process time stamps found in STIMOUT.

Output from DEMUX can be input into a user supplied program to analyze the stimulation.

STIMOUT FILE FORMAT

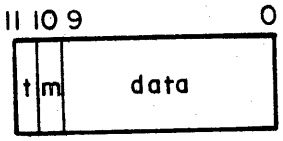
When output is to be recovered (RO=ON K-display parameter), a file called STIMOUT is created by STIMULA. Two types of informtion are found on this file; upline and downline data information and trace information.

The upline and downline information includes all data sent and received by the stimulator with time stamps for when a carriage return was sent by the terminal and a time stamp for when the first character of output was received. The format for this type of information is in the following format.



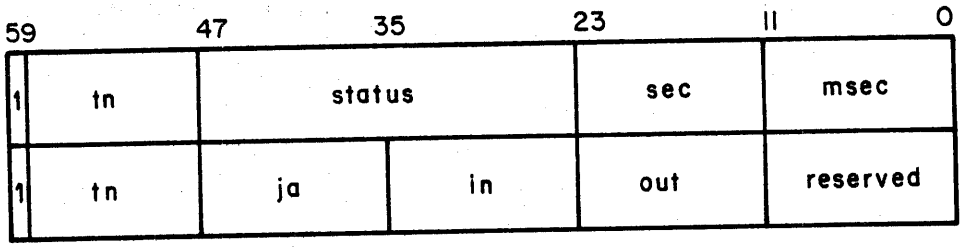
tn Terminal number for which data corresponds to.

char Data information in the following format.



t	Time stamp flag.
m	Millisecond flag.
data	ASCII character in bits 9 through 0 if t and m are zero. Second time stamp if t is 1 and m is 0. Millisecond time stamp if t and m are 1.

Trace information is written to the STIMOUT file if the TE and TL K-display parameters have been used. The trace information is contained in a two-word packet as follows.



tn	Terminal number
status	Byte 0 and 1 from the first word of TE
sec	Value of second clock
msec	Value of millisecond clock
ja	Value of TOCA entry for terminal
in	Input character
out	Output character

Trace information is helpful when debugging 1TE or verifying hardware.

DEMUX sorts information on the STIMOUT file by terminal number and by data or trace information. Data is also converted into readable display code.

EST ENTRIES USED FOR STIMULATIONS

Depending on the stimulator that is being initiated, certain EST entries must be present in the EST table. This section describes the format of these EST entries and when they are required.

Two types of EST entries are used for external stimulation while only one EST entry is needed for internal stimulation. The TT EST entry is always required by TELEX or IAF. The stimulator software does not get information directly from this type of EST entry. The stimulation software requires an EST entry (TS type EST) for external stimulation only. One TS EST entry is required for each 6676 or 2550-100 multiplexer to be driven by the stimulator. Up to eight 6676 or 2550-100 multiplexers, or 12 interactive lines can be driven by ASTIM and NSTIM. If NHP is being stimulated, the TT EST entries are replaced by NP EST entries.

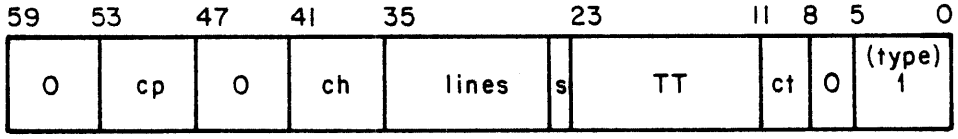
STIMULA EST ENTRY

The following CMR deck entry should be present when the STIMULA control statement is to be used.

```
EQnn=TT,st,ct,1,ch,0,lines.
```

nn	EST ordinal
st	Equipment status ON - equipment available for system use OFF - equipment unavailable
ct	Controller number
ch	Channel number
lines	Number of lines to be stimulated

The general format for this EST entry is as follows.



cp	Control point number
ch	Channel to be used for stimulation
lines	Maximum number of lines that can be stimulated
s	Equipment status: 0 = device available (ON) 1 = device unavailable (OFF)
ct	Controller number
type	MUX type: 0 = real 1 = stimulator

The channel should not have an equipment connected to it. If it does, 1TD/1TN and 1TS may hang. This EST entry must be turned on before TELEX or IAF are initiated.

The stimulator software does not use this EST entry directly. Routine 1TD/1TN finds the stimulator EST entry and assigns it to the timesharing subsystem control point. The maximum number of lines to be stimulated and the channel to be used for 1TD/1TN and 1TS communications are set into bytes 3 and 4 of the first word in the 1TD/1TN message buffer. Routine 1TS searches for the 1TD/1TN message buffer to get this informtion. For this reason, TELEX or IAF must be initiated before the stimulator.

ASTIM ENTRIES

Two types of CMRDECK entries are required to be present when the ASTIM control statement is initiated.

The stimulator software requires the following EST entry.

```
EQnn=TS,st,ct,0,ch,0,lines.
```

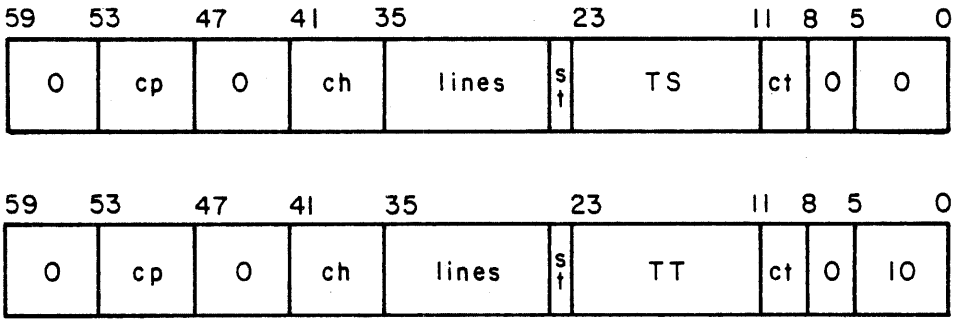
while the TELEX/IAF software requires the following entry.

```
EQnn=TT,st,ct,10,ch,0,lines.
```

nn	EST ordinal
st	ON/OFF status
ct	Controller number
ch	Channel
lines	Maximum number of lines to be stimulated

Two EST entries are required since the stimulator may be running in a different machine than is being stimulated. Also, there are two multiplexers being used; the stimulator is driving one and TELEX/IAF is driving one (refer to figures 24-3 and 24-4).

The central memory format for the TS and TT EST entries is as follows.



NSTIM ENTRIES

Two types of CMR deck entries are required to be present when the NSTIM control statement is initiated.

The stimultor software requires the entry.

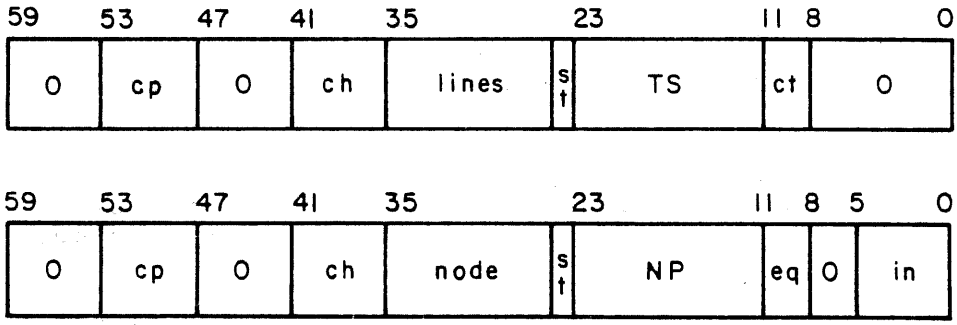
```
EQnn=TS,st,ct,0,ch,0,lines.
```

while the NHP software requires the following.

```
EQnn=NP,st,eq,in,ch,0,node.
```

nn	EST ordinal
st	ON/OFF status
ct	Controller number
eq	Equipment number
in	Index number
ch	Channel
node	Node number
lines	Maximum number of lines

The central memory format for the TS and NP EST entries is as follows.



TABLES USED FOR CPU/PP COMMUNICATION

This section describes the tables that are used for communication between STIMULA and 1TS/1TE. All tables are built by STIMULA and they reside in STIMULA's field length.

TSCR - SCRATCH TABLE

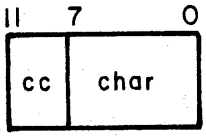
This table is used to contain code, FETs, and buffers that are used once the stimulation has been initiated. The code is responsible for recovering output and selecting the next best task to process. If output is not to be recovered, this table remains empty.

TTER - TERMINAL TABLE

TTER (refer to figure 24-5) is generated for 1TS and 1TE, to be used for control of the stimulation run.

TSTX - SESSION TEXT TABLE

A session consists of 5 bytes of session text per word. Each session is linked to the next session by a control word. There are n words per session. The byte format is as follows.

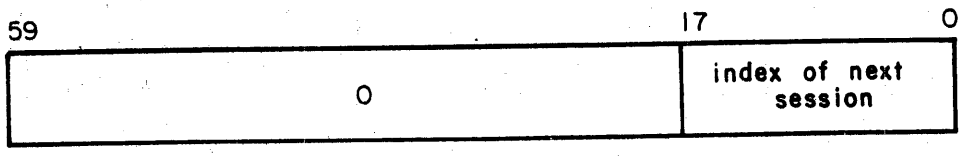


cc Output control code (binary):

100 0	Normal output
010 0	End of output line
001 0	End of session

char ASCII character

The format of the session link is as follows.



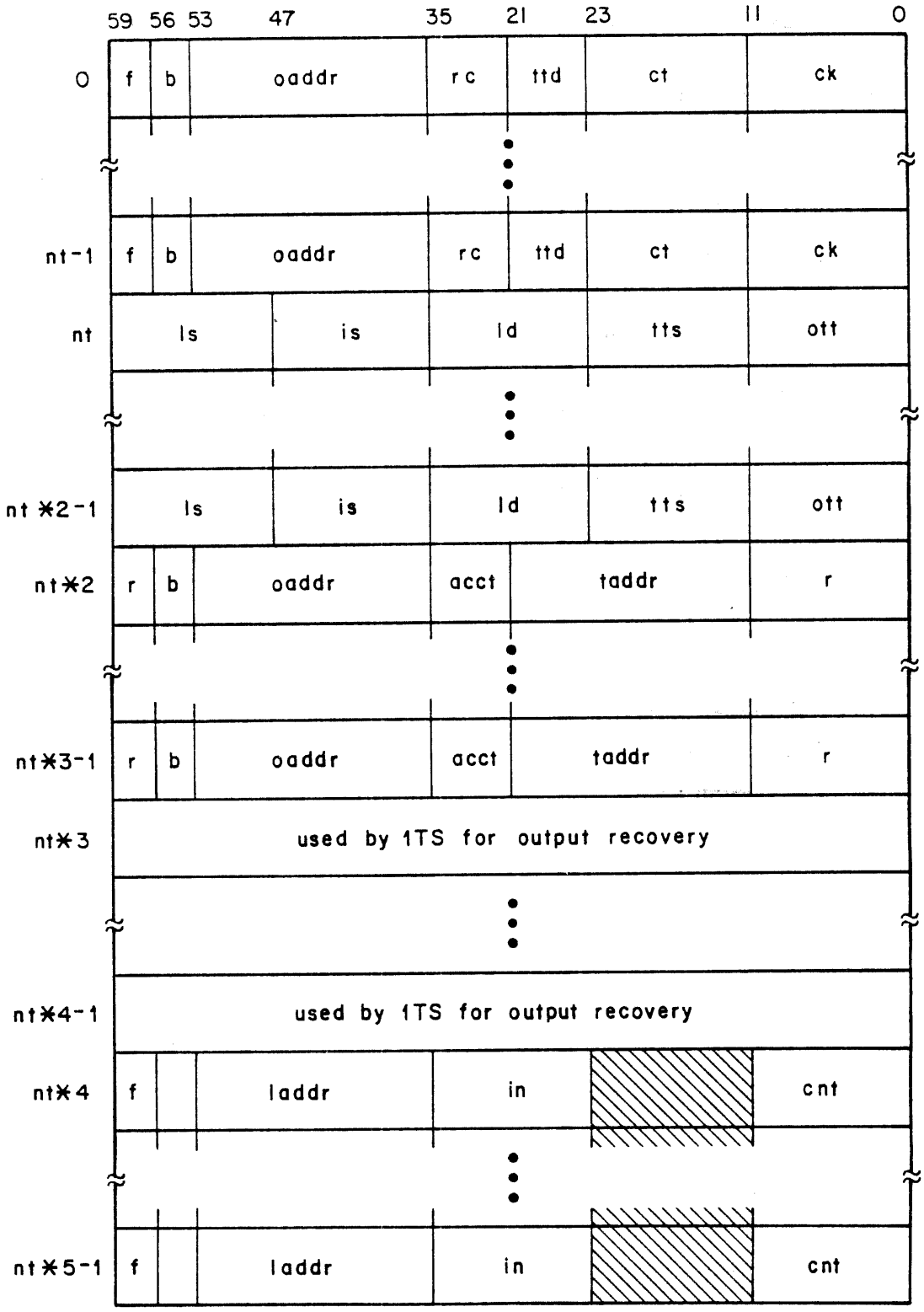


Figure 24-5. TTER Table

nt	Number of terminals (number of entries in table)
f	Flags

Bit	Description
59	Off line
58	Disabled
57	Character encountered

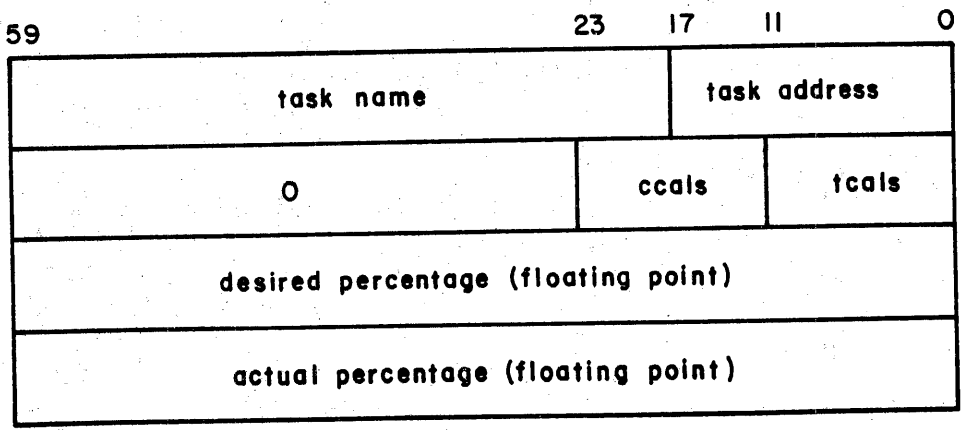
b	Current byte of output data word
oaddr	Address of current output data word
rc	Repeat count
ttd	Think time delay in seconds
ct	Character time in milliseconds
ck	Timing clock
ls	Line speed (character time in milliseconds)
is	Input speed (character time in milliseconds)
ld	Logout delay in seconds
tts	Think time in seconds
ott	Optional think time in seconds
r	Reserved field
acct	Special account number flag
taddr	Address for current task table entry
lb	Byte of output word for lst output line
laddr	Address of output data word for last output
in	Character counter
cnt	Number of times terminal encountered line regulation

Figure 24-5. TTER Table (Continued)

This table is built by STIMULA from the data contained on the session file. It is used by 1TS/1TE during the stimulation to get the next character to transmit to TELEX/IAF. The b and oaddr fields of the first word of the terminal table entry are used to get the next character for output.

TASK - TASK TABLE

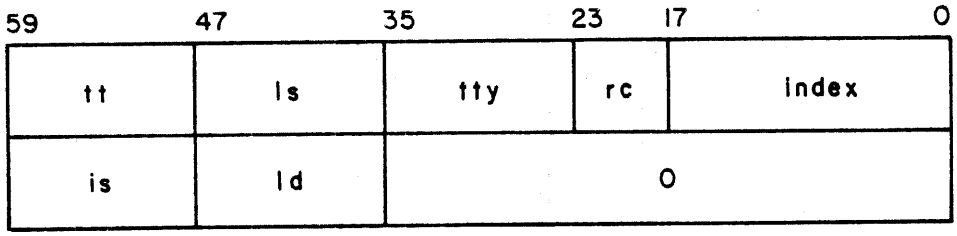
This table is used for selection of the next best task. It is built by STIMULA and is used by STIMULA during the stimulation to determine the next best task to be executed.



ccals	Completed task call
tcals	Total ask call

TSPT - SESSION POINTERS

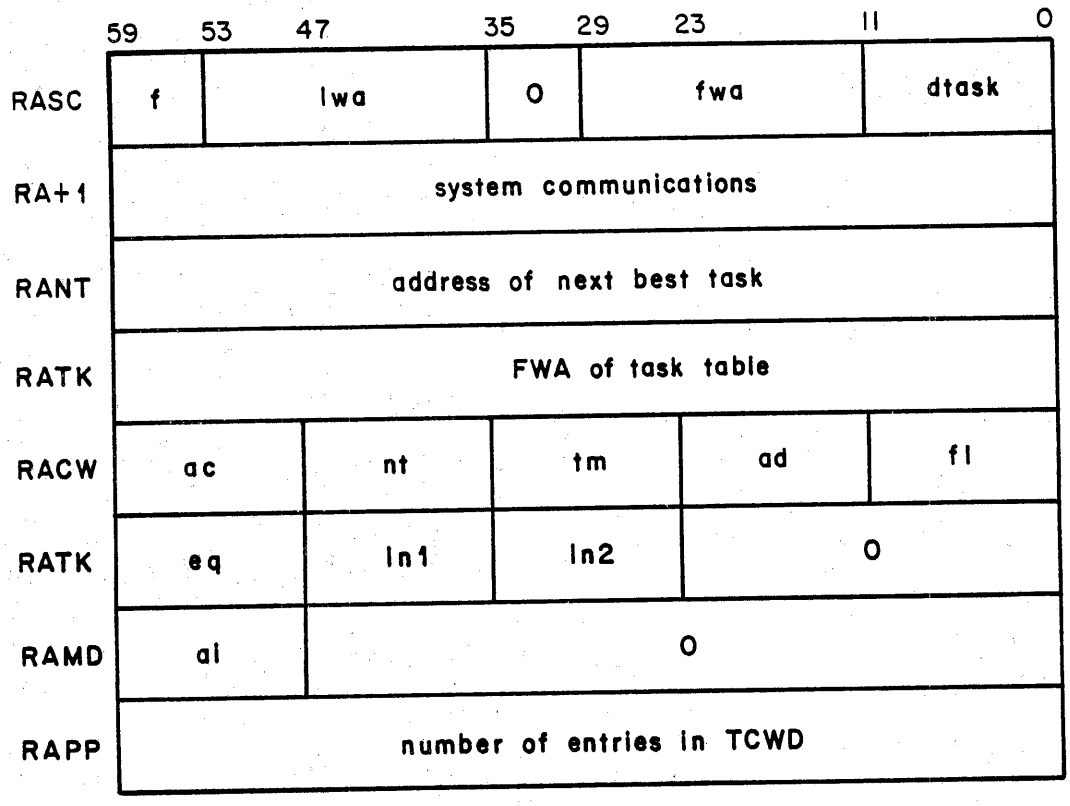
This table is used only during STIMULA initialization. It is not present when the stimulation is in progress.



tt	Think time
ls	Line speed
tty	Terminals assigned
rc	Repeat count
index	Index in TSTX of session
is	Input speed in characters/second
ld	Logout delay in seconds

RA LOCATIONS (STIMULATOR USAGE)

The RA locations table is shown in figure 24-6. These entries contain information that can be used by all active stimulation drivers.

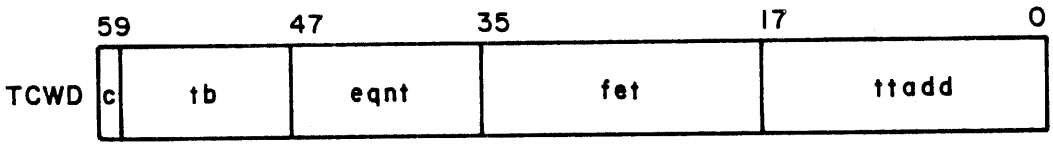


f	0, loop on session file 1, loop on session record
lwa	LWA+1 of session file
fwa	First word address of session file
dtask	Number of default task calls
ac	Activation count
nt	Total number of stimulated terminals
tm	Think time mask
ad	Activation delay
fl	Field length needed for stimulation
eq	Equipment ordinal for multiplexer
ln1	Minimum line number to recover trace information
ln2	Maximum line number to recover trace information
ai	Activation indicator contains control table ordinal of stimulation driver currently activating terminals

Figure 24-6. RA Location Table

TCWD - TABLE OF CONTROL WORDS

The table of control words includes one entry for each stimulation driver called as follows.



c	Completion bit (set by 1TS/1TE) 0 = stimulation driver active 1 = stimulation driver complete
tb	Terminal number bias (set by STIMULA)
eqnt	For 1TS (set by 1TS), 12-bit nt For 1TE (set by 1TE), 6-bit eq and 6-bit nt eq = Equipment EST ordinal nt = Number of terminals for equipment The total number of line regulations for the stimulator driver is returned in this byte at the completion of the stimulation.
fet	Output file fet (0 if no output to be recovered) (set by STIMULA)
ttadd	Terminal table entry address (set by STIMULA)

STIMULA ROUTINES

Figure 24-7 shows the main flow for STIMULA.

The three entry points STIMULA, ASTIM, and NSTIM control the setting of the NPS (network product stimulation) flag, the mode (internal/external stimulation) flag, and whether 1TS and 1TE are called. The NPS flag is set only when the NSTIM control statement is evoked. This flag is used by TSF (translate session file) to determine if an end of line is to be terminated by a carriage return (this would be the case for STIMULA and ASTIM) or by a DC3. The mode flag is used by PRS (preset), ICT (initialize control table) and SAC (set activation count). The mode flag is used to determine what format is used for the control table entries. The NSTIM and ASTIM entry points will also change the 1TS RA+1 requests to 1TE RA+1 requests.

PRS - PRESET ROUTINE

The routine PRS (preset) performs the following functions.

1. Crack the control statement arguments.
2. Call 1TE or 1TS to initialize TCWD (control word table) with the number of terminals and equipment number.
3. Set the maximum number of terminals that can be stimulated into NT, MNT, and DSNT.
4. Set the default activation rate.
5. Modify character translation table if 64 character set is enabled.
6. Rewind the session file if it was specified on the control statement.

TSF - TRANSLATE SESSION FILE

The routine TSF (translate session file) performs the following functions.

1. If the session file was not specified on the control statement, the routine RSF (request session file) is called to attach the session file. The routine RSF will request K-display input from the operator regarding the location and name of the session file.
2. Once the session file has been attached, TSF begins reading the session file and building the TSTX, TSPT and TASK tables. Each record in the session file is processed. If the record is a session record, the session text is converted to ASCII, set into the TSTX table and an entry is added to the TSPT table. If the

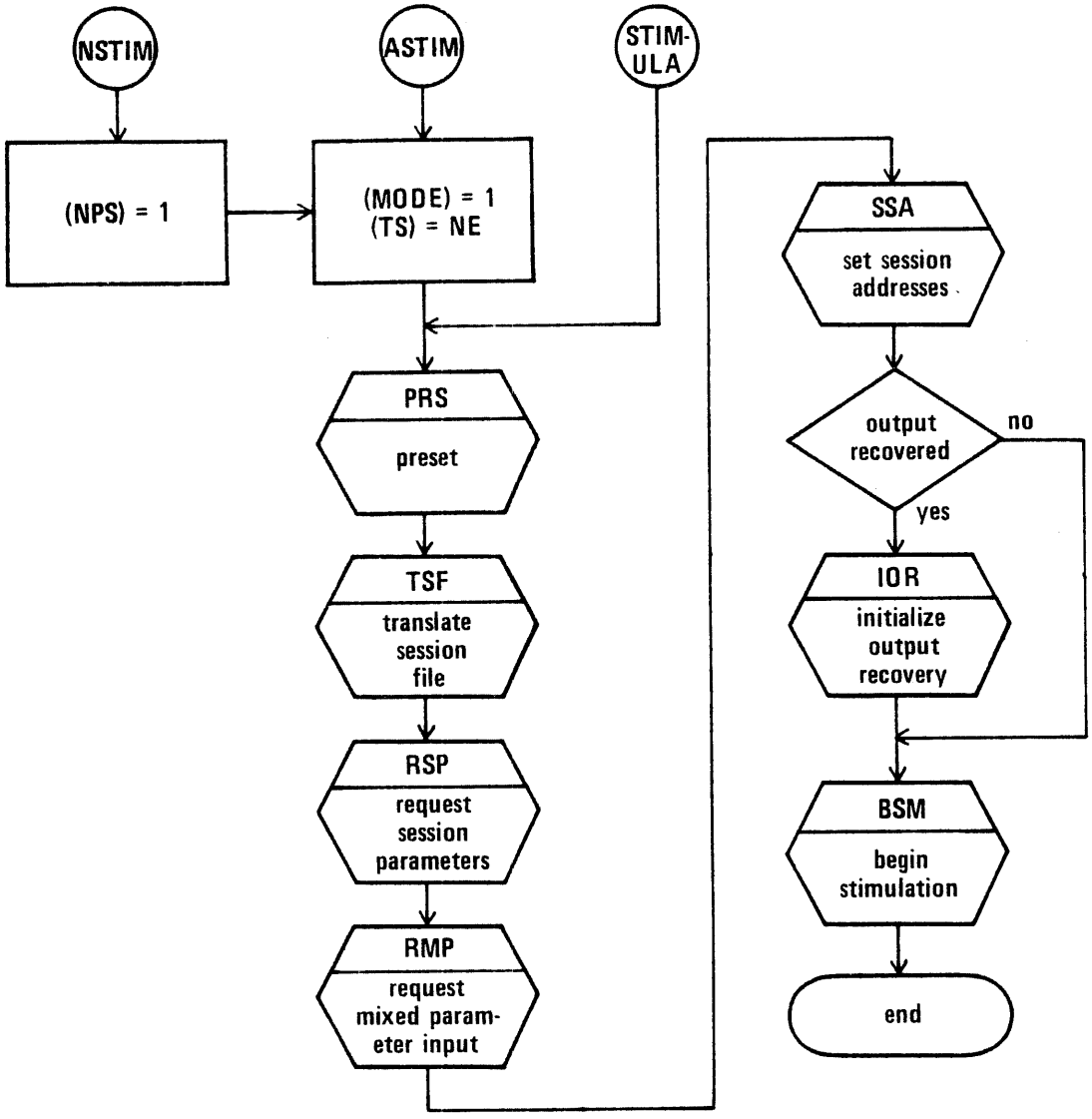


Figure 24-7. STIMULA Flow

record contains a task (the first character of the first word of the record is a $), then the task text is converted to ASCII, set into the TSTX table and an entry is added to the TASK table. The TSTX and TASK tables are completely built by TSF. The TSPT table will be modified at a later time.

3. End of line and end of session control characters are set in the text table as required. These characters will cause special processing to take place in 1TS/1TE.

The session file is returned to the system by the main routine.

RSP - REQUEST SESSION PARAMETERS

The routine RSP (request session parameters) is responsible for requesting K-display input from the operator concerning the session parameter settings. At this time the operator can set the following session parameters.

NT	Number of terminals to stimulate
LS	Line speed
IS	Input typing speed
TT	Think time
TI	Think time increment
AC	Activation count
AD	Activation delay
RC	Repeat count
LD	Logout delay
RO	Recover output option
LF	Loop on session file option
TE	Equipment ordinal for trace
TL	Line number for trace

Values entered at this time affect all terminals that are to be stimulated. On exit from RSP, the values to be used for the session parameters are stored in working storage.

RMP - REQUEST MIXED PARAMETER INPUT

The routine RMP (request mixed parameter input) allows the operator to set the session parameters on a script basis. The operator may enter input to the stimulator with the K display or a directive change file.

RMP calls the routine MXD to move the session parameters that were stored in working storage by RSP to TSPT. The routine DMX is called to build the K display to be used for mixed mode input. The routine KBI is called to process K-display input. The K-display input will be set into the appropriate TSPT entries immediately. On exit, the TSPT table is completely built. No more interaction with the operator is required by STIMULA.

SSA - SET SESSION ADDRESSES

The routine SSA (set session addresses) is responsible for the following functions.

1. Calculate the relocation address for the tables TSCR, TSER, TSTX, and TASK.

2. Set the script pointers into RASC.

3. Assign scripts to the terminals. When a script is assigned to a terminal, information is moved from the TSTP entry for the script into the TTER entry for the terminal. The number of terminals field is decremented in the TSPT entry and the next script is assigned. Script will be assigned to terminals until the number of terminals field is zero. Scripts are assigned in a round robin order. The first script is assigned to the first terminal, the second script to the second terminal, the last script to terminal n and then the first script is assigned to terminal n+1. When all scripts have been assigned any remaining terminals are disabled by setting bit 58 of the first word of the terminal table.

STA - SET TASK ADDRESSES

The routine STA (set task addresses) uses the relocation addresses calculated by SSA to set with the addresses of the task table.

IOR - INITIALIZE OUTPUT RECOVERY

If output is to be recovered from the stimulation run (that is, RO=ON was entered on the session parameter display), then the routine IOR (initialize output recovery) is called. IOR is responsible for setting up the FETs and buffers that are used by the stimulation drivers. The TSCR table is used to hold the code, FETs, and buffers used for recovering output. If output is not to be recovered, this table remains empty.

IOR allocates space in TSCR and moves the recovery code, FETs, and buffers into TSCR. The FET address is set into TCWD (table of control words).

BSM - BEGIN STIMULATION

The routine BSM (begin stimulation) is responsible for setting the control word into RACW, moving the table TSCR, TTER, TSTX, and TASK to low memory and calling the stimulation driver for the first entry in TCWD. The routine ICT (initialize control table) is called to set the terminal bias and the appropriate terminal table address into the control word table.

When BSM is ready to move the tables to low memory, a move loop is moved to high memory. When this move loop is executed, the tables TSCR, TTER, TSTX, and TASK are moved to low memory. Once the tables are moved, the stimulation drivers for the first entry in TCWD is called. The first stimulation driver called is responsible for dropping the CPU and transfer control to the output recovery routine if needed.

Figure 24-8 shows how BSM reorganizes memory in preparation for the stimulation.

RCO - RECOVER OUTPUT

If output is to be recovered, the routine RCO is relocated by BSM. The first stimulation driver is responsible for transferring control to RCO. RCO performs three tasks while the stimulation is in progress:

1. Compute the addresses of the next best task when a task address is needed.

2. Scan all the output FETs being used and call CIO to flush the buffers to disk when the buffers are more than half full.

3. Scan the TCWD to determine when the stimulation is complete.

Once the stimulation is complete, the control point queue priority and CPU priority are reduced. The routine COF (complete output files) is called to write the output data recovered by each stimulation driver to the file STIMOUT.

RCO is flowcharted in figure 24-9.

DESCRIPTION OF 1TS/1TE ROUTINES

Depending on the stimulator control statement evoked, the CPU program STIMULA will call either 1TS or 1TE. Routine 1TS is the stimulation driver that interfaces with 1TD/1TN via a channel while 1TE is a stimulation driver that interfaces with a 6676 or 2550-100 multiplexer.

Both 1TS and 1TE binaries are generated from the 1TS Modify deck. If a DEFINE ASTIM is entered into the Modify directive file, the 1TE binary will be generated. If the DEFINE ASTIM is not present, then the 1TS binary is defined. Differences between 1TS and 1TE include:

1. 1TS searchs for the 1TD/1TN message buffer to determine how many terminals to stimulate and what channel to use. 1TE gets this information from the TS EST entry.

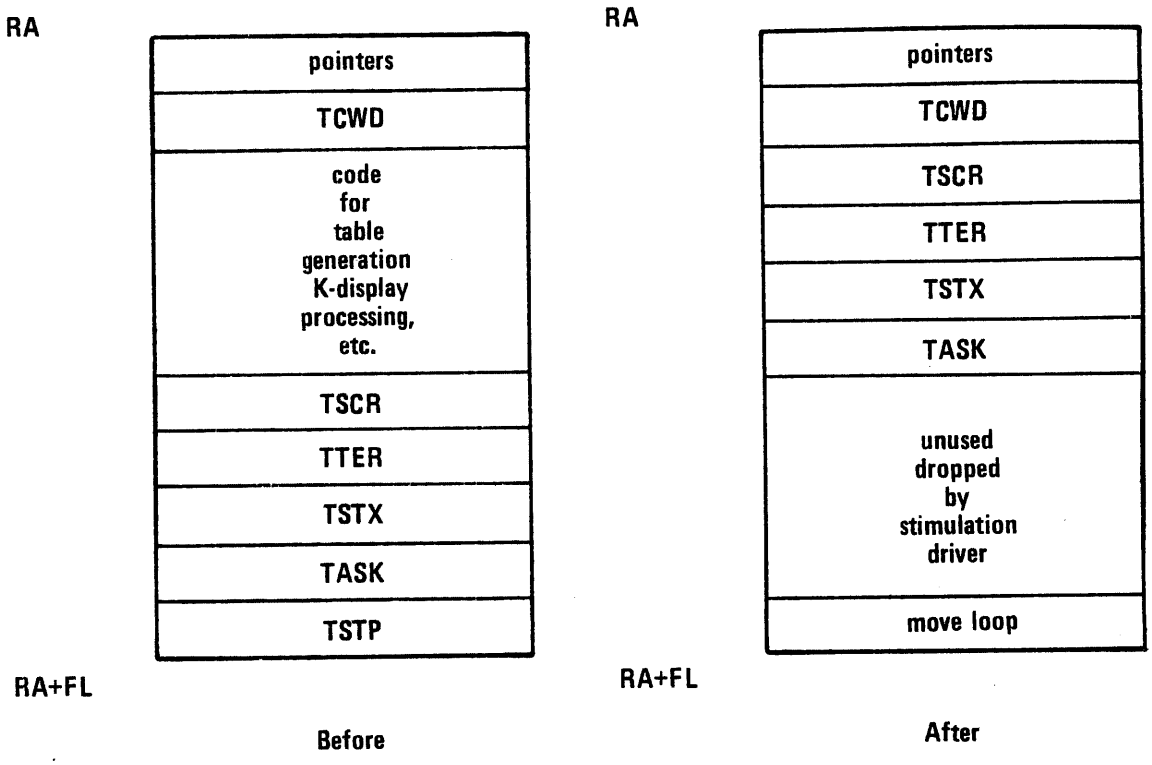


Figure 24-8. BSM Memory Control

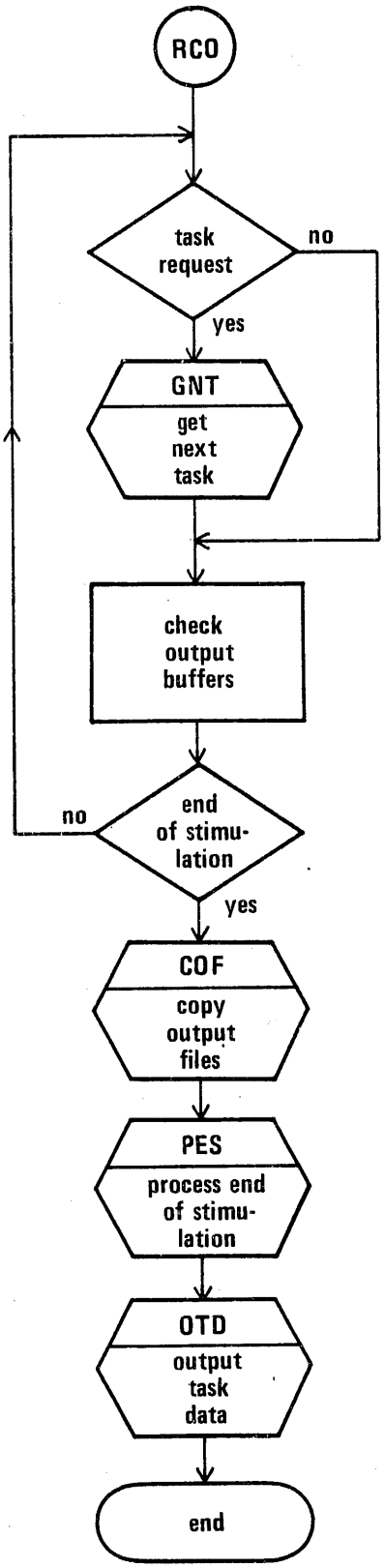


Figure 24-9. RCO - Output Recovery

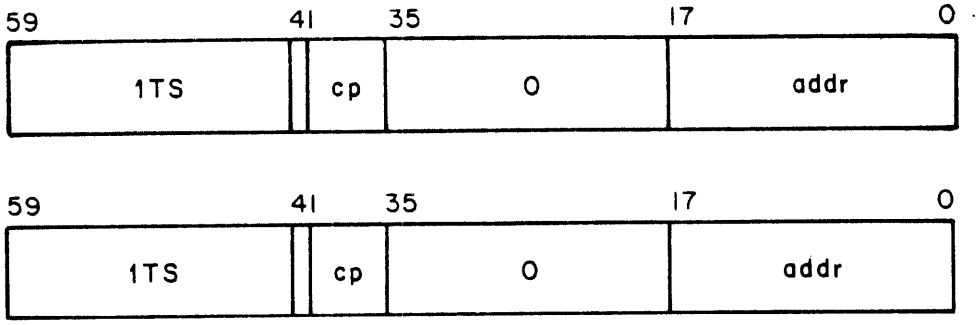
2. 1TE is limited to driving 64 (100B) terminals. 1TS can drive up to 512 terminals.

3. Channel interface in 1TS is geared to 1TD/1TN while 1TE interfaces with a 6676 or 2550-100 multiplexer.

4. 1TE provides a trace mechanism to allow the monitoring of 1TE during run time.

The 1TS/1TE initialization is shown in figure 24-10 and the main loop is shown in figure 24-11.

The formats of the 1TS and 1TE calls are as follows.



addr 0 if 1TS/1TE is being called to initialize the control word table TCWD.
Address of entry in control word table TCWD for this copy of 1TS or 1TE. This call is made to initiate the stimulation.

The CPU program makes an SPC RA+1 request when calling 1TS/1TE. This allows the system to check the validity of the call instead of 1TS and 1TE checking it.

The following discussion describes the more important routines in 1TS and 1TE.

PRS - PRESET ROUTINE

The preset routine is responsible for the following functions.

1. Set channel instructions to use stimulator channel as specified in the EST entry. The routine ISC (initialize stimulator channel) is called to do this.

2. 1TE preset will call the routine IMX to initialize the 6676 or 2550-100 multiplexer.

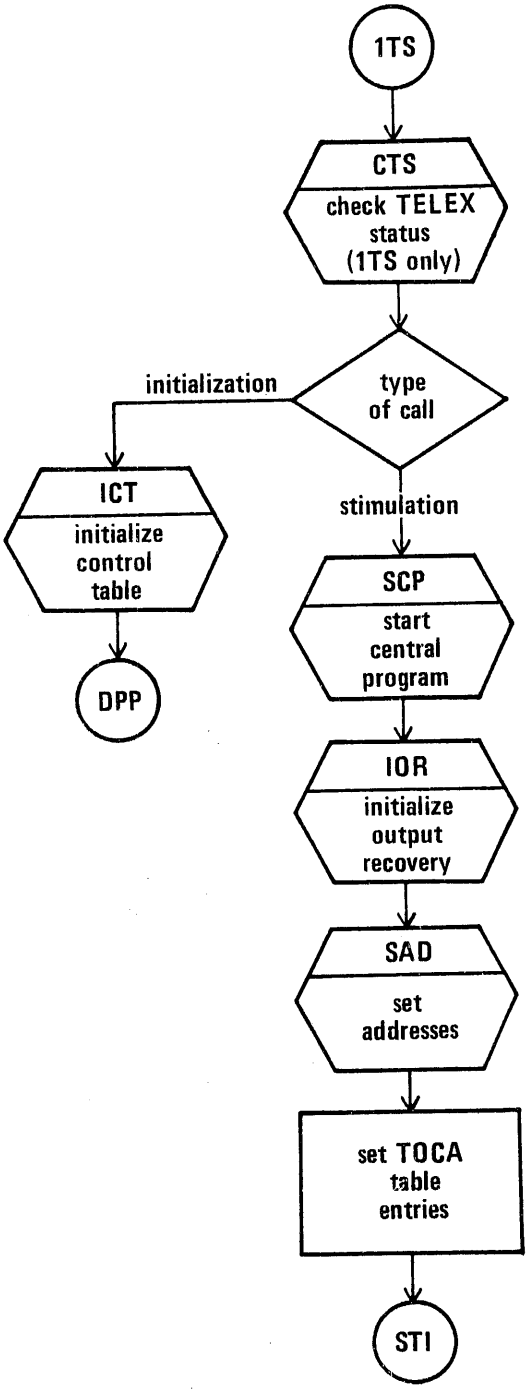


Figure 24-10. 1TS/1TE Initialization

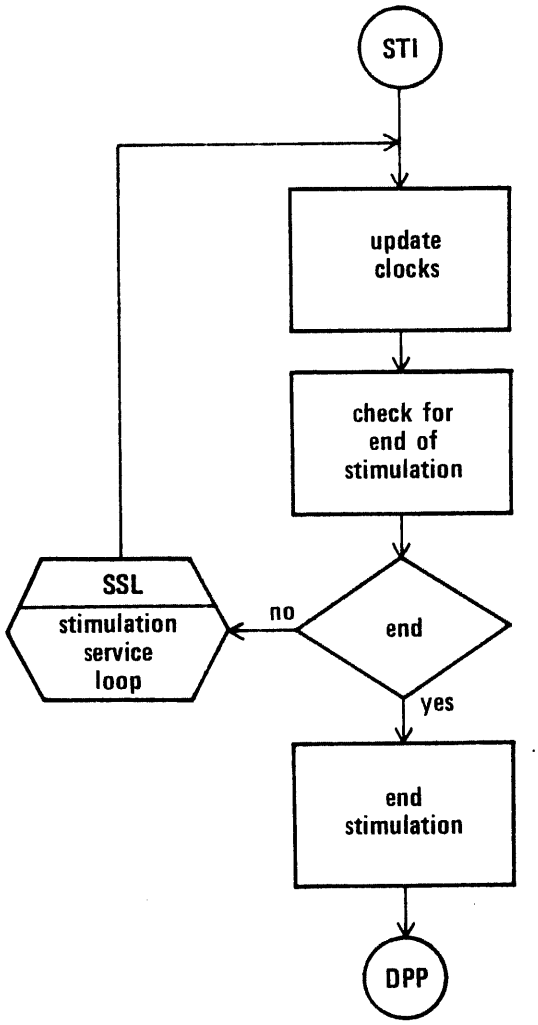


Figure 24-11. 1TS/1TE Main Loop

3. The control word set in RACW by the CPU program is read and the values saved for later use.

4. The routine SCP is called to start the CPU output recovery program and call remaining copies of 1TE.

5. The routine IOR is called to initialize the driver to recover output.

6. The addresses of the terminal table entries are set where requested by the TTADD macro.

7. The TOCA table is initialized. If a terminal is disabled by the CPU program, the TOCA entry for that terminal is set to OFL2, and the active terminal count is decremented by one. If the terminal has not been disabled its TOCA entry is set to LGI.

8. Preset checks the activation indicator in RAMD. Preset does not allow any lines to be activated (that is, initialization is not complete) until the activation indicator is set to the TCWD index for the stimulator driver. This is needed to control the activation of terminals when multiple copies of 1TE are called.

CTS - CHECK TELEX STATUS

The routine CTS (check TELEX status) is used by 1TS only. CTS scans the PP communication area for 1TD/1TN. If 1TD/1TN is not found (TELEX not active), then the control point is aborted. If 1TD/1TN is found, its message buffer is read to get the stimulator channel and terminal count. If no terminals are defined, the control point is aborted.

ICT - INITIALIZE CONTROL TABLE

In 1TS, ICT returns the terminal count read by CTS to the control word table. Only one entry is returned by ICT.

In 1TE, ICT searches the EST for TS type equipments, assigns the equipment to the control point, and returns the terminal count to the TCWD table. The number of TCWD entries is set into RAPP. Routine ICT will abort the control point if no TS equipment is found or no terminals are defined.

SCP - START CENTRAL PROGRAM

The routine SCP is executed only by the 1TS or 1TE that is assigned to the first entry of TCWD. SCP performs the following functions.

1. Drop the CPU.

2. Reduce field length of control point to the minimum required.

3. Set stimulation initiated bit in RASC and issue STIMULATION INITIATED message.

4. If output is to be recovered (RO=ON), the CPU is started by setting the P address to STMO into the control point exchange package and issuing RCPM function. (Refer to RCO description.)

5. Initiate 1TS or 1TE for remaining entries in TCWD table.

SSL - STIMULATION SERVICE LOOP

The routine SSL is responsible for processing all input and output for all active terminals. The routine RTC is called to process the last character input from the channel. Output processing is keyed off of the TOCA table. This table contains the address of the output routine to be used for the terminal. SSL calls the output routines as specified in TOCA.

LGI - PROCESS LOGIN

The routine LGI is responsible for activating terminals at the rate specified with the AD and AC K-display directives. Once the line can be activated, a carriage return is output on that line. After all lines are activated, the activation indicator in RAMD is incremented by one to allow activation of terminals by another copy of 1TE. LGI outputs null characters until an initiate input prompt is received for the terminal.

REJ - REJECT CHARACTER

The routine REJ is used by 1TS to control the rate at which data is received from the host. If data is being received too fast, 1TS will stimulate a character reject by the mux. Routine 1TD/1TN will reissue the character until it is accepted by 1TS.

Routine 1TE does not require this routine since the output data rate is determined by the hardware being used.

TTD - THINK TIME DELAY

The routine TTD generates and processes a think time delay before the next data message is sent upline. The think time is generated by adding a random think time increment (set by TT K-display directive) to a base think time (set by TT K-display directive). The random think time increment is generated by masking the low order bits of the system clock. TTD will

decrement the terminal clock one every second until the clock expires. The input typing speed (IS) is then set in the terminal clock and the TOCA table entry for the terminal is updated to the routine WTC.

WTC - WRITE TERMINAL CHARACTER

The routine WTC is responsible for sending data upline at the input typing speed (set by IS K-display directive). The terminal clock is used in conjunction with the millisecond clock. The first word of the TTER entry for the terminal is read for the location of the next character to be processed. The character is read from the TSTX table and the TTER entry for the terminal is updated. WTC then checks to see if the character read requires special processing (that is, end of line, end of script, optional think time, dynamic login, tasking) and if so, jumps to the appropriate processor. The special character processors will return to either WTC3 or WTC7. If output is to be logged (RO=ON), the routine SDC is called to log the upline character processed.

Special characters processed by WTC include the following.

Character*	Routine	Description
ELCR	EOL	End of line
ELXO	EOL	End of line
ESC	EOL	Escape (End of line)
ETX	EOL	End of text (End of line)
AUTO	SLI	Source line input
BTSK	SNT	Begin task
ETSK	PET	End of task
SACN	SAN	Dynamic login character
OPTT	OTT	Optional think time
ESCR	EOS	End of Script
ESXO	EOS	End of Script

EOL - PROCESS END-OF-LINE

The routine EOL is responsible for cleaning up the terminal table when an end-of-line is encountered. This includes resetting the optional think time, setting the line speed for downline data, clearing the user name flag, and clearing source line input flag. If the last line that was sent was source line input, the TOCA entry for the terminal is set to the routine TTD. This allows the stimulator to send the next upline message without receiving an initiate input prompt. The routines SDC and ETM are called to log the last character sent and a time stamp indicating when it was sent.

---

*These characters are defined in COMSSTM.

EOS - PROCESS END OF SCRIPT

The routine EOS is responsible for processing the repeat count (refer to RC K-display directive) and loop on session file flag (refer to LF K-display directive) when the last character of a session has been encountered. If a repeat count is not present for a given terminal, EOS sets the terminal off line bit in the TTER and jumps to EOL to process end-of-line. If a repeat count is present, EOS will check the loop on session file flag to determine if the same session should be repeated (LF=NO) or the next session in the TSTX table should be executed (LF=YES). The first word of the next session is set into the TTER and the repeat count is decremented. The routine EOL is called to process the end of line.

SLI - SOURCE LINE INPUT

The routine SLI is responsible for detecting source line input to the beginning of an upline message and setting the source line input flag in the terminal table for further processing by EOL. A line is considered to be source line input if the following conditions are met.

1. The first character of a line is a comma.

2. 1TE is the stimulation driver.

GNT - GET NEXT TASK

The routine GNT is responsible for initiating the next best task. To do this the following steps are taken:

1. The current session address and terminal status in the first word of TTER are saved in the third word of TTER.

2. The next best task address is read from RANB. If the CPU program has not set the next best task in RANB, the default task is read from RATK and the task indicator is incremented in RASC.

3. The task address is set in the first word of the terminal table entry.

4. The task address in RANT is cleared if the default task was not used.

5. The number of task call field in the task table is incremented.

PET - PROCESS END OF TASK

PET is responsible for incrementing the task completed counter in the task entry and resetting the session address saved in the third word of the TTER table.

OTT - OPTIONAL THINK TIME

OTT converts an optional think time that is specified in the script into a usable format and stores it into the terminal table. The routine EOL will set the optional think time as the base think time. The optional think time will be used until the initial think time is reset, another optional think time is set, or end of script is encountered.

SAN - SET ACCOUNT NUMBER

The routine SAN converts the special user name character in the script into the first, second, or third digit of the terminal number. The special account number flag in TTER table is used to determine what terminal number digit is to be set. The account number field is cleared at the end of each line.

RTC - READ TERMINAL CHARACTER

The routine RTC processes the last character input from the channel for a given terminal. If it is a character that requires special processing (such as initiate input or hang up phone), the appropriate routine is called. If it is not a special character the TOCA entry for the terminal is set to call REJ (reject character) to control the line speed. If the character being processed is the first output character received other than a line feed or null character since the last upline message, then ETM is called to log a time stamp for when the character is received. The routine SDC is called to log the data character.

In addition to the above, the RTC routine in 1TE interrogates the first several characters of data received from the host for the REPEAT.. message. If this message is received by a terminal, the routine REG is called to process the line regulation.

Character	Meaning	Routine called
IISI*	Initiate input	INI
IISE*	Hang up phone	HNU
REPEAT..	Line regulation encountered	REG

HNU - HUNG UP PHONE

The routine HNU checks the status of the disable flag in TTER that the routine EOS set. If the terminal has been disabled (that is, no more sessions are to be processed for this line), then HNU will decrement the active terminal count and set the terminal offline by setting its TOCA entry to OFL2. No more dialog will take place for this terminal. If EOS set a new

---

*This symbol is defined in COMSSTM.

session address into TTER and left the line enabled, then HNU will restore the initial think time for the terminal and allow the terminal to be activated again by setting its TOCA entry to LGI.

INI - INITIATE INPUT

The routine INI is called when the next upline message is to be sent to the host. INI determines if the terminal just came online by checking its TOCA entry. If it has not just come online, then its TOCA entry is set to jump to the routine TTD to proess the think time delay. If the terminal has just come online, the first upline message will be sent immediately by setting the TOCA entry to WTC.

REG - PROCESS REGULATION

The 1TE routine REG is called when RTC has determined that a line regulation has been encountered. REG will restore the location of the final character of the last message into TTER. This address was saved in TTER by the routine WTC when the first character of the message was sent upline. REG increments the line regulation counter for the terminal in TTER and the total line regulation counter, a flashing B-display message is also issued to notify the operator.

The CPU proram will issue the total number of line regulations encountered by all 1TEs if the output recovery code is enabled.

DATA FLOW

This section describes what happens to the K-display input that is entered on the session parameter display.

LINE SPEED (LS K-DISPLAY PARAMETER)

1. The STIMULA routine SLS (set line speed) is called by the K-display input processor to convert the line speed into binary and place the value in location LS.

2. The STIMULA routine MXD (mixed parameter initial setup) moves the value in LS to the session pointer table, TSPT (byte 1, word 0).

3. The STIMULA routine MXP (mixed mode processor) may update the line speed in TSPT depending on the mixed mode K-display input.

4. The STIMULA routine SSA (set session address) converts the line speed found in TSPT to line speed in characters per millisecond. This value is placed into the terminal table, TTER (byte 0, word 1).

5. The 1TS/1TE routine EOL (process end of line) will set the data rate (byte 3, word 0 of TTER) and with the line speed (from byte 0, word 1 of TTER) whenever the terminal is ready to accept data from the host.

6. The 1TS routine REJ (reject character) checks the terminal clock to ensure data is not being received faster than the line speed specifies. If the line speed is exceeded, REJ will reject the character by sending a 1400 code to the host.

NOTE

The line speed for 1TE is determined by the hardware. The LS parameter should be set appropriately.

INPUT SPEED (IS K-DISPLAY PARAMETER)

1. The STIMULA routine SIS (set input speed) is called by the K-display input processor to set the input speed into the location IS.

2. The STIMULA routine MXD (mixed parameter initial setup) moves the input speed from IS to the session pointer table, TSPT (byte 0, word 1).

3. The STIMULA routine MXP (process mixed mode input) may update the input speed in TSPT depending on the mixed mode K-display input.

4. The STIMULA routine SSA (set session addresss) converts the input speed in TSTP to characters per second. This value is set in the terminal table (byte 1, word 1) of TTER.

5. The 1TS/1TE routine TTD (think time delay) and INI (initiate input) move the input speed into the data rate (byte 3, word 0 of TTER) and the terminal clock (byte 4, word 0 of TTER) when data is to be sent to that host.

6. The 1TS/1TE routine WTC (write terminal character) checks the terminal clock to determine when the next character can be sent to the host. When the clock expires, the character is transmitted and the terminal clock is reset with the data rate value.

LOGOUT DELAY (LD K-DISPLAY DIRECTIVE)

1. The STIMULA routine SLD (set logout delay) is called by the K-display input processors. SLD will set the logout delay in the location LD.

2. The STIMULA routine MXD (mixed parameter initial setup) moves the logout delay from LD to the session pointer table TSPT (word 1, byte 1).

3. The STIMULA routine MXP (process mixed mode input) may update the logout delay in TSPT depending on the mixed mode input.

4. The STIMULA routine SSA (set session address) moves the logout delay from TSPT to the terminal table (byte 2, word 2 of TTER).

5. The 1TS/1TE routine HNU (hung up phone) sets the logout delay in the terminal clock at the end of a session if another session is to be processed on that terminal.

6. The 1TS/1TE routine LGI (login terminal) monitors the terminal clock to determine when the logout delay has expired. The terminal is then activated.

THINK TIME (TT K-DISPLAY PARAMETER)

1. The STIMULA routine STT (set think time) is called by the K-display input processor to set the think time value into location TT.

2. The STIMULA routine MXD (mixed parameter initial setup) moves the think time from TT into the session pointer table (byte 0, word 0 of TSPT).

3. The STIMULA routine MXP (process mixed mode input) may update the think time found in TSPT depending on the mixed mode input.

4. The STIMULA routine SSA (set session address) moves the think time found in TSPT to the terminal table (bits 31 through 24, word 0 of TTER). The think time is also set in byte 3 of word 1 of TTER as the initial think time.

5. The 1TS/1TE routine EOL (process end of line) will update the think time in TTER with the optional think time that was set by OTT (process optional think time) into TTER (byte 4, word 1). EOL will also reset the think time with the initial think time when required.

6. The 1TS/1TE routine TTD (think time delay) computes the user's think time from the think time, think time increment, and the system clock. The user's think time is set into the terminal clock. TTD will monitor the terminal clock until it expires, at which time data may be sent to the host.

THINK TIME INCREMENT (TI K-DISPLAY PARAMETER)

1. The STIMULA routine TTI (set think time increment) sets the think time increment mask into the location TTI.

2. The STIMULA routine BSM (begin stimulation) moves the think time increment mask from TM to byte 2 of RACW.

3. The routine PRS (in 1TS and 1TE) reads RACW and sets the think time increment mask into an LPC instruction in the routine TTD (process think time delay).

4. The routine TTD reads the system clock on channel 14 and uses the mask to generate a random think time increment. This value is added to the think time (as specified by the TT K-display directive) to produce a user think time.

ACTIVATION COUNT (AC K-DISPLAY DIRECTIVE)

1. The STIMULA routine SAC (set activation count) is called by the K-display input processor. SAC will set the activation count into the location AC.

2. The STIMULA routine BSM (begin stimulation) moves the activation count from AC to byte 0 of RACW.

3. The 1TS/1TE routine PRS reads RACW and sets the activation count into the direct cell LC.

4. The 1TS/1TE routine LGI (process login) checks the value of LC. If zero, no additional terminals are allowed to start the login sequence. If nonzero, (LC) is decremented by one and one terminal will proceed with the login sequence. Once the activation delay in the terminal clock expires, (LC) is incremented by one to let the next terminal start its login sequences.

   The maximum number of terminals that can be in the login sequence is equal to the activation count.

ACTIVATION DELAY (AD K-DISPLAY DIRECTIVE)

1. The STIMULA routine SAD (set activation delay) is called by the K-display input processor to set the activation delay into the location AD.

2. The STIMULA routine SSA (set session address) presets the terminal clock in the TTER entries with the activation delay.

3. The STIMULA routine BSM (begin stimulation) moves the activation delay from AD to byte 3 of RASC.

4. The 1TS/1TE routine LGI (login terminal) monitors the terminal clock. When it expires, the terminal may proceed with the login sequence.

REPEAT COUNT (RC K-DISPLAY DIRECTIVE)

1. The STIMULA routine SRC (set repeat count) is called by the K-display input processor to set the repeat count into location RC.

2. The STIMULA routine MXD (mixed mode initial setup) moves the repeat count from RC into the session pointer table, TSPT (bits 23 through 19 of word 1).

3. The STIMULA routine MXP (process mixed mode input) may update the repeat count in TSPT depending on the mixed mode input.

4. The routine SSA (set session addresses) moves the repeat count to see if another session is to be processed. If so, EOS will set the script address in the TTER entry and decrement the repeat count.

LOOP ON SESSION FILE (LF K-DISPLAY PARAMETER)

1. The STIMULA routine SLF (set loop on file) is called by the K-display input processor to set the loop on session file status into the location LF.

2. The STIMULA routine SSA (set session address) moves the loop on session file status from LF to bit 59 of RASC. The first word address and the last word address of the session text table TSTX is also set in RASC.

3. The 1TS/1TE routine PRS reads RASC into the PP buffer RBUF.

4. The 1TS/1TE routine EOS (process end of script) uses the information in RBUF to determine the address of the next session to be executed. If the loop flag is set, the next session in TSTX will be assigned to the terminal. If the loop flag is not set, the same session is repeated. The script addresses in RBUF (RASC) are used for table wrap around purposes.

RECOVER OUTPUT (RO K-DISPLAY DIRECTIVE)

1. The STIMULA routine SRO (set recover output) is called by the K-display input processor to set the recover output status in the location in RO.

2. The STIMULA routines SSA (set session addresses) and STA (set task address) checks the recover output

status to compute the location of the session and tasks during the stimulation when building the TASK and TTER tables.

3. The STIMULA routine, STI, calls the routine IOR (intialize output recover) if output is to be recovered. Routine IOR sets up the routines, FETs and buffers to be used during the stimulation in the TSCR table. If output is not to be recovered, TSCR will remain empty.

4. The first copy of 1TS or 1TE called determines if output is to be recovered by checking the output FET address in its TCWD entry. If an address is present, the CPU program is started. This is performed by the routine SCP (start CPU program).

5. The 1TS/1TE routine IOR (initialize output recovery) is called to initialize the PP program for output recovery. The routines affected are SDC (store data character) and ESD (end STIMOUT data).

The following mechanism is used to recover output.

1. The routines WTC (write terminal character) and RTC (read terminal character) receive the next character they are to process. The routine SDC (set data character) is called by WTC and RTC.

2. If output is not recovered, SDC returns unconditionally to the calling routine. If output is to be recovered, word four of the TTER table entry is read. Byte zero of this word indicates the next byte to store the data character. If the word is not full, the data character is stored, byte 0 is incremented, and the word is written back to the TTER entry. If the word is full, the routine SDW (set data word) is called.

3. SDW sets the data character into byte 4 and the terminal number into byte 0 of the fourth word of the TTER entry. The word is then written into the output buffer. If the output buffer is full, the message

   LOST STIMOUT DATA

   is issued and the word is not written to the buffer. SDC finally clears the fourth word of the TTER entry.

4. Time stamps are processed by the 1TS/1TE routine ETM (enter time stamp) by calling the routine SDC. ETM is called by RTC when the first character of a downline message is received or by WTC when an end of line is encountered.

5. The 1TS/1TE routine ESD (end STIMOUT data) is called at the end of the stimulation to flush the data stored in the TTER entries to the output buffer.

6. The CPU routine RCO (recover output) that was started by SCP is constantly monitoring the amount of data in the output buffers. When a buffer becomes more than half full, CIO is called to write the data disk. One file will be written for each copy of 1TS or 1TE that is running.

7. At the end of the stimulation, the CPU routine COF (complete output file) is called to copy all the stimulation output to the file STIMOUT.

8. The CPU program DEMUX is called to process the STIMOUT file.

Checkpoint/restart is composed of two CPU routines, CHKPT and RESTART, which use special entry points described in section 5. Special entry points allow these routines to access the privileged file DM*.

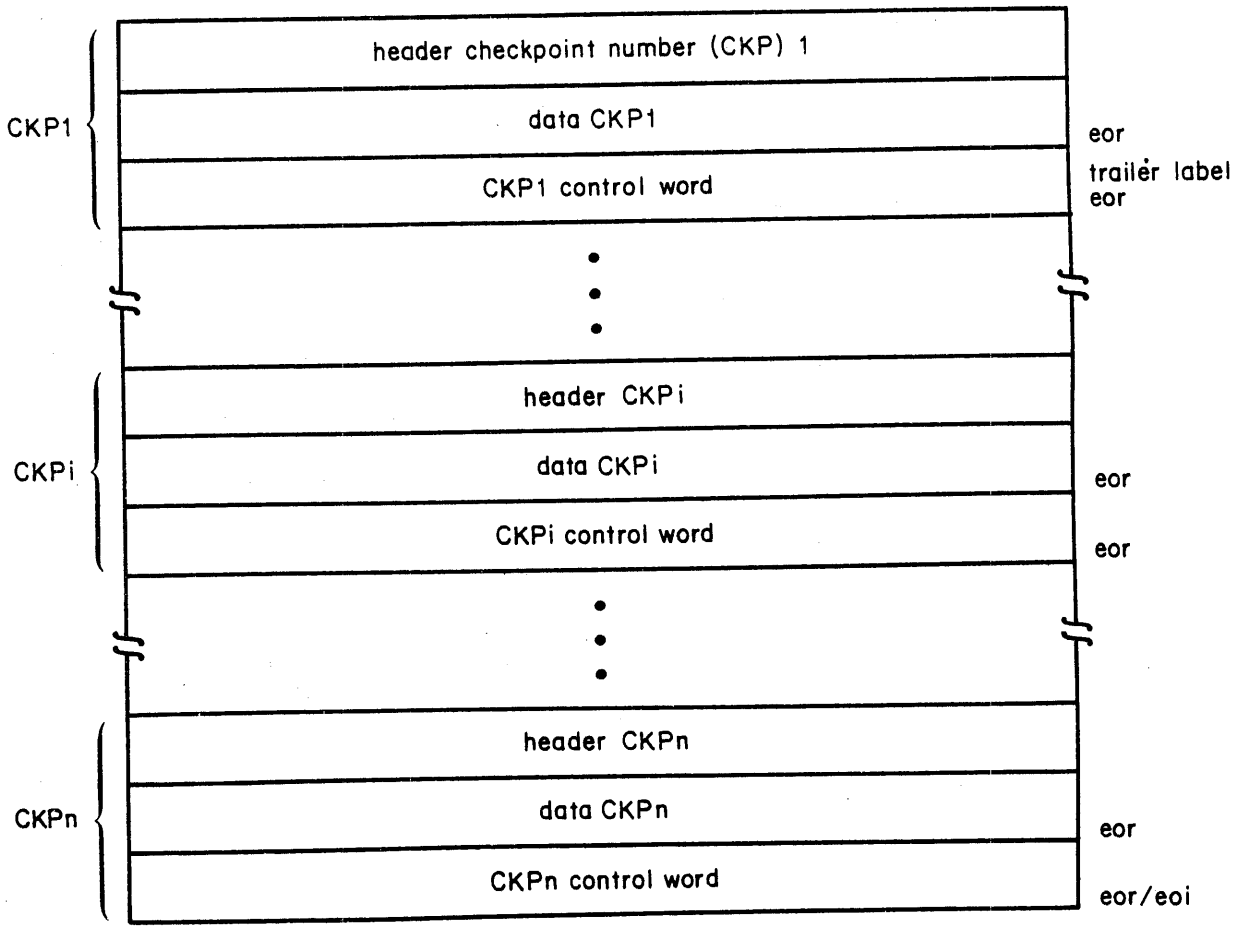
The user can checkpoint a program's progress for later restart by control statements, macro call, or RA+1 request.

By using the RESTART control statement the user can restart a job from any point that was previously checkpointed.

All calls and the use of these routines are described in the NOS Reference Manual, volumes 1 and 2.

CHECKPOINT FILE

The checkpoint file consists of a series of checkpoint (CKP) records. Each checkpoint dump is separated by an EOR, a checkpoint control word, and another EOR. An EOI terminates the file. A multicheckpoint file is formatted as follows.



There may be one or more CKPs on the file. If two files are used simultaneously, the CKPs alternate on the files. The files must be requested with the CK or CB option on the REQUEST, LABEL, or ASSIGN control statement.

There are five parts to each CKP dump (one large record).

- The header word
- The file table
- A copy of each of the files
- A copy of the DM* file of the requesting job
- A control word (trailer label) embedded between two EORs.

The file is written in control word blocks, using the READW and WRITEW macros. Buffers are always filled before transferring to disk, except for the final control word. Buffers are 1000B words in length which is 10 disk PRUs or 1 tape PRU. Therefore, there are no short PRUs and no EOR, EOF, or EOIs except on the control word block.

In order to indicate the EOR, EOF, and EOIs which occur in the data, a series of control words are used. These control words are:

1. 10002B; header.
2. 20nnnB; file table.
3. 30nnnB; start of a block which contains no EOR, EOF, or EOIs (file copy section).
4. 31nnnB; an EOR occurs at the end of the next nnn words.
5. 32nnnB; an EOF occurs at the end of the next nnn words.
6. 33000B; EOI flag. No data may occur directly before this flag.

NOTE

The following control words indicate that an EOR, EOF, or EOI follows the nnn words of data in the DM* file.

7. 40nnnB; start of a block which contains no EOR, EOF, or EOI (DM* file).
8. 43nnnB; last block containing DM* file.
9. 50000B; end of CKP dump.

Each CKP dump is one record followed by a control word record. Each block on the file is nnn+1 words in length, where nnn is the number of data words preceding this indicator. The maximum physical block size is 1000B words or 777B+1 words. The value of nnn varies due to EOR, EOF, and EOI occurring in the data. Figure 25-1 shows the format of one CKP file.

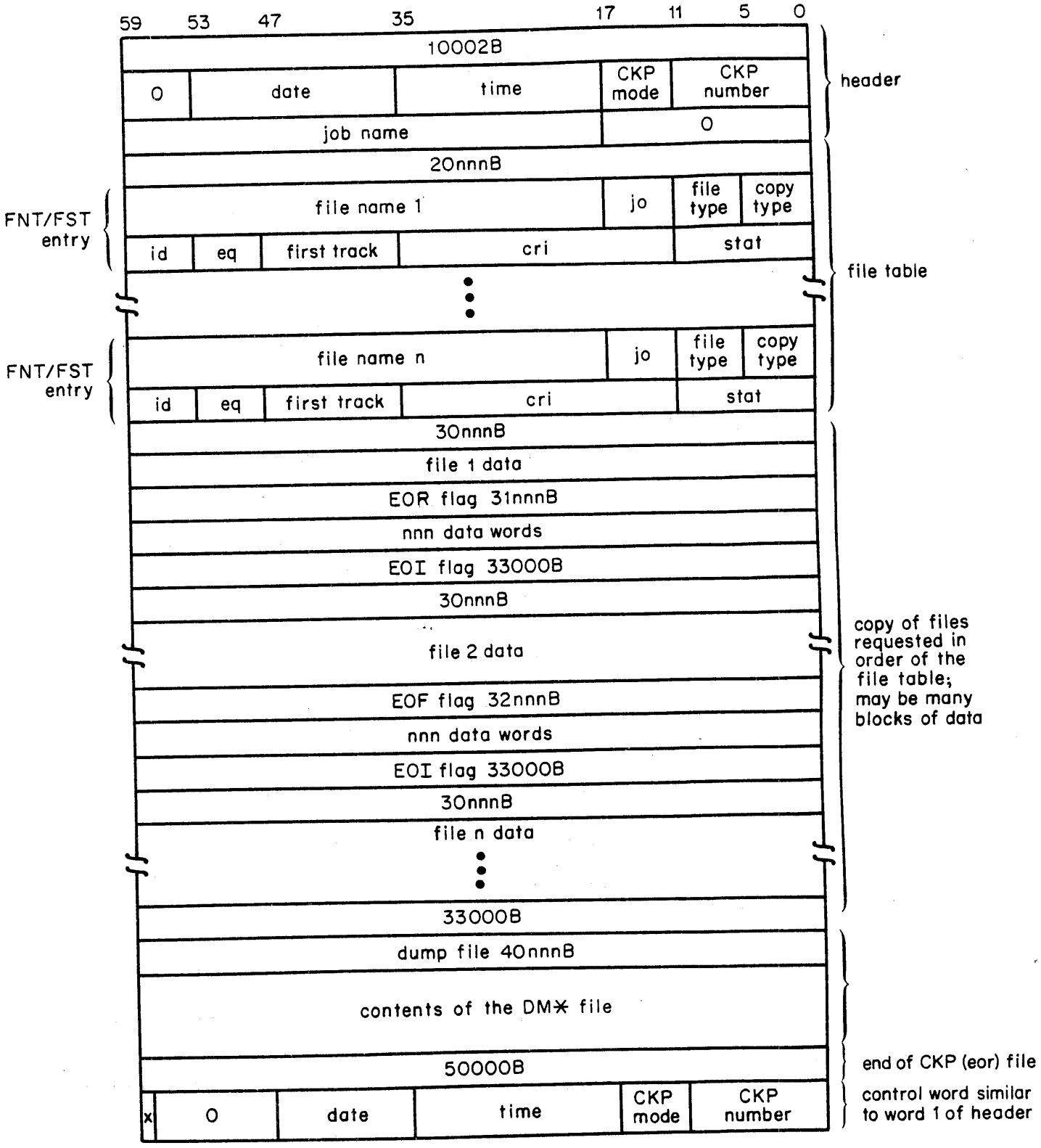


Figure 25-1. CKP Format

date	Date CKP record was written.
time	Time CKP record was written.
CKP mode	Indicates whether the CKP file is sequential (CK) or overwrite (CB) type.
CKP number	Sequential number of this CKP. Equals 1 after first CKP, 2 after second, and so on.
jobname	Name of job requesting CKP.
filename	Name of file to be checkpointed.
jo	Job origin or access bits from FNT.
file type	FNT file type (INFT, LOFT, for example).
copy type	Type of copy to perform. Unless otherwise specified by the user, files are copied according to their position and type of operation (read or write) prior to the CKP request. The copy types are: 0 BOI to present position 1 Present position to EOI 2 BOI to EOI 3 Last operation on file determines the copy type 4 No copy of file on CKP file, but information table is present
id*	FST id code.
eq*	FST equipment number.
first track*	FST first track if mass storage; if tape, MT; if terminal file, TT.
cri	Current random index. If tape file, cri is the block number. If terminal file, cri is 0.
stat*	Last status from the FET.
nnn	Number of words in this block (not including this word).
x	Bit 59 set if this is the last CKP dump on the file and is followed by an EOI PRU.

Figure 25-1. CKP Format (Continued)

---

*Standard FST information (except MT and TT for first track).

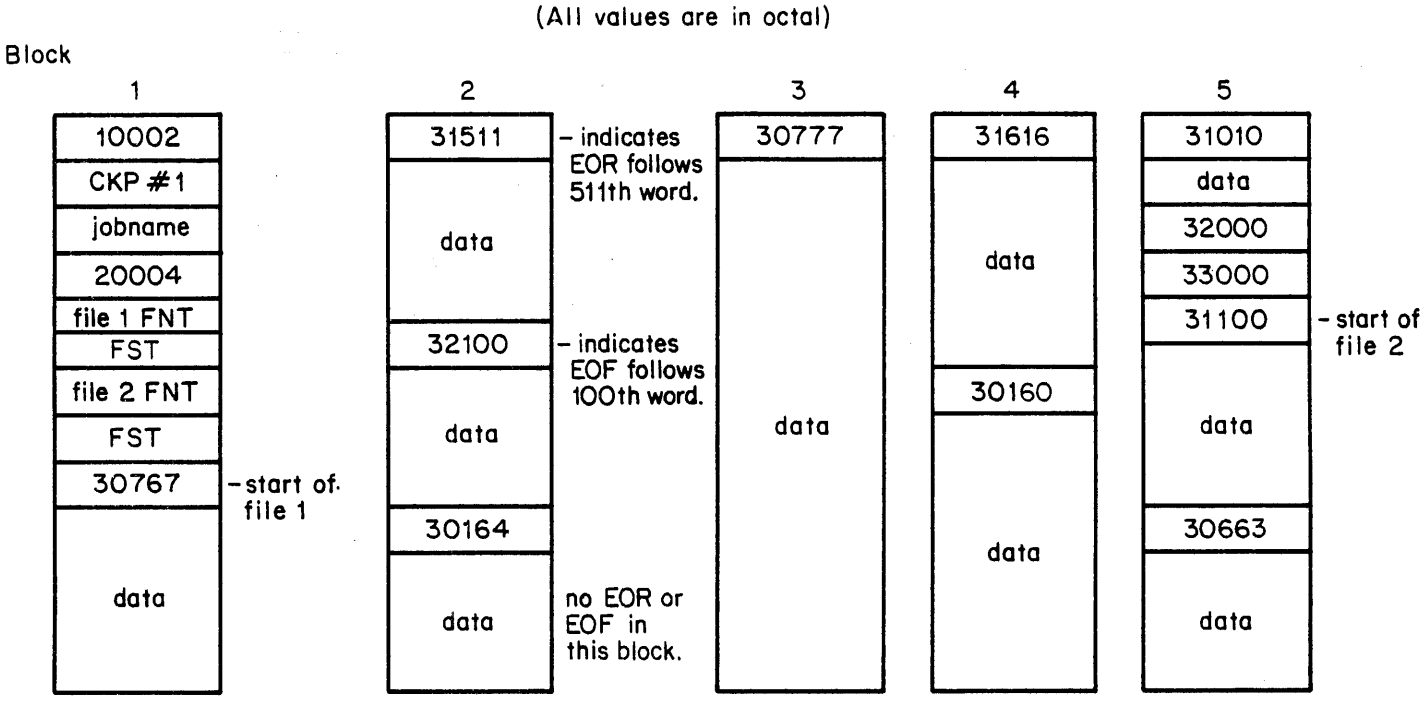
Figure 25-2 illustrates how the checkpoint file looks assuming a job has the following characteristics:

- FL = 2600B, control point area = 200B. So DM* file consists of 200 (CPA) + 2600 (FL) = 3000B words.
- Two files imply 4 words of FNT/FST information.

  File 1 consists of: BOI, 1500B words, EOR, 100B words, EOF, 2001B words, EOR, 170B words, EOR, EOF, EOI.

  File 2 consists of: BOI, 100B words, EOR, 1000B words, EOR, EOI.

- Assume this is a nonterminal job.



DM* file is identical to standard rollout file. Refer to section 5 for DM* file format.

Figure 25-2. Checkpoint File Structure

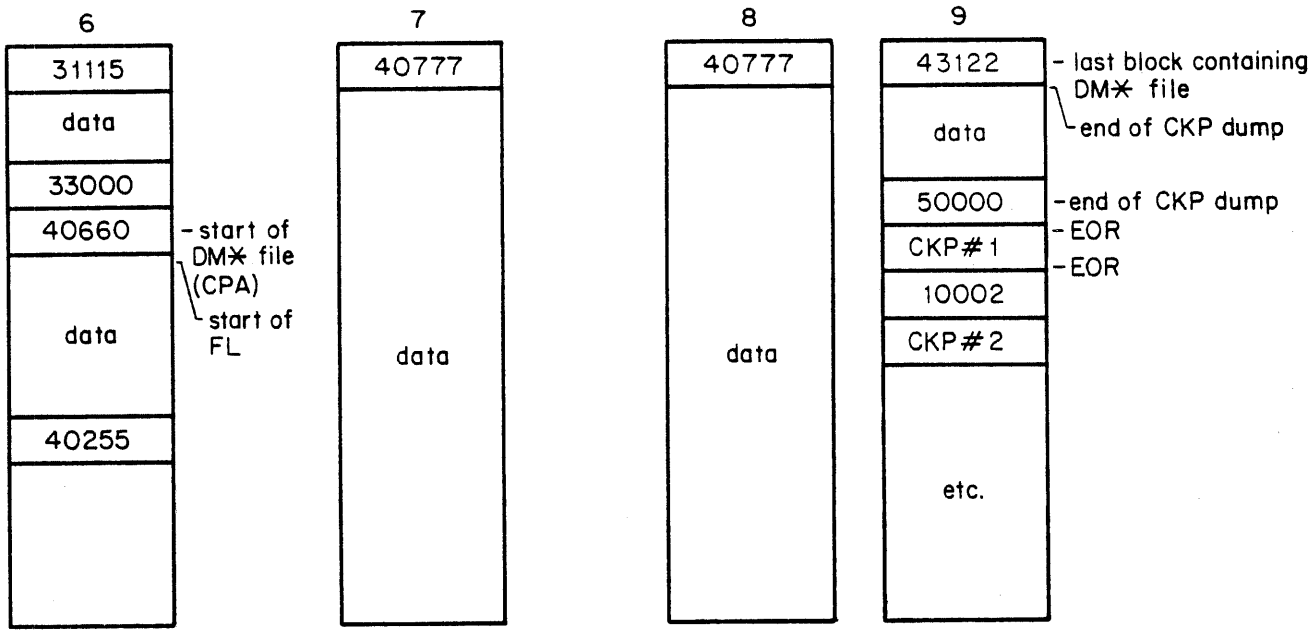


Figure 25-2. Checkpoint File Structure (Continued)

CHECKPOINT - CKP

CHKPT is a CPU routine which must reside either in the RCL or be disk resident (CLD - system). CHKPT can be initiated either by an operator command, a control statement call, a macro call, or by a product set call (refer to figure 25-3).

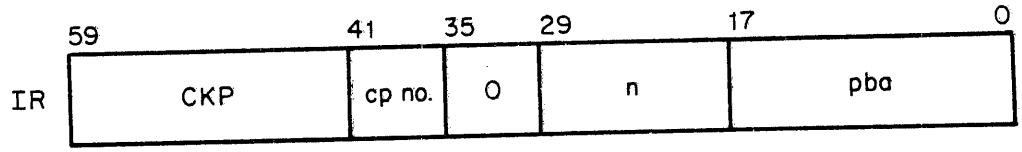
CHKPT has special entry point status (refer to section 5). CHKPT uses the following special entry points: DMP=, SSJ=, and RFL=.

If CHKPT is called by a control statement, 1AJ determines that it has an SSJ=, and a DMP= special entry point routine. 1AJ sets up SPCW, SEPW, and the control point area. Routine 1RO is called to create the DM* file. Since DMP= is equivalenced to zero in CHKPT, all of the job field length is saved on DM*. Routine 1AJ places the arguments from the control statement into RA+ARGR and sets RA+PGNR accordingly during the load of CHKPT. Then control is passed to CHKPT.

If CHKPT is called by a macro, an RA+1 request is made to CHKPT. This request is handled by SFP, an entry point in CHKPT.

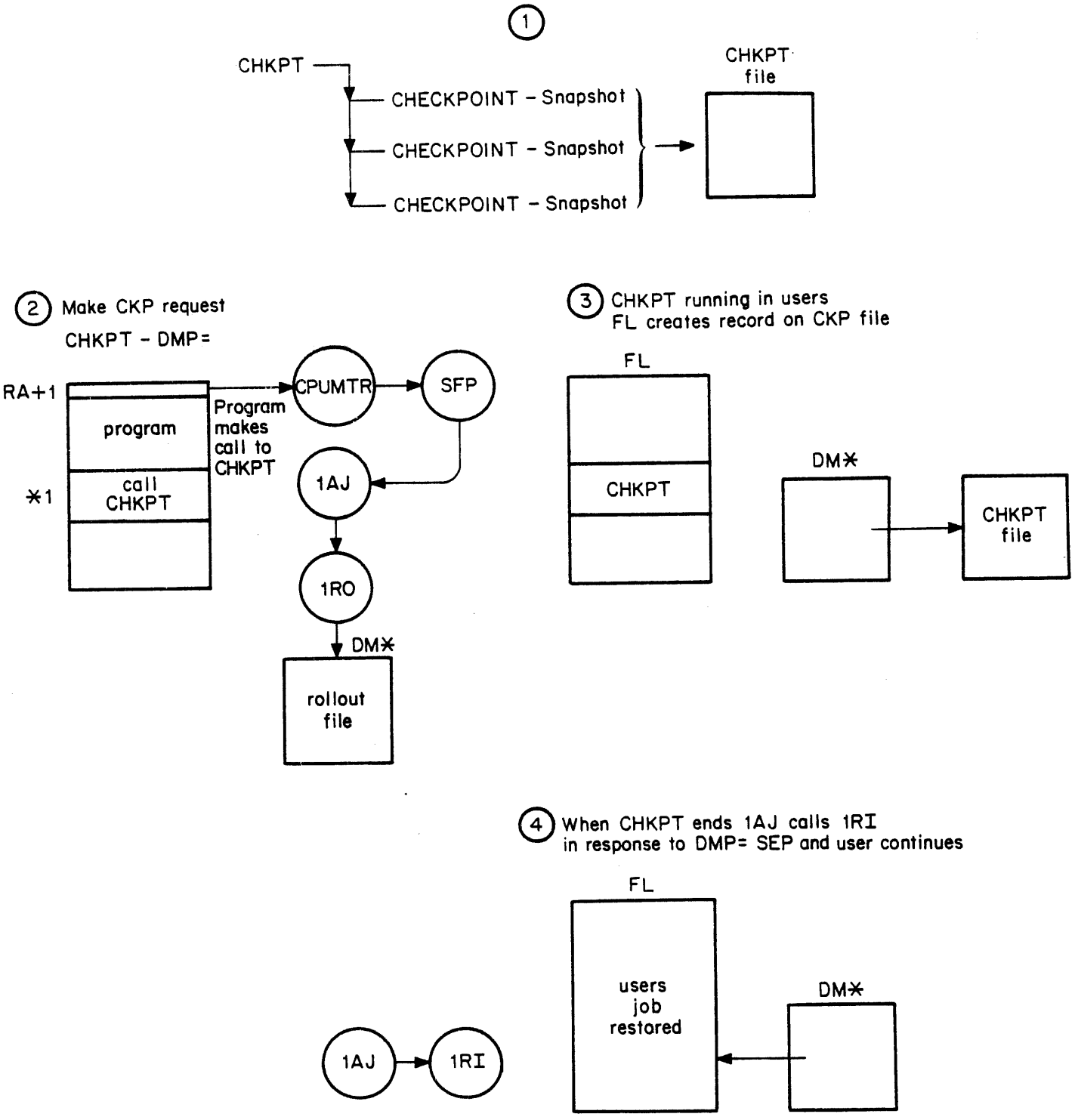
If CKP is called via a product set, such as FORTRAN or COBOL, an RA+1 request is made and the parameter list, if one is specified, is set up the same as in the macro call.

The RA+1 request is processed by CPUMTR, which places the call into the IR of an available PP.



n Number of parameters

pba FWA of parameter block

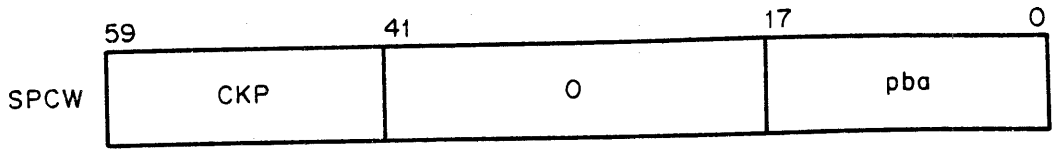


*1 Creates an RA+1 call.

Figure 25-3. Checkpoint Overview

Since PP resident does not find CKP in either the RPL or in the PLD, it calls SFP.

SFP finds CKP as one of its special processors. The SFP overlay 2SG (SRP - spcial request processor) sets up SPCW from the IR.



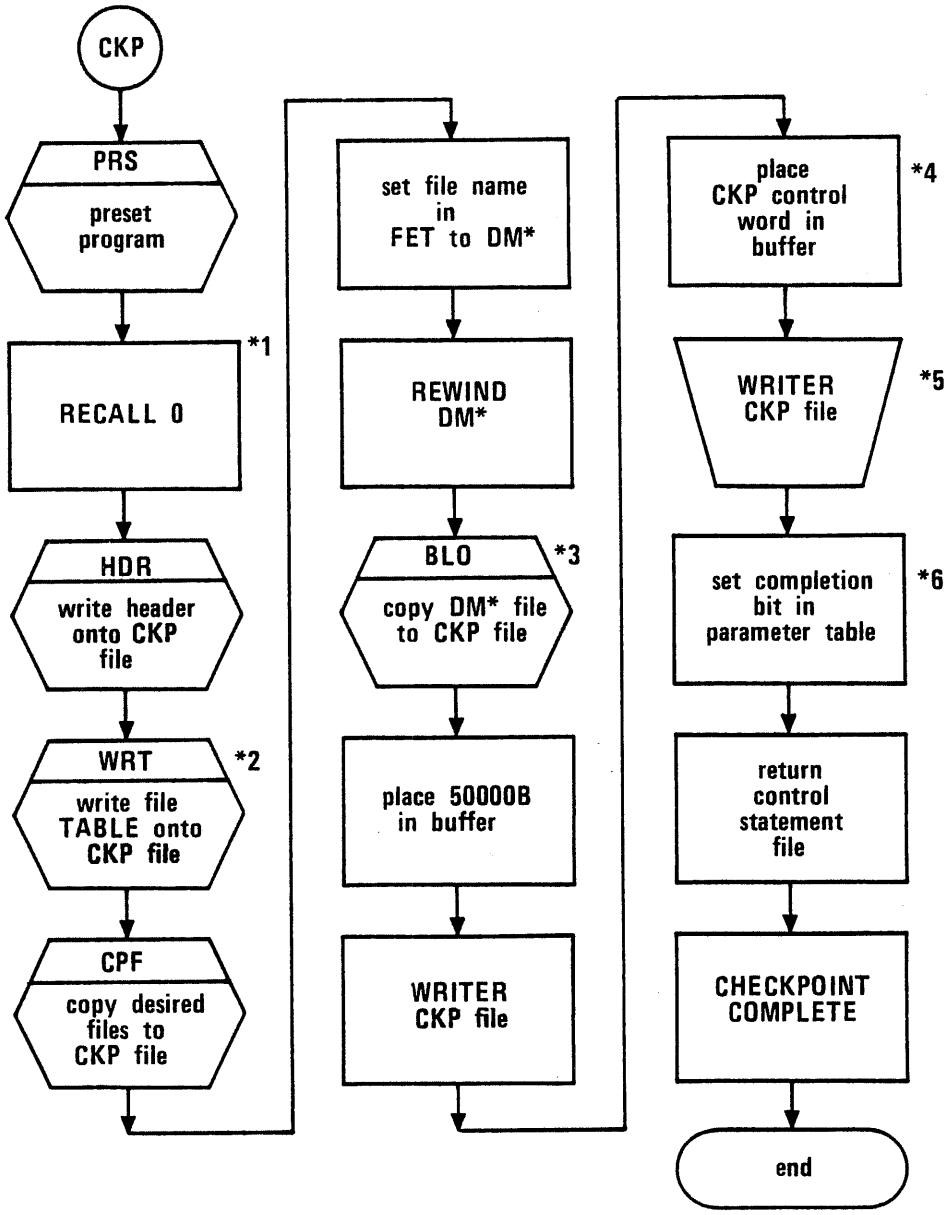
SFP exits normally and 1AJ finds SPCW set. It loads CHKPT, which has the entry point CKP, and sets RA+PGNR=0 to indicate a noncontrol statement call. Since a DMP= special entry point has been indicated in the CLD, 1AJ calls 1RO.

Routine 1RO finds the pba not equal to zero and gets a 20B-word block from central memory whose fwa is pba. It creates a full DM* file and then stores the 20B word block in RA+SSPR+1 through RA+SSPR+20. Routine 1AJ sets up SEPW and any priorities indicated by the SSJ= (in the case of CHKPT, there are no special priorities), stores the IR in RA+SSPR, and passes control to CHKPT at the entry point CKP. If more than a 20B-word parameter is passed (CHKPT can be passed up to 200), CHKPT has to read it from the central memory portion of DM*.

The preset routine in CHKPT is overlayed by the buffers, since its origin is at IBUF. In addition, RFL= is equated to the last word of CHKPT, which is necessary to use the SSJ= entry point.

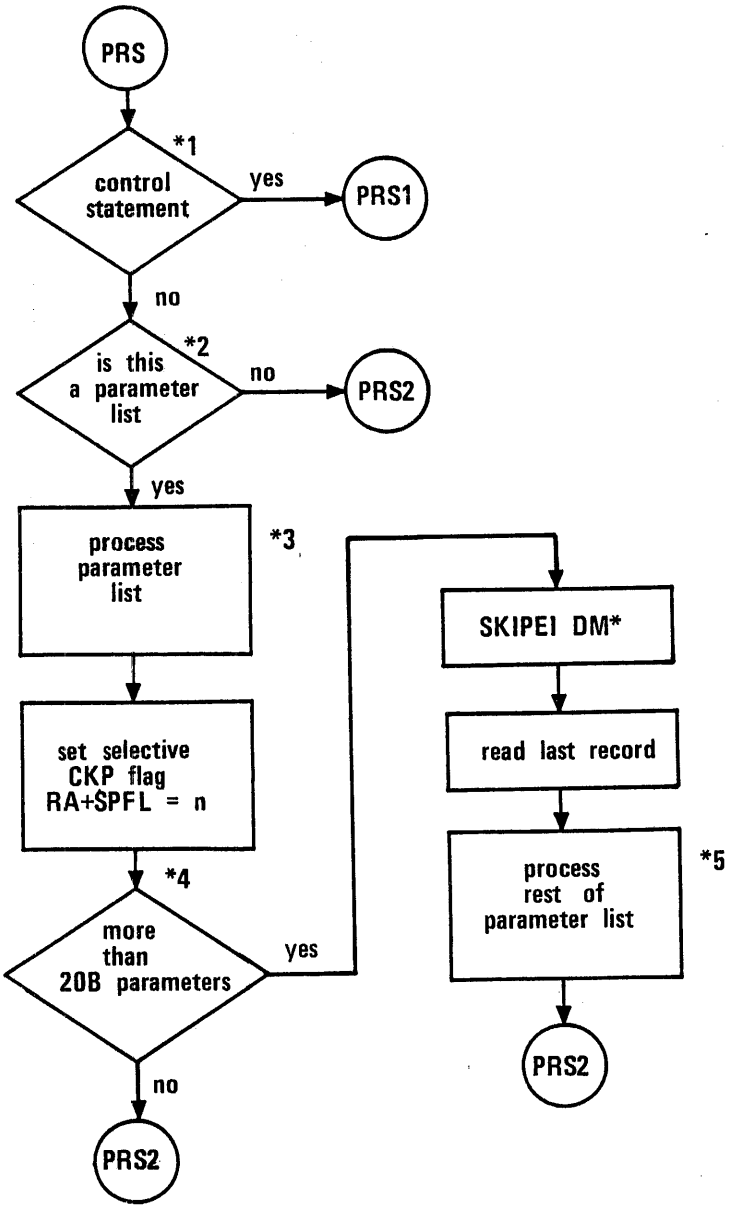
Tables 25-1 and 25-2 list some of the common decks used and the buffer assignments.

Figures 25-4 and 25-5 are flowcharts detailing the CHKPT main loop and preset routine.



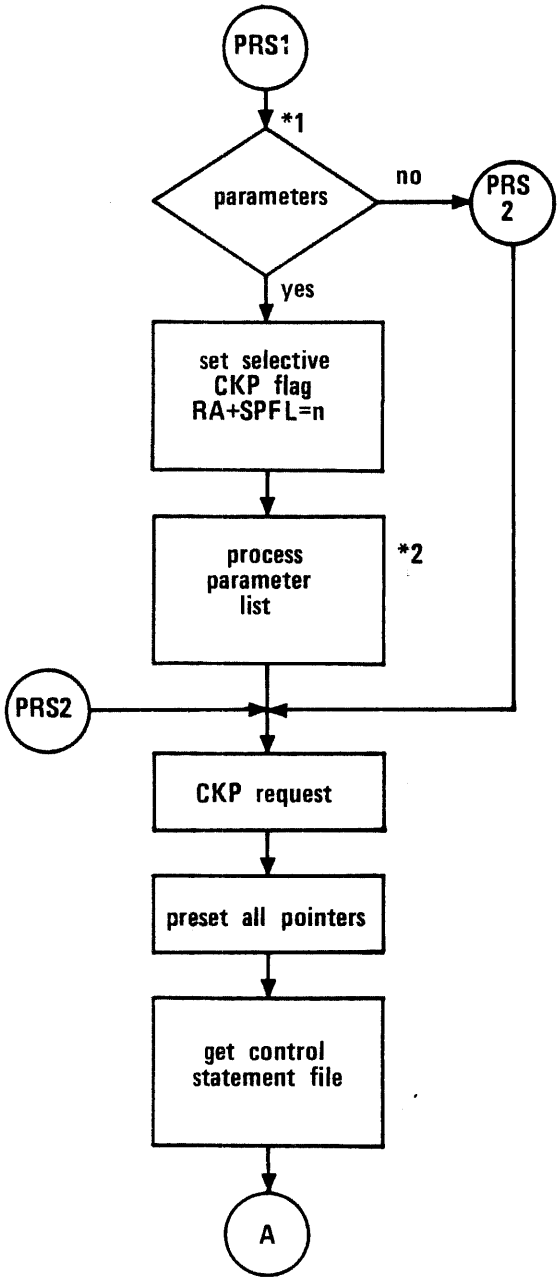
*1 Wait for any I/O initiated by PRS to complete.
*2 Use GETFNT macro which calls LFM to return a list of all FNT/FSTs assigned to this control point.
3 Refer to DMP= in section 5. Format of DM file is control point area, job field length. Copy complete DM* file.
*4 Copy of header word.
*5 Now CKP control word is embedded in EORs.
*6 SPRR+1 also backspaces file so trailer can be read by next CKP call.

Figure 25-4. CKP - Checkpoint Main Loop



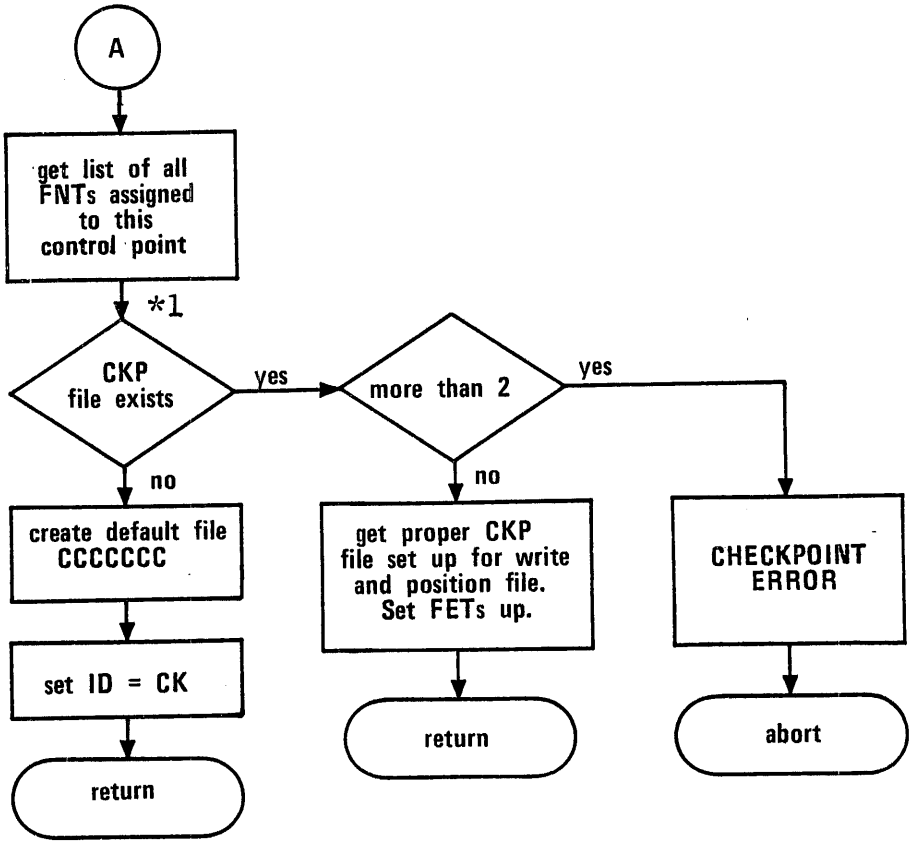
*1 Is (RA+PGNR) not equal zero?
*2 Is (RA+SPPR) lower 18 bits (that is, n from IR) not equal 0?
*3 Parameters = file names are placed in block PAR for use by WRT and CPF to get just selected files onto CKP file.
*4 Is n > 20?
*5 Only 77B parameters are allowed.

Figure 25-5. PRS - Checkpoint Preset



*1 Is RA+ACTR not equal zero?
*2 Maximum 63B parameters on a control statement.

Figure 25-5. PRS - Checkpoint Preset (Continued)



*1 See if any files local to job have type CK or CB.

Figure 25-5. PRS - Checkpoint Preset (Continued)

TABLE 25-1. CHKPT COMMON DECKS

Load Address	Common Deck	Description
724	COMCCDD	Constant to decimal display code conversion
736	COMCCIO	I/O function processor
752	COMCCPM	Control point manager processor
756	COMCDXB	Display code to binary conversion
776	COMCLFM	Local file manager processor
1006	COMCMVE	Move block of data
1070	COMCRDO	Read one word
1114	COMCRDW	Read words to working buffer
1245	COMCSFN	Space fill right justified zeroes
1255	COMCSYS	Process system request
1314	COMCWTO	Write one word
1331	COMCWTW	Write words from working buffer

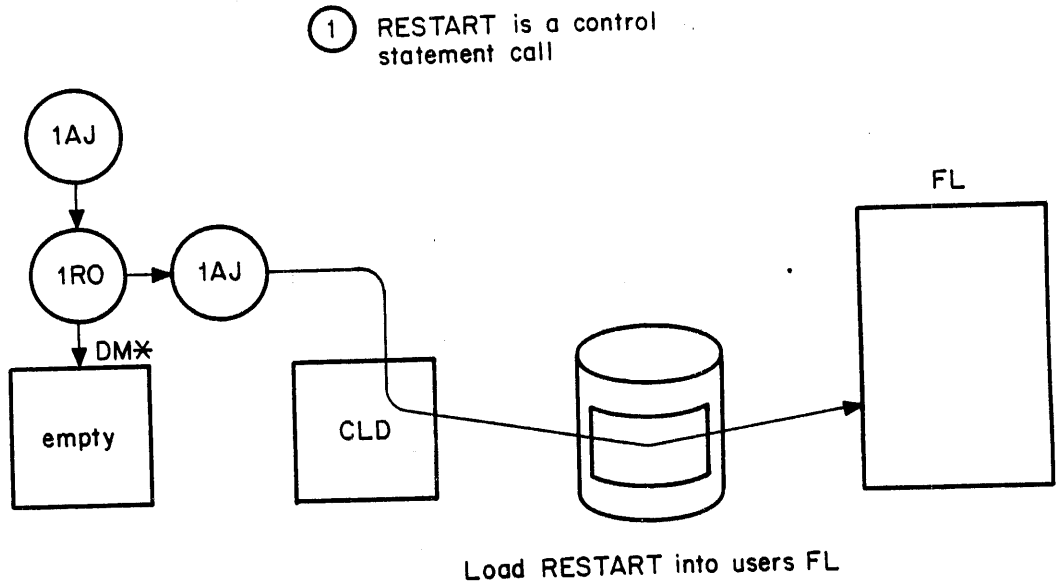
TABLE 25-2. BUFFER ASSIGNMENTS

Load Address	Buffer	Length
1454	BUF	Start of buffers
2454	IBUF	BUF+BUFL
4455	OBUF	IBUF+IBUFL
6456	SBUF	OBUF+OBUFL
7057	TBUF	SBUF+SBUFL
10060	RFL=	TBUF+TBUFL

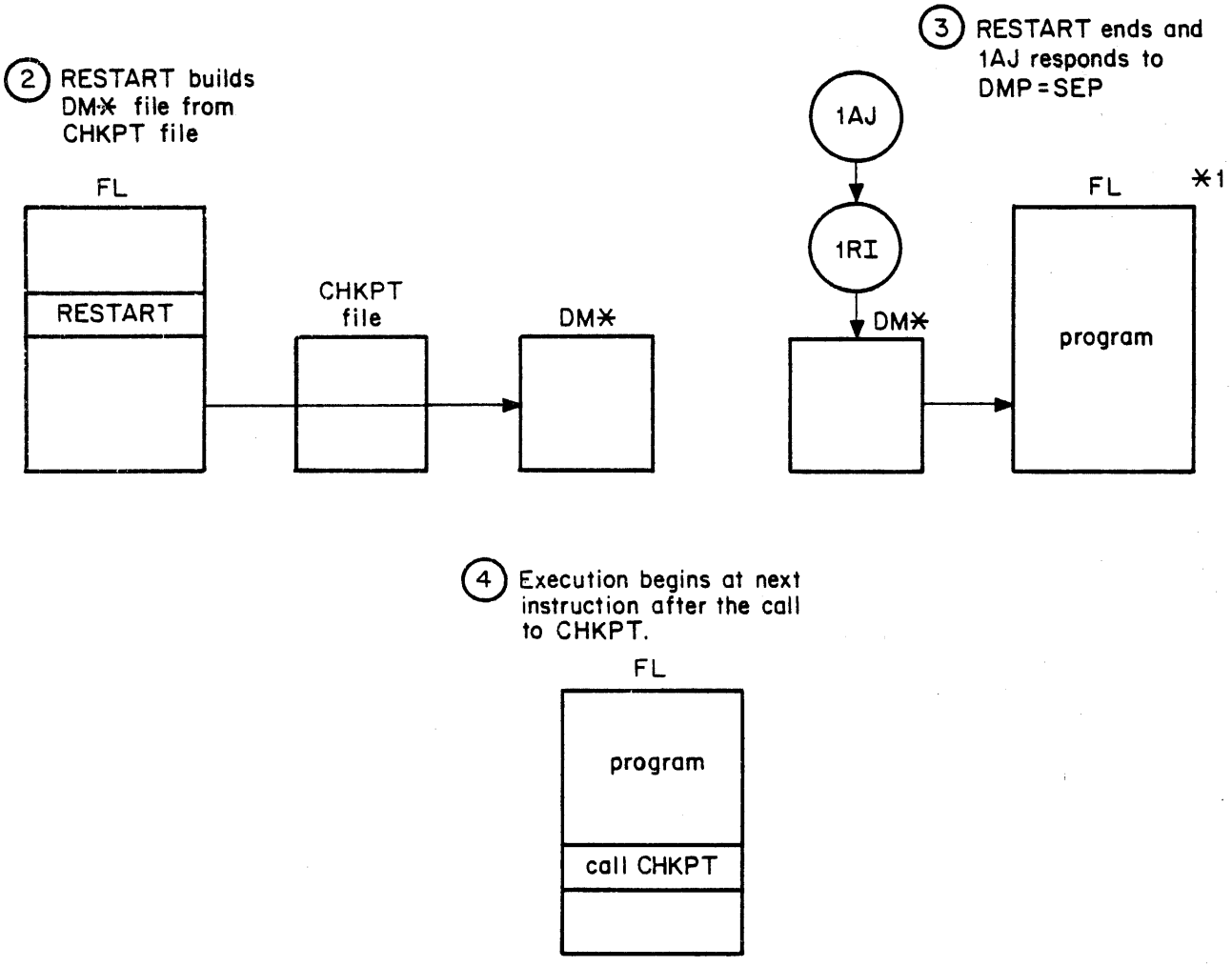
RESTART

RESTART is a CPU routine wich must reside either in the RCL or be disk resident (CLD). Whereas CHKPT writes the CKP file, RESTART restores the contents of the files copied to the CKP file and causes 1RI to restore the control point area and FL from the DM* file. (Refer to figure 25-6.)

RESTART has the special entry point DMP=. When 1AJ loads RESTART, it notes that special entry point is active from the CLD or RCL entry point word with bit 59 set. It calls 1RO which creates a DM* file. Since DMP= is equated to 450000R in RESTART, it creates an empty DM* file.



Fgiure 25-6. RESTART Overview



*1 This is the same area as RESTART of step 2.

Figure 25-6. RESTART Overview (Continued)

Routine 1AJ sets up the control point area (SEPW, and so on), loads RESTART, stores the argument list in RA+ARGR, sets RA+ACTR accordingly, and initiates RESTART. RESTART cannot be called from an RA+1 request, so the parameter passing ability of DMP= is not utilized. RESTART locates the proper CKP file, requests the FL required, restores the files required (including the DM* file from the CKP file), and exits.

Routine 1AJ then finds the control point idle and notes that this was a DMP= run. It calls 1RI, which rolls the job in using the DM* file created by RESTART. When 1RI is done, it clears the rollout flag, and the job is restarted from its position immediately following the checkpoint.

As in CHKPT, the preset routine is used as a buffer so that field length is minimized. Tables 25-3 and 25-4 list some of the common decks used and the buffer assignments.

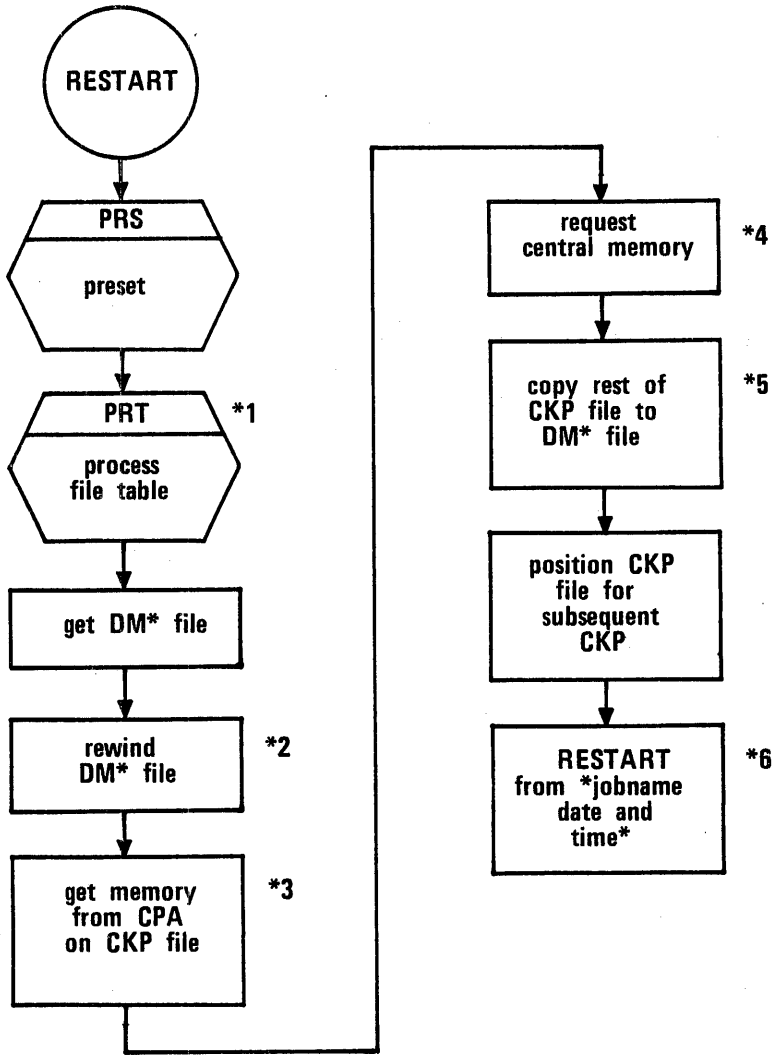
Figures 25-7 and 25-8 are flowcharts detailing the RESTART main loop and preset routine.

TABLE 25-3. RESTART COMMON DECKS

Load Address	Common Deck	Description
614	COMCARG	Process arguments
640	COMCCDD	Constant to decimal display code conversion
653	COMCCIO	I/O function processor
667	COMCCPM	Control point manager processor
673	COMCDXB	Display code to binary conversion
713	COMCEDT	Edit date or time from packed format
736	COMCLFM	Local file manager processor
746	COMCPFM	Permanant file processor
756	COMCRDC	Read coded line, -C- format
771	COMCRDO	Read one word
1014	COMCRDW	Read words to working buffer
1145	COMCSFN	Space fill name right justified zeroes
1155	COMCSYS	Process system request
1214	COMCWTO	Write one word
1231	COMCWTW	Write words from working buffer

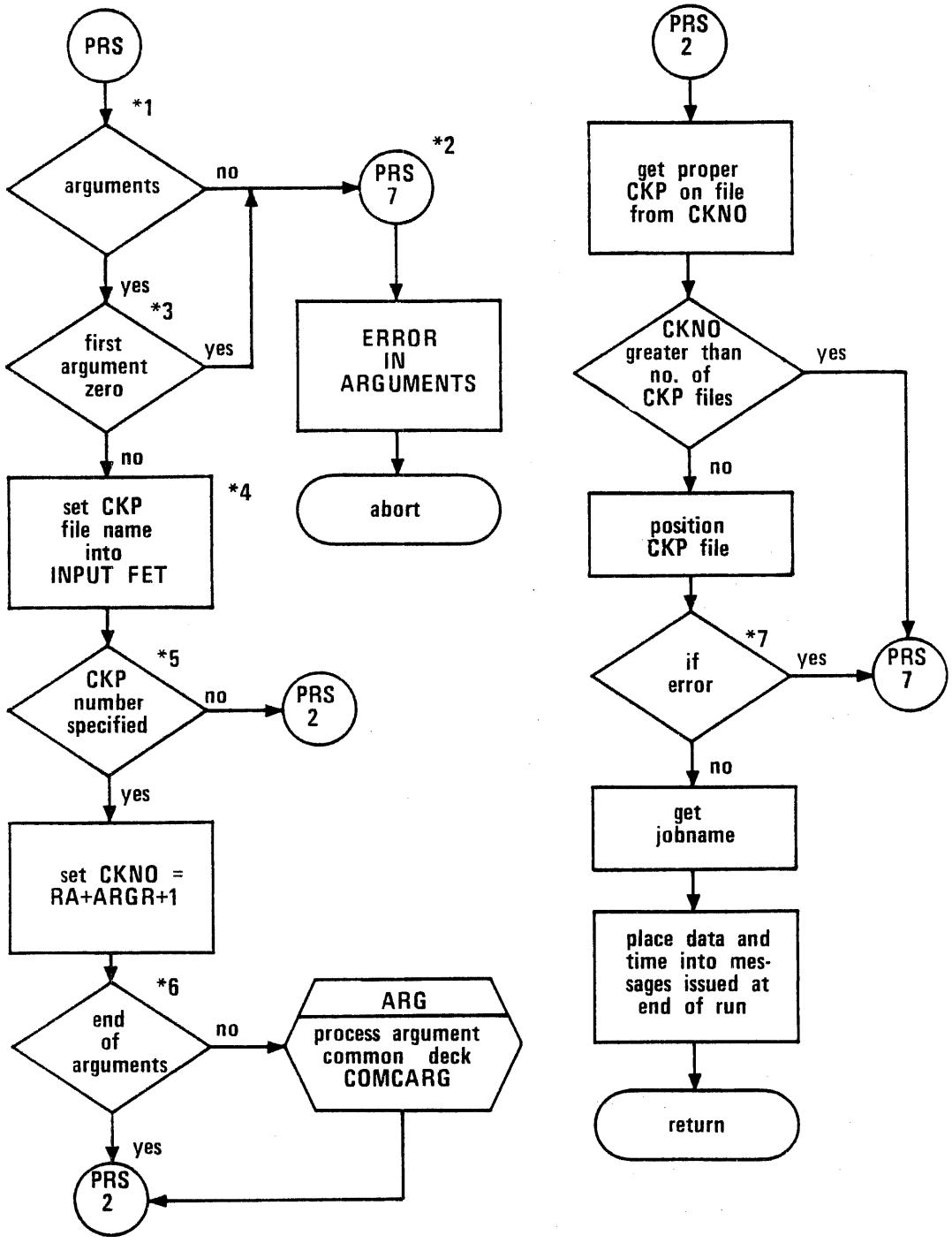
TABLE 25-4. RESTART BUFFER ASSIGNMENTS

Load Address	Buffer	Length
1470	BUF	Start of buffers
2471	IBUF	BUF+BUFL
4472	OBUF	IBUF+IBUFL
6473	SBUF	OBUF+OBUFL
7074	TBUF	SBUF+SBUFL
10103	RFL=	TBUF+TBUFL+4



*1 Copy all data files from the CKP file.
2 Prepare to write remainder of CKP file onto the DM file.
*3 Exchange package in first 20B of control point area and word CPA+2 is FL.
*4 If job card FL request is less than FL needed to RESTART, control point would be aborted by CPUMTR.
*5 Stop on EOR.
*6 Message set up by preset.

Figure 25-7. RESTART - Restart Main Loop



*1 Is (RA+ACTR) not equal zero.
*2 RESTART must have an argument list (checkpoint file name).
*3 Is (RA+ARGR)=0.
*4 (RA+ARGR)=CKP file name.
*5 Is (RA+ARGR+1)=0. Note that CKNO is preset to 0 at assembly.
*6 Is (RA+ARGR+2)=0.
*7 The error CHECKPOINT FILE ERROR if header word missing or CHECKPOINT NOT FOUND if asked for a CKP number that is not on the file.

Figure 25-8. PRS - Restart Preset

Deadstart is the process by which the system is made operational and ready to process jobs. Under normal circumstances, a deadstart is a function performed by the system operator. System deadstart can be a fully automatic procedure or it can involve considerable operator intervention.

The deadstart file is either a reel of magnetic tape, written in unlabeled I format, or an 844/885 disk device containing the programs which constitute the NOS operating system and its product set members. In addition, the deadstart file contains programs necessary to establish the system and its product set on the system equipment as well as maintenance routines used to test the condition of certain system equipments. As the deadstart file is not dependent upon a specific equipment configuration, the same file can be used on any supported equipment configuration.

To load the NOS operating system into a CDC CYBER 170, CYBER 70, or 6000 series computer, the deadstart medium is mounted, and a small bootstrap loader program (deadstart program) is set up on the hardware deadstart panel switches. The deadstart procedures themselves are explained in the NOS Operator's Guide.

HARDWARE DEADSTART

When the operator presses the deadstart button or toggles the deadstart switch on the deadstart panel, the following hardware activities occur:

1. Each channel is connected to its corresponding PP (that is, channel 1 is connected to PP1, channel 2 to PP2, and so on).

2. A master clear is done on all I/O channels. All channels are set to the active and empty condition, ready to accept inputs.

3. The contents of the A register of each PP is set to 10000B so that a PP can input its entire field length before automatically disconnecting from the channel. The P register of each PP is cleared.

4. The deadstart panel is sent across channel 0 into PP0 locations 1 through 20 (locations 1 through 14 on non-CYBER 170 equipment).

   The deadstart controller outputs one byte of zeros and the panel settings to PP0. It then issues a DCN instruction and PP0 begins execution at location (P)+1 = 0+1 = 1.

5. Each PP simulates an IAM instruction on its channel as if the contents of PP locations 0 through 4 were:

location	value	comments
0	0000	start at location P+1 = 1
1	2001	set A register to 10000B
2	0000	
3	71pp	IAM pp; pp is PP number (channel)
4	0000	location to input data (location 0)

6. The CPU initiates a hardware idle.

When the IAM begins, (P) +1 is stored in location zero. As each 12-bit PP word is transmitted across the channel, (A) is decremented by one. Whenever (A) is 0 or the channel is disconnected by another PP or a controller, the receiving PP stores (0)+1 into P and execution begins at that location. This procedure is used to autoload routines. A PP will IAM a set of words and then input as the final word an execution address minus one into location zero. The PP then begins executing at the execution address specified. PPs may communicate with hardware or another PP via a channel. If a PP communicates with some hardware, it must set its A register to the number of words it wishes to input. When this number of words has been input, it will execute at (0)+1. If two PPs are communicating when the transmitting PP does a DCN on the channel, the receiving PP begins execution as if A went to zero.

SOFTWARE DEADSTART

STARTUP

When the hardware is ready, PP0 begins executing the program on the deadstart panel. The following operations are performed:

1. PP0 disconnects the deadstart channel, then reconnects (via function) to the channel, equipment, and unit specified in the panel settings, and rewinds the deadstart device. PP0 then reads the first record from the deadstart medium into its memory starting at location 7301B (also specified on the panel).

The deadstart file has a fixed order through the routine SYSEDIT. Included in this ordering are zero length records which delineate certain areas of the deadstart file so that the logic for reading the file is as simple as possible during the deadstart sequence.

OSB

Routine OSB is the operating system bootstrap which is loaded by CTI into PP0 as the system file preloader. OSB contains a tape and disk driver and must be no longer than one disk sector in length including the prefix (7700) table. The only function of OSB is to load the next operating system module, DIO, from the deadstart medium. DIO is transmitted to PP10 along with deadstart parameters determined by CTI, after which OSB hangs on an IAM waiting to input SET on channel 10.

DIO

DIO is the centralized means of reading data from the deadstart medium, either tape or disk. This allows the other deadstart-related routines to remain unaware of the medium being read. Thus, 10B disk sectors are assembled if deadstart is from RMS into a facsimile of a tape PRU. Commands are issued to DIO via an intercommunications channel according to a protocol resembling that of a 3000 type tape controller and 6681. DIO in PP10 functions as the 6681, reading data from an external equipment and returning the same to the requesting PP. The intercommunications channel is defined as channel 13 unless the deadstart medium is there, in which case channel 12 is used.

DIO initially loads SET and transmits it to PP0. Next, the deadstart parameters as found by CTI are transferred to PP0 and SET begins execution. DIO now drops into an idle loop awaiting function requests on the intercommunications channel.

DIO remains active until such time as the deadstart file is normally rewound. In the case of level 1 or 3 deadstarts, where the running system file(s) is recovered, this occurs after the deadstart RPL has been built. On a level 0 or 2 deadstart, DIO functions until SYSEDIT is loaded. Then DIO may exit to PP resident to become a pool processor. The deadstart file is copied to SYSEDIT's buffer either by DIO or CIO, depending on the deadstart medium.

SET

SET initializes the system configuration by assembling system parameters such as equipment definitions and installation options from text decks that are present on the deadstart tape; namely, the CMRDECK which defines the configuration, and the IPRDECK which specifies installation options. SET communicates with PP10 as follows:

1. When SET begins execution in PP0, PP1 is waiting for input on channel 1*. SET transmits via channel 1* an idler program to PP1 and disconnects channel 1*.

2. PP1 begins executing the idler program. Besides an idle loop, the idler program consists of the following processors:

Value	Name	Comments
0	RSP	Reset PP/terminate
1	IFB	Input first buffer from tape
2	ISB	Input second buffer from tape
3	OFB	Output first buffer to SET
4	OSB	Output second buffer to SET
5	ONL	Output next line to SET
6	RCL	Replace current line
7	ANL	Add next line to PP1 buffer
10	ADS	Advance display

3. SET uses channel 10 to display the CMRDECK/IPRDECK and CMRINST/IPRINST on the display console. SET also accepts operator typeins via channel 10.

4. SET uses PP1 as a CMRDECK/IPRDECK buffer while building the appropriate DECK from operator input, if any.

5. SET communicates with PP1 via channel 1*. PP1 gets the appropriate CMRDECK/IPRDECK from the deadstart medium via the intercommunications channel and DIO in PP10.

6. When SET completes CMRDECK and IPRDECK processing, it issues the RSP function to PP1 via channel 1*.

7. PP1 loops waiting for input on channel 1*.

---

*If the deadstart medium is on channel 1, channel 12 is used.

SET initializes the system in the following sequence:

1. SET determines the properties of the hardware upon which the system is being established from the hardware descriptor table created by CTI. These properties include central memory size, number of PPs, presence of CMU and CEJ/MEJ, type of CPU and hardware, and so forth. Certain of these hardware features may be overridden by CMRDECK entries.

2. SET loads overlay CMR (next record on the deadstart file), which has the code to process CMRDECK entries; reads up the text deck CMRINST (CMRDECK instructions, the next record on the file); and reads the specified CMRDECK.

3. CMR displays via PP1 the CMRINST or CMRDECK decks and accepts input from the console. When the CMRDECK is changed to the operator's satisfaction, CMR skips all text records (CMRDECKs) on the file and loads the next nontext record (which must be ICM) as a secondary overlay.

4. Initialize central memory (ICM), returns control to SET after building the following central memory resident tables:

   Channel release table
   Equipment status table (EST)
   File name table (FNT)
   FNT interlock table
   Mass storage tables (MST/TRT)
   Job control area
   Control point areas
   Dayfile pointers
   PP communication area (initialization)

   The first five entries in the FNT are:

   SYSTEM
   VALIDUs
   SALVARE
   RSXDid
   RSXVid

5. SET loads the next record, IPR, from the deadstart medium.

6. SET reads IPRINST (IPRDECK instructions, the next record on the deadstart file) and the specified IPRDECK, displays them, and accepts console input if the last CMRDECK typein was the command NEXT. If the last CMRDECK command was GO, the specified IPRDECK is read and no input is accepted. IPR sets the appropriate portions of central memory resident (IPRL, job control area, etc.) as specified by the options in the IPRDECK or by operator typein. Control is returned to SET when IPR has completed.

7. SET scans all remaining text records (IPRDECKs) and loads PP resident (the first nontext record), into its PP buffer. SET sets the location of its PP buffer to PPFW-1. SET then transmits the buffer starting at PP buffer minus one into PP2 on channel 2*. This puts PPFW-1 into location 0 of PP2. SET reads the next deadstart record, system tape loader (STL), into the same PP buffer. SET then transmits this buffer to PP2. PP2 inputs STL starting at location PPFW. SET disconnects channel 2*, terminating the input of STL by PP2 and causing PP2 to begin execution at PPFW. SET (PP0) then issues an IAM to wait for input on channel 0*.

SYSTEM LOADING

The system tape loader (STL), processes the next group of records on the deadstart file. STL loads MTR and CPUMTR and enough of the system routines to allow recovery and SYSEDIT to function. The following sequence is performed.

1. Load a copy of PPR to all PPs except 0 and 1.
2. Load MTR (next record on tape) into PP0.
3. Load and initialize CPUMTR; start CPUMTR into execution via an EXN instruction.
4. Load DSD into PP1.
5. Disconnect channel 0*; this causes MTR in PP0 to begin execution.
6. Load RSL as an overlay to STL to generate the resident peripheral library (RPL) and peripheral library directory (PLD) containing those PP routines used during deadstart.
7. Activate other PPs by disconnecting their channels, causing the code transmitted in step 1 to be executed.
8. Load controlware for 7054/7154/7155 controllers. BCL reads the BCS/BCF/FMD controlware records from the deadstart medium and outputs the appropriate record on channels that have 7x5x controllers. The entire controlware block is sent one channel at a time until all 7x5x controllers have been properly loaded.
9. Load recover mass storage (RMS) into the next available PP (usually PP3).
10. Load system library loader (SLL) into the next available PP (usually PP4), if the system file is to be loaded (level 0 or 2 recovery).

---

*If deadstart medium is on this channel, channel 12 is used.

11. Load LSL (load system library) as an overlay to STL if level 0 or 2 deadstart (not on level 1 and 3).

12. LSL moves the PP assignment to the deadstart control point (control point 1) using CCAM. The deadstart control point exchange package and control point area are initialized; the FL used is the remainder of the available machine field length up to 131K.

13. LSL sets up an INPUT file FNT so that the deadstart control point resembles a control point in the running system.

14. LSL creates a deadstart file FNT if deadstart is from disk.

15. LSL loads SYSEDIT from the deadstart file into the deadstart control point field length and requests the CPU via an RCPM to begin SYSEDIT execution.

16. LSL calls DIO with a load system request and then returns to the main program of STL if level 0 or 2 deadstart (not on level 1 and 3). STL moves to the system control point via a CCAM and drops the PP with a DPPM.

17. SYSEDIT reads the deadstart file (using DIO if tape deadstart, or CIO if disk deadstart) and copies the data to each system device defined using SLL. SLL guarantees that each copy of the system uses the same tracks.

SYSEDIT

SYSEDIT begins to build CM and mass storage library directories when the end-of-file on the deadstart file is detected. The following steps are performed.

1. Reading from the first system device, the resident peripheral library (RPL) is built by SYSEDIT and copied to central memory by SLL. The routines are entered in the RPL without their 77 (prefix) tables.

2. A PPULIB file without prefix tables is built on each system device of all PP routines. The peripheral library directory (PLD) is built consisting of the name and residence location of the routine whether it be central resident, on an ASR device, or on a SYSTEM device.

3. The resident central library (RCL) is built using those CPU routines selected for central memory residency in the LIBDECK via the *CM directive.

4. The central library directory is then built for all CPU routines and consists of the routine name and residence location of the routine whether it be central resident, on an ASR device, or on a SYSTEM device.

5. SYSEDIT completes its execution via an ENDRUN macro. At this point, the operating system has been fully established on the system equipment.

SYSEDIT may also be run on-line during normal production. If a SYSEDIT is run beyond deadstart, then any new or replaced CP or PP routines, libraries, etc., are written starting at the end of the system file. The RPL, PLD, RCL, and CLD are regenerated using these new or replacement routines as necessary.

During deadstart SYSEDIT uses the LIBDECK specified by the CMRDECK directive from the system file to determine the residency and special properties of system routines. Each subsequent SYSEDIT operation appends the directives used to the LIBDECK initially used during deadstart. These LIBDECKs are linked together so that SYSEDIT can recreate the directories from any earlier time if directed to do so by a SYSEDIT(R=n) restoration.

MS RECOVERY OPERATIONS

Mass storage recovery takes place in the PPs. The deadstart portion of mass storage recovery is briefly described here; mass storage recovery is presented in detail elsewhere in this manual.

Recover mass storage (RMS), which was called by STL, performs mass storage recovery in the following sequence:

1. Reads MSTs from device labels.

2. Recovers TRTs when possible.

3. Validates and recovers system dayfiles.

4. Validates and recovers preserved files.

5. Validates the permanent file configuration.

6. Loads the system recovery processor (REC), into this PP.

REC performs system recovery in the following sequence:

1. Reads system tables.
2. Recovers FNT, control points, and so forth.
3. Initializes or activates dayfiles.
4. Waits for SYSEDIT/SLL to complete. The SYSTEM file FST is set not-busy when SYSEDIT completes.
5. Builds IQFT (level 0 deadstart only) from queue type files found on the disk and clears all direct access file interlocks.
6. Issues appropriate dayfile messages.
7. Recovers/allocates user ECS area.
8. Calls 1CK to checkpoint system devices (always on a level 0 deadstart).
9. Drops the PP via DPPM.

DSD, which was earlier activated by STL, processes the IPRDECK initial commands (AUTO. and MAI., for example) specified by the DSD directives.

The scheduler (1SJ) finds that control point 1 (the deadstart control point) now has a status of 0 and releases the control point. Since there is no output, no output or dayfile messages are issued.

The system is now fully operational.

PPR INITIALIZATION

When STL sends a copy of PPR (PP Resident) to each pool PP, the direct cells IA, OA, and MA are set with the correct addresses for the PP being loaded. When STL disconnects the channel, each PP begins executing at PRS which is the resident initialization routine (preset). PRS is at location MSFW and is overlayed by the mass storage driver. PRS performs the following:

1. Reads PPCP, the first word address of the PP communication area.
2. Reads IA, which is the address of this PP's IR.
3. Computes the exchange package addresses for this PP from the PP's IR.

4. Modifies instructions in FTN (function processor) with the exchange package address.

5. Modifies instructions in FTN depending upon the existence of CEJ/MEJ.

6. Jumps to PPR (idle loop) making the PP a pool PP.

RECOVERY

Deadstart recovery is an inhibition of part of the deadstart process. No special routines or special code is designed for different levels of recovery. The philosophy is that deadstart is always a recovery and the levels of deadstart only denote how much to recover and how much to reload. The various levels of deadstart recovery are discussed in detail in the NOS Operator's Guide. Mass storage recovery is detailed elsewhere in this manual.

Basically, there are four recovery (deadstart) options:

Level	Description
0	All of central memory resident, all of the system, and all PPs are reloaded. Permanent files and dayfiles are recovered from the labels of the mass storage devices if possible. Job queues are restored depending on the QPROTECT option.
1	A level 1 recovery rebuilds the running system, all jobs, and all active files from the checkpoint file instead of the deadstart tape. A deadstart automatically creates a checkpoint file as part of the deadstart process. The FNT is recovered from the checkpoint file with central memory resident rebuilt and all PPs reloaded. Mass storage is recovered from the MST/TRT image written on the label track by a device checkpoint which was done when the system was checkpointed.
2	A level 2 deadstart is the same as a level 1 recovery except that the operating system is loaded from the deadstart file rather than from the checkpoint file.
3	A level 3 recovery only reloads the PPs and CPUMTR; everything else is left intact. Level 3 reloads directories from the system table file. This voids any on-line SYSEDITs that were not completed with a system checkpoint. Jobs currently in central memory may be rerun with the INPUT file being rewound and placed back into the INPUT queue.

CHECKPOINT FILE

The system table file, usually called the checkpoint file, is used in level 1, 2 and 3 recoveries. The first track of this file is kept in byte 3 of the DULL entry in the MST for system devices. It contains a copy of central memory resident; namely, the low core pointers, the EST, the FNT, the dayfile buffers and their pointers, and from the beginning of the RPL to the end of central memory resident. Except for the system checkpoint done during the deadstart process, a system checkpoint should be followed by a level 1 or level 2 deadstart. To continue system operations after a system checkpoint jeopardizes the success of subsequent level 1 or 2 recovery.

DISK DEADSTART FILE

In order to deadstart from disk, it is necessary to create a system deadstart file (SDF) on a disk device. The two routines, INSTALL and 1IS, which control this function, are discussed below. CTI must have been previously installed on the disk and the device must have been initialized following CTI installation. Also, if the mass storage library is to be installed, this must be done before installing the system deadstart file.

INSTALL

This CPU routine works in conjunction with the PP program 1IS to install a SDF on a RMS deadstart device. The install procedure can be initiated automatically at deadstart by keying in the INSTALL directive at CMRDECK time, or else as a control statement issued by a system origin job. The result is a preserved LIFT type file, with pointers to it set in the deadstart sector.

The format of the control statement is as follows.

```
INSTALL,lfn,EQxx.
```

lfn Name of the install file.

xx Logical equipment number of the device to receive lfn as a SDF.

If the lfn parameter is omitted, default is to file SYSTEM, which must be attached to the control point. An install file named SDF is illegal as this is the name reserved for the SDF. A special form of the control statement, without parameters is issued by 1IS if the job was invoked during the deadstart process. Any error encountered during parameter checking aborts the job with the following message.

```
INSTALL - ARGUMENT ERROR.
```

After cracking parameters, INSTALL issues a 1IS request (function 1) to validate the install file, and then reads the directory record and verifies it for proper format. The install file must reside on mass storage. If a user wishes to install a tape file, it must first be copied to disk. The directory is now checked for more than one OPLD entry pointing to a directory record. If more than one entry is found (as would be the case when installing the SYSTEM file after SYSEDIT's had been performed), the relative sector address of the first OPLD is placed into FET+6, and another read is executed to get the correct directory. Two is added to the current random address (returned by CIO), and this value is saved as the length (relative sector address of EOI) of the install file. Next the directory is searched for the bootstrap program OSB, and its random address also saved. Any error encountered during directory processing aborts the job with the following message.

```
DEADSTART FILE FORMAT ERROR.
```

The preset portion of INSTALL completes by making a service call to 1IS (function 2) to initialize the SDF. At this point the install file is ready to be copied to the SDF device.

The disk to disk copy mechanism employed by INSTALL utilizes the single buffer approach advanced by COPYB. Two FETs point to the same buffer and are used in conjunction with control word I/O. The install file is copied to the SDF until an EOF is encountered. Routine 1IS is called once again (function 3) to complete operations on the SDF and insert pointer information in the deadstart sector. The job terminates with the following message.

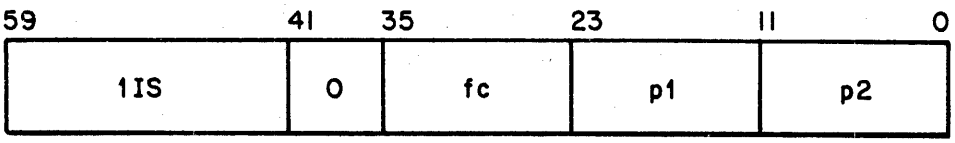
```
SDF INSTALLATION COMPLETE.
```

In the event that errors are detected while copying the install file, 1IS is called (function 4) to release the SDF disk space, clear interlocks, and abort the job.

ROUTINE 1IS

Routine 1IS is a PP routine which serves as a function processor for the CPU program INSTALL. INSTALL communicates with 1IS via the SYSTEM SPC macro, passing parameters in the call and in word 5 of the FETs. Recall is on word ARGR (RA+2). There are four functions associated with 1IS, plus the special case situation in which an install job was initiated at deadstart.

The format of a 1IS call is as follows.



fc Function code

p1 Relative address of SDF file FET

p2 Relative address of install file FET

If an install job is initiated at CMRDECK time, an input file FNT/FST is created, containing the logical equipment number of the device to receive the SDF. When the job gets scheduled, at completion of deadstart sequencing, 1IS is called into a PP. Because there is no function request in the input register, 1IS knows that the job was initiated during the deadstart process. It then creates an LOFT type FNT entry for the SDF with the specified equipment and a special file ID (77B).

The special file ID is set so that later on when the file is used (function 2) it can be determined that 1IS created the FNT/FST with proper equipment number, as opposed to a local file of the same name (SDF) associated with an install job initiated on line. The file type is changed to LIFT at job completion, before the system sector is written. Lastly, 1IS copies two control statements to the control statement buffer, sets buffer pointers, and exits to PP resident. The control statements are:

```
COMMON,SYSTEM.
INSTALL.
```

Function 1 - Validate Install File

Routine 1IS searches the FNT for an assigned file named in the FET, and if not found, aborts with the message INSTALL FILE NOT FOUND. Likewise, if the install file is not on mass storage, the job aborts with the following message.

```
INSTALL FILE NOT MASS STORAGE.
```

Next, 1IS locates EOI on the install file, stores the EOI address in the FST, stores the FNT address in the FET, and then exits (function complete).

Function 2 - Initialize SDF

Routine 1IS searches the FNT for an assigned file named SDF. If a file is found but not with the special file ID, the job is aborted with the following message.

ASSIGNED FILE CONFLICT - SDF.

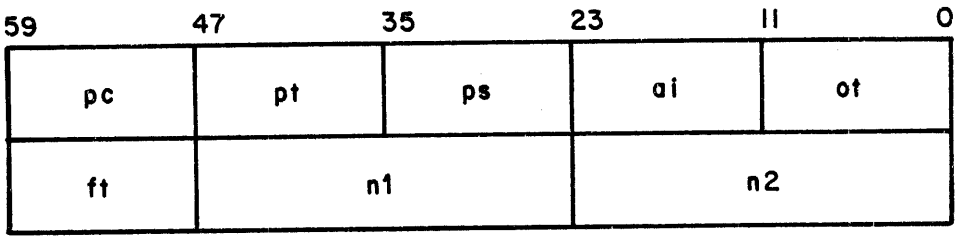
If a file is not found, an equipment number for the SDF device is extracted from the FET parameter word and validated, after which a FNT/FST is created. Next, 1IS attempts to set the global utility interlock and if successful, advances the active user count to prevent the device from being unloaded. If the interlock is unavailable, 1IS goes on recall (RLPW). A flashing message to this effect is sent to the B display. With the global interlock secured, 1IS checks the MST for presence of an existing SDF. If one exists, the deadstart sector is read via OPI, the SDF indicators cleared, the sector rewritten, the SDF track chain dropped, and the SDF flag in ACGL cleared. Routine 1IS now gets the length of the install file from its FET, and using SLM for the SDF device, determines the number of tracks required for the SDF. Using the RTCM function, 1IS requests all available disk space on the SDF device and counts the tracks in the allocated chain via COMPSEI. If the number of tracks assigned is insufficient to accommodate the install file, the track chain is released, the global interlock cleared (and user count decremented), and the job aborted with the following message.

TRACK LIMIT ON SDF DEVICE.

If more than the required number of tracks is available, 1IS drops into a loop, using COMPSNT and decrementing the assigned track count until it matches the required count. When this occurs, the first track for the SDF has been identified. Next the DLKM function is used to release the unneeded space from the beginning of the disk chain. Routine 1IS completes by setting first sector and first track equal to the current track in the FST.

Function 3 - Complete SDF Installation

Routine 1IS reads the SDF FNT/FST entry, changes file type to LIFT, and sets preserved file status on the SDF track chain. Next, the system sector is written and the FNT/FST entry cleared. The random address of OSB is extracted from the install file FET, and converted to logical track and sector (on the SDF file) via COMPCRA. Then the LDAM monitor function is used to calculate the physical address of OSB. Routine 1IS reads the deadstart sector via OPI, and sets parameters in the two-word common pointer table area reserved for the operating system. The layout of the parameter block is as follows.



pc Physical cylinder of OSB

pt Physical track of OSB

ps Physical sector of OSB

ai Algorithm index for device

ot Logical track containing OSB

ft First logical track of SDF

ni Next physical address when crossing logical track boundaries

After the deadstart sector is rewritten, 1IS sets the SDF flag in the ACGL word of the MST, sets a checkpoint on the SDF device, and completes by clearing the global interlock and decrementing the user count.

Function 4 - Process Mass Storage Error

Routine 1IS releases the SDF track chain, clears the global interlock, decrements the active user count, and aborts the job with the following message.

MS ERROR ON DEADSTART FILE.

DSD (dynamic system display) and DIS are display routines that require a dedicated PP. DSD is placed in PP1 by STL during deadstart and remains there while the operating system is running. DIS is called to a pool PP by the operator commands X.DIS, N.DIS or by the control statement DIS. when the system is in DEBUG mode and the user has system origin privileges.

When DIS is in a PP, it gets a control point and retains both the PP and the control point until it is ended by a DROP. or N.DROP. The console operator toggles between DSD and DIS by use of the * key. More information on the various commands and displays is included in the NOS Operator's Guide.

DSD and DIS use common decks, COMDSYS, COMDDIS, COMDDSP, and COMDTFN.

DYNAMIC SYSTEM DISPLAY (DSD)

DSD is loaded in PP1 at deadstart time and remains there throughout system operation. DSD provides an overall status display for all currently running jobs via the display console. The keyboard of the display console is monitored by DSD and is used for operator communication to the system (refer to figure 27-1).

DSD runs at the system control point; however, when an input requires operation on a job's control point area (change memory location or N.DROP., for example), DSD assumes the attributes of being assigned to the control point until the operation is complete.

If an operator typein requires some control statement action, it calls 1DS to initiate this action. If a typein specifies a particular display, DSD loads the appropriate overlay to fill the screen buffers.

As DSD receives input, it processes them one character at a time as they are received. Checking is performed on each character to validate the entry. DSD checks the first character and loads the proper syntax table overlay, if necessary. If, as the typein continues, the entry is determined to be unique, the remainder of the entry is filled in by the input processor. At this point, the entry is considered complete, and the keyboard echo line is flashed to indicate the complete entry.

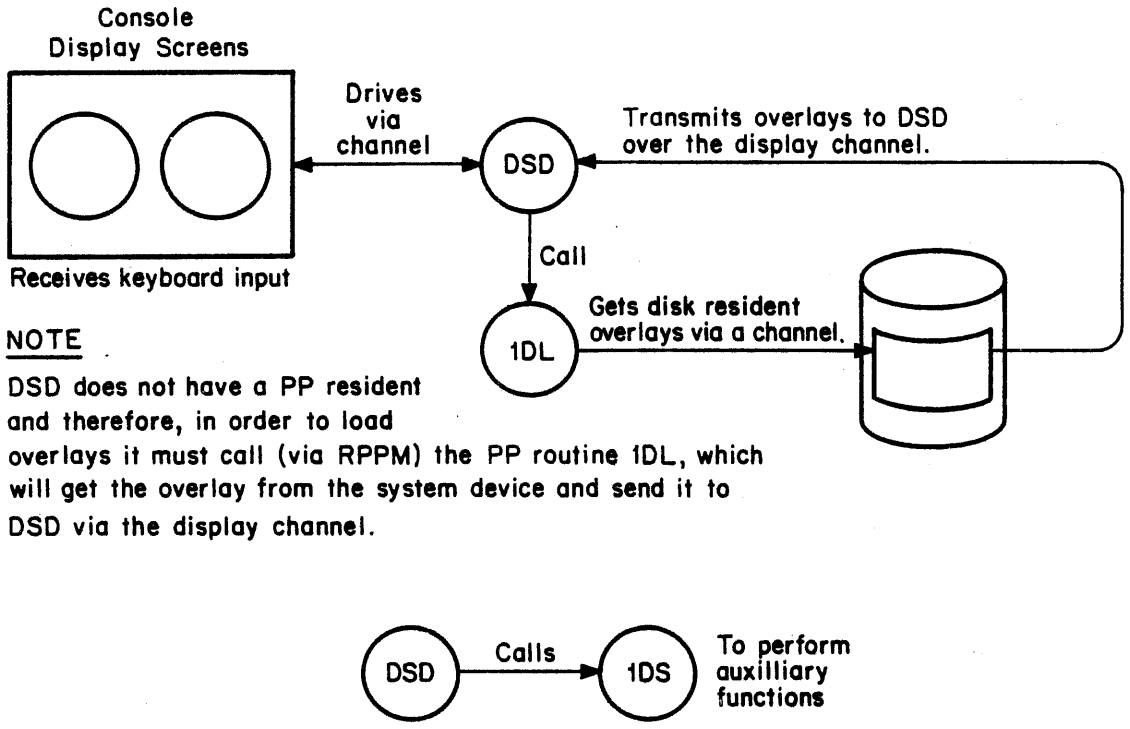


Figure 27-1. DSD Overview

The 99. command disables or enables syntax overlay processing and logging of DSD commands in the system and error log dayfile. When disabled, DSD does not load overlays to check syntax. This should only be done when the system is in an abnormal state to prevent PPs from being requested when they cannot perform the necessary tasks.

Each display is controlled by a separate overlay. If the same display is requested on each screen, there would be two copies of the same display overlay in memory at one time. Refer to section 28 for a description of the display screens.

Overlays may reside either in central memory (CM resident) or on the disk (disk resident - default). For those overlays that are disk resident, DSD calls the program 1DL to process the actual physical loading of the overlays. Routine 1DL then transmits the overlay to DSD via the display channel. The following overlays are recommended to be made CM resident: 9A1, 9A5, 9A6, 9A7, and 1DL. In addition, efficiency is increased if overlays 9AY, 9A0, 9A2, 9A3, and 9A4 are CM resident.

DSD uses three types of overlays; syntax, display, and command. Syntax and command overlays have absolute origins. Display overlays are written as location free routines since two display overlays must reside in DSD at one time for the two display screens.

STRUCTURE OF DSD

DSD is structured as follows:

- Resident command syntax table.
- Main program.
- Master display routine.
- Overlay loader.
- Display control tables.
- Keyboard input processing.
- Special character processing routines.
- Keyboard return processors.
- Resident display routines (G and D displays) that display central memory in 5 groups of 4 digits with display code translation.

- Resident command processors.
- Resident command processing subroutines.
- Tables and constants for overlay.
- Preset (overlaid after preset by command processors).
- Command Processor and syntax table overlay area.
- Left display overlay area.
- Right display overlay area.

The following are the command overlays.

Overlay	Description
9AY	N.SYNTAX table - characters A-C, E-N
9AZ	N.SYNTAX table - characters O-*
9A0	System syntax table, EN
9A1	System syntax table, B, C, D, E
9A2	System syntax table, F, I, K, L, M, O
9A3	System syntax table, P, R, S
9A4	System syntax table, T, U, V, W
9A5	Central memory changes
9A6	ECS memory changes
9A7	Channel commands
9A8	Send dayfile messages
9A9	Control point requests
9BA	Subsystem requests
9BB	TELEX message requests
9BC	BATCHIO requests
9BD	System requests
9BE	System requests
9BF	Job call requests
9BG	System control requests
9BH	ENABLE syntax table - A-N
9BI	ENABLE syntax table - O-Z
9BJ	DISABLE syntax table - A-N
9BK	DISABLE syntax table - O-Z
9BL	Enable/disable requests
9BM	Job control requests
9BN	Job control requests
9BO	Job control requests
9BP	Display change requests
9BQ	File control requests
9BR	File control requests
9BS	Resource control commands
9BT	Assign VSN to unit
9BU	Maintenance commands
9BV	Equipment availability commands
9BW	Mass storage validation
9BX	Enter time
9BY	Enter date

The following are the display overlays.

Overlay	Description
9AA	Display A, dayfile messages
9AB	Display B, system status
9AC	Displays F and G, central memory, 4 groups of 5 digits with display code translation
9AD	Display E, equipment status display
9AE	Display E, mass storage devices
9AF	Display E, mass storage devices
9AG	Display E, resource mounting previews
9AH	Display E, magnetic tapes
9AI	Display H, file name table
9AJ	Display I, BATCHIO status
9AK	Display J, control point status
9AL	Display K, central program buffer
9AM	Display M, ECS memory display
9AN	Display N, file display
9AO	Display O, transaction terminal status
9AP	Display O, sub-control point status
9AQ	Display O, task library directory
9AR	Display P, PP registers
9AS	Display Q, input, output and rollout queues
9AT	Display R, remote batch status
9AU	Display S, system control information
9AV	Display T, time-sharing status
9AW	Display Y, monitor functions
9AX	Display Z, directory

PROGRAMMING CONSIDERATION

The DISPLAY macro generates the linkage constants for display overlays. In addition, it generates the three-character name of the overlay via the OVLN macro.

The COMMAND macro generates the linkage constants for command processing overlays. Like the DISPLAY macro, it generates the three-character overlay name.

The ENTRY macro defines the name of the entry point within either a display or command overlay.

The ENTER macro specifies the format of the keyboard commands. Refer to a listing of DSD for use of special characters in the command syntax.

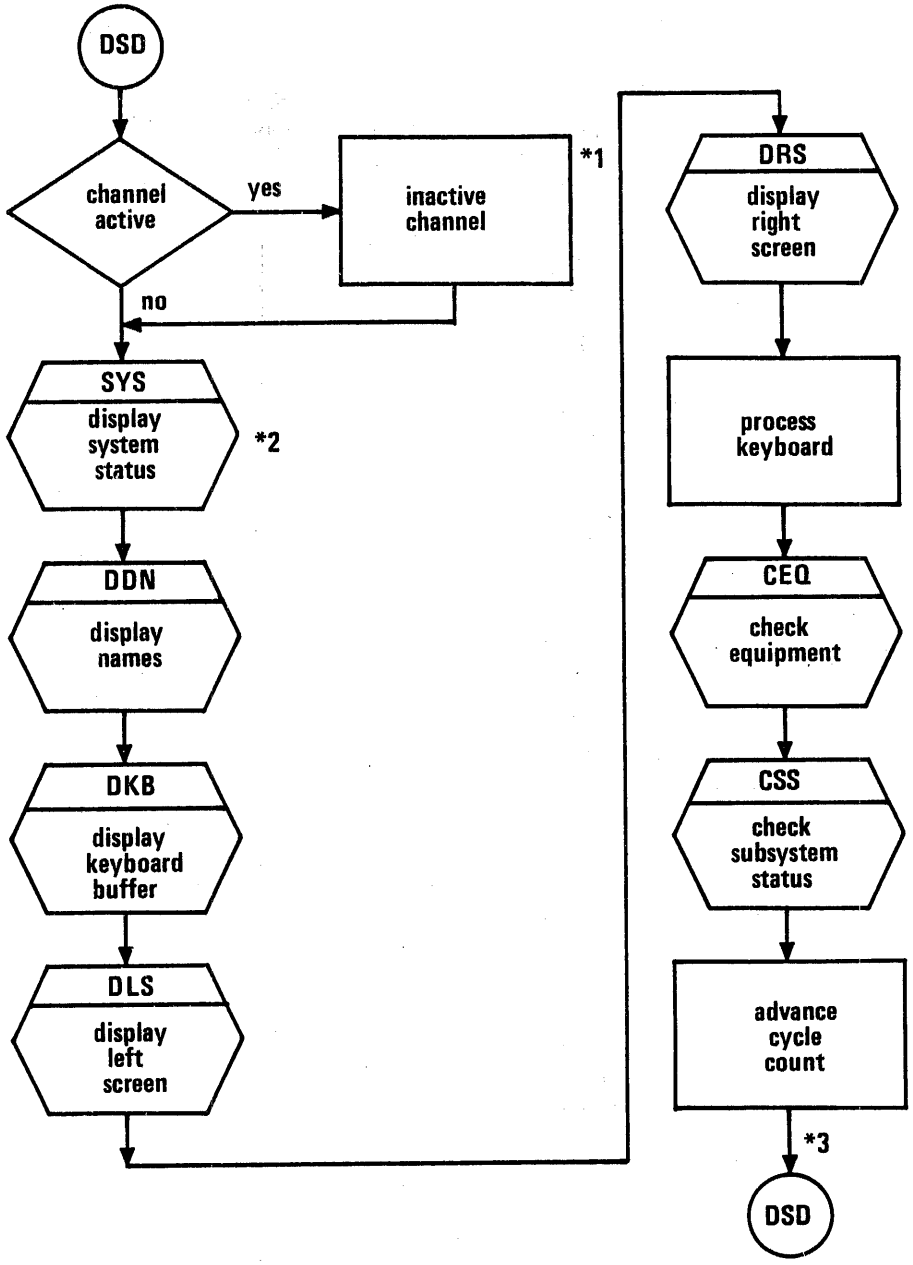
DSD must refresh the screen between 40 and 50 times a second to prevent flicker. Thus, display overlays should be short and efficient. If a command cannot be executed immediately, or requires special processing, DSD should call 1DS to process the command. Figure 27-2 is a flowchart of the main loop of DSD.

The * key is used to toggle between DSD and DIS sharing the same channel. Figures 27-3 and 27-4 are flowcharts showing how both routines get and release the channel. In DSD, when the * key is pressed, the hold flag (CEQB) in subroutine CEQ is cleared. Subroutine CEQ is entered each time through DSD's main loop. When DSD has the display channel, the display equipment (DS) is not assigned to any control point.

For DIS, when the * key is pressed, subroutine HDC is executed. When DIS is first loaded, it requests the equipment (DS). This causes DSD in subroutine CEQ to release the channel and then DIS requests the channel. DSD now loops waiting for the channel. When the * key is entered under DIS, DIS releases the channel but not the equipment. DSD senses this, requests the channel and sets the hold flag. DIS at this time is looping waiting for the channel. When the * key is pressed in DSD mode, the hold flag is cleared and DSD releases the channel and DIS gets it back.

ROUTINE 1DS

Routine 1DS processes those functions for DSD which are not possible for DSD to process. It is also called to enter jobs for 1AJ in certain cases, although not called directly by 1AJ.



*1 Ensure channel is free before attempting any action on it.
*2 COMDSYS display system status
*3 Loop. Screen must be referenced 40-50 times a second

Figure 27-2. DSD Main Loop

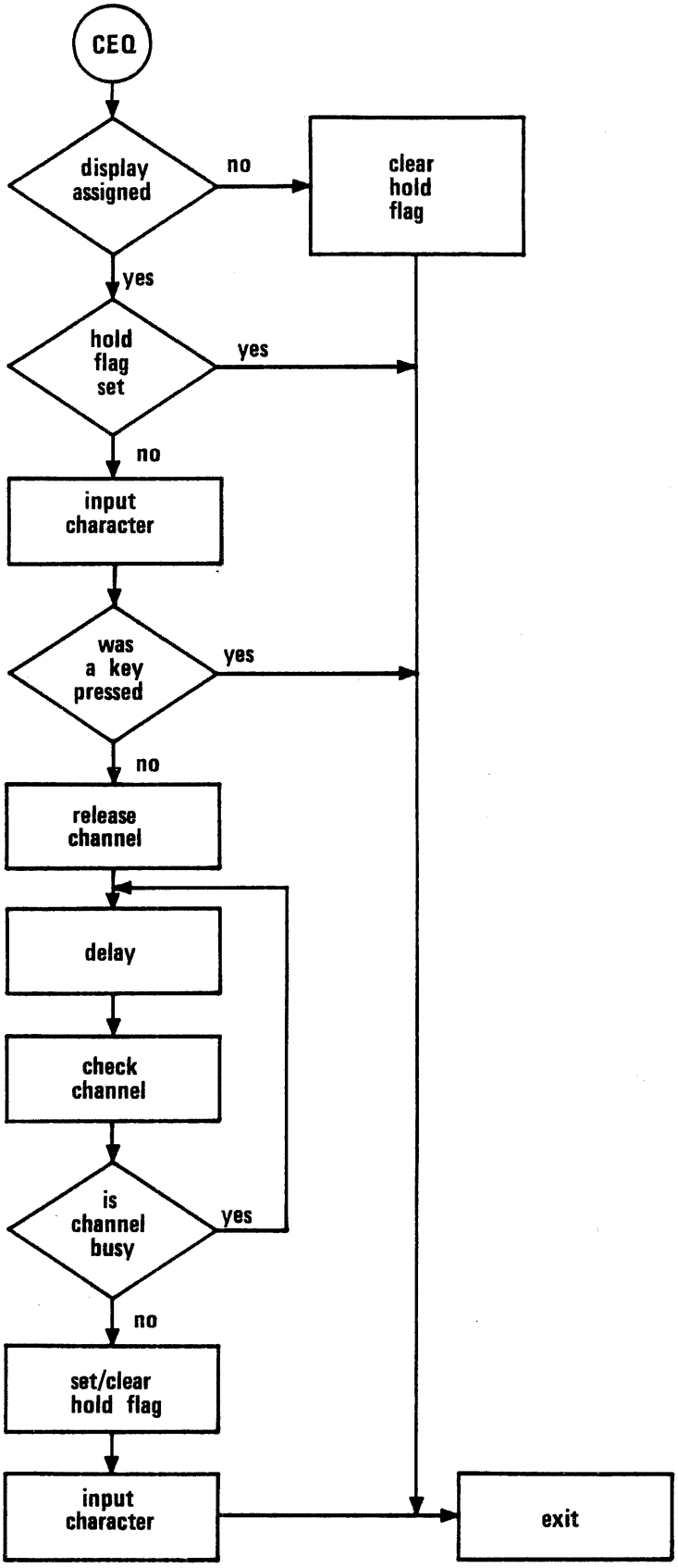


Figure 27-3. DSD Release/Request Channel Loop

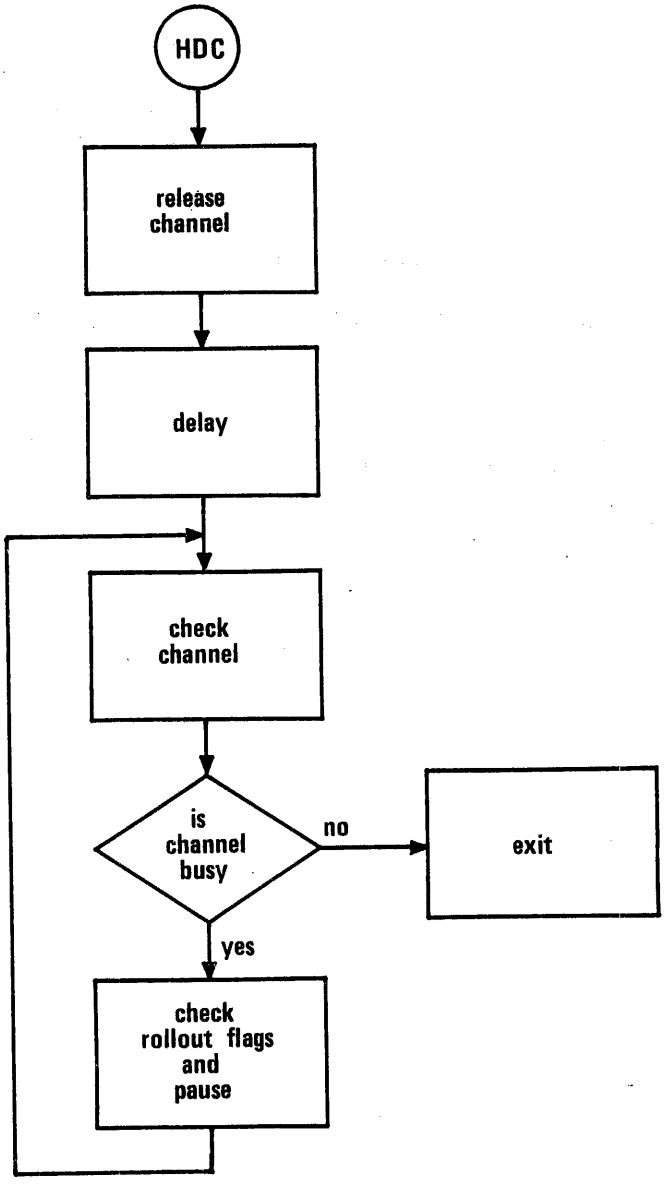
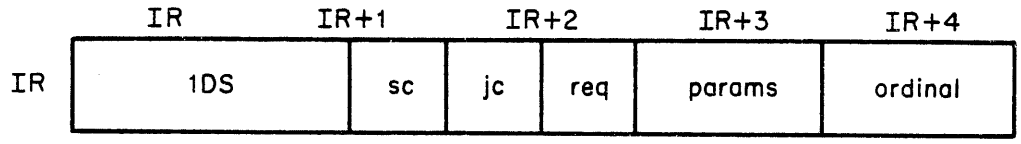


Figure 27-4. DIS Release/Request Channel Loop

The DSD call to 1DS is formatted as follows.



sc	System control point number.
jc	Zero if request for system control point, nonzero specifies control point to perform at.
req	Request.
params	Parameters for specific requests.
ordinal	FNT address of job if jc is nonzero.

Table 27-1 is a list of all current requests processed for DSD and table 27-2 lists the value of the params and ordinal for each request.

TABLE 27-1. TABLE OF REQUESTS

Request (Octal)	Description
0	Load display buffer
1	Send dayfile message
2	Go
3	On switch
4	Off switch
5	Enter central buffer
6	Purge files
7	Rerun job
10	Initiate jobs from table
11	Initiate job call
12	Dayfile dump
13	Account file dump
14	Error log dump
15	Load input jobs
16	Not used
17	Initiate control card job
20	Issue TELEX message
21	Issue TELEX warning message
22	Send TELEX user a message
23	Enter data to running job
24	Rollout job
25	Enter job CPU priority

TABLE 27-1. TABLE OF REQUESTS (CONTINUED)

Request (Octal)	Description
26	Enter job queue priority
27	Set job time limit
30	Assign equipment to job
31	Call DIS to job
32	Initiate specified subsystem
33	Initiate all enabled subsystems
34	Not used
35	Enter MAGNET UDT field
36	Toggle PF status
37	Call checkpoint to job
40	Format and send DSD message to error log and/or dayfile
41	Set/clear bits in MST word ACGL

TABLE 27-2. 1DS REQUEST

Request (Octal)	Params (IR + 3)	Ordinal (IR + 4)
0	Nonzero backspace file	FNT address; if zero, advance to next sector
1	FWA message	
2		
3	Switch number	
4	Switch number	
5	Address of message buffer	
6	File type if PURGEALL	FNT address if 1 requested
7	Rerun priority	
10		
11	Address of job name	Field length
12 .	Equipment number	
13	Equipment number	
14	Equipment number	
15	Equipment number	ID on FNT
16	Request number not used	
17	Address of job name	Field length
20	Address of message	
21	Address of message	
22	Address of message	Terminal number
23	FWA of message	
24	Rollout time, zero if not timed	
25	Priority	

TABLE 27-2. 1DS REQUEST (CONTINUED)

Request (Octal)	Params (IR + 3)	Ordinal (IR + 4)
26	Priority	
27	New time limit	
30	Equipment	
31		
32	Desired control point	Queue priority
33		
34		
35	Address of entry	
36	Bit to toggle or set	Equipment number
37		
40	FWA message	1-send message to error log, 2-send message to system dayfile, 3-send message to both
41	Bit to clear/set	Equipment ordinal

DIS DISPLAY PROGRAM

DIS is a display program that may be brought to an empty control point (via X.DIS.) to initiate utility programs or to an occupied control point to monitor the progress of a job. The main features of DIS are that while it is attached to a control point, the automatic advance of control statements is stopped unless set by the operator and that DIS does not drop when the error flag is set unless it is the operator drop flag. Hence the programmer is protected from losing the job if he enters invalid control statements, or other errors occur.

DIS provides an interpreted display of the exchange area for the job as well as the status of the job. Keyboard entries are provided to allow the user to alter central memory in several formats, and to execute control statements as if they had entered the system with an input file. In addition, breakpoint and 026 can be initiated via DIS, as well as other job related commands.

DIS, unlike DSD, does not have an interpretive command entry capability. If a command is not recognized as a legal DIS command, it is assumed to be a control statement and executed as such.

Only displays that are not overlays are permittted on the right screen since displays are not relocatable as they are in DSD.

Under the DSD section, a short discussion (with flowcharts) was given on the interaction of DSD with DIS if both share the same channel when using the * key. The following is a short discussion of what happens when the HOLD. command is issued under DIS mode. First, the equipment and channel are both released. When DIS has the channel, DSD is in a loop waiting for the channel. As soon as DIS releases it, DSD gets it. DIS then is in a loop waiting for the operator to assign the display equipment. As soon as the operator assigns the equipment (DS), DIS will request it and get it since it never is assigned to DSD. DSD will sense that the equipment has been assigned and release the channel. DIS will now get it back.

DIS can be called to a control point in one of three ways.

- DIS. control statement. If the running job has system origin privileges and the system is in DEBUG mode, DIS is loaded into a PP and assigned to the requesting job.

- N.DIS. This operator command brings DIS to a control point that is already occupied by a running job. If there is no job at the control point, the command is considered illegal. This method of calling DIS is known as the direct call.

- X.DIS. or X.DIS,xxxxxx. Where xxxxxx is requested field length. If either of these operator commands is used, DIS is brought to an empty control point. If the second form is used, the requested field length is assigned before giving control to the user. If the first form is used, 60K octal is the default field length requested.

Refer to the NOS Operator's Guide for information on commands and displays available under DIS.

The main loop of DIS is flowcharted in figure 27-5.

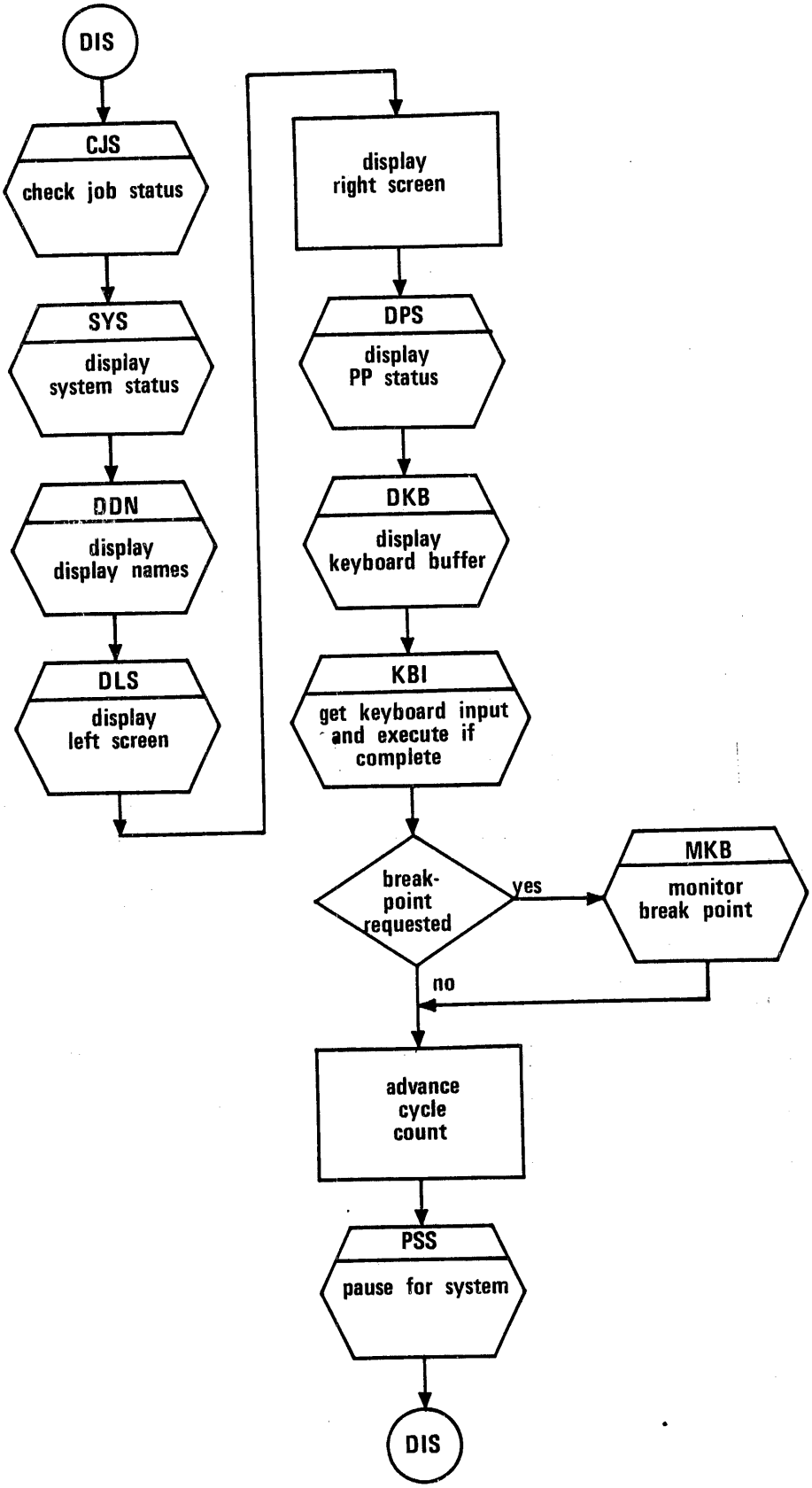


Figure 27-5. DIS Main Loop

STRUCTURE OF DIS

DIS is structured as follows. Resident routines include:

- Main program and subroutines.
- Keyboard input.
- Keyboard input subroutines.
- Interpret keyboard message.
- Resident display routines:

Display	Description
B	Control point status.
C	Central memory (5 groups of 4 digits with display code translation).
D	Same as display C.
N	Blank screen.
T	Text display. Displays text from central memory in coded lines. Lines are folded at 60 characters. Display is terminated after 256 words have been displayed.
U	Text display. Same as Display T.

- Display subroutines
- Display common decks: COMDDIS, COMDDSP, COMDSYS
- Resident command processor routines:

Routine	Description
DIS.	Call DIS and drop display
DROP.	Drop display
ERR.	Set error flag PPET
HOLD.	Drop display and wait for reassignment

- Command processor subroutines.
- Table of displays.
- Preset program (overlaid).

Overlay areas for displays include the following:

Overlay	Display	Description
9EA	A	Dayfile
9EB	E	Magnetic tapes (E,T display for DSD)
9EC	F	Central memory (4 groups of 5).
9ED	G	Central memory (4 groups of 5).
9EE	H	File name table (job).
9EF	J	System status.
9EG	K	Equipment status table.
9EH	L	File name table (system).
9EI	M	ECS memory display.
9EJ	P	PP registers.
9EK	Q	Input/output and rollout queues.
9EL	V	Central memory buffer.
9EM	Y	Monitor functions.
9EN	Z	Directory.

Overlay areas for commands include the following:

Overlay	Description
9E0	CPU commands
9EP	Statement entry
9EQ	Execute statements
9ER	Enter registers
9ES	Enter X register
9ET	Enter memory
9EU	Enter instruction
9EV	Enter ECS
9EW	Enter memory/enter instruction exit
9EX	CPU program interface commands
9EY	Enter field length
9EZ	Enter ECS field length
9E0	Call 026 to control point
9E1	Miscellaneous commands
9E2	Miscellaneous commands
9E3	Interpret keyset message
9E4	Check keyboard request
9FA	Interpret more messages
9FB	Call PP program

OVERLAY RESIDENCY AND 1DL

Unless DIS is used heavily, it is not necessary to make any of the overlays central memory resident.

In order to load any disk resident overlay, DIS employs the assistance of 1DL (same routine that DSD uses) to do the actual load from disk. After 1DL has the overlay in its memory, then it is transferred to DIS's memory via the display channel.

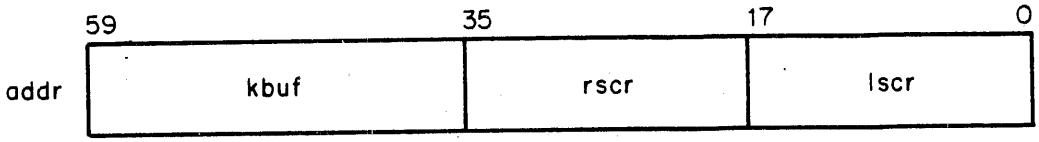
CONSOLE COMMUNICATION

The CPU programmer can display information on the K display and receive operator keyboard input with the CONSOLE macro. This macro causes the display of a specially formatted, central memory buffer. The format of the CONSOLE macro is as follows:

LOCATION	OPERATION	VARIABLE SUBFIELDS
	CONSOLE	addr

addr Address of parameter word

The parameter word, addr, must be defined by the user as follows:

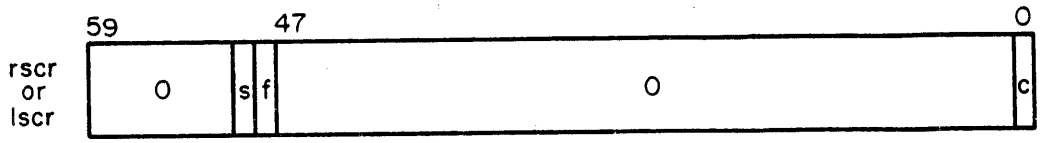


kbuf Address of 8-word area where operator keyboard input will be written in the user's program

rscr Address of specially formatted, right-screen buffer

lscr Address of specially formatted, left-screen buffer

The left- and right-screen data buffers must be preceded by control words that are formatted as follows:



s — Character size. If zero, indicates small characters (64 characters/line). Characters are eight units apart and lines are 10 units apart (refer to Display in this section). If s equals one, indicates medium characters (32 characters/line). Characters are 16 units apart and lines are 20 units apart.

f — Format. If f = 0, specified program format. After the display is selected, data is output until a zero is encountered in byte 0 of a word or until 512 words have been output. The data must contain all coordinates (refer to Display). If f = 1, coded format (C format) is specified. The buffer is assumed to be in C format (line is terminated when byte 4 of a word contains a zero) and is output until a zero is encountered in byte 0 of the first word of a line, or until 512 words have been displayed. Coordinates do not have to be specified. The data will be displayed starting with line 43 through line 7 (37 lines).

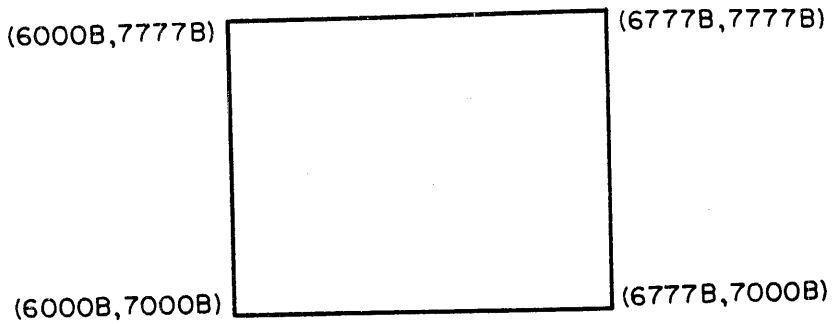
c — If this is preset to zero, it may subsequently be checked for nonzero which indicates data has been displayed at least once. In this way, the user can tell when the K display was actually assigned to his control point.

The CONSOLE macro causes the system to put out the message on the B display requesting the K display for this job.

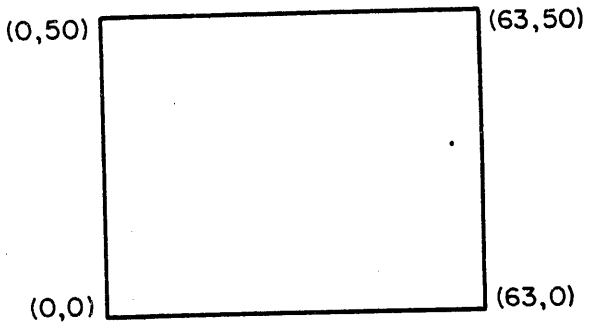
DISPLAY SCREEN

The display screen is divided into a grid that consists of 51 lines and 64 columns (based on small characters). The spacing between columns is eight coordinate positions or units, and between lines is 10 coordinate positions or units. The areas of the display screen (left or right) are limited to those lines above line 4 and below line 46. If the user displayed information outside of these limits, the display headers could be destroyed as well as other system information that is normally displayed on the bottom of each screem.

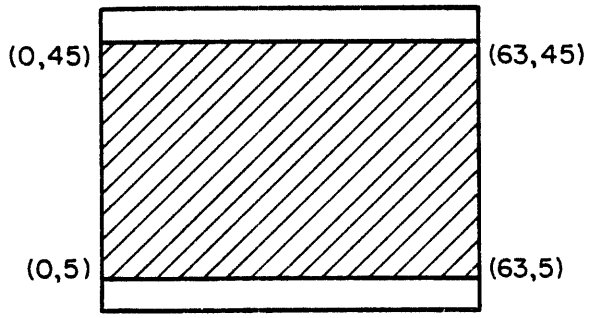
The display grid can be laid out using X and Y coordinates, where (6000B, 7000B) is the lower-left point of reference, as follows.



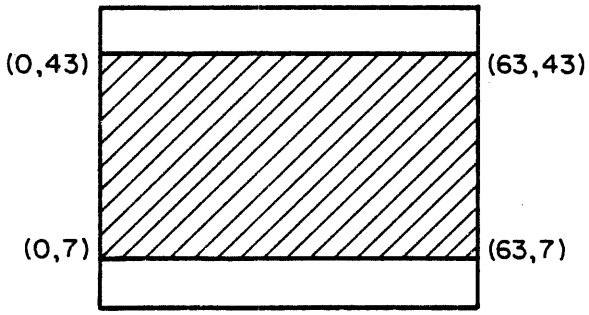
In terms of lines and columns, the display screen is formatted as follows. There are 51 lines each consisting of 64 characters (small size).



The user should only use lines above line 4 and below line 46. This means the first line that should be used at the top is line 45 and the last line t- be used at the bottom is line 5, as follows.



The area that is used if the coded format is selected is as follows.



The X and Y coordinates must be specified for program format and must be the first two bytes of each line; otherwise the data is displayed starting with the last value of the X and Y coordinates. To simplify the process of calculating the exact X and Y coordinates for a particular line, the DSL macro can be used (available on common deck COMCMAC). The format of the DSL macro is as follows.

LOCATION	OPERATION	VARIABLE SUBFIELDS
	DSL	x,y,string

x	Column or X-coordinate (character position 0 through 63 for small characters)
y	Line or Y-coordinate (line number 0 through 50 for small characters)
string	Character string to be displayed (assumes 64 characters per line)

NOTE

Medium characters are twice the size of small characters.

Refer to the sample program at the end of this section for the use of the CONSOLE and DSL macros.

DISPLAY PROGRAMMING

In order to have some parts of the display at a higher intensity, the user can output the same line two or three times. For example, if line 45 is to be intensified, use the following.

```
DSL     0,45,data
DSL     0,45,data
DSL     0,45,data
```

Flashing of selected parts of the display can be accomplished with the following procedure (also refer to the sample program for implementation of this feature). Since any word containing zero in byte 0 acts as an end-of-buffer, replacing the first word (X,Y coordinates plus first 6 characters of data) of the last line(s) to be displayed with a zero word causes the last line(s) not to be displayed. Then based on some counter, replace the zero word with the original contents and the line(s) will be displayed again. Do this alternately and a flashing message is created. Since the zero word indicates an end-of-buffer, the line(s) to be flashed must be the last set of data in the buffer. Another method is to put the duplicate lines immediately before the K-display buffer and change the pointer to the first word of the buffer.

When displaying using the program format, it is not necessary to display the lines in numerical order; that is, line 45 followed by 44, then 43, etc. The user can display any line or any part of a line in any order, since each line must begin with an X,Y coordinate and, if desired, a part of a line can begin with an X,Y coordinate. The X,Y coordinates can appear anywhere in the user's data buffer -- the 6xxx means an X coordinate, the 7yyy means a Y coordinate.

In addition to the sample program in this section, the user can find examples in routines MODVAL, PFS, and QFSP.

KEYBOARD INPUT

When receiving information from the keyboard, the buffer (KBUF in this case) is filled with characters when the carriage return (CR) key is pressed. Characters are transmitted to KBUF from the keyboard buffer in DSD left-justified, 10 characters per word. The last word is not filled beyond the final keyboard entry. Therefore, if the user zeros KBUF prior to receiving data, the first 6 bits of zero will signal the end-of-information.

If the user's CPU program needs to wait for keyboard input, a RECALL should be issued as in figure 28-1.

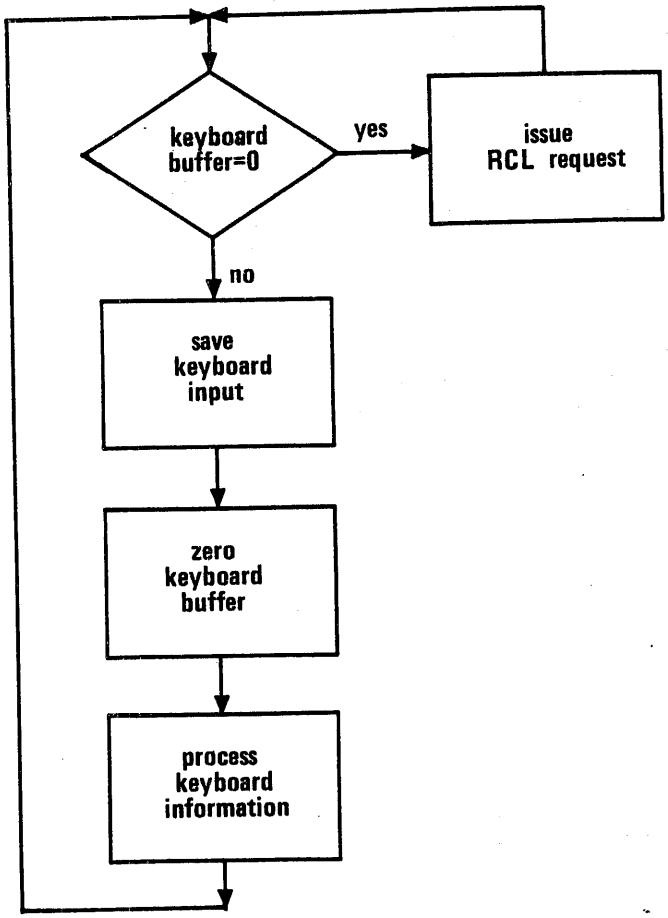


Figure 28-1. Sample Keyboard Main Loop

K-DISPLAY STANDARDS

If the programmer creates a K-display processor for the operating system or changes an existing processor, the guidelines described in the following paragraphs should be followed.

A K-display processor is defined as a program that interacts with the operator by means of a K display. The function associated with a particular processor is defined as a process. For example, MSI is a process and its process is initializing devices; PROFILE is a processor and its process is updating charge numbers in the profile file.

Two types of processes exist, dynamic and static. A dynamic process processes each directive as it is entered at the keyboard. The operator must indicate to the processor when to complete the process by typing a special command. Updating a charge number with PROFILE is an example of a dynamic process. A static process is where a complete set of directives must be entered before the process may be initiated. The operator must indicate to the processor when to initiate the process by typing in a special command. Initializing a device with MSI is an example of a static process.

K-DISPLAY ENTRIES

Directives and commands are the two types of K-display entries a processor is concerned with. Directives set the values of the parameters needed by the processor to complete a process. Directives generally take the form of dd=xxxx. Keywords should consist of two letters. An equal sign separates the keyword from the associated values. Standard keywords include the following.

Keyword	Description
UN	User number
UI	User index
DA	Date
FM	Family name
DT	Device type
DN	Device number
EQ	Equipment number
DM	Device mask
DE	Density
DD	Destination device number

More than one directive should be allowed on the same line.

Commands are instructions to the processor. This type of K-display entry tells the processor to initiate, terminate, or proceed with the current process. They do not affect the value of parameters needed by the processor to complete the process. Commands usually consist of a single keyword.

The following commands should be available for dynamic processors.

Command	Description
END	Terminate input of directives; complete the particular process. A new process begins with the entry of the next directive.
DROP	Terminate input of directive for current process. Do not complete the process, but rather ignore it. A new process begins with the entry of the next directive.
STOP	Terminate the program; do not complete process if one is in progress.

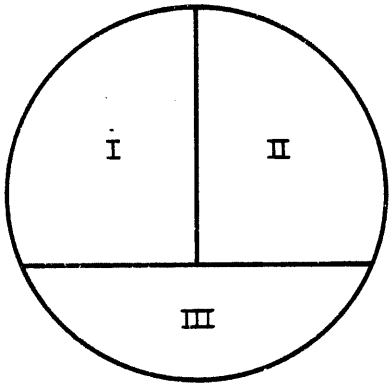
The following commands should be available for static processors.

Command	Description
GO	Proceed with the process using the directives that have been entered. The processor informs the operator of the completion of the process by asking for a new set of directives for the next process, or, if there is no next process, to end the run.
RERUN	Reinitialize the processor.
RESET	All directives are reset to the default values.
+	Page K display forward one page.

K-DISPLAY FORMAT

The left screen is the primary screen for the K-display processor. The right screen should be used for related information an operator would need for efficient use of the processor. This information should go on the right screen only if there is not enough room on the left screen.

The primary screen can be divided into three general areas, as follows.



Area I contains the directives that are accepted by the processor and the value currently assigned to that directive. As each directive is entered by way of the keyboard, the displayed value is updated to reflect any change.

Area II contains a description of each of the directives found in area I. This description should include what values the directive may assume. The directive and its value in area I should be on the same line as its description in area II.

Area III consists of at least two lines. The bottom line should be used to display the directives that have just been entered. The top line should be used to inform the operator of the status of the processor. The status would include any error messages, informative messages, or requests from the processor.

SAMPLE PROGRAM

The sample program executes the CONSOLE macro and the system requests the K display to be brought to the left screen (refer to figure 28-2).

The operator would type K,2. The first part of this program uses medium characters. The K display is brought to the left screen and the user's job displays the information as shown in figure 28-3. The left screen display asks the operator to bring the K display to the right screen also; he types in KK. Figure 28-4 shows what the two screens look like at this point.

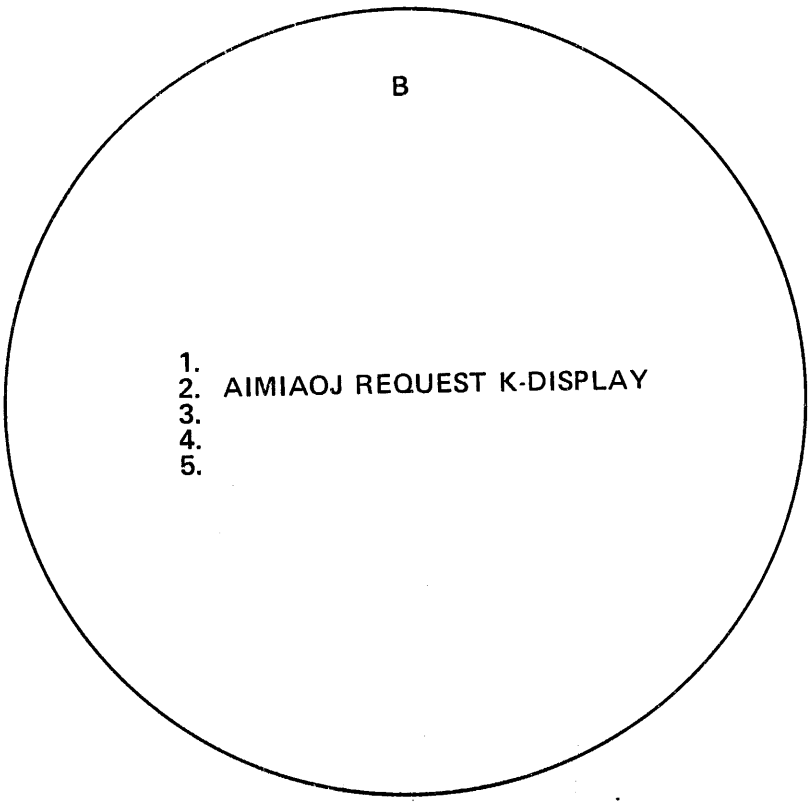


Figure 28-2. B Display

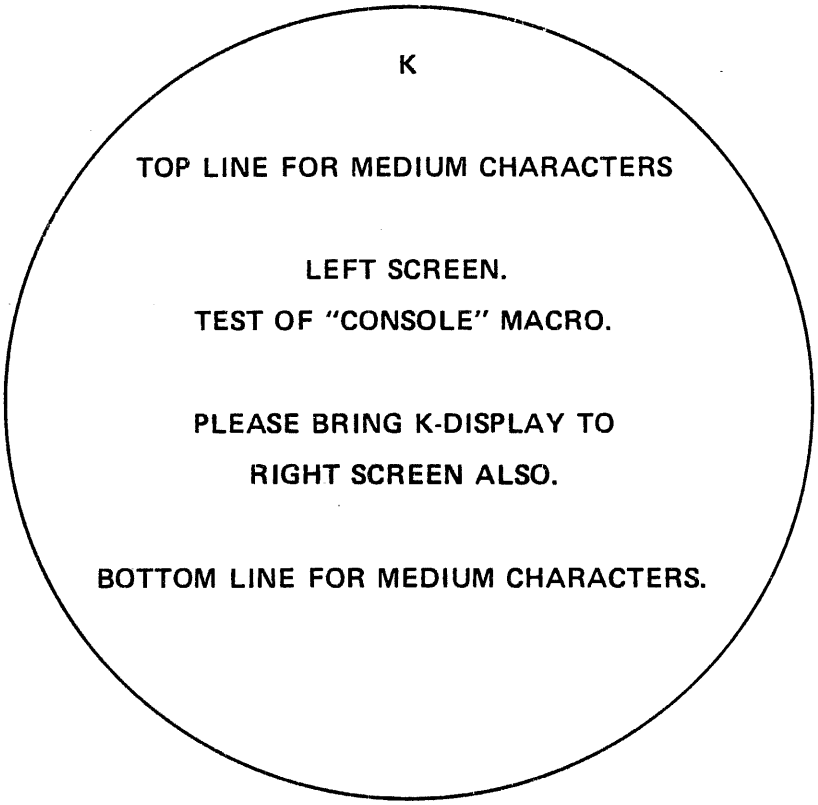


Figure 28-3. K Display, Left Screen

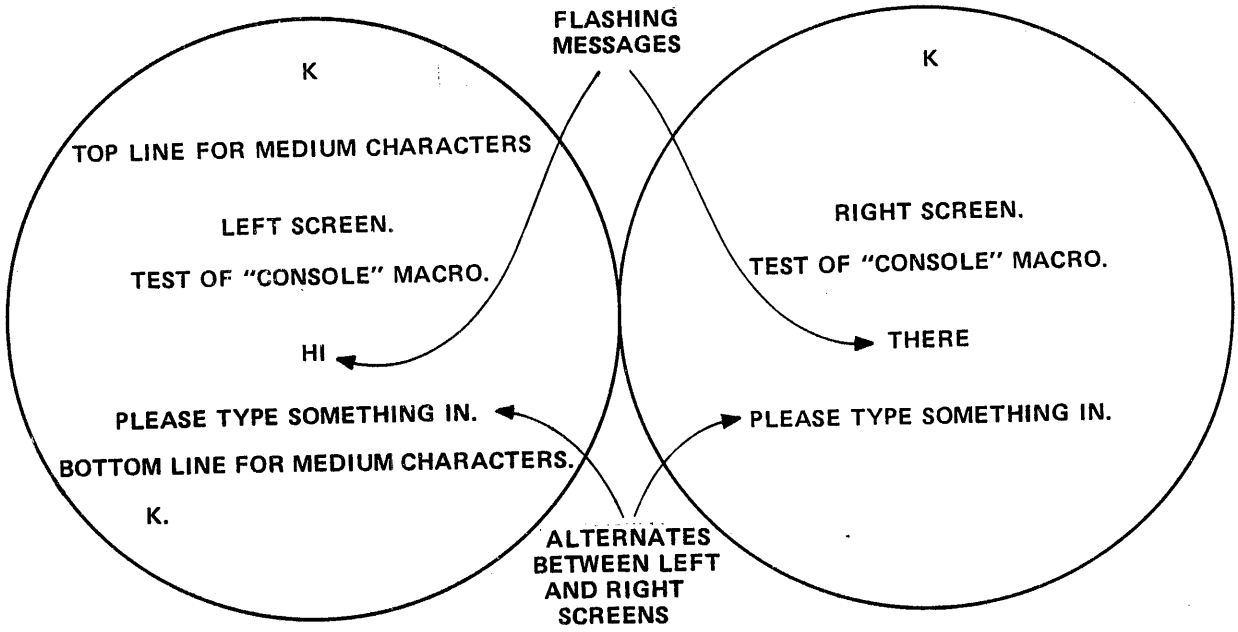
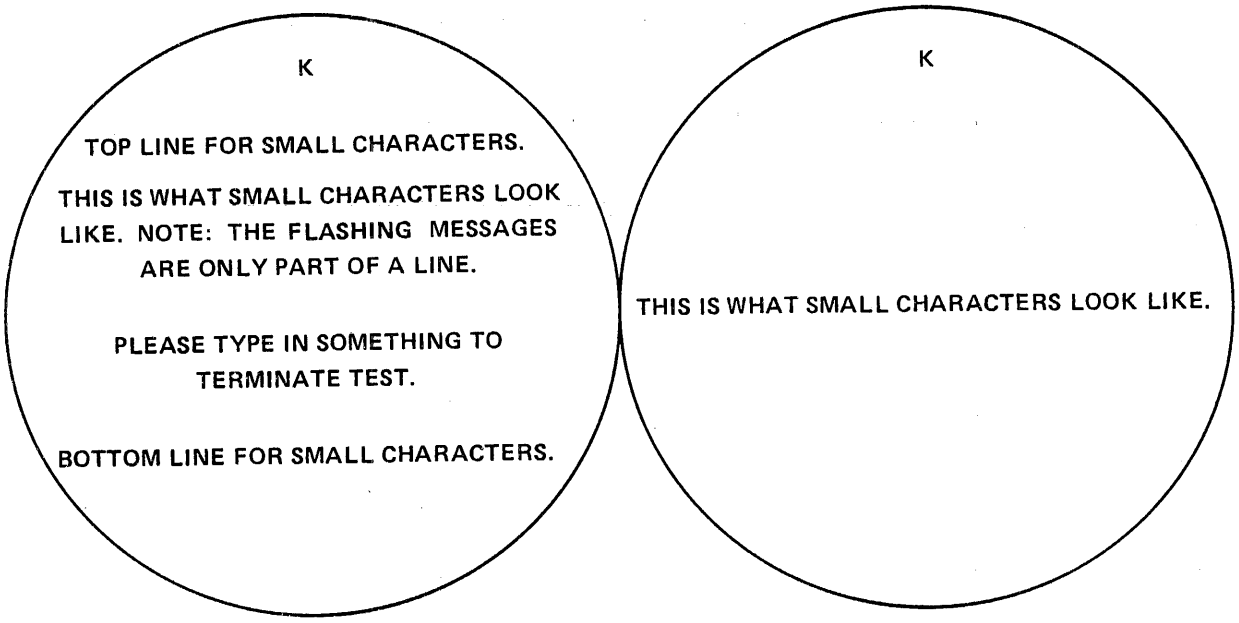


Figure 28-4. K Display, Left and Right Screens

In the middle of each screen is a flashing message (HI on the left screen and THERE on the right screen). Below the flashing messages is another request for the operator which alternates between the left and right screen. The operator types in any messge terminated with a CR. This causes the program to print out INFORMATION RECEIVED in the user's dayfile and the content of the screens is changed. The next part of the program uses small characters and demonstrates the use of flashing parts of lines instead of entire lines. Figure 28-5 illustrates this display.

To terminate the program, the user types in any message terminated with a CR. The following messges are put in the user's dayfile before ending.

```
INFORMATION RECEIVED
END OF TEST
```



NOTE

Words SMALL and NOTE flash on the left screen, along with input request.

Figure 28-5. Small Characters, Left and Right Screens

```
          IDENT  KDS
          ENTRY  KDS
          SYSCOM B1

*****     THIS PROGRAM USES THE DSL AND CONSOLE MACROS FROM
*         COMMON DECK COMCMAC TO DISPLAY MESSAGES ON THE
*         LEFT AND RIGHT SCREENS VIA THE K DISPLAY.


*         THIS FIRST SET OF DISPLAYS USES MEDIUM CHARACTERS.

 KDS      SB1    1
          CONSOLE DSW1        SYSTEM WILL REQUEST *K* DISPLAY

 KDS1     SA1    MLS
          LX1    59-0
          NG     X1,KDS2      IF LEFT SCREEN DISPLAYED
          RECALL
          EQ     KDS1         LOOP UNTIL LEFT SCREEN IS DISPLAYED

 KDS2     SA1    BLK
          BX6    X1
          SA6    MLS1         DISPLAY MESSAGE TO OPERATOR

 KDS3     RECALL
          SA1    MRS          CHECK RIGHT SCREEN
          LX1    59-0
          PL     X1,KDS3      IF NOT DISPLAYED YET

*         THE FOLLOWING CODE PUTS OUT A FLASHING MESSAGE ASKING FOR
*         AN OPERATOR TYPE-IN.  THE MESSAGE IS ALTERNATED BETWEEN THE
*         LEFT AND RIGHT SCREENS.

          MX6    0
          SA6    MLS1
          SB7    B0
          SB5    LTYP

 KDS4     SA1    TYP+B7       MOVE OPERATOR MESSAGE TO DISPLAY AREA
          BX6    X1
          SA6    MLS2+B7
          SA6    MRS2+B7
          SB7    B7+B1
          NE     B5,B7,KDS4   IF NOT FINISHED
          SA1    BLK
          BX6    X1
          SA6    MLS1

 KDS5     RECALL
          SA1    KBUF
          NZ     X1,KDS7      IF SOME KEYBOARD INPUT
          SA1    BLK
          SA2    FLS
          SA3    MLS1
          SX6    X2-1         DECREMENT FLASH COUNTER
          SA6    FLS
```

```
          NZ      X6,KDS5       IF NOT TIME TO CHANGE FLASH STATUS
          SX7     FLC           RESET FLASH COUNTER
          SA7     FLS
          ZR      X3,KDS6       IF LEFT DISPLAY OFF
          MX6     0             LEFT DISPLAY ON ....
          SA6     MLS1          SO TURN IT OFF
          BX6     X1
          SA6     MRS1          TURN ON RIGHT SCREEN FLASH
          EQ      KDS5

KDS6      MX6     0             TURN OFF RIGHT SCREEN FLASH
          SA6     MRS1
          BX6     X1
          SA6     MLS1          TURN ON LEFT SCREEN FLASH
          EQ      KDS5          LOOP WAITING FOR KEYBOARD INPUT

*         THE FOLLOWING CODE WILL DISPLAY SMALL CHARACTERS AND
*         WILL DEMONSTRATE THE CAPABILITY OF FLASHING PARTS OF
*         LINES INSTEAD OF ENTIRE LINES.

KDS7      MESSAGE KDSA,,R
          MX6     0
          SA6     KBUF          CLEAR KEYBOARD BUFFER
          SX6     FLC
          SA6     FLS           RESET FLASH COUNTER
          CONSOLE DSW2

KDS8      SA1     SLS
          LX1     59-0
          NG      X1,KDS9       IF LEFT SCREEN DISPLAYED
          RECALL
          EQ      KDS8          LOOP

KDS9      RECALL
          SA1     KBUF
          NZ      X1,KDS11      IF SOME KEYBOARD INPUT
          SA1     BLK
          SA2     FLS
          SA3     SLS1
          SX6     X2-1          DECREMENT FLASH COUNTER
          SA6     FLS
          NZ      X6,KDS9       IF NOT TIME TO CHANGE FLASH STATUS
          SX7     FLC
          SA7     FLS
          ZR      X3,KDS10      IF FLASH PARTS OFF
          MX6     0             FLASH PARTS ARE ON - TURN OFF
          SA6     SLS1
          EQ      KDS9

KDS10     BX6     X1            FLASH PARTS ARE OFF - TURN ON
          SA6     SLS1
          EQ      KDS9

KDS11     MESSAGE KDSA,,R
          MESSAGE (=C* END OF TEST.*),,R
          ENDRUN
```

```
KDSA    DATA    C* INFORMATION RECEIVED.*

BLK     DATA    10H
FLC     EQU     100             FLASH COUNTER VALUE
FLS     CON     FLC             FLASH COUNTER
DSW1    VFD     24/KBUF,18/MRS,18/MLS


DSW2    VFD     24/KBUF,18/SRS,18/SLS



KBUF    BSSZ    8
MLS     VFD     10/0,1/1,1/0,47/0,1/0
        DSL     0,44,(TOP LINE FOR MEDIUM CHARACTERS.)
        DSL     0,5,(BOTTOM LINE FOR MEDIUM CHARS.)
        DSL     8,39,(LEFT SCREEN.)
        DSL     8,39,(LEFT SCREEN.)
        DSL     0,37,(TEST OF *CONSOLE* MACRO.)
MLS1    DATA    0
        DSL     16,33,(HI)
        DSL     16,33,(HI)
MLS2    DSL     0,29,(PLEASE BRING K DISPLAY TO)
        DSL     0,27,(RIGHT SCREEN ALSO.)
        DATA    0
TYP     DSL     0,27,(PLEASE TYPE SOMETHING IN.)
        DSL     0,27,(PLEASE TYPE SOMETHING IN.)
        DSL     0,27,(PLEASE TYPE SOMETHING IN.)
        DATA    0
LTYP    EQU     *-TYP
MRS     VFD     10/0,1/1,1/0,47/0,1/0
        DSL     8,39,(RIGHT SCREEN.)
        DSL     8,39,(RIGHT SCREEN.)
        DSL     0,37,(TEST OF *CONSOLE* MACRO.)
MRS1    DATA    0
        DSL     16,33,(THERE...)
        DSL     16,33,(THERE...)
MRS2    BSSZ    LTYP

SLS     VFD     10/0,1/0,1/0,47/0,1/0
        DSL     0,45,(TOP LINE FOR SMALL CHARACTERS.)
        DSL     0,5,(BOTTOM LINE FOR SMALL CHARACTERS.)
        DSL     3,30,(    THIS IS WHAT)
        DSL     25,30,( CHARACTERS LOOK LIKE.)
        DSL     6,28,(THE FLASHING MESSAGES ARE ONLY PARTS OF A LINE.)
SLS1    DATA    10H
        DSL     19,30,( SMALL)
        DSL     19,30,( SMALL)
        DSL     0,28,(NOTE- )
        DSL     0,28,(NOTE- )
        DSL     0,16,(PLEASE TYPE IN SOMETHING TO TERMINATE TEST.)
        DATA    0
SRS     VFD     10/0,1/0,1/0,47/0,1/0
        DSL     3,30,(    THIS IS WHAT SMALL CHARACTERS LOOK LIKE.)
        DATA    0
        END     KDS
```

During execution, a PP routine may need some special operations performed. Depending on the routines, different areas of PP memory may be available for loading special routines.

In order to load a PP routine anywhere in PP memory, the concept of location-free routines is used.

Two macro packages, COMPREL and COMPRLI, provide the capability for a routine to be self-relocating.

By convention, any PP routine whose name begins with a zero is considered a location-free routine. A routine that needs to load a location-free routine sets LA (load address direct cell) to the location where the subroutine is to reside, sets the A register to the name of the routine, and calls EXR to load and execute it. The EXECUTE macro from COMPMAC accomplishes this for the user. There are two common decks that control the use of location-free routines, COMPREL and COMPRLI.

COMMON DECK COMPREL

If the user inserts common deck COMPREL in his program, all M-type instructions automatically have LA inserted in the d field. Hence, the user may not specify a d field in any M-type instruction. In addition, CRM, CWM, AJM, IJM, FJM, EJM, IAM, and OAM cannot be used. If the user needs to specify an M-type instruction without relocation ordinal definition, he must append a period onto the instruction, as follows

```
LJM. tag
```

Any M-type instruction which references a cell defined in NOSTEXT (PPCOM) is not relocated if REL$ is defined equal to 1; otherwise, the instruction will be relocated. If the user wishes to code nonrelocatable code after his relocatable code, he uses the macro RSTR, which is contained in COMPREL. COMPREL relocates instructions with reference to LA as they are encountered in the code. This then causes all relocation to occur at execution time through the indexed direct-addressing scheme.

COMMON DECK COMPRLI

The second method of coding a location-free routine is to use COMPRLI, which relocates indirectly. All the rules of COMPREL apply, with the exception that it is legal to relocate I/O instructions. In addition, the three C-type instructions, LDC, ADC, and LMC, are also relocatable.

Where COMPREL relocates instructions as it encounters them, COMPRLI builds a remote table using the RMT pseudo-op (refer to the COMPASS Reference Manual) containing the address of the instructions that need to be relocated. The first executable statement must be:

```
RJM. REL,LA
```

The routine REL is in COMPRLI. REL searches through the remote table and relocates all instructions whose addresses are stored in the table. The user must call COMPRLI.

Listings of COMPREL and COMPRLI are obtained by assembling CALLPPU.

The following is a list of location-free routines available in NOS.

Routine	Description
OAU	Update project profile file
OAV	Verify user name
OBF	Begin file
OBP	Banner page
ODF	Drop file
OFA	Release fast-attach files
OMF	Initialize MMF link device
ORF	Update resource files
ORP	Release permanent file
OSE	Process system device errors
OVJ	Verify job/user statements
OCI	Firmware ident processor (6DI driver)
OTI	Track flaw processor (6DI driver)
OPI	Pack serial processor (6DI driver)

LOADING ZERO-LEVEL OVERLAYS

In NOS, location-free routines are called zero-level overlays. Since location-free routines can reside anywhere within PP memory, it is quite important that the caller of a location-free routine guarantee that the loading of such a routine does not destroy meaningful data or instructions. Common deck COMSZOL defines symbols for the length of the zero-level overlays. These values represent the number of words needed by the routine to perform its task. This value is less than or equal to the amount of PP memory used to load the overlay. The amount of memory destroyed is rounded to the next highest number of sectors if the routine is loaded from RMS (that is, if the routine requires 501B bytes, 1200B bytes are destroyed by the load from RMS). The user should employ the OVERFLOW macro, found in COMPMAC and COMSZOL, to guarantee that the memory destroyed by the loading of a zero-level overlay is not meaningful. Listings of COMPMAC and COMSZOL can be obtained by assembling CALLPPU and CALLSYS.

NOS supports the CYBER common product set. Since the product set executes under several operating systems, there are some operations done by the products which are processed by various PP routines that may not be present on all operating systems.

From a historical perspective, the predecessor versions of NOS (KRONOS 2.1) did not have PP routines that were called by the product sets running on the predecessor to NOS/BE (SCOPE 3.4). When the common product set was supported under KRONOS, the processors for RA+1 calls not available under KRONOS were grouped into a single routine named the SCOPE function processor (SFP).

SCOPE FUNCTION PROCESSOR

SFP is a function processor that is called when the PP program requested cannot be found in the peripheral library directory (PLD).

The RA+1 calls processed by SFP consist of the following.

Call	Description
STS	Status processor
MSD	SDA/SIS message generator
PFE	Alter function
ACE	Advance control statement
PRM	Permission check
CKP	Checkpoint request
REQ	NOS/BE equipment request
DMD	Dump CM field length in display code
DMP	Dump CM field length
DOO	Error text processor
FIN	Translate FIN (NOS/BE) requests
DEP	Dump ECS field length
DED	Dump ECS field length in display code

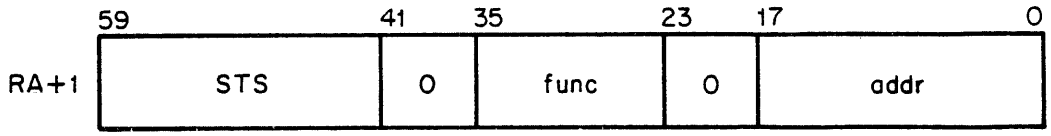
SFP STRUCTURE

SFP consists of a main program and overlays that are loaded depending upon the function being processed.

The main program in SFP determines which overlay needs to be loaded by processing the RA+1 call.

STS REQUEST

The STS request is processed by overlay 2SA. The RA+1 format for the STS call is as follows.

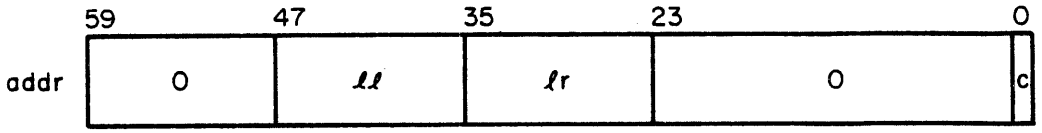


func Function code

addr Status return address

Function 01

This function returns the status of mass storage devices starting at location addr+1. The data returned consists of the following.

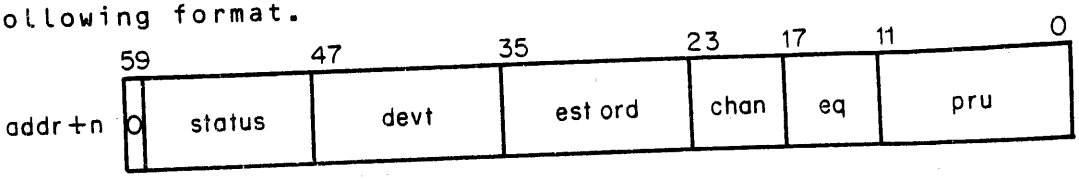


ll Number of words, excluding addr+0, to be used for return information; this value must be set nonzero by the caller

lr Number of status words returned

c Set to 1 when operation is complete; caller must set to zero

The mass storage device status is returned, 1 word per device, in the following format.



status Status of mass storage device as follows.

status	Description
000	Not available, off, not in use
040	Unloaded pack
120	NOS system routines
140	NOS system routines on pack
620	Contains permanent files
640	Pack with permanent files
700	NOS system and permanent files
740	NOS system and permanent files on pack

devt NOS/BE hardware mnemonic in display code (for mass storage devices).

Mnemonic	Description
AA	6603 Disk System
AB	6638 Disk System
AD	865 Drum System
AF	814 Disk System
AL	821 Disk System
AM	841 Disk System
AP	854 Disk System
AY	844-2X Disk System
AZ	844-4X Disk System
DE	ECS System
DP	DDP System

est ord EST ordinal of the mass storage equipment.

chan Primary I/O channel for the equipment.

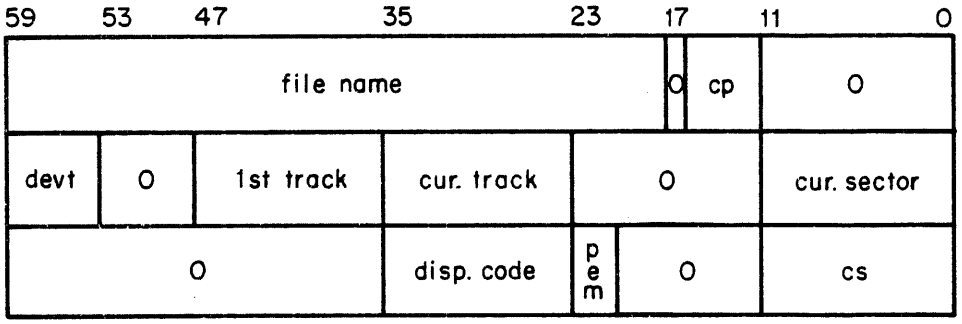
eq Equipment number.

PRU Number of PRUs/100B of space remaining on the device. A value of 7777B indicates at least 262,100 PRUs available.

Function 02

Returns to the calling program the FNT/EST entries of files requested whose names are set in every third location starting with addr+1. If the file exists, the file name is replaced by the FNT/FST of NOS mapped into the NOS/BE FNT/FST. If the file does not exist, the file name is zero.

The data returned consists of the header word in addr+0 in the same format as for function 01. The mapped FNT/FST for the named file is returned 3 words per entry as follows.



file name	Name of the file
cp	Control point assignment
devt	NOS/BE device type
1st track	First track
cur. track	Current track
cur. sector	Current sector
disp. code	Disposition code 40 Print 10 Punch
pem	Permanent file permissions 1111 No controls 0000 Execute only 0001 Read 1101 Extend 1011 Modify
cs	Code and status (same as byte 4 of the NOS FST entry)

The device type (devt) file, may contain those mass storage values listed for function 01 as well as the following items.

Device Type	Description
CR	Card reader
CP	Card punch
LP	Line printer
LQ	Line printer (512)
LR	Line printer (580)
TA	Time-sharing terminal

For magnetic tapes, the device types are as follows.

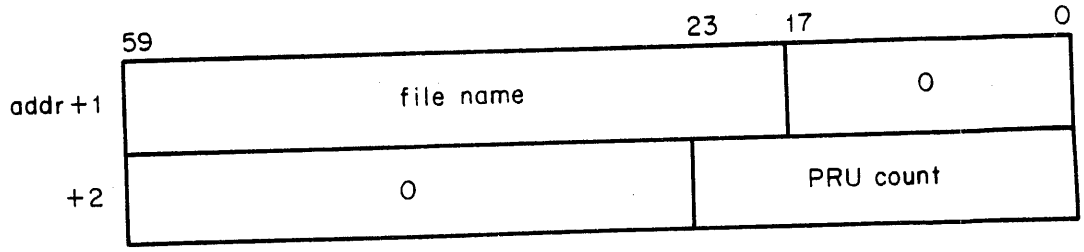
Device Type	Description
MT	Seven-track I, F formats
NT	Nine-track I, F formats
40nn	Seven-track SI, S, L formats
41nn	Nine-track SI, S, L formats

nn	Meaning
xxxx10	Density (always 800)
xx00xx	Unlabeled
xx01xx	Standard labels
10xxxx	SI format
10xxxx	S format
11xxxx	L format

Function 03

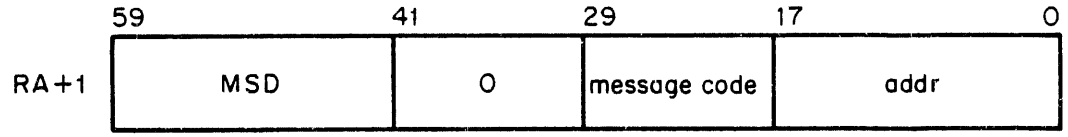
Returns to the calling program the number of PRUs of the files requested whose names are set in every second word starting at addr+1. If the file exists, the PRU count is returned in bits 23 through 0 of the second word. If the file does not exist, the second word is zero.

The data returned consists of the header word as described for function 01 and data words of the following format.



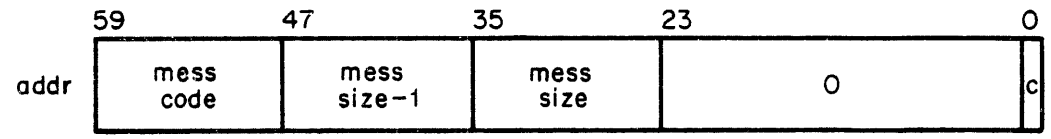
MSD REQUEST

The MSD request is processed by overlay 2SB. The RA+1 call for MSD has the following format.



message code	Ordinal of message to be returned
addr	Address to receive message

The format of addr is as follows.



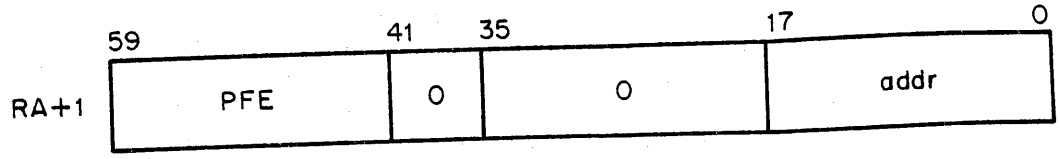
mess code	Message code
mess size	Length of message in CM words
c	Completion bit; set to 1 when function is completed

The message text is returned beginning at location addr+1.

PFE REQUEST

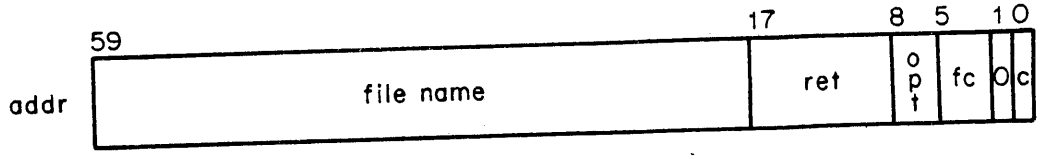
The PFE request is processed by overlay 2SD. The PFE request alters the requested file with an EOI recorded at the current position of the mass storage file.

The format of the RA+1 PFE call is as follows.



addr — Address of a parameter word

The parameter word has the following format.



file name — Name of the file

ret — If bit 6 is set in opt, a return code is available to the caller. The return codes are:

000	Function successful
003	Unknown file
025	File unavailable

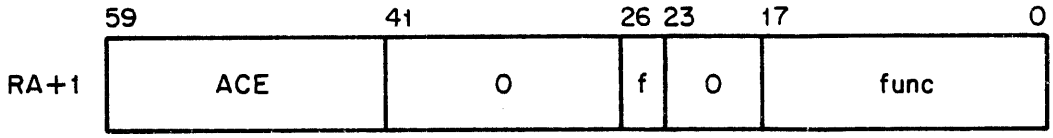
opt — Options (bit 6 is return code to user in ret)

fc — Function code for alter, bits 5-2=0111

c — Completion bit (set to 1 when function is completed)

ACE REQUEST

The ACE request is processed in overlay 2SE. ACE reads/backspaces the next/previous control statement into RA+70B through RA+77B with the option to place the control statement in the dayfile and/or to crack and store the control statement parameters in product set or operating system format into RA+2 through RA+53B. If a read function is issued and the pointer is at the end of the control statement record, an EOR status (bit 4 set in the function code is set and RA+70B through RA+77B are cleared. If a backspace function is issued and the pointer is at the beginning of the control statement record, the pointer is not changed and an EOR status is returned. When the function is complete, the completion bit (bit 0) is set and returned to the user. The format of the ACE call is as follows.



f Format:

x01	Crack parameter in operating system format
x10	Crack parameter in product set format
1xx	Issue control statement to dayfile

func CM word containing function (bits 11 through 0) to be performed.

The following functions are valid.

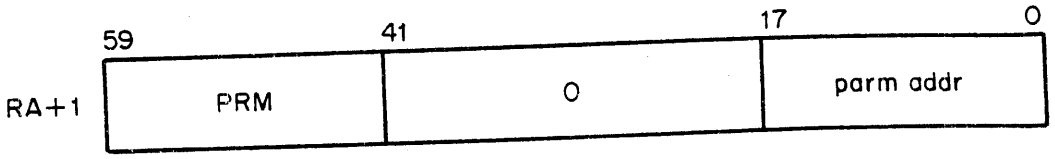
Function	Description
0010	Read next control statement and advance control statement pointer
0040	Backspace to previous control statement

ACE calls PP routine TCS after mapping the calling parameters for ACE into those recognizable by TCS.

PRM REQUEST

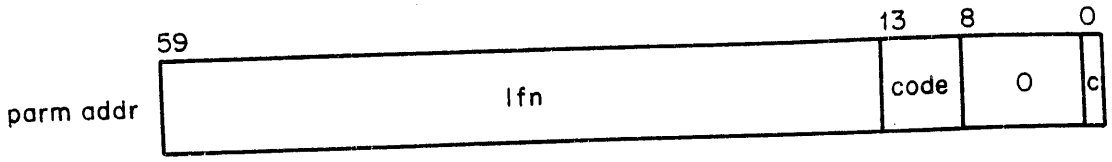
The PRM request is processed by overlay 2SF. PRM scans for an FNT entry whose name is contained in the call and, if found, maps the NOS file permission bits into the NOS/BE permission bits and returns them to the caller as a status. If the call address is out of range or the requested file does not exist, no diagnostic is issued and no status is returned to the caller.

The format of the PRM RA+1 call is as follows.



parm addr CM address which contains the file name to search for

The format of parm addr is as follows.



lfn logical file name

code A 5-bit code returned by PRM in bits 13 through 9. The rightmost four bits are the permission bits. The octal values for these bits are:

01	Read
02	Extend
04	Modify
10	Control

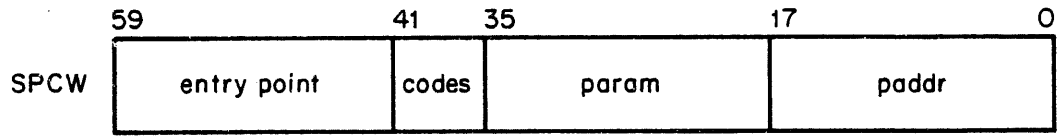
Bit 13 is zero if the file found is a permanent file. If equal to 1, the file, though found, is not a permanent file.

For a permanent file whose write lockout and/or execute bit(s) are set in the FNT, read only permission status will be returned to the caller. This is due to differences in permanent file structures between NOS and NOS/BE.

SPECIAL REQUEST PROCESSING

The CKP, REQ, DMP, DMD, DEP, and DED requests are processed in overlay 2SG, special request processing (SRP).

SRP consists of routines which set up a special processing word in the calling control point area (SPCW) for follow-up processing by 1AJ and a CPU program associated with the call. The format of SPCW is as follows.



entry point Name of entry point in CPU program

codes Control codes for use by 1AJ

Bit	Description
41	Request active (1AJ use only)
40	Clear RA+1 before reload if not set
39	Remainder of word is parameter list (not address of parameter list)
38	Do not restart CPU (1AJ use only)

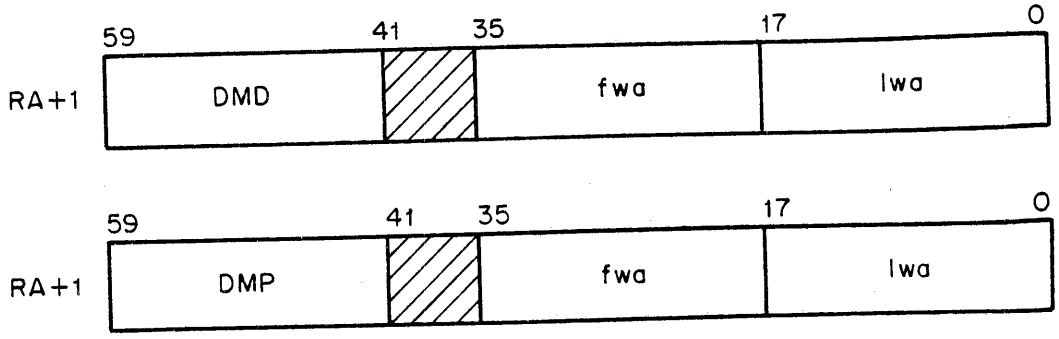
param Input parameter (bits 35-18) or status on output (bits 35-24)

paddr Parameter address passed in call

The SPCW word, with the exception of the codes field, is the same format as the RA+1 call.

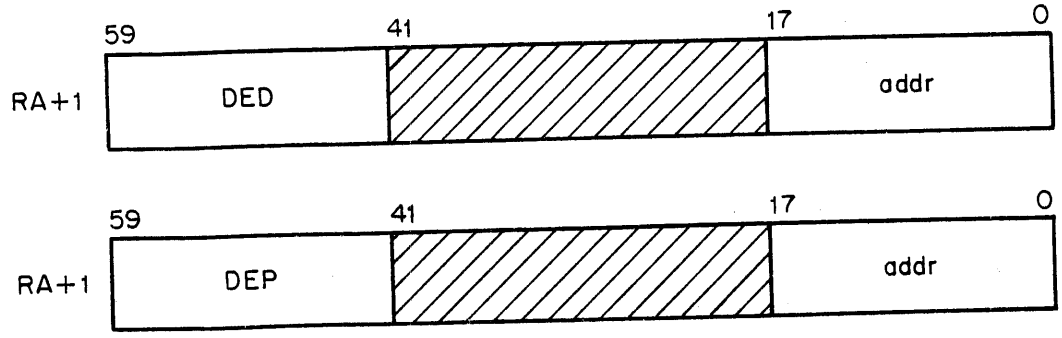
The CKP RA+1 request is discussed in section 25. The REQ RA+1 request is for compatibility purposes only. Equipment assignments under NOS should be made using the proper LFM functions.

The DMD and DMP RA+1 requests have the following formats.



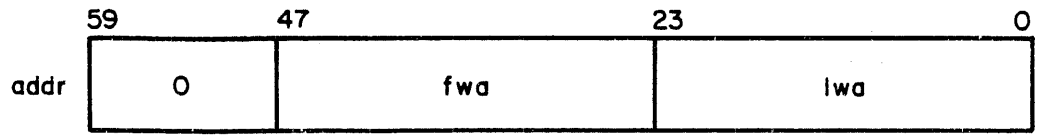
fwa First word address of CM dump
lwa Last word address plus 1 to be dumped

The DED and DEP RA+1 requests have the following formats.



addr Address of parameter word

Location addr is formatted as follows.



fwa First word address of ECS dump
lwa Last word address plus 1 to be dumped

ERROR PROCESSOR

Error conditions detected by SFP and its overlays are processed in overlay 2SH, error processor (ERP).

Routine 2SH is called with an error code in direct cell EI which indicates one of the following error conditions.

Error Code	Description	Issuing Routine
SCE	SFP call error	SFP
PCE	xxx not in PPLIB	SFP
IFR	Illegal function	2SA,2SE
IOC	Illegal origin code	--
PAE	Parameter error	SFP,2SA, 2SC,2SI
IAF	Illegal function code	2SD
SRE	Special request processing error	2SG
NET	Error text not found	2SI
NMS	Error text not on mass Storage	2SI
BET	Bad error text	2SI
IMN	Invalid message number	2SI
IOS	I/O sequence error	2SD

When issued to the dayfile the name of the monitor call being processed is prefixed to the message such that the message that appears is of the form

SFP/STS UNKNOWN DEVICE NAME/TYPE.

All errors except NET, NMS, BET, and IMN are considered fatal and abort the control point through the ABTM monitor function after issuing the message. The nonfatal errors drop the PP after issuing the message.

MONITOR CALL ERRORS

Since SFP is called after a PLD search has failed to yield a match on the desired PP program request, SFP contains the logic to process monitor call errors. If SFP does not find the desired routine in its function table, 2SH is called with error code PCE.

In the processing of PCE errors, a distinction is made for RA+1 requests and PP input register requests. If the request is an RA+1 monitor request from a user program, then the error flag is changed to PCET (program call error), SFP's input register is written to RA+1 and the PP is dropped. Routine 1AJ is called to process the error and upon detecting the PCET error type issues the diagnostic.

```
MONITOR CALL ERROR.
```

If no RA is present, the request was made through a PP input register. The diagnostic

```
xxx, NOT IN PP LIB.
     CALLED BY yyy.
```

is issued and the PP is dropped. In all other cases the diagnostic

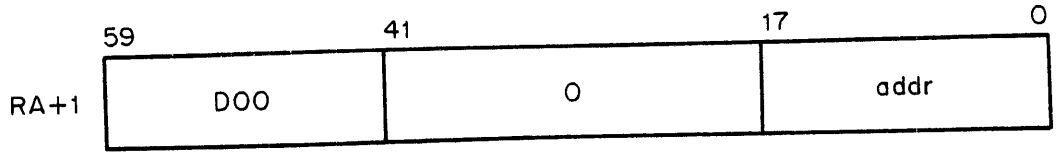
```
xxx NOT IN PP LIB.
```

is issued and the control point is aborted.

D00 REQUEST

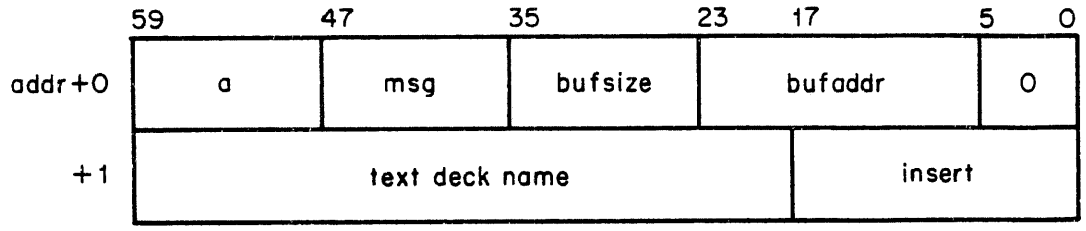
The D00 request is processed by overlay 2SI, extract error text. D00 extracts messages from specially-created system text decks to aid in analyzing error conditions resulting from product set execution. By using an error number and the proper system text deck, an error diagnostic is transmitted to a specific CM buffer and/or issued to the dayfile. All system decks to be used with D00 must reside on mass storage.

The RA+1 call for D00 has the following format.



addr Address of 2-word table

The format of location addr is as follows.

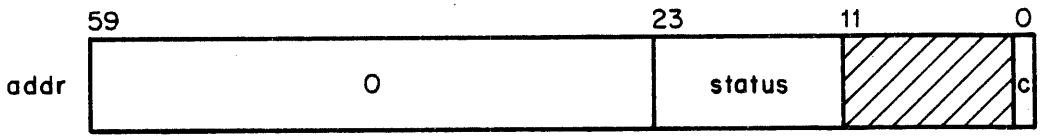


a destination

4000B If insertions to messages
2000B If dayfile message transfer
1000B If CM buffer message transfer

msg Number of the message to be transmitted
bufsize Size of CM buffer to receive message
bufaddr Address of CM buffer to receive message
deck Name of text deck containing message
insert Data to inserted into message

The response to the D00 request is received at the address specified in the call in the following format.



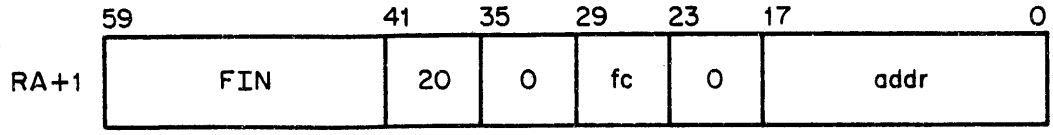
status Response status:

0 If transmitted to dayfile only
7777 If error
other Number of CM words written if CM buffer transmittal

c Completion bit; set to 1 when function is completed

FIN REQUEST

The FIN request is processed by overlay 2SI. Processing consists of translating the request into the appropriate CPM or LFM function request and reissuing it. The RA+1 format for the FIN request is as follows.



fc Function code:

01	FILINFO
02	ACCSF
03	ENCSF
04	GETJCI
05	SETJCI

addr Address for corresponding function

Any other value of fc results in the following message.

SFP - ILLEGAL FUNCTION CODE.

NOTE

This section has been transferred to the NAM IMS (publication number in preface).

This section describes KRONREF and common decks defined on the system old program library (OPL).

KRONREF

KRONREF is used by the programmer to locate a particular use of a symbol, type, error flag, common deck, or PP package.

KRONREF generates a cross-referenced listing of system symbols used by decks on a Modify program library. The names of programs on the OPL are listed for those decks that referenced the following.

- PP direct cell locations
- PP resident entry points
- Monitor functions
- Central memory pointers (in low core)
- Central memory locations (in low core)
- Control point area words
- Dayfile message options
- File types and mass storage constants
- Job origin types, queue types, and priorities
- Error flags referenced
- Miscellaneous NOSTEXT symbols
- Common deck calls
- PP packages called *
- Special entry points
- System macro request references

---

* Macro EXECUTE nme, = does not generate code to RJM to EXR, but is used exclusively to make a reference for KRONREF to use.

The KRONREF control statement format is as follows:

```
KRONREF(P=lfn1,L=lfn2,S=lfn3,G=lfn4)
```

P=lfn1 OPL input from file lfn1. If the P option is omitted or P alone is specified, file OPL is assumed.

L=lfn2 List output on line lfn2. If the L option is omitted or L alone is specified, file OUTPUT is assumed.

S=lfn3 System text from overlay lfn3. If the S option is omitted or S alone is specified, file NOSTEXT is assumed.

G=lfn4 System text from local file lfn4. If the G option is omitted, the system text is acquired as specified or defaulted by the S option. If G alone is specified, the local file TEXT is used. Use of the G option overrides any S specification.

COMMON DECKS

A system common deck is a COMPASS subroutine or group of macro or symbol definitions that have been tested, optimized, and designed to interface with the operating system. Common decks are used to increase efficiency in writing code, to ensure uniformity of code, and to decrease debugging time.

The NOS common decks are organized in the following classes.

- CPU common decks
- PP common decks
- Equivalences
- Table management
- Display routines
- TAF common decks
- Mass storage common decks

Each common deck is identified by the name COMxnnn, where x is the letter signifying the type of common deck as follows:

x	Significance
C	CPU common deck
I	Product installation configuration decks
P	PP common deck
S	Equivalences (subsystem symbols, constants, etc.)
T	Table definitions
M	Mass-storage common decks
D	Display routines
B	TAF data manager
K	Transaction subsystem

The 3-character designator nnn usually indicates the entry point used in the common deck.

COMMON DECK USAGE

Common decks of particular interest are COMCCMD, COMCMAC, and COMPMAC. These common decks contain macros generally used by the system programmer in system-origin jobs and PP routines. The most frequently used macros are defined in SYSTEXT through CPCOM. The macros defined on COMCMAC and COMCCMD are also defined in systems text PSSTEXT. Thus the majority of programs can be written without the need for calling a special common deck of macro definitions. In either case, whether the macros used are defined in a common deck or in a systems text, the program must call the common deck or systems text that contains the code to perform the operation required by the macro.

Most CPU system macros require that the common deck related to a function processor be available in the program; however, these common decks need not specifically be called by the user when writing relocatable routines since all CPU macros specify entries to common decks as external symbols. When these relocatable subroutines are loaded, the routines required (such as CIO= and LFM=) are satisfied from SYSLIB. The routines in SYSLIB are listed in this section.

If a program is not relocatable or if the desired common deck is not found on SYSLIB, then the common deck must be accessed from the system OPL. To use such a common deck, the programmer must insert the Modify directive *CALL in the text of the program or use the COMPASS pseudoinstruction XTEXT.

To obtain detailed documentation on a particular group of common decks, assemble the deck CALLxxx from the system OPL where xxx denotes the following.

xxx	Description
CPU	CP common decks
PPU	PP common decks
SYS	Equivalences
DIS	Display routines
TAB	Table management routines

The following are OPL common decks.

Common Deck	Description
COMCMAC	CPU system macros
COMCARG*	Process arguments
COMCARM	Multiple-word argument processor
COMCCDD*	Constant to decimal display code conversion
COMCCDM	CPU debugging macros
COMCCDP	CPU debugging package
COMCCFD*	Constant to F10.3 conversion
COMCCHD	Constant to hexadecimal display code conversion
COMCCIO*	I/O function processor
COMCCMD	Central program macro definitions
COMCCOD*	Constant to octal display code conversion
COMCCPA*	Convert positional arguments
COMCCPM	Control point manager processor
COMCCPT*	Copy prefix table
COMCCVI	User control limit formula
COMCCVL	Common validation interface processor
COMCDXB	Display code to binary conversion

*This common deck is also available on the COMPASS program library (PL).

Common Deck	Description
COMCECM	ECS interpretive mode macro definitions
COMCECS	ECS interpretive mode macro processor
COMCEDT	Edit date or time from packed format
COMCFCE	Format catalog entry for output
COMCFQO	Format queued file output
COMCHXB	Hexadecimal display code to binary conversion
COMCIQP	IQFT file processors
COMCLFM	Local file manager processor
COMCLOD	User call loader interface
COMCMTM*	Managed table macros
COMCMTP*	Managed table processors
COMCMVE*	Move block of data
COMCOVL	Overlay load processor
COMCPFM	Permanent file processor
COMCPFU	Permanent file utility function processor
COMCPOP	Pick out parameter
COMCQFM	Queue file manager processor
COMCRDC*	Read coded line, -C- format
COMCRDH*	Read coded line, -H- format
COMCRDO*	Read one word
COMCRDS*	Read coded line to string buffer
COMCRDW*	Read words to working buffer
COMCRSP	Remove secure parameter from control statement
COMCRTN	Read terminal network description
COMCSFM	System file manager processor
COMCSFN*	Space fill right-justified zeros

*This common deck is also available on the COMPASS PL.

Common Deck	Description
COMCSNM	Set name in message
COMCSRT*	Set record type
COMCSSN	Skip sequence number
COMCSST*	Shell sort table
COMCSTF*	Set terminal file
COMCSYS*	Process system request
COMCUPC*	Unpack control statement
COMCUSB	Unpack data block to string buffer
COMCVFE	Validate FNT/FST entry
COMCWOD*	Convert word to octal display code
COMCWTC*	Write coded line, -C- format
COMCWTH*	Write coded line, -H- format
COMCWTO*	Write one word
COMCWTS*	Write coded line from string buffer
COMCWTW*	Write words from working buffer
COMCZAP	Z argument processor
COMCZTB*	Convert zeros to blanks in a word

The following are PP common decks.

Common Deck	Description
COMPMAC	PP system macros
COMPACS	Assemble character string
COMPANS	Assemble numeric string
COMPCDI	Clear local MST utility interlock
COMPCEA	Convert FCS address
COMPCFP	Clear format pending
COMPCHI	Redefine I/O instructions

*This common deck is also available on the COMPASS PL.

Common Deck	Description
COMPCHL	Redefine I/O instructions
COMPCIB	Check input buffer
COMPCKP	Set checkpoint bit in EST entry
COMPCLD	Search central library directory
COMPCLX	Clear exchange package
COMPCMA	Central memory available on recovery
COMPCMX	Computer maximum field length
COMPCOB	Check output buffer
COMPCRA	Convert random address
COMPCRS	Check recall status
COMPCTI	Clear track interlock
COMPCUA	Check user access
COMPCUN	Compare user numbers
COMPCUT	Clear permanent file utility interlock
COMPCVI	Convert validation indexes
COMPC2D	Convert two octal digits to display code
COMPDMS	Determine memory size
COMPDTS	Determine track interlock status
COMPDV5	Divide by five
COMPECX	Compute ECS maximum field length
COMPFAT	Search for fast attach file
COMPGBN	Generate banner name
COMPGJN	Generate job name
COMPGTN	Generate terminal number
COMPIFR	Set/clear flag register interlock
COMPIRA	Initialize random access processors
COMPMRQ	Monitor request

Common Deck	Description
COMPMSD	Mass storage processor for 3553-1
COMPRBB	Read binary buffer
COMPRCB	Read coded buffer
COMPRCS	Read control statement
COMPREI	Request ECS increase
COMPRJC	Read job control word
COMPRLS	Release storage
COMPRNS	Read next sector
COMPRSI	Request storage increase
COMPRSS	Read system sector
COMPSAF	Search for assigned file
COMPSCA	Set catalog address
COMPSCE	Status/control register error processor
COMPSDI	Set local MST utility interlock
COMPSDN	Search for device number
COMPSEI	Search for end-of-information
COMPSES	Set error status in local MST (STLL)
COMPSFB	Set file busy
COMPSFE	Set family equipment
COMPSFI	Set FNT interlock
COMPSFN	Space fill name
COMPSNT	Set next track
COMPSPA	Set pot address
COMPSRA	Set random address
COMPSSE	System sector error processor
COMPSTA	Set terminal table address
COMPSTI	Set track interlock

Common Deck	Description
COMPSUT	Set permanent file utility interlock
COMPTGB	Set/clear global MST flag (ACGL)
COMPTLB	Set/clear local MST flag (STLL)
COMPUPP	Update pot pointer
COMPUPS	Unpack statement
COMPUSS	Update system sector for disposable files
COMPVFC	Verify forms code
COMPVFN	Verify file name
COMPVMS	Validate mass storage ordinal
COMPWBB	Write binary buffer
COMPWCB	Write coded buffer
COMPWEI	Write EOI sector
COMPWSS	Write system sector
COMPWVE	Write and verify with EOI sectors
COMPREL	Location free overlay macros
COMPRLI	Relocatable overlay macros
COMPCHL	Redefine I/O instructions
COMP3XD	3000 equipment driver subroutines

The following are display common decks.

Common Deck	Description
COMDDIS	Display subroutines
COMDDSP	Display program routines
COMDSYS	Display system status and associated routines
COMDTFN	Table of monitor functions for display

The following are mass storage common decks.

Common Deck	Description
COMMMSD	Universal mass storage driver
COMMMSE	Mass storage error processor

The following common decks contain subsystem equivalences, symbol definitions, and constants.

Common Deck	Description
COMSACC	User file equivalences
COMSBIO	BATCHIO equivalences
COMSCIO	CIO/driver equivalences
COMSCPS	CPUMTR subfunction codes
COMSDSL	Deadstart load parameters
COMSESS	Engineering services support definitions
COMSEVT	Event descriptor formats
COMSEXP	EI/200 tables and constants
COMSIOQ	Dayfile/queue protect equivalences
COMSJCE	Job control equivalences
COMSJIO	Job input/output equivalences
COMSJRO	Job rollout equivalences
COMSLDR	CPU program loading equivalences
COMSLSD	Label sector definition
COMSMMF	Multimainframe equivalences
COMSMRT	Machine recovery equivalences
COMSMSI	MST/PP equivalences
COMSMSP	Mass storage processing equivalences
COMSMST	MST flag/interlock definitions

Common Deck	Description
COMSMTR	MTP/CPUMTR equivalences
COMSMTX	Magnetic tape executive equivalences
COMSNCD	Network communications definitions
COMSNET	Terminal network equivalences
COMSPFM	Permanent file equivalences
COMSPFS	Permanent file supervisor equivalences
COMSPFU	Permanent file utilities equivalences
COMSPRD	Priority definitions
COMSPRO	Project profile file structure
COMSQFS	Queued file equivalences
COMSREM	IAF/TELEX system parameters
COMSRSX	Resource executive equivalences
COMSSCP	System control point equivalences
COMSSCR	S/C register equivalences
COMSSFS	SFS equivalence and table definitions
COMSSRU	Define SRU parameters
COMSSSE	System sector equivalences
COMSSSJ	Special system job parameters
COMSTCM	TELEX communications micros
COMSTDR	Terminal driver equivalences
COMSTRX	Transaction subsystem equivalences
COMSWEI	EOI sector definitions
COMSZOL	Zero-level overlay lengths

The following common decks contain tables used by the system.

Common Deck	Description
COMTBCD	Display code to BCD
COMTDPC	BCD to display code
COMTDP6	Display code to 026 punch
COMTDP9	Display code to binary 029 punch

Common Deck	Description
COMTNAP	Define application access bits
COMTVXD	ASCII/display conversion table
COMT6DP	026 punch to display code
COMT9DP	029 punch to display code

SYSLIB

The following common decks, unless otherwise noted, are assembled from the COMPASS PL and are available in relocatable form in SYSLIB.

Common Deck	Description
COMCCIO	I/O function processor
COMCCPM*	Control point manager processor
COMCECS*	ECS interpretive mode macro processor
COMCLFM*	Local file manager processor
COMCMVE	Move block of data
COMCOVL*	Overlay load processor
COMCPFM*	Permanent file
COMCRDC	Read coded line, -C- format
COMCRDH	Read coded line, -H- format
COMCRDO	Read one word
COMCRDS	Read coded line to string buffer
COMCRDW	Read words to working buffer
COMCSYS	Process system request
COMCWTC	Write coded line, -C- format
COMCWTH	Write coded line, -H- format
COMCWTO	Write one word
COMCWTS	Write coded line from string buffer
COMCWTW	Write words from working buffer

*Assembled from the system OPL.

INTRODUCTION

The Export/Import (E/I) subsystem controls communication between NOS and remote batch terminals operating in the mode 4A 200 User Terminal protocol. The Export/Import subsystem is also referred to as E/I 200. E/I 200 supports dial-up or hard-wired terminals at line speeds of 2000, 2400, or 4800 bits per second with the BCD character set.

E/I 200 PROGRAMS

The E/I 200 subsystem consists of the following routines.

- E200CP, a CPU program
- 1LS, a transient PP routine
- 1ED, a dedicated PP routine
- XSP, a transient PP routine

E200CP is the CPU program that handles reformatting of data to and from the remote terminals. Its field length is also used for all communication tables and FETs for the subsystem. Common deck COMSEXP is used to establish the constants, pointers, and communication table areas. Table 33-1 illustrates the general layout of these areas.

A local resident peripheral library (RPL) for the 1LS overlays is contained in the E200CP field length. The FETs and buffers are kept in the upper portion of the FL so that the FL may expand and contract as the need arises.

Routine 1LS is a transient PP program that processes terminal commands, assigns files, performs functions for 1ED, and functions as the executive routine for the subsystem.

Routine 1ED is a dedicated PP program that controls communications between the system and the remote batch terminals. Routine 1ED must service the 6671 multiplexer at a rate fast enough to ensure that no data loss occurs for the highest speed line connected. The program does data conversion of the input/output data to and from the display code used internal to the rest of the system. Routine 1ED must get E200CP out of autorecall when there are buffers to be filled or emptied.

XSP is a transient PP program called by 1LS to perform certain time-consuming tasks.

The preceding routines are discussed in detail later in this section.

E/I 200 OVERVIEW

Figure 33-1 details E/I 200 interaction between E200CP, 1LS, 1ED, XSP, and the remote batch terminals it services. Figure 33-2 illustrates the sequence of operations and data flow for a job using the subsystem. E/I 200 can service up to sixteen 2400 bps terminals. For each terminal E/I 200 logically maintains the table shown in figure 33-3.

TABLE 33-1. E/I CM LAYOUT

Location	Description
RA+0 through RA+22	Pointer table. The following locations contain the indicated pointers. Location — Pointer 2 — FETs and Buffers 3 — TFS 4 — MSGB 5 — LINF 6 — CPIK 7 — DPJT 12 — JST 13 — FAMT 14 — UNJC
DRCL	E200CP autorecall word set by 1ED when activity needed.
TFS	Function/status table (2 words per port).
MSGB	Message buffer area (4 words per port).
LINF	Login table (2 words per port).
CPIK	CPU interlock table (1 word per port).
DPJT	Drop job table (1 word per port).
JST	Job statistics table (1 word per port).
FAMT	Family name table (1 word per port).
QAPB	QAC parameter block.
--	E200CP code.
FALOC	Beginning of dynamic storage.
--	Allocated FETs and buffers.

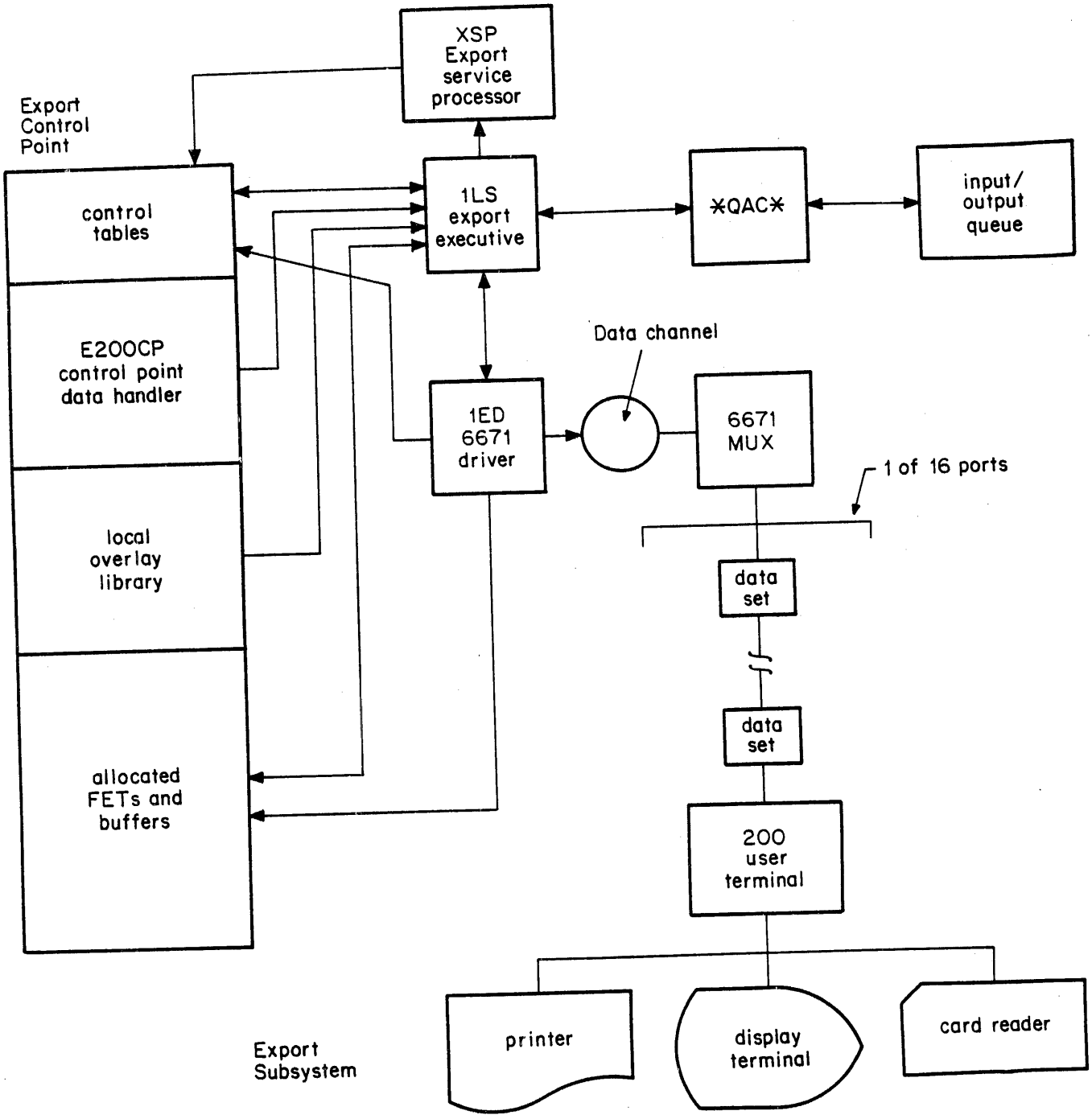
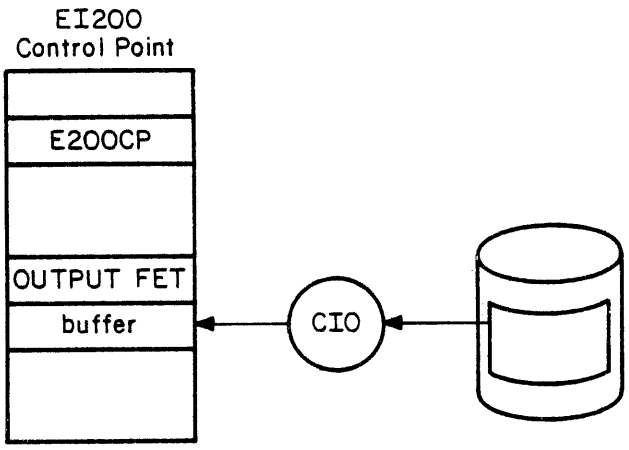
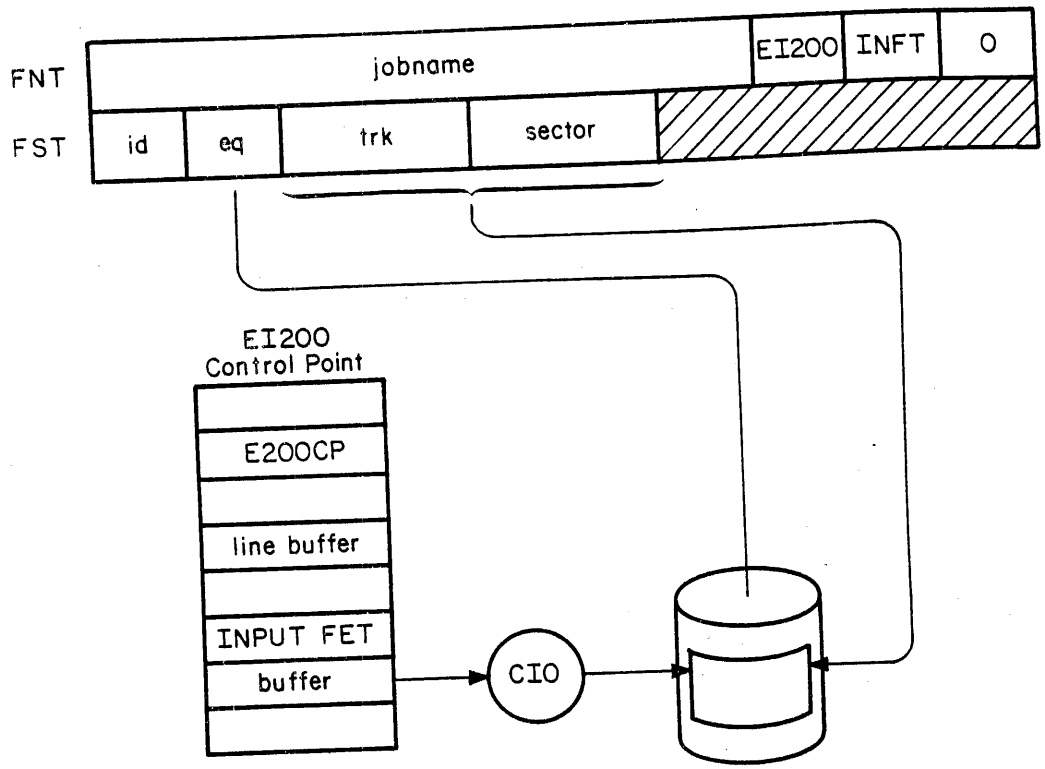


Figure 33-1. E/I 200 Interaction

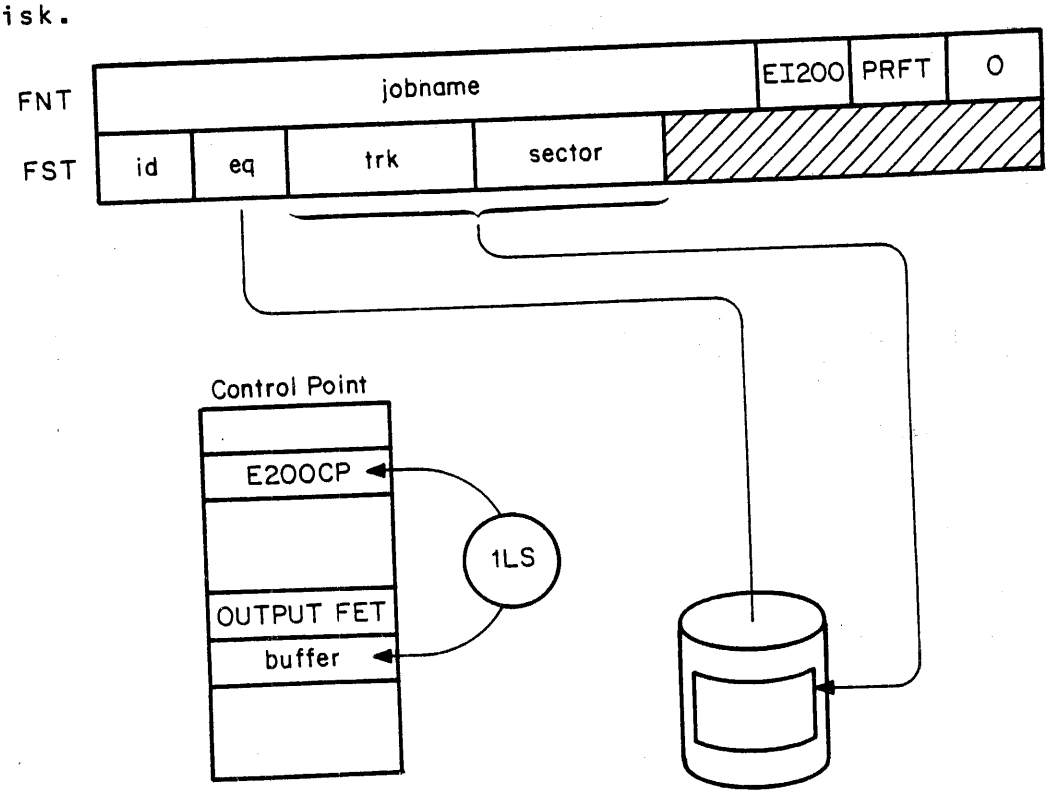


E200CP reads the output file via CIO into the output FET buffer.

Figure 33-2. E/I 200 Operation

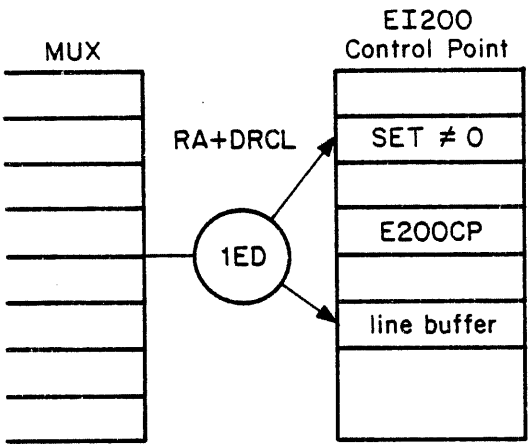


E200CP calls CIO to write the data from the input FET buffer to the disk.

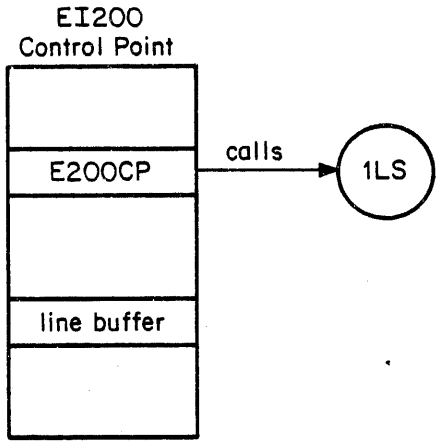


1LS finds an output queue entry and calls OBP to create a banner page in the output FET buffer and informs E200CP.

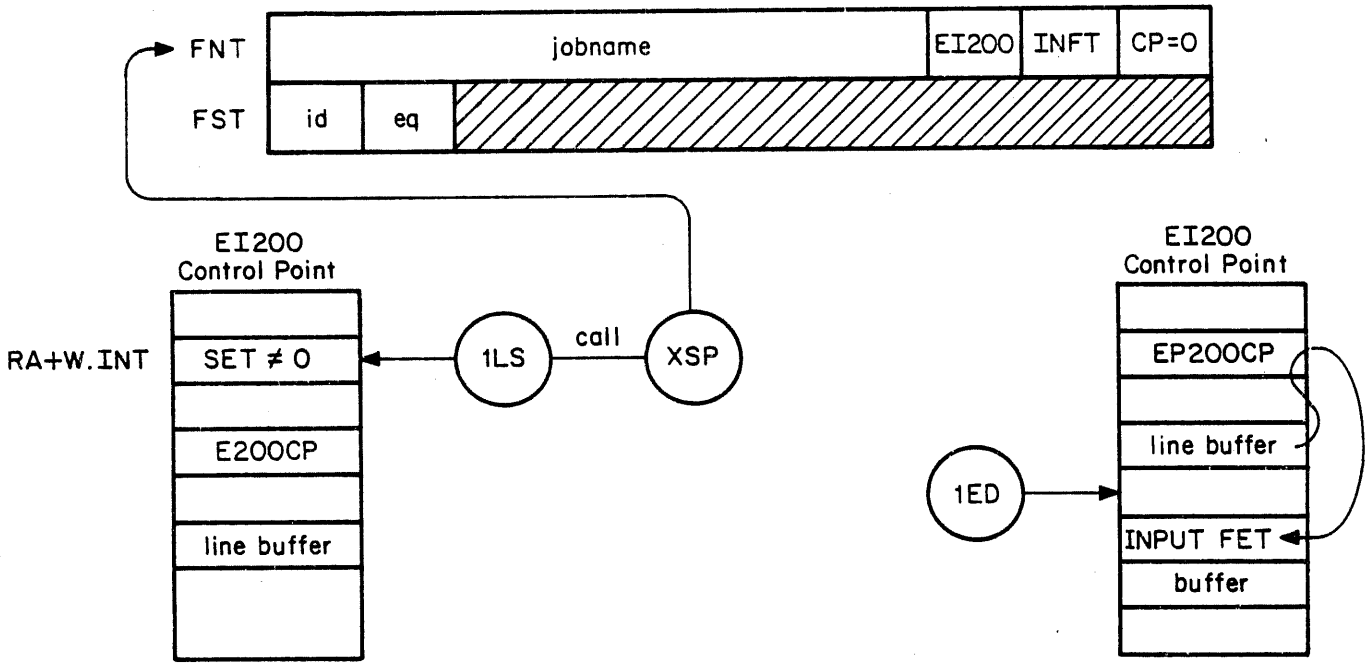
Figure 33-2. E/I 200 Operation (Continued)



1ED reads from multiplexer to line buffer and sets RA+DRCL words to nonzero. This takes E200CP out of autorecall.



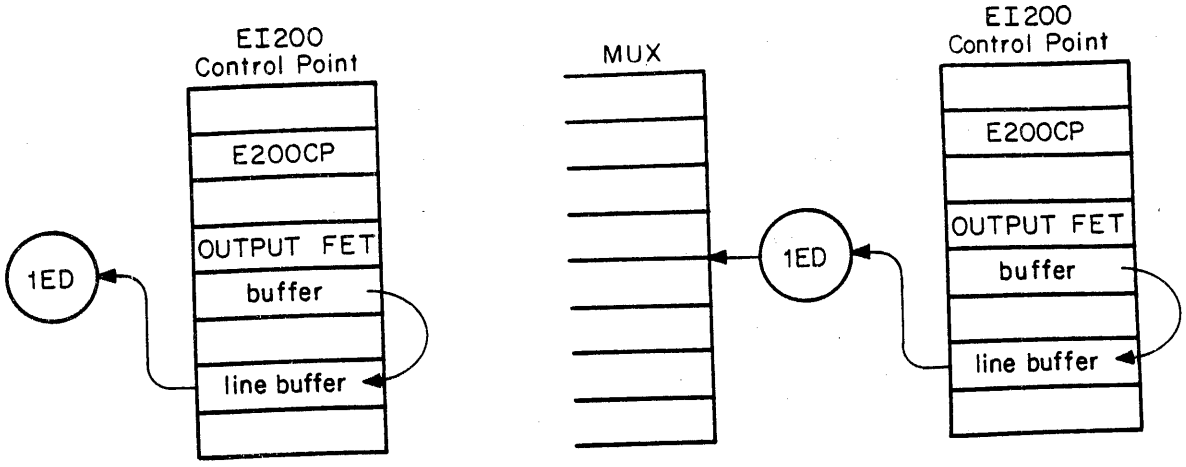
E200CP calls 1LS and goes into autorecall.



1LS sets RA+W.INT to nonzero, which takes E200CP out of autorecall. 1LS calls XSP to create an FNT/FST input queue entry for the job in the line buffer, using OBF and OVJ to crack the job card.

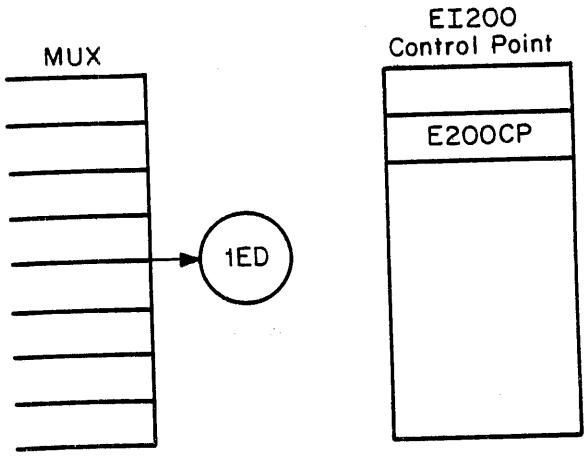
E200CP reformats the line buffer data as 1ED passes it and moves the data to the input FET buffer.

Figure 33-2. E/I 200 Operation (Continued)



E200CP formats the output FET buffer for the remote printer and informs 1ED.

1ED sends the line buffer data, one line at a time, to the remote printer.



E200CP goes into autorecall and 1ED continues to poll the multiplexer.

Figure 33-2. E/I 200 Operation (Continued)

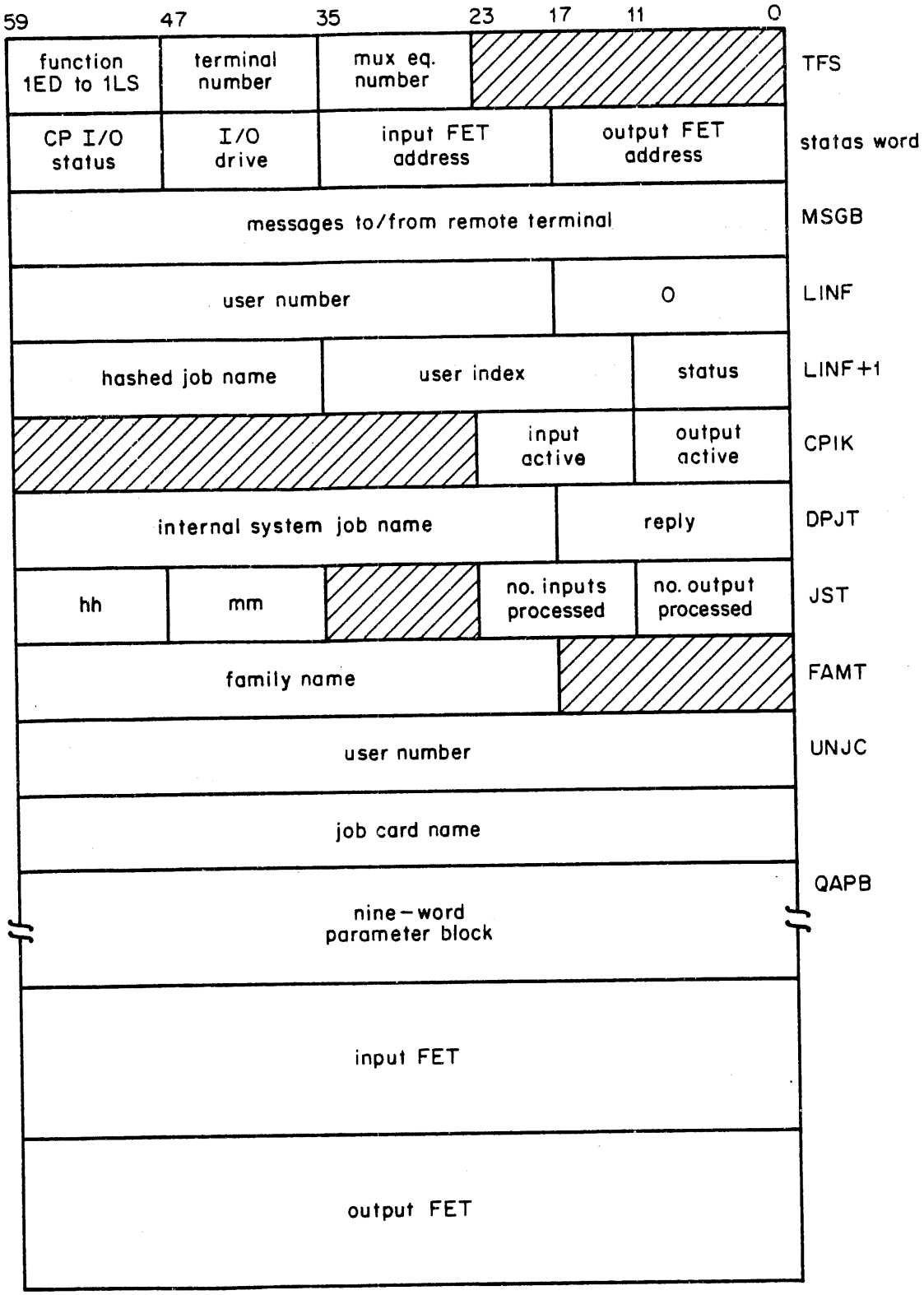


Figure 33-3. Port Table Layout

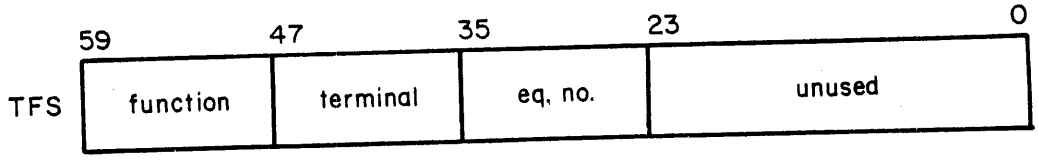
EXPORT/IMPORT COMMUNICATION AREAS

The following paragraphs detail the communication areas shown in figure 33-3. These tables are in the E200CP field length and are used as communication areas for all parts of the subsystem.

Area	Description
TFS	Function/status table
MSGB	Message buffer
LINF	Login information table
CPIK	CPU interlock table
DPJT	Drop job table
PWLT	Password table
FAMT	Family name table
--	Export/Import FETs

FUNCTION/STATUS TABLE

The function/status table (TFS) is used by 1ED to issue function requests to 1LS. Its format is as follows.



function — Functions are set by I/O driver 1ED to communicate with 1LS; defined in the 1LS field length and in COMSEXP

function	Description
00	Null function
02	Message from terminal
04	Print block complete
06	Special end read
10	Write message complete

function	Description
12	Multiplexer not available
14	Multiplexer not operational
16	Initialization complete
20	Terminal connected
22	Printer not ready
24	Message read error
26	Terminal disconnected
30	Operator interrupt
32	Read E3, no EOF*
34	Read E3, with EOF*
36	Read E2, no EOF*
40	Read E2, with EOF*

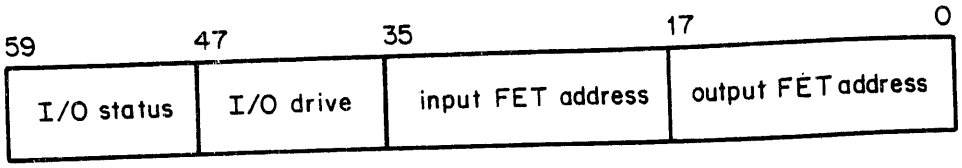
terminal — Site address of logged-in terminal

eq no — Equipment number of multiplexer assigned; this field used in entry 0 only

---

* E1, E2, and E3 are hardware functions set by both the card reader and printer. They are specified in the appropriate E/I 200 hardware manual.

The TFS status word has the following format.



I/O status CPU I/O status from 1LS:

status	Description
0001	Run CPU (output, coded mode)
0002	Run CPU (input, coded mode)
0004	Print line limit exceeded when set
0010	Initialization in progress
0020	XSP initialization required flag (output)
0040	Return sequence number
0100	Output file active
0200	Input file active
0400	Not assigned
1000	Output file suspended
2000	Read, wait for operator GO
4000	If 0, read E3, if 1, read E2 (on previous read)*

---

* E1, E2, and E3 are hardware functions set by both the card reader and printer. They are specified in the appropriate E/I 200 hardware manual.

I/O drive Driver status to 1ED from 1LS:

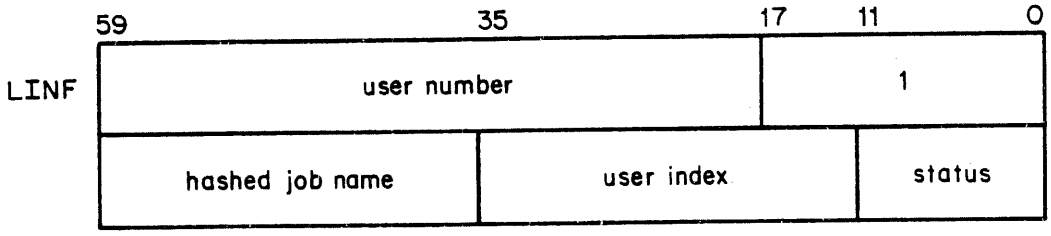
status	Description
0001	Terminal on-line
0002	Terminal logged-in
0004	Interrupt during print transmission
0010	Interrupt during read transmission
0020	Not assigned
0040	Not assigned
0100	Execute print control program
0200	Execute read control program
0400	Execute write message to terminal screen
1000	Not assigned
2000	Not assigned
4000	Not assigned

MESSAGE BUFFER

Each message buffer (MSGB) is four CM words in length. The messages to and from the remote terminals are placed in the appropriate message buffer with a 0000 termination byte.

LOGIN INFORMATION TABLE

The login information table (LINF) contains two words per terminal. This table is used by XSP to respond to 1LS. Its format is as follows.



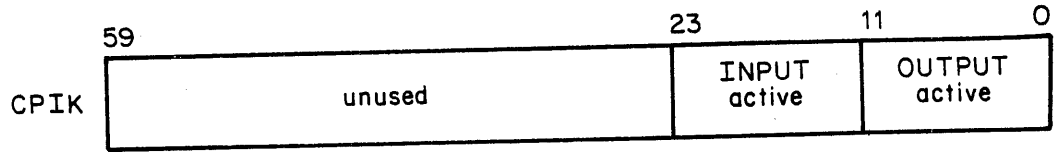
user index User index; if zero, indicates illegal user number

status Status of terminal:

status	Description
0	Login active
1	Login complete
2	Request PP again (system busy)
3	Duplicate user number

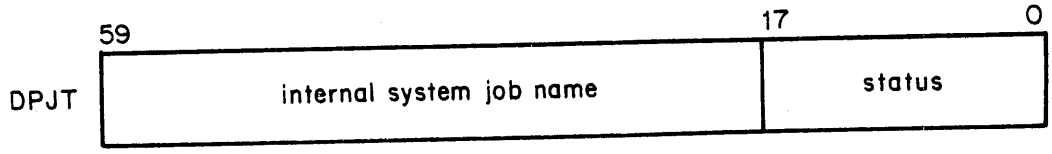
CPU INTERLOCK TABLE

The CPU interlock table (CPIK) is used by E200CP to control I/O activity. E200CP sets the proper byte to nonzero when there is activity on a file and zeros the proper byte when it detects the CPU drive bit off for the appropriate channel (INPUT or OUTPUT). Its format is as follows.



DROP JOB TABLE

The drop job table (DPJT) contains the password at login. At other times it contains status responses in the following format.



status Response status:

status	Description
0000	Entry available
0001	QAC called and active
0002	QAPB in use
0003	PP not available

PASSWORD TABLE

The password table (PWLT) uses the same location as DPJT. At login time, this table is used for the user password instead of drop job.

FAMILY NAME TABLE

The family name table (FAMT) at login time is used for the user's family name.

EXPORT/IMPORT FETs

E/I 200 creates a FET for each terminal that is logged in. The formats are shown in figure 33-4.

INPUT FET

FET+0	internal system name				code/status
1					FIRST
2					IN
3					OUT
4	FNT address	0			LIMIT
5	full/empty driver flag	job card processing in progress	address of line following EOR		address of line following EOF
6	job sequence number		0		pointer to next allocated FET
7	job priority	job time limit	job FL	0	card count

OUTPUT FET

FET+0	internal system name				code/status
1				01	FIRST
2	0				IN
3	0				OUT
4	FNT address	dayfile first track	dayfile first sector	0	LIMIT
5	full/empty driver flag	0			
6	job sequence number		0		pointer to next allocated FET
7	print line count				

Figure 33-4. Export/Import FETs

PROGRAM E200CP

The central processor portion of E/I 200 is used for the processing of data to and from the remote site.

Data being received from the remote site card reader is placed in the line buffer allocated to the active terminal by the I/O driver program. The I/O driver, 1ED, strips off the protocol header and trailer. The data is converted to display code and written, one card image at a time, into the line buffer. When the I/O driver senses an end of message code, the CM line buffer is marked full so that E200CP processes that data at the next opportunity. Trailing blank suppression and detection of end-of-record and end-of-file is accomplished by E200CP.

Transmission of data to the system allocatable mass storage device is also requested by the E200CP.

The buffer space for an output file is allocated by the executive program 1LS. The banner page data is placed in the circular buffer by the executive program 1LS. All subsequent I/O requests are issued by the E200CP program. Data from the circular buffer is compressed according to the 200 User Terminal specification and placed into the line buffer for transmission to the terminal. As much data as possible is placed in the line buffer on each cycle. A full line buffer is not always possible to generate because the print line cannot be split between messages.

The control for the CPU program is a switched circular scan of the terminal control table. Switching is performed by the executive via the status word in the function/status table. Control interlock is through the CPIK table within CM. Every complete scan attempts to complete an entire operation on every active terminal. When an entire scan is completed. E200CP goes into autorecall.

The following common decks are called by E200CP.

Common Deck	Description
COMCCIO	I/O function processor
COMCRDC	Read coded line, C format
COMCWTH	Write coded line, H format
COMCSYS	Process system request
COMCRDW	Read words to working buffer
COMCWTW	Write words from working buffer
COMCMAC	CPU system macros
COMCCPM	Control point manager processor
COMSEXP	E/I constant definitions

Figure 33-5 is a flowchart of the main scanner control portion of E200CP. The following paragraphs discuss Export/Import processors INP and OUT.

INP - INPUT DATA PROCESSOR

The following functions are performed by this program.

- Moves data from the line buffer into the file cirular buffer, removing trailing blanks in the process.
- Writes data to the system mass storage device using CIO and standard I/O techniques.
- Senses and processes end-of-records. An EOR is indicated by a block of eight words in the line buffer containing the character K.EOR. This 12-bit value (30B) is defined in COMSEXP in byte zero of block word zero.
- Issues a CIO request to write EOR from the buffer. If the first word of the next block does not contain EOM (zero, end of message), sets the beginning address of the next block in FET+5, bits 35 through 18 and continues processing when the FET becomes free.
- Senses and processes end-of-file. An EOF is indicated by a block of eight words in the line buffer containing the character K.EOF (27B) in byte zero of block word zero. If the word following this eight word block does not contain an EOM code (zero, end of message), records the beginning address of the next unprocessed data block in FET+5 bits 17 through 0, sets byte one nonzero, and does not alter byte zero (full/empty control). The program waits for FET+5, byte one to be set back to zero by 1LS when it has processed the input file. Processing of data then continues at the block address stored in FET+5, bits 17 through 0.
- Senses and processes end-of-message. An EOM is indicated by byte zero of a block (or special last word) containing the character K.EOM (0) in byte zero. The full/empty status (byte zero) of FET+5 is set empty and normal data processing continues.

  These special values are as follows.

Character	Value
K.EOR	0030B
K.EOF	0027B
K.EOM	0000B
K.EOI	0055B

They are specified in the COMSEXP common deck.

Data from the circular buffer is placed into the line buffer by this phase of the E200CP program.

Strings of blanks greater than two characters in length and up to MAXB characters are replaced by a two-character compression set. Strings greater than the maximum length are processed as one or more strings of maximum length and a remaining short string if necessary. End-of-line codes are placed on every line sent to the remote printer. Only complete lines are placed in the line buffer and lines of more than 136 characters are treated as more than one line, but some characters may be lost.

An attempt is always made to fill the line buffer with the maximum number of characters allowed. A restriction of the terminal hardware forces a full line to be transmitted before an end-of-message. This means that not all transmissions are maximum length.

The 200 User Terminal has three buffers; the screen, card reader, and printer. The screen buffer is used for transmission to the multiplexer; consequently card images are transferred from the card reader buffer to the screen buffer for transmission to the multiplexer. Similarly, output is transmitted by the multiplexer to the screen buffer, and is then transferred to the printer buffer for printing.

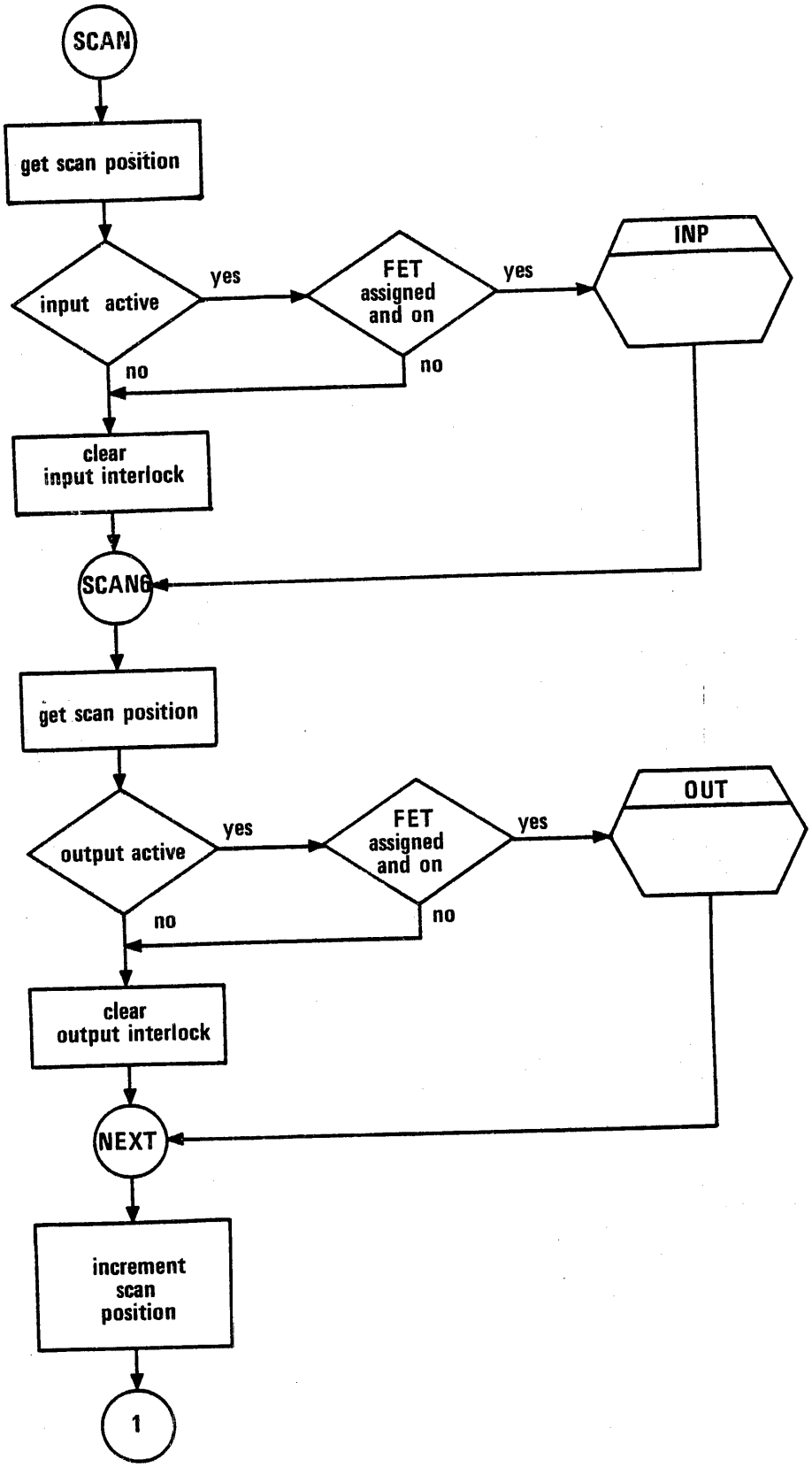


Figure 33-5. E200CP Control Scanner

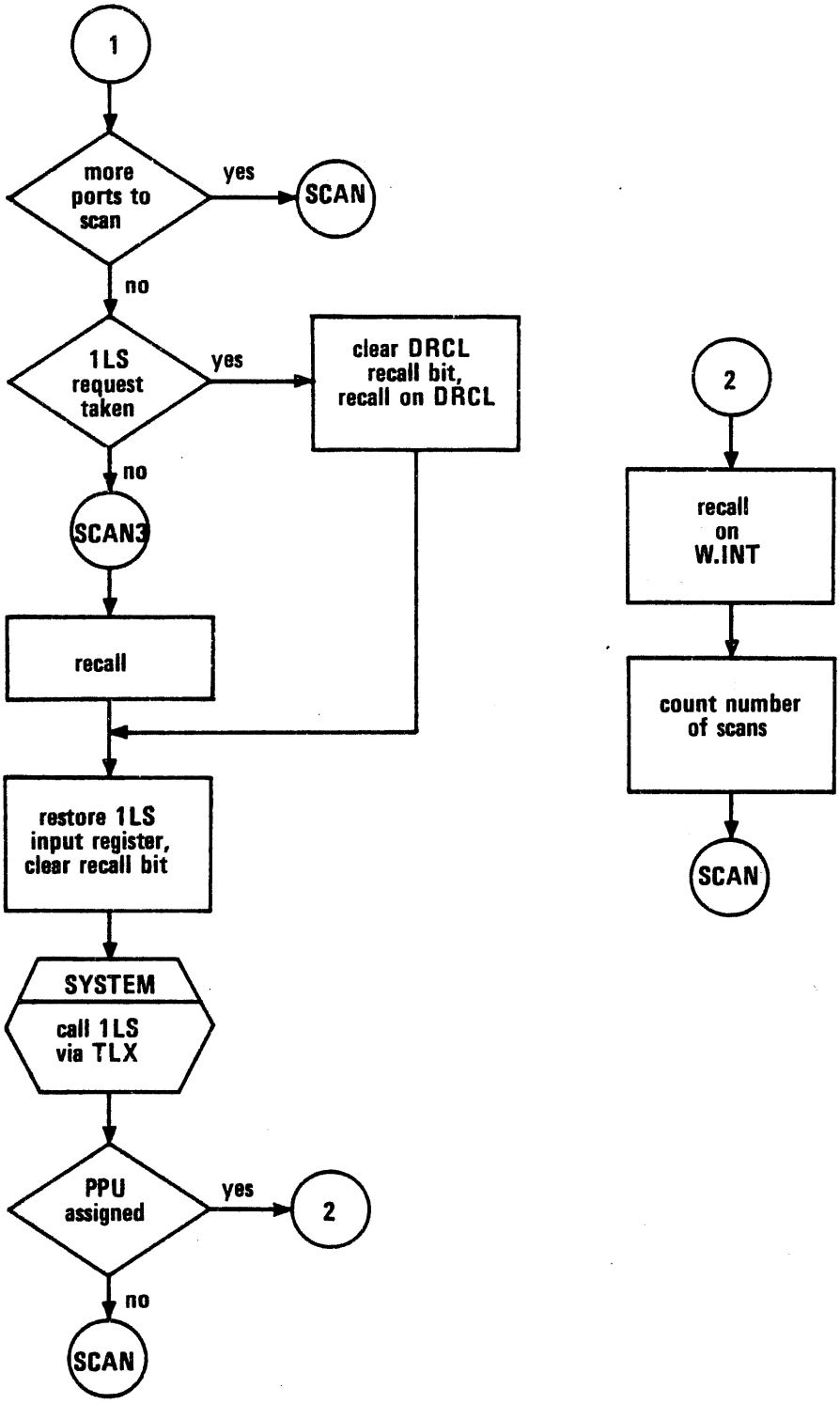


Figure 33-5. E200CP Control Scanner (Continued)

E200CP waits in autorecall until 1ED sets RA+DRCL to 1, indicating some input was received from some remote terminal. E200CP calls 1LS to a PP and goes into autorecall until 1LS is ready for E200CP to begin processing the input or output.

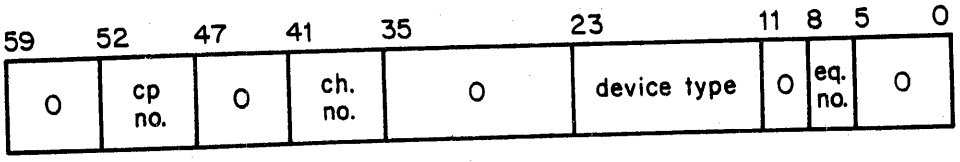
Routine 1LS may load any of the following overlays at anytime, depending on the action required.

- Initial load. Routine 1LS (loaded by system) loads executive subroutines at 7000B. These two segments are expected to be resident at all times (1LS and OVRS in memory.
- Function processing. The function processor segment is loaded if any outstanding functions from the driver are found (1LS, OVRS, and OVFP in memory).
- Job file processing. The enter queue segment is loaded if function processing found any outstanding input activity (1LS, OVRS, OVJF).
- Search for and initiate output. The FNT search segment is loaded if the time interval for FNT search is satisfied (1LS, OVRS, OVFA, and possibly OBP in memory).
- Storage management. The storage manager is loaded if the time interval for buffer check is satisfied (1LS, OVRS, and OVCS in memory).

Any number of the preceding actions could occur during an executive pass.

The EST entry is expected to be 3000 type equipment (ST). If a different equipment type code is desired, the definition of MUXDT in the I/O driver must be changed.

The EST entry is located by the multiplexer I/O driver program. The search finds the first entry of the proper type which is not set off or assigned to another control point. The EST format is as follows.



All of the normally used executive overlays are stored in central memory within the field length of the E200CP program during initialization. This technique increases the load speed of the PP executive without using large amounts of CMR space if

E/I 200 is not loaded. For this reason, the programs and overlays associated with E/I 200 should be disk resident. The only part of E/I 200 that must be CM-resident is the short executive main program, 1LS.

The local RPL map is identical in format with the system RPL. Starting at the address in pointer word P.RPL, a zero word ends the library.

The routines in the library are:

- 9IA overlay OVFP. Function processor, when 1ED communicates with 1LS via the TFS table.
- 9IB overlay OVFA. Searches the FNT for files to be printed at the remote sites. If any such files are found, a buffer is allocated and the header information is placed in the buffer for the initial print operations. Subsequent data handling is performed by the central processor program associated with this system. It calls overlay OBP to generate the banner page.
- 9IC overlay OVJF. Job file processor called by executive main control to release completed job files to input queue and/or to initialize new job file.
- 9ID overlay OVCS. Central memory manager executes every few seconds in an attempt to reduce the amount of storage used by Export/Import central memory.
- 9IE overlay OVIN. The first time 1LS is called by E200CP, this overlay initializes all of Export/Import.
- 9IF overlay OVAB. All error modes, operator STOP, and error messages are processed by the abort E/I overlay.
- 9IG overlay OVRO. Initialize resident library programs in control point FL area. Programs are stored in the same format as RPL system programs. Pointer P.RPG the address where this library begins.
- 9IH overlay OVRS. Resident subroutines used by the main segment and loaded into the upper portion of PP memory to allow for expansion of the main segment or any other overlay.

In addition, the system overlay, OBP (generate banner page) is used. Also, 1LS calls the following system programs.

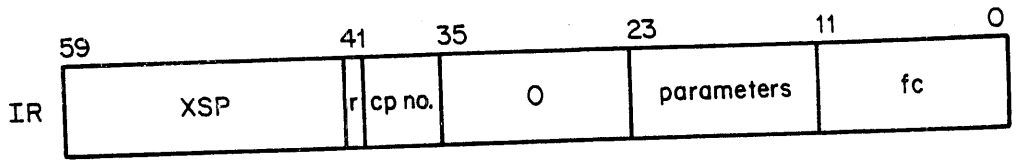
Program	Description
ODF	Drop files
1AJ	Job advancer

Program	Description
1DL	Display overlay loader
CIO	Combined input/output
XSP	E/I service processor
QAC	Queue file processor

Figures 33-6 and 33-7 are flowcharts of the 1LS routine.

XSP - SERVICE PROCESSOR

XSP is called by the Export/Import executive to assist in certain functions that require more time or space than are available for individual processing tasks within the executive. The call to XSP is formatted as follows.



fc Function code:

1 Validate user number (VUN)

2 Make job entry (MJE)

XSP calls the following programs.

Program	Description
OAV	Verify user number
OBF	Begin file
OVJ	Verify job and user cards

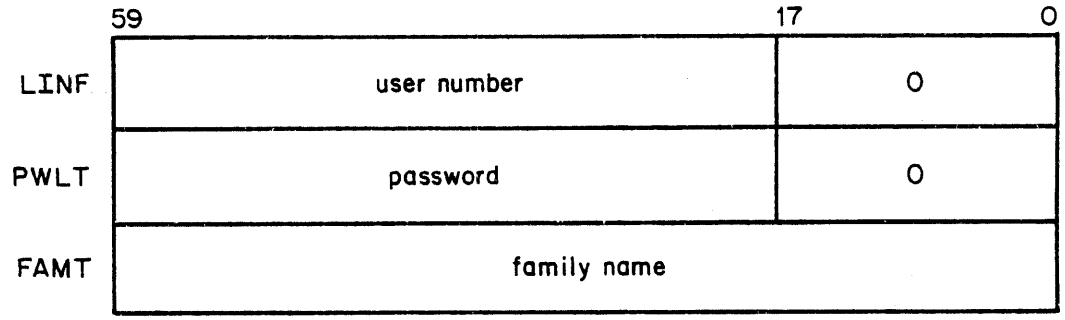
XSP consists of the IVE (process function code) routine which determines that it was called by E/I 200 and that the function code is valid. IVE (refer to figure 33-8) then exits to either VUN or MJE.

VALIDATE USER NUMBER (VUN)

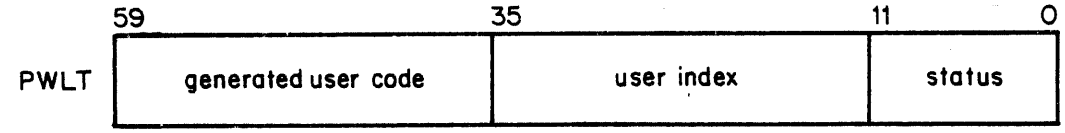
This routine validates the user number for terminal login. If present, the family name is used to establish the correct validation file. The validation file is searched (via call to OAV) for the user number and if it is found and the password given matches the one required, the user index is placed in the response word. If the user number is invalid, the user index is set to 000000.

The input register parameters for this routine include the table index of the user number received in bits 23 through 12 and the function code (V.CUN) in bits 11 through 0.

The format of the request words are as follows.



The format of the response word (returned in PWLT) is as follows.



status Status response:

1 Returned for complete response

3 Returned if user logged in before

5 Returned if user not permitted access to the system (security count exceeded)

MAKE INITIAL JOB FILE ENTRY (MJE)

This routine calls OVJ to create the job card, assigns the sequence number, creates the FNT/FST entry, writes the system sector, and exits with the FNT address in FET+4 and the status in FET+0 set to 15B.

The input register parameters for this routine include the scan index in bits 36 through 30 and the FET address in bits 29 through 12. FET+0 contains the first four characters of the job name (hashed user index). FET+5 is the job card address.

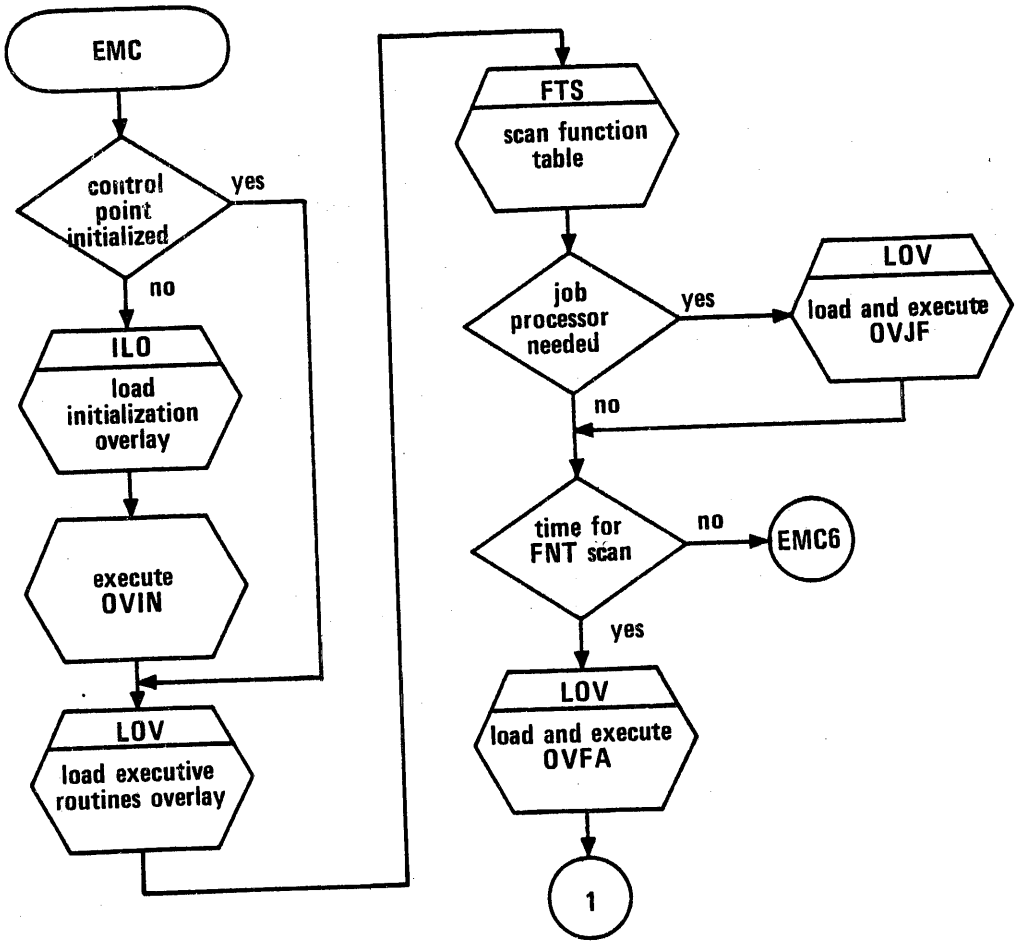


Figure 33-6 1LS - Executive Main Control

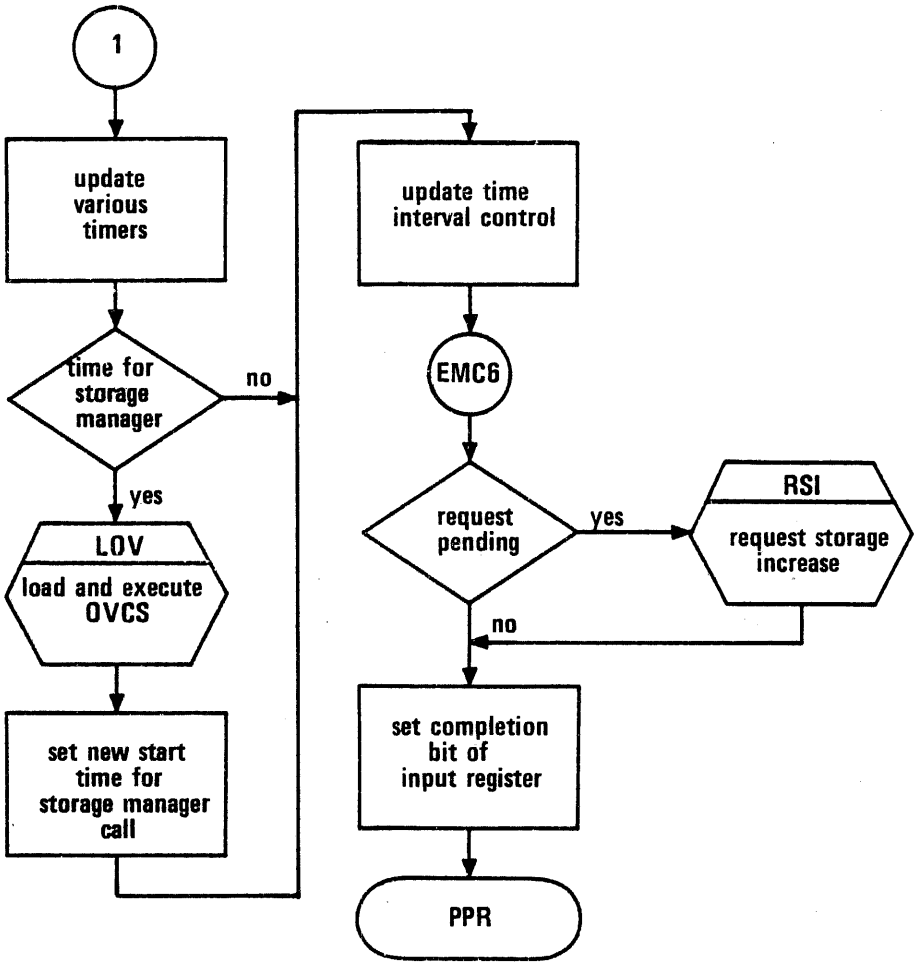


Figure 33-6 1LS - Executive Main Control (Continued)

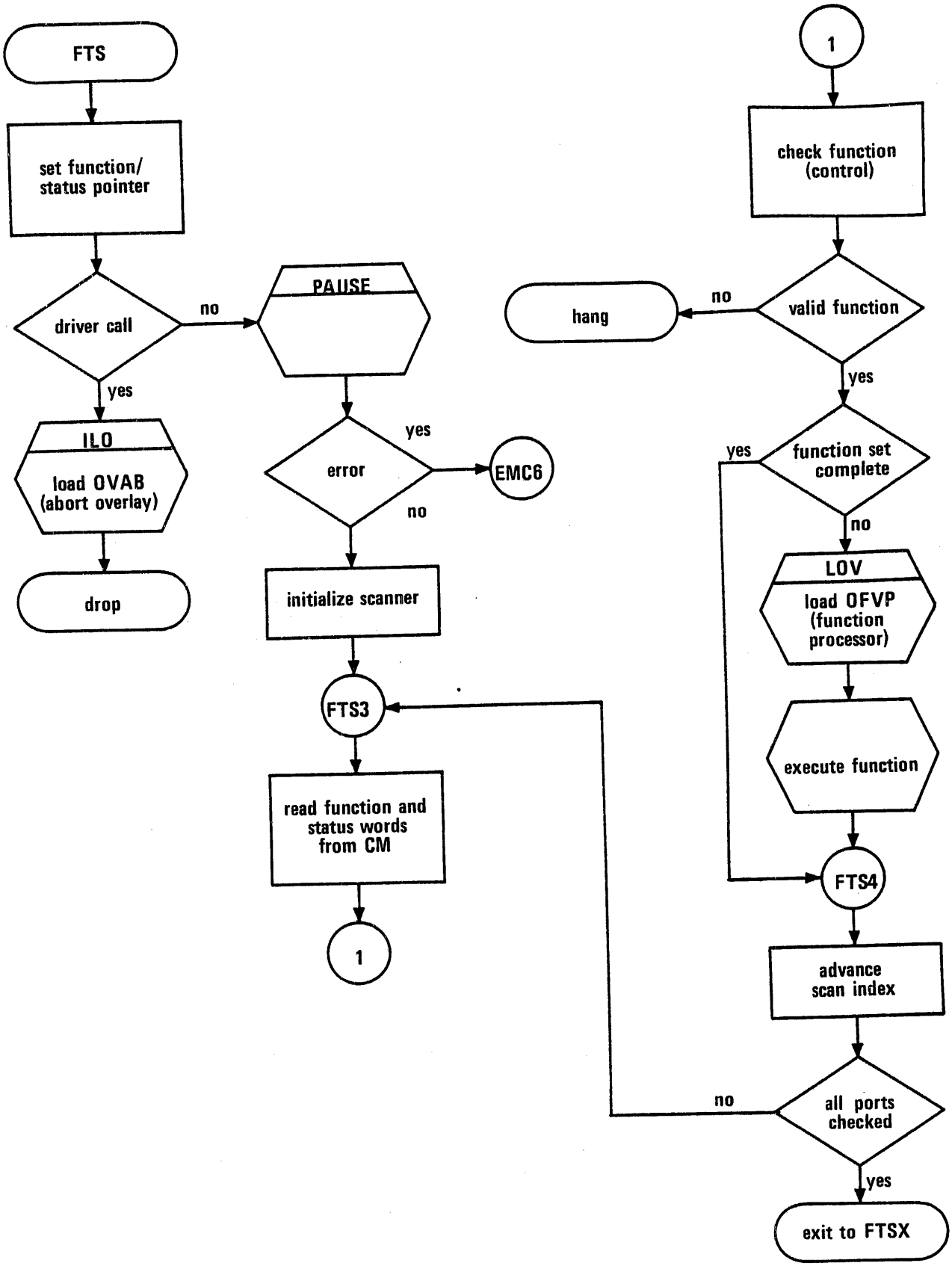


Figure 33-7 Function Table Processor

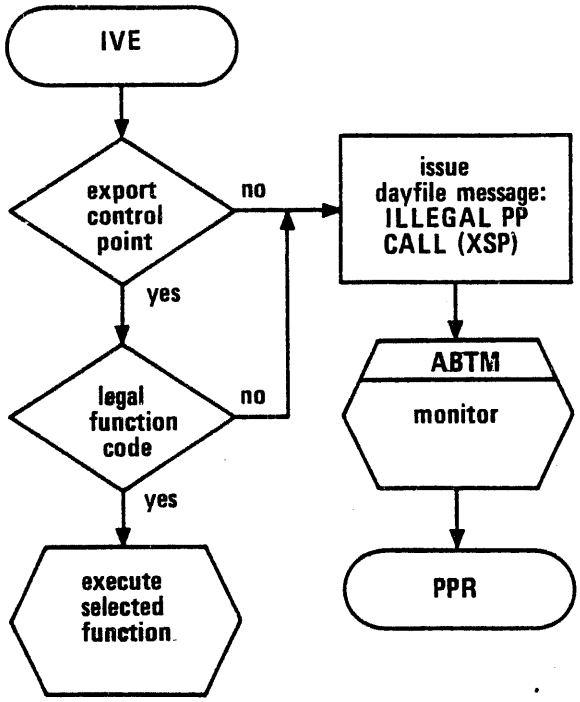


Figure 33-8. XSP - Main Entry

1ED - MULTIPLEXER DRIVER

The multiplexer driver program is a dedicated PP program designed to drive one 6671 multiplexer connected with up to sixteen 200 User Terminals or other devices with similar interface characteristics. The designed line rate is 2400 bps.

This program is initially loaded by the E/I 200 subsystem executive and is controlled by that executive. The driver periodically checks the system storage move flag and, if necessary, issues a pause function to the monitor. During storage move, no references to central memory are allowed. Activity with the terminals is not disrupted in most cases of storage move because of internal buffering in the driver. If a drop of the E/I 200 subsystem is necessary (either because of an operator stop or subsystem malfunction), the executive must set the stop bit in status word zero to cause the I/O driver to release the channel, its reserved equipment, and stop. External to internal codes and vice versa are done via conversion tables.

The following are the major divisions within the driver program.

- IED (control driver). Times the I/O cycles to the multiplexer.
- IOS (input/output with multiplexer). Performs the actual input/output with the multiplexer when directed by the control driver and resets the reentry table addresses.
- COS (control switch). Directs the specific activity for each multiplexer port, initiates new activity as directed by the executive, and transfers control to the following reentrant routines.
- CON (poll to connect multiplexer line). Probes each line with all addresses searching for a response. When a response is sensed, the executive 1LS is informed to login the terminal.
- MSG (write message to display). When directed by 1LS, this section is activated to send one message from the message buffer to the remote display screen.
- PRT (print on remote printer). When directed by 1LS, this section is activated to transmit one buffer block to the remote printer. Routine 1LS is informed at the end of each block so that end of output processing or remote operator directives can be processed if necessary.
- RDC (read cards from remote card reader). One block of cards is read from the terminal and the appropriate function is issued to 1LS to inform it of more cards, last block, bad codes, and so on.

- STA (read operator's message). This, along with sense terminal condition, is used to process input messages from the remote device. The messages are placed in the terminal message buffer for translation by E200CP. Any action required by an operator message is initiated from 1LS.

- STA (sense terminal condition). When a connected terminal is otherwise inactive, it is periodically checked for messages originating from the remote terminal or other action required by the remote terminal when not active.

Figure 33-9 shows the 6671 multiplexer port data words. Figure 33-10 is a flowchart of the 1ED main loop.

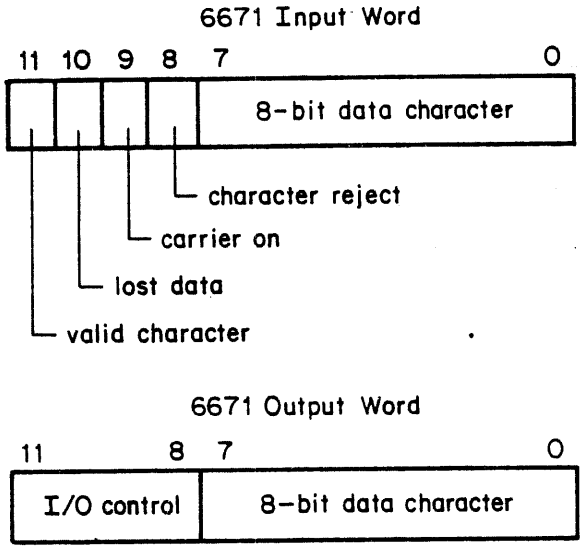


Figure 33-9. 6671 Port Data Word

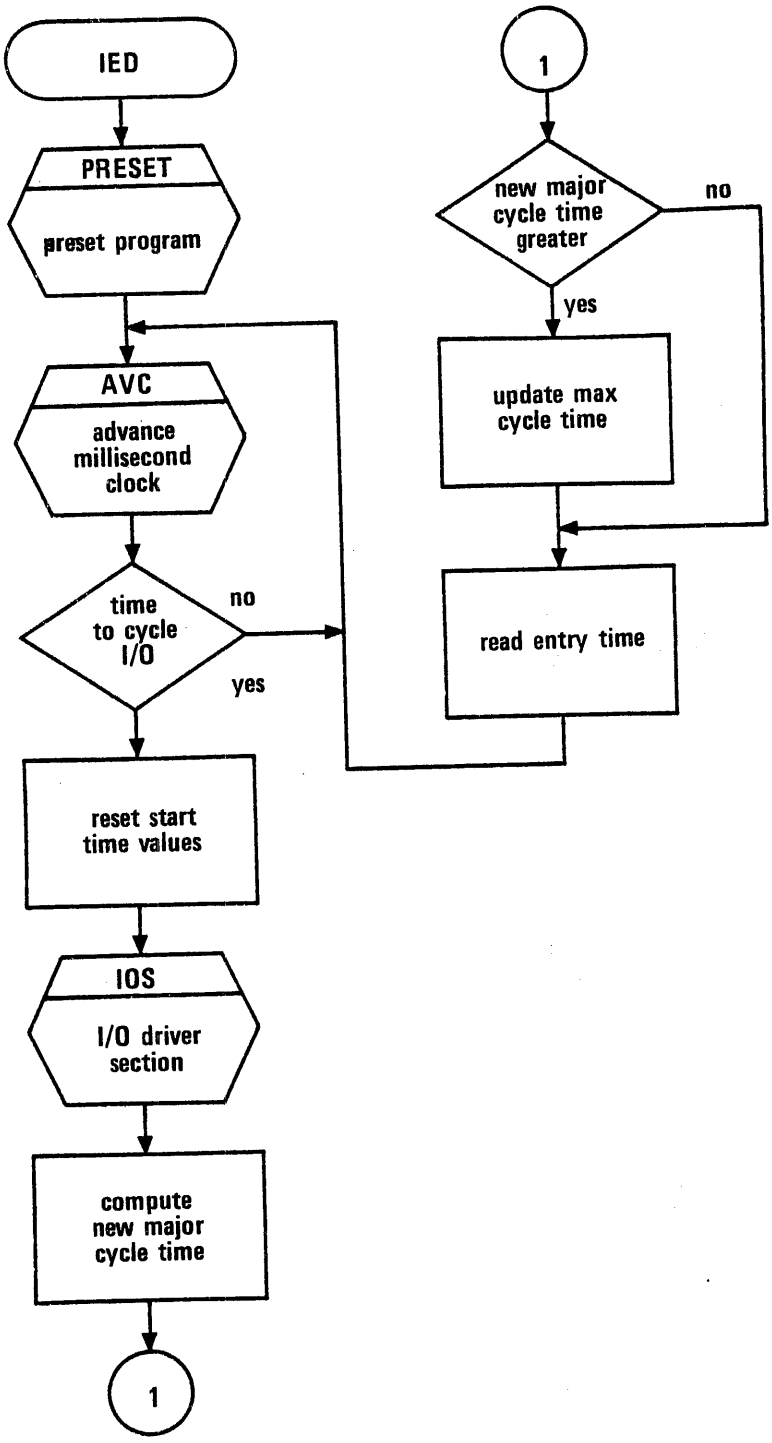


Figure 33-10. 1ED Main Loop

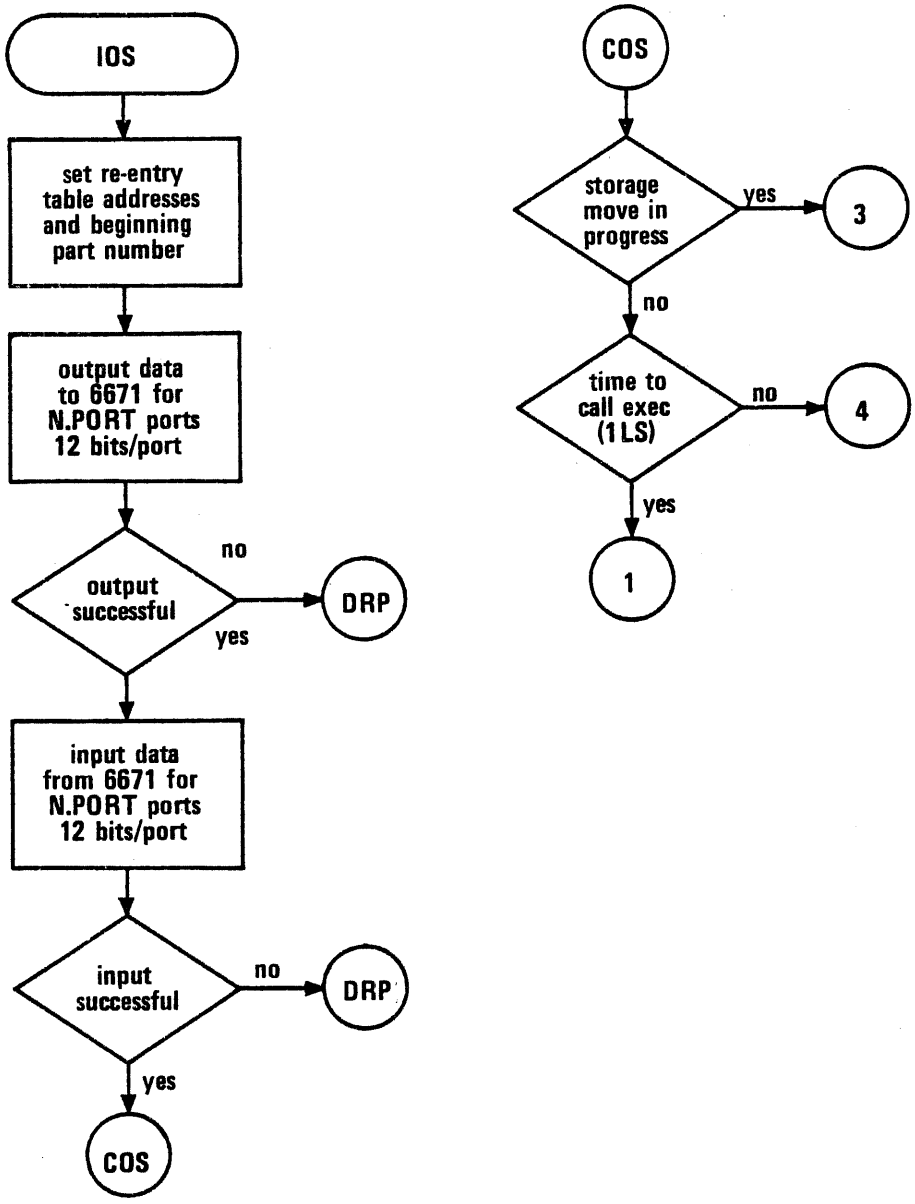


Figure 33-10. 1ED Main Loop (Continued)

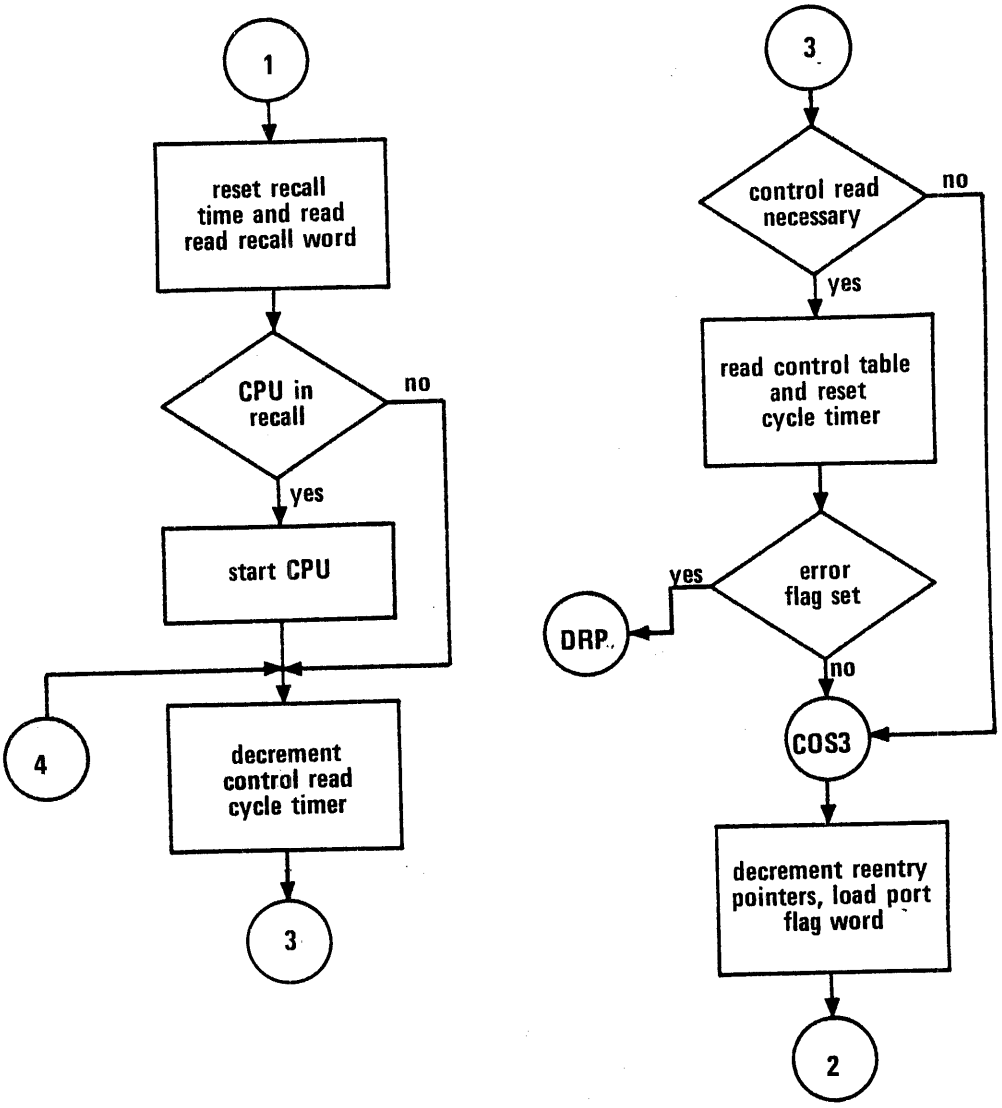


Figure 33-10. 1ED Main Loop (Continued)

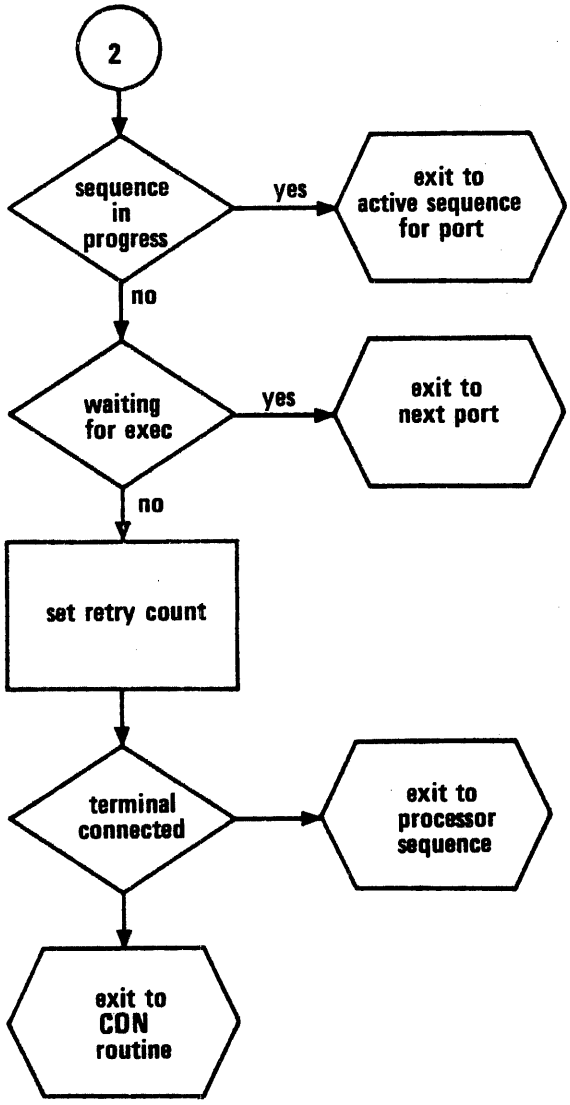


Figure 33-10. 1ED Main Loop (Continued)

INTRODUCTION

A queued file is any file residing in a queue. Queued files have file types INFT (input), ROFT (rollout), PRFT (print), PHFT (punch), TEFT (timed/event rollout), or one of the special file types (S1FT, S2FT, and S3FT). When the file is in the queue, it is not at a control point; that is, the control point number field in the FNT is zero.

File routing is the process by which a file is entered into a queue or retrieved from a queue. The queue management concepts presented in this section deal only with input and output (print and punch files) queues; the file types ROFT and TEFT are dealt with by job scheduling concepts.

The special file types (S1FT, S2FT, and S3FT) are provided to ease local modifications that may require special queue manipulations and are treated as distinct queue types throughout the system. Due to their similarities, print and punch files are usually referred to as output files with the peculiarities of each type being noted on an exception basis.

QUEUED FILE CONTROLS

The following paragraphs describe validation requirements for the use of queued files.

DISPOSED OUTPUT VALIDATION

Output file validation controls the total number of files that can be queued to the output queue by a job during its execution. The intent of this control is to limit the total number of output queue FNT entries consumed by the job's output files. Output files include all files queued to the print or punch queue by DSP, LFM, OUT, and CIO.

The MODVAL parameter OF defines the job output file limit for each user and is maintained in the system validation file. During execution of the user's job, the limit resides in a six-bit field in the control point area. Whenever a queuing function is processed, the output file limit is interrogated prior to entering the file into the output queue. If the file limit has not been exceeded, the file is entered into the designated queue. Unless the user is validated for an unlimited number of job output files, the limit maintained in the control point area is documented by the UDAM monitor function. In the event that the output file limit has been reduced to zero, a dayfile message is issued indicating that the job's output file limit has been exceeded. The job is then aborted.

A six-bit field is defined in the control point area, bits 23 through 18 of word ACLW, to maintain the job's current output file limit. If the field is initially set to 77B, the user's job output files are not limited. Otherwise, the value is decremented as print and punch files are queued until the count reaches zero for the user's job.

DEFERRED BATCH VALIDATION

Deferred batch validation controls the total number of deferred batch jobs the user is allowed to have in the system concurrently. The intent of this control is to prevent a user from inadvertently or intentionally flooding the system with deferred batch jobs. Deferred batch jobs are jobs the user has initiated with the SUBMIT, LDI, or ROUTE control statements, or with the SUBMIT or ROUTE macros.

The MODVAL parameter DB defines the deferred batch limit for each user and is maintained in the system validation file. During execution of the user's job, the limit resides in a three-bit field in the control point area. This field is read by QFM and DSP to determine the user's deferred batch limit before a file is placed into the input queue. If the user has system origin privileges (determined by COMPCUA) or is validated for an unlimited number of deferred batch jobs (determined by COMPCVI) the file is entered into the input queue.

Otherwise, the control point areas and FNTs are scanned for jobs belonging to the user. Only batch (BCOT) and remote batch (EIOT) origin jobs belonging to the user are counted. These jobs include all jobs initiated by BATCHIO, RBF, or EI200 and by the user with the SUBMIT, ROUTE, or LDI control statements and any queued file generated by these jobs. The job doing the submitting is not counted toward the deferred batch limit. The system determines if a job belongs to a user by comparing the user index hash of the job name with the user index hash generated by the user index found in the control point area. If the total number of jobs belonging to the user is less than the deferred batch limit, the file is entered into the input queue. If the deferred batch limit is exceeded, a dayfile message is issued indicating the user has exceeded his deferred batch limit and the job is aborted.

SECURITY COUNT VALIDATION

Security count validation controls the number of security violations a user may attempt before the user is denied access to the system. The intent of this control is to prevent a user from using the SUBMIT and ROUTE macros to determine valid user number/password combinations.

The MODVAL parameter SC sets the security count for each user. This count is maintained in a six-bit field in the system validation file. When a breach of security is detected, OAV is called to decrement the violator's security count in the validation file. When a user's security count has been decremented to zero, the user is not allowed access to the system. The following message is issued to the use's dayfile and the job is aborted or logged off.

ILLEGAL USER ACCESS - CONTACT SITE OPR.

The security count must be reset by MODVAL before the user is permitted access to the system.

Currently, security violations are detected by CPM when an invalid secondary user statement is entered, or by QFM and DSP when a job submits a job file with an invalid primary user statement. In addition to decrementing the user's security count, the user's job is aborted or logged off. This allows 1TA, 1LS, and CPM to check the user's security count when he attempts to regain access to the system.

RBF and EI200 do not decrement the security count of the user number that is logged in to the remote terminal when an invalid primary user statement is encountered in a job file.

A six-bit field for the security count is maintained in the system validation file for each user. If this field is set to 77B, the user's security count is unlimited. Otherwise, OAV decrements the field by one for each security violation attempted until the count becomes zero.

QUEUED FILE SYSTEM SECTOR

All information describing the properties of a queued file is kept in the file's system sector. This allows the descriptive information to be recovered over any type of deadstart and it allows the limited space available in the FNT/FST entry for the queued file to be used efficiently.

The queued file system sector consists of four general areas. The first 10B CM words are used for data common to all file types. In most cases, these fields are entered into the system sector by common deck COMPWSS (write system sector). The next three CM words are used for file dependent data. For input files, this area is set with job statement information. For output files, this area contains repeat count, line/card limit, and dayfile random address. The third area of the system sector is common for both input and output files. This area contains file routing and queue recovery information. The final area in the queued file system sector is reserved for installation usage.

The definition of the individual fields in the queued file system sector is found in common deck COMSSSE (System Sector Equivalences). The system sector is illustrated in section 2.

INPUT FILE EQUIVALENCES

The following input file equivalences are defined for input file system sectors (location relative to BFMS).

Relative Location	Symbol	Definition
50	JISS	Job input data
50-51	JSSS	Job sequence number
53	JTSS	Job step time limit
54	JFSS	Job flags: 1/EI job, 10/0, 1/KEYPM
55	JCSS	Job card CM field length
56	JESS	Job and ECS field length (not used)
57-60	CRSS	Number of cards read
62-65	TNSS	Terminal name

OUTPUT FILE EQUIVALENCES

The following output file equivalences are defined for output file system sectors.

Relative Location	Symbol	Definition
51	PFSS	Reserved for previous system compatibility
52-53	RASS	Dayfile random address
54	SCSS	Spacing code for 580-PFC
55	LCSS	Line/card limit index
62	RCSS	Repeat count
63-64	RTSS	Random index
65	RBSS	Requeue buffer

COMMON INPUT/OUTPUT FILE EQUIVALENCES

The following equivalences are common to input and output file system sectors.

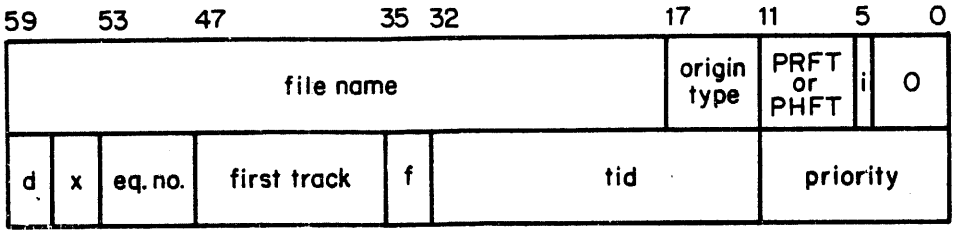
Relative Location	Symbol	Definition
67	OTSS	File origin type
70	PRSS	File priority
71	MISS	MMF machine ID
72-73	FLSS	File length in sectors/10B
74	ICSS	Internal characteristics
75	ECSS	External characteristics
76	FCSS	Forms code
77	DVSS	Device code
100	DCSS	NOS/BE device code
101-105	DASS	User number and index of destination
106-111	FDSS	Family name of destination
112	ODSS	Family ordinal of destination (not used)
113-114	DISS	Destination terminal ID
120-124	FSSS	FST entry for queue protected files
125-130	FMSS	Family name of creator
131	OOSS	Family ordinal of creator (not used)
132-136	ACSS	User number of creator
137-143	CDSS	Recovery flag; queued file creation date and time
144-150	JNSS	Job card job name
151-155	OHSS	Origination host name
156-162	DHSS	Destination host name
163-167	FRSS	File routing control includes data in user block and file placed in queue flags
170	VASS	User validation block
315	EISS	End of system initialized data
315	UBSS	User data block (10 CM words)

NOTE

If the same information is contained in both the FNT/FST and system sector (for example, job origin or forms code), it must agree in order for queue management to work properly. It is the responsibility of the routine that put the file into the queue to properly initialize fields in the system sector.

QUEUED FILE FNT/FST

To reduce the amount of overhead required to read the system sector of output queue files to obtain routing information, abbreviated forms of the forms code, device code, and external characteristics are kept in the queued file FNT/FST entry. This information is usually sufficient to allow the system to determine if an output file should be selected for processing without having to read its system sector. The format is as follows.



i	System sector information bit: 0 No queue information present 1 Queue information present
d	Device selection code
x	External characteristics code
f	Forms code
tid	Terminal identification

The values for the d, x, and f fields are specified in common deck COMSJIO.

DEFERRED ROUTE

Bit 5 of the FNT is used to indicate that the system sector contains queued file information. The concept of a deferred route involves the use of this bit. A deferred routed file is a queue type file assigned to a control point with bit 5 set; that is, the system sector has already been correctly formatted for a queue type file that has not yet been released to its queue.

A file becomes a deferred file in one of the following manners.

- A ROUTE/DSP call was made to do a deferred route on a file
- A queued file with a formatted system sector was extracted from its queue and assigned to a control point (for example,) QAC attaches an output file to BATCHIO or RBF for processing
- An input file is created with system sector information when reading card decks by QAP, 1LS, or VEJ.

FILE ROUTING CONCEPTS

This section discusses different routing options and how they are related to each other.

TERMINAL ADDRESSING

Associated with every queued file is a terminal address used by the system to route files to a particular line printer, card punch, or remote batch terminal. The terminal address consists of three items: destination family name (FDSS); destination user number (DASS); and terminal ID (DISS). These items are used dependent upon the origin type of the file. For batch and system origin (BCOT/SYOT) files, the destination family name and user number (FDSS/DASS) are zero. The terminal ID (DISS) is set to the batch ID code associated with the equipment that is to be used to process the file. For remote batch origin (EIOT) files, the destination family, user number and TID are the family name, user number and user index respectively of a remote batch terminal to which the file must be routed for disposition.

The default terminal address for any job is determined at the time the job's input file is created. VEJ, QAP, 1LS, DSP, and QFM set this default information into the job's input file system sector. When output is disposed by the job, the disposing routine through common deck COMPUSS (update system sector) moves the default terminal information from the input file system sector to the output file system sector. In this manner, all output is routed back to the terminal that created the job's input file.

When processing EIOT queued files, the terminal ID field of the FNT is insufficient for proper routing since destination family/user information is also required. Therefore, the system sector of a remote batch terminal (EIOT) queued file is always read in terminal address processing.

ALTERNATE ROUTINGS

There are times when a file is to be routed to a terminal other than the one specified by the default terminal address. This diversion may be done through ROUTE, DISPOSE, LDI, SETID, or SUBMIT control statements or their equivalent macros. These routines all allow the specification of a destination family/user number/TID. The information so specified replaces the default FDSS, DASS, and DISS values in the system sector of the new queued file.

The terminal address of a queued file may also be changed by the ALTER function of QAC while the file is in the queue.

Once the file has been entered into the queue, the terminal information is used by QAC and 1LS for selection of the queued file for the specified terminal.

SPECIAL FILE ID CODES

File ID codes 70-77 (octal) are reserved for system usage as defined in common deck COMSSSJ (special system job equivalences). Two special ID codes ZRID (71) and SOID (77) are related to queue file management. ZRID is used to indicate that an ID code of zero has been assigned to a file by LFM (SETID statement or macro) to differentiate it from the absence of an ID (which is also zero). LFM sets ZRID into the FST ID field (bits 59 through 54) and COMPUSS converts ZRID to its proper zero value when queuing the file. SOID indicates that the file should be dropped rather than queued at job completion. If the input file has an SOID, all nondeferred, zero ID output files are dropped at job completion.

NOTE

The SETID processing in LFM changes the ID field in the FST only. COMPUSS moves the ID into the systems sector when the file is queued. The DSD console command ENID performs a similar operation for BCOT and SYOT output files. The ID so specified is lost if the queued files are recovered over a level 0 deadstart.

DEVICE SPECIFICATION

The terminal address defines a set of printers and punches for output file disposal. Additional routing information may be specified to further define the type of equipment that should process the output file. This routing information consists of device selection code, external characteristics, and forms code.

COMPUSS sets default values for device code (DVSS), external characteristics (ECSS), and forms code (FCSS) into the system sector and FNT/FST entry. DSP is the only system program that alters all these values. Once a file has been queued, the ALTER function of QAC can be used to change the forms code. The device selection code and external characteristics cannot be changed once the file has been queued.

This information is used by QAC to select a queued file for disposal. Device selection is used to specify a particular type of device that is to process the file; for example, a 580-12 (LR) printer. The external characteristics specify what format the data is in. This information is used to specify the print train for line printers and the punch format for card punches. For example, for print files, this could be 64 character set BCD (B6). For punch files, it could be 80 column binary (80 COL). The external characteristics is dependent upon the file type. The device selection codes and external characteristics are defined in common deck COMSJIO.

FORMS CODE

Forms code consists of two alphanumeric characters that can be assigned to an output file. An output file with a forms code can be processed only on a device with a matching forms code. The forms code is set by DSP but can be altered by QAC. Common deck COMPVFC (verify forms code) is used to validate the forms code when it is associated with a file.

In the queued file FNT/FST entry, only 3 bits are available for the F field (forms code). Only the forms codes AA, AB, AC, AD, AE, AF, and null can be represented by this restricted field. A forms code value of EXIN (7) in the FST entry indicates that the system sector must be read to determine what forms code is required to properly dispose the queued file.

A special output file is a file that has been routed to a microfilm printer or plotter, a hard copy printer or plotter, or a plotter. Such a routing is indicated by the special equipment code value SPDV (67B) in the D, X field of the queued file FNT/FST. The actual device code is set in the F field. In this situation, the system sectors must be read to determine the proper external characteristics and forms code for the file.

QUEUE MANAGEMENT EQUIVALENCES

Common deck COMSJIO defines equivalences for programs doing queue management. The values defined in COMSJIO are the following (these codes are used in the FST and indicates that the information is too detailed for the FST and the system sector should be used).

Value	Symbol	Definition
7	EXIN	Extended information flag
6	SPEQ	Special device
67	SPDV	Special equipment

Print file external characteristics (these codes are used in both the FST and system sector) are as follows.

Value	Symbol	Definition
0	DFEX	Default printer
1	RREX	Reserved
2	A4EX	A4 - ASCII graphic 48 character set (not supported)
3	B4EX	B4 - BCD graphic 48 character set (not supported)
4	B6EX	B6 - CDC graphic 63/64 character set
5	A6EX	A6 - ASCII graphic 63/64 character set
6	A9EX	A9 - ASCII graphic 95 character set

Punch queue external characteristics (these codes are used in both the FST and system sector) are the following.

Value	Symbol	Definition
0	DFFR	Default punch
1	PBFR	SB (system binary)
2	P8FR	80COL (80 column binary)
3	RIFR	Reserved
4	PHFR	026
5	P9FR	029
6	ASFR	ASCII

Internal characteristics (these codes appear only in the system sector) include the following.

Value	Symbol	Definition
0	DCIC	Display code
1	ASIC	ASCII
2	BNIC	Binary
3	RRIC	Reserved

NOS device codes (these codes are used only in the queued file FST) are the following.

Value	Symbol	Definition
0	PRDV	LP (any printer)
1	P1DV	Reserved
2	P2DV	Reserved
3	LRDV	LR (580-12)
4	LSDV	LS (580-16)
5	LTDV	LT (580-20)
0	SBDV	SB (system binary)
0	P8DV	P8 (80 column binary)
0	PBDV	PB (system binary)
0	PUDV	PH (punch coded)

The following device codes are reserved for future enhancements.

Value	Symbol	Definition
0	FRDV	FR (microfilm printer)
1	FLDV	FR (microfilm printer)
2	PTDV	PT (plotter)
3	HRDV	HR (hard copy printer)
4	HLDV	HL (hard copy plotter)
5	I1DV	Reserved for installations
6	I2DV	Reserved for installations

NOS/BE compatible device codes (these codes are only used in the systems sector) include the following.

Value	Symbol	Definition
10	PUDC	PU (punch)
20	FRDC	FR (microfilm printer)
22	FLDC	FL (microfilm plotter)
24	HRDC	HR (hard copy printer)
26	HLDC	HL (hard copy plotter)
30	PTDC	PT (plotter)
40	PRDC	LP (any printer)
41	P1DC	Reserved
42	P2DC	Reserved
43	LRDC	LR (580-12 printer)
44	LSDC	LS (580-16 printer)
45	LTDC	LT (580-20 printer)

CREATING A QUEUED FILE

Queued files are normally created under two conditions. The first case involves creating a job input file by reading a job deck of cards from a card reader via BATCHIO, RBF, or EI200. In this case, the terminal address for the input file system sector is determined by the particular queue file processor based on properties of the card reader or remote batch terminal reading the card deck. The second case is the creation of a queued file by a job, that is, the file is created at a user control point. In this case, COMPUSS is used to move the default terminal address from the input file system sector to the system sector of the file being queued. This situation is encountered as part of ROUTE, DISPOSE, SUBMIT, OUT, CIO close/unload and close/return, or job completion (1CJ) operations.

QUEUE MANAGEMENT ROUTINES

Many system routines are involved in file routing and queue management. Utilities which manipulate active and inactive queues (QREC, QFSP, QDUMP, QLOAD, and QMOVE) and their helper PP routine (QFM) are described in the NOS System Maintenance Reference Manual. The remainder of this section details the key file routing and queue management functions: COMPUSS, DSP, and QAC.

COMPUSS

The key routine in creating output queue files is common deck COMPUSS (update system sector for disposable files). This

common deck is used by all PP routines that place files into the output queues. Its purpose is to generate and write the system sector for a queue file.

COMPUSS consists of two main subroutines: USS (update system sector) and WQS (write queued file system sector).

USS - UPDATE SYSTEM SECTOR

USS is used to update the information in a file's system sector prior to entering the file in the queue. The preserved file bit is set if I/O queue protect is active.

COMPUSS requires the presence of common decks COMPRSS, COMPSEI, COMPWSS, COMPSSE, COMSCPS, COMSJIO, and COMSSSE.

Since mass storage error processing is selected by COMPUSS, the error processor for COMPRSS should be defined.

If USS$ is defined, COMPUSS allows the calling program to modify the queued file system sector before it is written back to mass storage. The calling program should make a call to USS to set up the default system sector in BFMS. The desired fields are then set and a second call is made to WQS to write the system sector and preserve the file. If USS$ is not defined, the queued file system sector is built and written to mass storage and the file is preserved with one call to USS.

The system sector fields listed below are set by COMPUSS when one of the following files are processed.

- Any local (LOFT) file
- Print (PRFT) files with system sector information bit not set
- Punch (PHFT) files with system sector information bit not set
- Input (INFT) files with system sector information bit not set

The fields updated are the following.

Field	Description
(JNSS - JNSS+4)	Job card name from job file
(ACSS - ACSS+4)	Origination user number from job file
(OHSS - OHSS+4)	Origination host name from job file (for future use)
(FMSS - FMSS+4)	Origination family name from job file
(OOSS)	Origination family ordinal from job file (for future use)

Field	Description
(DASS - DASS+4)	Destination user number from job file
(DHSS - DHSS+4)	Destination host name from job file (for future use)
(FDSS - FDSS+4)	Destination family name
(ODSS)	Destination family ordinal from job file (for future use)
(DVSS)	Destination device identification from job input file
(DISS - DISS+1)	Destination terminal identification from job file
(RCSS)	0, repeat count (output files only)
(RTSS - RTSS+1)	0, restart random address (for output files only)
(RBSS - RBSS+1)	0, restart buffer (output files only)
(LCSS)	Lines or cards limit (output files only)
(FCSS)	0, forms code (output file only)
(ICSS)	0, internal characteristics (output files only)
(ECSS)	0, external characteristics (output files only)
(SCSS)	0, spacing code (print files only)
(DCSS)	NOS/BE device code
(CDSS - CDSS+4)	Creation date
(MISS)	Machine ID

The following fields are set whenever COMPUSS is called to write the queued file system sector.

Field	Description
(OTSS)	Origin type
(FLSS)	File length in sectors/10B
(FASS)	FST address

If USS$ is defined, the calling program must set:

Field	Description
(FNSS - FNSS+4)	FNT entry
(FSSS - FSSS+4)	FST entry
(PRSS)	Priority

The main subroutine of COMPUSS is USS and is flowcharted in figure 34-1.

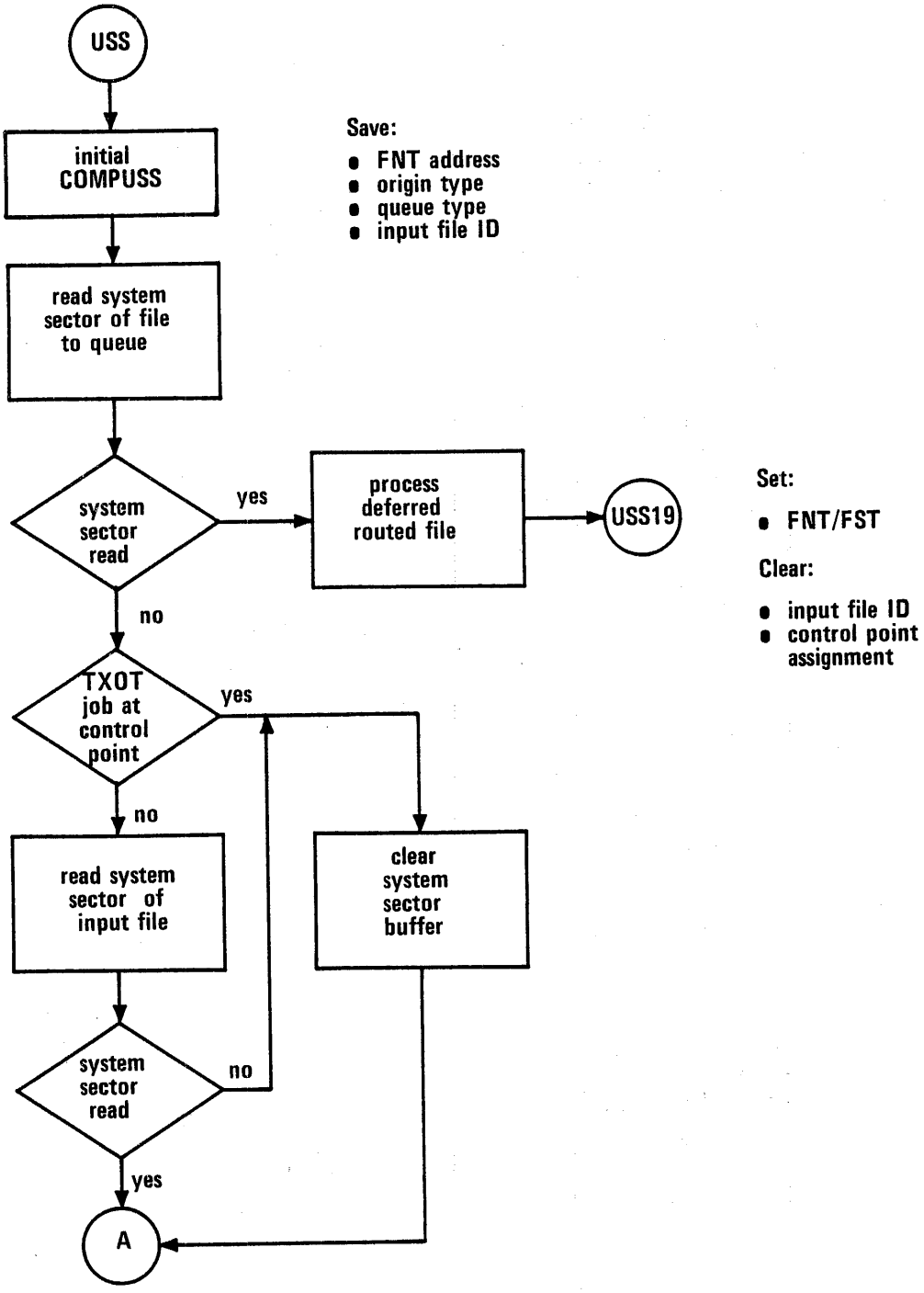


Figure 34-1. COMPUSS - Subroutine USS

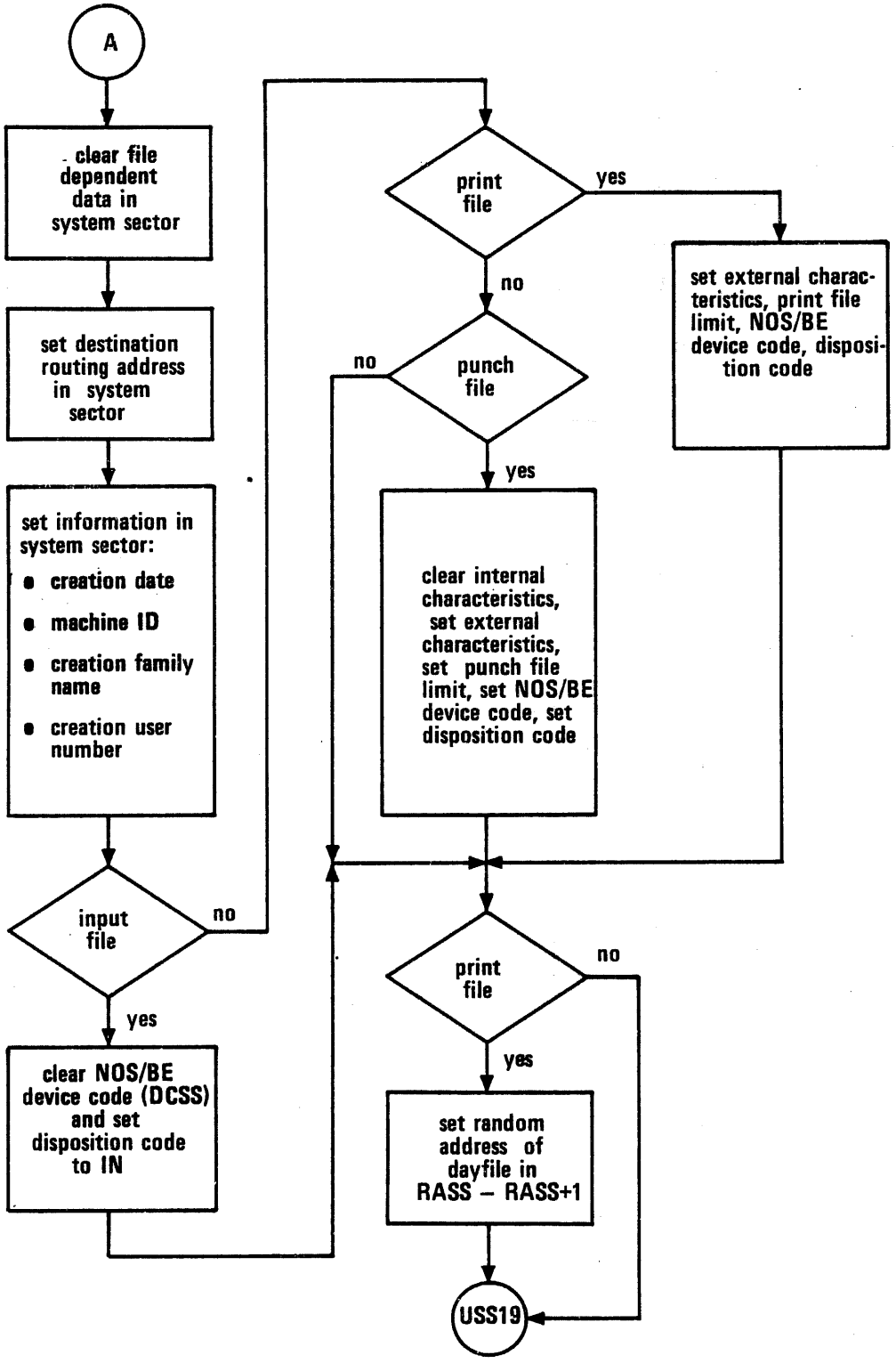


Figure 34-1. COMPUSS - Subroutine USS (Continued)

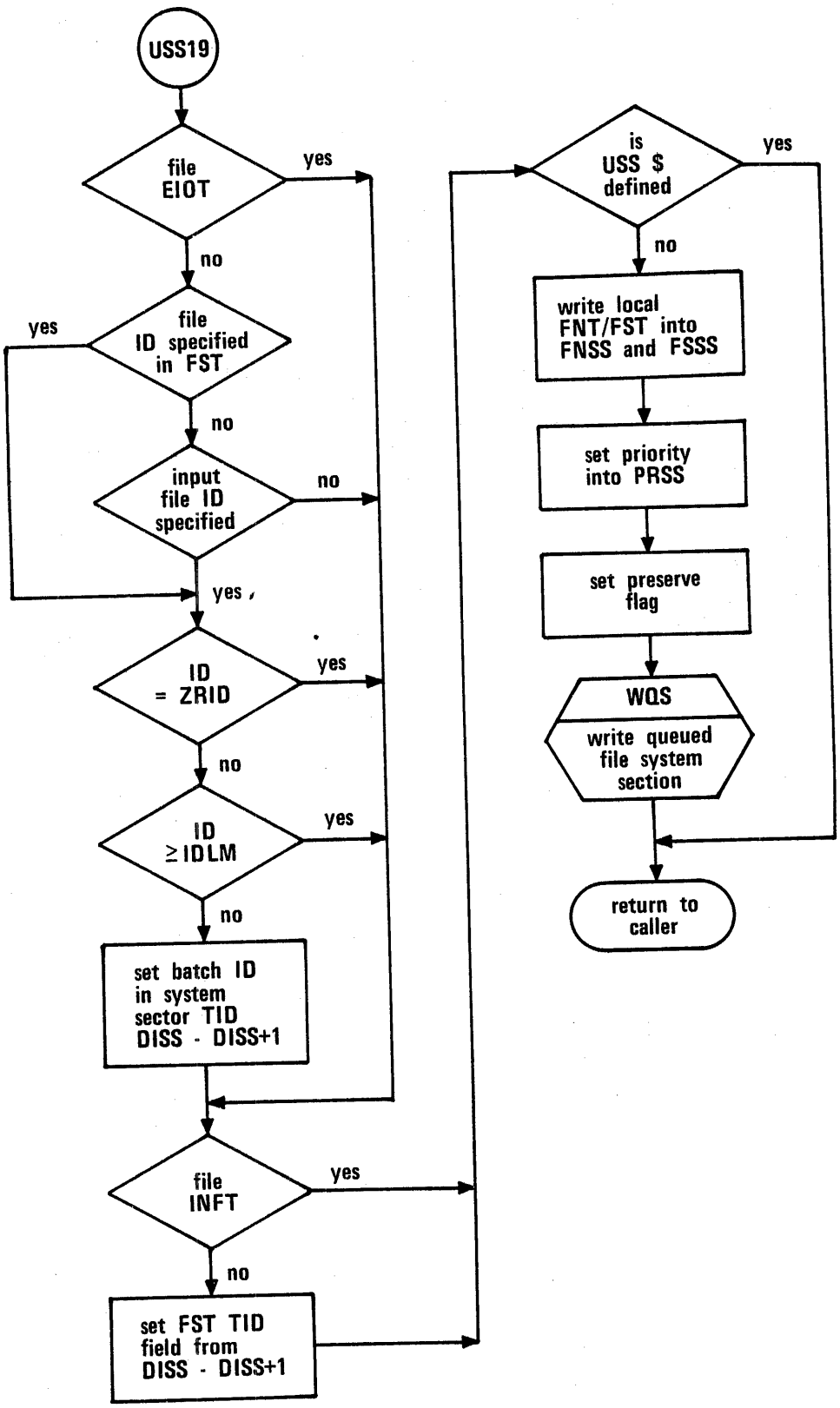


Figure 34-1. COMPUSS - Subroutine USS (Continued)

The manner in which USS sets up the system sector buffer depends on the status of bit 5 in the FNT of the file that is being queued. If bit 5 is set, the system sector has been formatted previously. The system sector is read and if the read is successful, the dayfile random address (if one exists for print files) is entered into RASS through RASS+1. The file ID is entered into the system sector for non-EIOT jobs as the terminal ID (DISS through DISS+1) and this value (terminal ID) is entered into the FST for all output files. The system sector is then written according to the USS$ setting (as described below). The FNT/FST entries in use by the caller are then replaced with those from the system sector (FNSS/FSSS).

If the system sector had not previously been written (bit 5 is not set in the FNT entry) or was unable to be read successfully, a default system sector is built as follows. If the job is not a time-sharing origin (TXOT) job, then there may be routing information in the input file system sector that needs to be passed on to the output file system sector. An attempt is made to read the input file system sector and if successful, the destination routing information (FDSS, DASS, DISS) contained there is used for the output file. If no data existed in the input file system sector (bit 5 of the FNT was not set) or the sector was not able to be read, the destination family (FDSS), destination user number (DASS), and terminal ID (DISS) are entered into the system sector dependent upon the origin type of the file being queued and the caller's origin type.

The terminal ID is either the file ID of the input file or the user index of the destination family/user number. The creation date is entered into CDSS from PDTL, the machine ID set into MISS, and the creation family name (FMSS) and user number (ACSS) are entered from either the control point area (UIDW) or SSJ= call block (UIDS). Defaults are set for the disposition code (DCSS), spacing code (SCSS), internal and external characteristics (ICSS/ECSS), line/card limits (LCSS, as obtained from the control point area or SSJ= call block), and device code DVSS). Finally, the random address of the dayfile for print files (if one exists) is entered into RASS through RASS+1. The file ID is then set into the system sector for non-EIOT jobs as the terminal ID (DISS through DISS+1) and into the FST. The system sector may then be written (as described below) and the FNT/FST entry replaced with that from the system sector in the calling program.

If symbol USS$ is not defined in the calling program, the FNT/FST entries are reentered into the system sector (FNSS/FSSS), and into central memory and the priority set into the system sector (PRSS). At this point all places that contain the FNT/FST entries (system sector, FNT/FST in central memory, and in the caller's direct cells) will have the same values. The system sector is then written and the file preserved by a call to subroutine WQS. To queue the file in this case, the caller only needs to write the FNT/FST entry returned by the USS call. If USS$ is defined, the newly built (or rebuilt) system sector is left in the buffer for use by the caller.

DSP and 1CJ are examples of the usage of COMPUSS with USS$ defined and not defined respectively. DSP builds the system sector by a USS call but then adds disposition characteristics to the sector before writing it using a WQS call. 1CJ, on the other hand, relies entirely on default information and subsequently lets the USS call build and write the system sector.

WQS - WRITE QUEUED FILE SYSTEM SECTOR

WQS completes the system sector as it writes it. If the file is being requeued and preserved, the file in queue flag is set in FRSS. (File has been queued; this flag is used by CIO to determine whether to queue or drop a file on a close/unload operation. Close/unload is used by QAP to end a file.)

The recovery flag is set in CDSS (the recovery flag, when not set, indicates that the system sector was not created by an NOS 1.2 system; QFM will reformat the system sector if it recovers a queued file without this bit set), the file length (determined by rounding up to the next multiple of 10B sectors the result of a SEI call) is set in FLSS through FLSS+1, the origin type is retrieved from the FNT (FNSS) and entered into OTSS, and the FST address is set into FASS. The sector is then written by a call to WSS. WSS enters the common system sector information as it writes the sector: equipment (EQSS); first track (FTSS); and packed date/time (DTSS).

Upon return from WSS, if the file is to be preserved, an STBM monitor function is issued to set the preserved file bit and request a checkpoint of the device. This request is ignored by CPUMTR if QPROTECT is not enabled.

Callers of COMPUSS

COMPUSS is used by all routines that enter files into the input/output queues (CIO, DSP, LFM, OUT, QFM, and 1CJ). Any routines that are locally introduced must follow COMPUSS conventions if they wish to queue input/output files in order to guarantee correct processing of queued files.

DSP - DISPOSE FILE TO I/O QUEUE

DSP places a file into the input or output queue. DSP is called by QAP, RBF, the queue utilities (QDUMP, QLOAD, and QMOVE), and through use of the ROUTE macro. The SSJ= caller may provide a system sector to be used in the disposal; this technique is typically used by the queue utilities.

The main routine of DSP is flowcharted in figure 34-2. For a complete description of the DSP call and parameter block refer to volume 2 of the NOS Reference Manual.

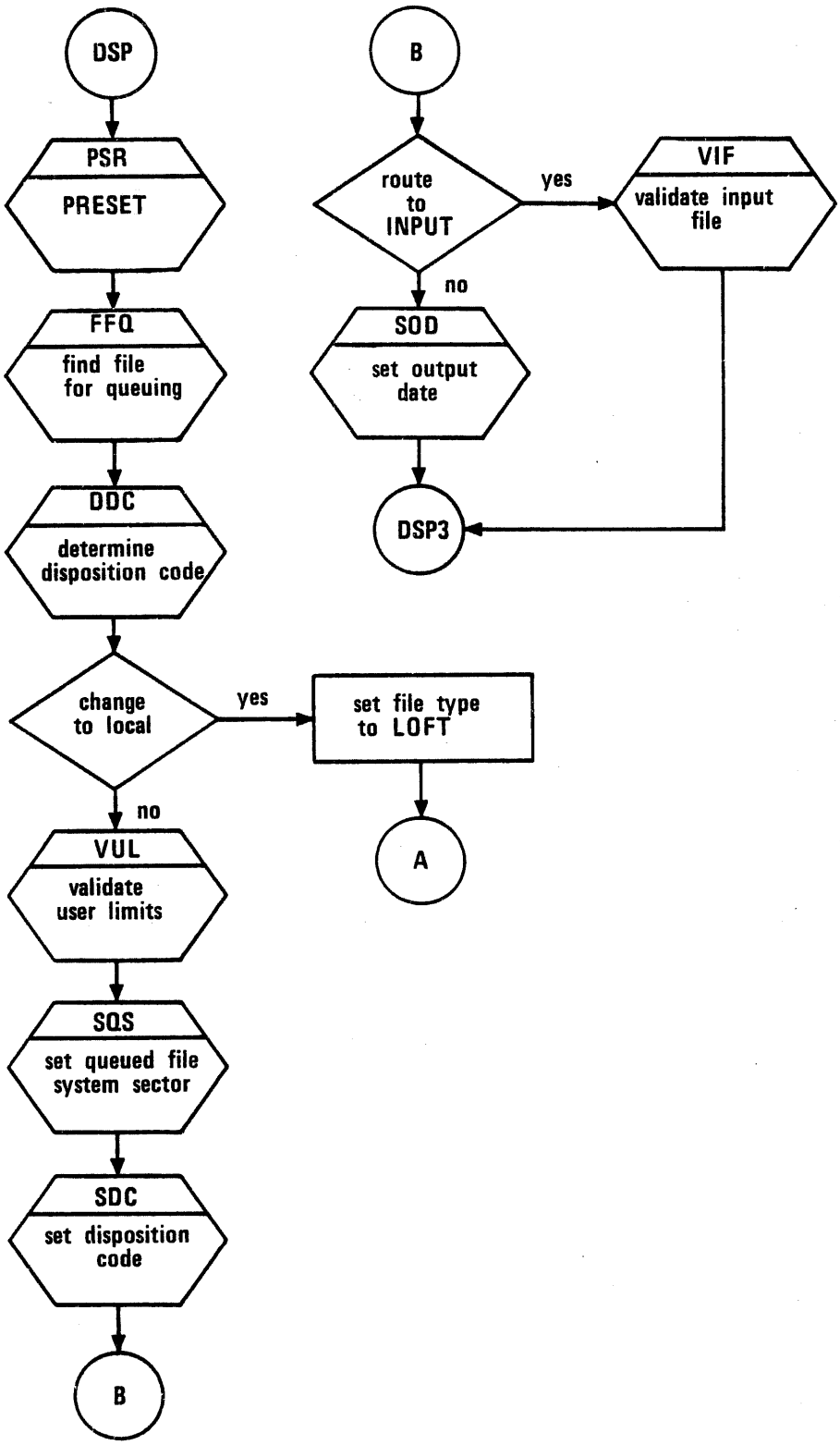
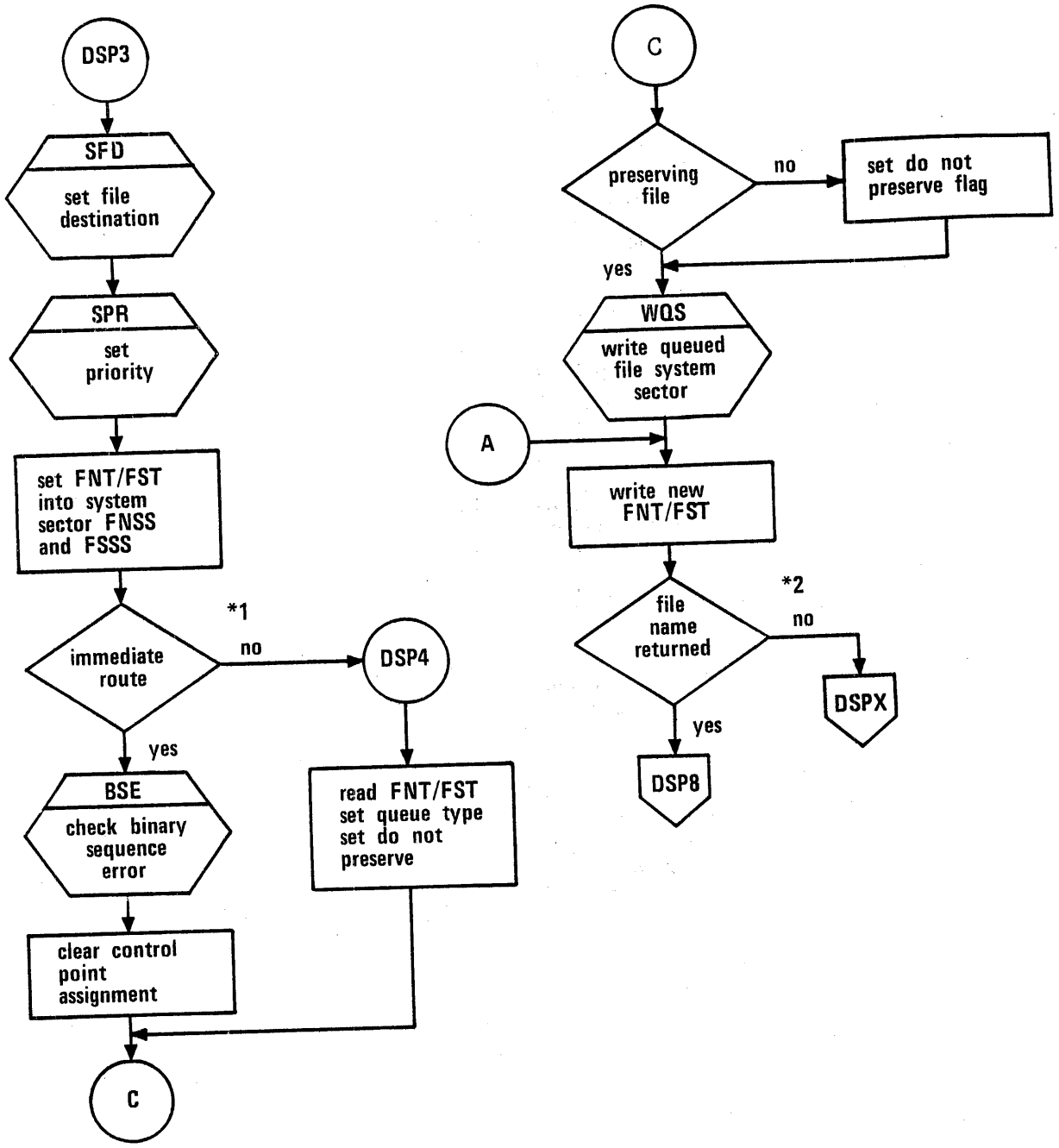
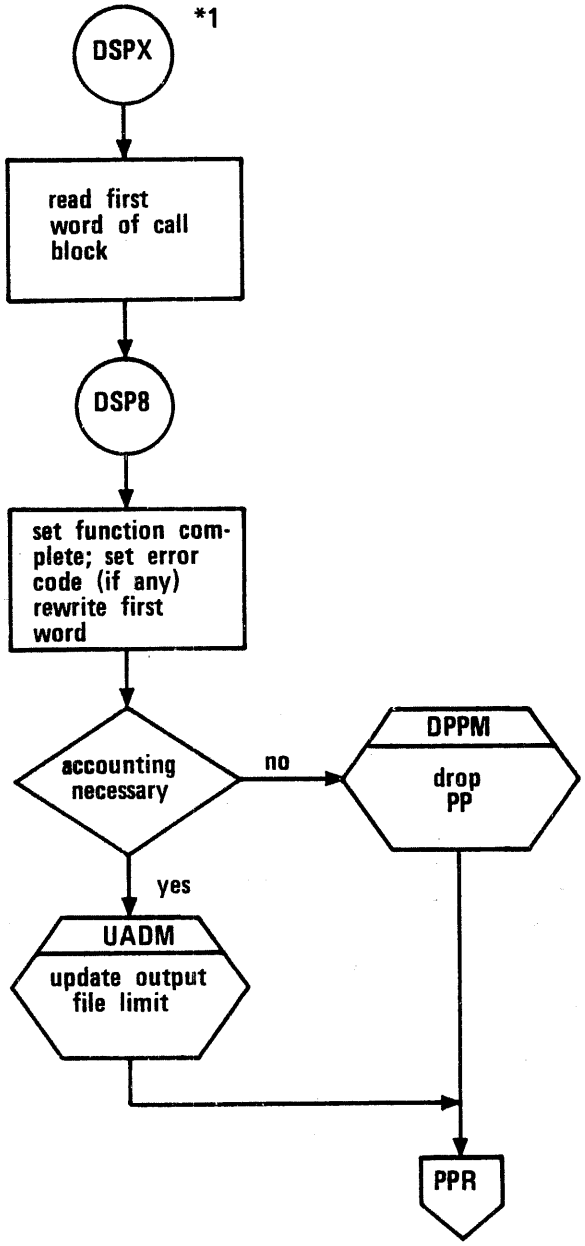


Figure 34-2. DSP Main Routines



*1 Flag bit 0 not set if immediate route.
*2 Flag bit 5 set if returning name.

Figure 34-2. DSP Main Routine (Continued)



*1 Entry from Error Processor.

Figure 34-2. DSP Main Routine (Continued)

DSP preset (PRS) performs the central memory address and parameter validation. The flag bits and disposition code are extracted from the parameter block for future use. If an SSJ system sector block is specified (as would be done in requeuing the file by the queue utilities), the SSJ= validation of the caller is done and if successful the system sector block is read to DSP's system sector area and rewritten with DSP exiting. If no system sector address is specified, default keypunch mode is retrieved from IPRL and appropriate modifications made to default values in the disposition table (TODC). If an origin code is forced, it is validated to be an acceptable NOS origin type for a queued file (BCOT, EIOT, or SYOT). The forcing of the origin type is only valid for system origin callers and is primarily used for network startup.

Subroutine FFQ (find file for queue) is called to verify that the file either exists, is not busy and can be queued, or creates a file to be queued.

Subroutine DDC (determine disposition code) is called to set up the disposition code. The disposition code is passed in the parameter block, but is only used if the disposition code bit is also set in the DSP flags. If the disposition code bit is not set and file is not a queued file or is a queued file without the system sector information bit set, then a default disposition code is established by scanning the list of special file names for the code associated with the special name (OUTPUT = LP, PUNCH = PU, PUNCHB = PB, and P8 = P8). The disposition code is then validated and an index to the properties for it (TODC table) is saved for future use. A DSP call for a local file not having a special name will in effect do nothing; the file remains at the control point.

Subroutine VUL is then called to validate user limits (deferred batch and output file limits).

Subroutine SQS is called to retrieve the system sector into BFMS. This is accomplished by a call to USS in common deck COMPUSS. DSP has COMPUSS assembled with USS$ defined, thus not writing the system sector during the USS call.

The USS either returns the system sector from the queued file or builds a defaulted system sector.

Subroutine SDC is then called to set the disposition code in the system sector. This involves setting the FST and system sector words ECSS, ICSS, and DVSS with appropriate default information for the particular disposition code. The data is obtained from the table of default routine information (TODC) which contains the NOS device mnemonic, queue type, internal/external characteristics, forms code, device code, external characteristics for FST, NOS system sector device nmemonic, and NOS/BE system sector device mnemonic.

If the routing is to the input queue, subroutine VIF is called. If there is no data in the system sector, then the routing is the first attempt to put a file (job) into the input queue. The first sector must not be empty as this sector should contain the control statements for the job. These control statements are validated as job/user statements by a call to OVJ with appropriate error processing for job and user control statement errors.

If the routing is to the output queue, subroutine SOD is called. A repeat count, if specified, is entered into system sector word RCSS, internal and external characteristics (ICSS, ECSS) are entered as specified or defaulted, and forms code (FCSS) and spacing code (SCSS) are entered. The flag bits are used to determine if these values are specified.

Subroutine SFD is then called to set the file destination and special ID in the TID field of the FST and in DISS through DISS+1 of the system sector.

Subroutine SPR is called to set the file's priority in the FST and system sector word PRSS from the job control block for the queue and origin type. The priority is set to ERPS if a job statement error is detected through JCSS. For output files, DSP rounds the specified priority to fit the allowable priority range for the particular origin type.

The file's FNT/FST entries, as updated by all these subroutine calls, are then updated in the system sector words FNSS and FSSS.

On input files from BATCHIO (QAP), if an immediate route is being done (flag bit 0 not set), subroutine BSE is called to set the FST for binary sequence error processing if it had been detected.

Subroutine WQS in COMPUSS is called with file preservation specified unless input flags were set to not preserve the input file.

DSP completes its processing by writing the FNT/FST for the file, returning information to the call block, and drops the PP. It is at this time when the FNT/FST is rewritten that the file is either queued (immediate route) or left assigned to the control point (deferred route). If an output file is being disposed, the number of disposed output files are decremented through a UDAM monitor function.

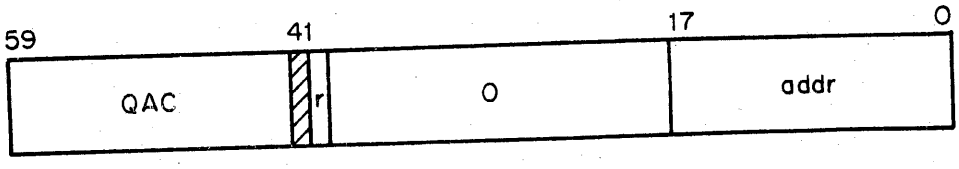
Overlay 3DA is called to process error conditions detected by DSP processing.

The error messages issued by 3DA are listed in volume 1 of the NOS Reference Manual. If in the call block, bit 12 of the flags is set, the error code is returned and no message issued.

Once a file has been entered into the queues, QAC may be used to attach the file to a control point, alter destination routing information, purge the file, or return information about the file. QAC is used by BATCHIO to attach output files; EI200 to drop jobs; RBF to attach, alter, and status jobs and files; FNTLIST to status queues; and QALTER to status and alter queues.

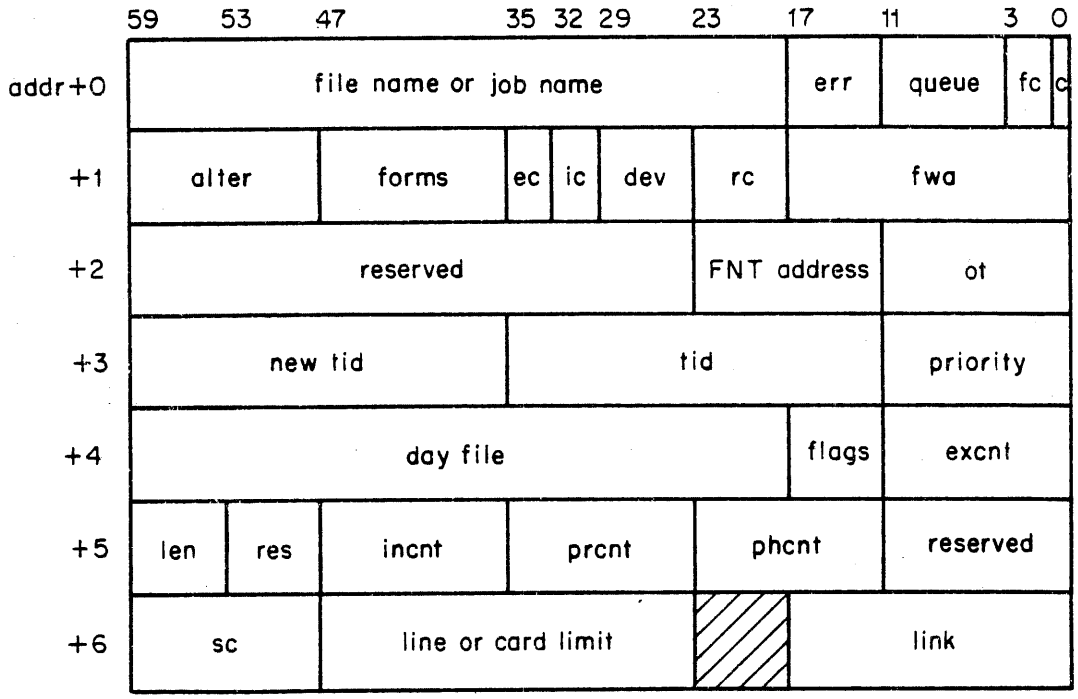
QAC is called by 1IO and RBF to identify files for disposal on unit record equipment or at a remote terminal. The ACQUIRE macro may also be used to make QAC calls. QAC calls are valid only if the caller has a priority greater than MXPS.

The format of the QAC call, its parameter block and error codes are as follows.



r Autorecall bit (optional)
addr FWA of parameter block

The parameter block format is as follows.



err Error code returned (complete description follows).

queue Queue type:

Queue	Description
1	Input file
2	Output file
4	Punch file
8	Special output
16	Executing job (at control point or in rollout queue)

fc Function code:

Code	Description
0	ALTER function
1	GET function
2	PEEK function
3	COUNT function

c Completion bit. Must be 0 on call; set to 1 upon completion.

alter ALTER flags:

Bit	Description
56	FNT flag (check specified FNT only)
55	Ignore upper six bits of disposition field
54	Change spacing code (output files only)
53	Abort job or evict file
52	Change repeat count (output files only)
51	Change forms code (output files only)
50	Change priority (output files only)
49	Change destination TID to new TID
48	Send to central site

form Forms code. IF bit 51 of flag field is set, GET, PEEK, and COUNT use forms code as search criteria. ALTER changes forms code of output files to the specified forms code. If bit 51 is not set, all functions ignore the forms code field.

ec External characteristics (For detailed description, refer to DSP documentation in volume 2 of NOS Reference Manual).

ic Internal characteristics (For detailed description, refer to DSP documentation in volume 2 of NOS Reference Manual).

dev Device code:

Mnemonic	Code	Description
PU	10	Punch
FR	20	Microfilm printer (for future)
FL	22	Microfilm plotter (for future)
HR	24	Hard copy printer (for future)
PL	26	Hard copy plotter
PT	30	Plot
PR	40	Any printer
LR	43	Select 580-12 printer
LS	44	Select 580-16 printer
LT	45	Select 580-20 printer
--	46-67	Reserved for CDC
--	70-77	Reserved for installations

rc Repeat count (maximum value is 37B).

fwa FWA of reply buffer or FWA of dayfile message.

ot Origin type (0 = system, 1 = batch, 2 = remote batch).

tid Terminal ID. If the upper six bits are equal to 77B, the lower bits contain the complement of the CM address of a two-word family name/user number area (each in upper 42 bits of word). If the upper six bits are not equal to 77B, the lower eighteen bits contain the TID.

dayfile Random address of dayfile left justified.

flags Flag bits:

Bit	Description
13	Return extended information
14	Inhibit duplicate file search
15-16	Reserved
17	Dayfile present (output files only)

excnt Executing job count.

len Parameter block length.

res Reserved.

incnt Input file count.

prcnt Print file count.

phcnt Punch file count.

sc Spacing code for output files.

link Link address to next QAC call block. If 0, no address is specified.

The following error codes are returned to the QAC call block when the specified situation occurs.

Error Code	Message	Description
1	INVALID QUEUE TYPE.	QAC was called with no queue type specified, function 1 (GET queued file) was called with than one queue type specified or with input and/or executing queue flags set.
2	NO FILE FOUND.	QAC did not find a file that met the selection criteria.
3	ILLEGAL PRIORITY SPECIFICATION.	The priority field in the call block is 7777B. This is illegal for function 0 (ALTER) and function 2 (PEEK).
4	--	Reserved.
5	CM ADDRESS OUT OF RANGE RANGE.	May occur in one of the following conditions. • The address of the family name/user number block passed in the TID field is out of range • The FWA of the message buffer for function 0 (ALTER) is out of range • The PEEK reply buffer address is out of range
6	--	Reserved.
7	--	Reserved.
10	--	Reserved.
11	TOO MANY FILE TYPES SPECIFIED.	Function 2 (PEEK) must not be called with more than one file type specified in the call block.

Error Code	Message	Description
12	DUPLICATE FILE FOUND.	Function 1 (GET queued file) attempted to attach a file to a control point which had a file with the same name already attached.
13	COUNT OF ZERO INVALID.	The number of reply entries requested from function 2 (PEEK) must be greater than zero.
14	--	Reserved.
15	INVALID FNT ADDRESS/ORDINAL.	The FNT address/ordinal is out of the range of the FNT.
16	--	Reserved.
17	INVALID FORMS CODE.	Forms code specified is not two alphanumeric characters.
20	INVALID TID.	May occur in one of the following situations. • Family name/user number specified is invalid • Specified TID for a batch or system origin job is greater than IDLM • Function 0 (ALTER) is attempting to change the destination origin of a file without specifying a destination TID
21	INVALID ORIGIN TYPE.	The origin type specified in the parameter block is invalid.

QAC is structured into a main routine, resident subroutines, and the following function processors.

3QR	Error processor overlay
3QS	GET function overlay
3QT	ALTER function overlay
3QU	PEEK/COUNT function overlay

All QAC functions use the same search routines located in the resident subroutine area. QAC attempts to validate as much information as possible from central memory (FNT/FST and control point areas) to avoid incurring the overhead of reading queued file system sectors.

The primary search routines are VCI (validate central memory information) and VMI (validate mass storage information). VCI is used to validate information that must be found in the FNT/FST or control point area (origin, priority, queue type, file/job name, etc.). VMI is used to validate information that is kept in the system sector (forms code, external characteristics, destination information, etc.). A simplified flowchart of the validation and interlocking for QAC is shown in figure 34-3.

QAC can process a chain of QAC call blocks. This is accomplished by the link field in word six of the parameter block. Once QAC is finished with a parameter block, the field is checked. If zero, there is no further processing needed and QAC releases the PP. If the field is nonzero, QAC calls PRS (preset) and begins processing the new QAC call block. To reduce system overhead, all of the QAC function overlays are reentrant. This reduces the number of overlay loads if the call blocks are arranged in the proper order.

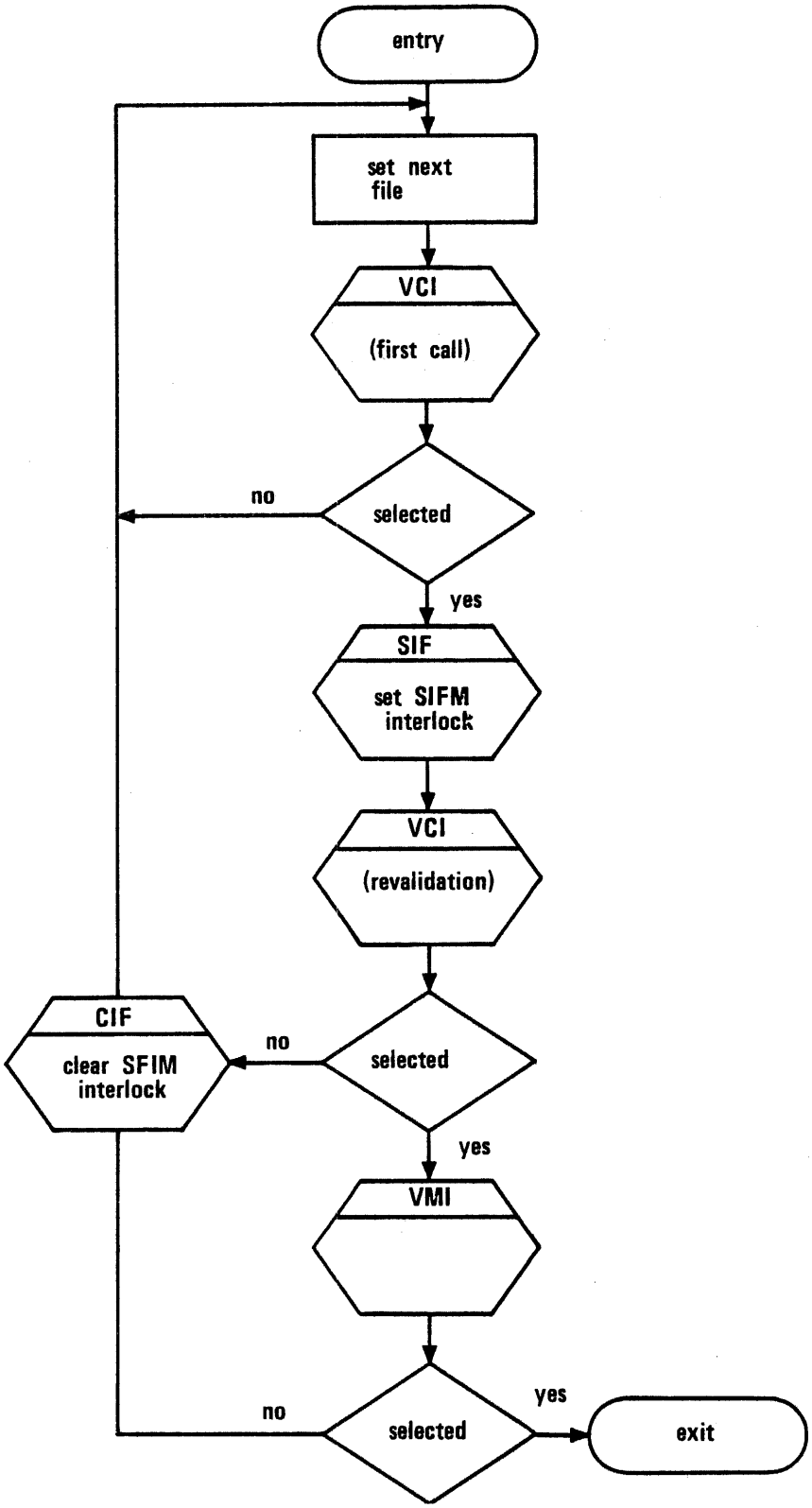


Figure 34-3. QAC Search

It is the validation routines (VCI and VMI) that make use of the routing fields found in the FNT/FST and system sector of queued files. Once a file has been found and interlocked, QAC is able to perform one of its functions upon the file.

QAC Preset

The preset phase of QAC validates the function and identifies the function processors and properties of the function, such as, FNT ordinal passed in QAC block, priority of 7777B illegal, priority is search criteria, and forms code not search criteria.

Function 0 - ALTER

The ALTER function changes requested fields in a queue file or an executing job file system sector and FNT/FST for all files and/or jobs matching the selection criteria. Required fields in the call block include the following.

- Function code (=0) and cleared completion bit
- Queue type (more than one may be specified)
- Alter flags
- Origin type

The beginning of FNT search address, device selection, TID, message text, and file/job name may optionally be specified as search criteria.

The code to alter a queued file is found on overlay 3QT.

The alteration that is requested to be performed is determined by the alter flags in the call block. The flags include the following.

- Send to central site
- Change terminal ID (TID)
- Change priority
- Change forms code
- Change repeat count
- Abort job/evict file
- Change spacing code

Send to Central Site (Output Files)

The alteration for send to central site is to allow EIOT output to be processed at the central site (usually by BATCHIO) rather than at a remote terminal. Although batch and system origin output is processed at the central site, it is possible to route batch/system output to a specified terminal ID if this bit is not set. The TID field and origin in the system sector and FST are changed to the new origin and destination information.

Change Terminal ID (TID)

The alteration of the terminal ID is done in both the FST and system sector (DISS through DISS+1). For system and batch origins the terminal ID must be less than IDLM (70).

Change Priority (Output Files)

The alteration of queue file priority is done within the bounds of the service limits for the origin type of the file as determined by QUEUE parameter settings. The new priority is set in the FST and system sector (PRSS).

Change Forms Code (Output Files)

The new forms code is validated by common deck COMPVFC. A valid forms code is any combination of two alphanumeric character or null. If the forms code is a special forms code (greater or equal to AG), it is set only in the system sector (FCSS). Otherwise it is set in the FST as well. A special forms code is any code that is not null, AA, AB, AC, AD, AE, or AF. These nonspecial values provide the optimum performance since only they can be represented by the 3 bits available for forms code in the FST.

Change Repeat Count

The repeat count supplied in the call block is set into the system sector (RCSS).

Change Spacing Code

The spacing code supplied in the call block is set into the system sector (SCSS).

Abort Job

The abort job option reads the optional message, writes it to the dayfile (system and control point) and sets the operator drop error flag (ODET) by the CEFM monitor function. QAC will change control points to issue the message and set the error flag.

If the job was not at a control point (that is, rolled out), the control point area is read from the rollout file, the message is written into MS1W, the operator drop error code (ODET) set into MS2W, and the control point sector rewritten. The priority of the rollout file is set to SEPS (special error priority) for appropriate processing by the scheduler and rollin mechanism.

Evict File

If a queued file is being dropped, an accounting message with identifier AEPQ is issued to the account dayfile and the file dropped via a call to ODF.

Function 1 - GET

The GET function attaches a print or punch file to the requesting control point. If the priority specified in the parameter block is less than 7777B, the first file found in the FNT search that meets the selection criteria is attached. If the priority is 7777B, the entire FNT is searched for the best file that meets the selection criteria (highest priority file). Required fields in the call block for the GET function include the following.

- Function code (=1) and cleared completion bit
- Queue type
- Priority
- Origin type

The file name, forms code, disposition code, starting FNT ordinal, destination terminal ID, and inhibit duplicate file search flag may be optionally specified.

The code for attaching the selected queued file is contained in overlay 3QS.

If a file that meets the selection criteria is found, information about this file is returned to the caller in the parameter block and an updated FNT/FST entry built for this file at the caller's control point. The information returned includes the dayfile random address (from RASS through RASS+1), file at beginning of information flag, and file length (from FLSS through FLSS+1).

While the response is being formatted and FNT/FST updated, the file is individually interlocked by the FNT interlock mechanism (SFIM) and the FNT interlocked using the FNCT channel.

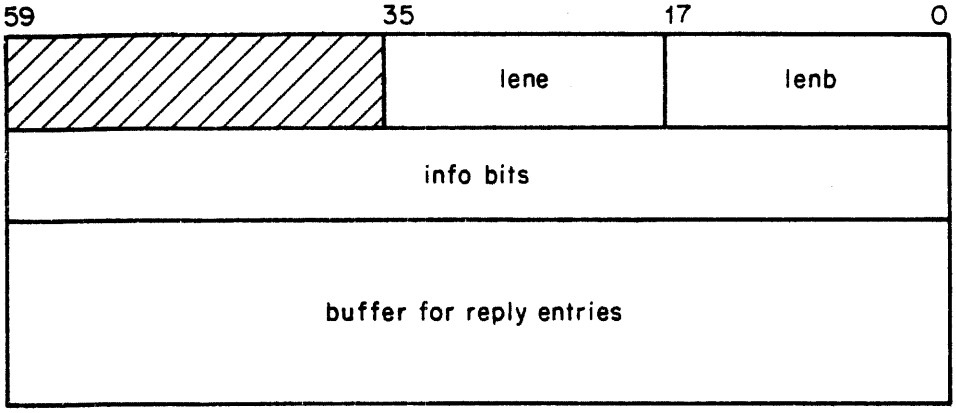
Function 2 - PEEK

The PEEK function returns a list of responses for files and jobs matching the selection criteria. The required fields in the parameter block for a PEEK include the following.

- Function code (=2) and cleared completion bit
- Priority
- First word address (FWA) of reply buffer
- Queue type
- Number of replies
- Origin type

The file/job name, terminal ID, disposition code, forms code, and starting FNT ordinal may optionally be specified as search criteria.

The first two words of the PEEK reply buffer are formatted as follows.



lene Length of reply entries returned by QAC/PEEK.

lenb Length of buffer to receive reply entries.

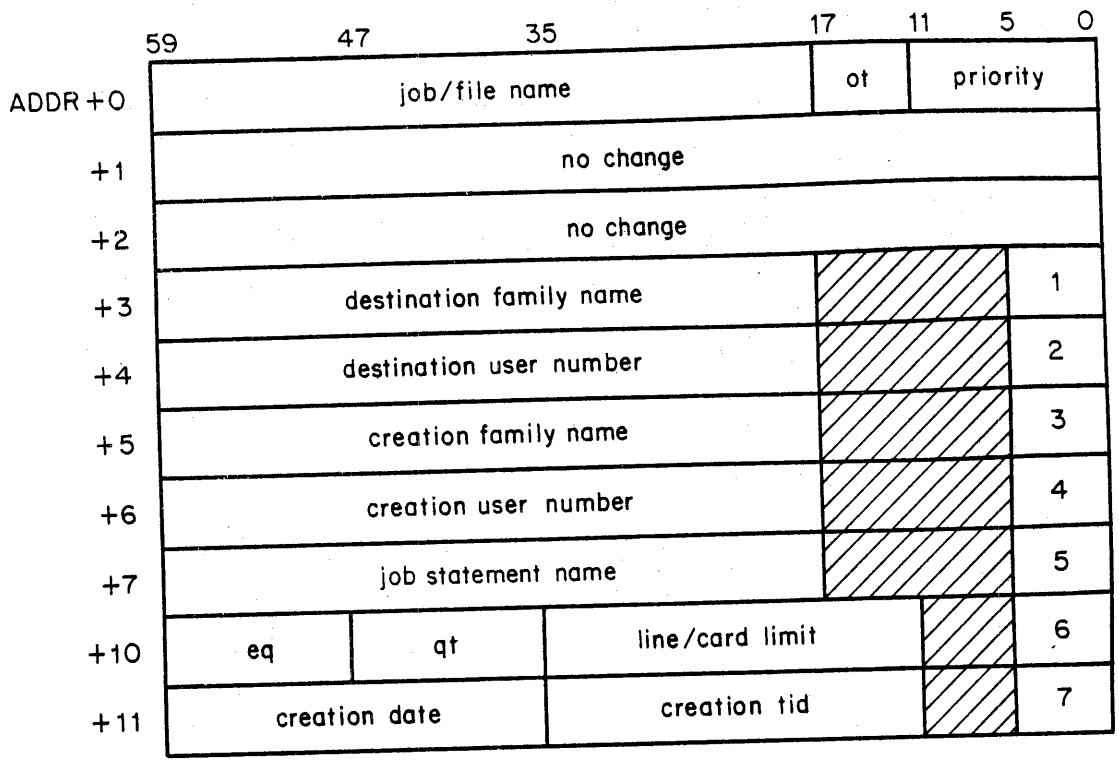
info bits Each bit represents a different word of information that can be returned. If a bit is set, the corresponding information is returned as an additional word to the reply entry. The lower six bits (bits 5-0) of each additional word returned contain an information number that may be used to identify the type of information in that word.

Table 34-1 defines the information bits and information number for each type of information that can be requested.

TABLE 34-1. INFORMATION BITS

Bit	Information Requested	Information Number
-	No information in word	0
0	Not used	-
1	Destination family name	1
2	Destination user number	2
3	Creation family name	3
4	Creation user number	4
5	Job statement name	5
6	Equipment, queue type, and line/card limit	6
7	Creation date, creation TID	7

The format of the PEEK reply entry when all information is requested is as follows.



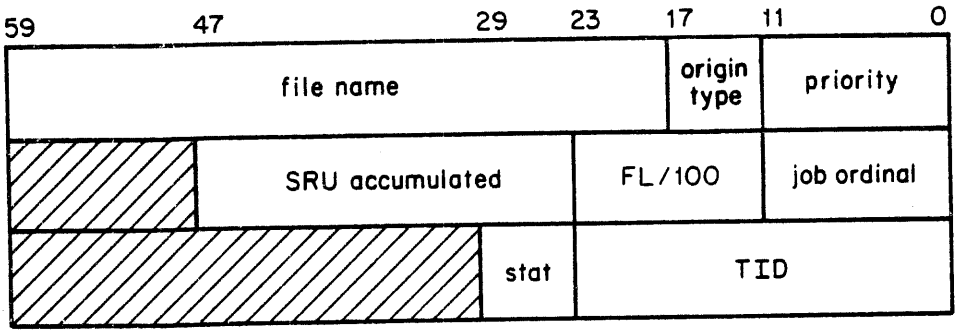
ot Origin type:
- 0 System origin
- 1 Batch origin
- 2 Remote batch origin

eq Equipment number file resides on

qt Queue type:
- 1 Input file type
- 2 Print file type
- 3 Punch file type

The code for the PEEK function is found in overlay 3QU.

The response returned if an executing job satisfies the search criteria has the following format.

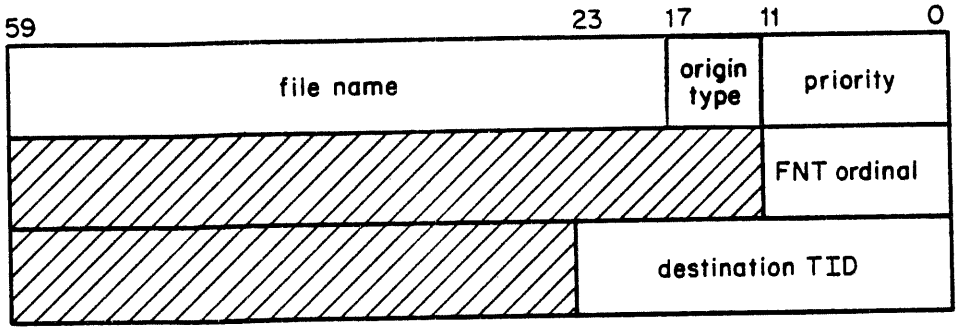


stat Job status:

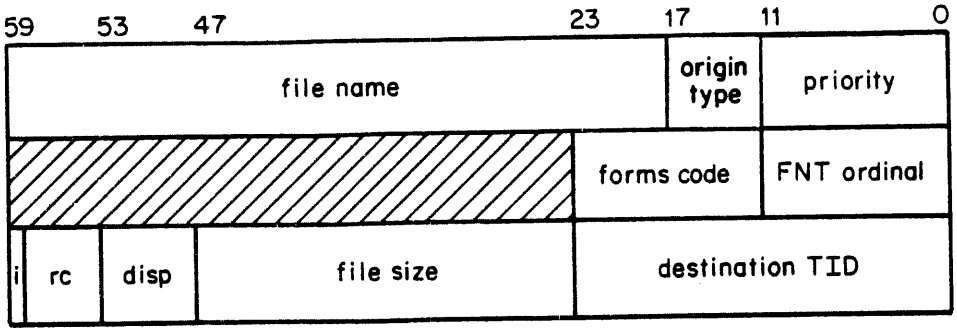
10 Job rolled out

02 Job at control point

The response returned if an input file satisfies the search criteria has the following format.



The response returned if a print/punch file satisfies the search criteria has the following format.



i File interrupted (if set = 1)

rc Repeat count

disp Disposition code (this field contains internal and external characteristics and device code)

If the extended information bit is set in the call block, the forms code, interrupt flag, repeat count, file size, and disposition code (internal/external characteristics) are returned.

Function 3 - COUNT

The COUNT function counts the number of files and/or jobs of a specified type satisfying the selection criteria. Multiple queue types can be set in a single COUNT request with the count of each type being returned to the caller. The required parameters for the COUNT call include the following.

- Function code (=3) and cleared completion bit
- Queue type
- Priority
- Origin type

The terminal ID, starting FNT ordinal, disposition code, and file/job name may be specified as optional search criteria.

The queue type, disposition code, FNT address, priority and file/job name are also returned for the first file that meets the selection criteria.

The code for the COUNT function is also found in overlay 3QS.

QAC - KEY RESIDENT SUBROUTINES

The following paragraphs describe key resident QAC subroutines.

SEJ - Search for Executing Job

SEJ scans the FNT for queued files which represent jobs in an executing state, namely INFT (at a control point), ROFT (rolled out), or TEFT (timed/event rolled out). If one of these types is found, subroutine VCI is called to compare the selection criteria with information about the job that is normally found in the control point area.

If a job has been selected, its FNT entry is interlocked via the FNT interlock mechanism (SFIM). The queue type is revalidated and VCI called again to recheck the selection criteria since information about the job could have changed while waiting for the FNT entry to be interlocked or it may be a different job. If the recheck rejects the job, the interlock is cleared and the next FNT entry is examined.

If the recheck verifies the job as satisfying the selection criteria, subroutine VMI is called to compare the selection criteria with information about the job that is normally found in the system sector or FNT/FST for it. If this comparison is successful, a job found status is returned to the caller.

SFF - Search For File

SFF scans the FNT for queued files having the desired selection criteria. The logic is similar to that found in SEJ except that the information processed is related to files rather than jobs. VCI is called to compare selection criteria that is control point related. If a candidate is found, it is interlocked and VCI called to recheck the selection. If the recheck rejects the file, the interlock is cleared. If the recheck is successful, VMI is called to compare selection criteria that is FNT/FST or system sector resident. If this comparison is successful, the FNT address of the file is returned to the caller.

VCI - Validate Central Memory Information

VCI compares selection criteria with information normally found in the control point area. The processing done by VCI is dependent upon the type of queue selection (job vs file) and check or recheck.

In the queued file check and recheck case (call is from SFF), the file must not be assigned to a control point, must match the desired queue type, must match the desired priority (if this is a selection criterion), and must match the job name and origin type (if these are selection criteria).

In the executing job case of a job at a control point (file type is INFT), the input file must be assigned to a control point (that is, not in the queue), must be the job's input file, and must match the job name and origin type (if these are selection criteria). In the recheck case, the control point area is read into CBUF. and TFSW is checked to be sure that this is the job's input file. The job name, origin type, and sequence number are read from JNMW and RFCW for use if QAC must move to the control point. The job name and origin must match the selection criteria job name/origin if these are requested. The read rollout file is not set as control point data has already been read into CBUF.

In the case of rollout files, whether ROFT or TEFT, the read rollout file flag is set. The job name and origin must agree with the selection criteria if so specified.

VCI is flowcharted as figure 34-4.

VMI - Validate Mass Storage Information

VMI compares the selection criteria with information contained in the file's FNT/FST or system sector.

For input files (INFT) or executing job files (ROFT), the file's system sector is read and destination TID is compared (DISS through DISS+1) with the selection criteria. If no match occurs, the not found status is returned. If matching and not remote batch origin (EIOT), a found status is returned. If EIOT, family names are compared (FDSS through FDSS+3). If family names match, a found status is returned with not found returned otherwise. A check is then made for queue type and special output.

For all other queued file types, the desired terminal IDs is compared with the FST and a not found status returned if they do not agree. The forms code is then compared.

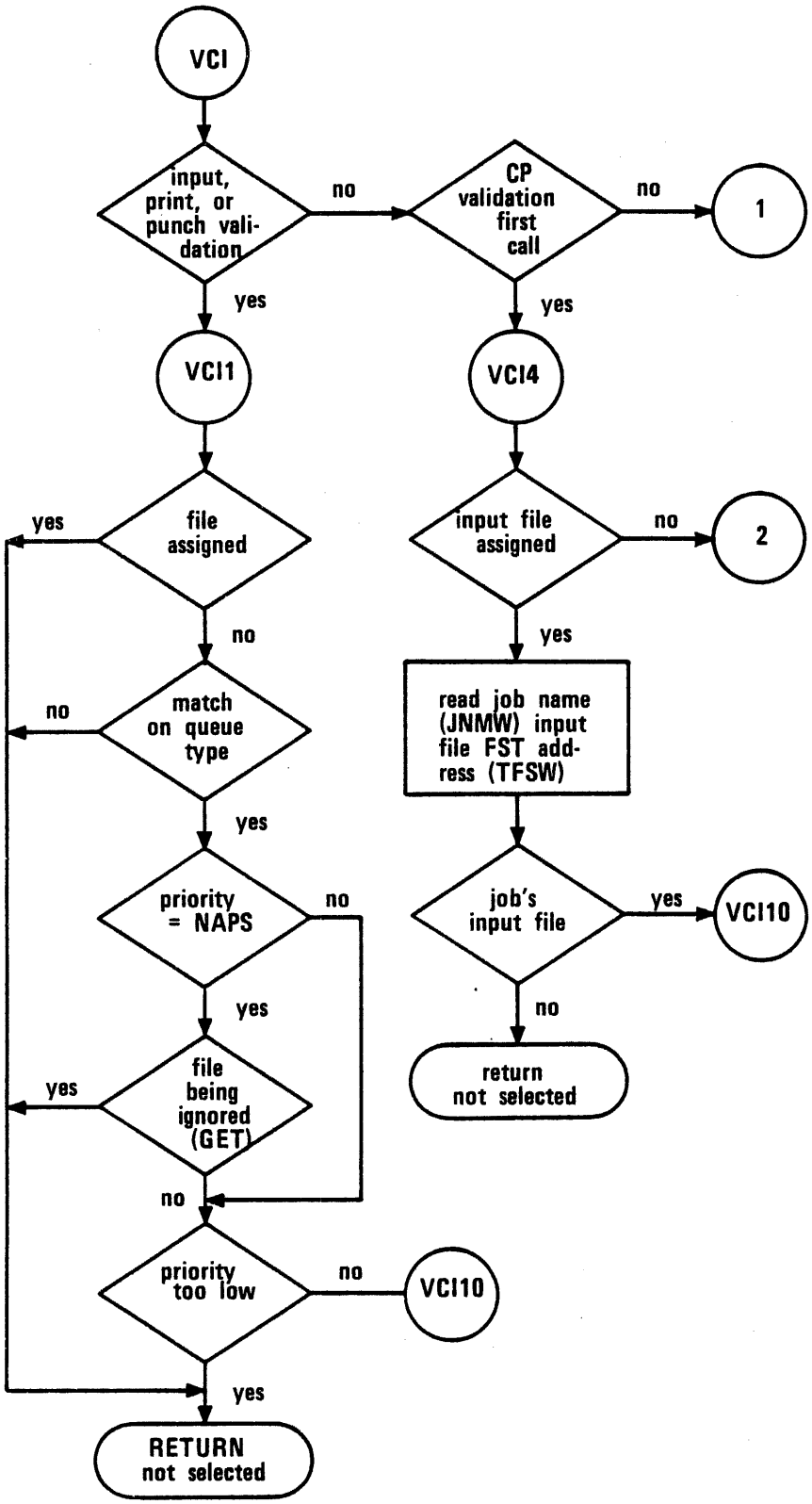


Figure 34-4. VCI - Validate Control Point Information

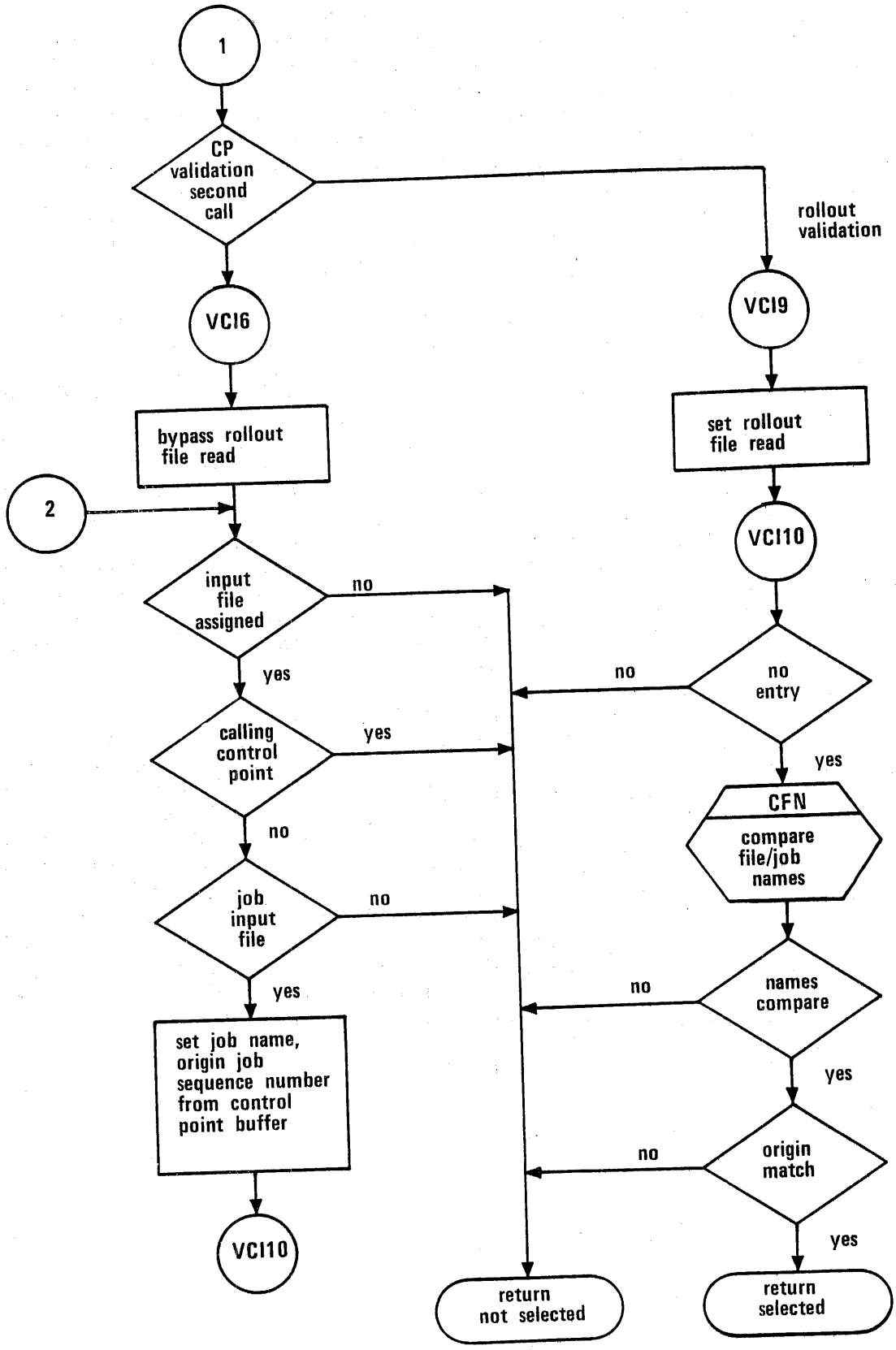


Figure 34-4. VCI - Validate Control Point Information (Continued)

If forms code is not a selection criterion, the device code is checked. If the file is a special queued file (d/x fields equal SPDV), the forms code is compared with the system sector. If the file is not a special queued file, the forms code from the FST is compared with the selection criterion. If a match occurs, the device code is checked. If a match does not occur and the forms code from the FST is not an extended code, a not found status is returned. If the forms code is an extended code, the system sector is read, and the forms code (FCSS) is compared with the selection criterion. If no match occurs, a not found status is returned. If matching the device code check is made.

If device code is not a selection criterion, and the file is not EIOT, a found status is returned. A found status is also returned for EIOT files with matching family names. If the file is a special queued file (file ID is SPDV) and extended codes are available, the device code validation is done using the system sector. If not a special queued file, and any device can be used, the external characteristics are checked. If a device code is specified, the device code from the FST is compared with the selection criterion. If a match occurs, the external characteristics are checked. If a match does not occur, not found status is returned. If the comparison for device code is to be made from the system sector, the sector is read and the device code (DVSS) compared with the selection criterion. If no match occurs, a not found status is returned. If matching, the external characteristics are checked.

If the file is a special queued file (greater than SPDV), the external characteristics are processed from the system sector. If the external characteristic are extended values, they are also processed from the system sector. If external characteristics are specified in the FST and do not match the selection criterion, a not found status is returned. If any printer can be used, and not EIOT, a found status is returned; if EIOT and matching family names, found is also returned. If the characteristics must be processed from the system sector, the system sector is read and the external characteristics (ECSS) are compared to the selection criterion. If no match occurs, a not found status is returned. If matching, the EIOT/family check must agree before the found status is returned.

VMI is flowcharted as figure 34-5.

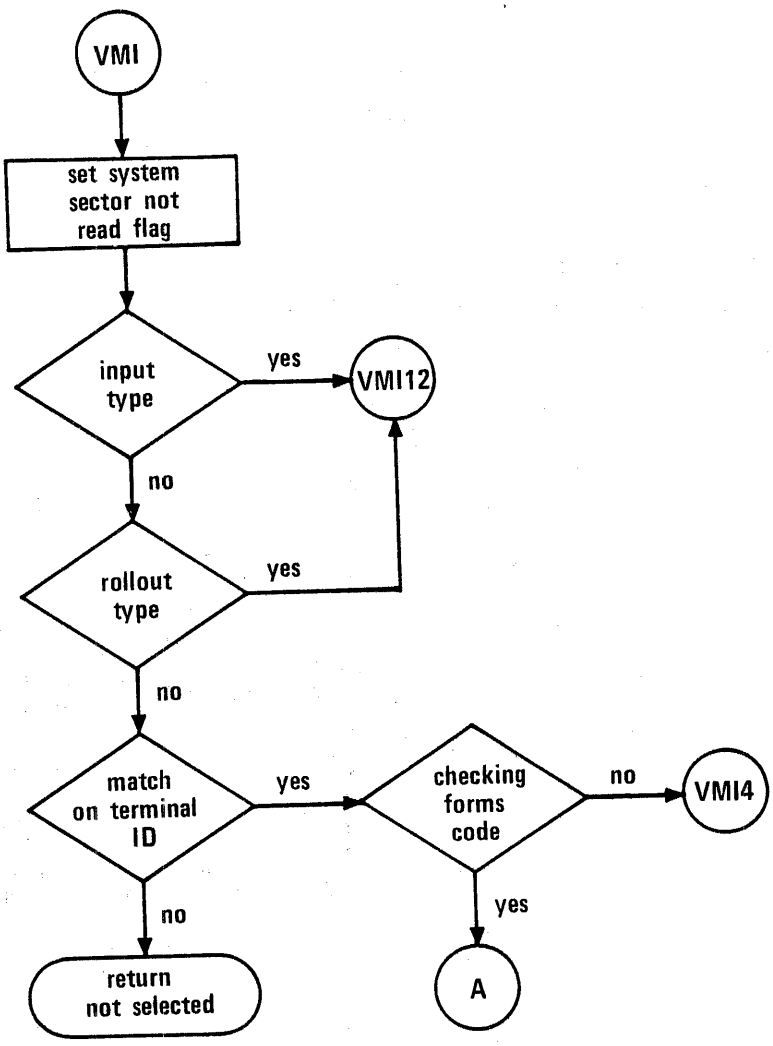


Figure 34-5. VMI - Validate Mass Storage Information

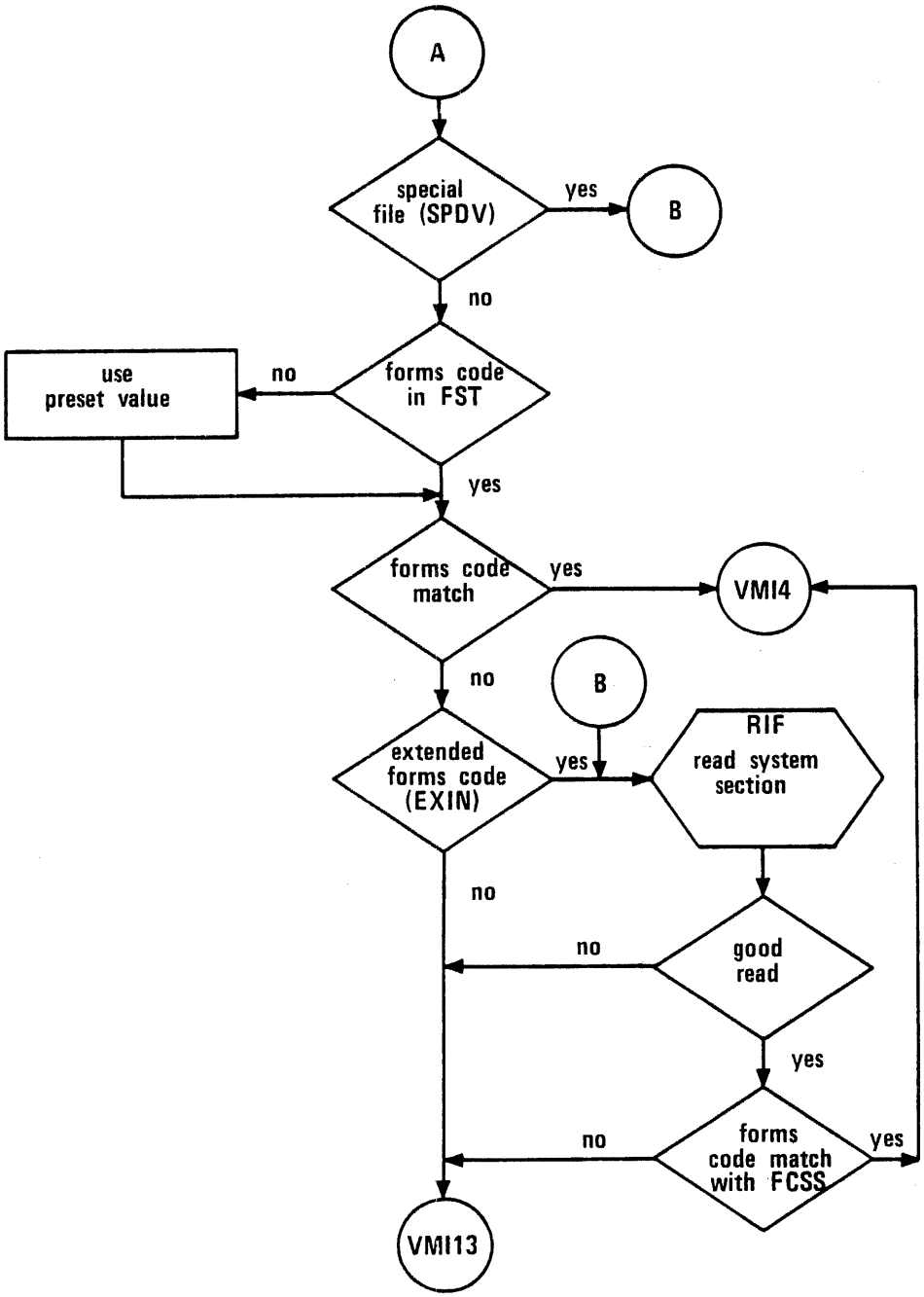
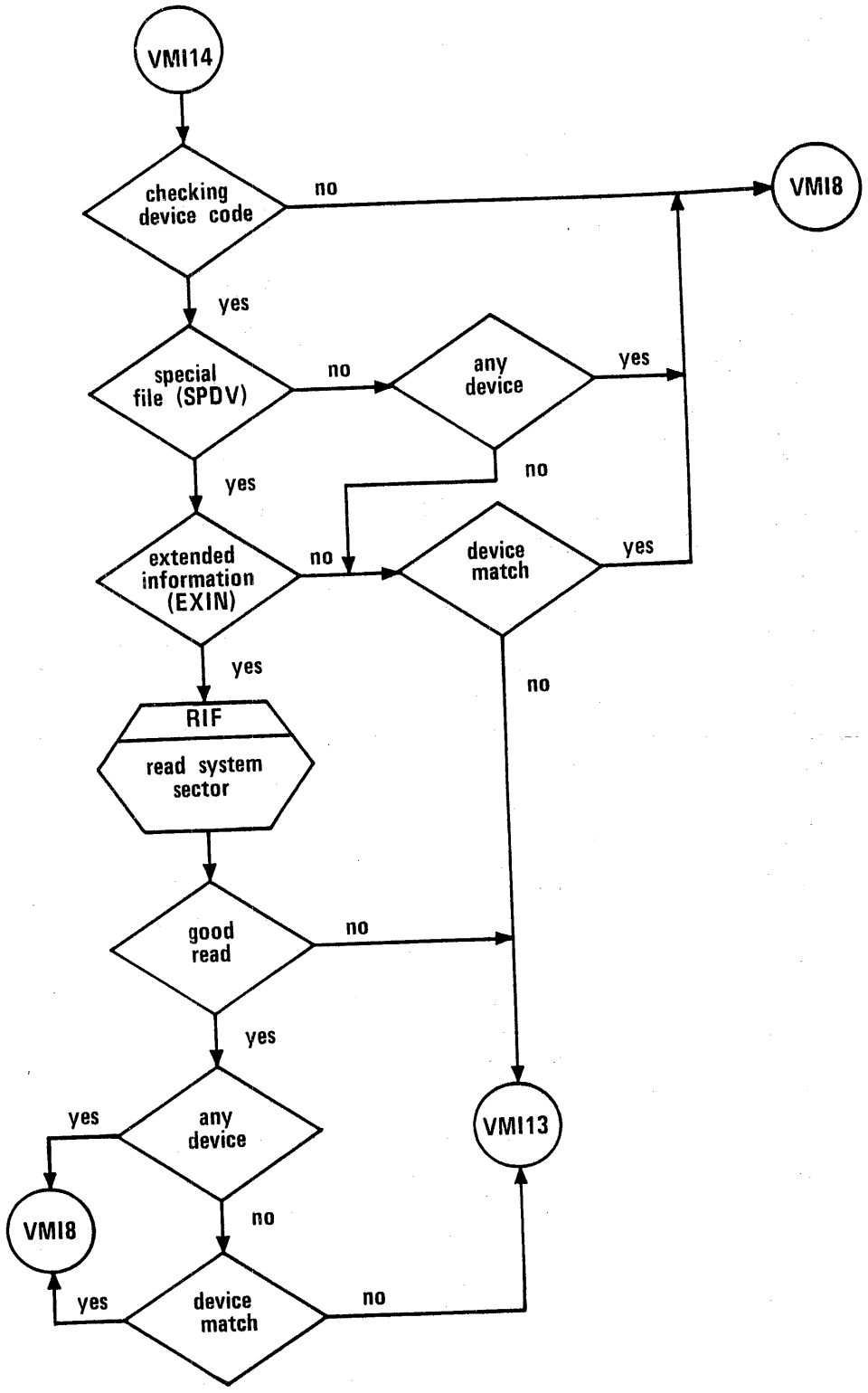


Figure 34-5. VMI - Validate Mass Storage Information (Continued)



*1 Input conditions for VMI13:
0 = match, 1 = no match or error

Figure 34-5. VMI - Validate Mass Storage Information (Continued)

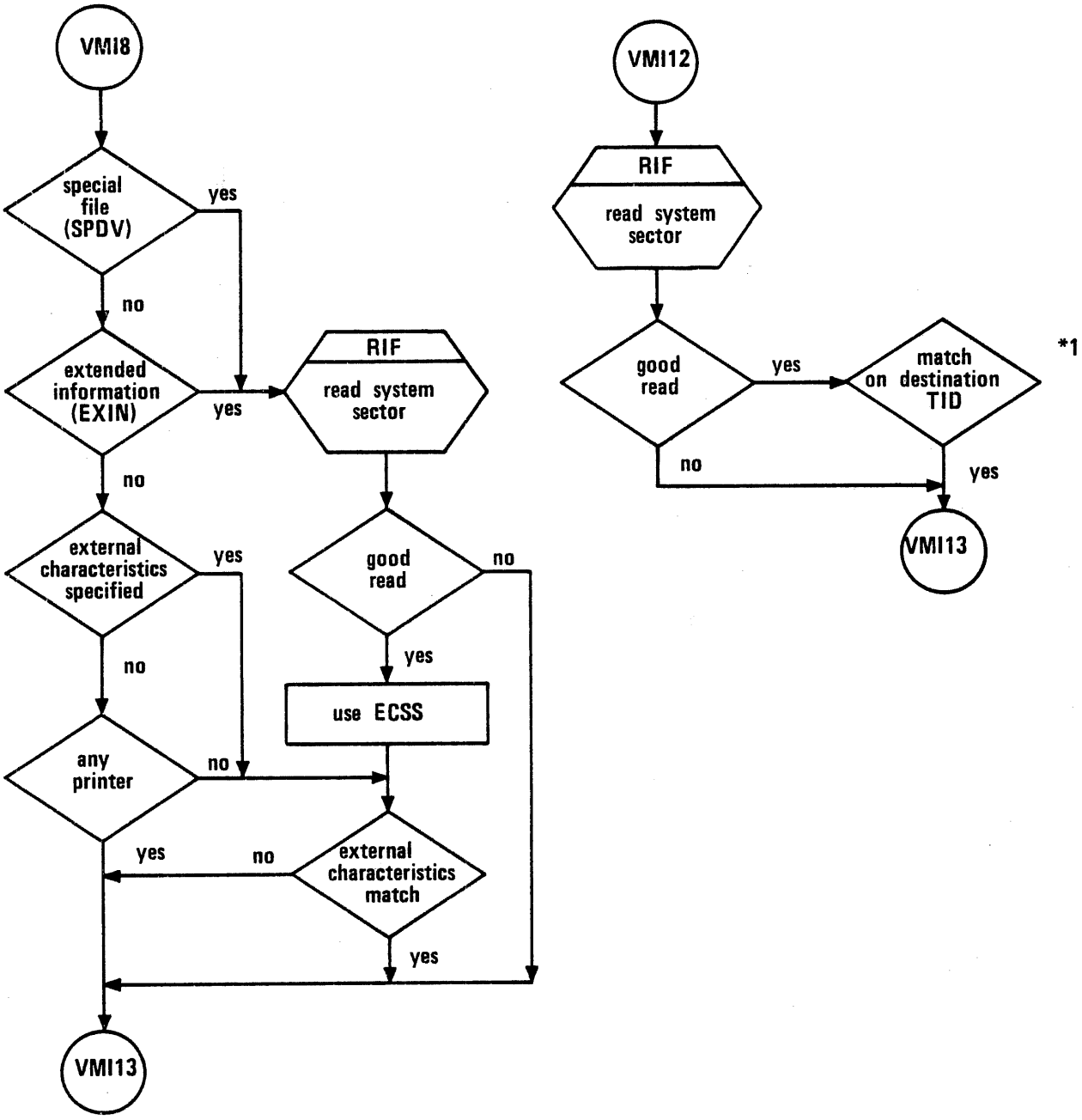


Figure 34-5. VMI - Validate Mass Storage Information (Continued)

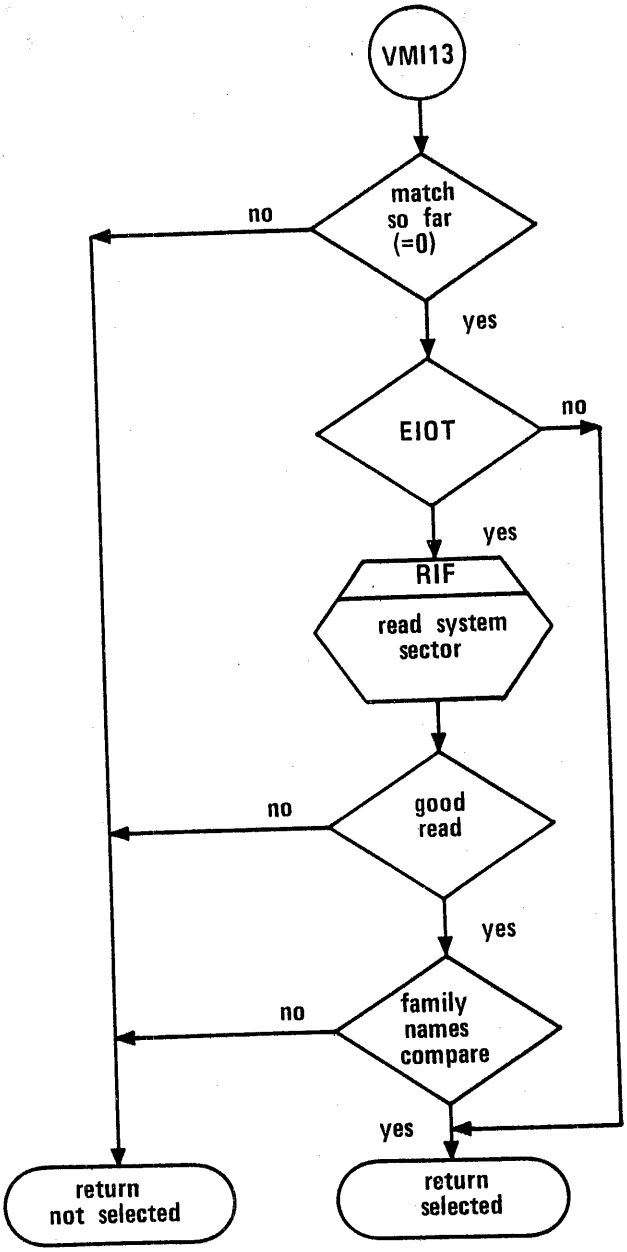


Figure 34-5. VMI - Validate Mass Storage Information (Continued)

In summary, for output files with matching terminal IDs, compare forms code, device code, and external characteristics (if selection criteria) with FST if not a SPDV file. If SPDV or forms code, device code, and external characteristics are extended, compare the selection criteria to their system sector values. If no match occurs, return not found. If a match occurs and file is not EIOT, return found. If file is EIOT, return found if family names compare; not found if otherwise.

REPRIEVE OVERVIEW

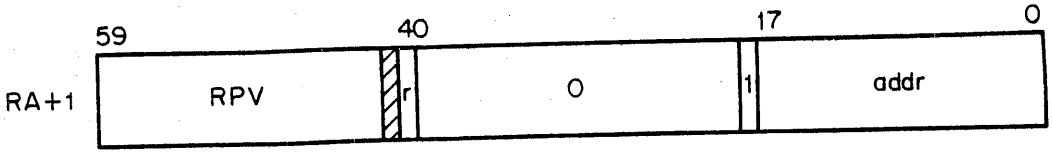
Reprieve (RPV) allows a CPU program to perform the following functions.

- Declare an address to which control is returned after execution has been stopped by an error, a terminal interrupt, or normal job termination. These cases are called interrupts.
- Resume execution of an interrupted program at the point of interrupt.
- Reset an error flag which resulted in a reprieve so that job exit processing can occur.

Reprieve processing is controlled by calls to the PP program RPV. RPV consists of two functions which previously resided in SFP and three extended functions: setup, resume and reset. Extended reprieve combines the features of EREXIT, DISTC, and standard reprieve, with additional capabilities of resuming execution after an interrupt and queuing of pending interrupts. The following discussion concerns extended reprieve only.

RA+1 CALL

The format of the call to RPV is as follows.



addr First word address of the parameter block

REPRIEVE FUNCTIONS

RPV supports the following functions.

- Setup
- Resume
- Reset

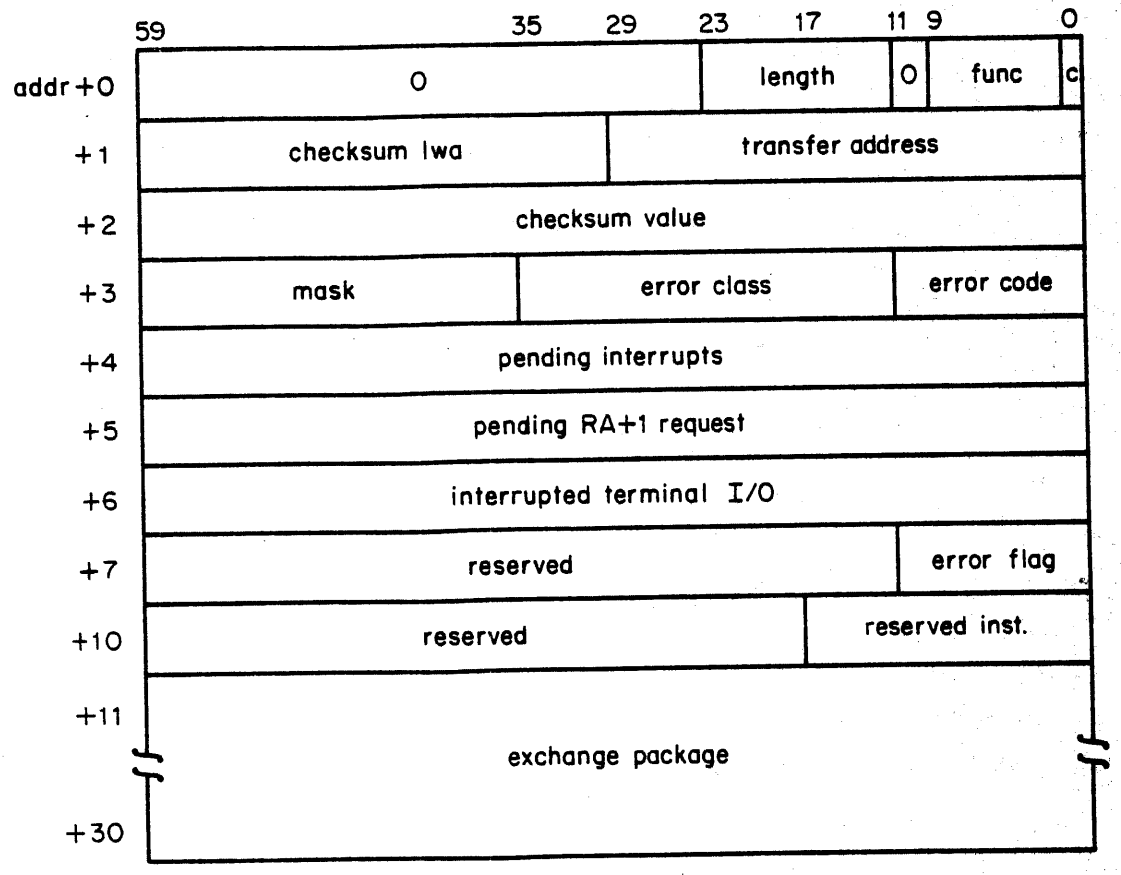
The setup function is used to initialize the parameters for the RPV interface (that is, setting the mask bits that indicate which errors are to be intercepted). It also initializes all RPV data in the user's control point area and if the parameter block indicates pending interrupts or I/O requests, these are processed at that time.

The resume function is used to restart the program after an interrupt has been processed. Any pending interrupts that occurred during the processing of the previous interrupt are detected then and the interrupt handler is restarted to process these interrupts. Optionally, the mask bits may also be changed during the resume function.

The reset function is used to reset a previous error and allow the operating system error handling to process the error. That is, the error is processed by the system as if the appropriate mask bit had not been set. Pending interrupts are not processed; however, any pending RA+1 request is reset.

PARAMETER BLOCK

The format of the parameter block is as follows.



length — Length of the parameter block including the exchange package area [minimum of 25 (31B) words].

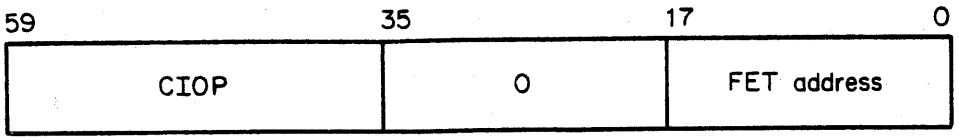
func — Function code:

1 Setup
2 Resume
3 Reset

c	Completion bit (set when operation is complete).
checksum lwa	Specified by the user to indicate the end of the area to be checksummed and compared or set. If zero, no checksum is desired (checksum area begins at transfer address).
transfer address	Address to which control is transferred when an interrupt is processed.
checksum value	Either set to the checksum of the indicated area when RPV is called or compared against the computed checksum (if checksum lwa is specified) when a reprievable error is processed.
mask	Mask bits to be set by call (specifies class of interrupts to be intercepted):

mask	Description
001	CPU error exit
002	PP call error
004	Resource limit
010	Operator termination
020	PP abort
040	CPU abort
100	Normal termination
200	Terminal interrupt

error class	Set to the value of the mask bit which intercepts the indicated error (that is, if error x is intercepted by mask bit n, then bit n in the error class field is set). Refer to table 35-1 for a list of error classes.
error code	Octal code indicating error encountered. Refer to table 35-1 for a list of error codes and their meanings.
pending interrupt	Used to queue pending interrupts (that is, the nth error code sets bit n in this field).
pending RA+1 request	Contents of RA+1 at time of interrupt. RA+1 reset from this field on a resume or reset call.
interrupted terminal I/O	Contains interrupted input request if an interrupt occurs while a terminal input request is pending. The format is as follows.



The CIO call is reissued on a resume call.

error flag — Value of the operating system error flag at the time of the interrupt (refer to section 2 for a list of error flags).

reserved inst. — This area is reserved for use by the installation.

exchange package — A copy of the exchange package at the time of the interrupt (unchanged from the executing package at the time of the error). This is the exchange package that is used when the interrupt handler is started. A reset or resume call sets the running control point registers from this area.

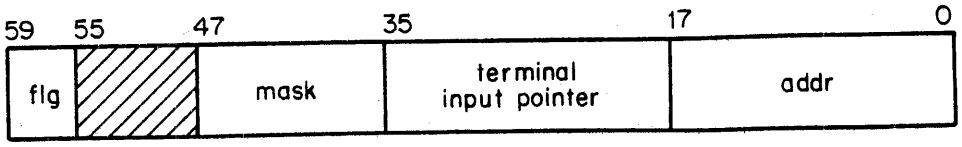
Table 35-1 lists the return information for RPV error codes, classes, and flags. The operator kill error (OKET) is only reprieved once per job. The first time it is handled as any other error; however, the second occurrence unconditionally terminates the job. Additionally, reprieve processing is not allowed on a system abort error (SYET) or override error (ORET) for security reasons.

TABLE 35-1. RPV ERROR CODES, CLASSES, FLAGS

error code	Description	error class	Corresponding Error Flags
0	Normal termination	100	None
1	Time limit	004	TLET
2	CPU error exit	001	ARET, PSET
3	PP abort	020	PPET
4	CPU abort	040	CPET
5	PP call error	002	PCET
6	Operator drop	010	ODET
7	Operator kill	010	SSET, OKET, ORET, SYET, FSET
10	Operator rerun	010	RRET
11	Control statement error	040	None
12	ECS parity error	020	ECET
15	Autorecall error	002	None
16	Job hung in autorecall	002	None
17	Mass storage limit	004	FLET, TKET
20	PP program not in library	002	None
21	I/O limits	004	SRET
40	Terminal interrupt	200	TIET

CONTROL POINT AREA USE

Word EECW of the user's control point area is formatted as follows for extended reprieve processing.



flg Flags; each bit defined as follows:

Bit	Description
59	Not used by extended RPV
58	Extended RPV active
57	Interrupt handler in progress
56	One-time error entered

mask	Mask bits as described in parameter block
addr	RPV parameter block address

Bit 58 is set on a setup or resume call. When extended reprieve is active, bits 35 through 23 of TIAW (otherwise the DISTC address) contain the previous error flag value. Bit 57 is set whenever the system (1AJ or 1RI) starts the interrupt handler at the transfer address. If a subsequent interrupt occurs while this bit is set, the interrupt is set pending in the parameter block and processing of the first interrupt is allowed to continue. An exception is if the interrupt was an error initiated by the interrupt handler (such as an arithmetic error). In this case the interrupt is not set pending and the job is aborted. Bit 56 is set when a one-time error is initially reprieved (currently only OKET). The second occurrence of a one-time error is not reprieved even if selected by the mask bits. The mask bits and parameter block address are set in EECW by a setup or resume call.

SETUP FUNCTION

This function is used to initialize the parameters for the RPV interface. It sets the mask bits which indicate which errors are to be intercepted and initializes all RPV data in the control point area. If the parameter block indicates pending interrupts or I/O requests, these are processed at this time.

The setup reprieve function is processed in the following manner.

1. Abort if one of the following parameter errors.
    - Length less than 31B words
    - FWA (addr) plus length greater than FL
    - Checksum LWA nonzero and less than transfer address
    - Checksum LWA less than 2 or greater than or equal to FL
    - Transfer address less than 2 or greater than or equal to FL
    - Undefined mask bit specified
    - RPV called without autorecall

2. The following values are set in the control point word EECW.

   - Extended RPV flag
   - RPV mask
   - FWA of parameter block
   - Previous error flag is invalidated

3. If pending interrupts are nonzero, perform the following.

   - Validate pending interrupts (that is, is pending interrupt of defined error)
   - Select highest priority interrupt
   - Clear selected interrupt bit in pending interrupts
   - Set return status flags (error number, error class, and so on)
   - Set interrupt handler active
   - Set running P register to transfer address
   - Reset running registers from the parameter block (except P)
   - Set complete bit
   - Return

4. If pending interrupt equals zero, perform the following.

   - Compute checksum if LWA nonzero
   - Clear interrupt handler in progress
   - Reset RA+1 from parameter block
   - Clear parameter block fields used (pending RA+1 request word, interrupted terminal I/O word, and so on)
   - Set complete bit
   - Request CPU
   - Initiate pending terminal input (call CIO over RPV)

RESUME FUNCTION

This function is used to restart the running program after an interrupt has been processed. Any pending interrupts which occurred during the processing of the initial interrupt are detected at this time and the interrupt handler is restarted to process these interrupts. Optionally, the mask bits can also be changed at this time. The processing of RESUME is identical to that of SETUP except that the running registers are set from the parameter block.

The resume reprieve function is processed in the following manner.

1. Abort if parameter error.
    - Length of parameter block less than 31B
    - Parameter block outside of FL
    - Checksum LWA nonzero and less than transfer address
    - Checksum LWA less than 2 or greater than or equal to FL
    - Transfer address less than 2 or greater than or equal to FL
    - Undefined mask bit specified
    - RPV called without autorecall

2. Set control point values in EECW.
    - Set new mask bits
    - Set FWA of RPV area
    - Set extended RPV flag
    - Invalidate previous error flag

3. If pending interrupts nonzero.
    - Validate pending interrupts (that is, pending interrupt is a defined error)
    - Select highest priority interrupt
    - Clear selected interrupt bit in pending interrupts
    - Reset running exchange package (except P)

- Set return status flags (error number, error class, and so on)
- Set interrupt handler active
- Set running P register to transfer address
- Set complete bit
- Return

4. If pending interrupt zero.
    - Compute checksum if LWA nonzero
    - Clear interrupt handler in progress
    - Reset running exchange package (including P) and RA+1 from parameter block
    - Clear parameter block fields used (pending RA+1 request word, interrupted terminal I/O word, and so on)
    - Set complete bit
    - Initiate pending terminal input (call CIO over RPV)

RESET FUNCTION

This function is used to reset a previous error and allow the operating system error handling to process the error. Pending interrupts are not processed; however, any pending RA+1 is reset. A reset causes the same action as would have occurred if the appropriate mask bit had not been set (resumed errors cannot be reset since there is no way to validate the information coming from the users FL).

The reset reprieve function is processed in the following manner.

1. Abort if parameter error.
    - RPV extended mode not set
    - Length of parameter block less than 31B
    - Parameter block outside of FL
    - Transfer address less than 2 or greater than or equal to FL
    - Undefined mask bit specified

- RPV called without autorecall
- Interrupt handler not active
- Previous error flag not set (invalid)

2. Reset error flag to previous error flag (from TIAW).

3. Reset previous job.
   - Clear mask bits (turn off further RPV processing)
   - Clear interrupt handler in progress
   - Reset running exchange package (including P) and RA+1 from parameter block
   - Set complete bit

INTERRUPT PROCESSING FOR EXTENDED RPV

The following is the processing taken by the system (1AJ or 1RI) when an interrupt occurs. Error flags and normal termination are processed by 1AJ. Terminal interrupts are processed by 1RI.

1. Abort job if mask bit for error not set.

2. If error flag has undefined mapping, hang.

3. Abort if parameter error.
   - RPV extended mode not set
   - Length of parameter block less than 31B
   - Parameter block outside of FL
   - Checksum LWA nonzero and less than transfer address
   - Checksum LWA less than 2 or greater than or equal to FL
   - Transfer address less than 2 or greater than or equal to FL

4. If interrupt handler not active.

   - Abort if checksum mismatch
   - Move error flag to previous error flag
   - Copy exchange package to user return package
   - Set system dependent and common error codes
   - Copy contents of RA+1
   - Clear RA+1
   - Reconstruct terminal input request (if present); if no request present, store zero
   - Set running P register to transfer address
   - Clear error flag
   - Set interrupt handler in progress
   - Return

5. If interrupt handler active.

   - Abort if execution initiated interrupt (error codes 0, 2, 3, 4, 5, 11, 15, 16, 20)
   - Set pending error in pending interrupt status word in request block
   - Clear error flag
   - Return

TERMINAL INPUT REQUESTED

If an interrupt occurs while a request for terminal input is pending, control is given to the interrupt handler in such a way that the I/O request can be reissued after interrupt processing has been completed. The mechanism for accomplishing this is as follows.

1. Do not set the FET complete.
2. Set no error code in the FET.
3. No data is passed to the buffer (IN/OUT remain unchanged).
4. Reconstruct the CIO request which initiated the input request and pass it to the interrupt handler in the interrupted terminal input request word of the return package; this word is zero if the program was not waiting for terminal input

INTERRUPT FLOW

Figure 35-1 describes the flow of control during the processing of an interrupt. The module performing a function is enclosed in parentheses.

Figure 35-2 is a flowchart illustrating 1AJ interrupt processing. Figure 35-3 shows the processing by 1RO when a terminal job is to be returned to time-sharing executive control following a terminal interrupt. Figure 35-4 flowcharts 1RI processing when a terminal job is given control back following a terminal interrupt.

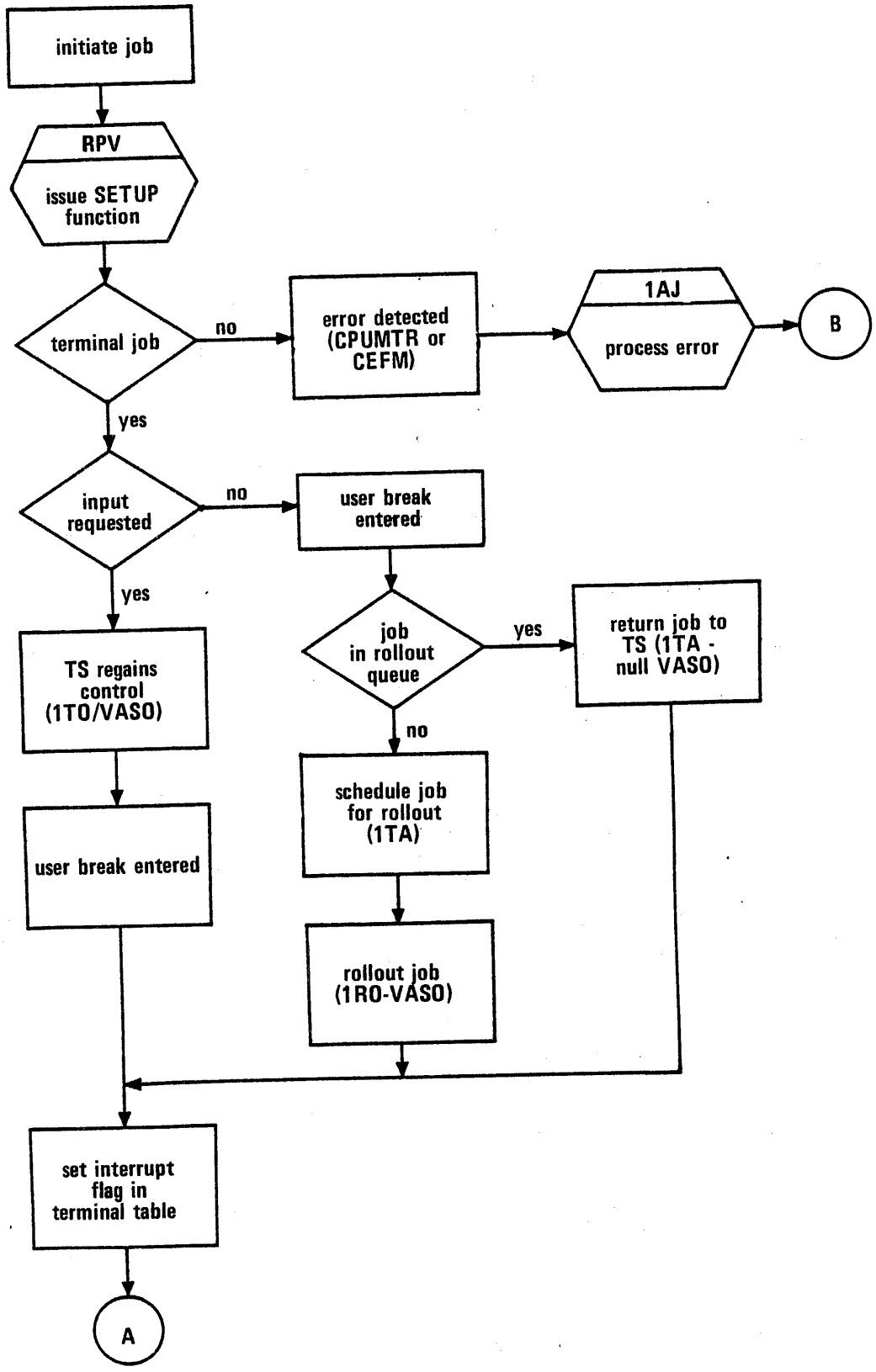


Figure 35-1. Interrupt Processing

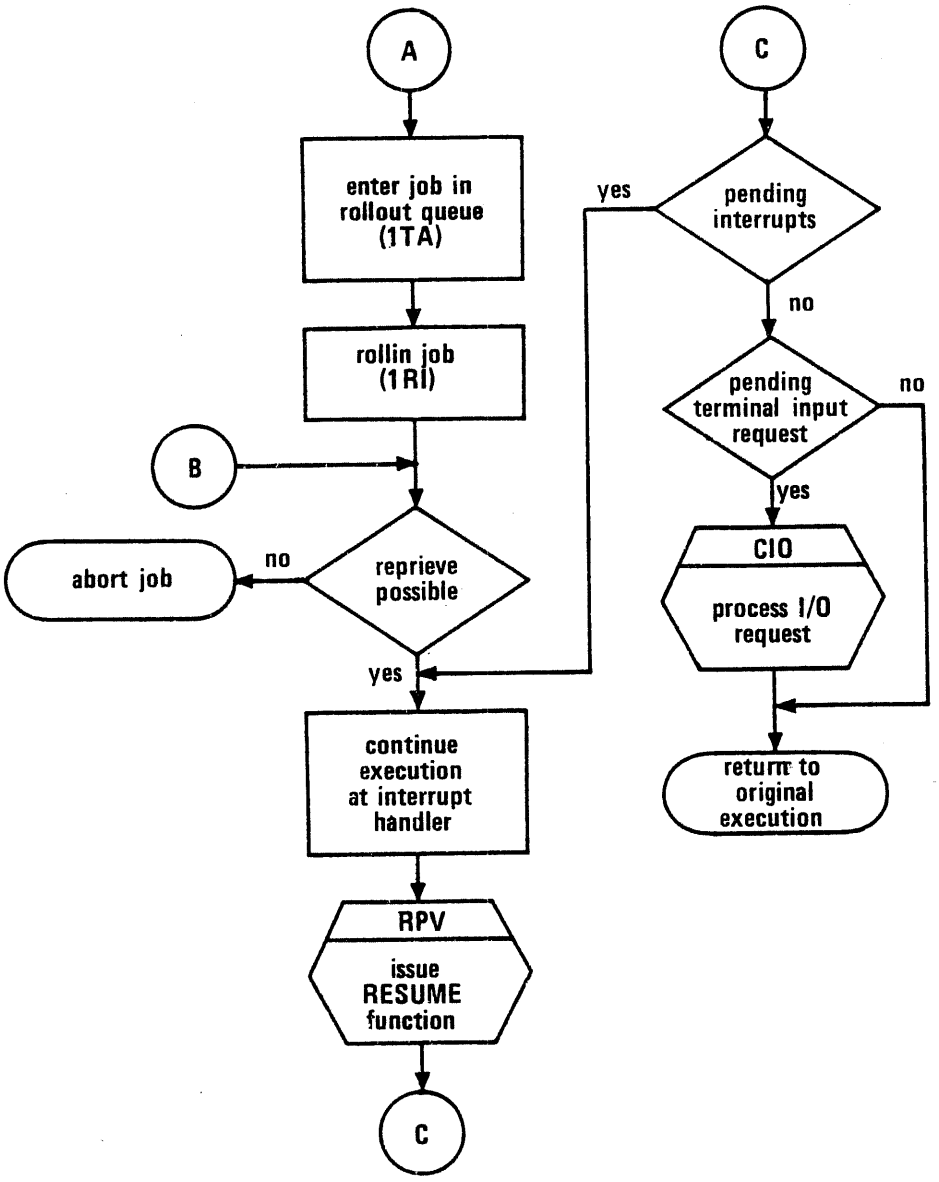
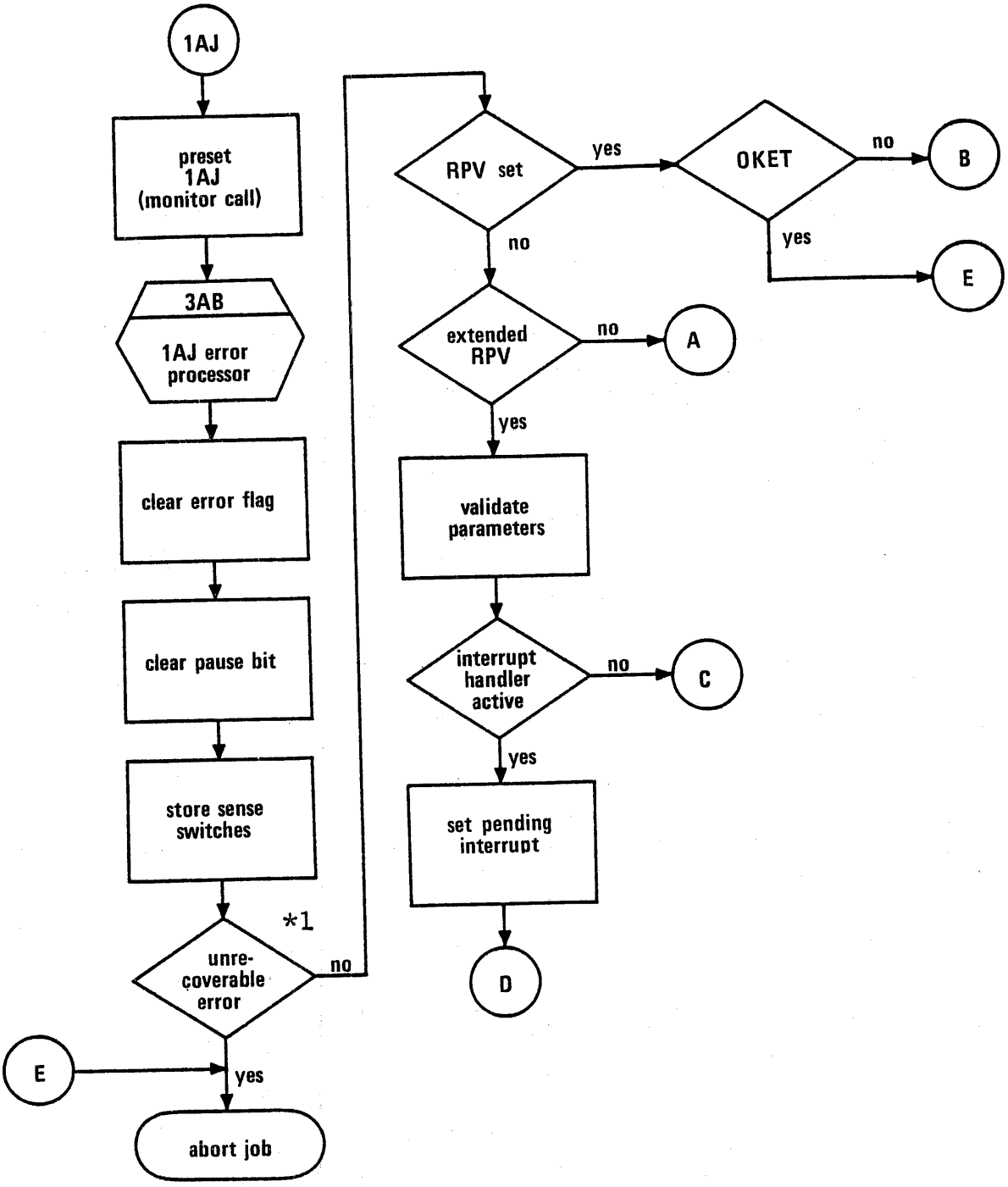


Figure 35-1. Interrupt Processing (Continued)



*1 SYET, ORET, second OKET, or fatal mainframe error.

Figure 35-2. 1AJ Interrupt Processing

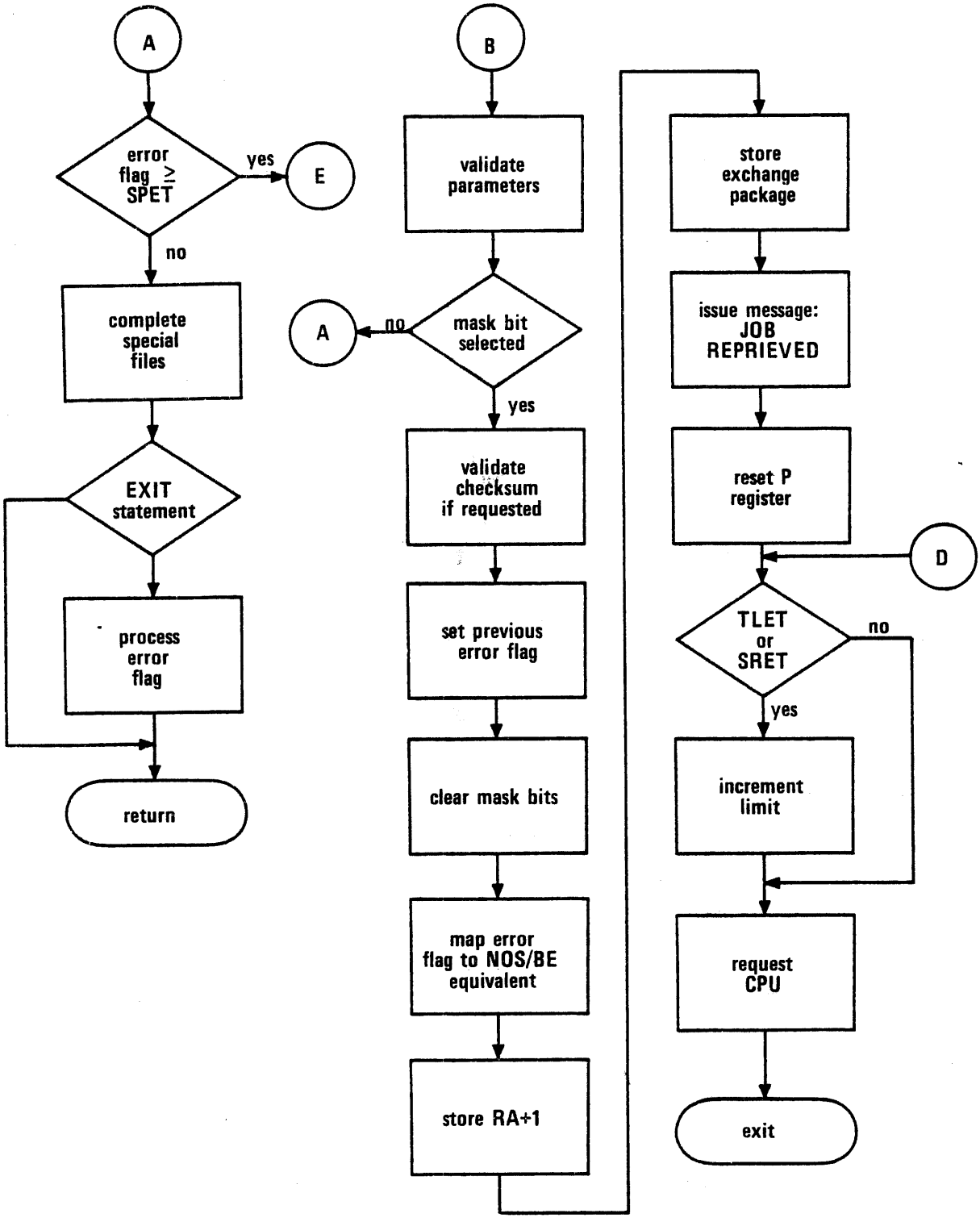


Figure 35-2. 1AJ Interrupt Processing (Continued)

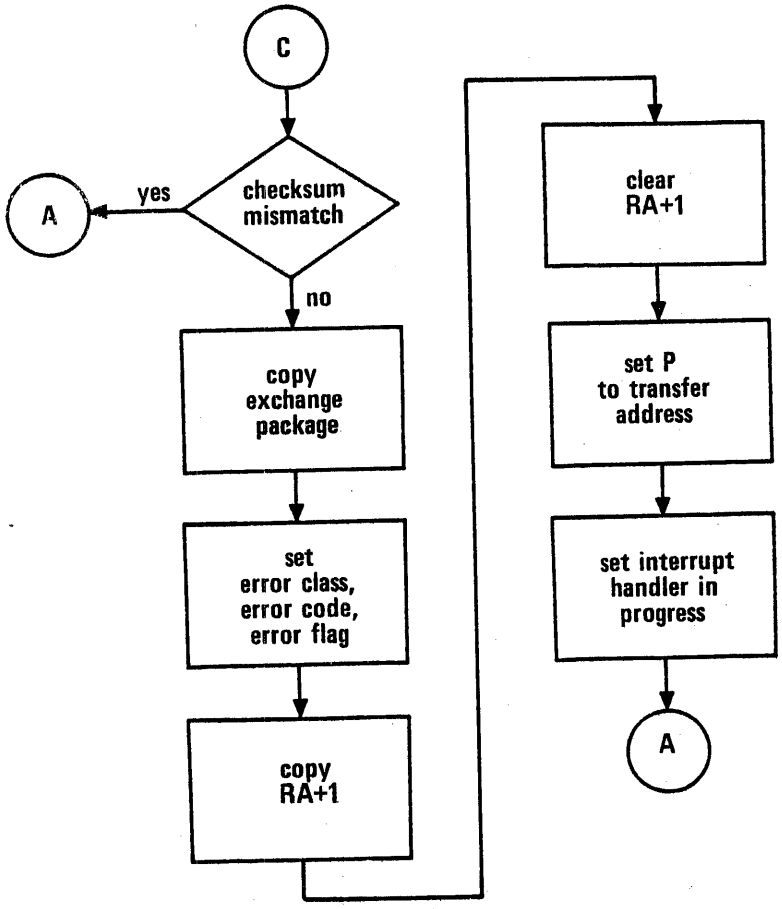


Figure 35-2. 1AJ Interrupt Processing (Continued)

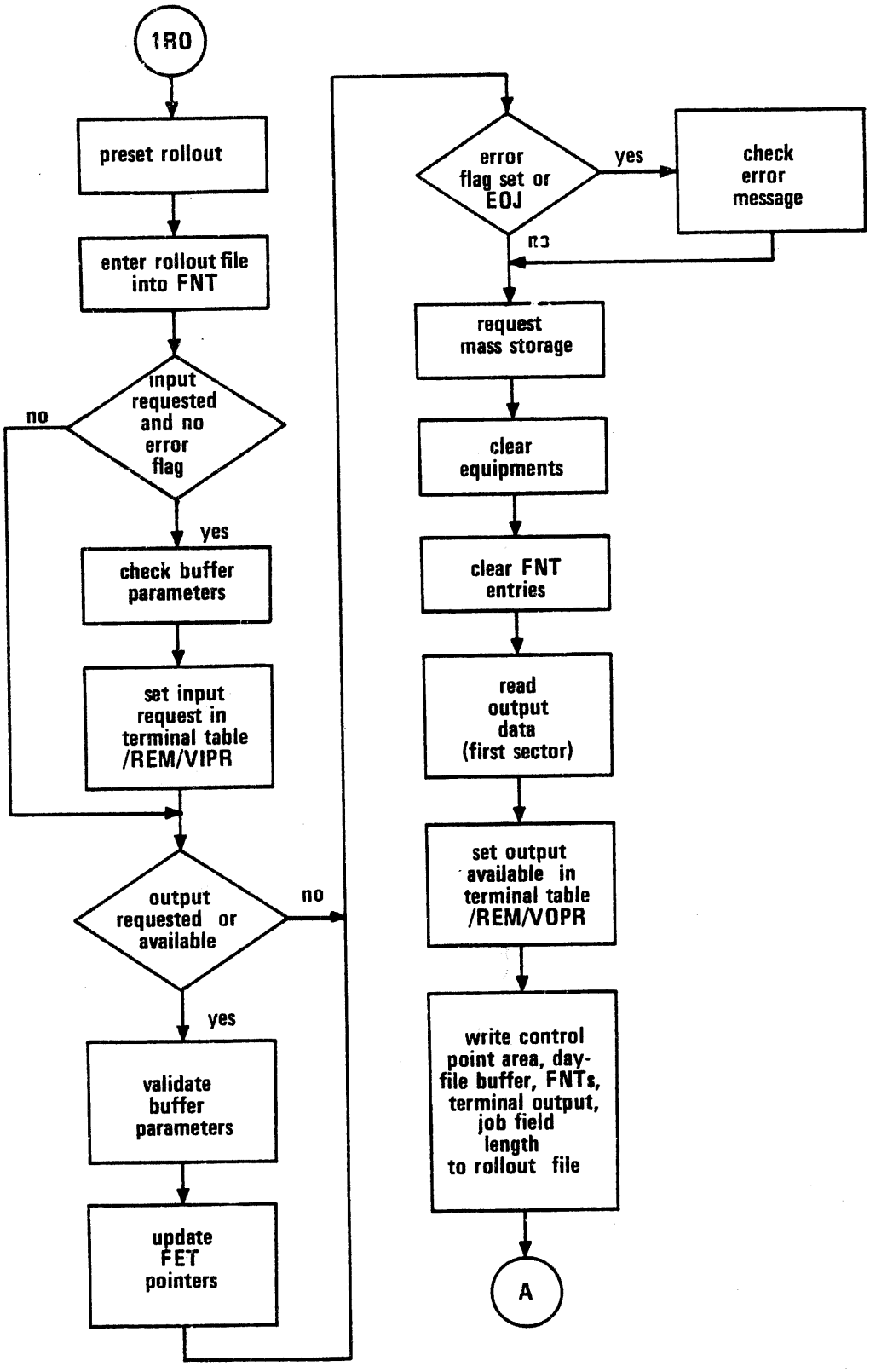
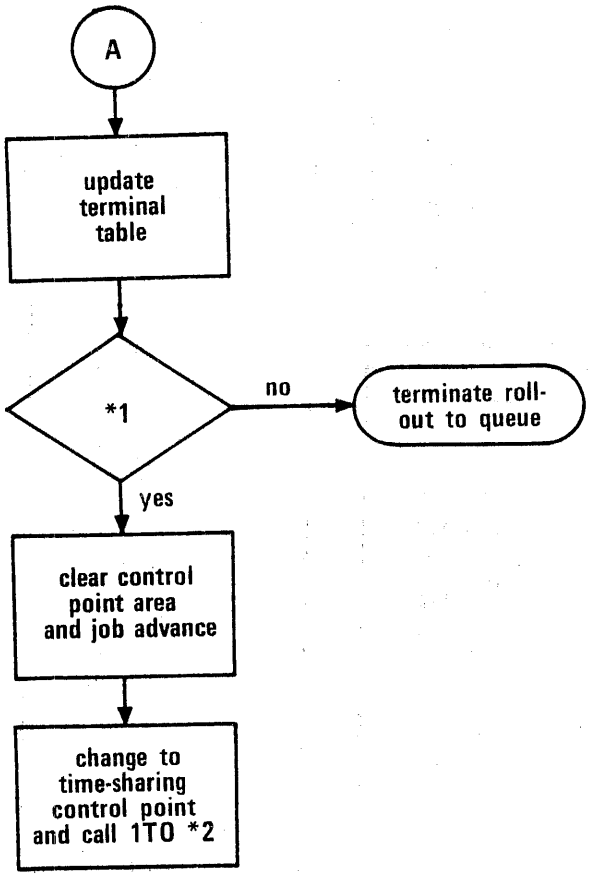


Figure 35-3. 1RO Interrupt Processing



*1 Input requested, output available, job termination or forced rollout.

*2 Refer to figure 15-22.

Figure 35-3. 1RO Interrupt Processing (Continued)

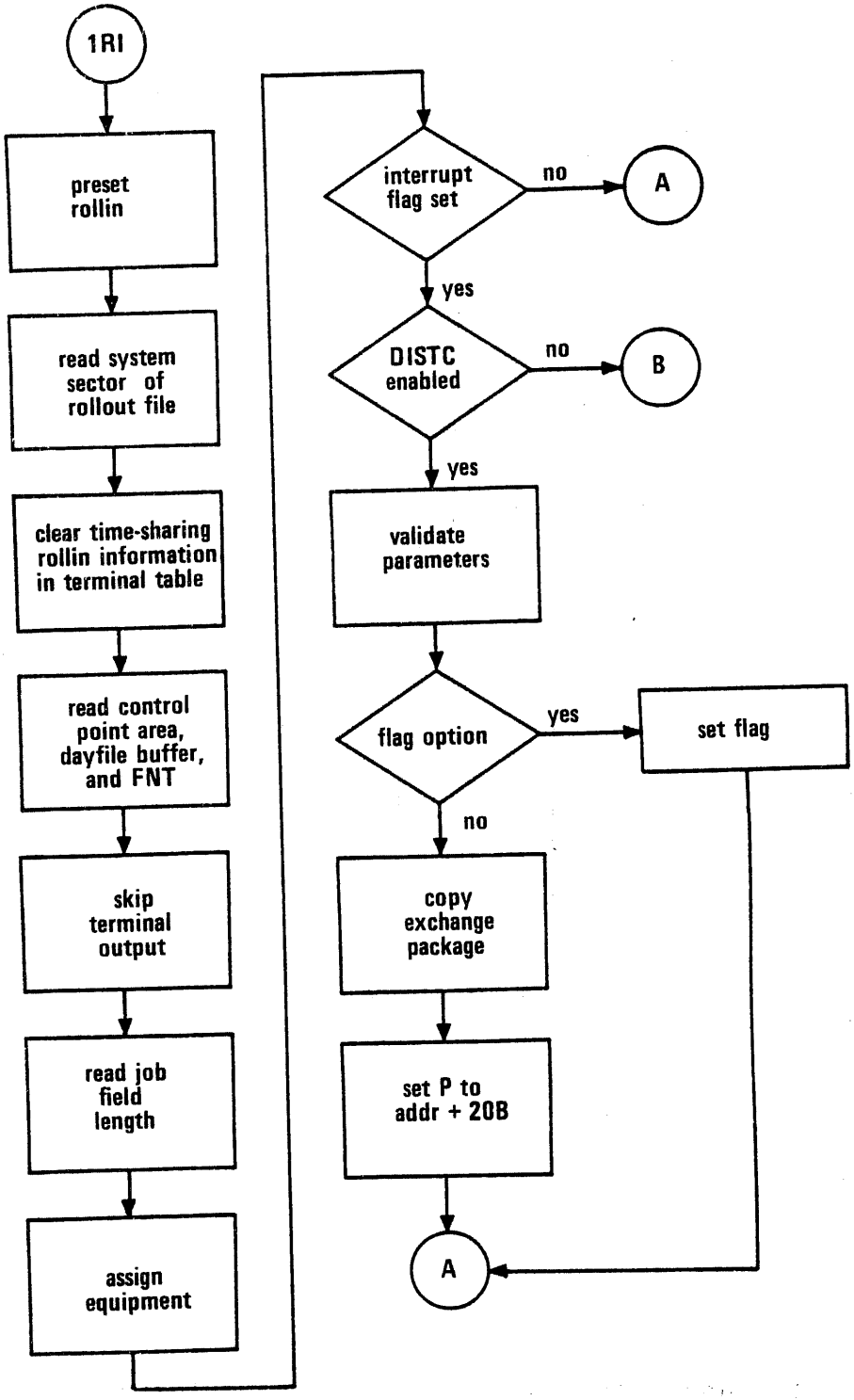


Figure 35-4. 1RI Interrupt Processing

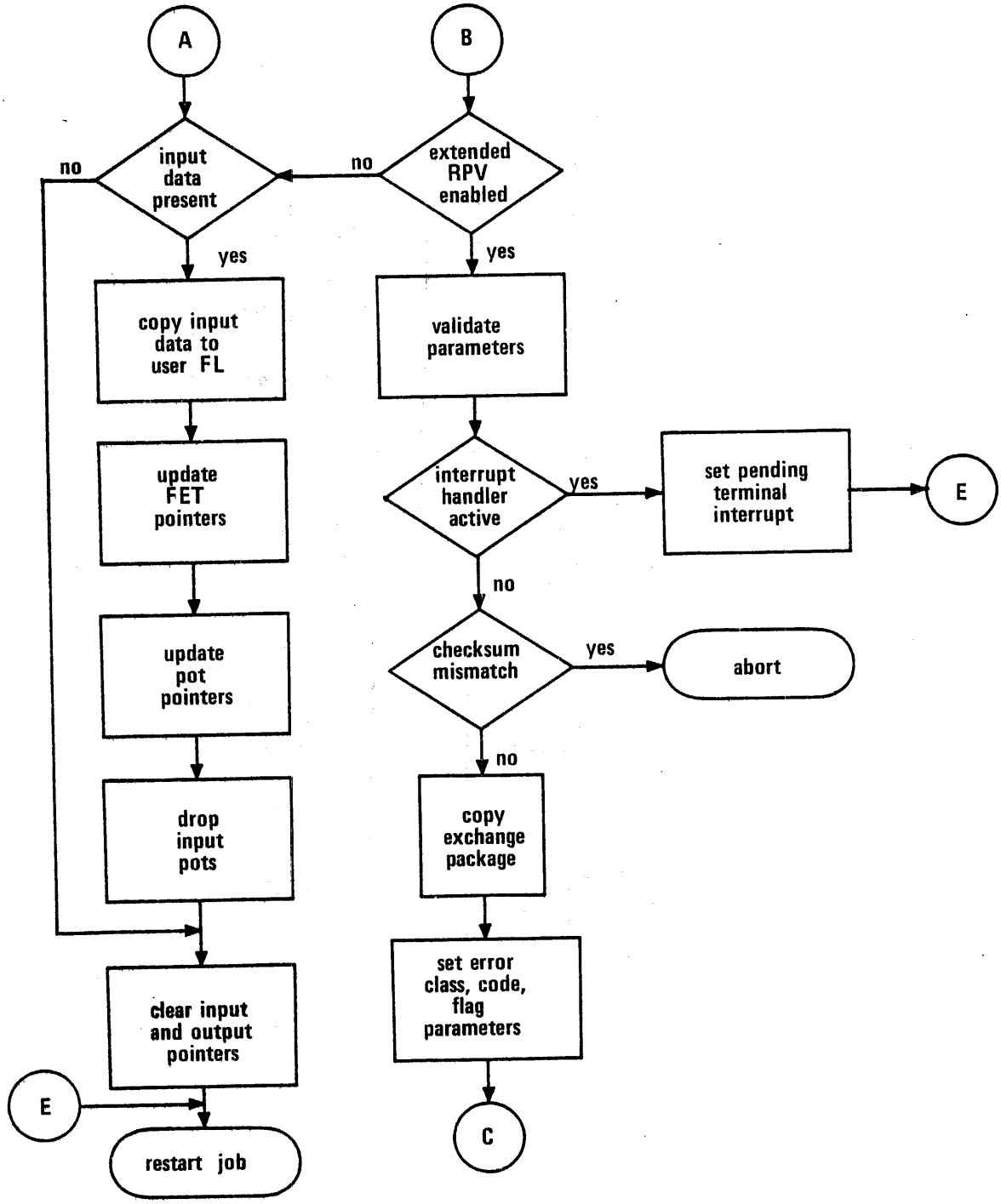


Figure 35-4. 1RI Interrupt Processing (Continued)

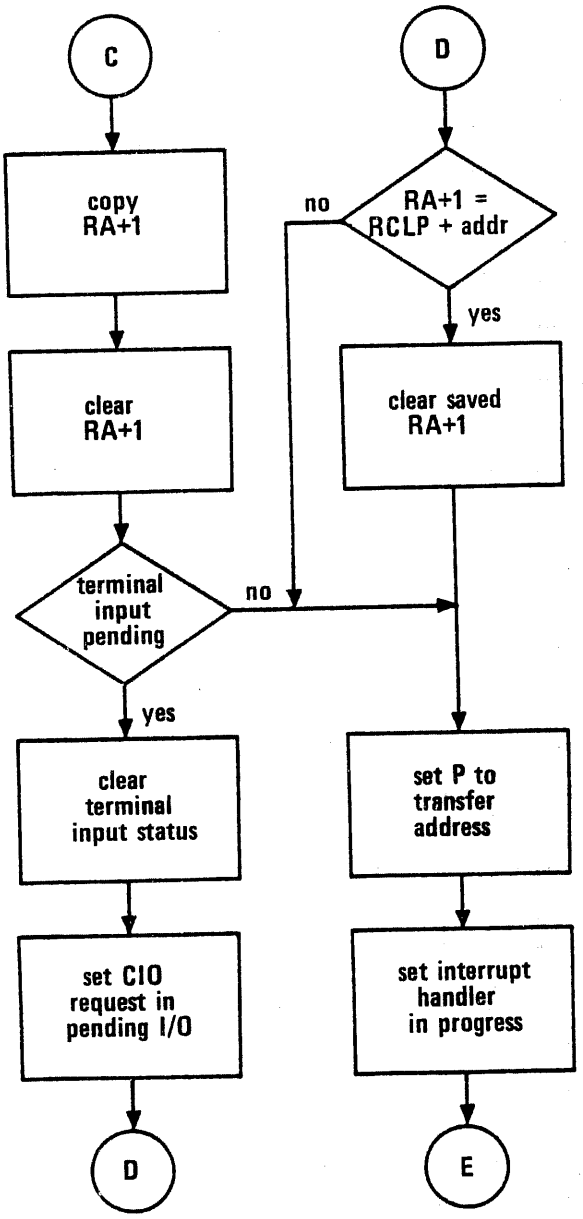


Figure 35-4. 1RI Interrupt Processing (Continued)

INTRODUCTION

The permanent file utilities are used to maintain the permanent file system. These utilities provide for the dumping and loading of permanent files, the cataloging of files in the system and on an archive tape, and the copying of specific files to a control point.

The permanent file utilities consist of the following.

- Permanent file supervisor (PFS)
- Permanent file utility processor (PFU)
- Dump permanent files (PFDUMP)
- Load permanent files (PFLOAD)
- Catalog permanent file device (PFCAT)
- Catalog archive tape (PFATC)
- Copy archived files to control point (PFCOPY)

These utilities can be initiated from system origin jobs or by users with system origin privileges and DEBUG mode on at the console.

This section assumes that the reader is familiar with NOS permanent file concepts. If a review of these concepts is needed, the NOS System Maintenance Reference Manual is recommended, not only for PF concepts but for the external properties of the permanent file utilities themselves.

PFS - PERMANENT FILE SUPERVISOR

PFS processes the permanent file utility control statements. It performs parameter processing for all of the utilities for both control statement and K-display parameter entries. After the arguments have been processed, PFS loads the desired utility.

Where possible, the available parameters for the utilities are compatible; that is, a parameter (FM for example) means the same (family name) in each of the utilities for which it is applicable.

A list of the parameters and the utilities for which they are applicable is shown in table 36-1. Refer to the NOS System Maintenance Reference Manual for a complete description of the parameters.

TABLE 36-1. PARAMETERS AND UTILITIES

Param.	Description	PFLOAD	PFDUMP	PFCAT	PFATC	PFCOPY
FM	Family name	X	X	X		
PN	Pack name	X	X	X		
DN	Device number	X	X	X		
TD	True device number	X	X			
T	Archive file name	X	X		X	X
LO	List option					
	T Files processed	X	X	X	X	X
	C Catalog files	X	X		X	
	E Errors	X	X	X	X	X
	S Summary			X		
L	Output file name	X	X	X	X	X
OP	Utility option					
	C Creation	X	X	X	X	X
	A Last access	X	X	X	X	X
	M Last modification	X	X	X	X	X
	I Indirect	X	X	X	X	X
	D Direct	X	X	X	X	X
	B Before date and time	X	X	X	X	X
	P Purge after dump		X			
	R Replace	X				X
	Q Catalog and permit records					
	N Noninitial	X				
	E Extract cir only	X				
	O Omit cir	X				
EO	Error option	X	X			
NT	Nine track	X	X		X	X
NR	No rewind	X	X		X	X
NU	No unload		X			
SF	Number of files to skip	X	X		X	X
N	Number of files to process	X			X	X
DT	Date	X	X	X	X	X

TABLE 36-1. PARAMETERS AND UTILITIES (CONTINUED)

Param.	Description	PFLOAD	PFDUMP	PFCAT	PFATC	PFCOPY
TM	Time	X	X	X	X	X
UI	User index	X	X	X	X	X
PF	Permanent file name	X	X	X	X	X
DI	Destination user index	X				
VF	Verify file name		X			
V	Verify file generation		X			
DD	Destination device number	X				
UN	User number	X	X	X	X	X
MF	Master file name					X

PFS has two modes of entry: entered as PFS or entered as one of the permanent file utilities (PFATC, PFCAT, PFCOPY, PFDUMP or PFLOAD). When entered at one of the utility entry points, PFS uses the K display only if one of the entered parameters was incorrect. When entered at the PFS entry point, PFS uses the K display for utility and parameter selection.

PFS first formats the K display for utility selection. The utility to be run is chosen by the console input of a two-character parameter that specifies the utility. These values are as follows.

Parameter	Utility
AT	PFATC
CA	PFCAT
CP	PFCOPY
DU	PFDUMP
LD	PFLOAD

Once a utility has been selected, PFS formats the K display for that utility. The parameter entry display includes information as to which parameters are applicable for the utility. This display is also used for parameter errors on control statement entries for the individual utilities.

After all parameters have been processed, the individual utility overlay is loaded via an OVERLAY macro request and

control transferred to it. The utility is loaded at one of two addresses: OVLA or OVLB. The difference between these addresses is that OVLB includes PFS's keyboard entry processor as resident code while OVLA does not. This allows the PFDUMP and PFLOAD utilties to further communicate with the operator during execution. Since PFATC, PFCAT, and PFCOPY require no subsequent console intervention they are loaded at OVLA, while PFDUMP and PFLOAD are loaded at OVLB. A memory map of PFS's field length is shown as figure 36-1.

The communication between PFS and the utilities is facilitated by common deck COMSPFS. This common deck contains location symbols for the K display, converted parameter area, keyboard input processor, and utility load addresses so that each utility is able to find the parameters PFS has processed and continue to communicate with the operator's console as necessary via the K display.

Parameter processing consists of cracking arguments entered by a control statement or through console entry, ultimately storing them in the table of converted parameters (PARC). The position within PARC for each parameter is defined by the converted parameter location table in COMSPFS.

The entered parameters are processed by common deck COMCARG and therefore require an equivalence table for the parameters. The call to ARG places the entered parameters into the table of entered parameters (PARE) at a position specified in the equivalence table that is defined by the option value table (OTBL). The parameters are read from PARE, interrogated, validated, and then stored in PARC.

The validation of the parameters for a given utility is accomplished by using a bit position table defined for each utility. The utility valid option mask table has a one-word entry for each utility. This entry is generated by using the OPTION macro, defined in COMSPFS, which sets a bit in the word based on the value for the parameter as specified in the OTBL. Thus the nth entry of PARE is a valid option for the utility only if bit 59-n is set in the utility valid option mask table.

This technique allows for a convenient shift and read mechanism to validate the variety of options available within the permanent file utilities.

PFS processes the parameter conversion (moving the parameters from PARE to PARC) by groups in the following order.

1. The name parameter group consists of those parameters which have their keyword equated to the name of the item. Name parameters are FM, PN, T, VF, L, PF, MF, and UN.

2. The list option parameter group is the string of list options specified with the LO keyword. These options are unpacked from the string entry, validated, and stored in PARC by subroutine OCK (option cracker).

3. The utility option parameter group is the string of utility options specified with the OP keyword. These options are unpacked from the string entry, validated, and stored in PARC by subroutine OCK.

4. The octal parameter group consists of those parameters that are entered as octal values. These values are converted and stored as binary numbers in PARC. The keywords in this group are UI, DI, DN, DD, and TD.

5. The decimal parameter group consists of those parameters that are entered as decimal values. These values, entered with the N and SF keywords, are converted and stored as binary numbers in PARC.

6. The single entry group consists of those parameters that represent a flag setting to indicate whether to perform a given operation or not. This group contains keywords NR, NU, V, NT, and EO.

7. The final group consists of date and time. The yymmdd entry for the DT keyword and the hhmmss entry for the TM keyword are converted and stored in packed date and time format in PARC.

The argument processing of PFS is flowcharted as figure 36-2.

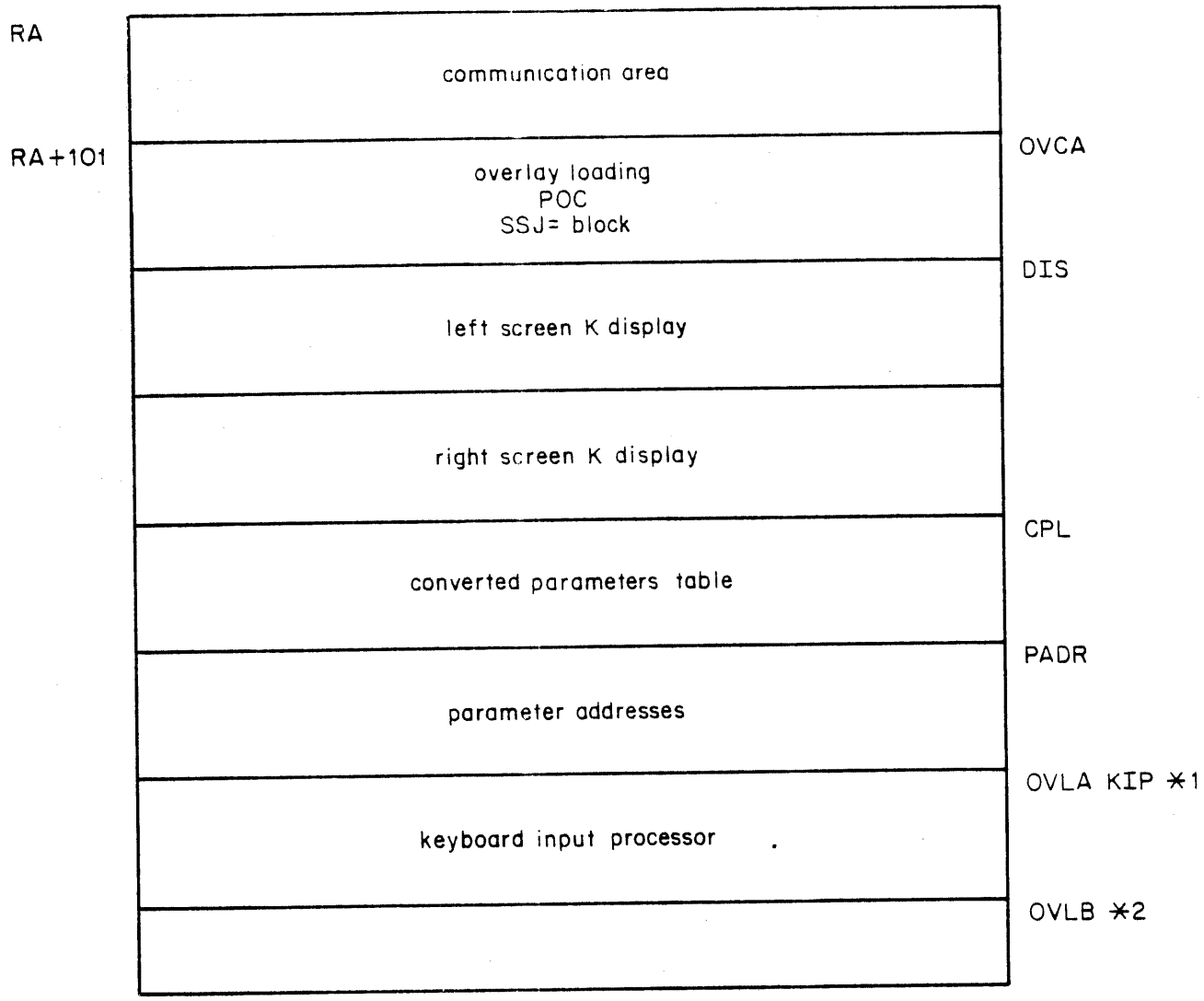
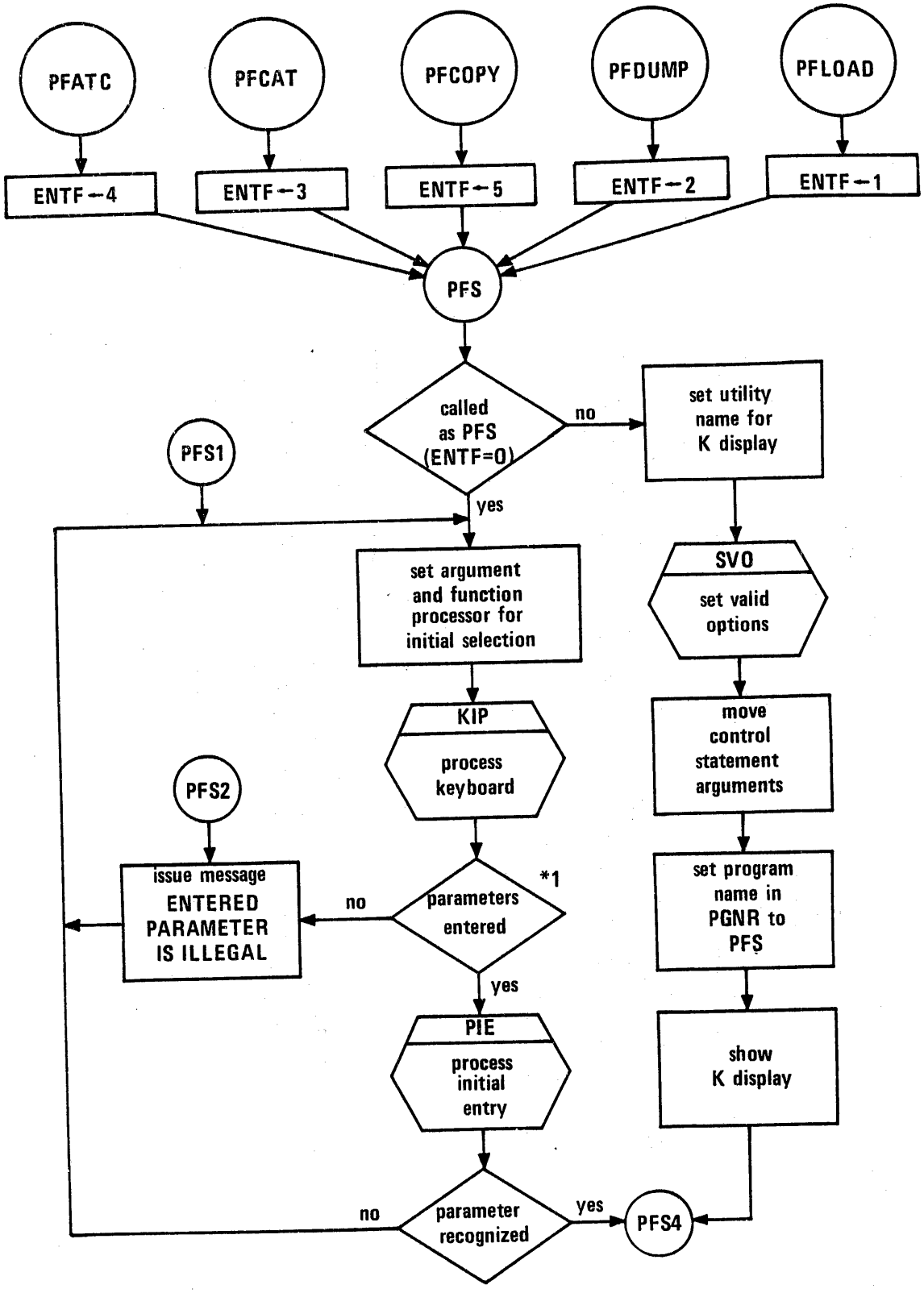


Figure 36-1. PF Utilities Memory Map



*1 Entry to KIP is set up to return to this address.

Figure 36-2. PFS Argument Processing

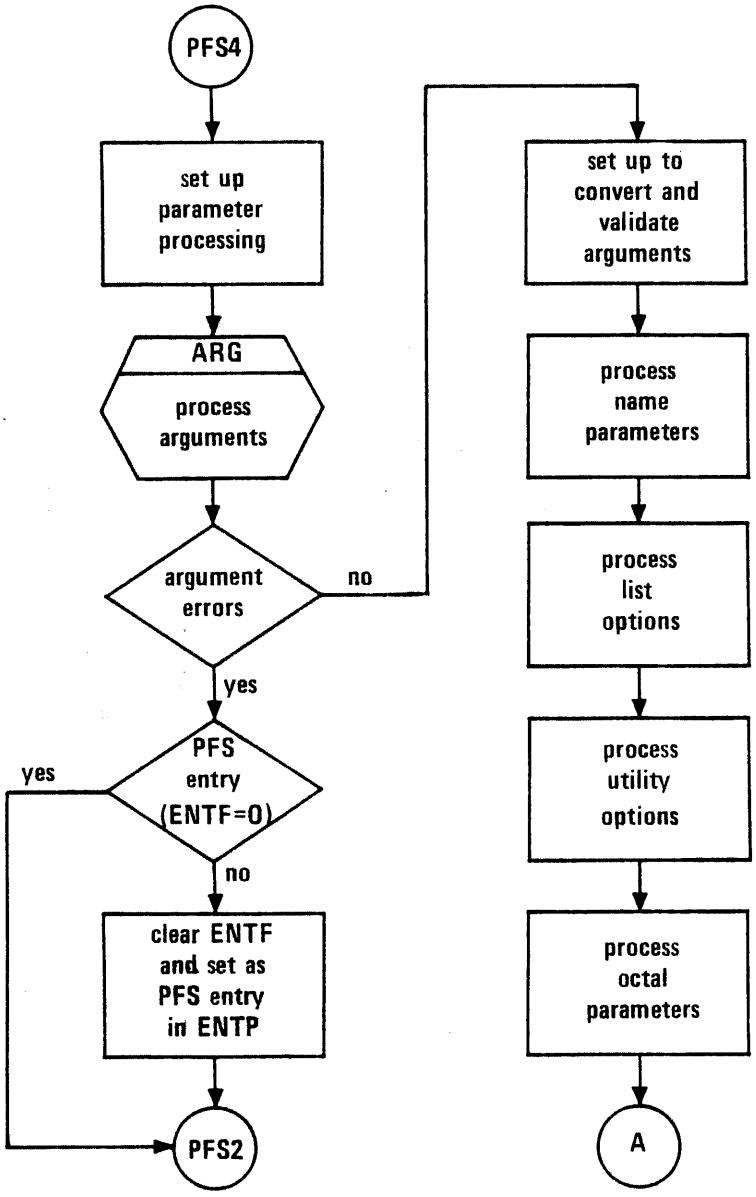


Figure 36-2. PFS Argument Processing (Continued)

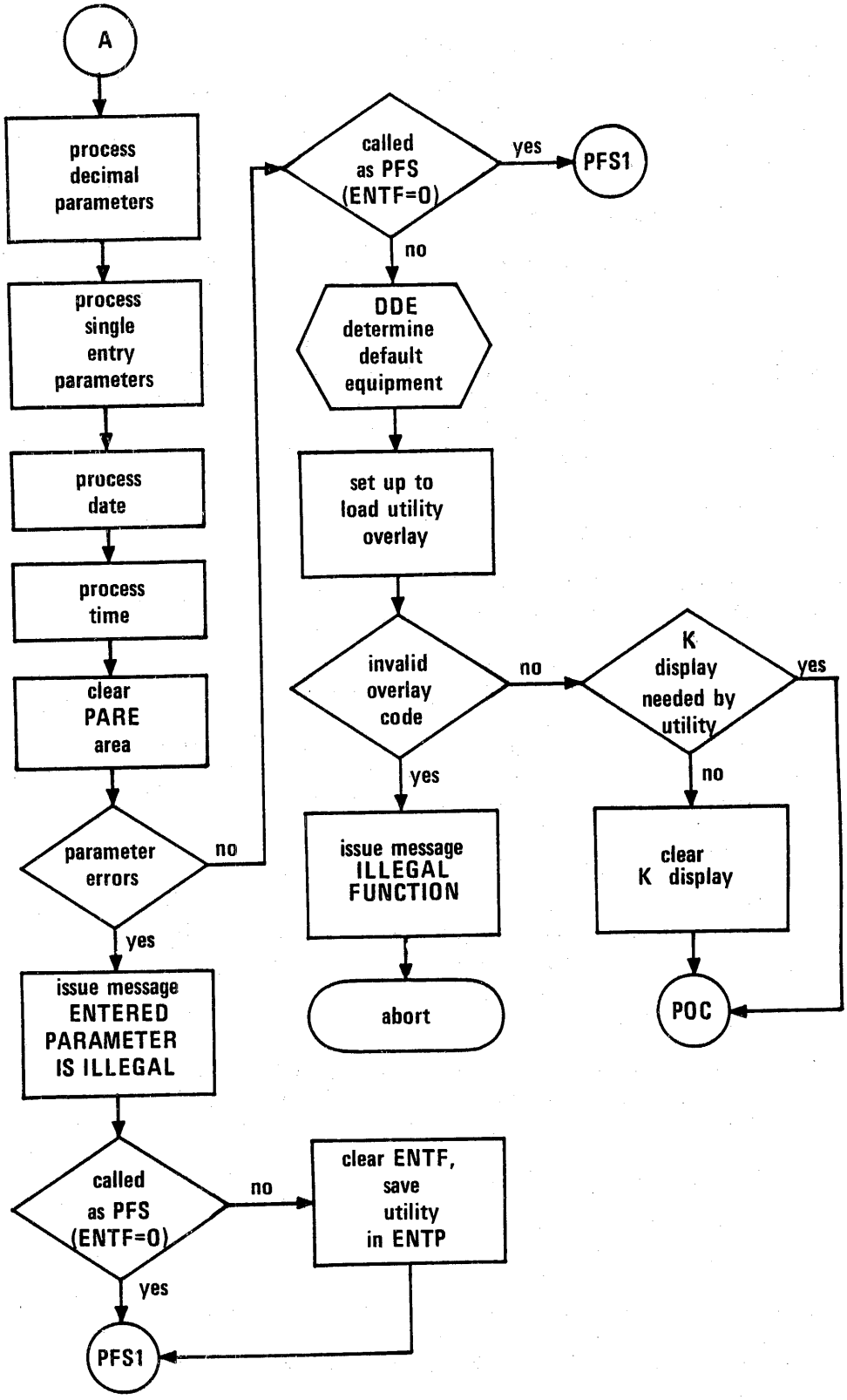


Figure 36-2. PFS Argument Processing (Continued)

The following paragraphs describe the main subroutines in PFS.

POC - PROCESS OVERLAY CALL

POC loads the individual permanent file utility and transfers control to it.

KIP - KEYBOARD PROCESSOR

KIP unpacks input parameters that have been entered via the operator's console under K display control.

KIP is entered with two exit addresses: one for argument processing and one for function processing. The argument processing address is used when processing argument parameters while the function processing address is used only when the GO parameter has been issued.

CDT - CONVERT DATE AND TIME

CDT converts the date and time from yymmdd and hhmmss to the system packed date and time format.

DDE - DETERMINE DEFAULT EQUIPMENT

DDE validates the use of the family name (FM) and pack name (PN) parameters. If both are specified the following diagnostic is issued, since these two parameters are mutually exclusive.

BOTH FAMILY AND PACK NAME.

The EST and MST are read using the RSB monitor function to verify the equipment is available by the specified name. If an entry is not found, the following diagnostic is issued.

FAMILY/PACK NOT FOUND.

If a private pack is being processed, the pack's user number and that supplied via the UN parameter must agree or no match occurs.

If a user number (UN) has been specified, it is validated. If it is not valid, the following message is issued.

USER NUMBER INVALID.

DDE also verifies that if the PF parameter is used, the user index (UI) or user number (UN) has also been specified. If not, the following diagnostic is issued.

PF SPECIFIED BUT NOT UI.

OCK - OPTION CHECK

OCK processes the string of options that are associated with the LO and OP parameters. OCK unpacks the strings and validates the option using the utility valid option mask table.

OCP - OPTION COMBINATION PROCESSOR

OCP validates combinations of utility options (OP) to insure that conflicting options have not been specified.

PIE - PROCESS INITIAL ENTRY

PIE processes the arguments entered when selecting which utility will be used via the K display. PIE calls SVO (set valid options) to format the individual utility K display.

SVO - SET VALID OPTIONS

SVO sets list options, utility options, and parameter values that are valid for the particular utility on the left screen K display for the utility. SVO uses the utility valid option mask table to indicate what options are available for the utility.

PFU - PF UTILITY PROCESSOR

PFU performs a variety of utility functions for the permanent file utilities. Functions are provided for managing the interlocking of permanent file system activity (PFNL), the interlocking of individual devices (utility interlock), the interlocking of individual catalog tracks, and the setting of device error idle status. Functions are also provided for managing a scratch file associated with permanent file catalogs (CATS), permits (PETS), and data (DATA). PFU provides for creating these special files, positioning them, transferring data to or from them, determining their length, and so on. The FNT/FST entry and the FET for these files contain information that is relevant to their use within the permanent file utilities.

Common deck COMSPFU provides for the definition of function codes. FET equivalences, and a CPU program macro for calling PFU.

Also in COMSPFU, the format of the archive tape label (this describes the data, not the tape) and the archive tape control word are defined. These formats are described under Writing an Archive File in the PFDUMP subsection.

The following is a list of PFU function codes.

Symbol	Value	Definition
CTOP	000	Open file no lockout
CTOL	100	Open file with write lockout
CTAC	200	Advance catalog track
CTRL	300	Read data list (PFDUMP)
CTLM	400	Load main loop (PFLOAD)
CTSU	500	Set PF utility interlock
CTCU	600	Clear PF utility interlock
CTRC	700	Rewind catalog file
CTCF	1000	Change file name
CTFL	1100	Set file length
CTSC	1200	Set catalog track interlock
CTCC	1300	Clear catalog track interlock
CTEI	1400	Set error idle status
CTCT	1500	Locate catalog track
CTIA	1600	Increment PF activity count
CTDA	1700	Decrement PF activity count
CTTU	2000	Test PF utility interlock

Table 36-2 illustrates which functions are used by each PF utility (PFATC does not do any PFU activity). Figure 36-3 illustrates the PF utility FET.

TABLE 36-2. PFU FUNCTION USAGE

Function	PF Utility			
	PFCAT	PFCOPY	PFDUMP	PFLOAD
CTOP				X
CTOL	X		X	
CTAC	X		X	
CTRL			X	
CTLM				X
CTSU				X
CTCU				X
CTRC	X			
CTCF		X		
CTFL	X			
CTSC			X	
CTCC			X	
CTEI			X	X
CTCT	X		X	X
CTIA	X		X	
CTDA	X		X	
CTTU	X		X	

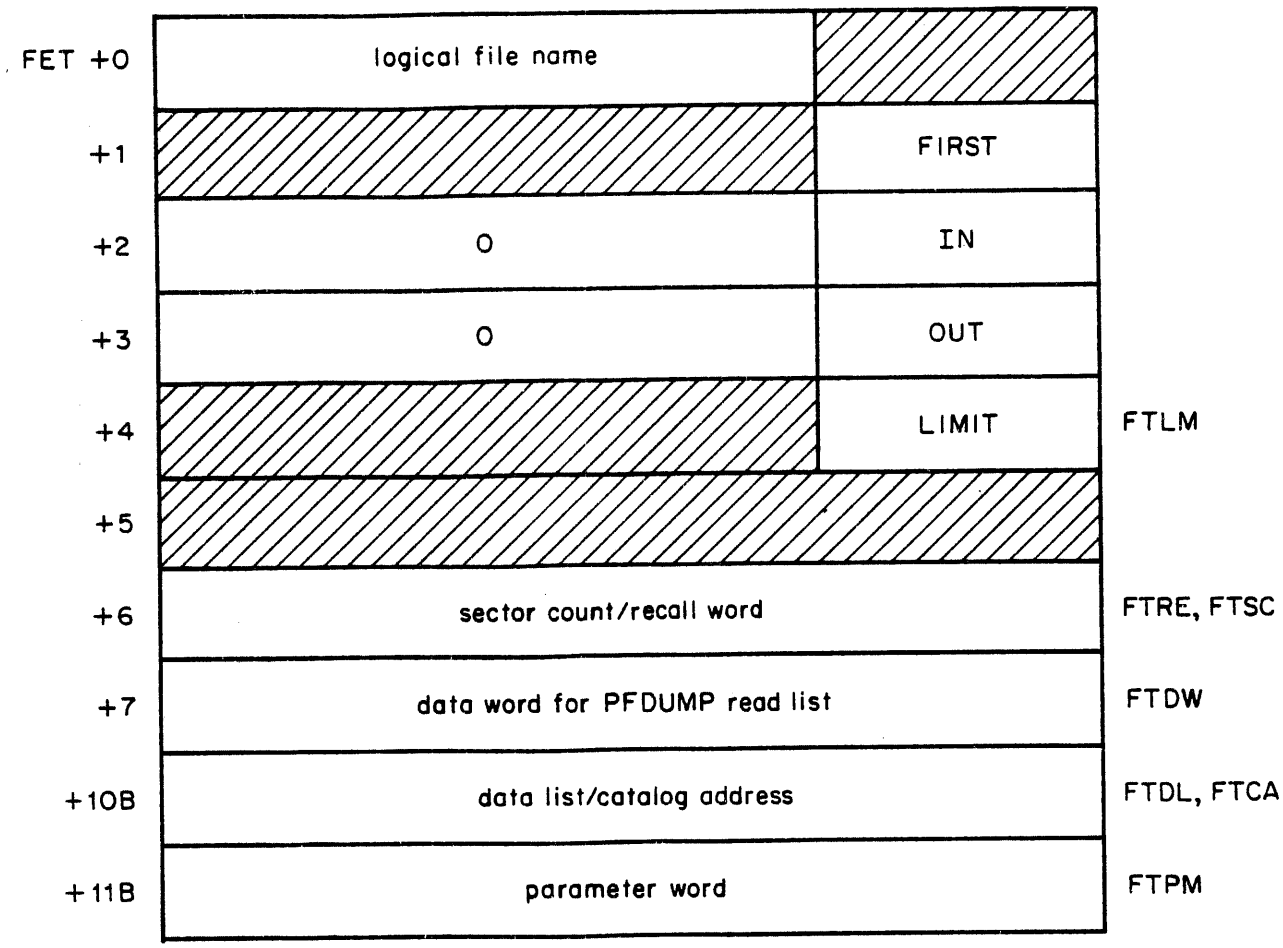


Figure 36-3. PF Utility FET

PFU STRUCTURE

PFU consists of function processors for each of the PFU functions plus a group of common subroutines.

Upon entry to PFU, a preset routine (PRS) is called. PRS validates the FET address to be within the program's field length, checks to see that the completion bit is not set, and whether the caller is a system origin job or has system origin privileges with the system in debug mode. The function code is then validated and the processor address and overlay name returned to the main program. The main program transfers control to the processor via a return jump. When the processor has completed, the main program is reentered. The file and FET are set complete by calls to subroutines SFC (set file complete) and CFS (complete FET status). The PP is then dropped via a DPPM monitor function and control is returned to PPR.

PFU major subroutines include the following.

CAU - CLEAR PFU ACTIVE FLAG

CAU clears the PF utility active flag that is kept in the utility's field length. The address of the PFU active flag is supplied when the flag is set and the address is saved in locations PFAF through PFAF+1.

CCA - CHECK CENTRAL ADDRESS

CCA is called as part of the VADDR macro. VADDR validates an address that is right- or left-justified within two adjacent PP memory locations. CCA gets the address and calls subroutine VCA (validate central address) to validate the address. CCA aborts with the following diagnostic if the validation is not successful.

PFU - PARAMETER ERROR.

CFA - COMPUTE FET ADDRESS

CFA is called as part of the FETA macro. CFA returns the absolute address of the FET word from a supplied word number, the FET address (IR+3 through IR+4), and RA.

CFS - COMPLETE FET STATUS

CFS sets the completion bit (bit zero in word zero of the FET) and clears the FET address from the input register (IR+3 through IR+4).

DCH. - DROP CHANNEL IF RESERVED

DCH. examines the channel flag (CF) and drops the channel (T4) if CF is nonzero. CF is cleared before returning to the caller.

FAR - FORCE AUTORECALL

FAR calls CRS (COMPCRS - check autorecall status) and aborts with the following diagnostic if PFU is called without autorecall.

PFU - PARAMETER ERROR.

FFE - FIND FNT ENTRY

FFE is called to find the FNT entry of the file name specified in the FET. FFE is called with two options: make LIFT check, and abort if file not found. If the file name in the FET agrees with the name in the FNT at the address specified in FET word FTLM, the FNT search is by-passed. If the file names do not agree, the FNT is searched for a match. If no match occurs and abort on error is selected, the following diagnostic is issued and PFU is aborted. If no abort is selected, the exit address is advanced by one and control transferred to that address.

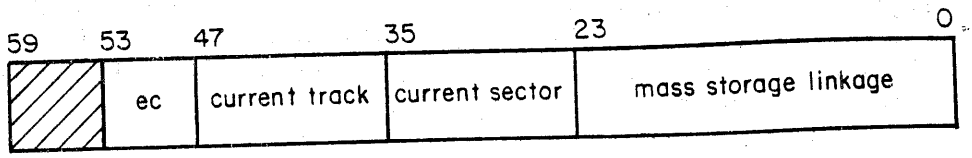
PFU - FILE NOT FOUND.

When an FNT entry with the correct file name is found, the library type check is done if selected. If the file found is not a library file, the condition is treated as if a match did not occur. If the library condition is satisfied, the FNT address is entered in FET word FTLM, and the file is set busy via the STBM monitor function. The FST is read and control is returned to the caller if the file is successfully set busy. If the file is not set busy, the following diagnostic is issued and PFU is aborted.

PFU - I/O SEQUENCE ERROR ON filenam AT addr.

LDB - LOAD BUFFER

LDB transfers a PRU of data from mass storage to a FET buffer in the utility's field length. Data is transferred until the FET buffer is full or until the end-of-information for the file being transferred is reached. Each block of data is preceded by a control word in the following format.



ec Error code:

0 No error
1 File too long; linkage set to EOI
2 Nonfatal mass storage error
3 Fatal mass storage error; linkage set to EOI

PAR - PAUSE AND RESET ADDRESSES

PAR does a PAUSE and then checks to see if absolute addresses need to be changed due to a change in the utilities RA.

PDA - PROCESS DIRECT ACCESS FILE

PDA determines if the direct access file (DAF) being processed resides on the master device being processed. If it does not, the master device channel is released and mass storage processing is set up for the DAF device. If the DAF device is not a valid member of the family, the following diagnostic is issued and PFU is aborted. PDA sets the DAF flag (DA) prior to returning to its caller.

ALTERNATE DEVICE NOT FOUND.

RCH. - REQUEST CHANNEL IF NOT RESERVED

RCH. checks the channel flag (CF) and if it is not set, requests the channel specified in T4. The channel flag is set before returning to the caller.

RPP - RECALL PP

RPP takes the PP call word contained in direct cells IR through IR+4, converts it to an RA+1 call with autorecall, writes it to RA+1, drops the PP via a DPPM monitor function, and then transfers control to PPR.

SAP - SET ADDRESSES FOR DUMP AND LOAD

SAP presets instructions with absolute FET buffer address for data transfers between the FET buffer and PFU's buffer.

SAU - SET PFU ACTIVE FLAG

SAU sets a PF utility active flag at a specified address within the utility's field length. The address of this flag is saved internally to PFU so that the flag may be cleared by CAU before PFU is dropped.

SBA - SET BUFFER ARGUMENTS

SBA reads FIRST, IN, OUT, and LIMIT from the FET into direct cells FT through FT+1, IN through IN+1, OT through OT+1, and LM through LM+1.

SCT - SET CATALOG TRACK

SCT reads the permanent file descriptor word and user index from the utility. The catalog track for the user index is determined via a call to subroutine SCA (COMPSCA - set catalog address). If the catalog track cannot be found, the following diagnostic is issued and PFU is aborted.

```
PFU - CATALOG TRACK NOT FOUND.
```

SFC - SET FILE COMPLETE

SFC sets the file not busy bit in the FST for the file. FA is cleared on return to the caller.

SFF - STORE FILE NAME AND FET ADDRESS

SFF completes the diagnostic message which is of the form:

```
filenam AT addr.
```

by supplying the file name and address.

SFT - SET FILE TYPE

SFT sets the file type and mode in the FNT entry for the specified file.

SOC - STORE ONE CHARACTER

SOC stores one display code character at the position specified in the call.

STS - STORE STRING

STS stores up to three characters at the position specified in the call.

UFP - UPDATE FET POINTERS

UFP writes the IN pointer from direct cells to the FET and reads the OUT pointer from the FET to its direct cells.

VCA - VALIDATE CENTRAL ADDRESS

VCA verifies that a central memory address falls within the control point's field length.

VME - VALIDATE MASS STORAGE EQUIPMENT

VME checks that a specified equipment is a mass storage device. If not, the following diagnostic is issued and PFU is aborted.

PFU - PARAMETER ERROR.

WIF - WRITE INTERLOCK FLAG

WIF sets/clears an interlock flag in the utility's field length.

PFU COMMON DECKS

PFU also uses the following common decks as common subroutines.

Common Deck	Description
COMPCRA	Convert random address
COMPCRS	Check recall status
COMPCII	Clear track interlock
COMPCUT	Clear permanent file utility interlock
COMPRSS	Read system sector
COMPSCA	Set catalog address
COMPSDN	Search for device number
COMPSEI	Search for end-of-information
COMPSNT	Set next track
COMPSRA	Set random address
COMPSTI	Set track interlock
COMPSUT	Set permanent file utility interlock
COMPIRA	Initialize random access processors

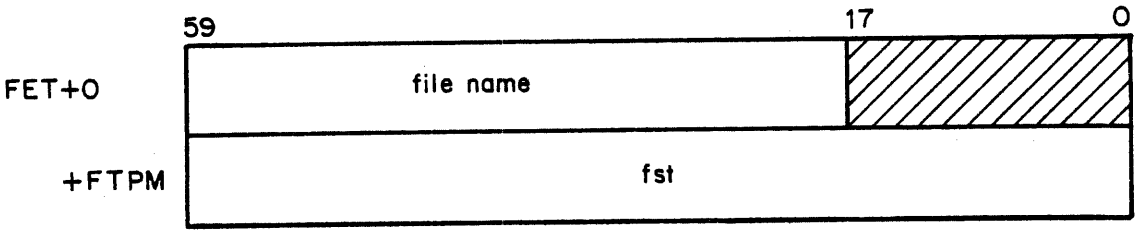
The following subsections describe the PFU functions.

OPN - OPEN FILE

The open file (OPN) function includes the following options.

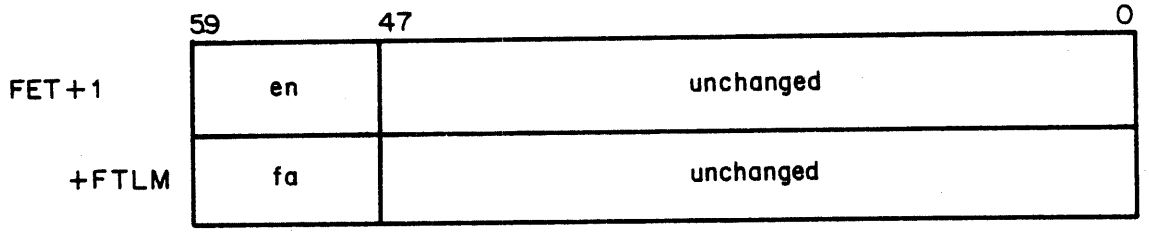
Function	Symbol	Description
00	CTOP	No write lockout
01	CTOL	With write lockout

Upon entry to OPN, FET+0 and FET+FTPM must have the following format.



fst FST for file

Upon exit from OPN, FET+1 and FET+FTLM have the following format.



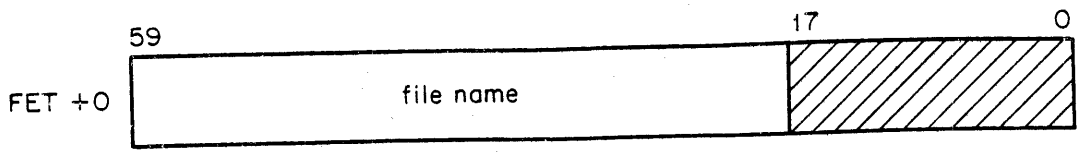
en Equipment mnemonic
fa FNT address of file

OPN makes an FNT/FST entry for the specified file, makes it library type (LIFT), and uses the FST entry passed in the FET for the file's FST. The write lockout bit (bit 12) is set in the FNT for the CTOL function. The equipment mnemonic and FNT address are set into the FET before returning to the utility.

ACF - ADVANCE CATALOG FILE

The ACF function, CTAC (02), advances the position of the catalog file to the next track in the chain and advances the catalog interlock if set.

Upon entry to ACF, FET+0 must have the following format.

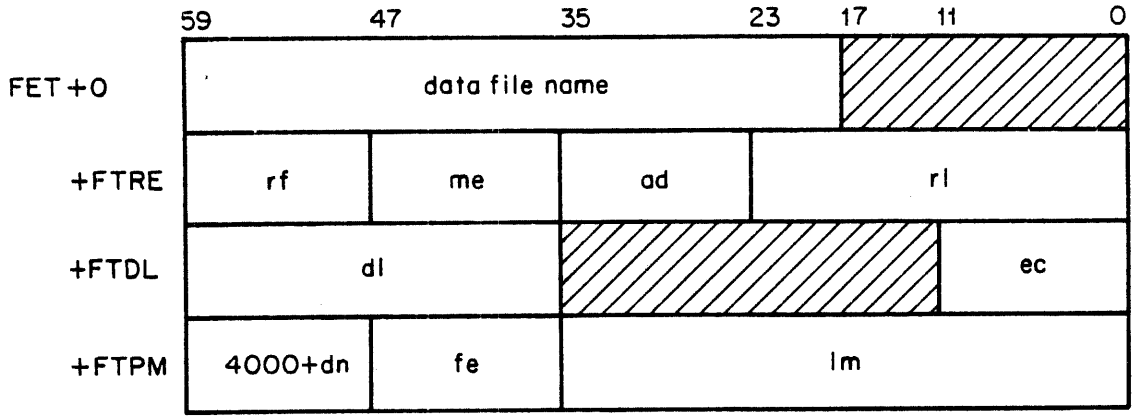


ACF advances to the next catalog track by a call to subroutine SNT (COMPSNT - set next track) which returns the next track in the catalog chain. If the previous catalog track had been interlocked (bit 9 of FST is set), then the previous catalog track interlock is released by a call to subroutine CTI (COMPCTI - clear track interlock) and the new track is interlocked by a call to subroutine STI (COMPSTI - set track interlock). The file is then set to begin reading the new catalog track.

RRD - READ DATA LIST

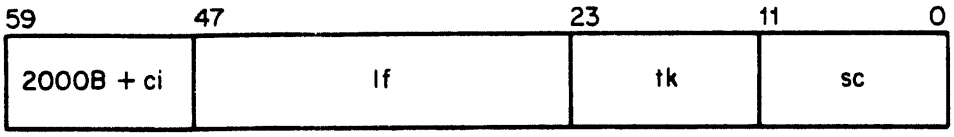
RRD, function CTRL (03), reads a list of files from disk to a CM buffer.

Upon entry, the following FET words must be formatted as shown.



rf	Recall flag: 0 Not recall call (zero word) 7777 RA+1 recall call
me	Master equipment number
ad	Alternate device number
rl	Remaining file length
dl	Data list address
ec	Element count of data list
dn	Device number
fe	Family equipment
lm	List modification date and time

RRD is the primary function used in permanent file dumping. PFDUMP builds a data list buffer in its field length which indicates the file(s) to be dumped. The data list buffer entries have the following format.



ci	Catalog index into catalog buffer
lf	Length of file (indirect access only)
tk	First track of file
sc	First sector for indirect access files; 1/1,1/f,4/0,6/rd for direct access files f 0 for normal processing; 1 for forced dump (no date checking) rd Residency device number

RRD works through the data list dumping (copying the file data to the DATA FET buffer) each file encountered.

For indirect access files, the FST for the DATA file is built using the track and sector specified in the data list entry. For direct access files, the DATA file FST is built, with the equipment number and track information in the data list entry. The system sector is validated by a call to CSS (check system sector) to verify that the file may be dumped.

The data is transferred from mass storage to the FET buffer by a call to LDB (load buffer). When control is returned from LDB, a check is made to determine if the EOI had been processed. If the EOI had not been processed, a check is made to determine if PFU should continue transferring data. If no PPs are available or the file being dumped is a large direct access file, the recall word is set into FET word FTRE and the data list control word is set into FET word FTDL and PFU drops. If PPs were available (archive tape can be written simultaneously with buffer filling) or the file being dumped is a long direct access file, RRD pauses by a call to PAR, sets the length for the remaining read, and calls LDB to continue reading.

If an EOI had been processed, dumping proceeds with the next element in the data list. Before continuing, however, direct access file pointers are cleared and if a direct access file was being dumped, master device pointers are restored (if the file was not on the master device) and fast attach controls updated (by a call to RRF - return fast attach file) if the file was a FAFT file.

If there are no more data list items, the data list word is cleared from the FET (word FTDW), the sector number is cleared in the DATA file FST and PFU is dropped.

If RRD had been entered with a recall condition (FET word FTRE set), the working direct cells are restored from FTRE and the data transfer activity is resumed.

RRD uses a collection of subroutines that are primarily associated with direct access file dumping.

Subroutine CSS (check system sector) is called to process the direct access file's system sector. If the system sector cannot be read, a bad system sector status is returned.

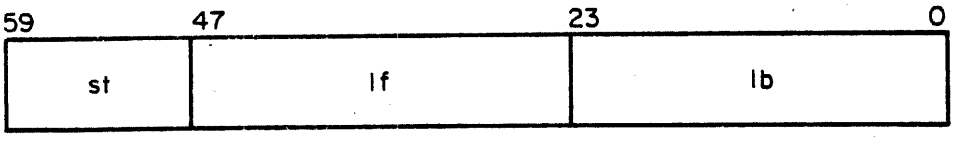
Once the system sector has been read, it is checked to determine if the file can be dumped. Subroutine CDS (check dump status) first determines if the file is a fast attach file and if so calls subroutine FAF (fast attach file) to interlock it for dumping. FAF searches the FNT for the file, and if found, attaches it in read allow modify mode and sets the user count using the PTRM/AFAS options on the IAUM and IUCS option on the STBM monitor functions. If the file is successfully attached, subroutine SFT is called to set the DATA file as a permanent, read allow modify file and CDS returns to CSS with the dump file status set to dump the file.

If the file is not a fast attach file or is not found in the FNT as fast attach, the file's mode is checked (FCCA in the system sector) and if the file is in write mode, that status is returned to CSS.

If the file is attached in modify or extend mode, the DATA file is set as a read allow modify library file by a call to SFT and CDS returns to CSS with the dump status set to dump the file.

If the file is not attached in any mode allowing writing, the DATA file is set as a read mode library file by a call to SFT. If the force dump flag is set, the dump file status is set to dump the file. Otherwise, the modification date is checked by subroutine CMD (check modification date) and that result is returned to CSS by CDS.

Once CSS resumes control, it sets the file length (by a call to subroutine SEI (COMPSEI - search for end-of-information), sets the working length by a call to subroutine SDL (set direct length), set first track and sector pointers, writes a control word having the following format into the FET buffer, and advances the IN pointer. CSS then returns to RRD with the dump status.



st File Status:

0	Dump file
1	File in write mode
2	Zero length file (no EOI)
4	Bad system sector
10	File does not meet modification date and time criteria

lf Length of file (includes EOI)

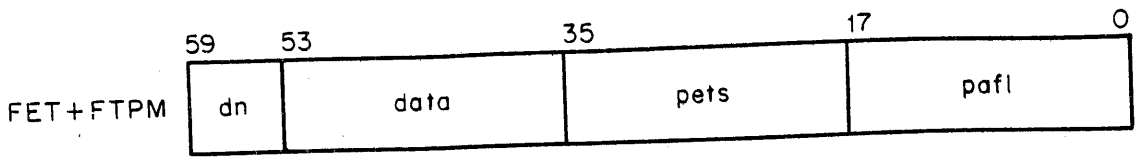
lb System sector linkage bytes

Subroutine RFF (return fast attach file) is used to return the fast attack file after dumping. RFF returns the file using a call to OFA and then resets the DATA file entry to a read mode library file by a call to SFT.

Subroutine SDL (set direct access file length) is used to determine the amount of the file to dump (transfer) at one time. The amount transferred in one LDB call is limited to the field length of the utility.

LML - LOAD MAIN LOOP

Function 4, LML, processes CATS, PETS, and DATA files. Upon entry, FET & FTPM has the following format.



dn	Master device number
data	Address of DATA file FET (0 if no DATA FET)
pets	Address of permits file FET (0 if no permits FET)
pafl	PFU active flag address; set to one when the PFU load processor is activated and set to zero when the PFU load processor completes or aborts.

LML is a separate overlay to PFU named 3FA. All the code required for permanent file loading by PFU is contained in this overlay.

LML processes functions on each of the three utility files: CATS, PETS, and DATA. These functions are passed in byte 4 of the first word in each FET. The functions are as follows

Function	Description
0	Position file
2	Write file
4	Read file (fill CATS buffer)
10	Completion of load

The main loop of loading (LML) performs the following functions.

- Checks for functions on CATS and performs the function desired.
- Checks for functions on PETS and performs the desired function.
- Checks for functions on DATA and performs the desired function.
- Checks for termination (termination function on CATS) and if termination is not desired goes through the loop again.

When the termination function is set, the utility active flag is cleared and PFU exits.

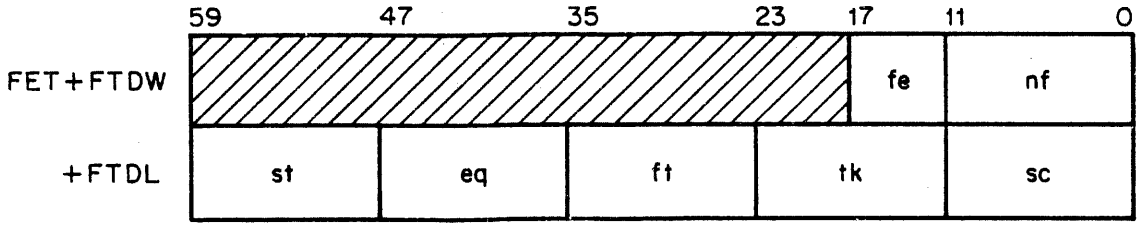
The operations (functions) performed follow the general pattern of a call to subroutine PIO (preset I/O); a call to the function processing subroutine, a call to subroutine CIO (complete I/O), and then a check of the next FET for an operation to perform. Unrecognized functions proceed to the next FET.

The function processors are as follows.

Function	Description
PCF	Position catalog file
PCW	Process catalog write
PCR	Process catalog read
PPF	Position permit file
PPW	Process permit write
PDF	Position data file
PDW	Process data write

The load sequence begins with a call to subroutine PLL (preset load loop). PLL reads the loading control word from FET word FTPM. PLL sets the PF utility active flag by a call to subroutine SAU. The device number, CATS FET address (from IR+3 through IR+4), PETS FET address, and DATA FET address are set into direct cells. The CATS FET is set complete by a call to CFS. If the PETS and DATA addresses are present, they are validated by a VADDR request. The DATA address is then moved to the input register IR+3 through IR+4, the files positioned by a call to subroutine POF, the DATA FET completed by a call to subroutine CFS, and control returned to the main loop.

Subroutine POF (position files) works with the DATA and PETS files. POF relies on the following information to be passed to PFU in the DATA FET



fe Equipment number of a family device

nf Noninitial load flag (zero to check that indirect access file chain is empty, nonzero to bypass check)

st Status:

- 0 If beginning of file
- 1 If middle of indirect access file
- 3 If middle of direct access file

eq Master equipment number if middle of direct access file

ft First track of indirect access file chain if middle of direct access file

tk Current track of indirect access file chain if middle of direct access file

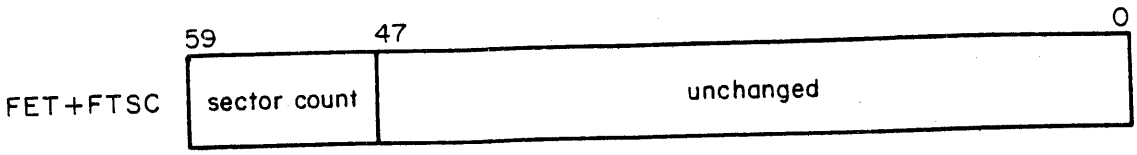
sc Current sector of indirect access file chain if middle of direct access file

After validating the DATA FNT/FST entry, POF finds the EOI for the DATA file by a call to SEI. If processing an indirect

access file, the NF value from FTDW is checked to determine the condition of the indirect chain. If NF is zero, the indirect chain must be empty otherwise the following diagnostic is issued and PFU is aborted.

DEVICE NOT INITIALIZED.

If PFU is not aborted, POF updates the current track and sector fields in the data file FST by calling SFC. POF then sets the last track and sector of the PETS file by performing the SEI/SFC sequence. The current sector count of the PETS file is returned to the PETS FET word FTSC in the following format.



The PETS file is then set complete and control is returned to the caller (PLL).

Load processing now continues its main loop, processing functions identified in the CATS, PETS, and DATA FETs.

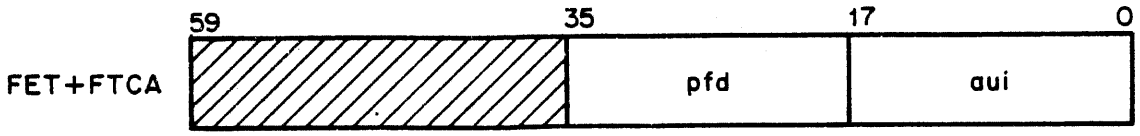
Most functions are performed by subroutine call sequences that begin with a call to subroutine PIO (preset I/O) and terminate with a call to CIO (complete I/O).

Subroutine PIO sets up direct cells for reading or writing by validating the file FNT/FST, initializing random addresses (subroutine IRA), setting buffer addresses (subroutine SBA and SAP), and clearing the sector count in subroutine SSC.

Subroutine CIO updates the status, current track, current sector and equipment fields of the direct cell copy of the FST and writes it to central memory using a call to SFC. The FET completion bit (bit 0) is set to 1 to indicate that the operation has completed.

CATS Position

Subroutine PCF (position catalog file) performs the positioning of the catalog file. Word FTCA is read from the CATS FET and subroutine SCT is called to identify the catalog track for the specified user number. The current sector position is cleared and the current sector count cleared in subroutine SSC. The FTCA word has the following format.

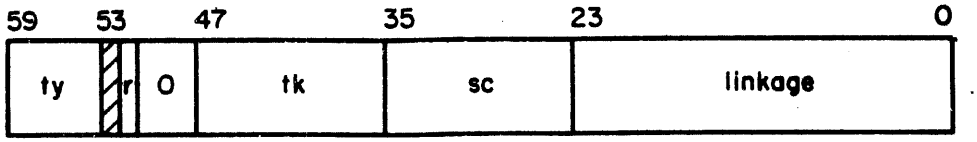


pfd Address of permanent file descriptor word
aui Address of user index word

CATS Write

Subroutine PCW (Process Catalog Write) calls subroutine SSC to get the count of the number of sectors remaining to be written. If no sectors remain, the sector counter in SSCT for the CATS file is cleared and PCW returns to the caller.

If sectors remain, the control word is read from the FET buffer by a call to RCW (read control word). Subroutine PCA (process CATS addressing) is called to set the track (T6) and sector (T7) pointers from the control word.



ty 2 if EOI
1 if EOF
0 if EOR or full sector

r Rewrite:

0 Normal
1 Rewrite in place

tk If nonzero, tk and sc are the random position for the catalog file; otherwise the current position is used

The channel is reserved and the data transferred from the FET buffer to mass storage by subroutine EMB (empty buffer). Data is transferred until the EOI has been reached or a mass storage error occurs. If an error occurs, the channel is released, the sector counter is cleared (SSCT) and control is returned o the caller.

At end-of-information, subroutine PCE (process catalog EOI) is called, the sector counter is cleared (SSCT), and control is returned to the caller.

CATS Read

Subroutine PCR (process catalog read) modifies the LDB subroutine to bypass field length counting, reserves the channel, positions, and calls LDB to read the catalog to the CATS FET buffer. The channel is then released and control is returned to the caller.

If a normal write is indicated in the sector control word, subroutine PCE writes an EOI sector if one has not already been written and then releases the channel and returns to the caller. If a rewrite is indicated, PCE merely releases the channel and returns to the caller.

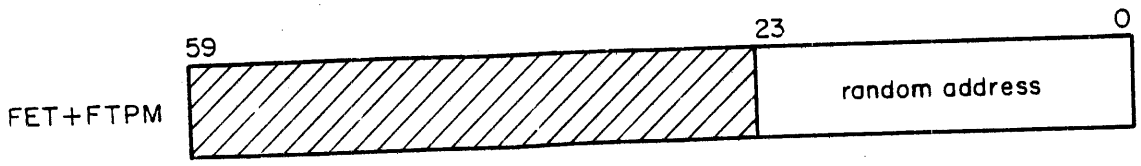
The ty field of the last sector written is used to determine if an EOI sector has already been written. If an EOI sector must be written by PCE, WDS is called with the write last sector option (WLSF). Errors in the write are ignored, to be caught when the next sector of catalog entries is written.

PETS Position

Subroutine PPF (position permit file) reads the FTPM word from the PETS FET and stores the random address in direct cells RI through RI+1. Subroutine CRA (convert random address) is called to compute the track (T6) and sector (T7) for the random address. If the random address is not on the file, the following diagnostic is issued and PFU is aborted. Otherwise, subroutine CIO is called and PPF returns to the main loop.

PFU - PARAMETER ERROR.

The format of FTPM is as follows.



PETS Write

Subroutine PPW (process permit write) calls subroutine SSC to determine if there is data available in the FET buffer. If none is available, PPW returns to the caller. If data is present, the channel is reserved and the permit data transferred by a call to subroutine EMB (empty buffer). PPW loops on calls to EMB until the buffer is empty or a mass storage error occurs.

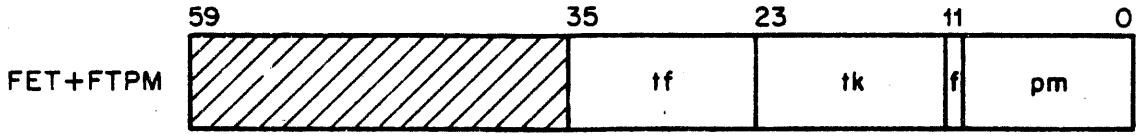
If an error occurs, the channel is released and subroutine STE is called to set an EOI in the TRT. PPW returns to the caller upon return from STE.

When all permit data has been written, PPW checks the beginning of files table for PETS (BEGF) and determines if an EOI has already been written. If not, subroutine WEI is called to write one. PPW then releases the channel, sets the EOI in the TRT by a call to STE, and returns to the main loop.

DATA Position

Subroutine PDF (position data file) checks the direct access flag (DA, set by POF) and, if set calls subroutine EDF (end direct access file) to restore master device controls in DSLA and reset the DATA FST to point to the current indirect chain mass storage position. EDF clears DA before returning.

PDF then sets the beginning of file flag in BEGF and reads the FTPM work from the DATA FET. This word has the following format.



f 0 if track and sector position request:

tf Ignored
tk Track
pm Sector

File is positioned to the sector preceding the tk/pm location

1 if drop and flaw track request:

tf 0 if no track to flaw or track to flaw (if nonzero)
tk First track to be dropped
pm Equipment (lower 6 bits).

Track chain is dropped and track is optionally flawed.

If the request is for a track and sector positioning, PDF calls subroutine TSP (track and sector position) which returns the track (T6) and sector (T7) of the PRU preceding the track and sector specified in FTPM. If the address is not on the file, the diagnostic PFU - PARAMETER ERROR is issued and PFU is aborted. If not aborted, PDF calls subroutine STE to set an EOI at the new track and sector. Subroutine SDS (save data state) is called to save the current state of the data file in word FTDL of the DATA FET. Since the beginning of file flag is set (BEGF), FTDL is set to zero and control is returned to PDF. Subroutine CIO is called and control is returned to the main loop.

If the request is for a drop and flaw track request, PDF calls subroutine DFT (drop and flaw track) which drops the track chain using the DTKM monitor function. If a track is to be flawed, DFT issues an STBM function with option STFS to set the track flaw. The following diagnostic is issued if the track is successfully flawed.

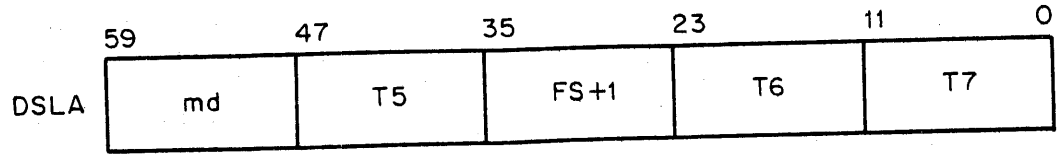
PFU - TRACK FLAWED, EQxx, Tyyyy.

Subroutine SDS is called by PDF to clear FTDL, subroutine CIO is called to complete the operation, and control is returned to the caller.

DATA Write

Subroutine PDW (process data write) begins with a call to SSC (set sector count). If there is no data to be transferred, control is returned to the caller. Otherwise, the control word is read from the FET buffer using a call to subroutine RCW. Subroutine PBF (process beginning of file) is called. If the file has already been started (BEGF is not set, subroutine PBF returns to PDW. Otherwise subroutine PBF begins the file as follows.

Subroutine BDF (begin direct access file) is called if the file is to be a direct access file. Subroutine BDF begins by reading the file name from the catalog entry at the address passed through the control word. Master device mass storage parameters are saved in DSLA as follows.



md	Master device number
T5	Master device equipment
FS+1	First track
T6	Current track
T7	Current sector

BDF then calls subroutine PDA (process direct access file) which completed DSLA by setting the master device (MD), initializes mass storage parameters for the residency device, and sets the direct access flag (DA). Subroutine BDF then calls subroutine RTK (request tracks) to obtain a track chain for the file.

Subroutine BDF builds the system sector for the direct access file, sets the preserved file status (SPFS subfunction to STBM) reserves the channel and writes the system sector. BDF then resets central memory addresses using a call to subroutine SAP, if necessary, writes the DATA file FST entry, and returns to subroutine PBF.

Subroutine BIF (begin indirect access file) is called if the file is an indirect access file. Subroutine BIF increments the current sector and if not at the track sector limit returns to subroutine PBF. If sector limit is reached, subroutine SNT is called to move to the next track. If there is not a next track, subroutine RTK is called to reserve more tracks. When the tracks have been obtained, control returns to PBF.

Having begun the direct or indirect file, subroutine PBF calls subroutine RBA (return beginning address) which writes the beginning track and sector information into the catalog entry in PFLOAD's field length. Subroutine PBF then clears the begin file flag (BEGF) and returns to subroutine PDW.

Upon return from subroutine PBF, subroutine PDW reserves the channel and calls EMB (empty buffer) to transfer the data to mass storage. Control returns from EMB in three cases: buffer empty, mass storage error, or EOI written and buffer not empty. In the first two cases, the BEGF flag is checked to determine if an EOI has been written. If an EOI has not been written, an EOI is forced using a call to WEI, the channel is released, and the EOI is set in the TRT using a call to STE. If an EOI has been written, the channel is released, and subroutine PEF (process end-of-file) is called with IAF processing desired. Control now returns to the main loop. If an EOI is written and the buffer is not empty, subroutine PDW releases the channel and reads the next control word from the buffer using a call to subroutine RCW.

Subroutine PEF (process end of file) is called to end the current file base depending upon what the next file is. If both the current file and the next are indirect access, subroutine PEF returns to the caller. If either file is a direct, subroutine STE is called to set an EOI in the TRT. If the current file is a direct, subroutine EDP is called to restore the DATA file FST to reflect the indirect chain pointers, clear the direct access flag (DA), and reset master device controls from DSLA.

The following decision table shows subroutine PEF operation.

Next File	Current File: IAF	Current File: DAF
IAF	Return	STE/EDP
DAF	STE	STE/EDP

After subroutine PEF has ended the file, PDW proceeds from the process beginning of file step again.

EMB - Empty Buffer

Subroutine EMB transfers data from the FET buffer in the utility's field length and writes that data on mass storage. Since subroutine EMB is used to write all three files (CATS, PETS and DATA) it must be somewhat dependent upon the file being written and the control word associated with it.

After the data has been read from central memory, subroutine EMB advances T7 (current sector) to the next sector. If the next sector is the track sector limit, processing depends upon the file itself. Catalog tracks are not linked in the same manner as other mass storage tracks. If the CATS file is being processed, the control word is checked for the type of operation (byte 0). If the operation is not an EOI (operation not 2), subroutine EMB calls subroutine PCE (process catalog EOI) to write an EOI sector if necessary; subroutine PCL (process catalog linkage) to identify (create, if necessary) the catalog overflow track; requests the channel; and positions to write on the overflow track.

If the track overflow occurs when an EOI is being written, subroutine EMB writes the EOI and continues.

If the current write is not on the catalog file (but still at track limit), subroutine EMB determines if a next track is available, using a call to SNT, and obtains a track chain if necessary.

Subroutine EMB now continues, building track linkage bytes based on the next track, sector, and the type of operation specified in the control word.
The sector is then written to mass storage (a call to WEI is done if writing an EOI). If the utility program has moved (RA

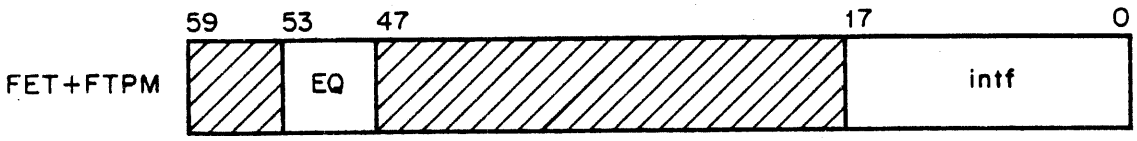
changed), subroutine SAP is called to reset absolute buffer address to reflect the new RA. Subroutine EMB then advances to the next sector or sets the BEGF flag to 1, depending upon the linkage bytes for the sector.

Subroutine EMB now decides to continue or exit. If there are no sectors left to transfer, subroutine SSC is called to determine if more sectors are present in the buffer. If no sectors are present, subroutine EMB returns to the caller with an empty buffer status. If sectors are present, BEGF is interrogated and if nonzero, subroutine EMB returns to the caller with that status. This indicates that an EOI was written but the buffer is not empty. If BEGF is still zero, subroutine EMB loops to read the next block of data from central memory.

The final exit condition is for write errors. All write operations are done with MSEO user error processing (UERP) selected. This allows subroutine EMB to regain control on write errors. Subroutine PWE (process write errors) is called if an error occurs. Subroutine PWE sets the write error code (FTWE=1) into the first word of the FET (bits 17 through 12) and rewrites that word. The EOI flag is set in BEGF (that is, BEGF=1), the FST entry of the file (with the current track and sector updated) is written to the FET at word FTPM and the sector counter (SSCT) for the file is set to reflect the amount of data in the buffer. Subroutine EMB then returns to the caller with write error status set.

STU - SET PF UTILITY INTERLOCK

Function 05 (CTSU) sets permanent file utility interlocks. Upon entry, FET + FTPM has the following format.



eq	Equipment to set utility interlock on
intf	Interlock flag address; set to one when the PF utility interlock is set; set to two if PFU has to go on recall to wait for no PF activity

STU attempts to set the permanent file utility interlock for the equipment specified. STU uses subroutine FAR (force auto recall) to ensure that PFU is called with autorecall. This is done so that PFU may reissue the STU request to RA+1 if the attempt to set the utility interlock fails. If PFU is not called with autorecall the following diagnostic is issued and PFU is aborted.

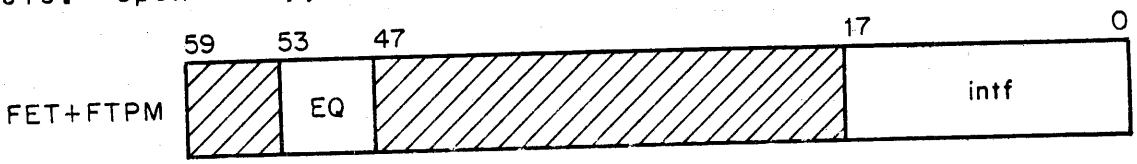
PFU - PARAMETER ERROR.

The attempt to obtain the utility interlock is made by a call to subroutine SUT (COMPSUT - set utility interlock). If the utility interlock is successfully set, subroutine WIF (write interlock flag) is called to set the interlock flag to one and then PFU is dropped. If the interlock is not set, the STU request is rewritten to RA+1 and PFU is dropped. If subroutine SUT sets the request for PF system interlock as part of its processing, the interlock flag in CM is set to two before the STU function is placed on recall. The interlock flag enables the calling utility to determine when an interlock is actually set and to clear it should an abort occur.

Subroutine SUT returns status to the STU function if the utility interlock could not be set because it was already set or if PF system activity prohibits setting the interlock. If system activity prohibits the interlock, subroutine SUT toggles the request for PF system interlock bit. Toggling this bit causes alternating periods of time when the bit is set and when it is clear. Thus new PF activity is alternately disabled and enabled for one second intervals (PP recall delay) until the utility interlock can be set. If the request for PF system interlock bit were set every time the STU function were recalled, activities such as a PFDUMP could cause STU to lock out all new permanent file activity for considerable periods of time.

CLU - CLEAR PF UTILITY INTERLOCK

CLU, function 06 (CTCU), is used to clear the interlock set by STU. Upon entry, FET + FTPM has the following format.



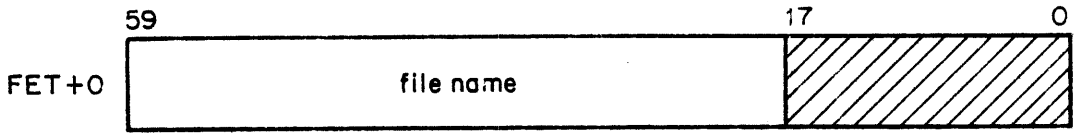
eq Equipment to clear utility interlock on

intf Interlock flag address; if this word is equal to one the utility interlock is cleared, otherwise the request for PF system interlock will be cleared; the word is set to zero when one of the interlocks is cleared

If the interlock word is 1, CLU clears the interlock by a call to subroutine CUT (clear utility interlock). Otherwise, the permanent file system interlock request is cleared by a call to subroutine PIR (process interlock request) with option CIRS (clear request for interlock). PIR is part of common deck COMPSUT. In all cases the interlock flag word is cleared by a call to WIF.

RCF - REWIND CATALOG FILE

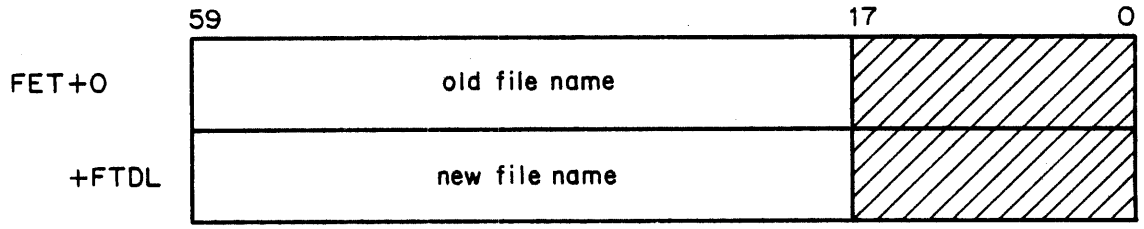
Function 07 (CTRC) rewinds catalog files. Upon entry FET+0 is formatted as follows.



This function is used rather than a CIO rewind since the catalog file (CATS) does not have system sectors (that is, catalog data begins in sector zero). The current sector is set to 0 and current track is set to the first track in the FST.

CHF - CHANGE FILE NAME

Function 10 (CTCF) changes the file name. Upon entry, FET+0 and FET+FTDL are formatted as follows.

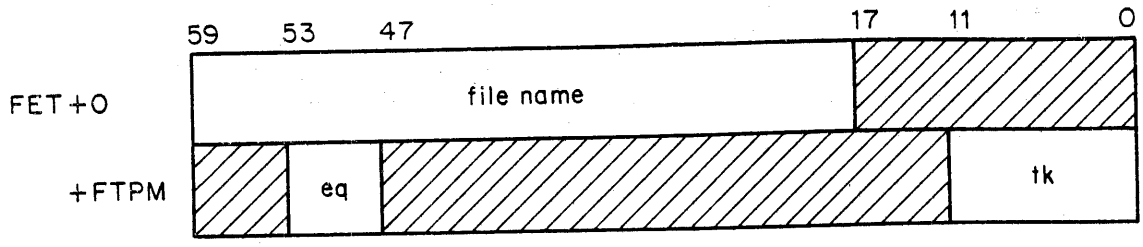


CHF finds the old file name in the FNT, replaces the file name with the new file name, and replaces the old file name with the new file name in the FET.

If the file is not found, one is created.

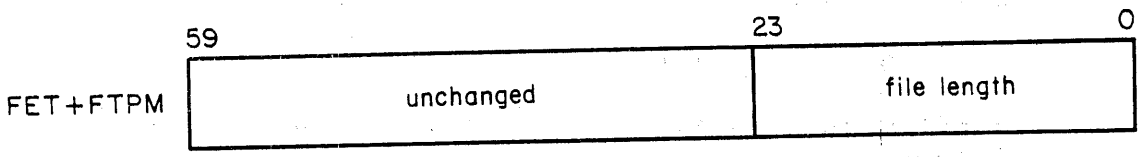
SFL - SET FILE LENGTH

Function 11 (CTFL) determines the length of a file. Upon entry, FET+0 and FET+FTPM have the following format.



eq Equipment file resides on
tk First track of file; if zero the first track is taken from the FST for the file

Upon exit, FET+FTPM has the following format.



SFL finds the file in the FNT. If TK is not specified, the first track field of the FST is used. The equipment for the file is verified to ensure that it is a mass storage device. If it is not mass storage, the following diagnostic is issued and PFU is aborted.

PFU - PARAMETER ERROR.

If not aborted, SFL then calls subroutine SEI (COMPSEI - search for end of information) and returns the length of the file from the track specified to the EOI.

SEC - SET CATALOG TRACK INTERLOCK

Function 12 (CTSC) sets the catalog track interlock. Upon entry, FET+0 contains the file name, left-justified.

SEC finds the file in the FNT and ensures that the equipment for the file is mass storage. If it is not mass storage, the following diagnostic is issued and PFU is aborted.

PFU - PARAMETER ERROR.

If bit 9 is set in the FST entry, the track is already interlocked and PFU aborts with the following diagnostic.

```
PFU - TRACK INTERLOCK ALREADY SET ON filenam AT address.
```

If the track is not interlocked for the file, SEC calls subroutine STI (COMPSTI - set track interlock) to set the track interlock. If successful, bit 9 is set in the FST. If the interlock was not set, PFU is aborted with the following diagnostic.

```
PFU ABORTED.
```

CLC - CLEAR CATALOG TRACK INTERLOCK

Function 13 (CTCC) is used to clear the interlock set by SEC. Upon entry, FET+0 contains the file name, left-justified. The equipment and first track of the file define the track to clear the interlock on.

CLC finds the file in the FNT and ensures that the equipment for the file is mass storage. If it is not mass storage, the following diagnostic is issued and PFU is aborted.

```
PFU - PARAMETER ERROR.
```

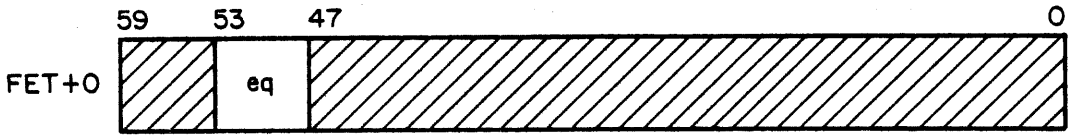
If bit 9 in the FST is not set, PFU aborts with the following diagnostic.

```
PFU - TRACK INTERLOCK ALREADY CLEAR ON filenam AT addr.
```

Otherwise the interlock is cleared by a call to common deck COMPCTI. The interlock bit in the FST (bit 9) is also cleared.

SES - SET ERROR IDLE STATUS

Function 14 (CTBI) sets error idle status. Upon entry, FET+0 has the following format.

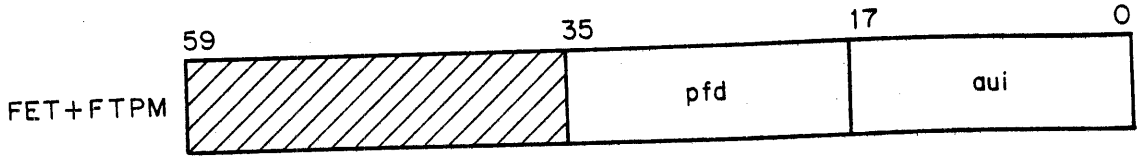


eq Equipment to set error idle status on

SES sets the error idle bit in the MST using the SGBS subfunction of the STBM monitor function.

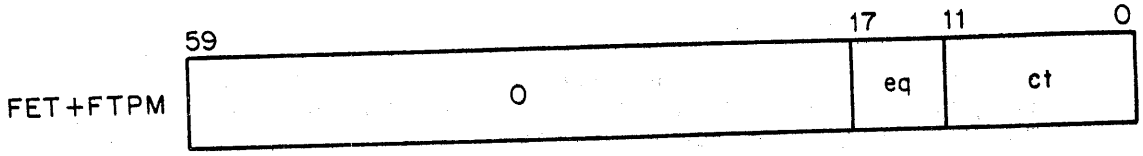
LCT - LOCATE CATALOG TRACK

Function 15 (CTCT) locates the catalog track. Upon entry, FET + FTPM has the following format.



pfd Address of permanent file description word
aui Address of user index word

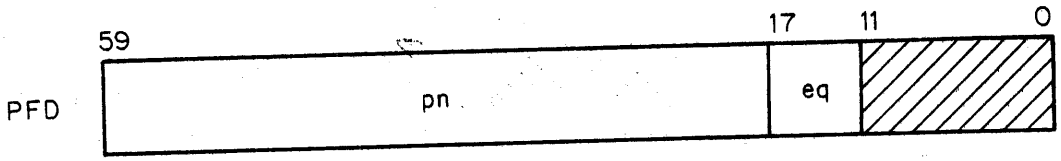
Upon exit, FET+FTPM has the following format.



eq Equipment number of master device
ct Catalog track of user

LCT makes a call to subroutine SCT, which returns the equipment and catalog track for the given user index.

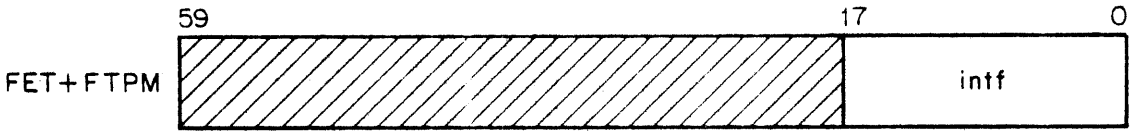
The format of the Permanent File Description word is as follows.



pn Pack name if upper byte is nonzero; otherwise ignored
eq Equipment number of any family member if upper byte of pack name is zero; otherwise ignored

IAC - INCREMENT PF ACTIVITY COUNT

Function 16 (CTIA) increments the permanent file activity count. Upon entry, FET+FTPM has the following format.



intf Interlock flag address; this word is set to one when the PF activity count is incremented.

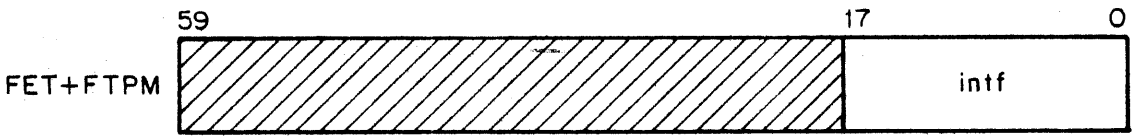
IAC verifies that PFU has been called with autorecall by a call to subroutine FAR.

IAC requests that the permanent file activity count be incremented by using the IPAS option to subroutine PIR. If the interlock is not obtained, PFU goes on recall using a call to RPP.

If the interlock was obtained, the interlock flag is set using a call to subroutine WIF.

DAC - DECREMENT PF ACTIVITY COUNT

Function 17 (CTDA) is used to decrement the activity count set by IAC. Upon entry, FET+FTPM has the following format.

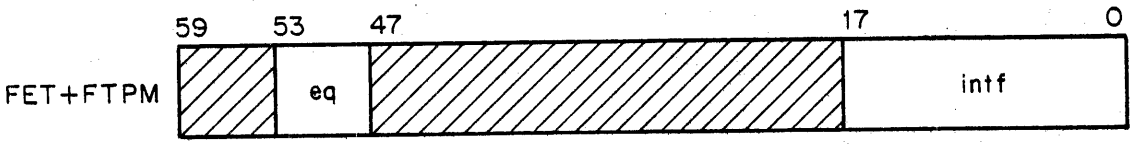


intf Interlock flag address; this word is set to zero when the PF activity count is decremented

DAC requests that the permanent file activity count be decremented using the DPAS option to subroutine PIR. The interlock flag is then cleared using a call to subroutine WIF.

TSU - TEST PFU INTERLOCK

Function 20 (CTTU) tests for a permanent file utility interlock. Upon entry, FET+FTPM has the following format.



eq Equipment to test utility interlock on

intf Interlock flag address; this word is set to one if the utility interlock is set and to zero if the interlock is clear.

TSU obtains the Utility interlock status of the specified device using the TGBS subfunction of the STBM monitor function. The 4-bit utility interlock mask field is extracted from the STBM return status and written to the interlock flag word using subroutine WIF.

PF UTILITY PROGRAMS

The Permanent File utility programs have entry points PFATC1, PFCAT1, PFCOPY1, PFDUMP1, and PFLOAD1 and are found in decks PFATC, PFCAT, PFCOPY, PFDUMP, and PFLOAD, respectively.

The key to the output listings produced by these utilities is common deck COMCFCE - format catalog entry for output. FCE converts and formats a catalog entry into two 70 character lines for output. Default header lines may be retrieved by the calling program, if desired, by referencing the locations CHDR1 and CHDR2. The caller may optionally request that the file length be formatted in characters or sectors, the user index be formatted, or a four decimal digit line number be attached to the beginning of the line.

FCE maps the internal codes for access modes, file types, permission modes and the creating subsystem type into appropriate display code equivalences.

COMCMAC macro EDCAT calls FCE. This macro has the following format.

LOCATION	OPERATION	VARIABLE SUBFIELDS
	EDCAT	caddr,waddr,lnum,len,index

caddr	Address of catalog entry
waddr	Address of working storage area (14 CM words) to return edited catalog entry to
lnum	Line number to be added to beginning of first line
len	Length code (1 = sectors, 0 = characters)
index	Nonzero if index field is to be listed

INTERLOCKS

The permanent file utilities make use of the following interlock and activity controls available in NOS.

Permanent File Activity Count

The permanent file activity count is a count of the number of routines currently accessing the permanent file system. This count is maintained in the PFNL word in low core CMR.

Permanent File Utility Interlock

The permanent file utility interlock is a bit that is set in the MST for a device to indicate that a recovery, initialization or permanent file loading activity is being performed on that device. The setting of the permanent file utility interlock means that the routine requesting the interlock requires exclusive usage of the device. When set, the permanent file utility interlock ensures that no new permanent file activity will be initiated on a device. In order to ensure that no activity actually exists on the device, the PF system interlock is obtained while the utility interlock is being set.

Total PF Interlock

The total or system permanent file interlock is an interlock set in PFNL that prohibits permanent file activity. This interlock can only be obtained when the permanent file activity count is zero.

Catalog Track Interlock

The catalog track interlock is set on a particular catalog track to insure that neither the catalog track nor any of the files pointed to by the catalog entries in that catalog track are modified.

PFU functions are used by the utility programs to secure the interlock or manipulate the activity count.

PFATC UTILITY

PFATC is a permanent file utility program that catalogs archive files.

PFATC reads the archive file and lists the catalog entry by an EDCAT macro call (LO = T option-default) and list the catalog image record (CIR) (LO = C option).

Samples of PFATC output may be found in the NOS System Maintenance Reference Manual.

The main routine of PFATC is flowcharted in figure 36-4.

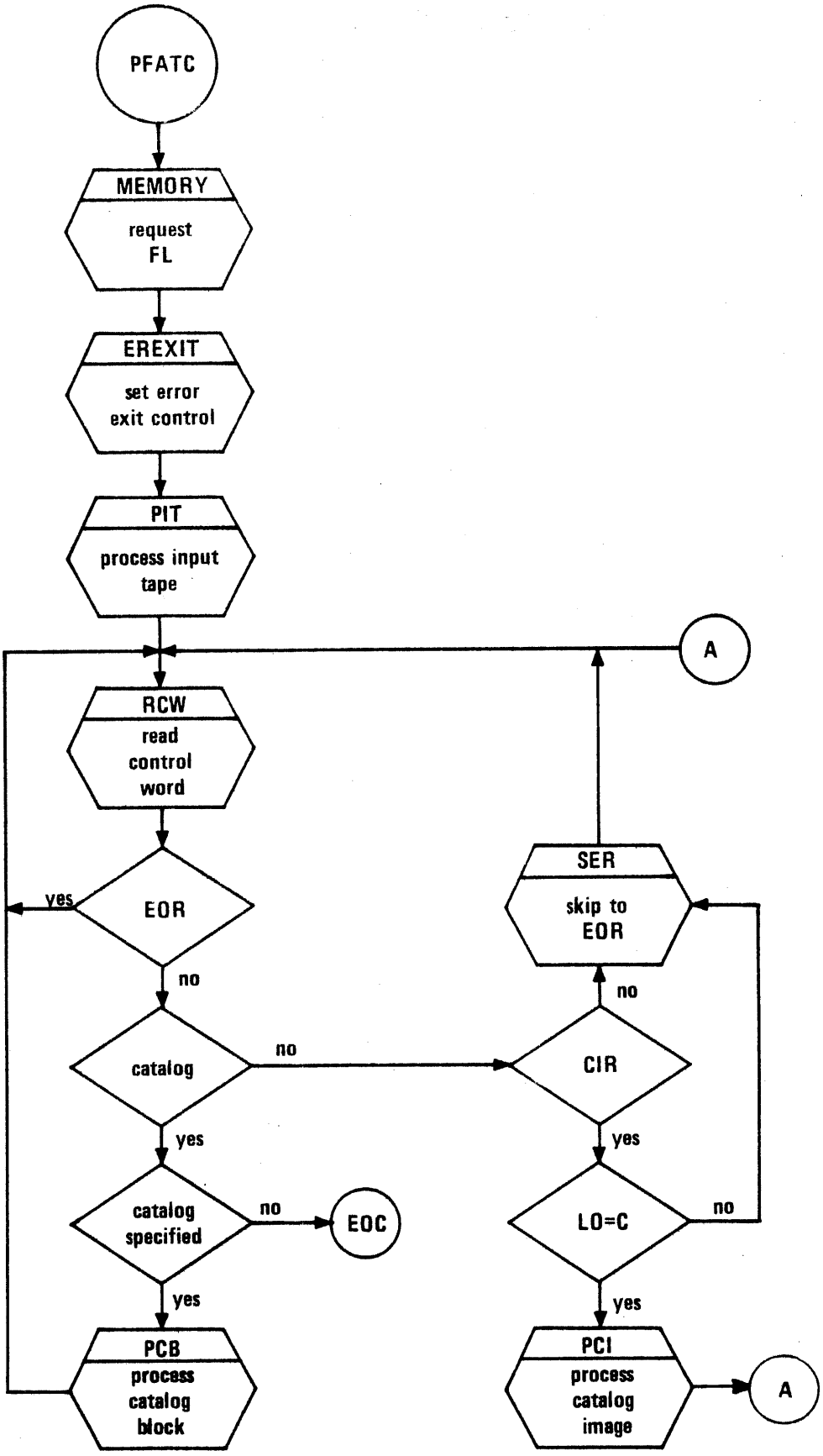


Figure 36-4. PFATC

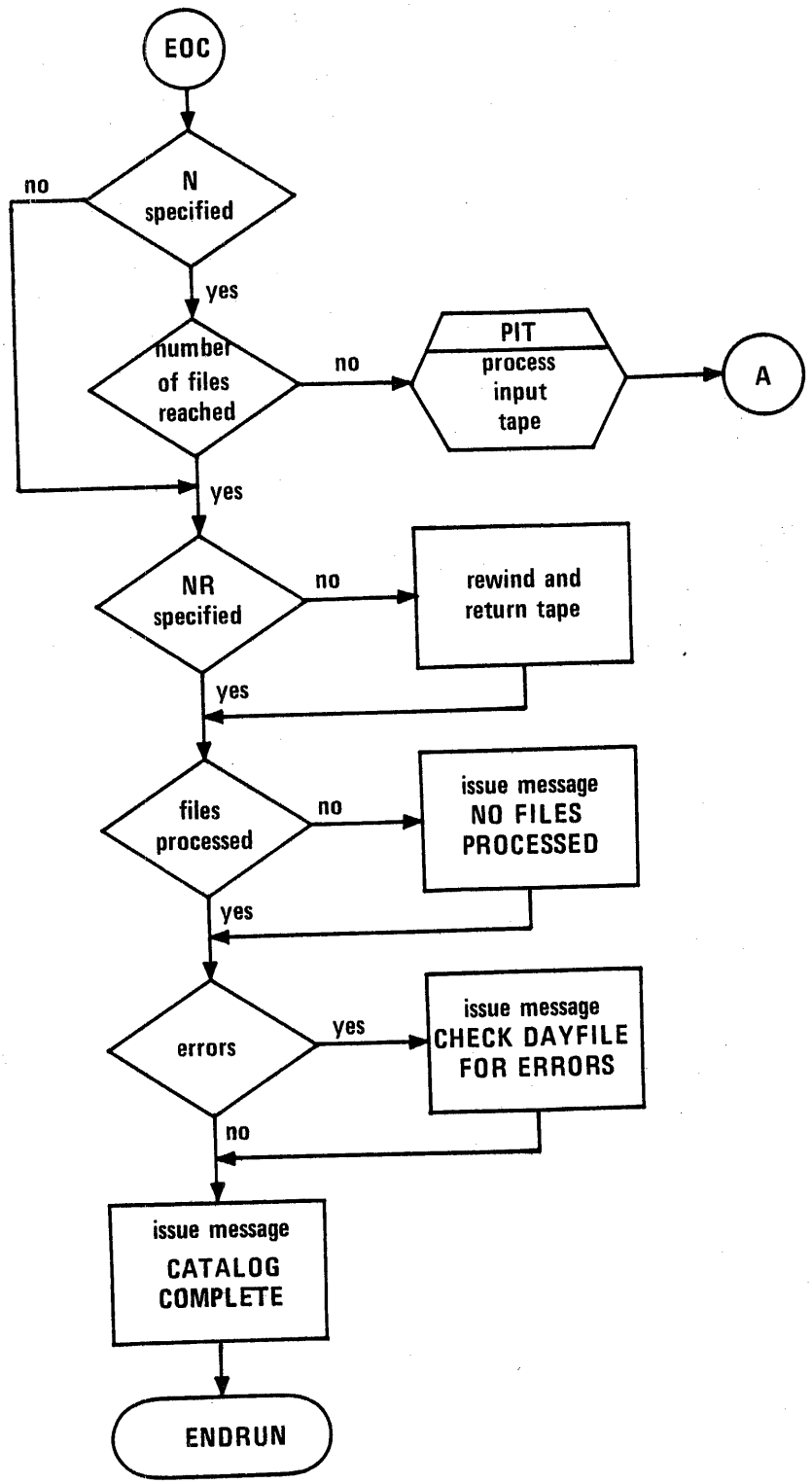


Figure 36-4. PFATC (Continued)

PFCAT UTILITY

PFCAT is a permanent file utility program which generates reports from the permanent file catalog tracks on a master device. The reports generated are as follows.

- Listing of catalog file with files grouped by user index (LO=T).
- Statistical report of device usage (LO=S).

In the case of the first report (LO=T), a status listing of the mass storage configuration showing the usage of each mass storage device is also printed.

A flowchart of PFCAT is shown in figure 36-5.

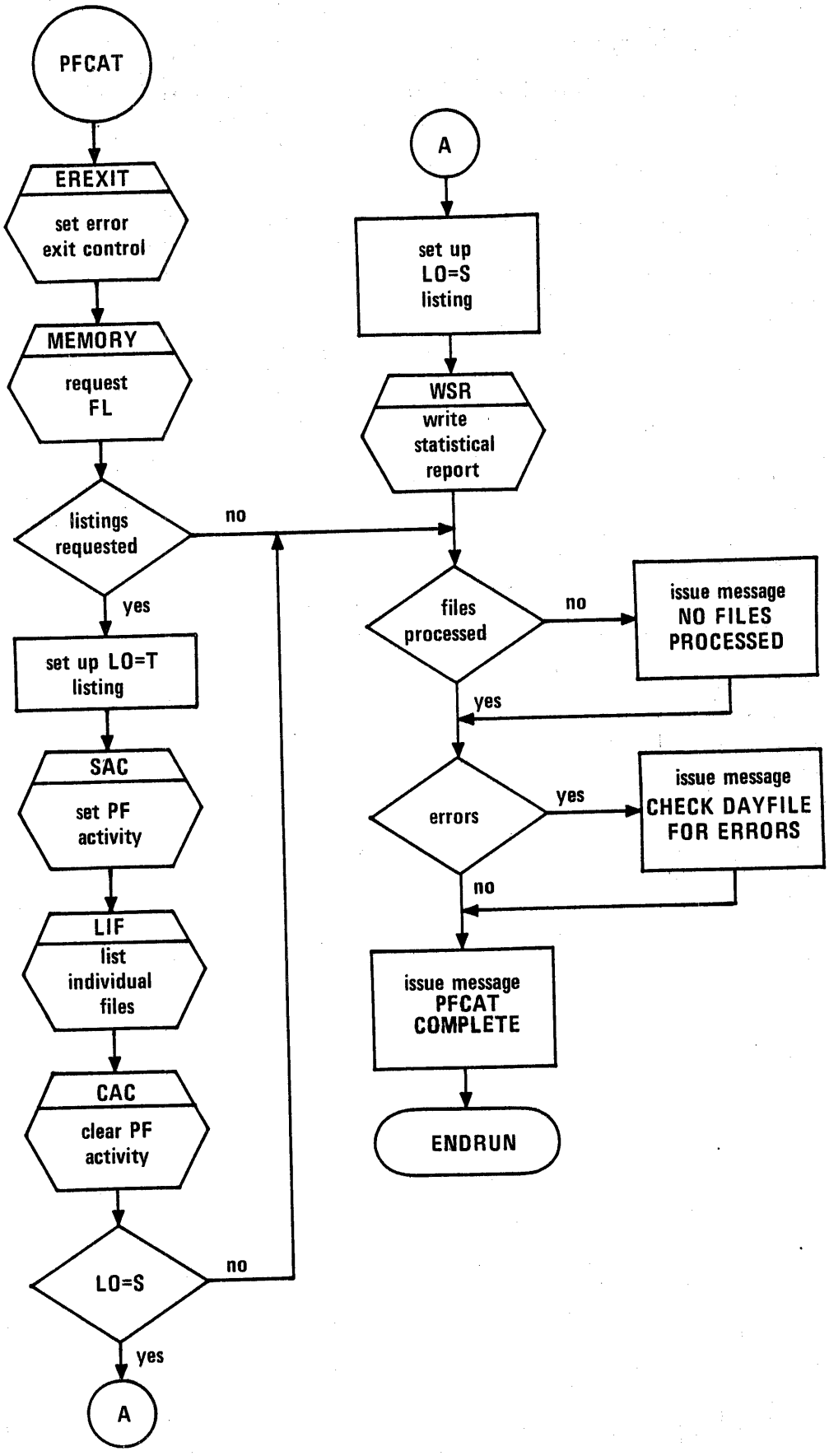


Figure 36-5. PFCAT

PFCOPY UTILITY

PFCOPY is a permanent file utility program that extracts files from an archive file and copies them to one or more local files.

PFCOPY copies files from the archive tape to local files at the control point. If a master file (MF) has been specified, all files are copied to that local file, otherwise files are copied to local files having the same name. Each file may be preceded by its catalog and permit entries if the Q option had been specified.

PFCOPY uses the CTCF PFU function to change names of various files at the control point. PFCOPY is the only utility that uses CTCF and does so only to avoid conflict between data file names and the archive file name.

A flowchart of PFCOPY is shown in figure 36-6.

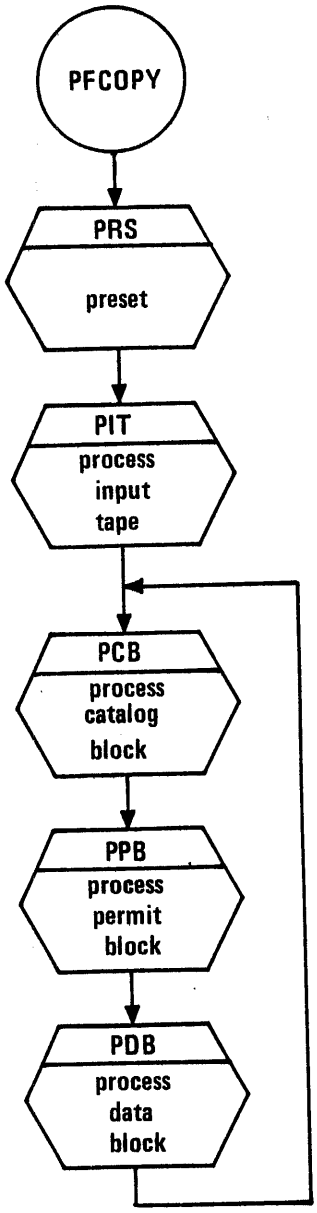


Figure 36-6. PFCOPY

PFDUMP UTILITY

PFDUMP is a permanent file utility that copies files stored on a permanent file device to a backup storage (archive) file. The files archived by PFDUMP may be restored by the PFLOAD utility.

The dumping of permanent files has many options. A full dump is a dump of the entire family or removable pack in which only the family or pack name is specified (no options selected by the OP keyword. A selective or incremental dump is a dump of a family, device, or removable pack in which the modification date option (OP=M) has been specified. If any option other than M is specified, the dump produced is said to be a partial dump.

Permanent file dumps do not require the system to be idle. However, as each catalog track is being processed, no user may access any files defined on that catalog track.

A flowchart of PFDUMP main routine is shown in figure 36-7.

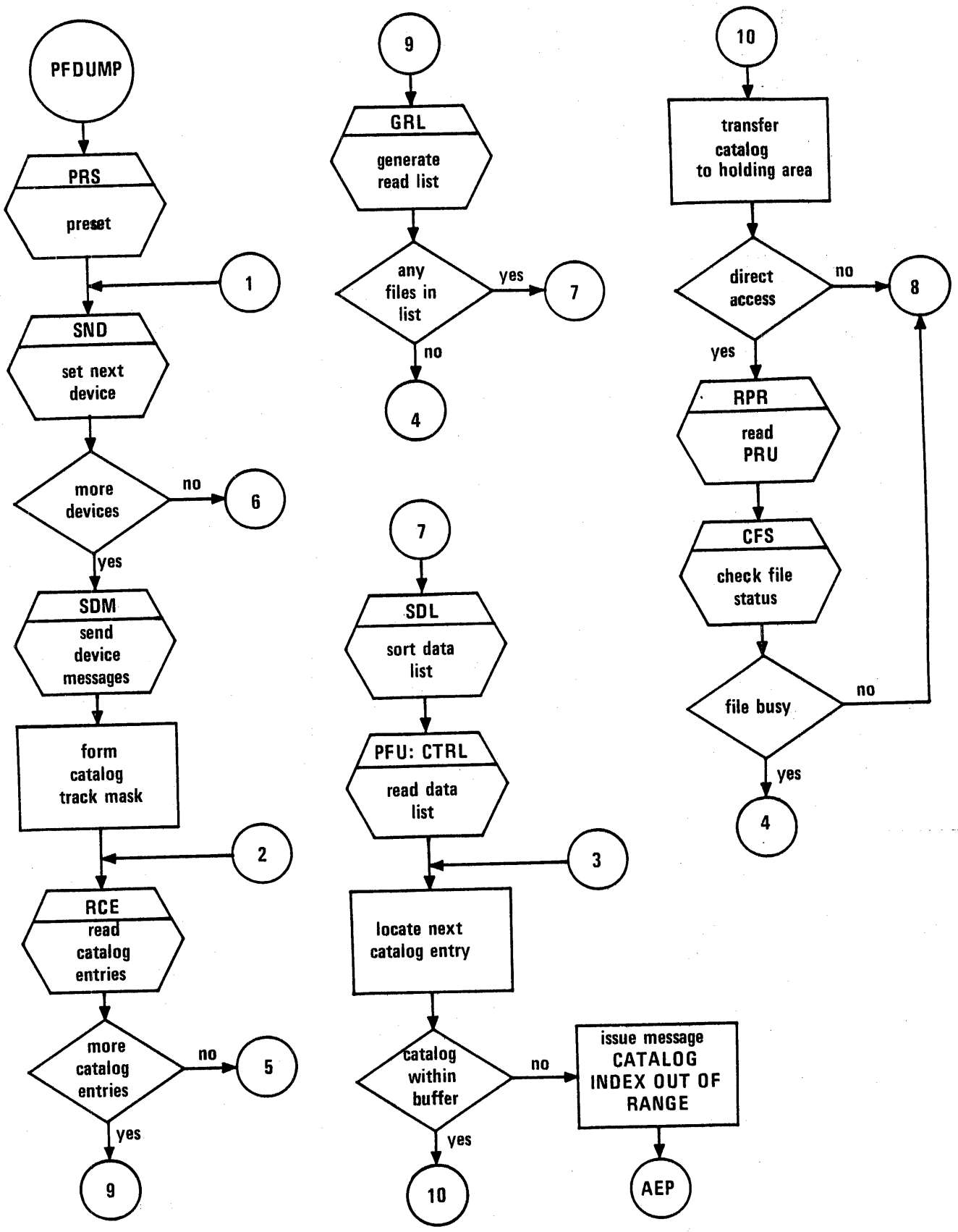


Figure 36-7. PFDUMP

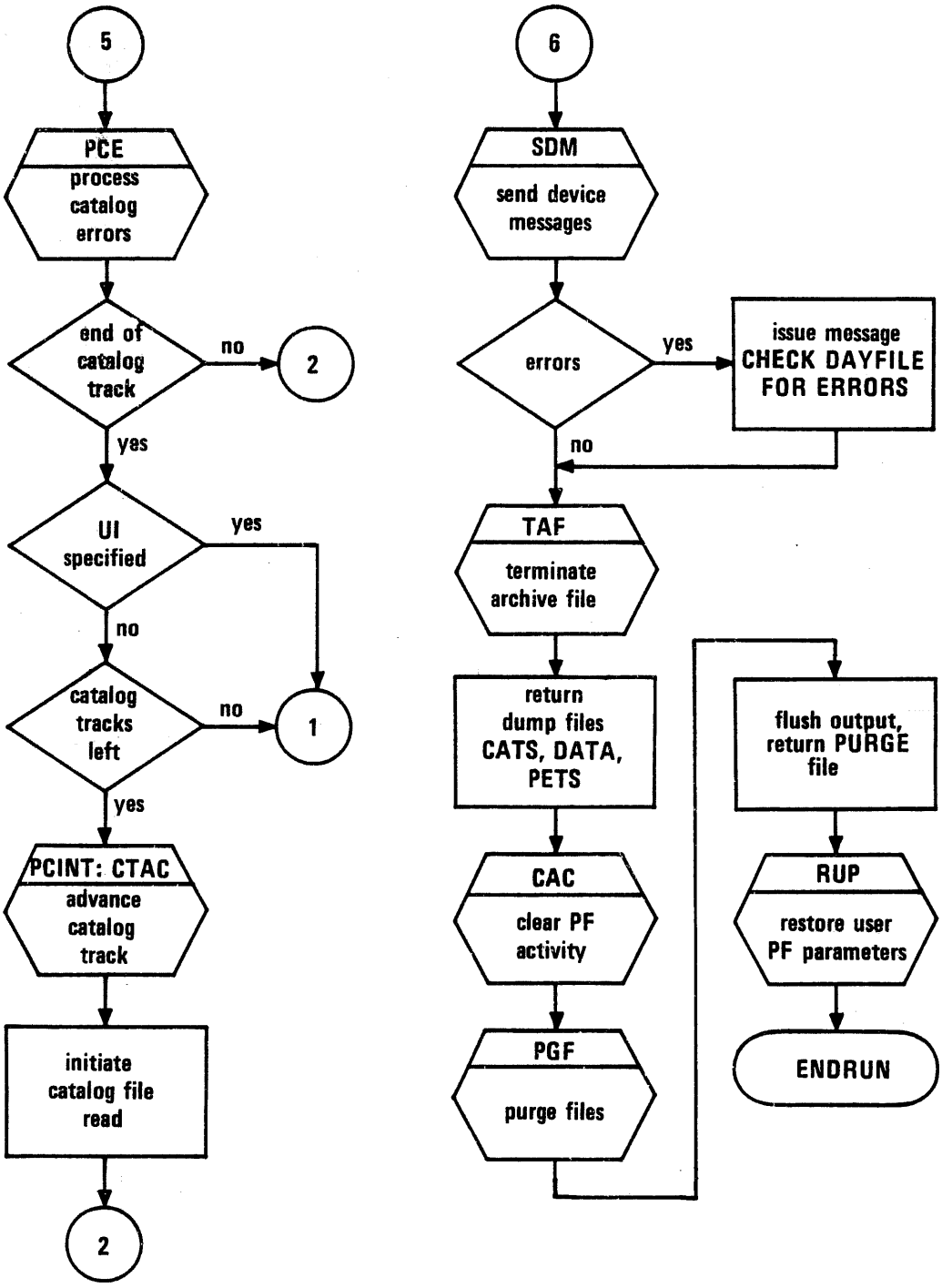


Figure 36-7. PFDUMP (Continued)

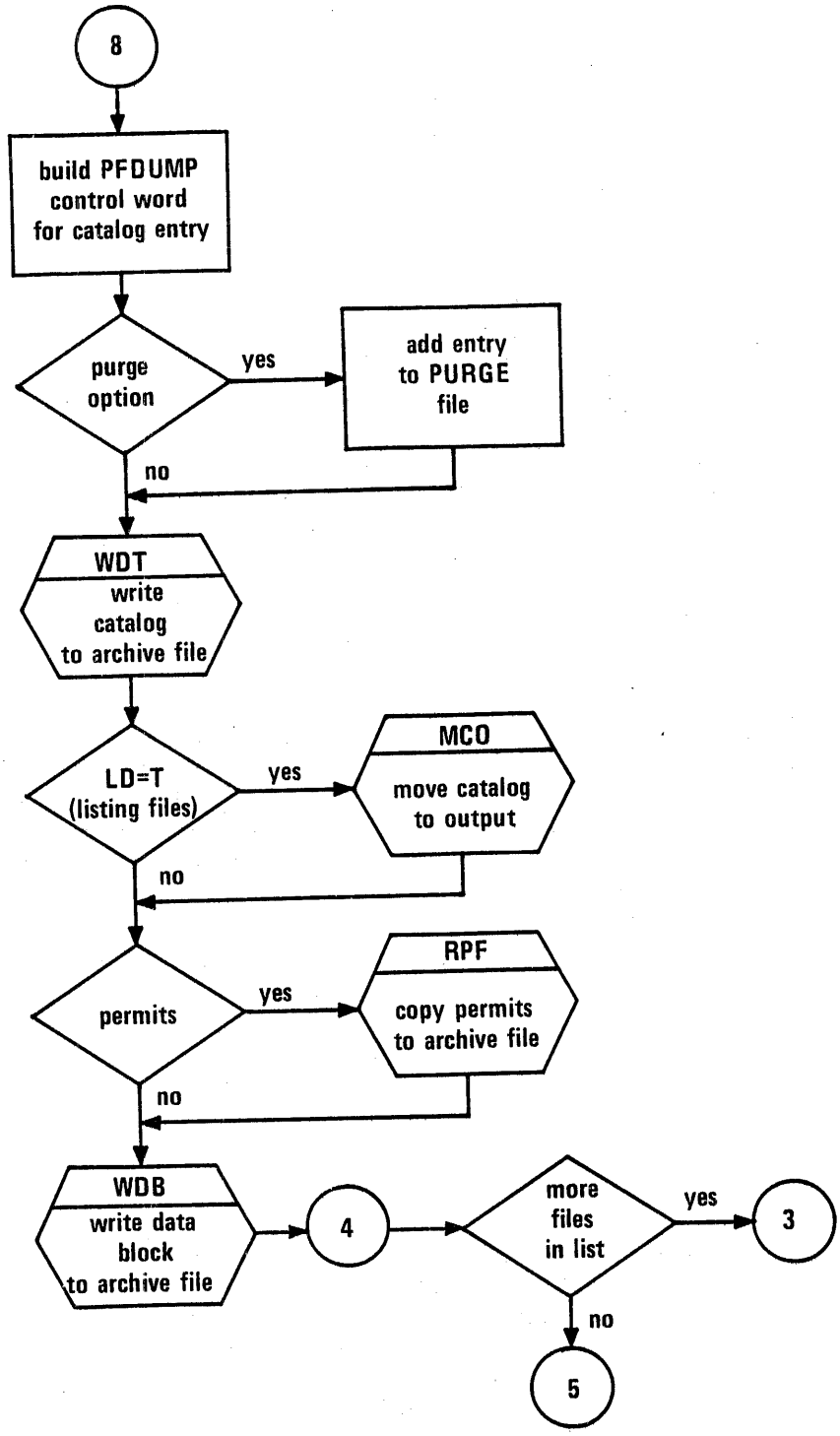


Figure 36-7. PFDUMP (Continued)

PFDUMP obtains information and data for a permanent file by using PFU calls and writes this information on an archive file that is usually a magnetic tape file.

This discussion of PFDUMP concentrates on the following areas.

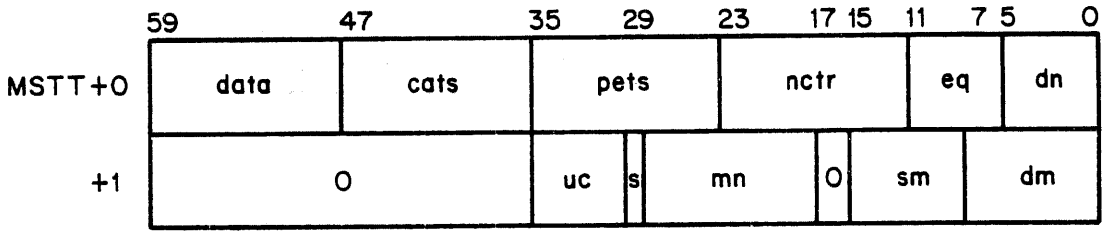
- Obtaining the file from the permanent file subsystem
- Writing the archive file
- Error processing
- Interlocking

Obtaining the File

Obtaining the file from the permanent file system consists of device selection and file selection. Device selection is done during PFDUMP preset based on information read from the EST and MST contained in CMR. File selection is performed during dumping and is based on information read from the permanent file catalogs and direct access file system sectors.

Device Selection

Device selection consists of subroutines BMT (build mass storage tables), SBS (set boolean selection parameters), SSP (set selection parameters), and SMP (set master device parameters). These subroutines build or access the MSTT (table of MST information) which is a table of information extracted from the MST for each device under consideration. This table has the following format.



data	First track of indirect data chain
cats	First catalog track
pets	First track of permit chain
nctr	Number of catalog tracks
eq	Equipment number of device
dn	Device number
uc	Unit count - 1
s	On/off status
mn	Equipment mnemonic
sm	Secondary mask (direct access files)
dm	Device mask (indirect access files and catalogs)

The MSTT is built by subroutine BMT. BMT reads the EST and MSTs for mass storage devices using the RSB monitor function. A device is included in the MSTT if it meets family, pack name and user number selections (keywords FM, PN, and UN) supplied in the utility call. The MSTT is terminated by two zero words.

Subroutine SBS correlates the device selections supplied in the utility call (keywords DN, TD, DI, and DD) to devices entered in the MSTT. The output from this correlation is a list of boolean file selection parameters (BFSP). Each item in the list is a boolean selection property and is represented by a bit in the word BFSP. For example, the condition device number was specified (DN=xx) is represented by the bit position defined by symbol DNSP. If the device number was specified, bit position DNSP is set to 1 (true). If not, bit position DNSP is set to 0 (false).

The symbols and their conditions are as follows.

Symbol	Condition
DAFO	Direct access files only (OP=D)
DIDN	DI cataloged on DN
DISP	DI specifed (DI=xx)
DITD	DI cataloged on TD
DNMD	DN is a master device
DNSP	DN specified (DN=xx)
DNTD	DN same as TD
FNFS	Force no files selected
IAFO	Indirect access files only (OP=I)
TDMD	TD is a master device
TDSP	TD specified (TD=xx)

In addition to producing the BFSP list, SBS modifies the destination device converted parameter (CPDD) to include the address of the MSTT entry of the destination device if one was specified.

Subroutine SSP interrogates BFSP and sets the single device status (FSSD), alternate device selection (FSAD), secondary alternate device selection (FSSA) and sets the first entry in the MSTT to the first device to be processed. The values set by SSP are used primarily by subroutine SND (set next device) to select the next device to be dumped.

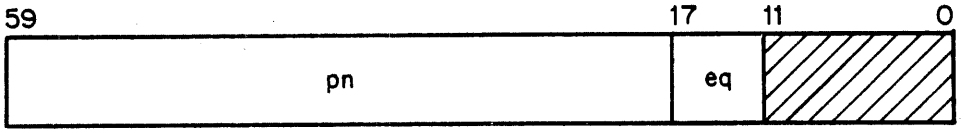
The alternate device selection, when nonzero, requires that a file be resident on the alternate device. If the alternate device selection is zero after the first master device has been processed, the secondary alternate device selection replaces it. Thus if a secondary alternate device selection is present but the alternate device selection is not, files residing anywhere are selected from the first master device, but only files resident on the secondary alternate device are selected for subsequent master devices. The secondary alternate device selection is used for processing the true device option (TD).

SSP calls subroutine EBE (evaluate boolean expression) to determine if the desired condition has been properly selected;

that is, appropriate bits are set in BFSP that agree with a file selection boolean expression. The selection expression is built through a macro (BOOL) using the boolean file selection variables described above. Each expression is referenced by a location where the mask is stored for the bits that must be set in BFSP for the condition to have been met. The expressions are as follows.

Symbol	Condition Tested
SDFL	Single device selected
ADDN	DN is alternate device
ADTD	TD is alternate device
SATD	TD is secondary alternate device
FMDN	DN is first master device
FMTD	TD is first master device
FMDI	DI is first master device
NFSL	No files selected

Subroutine SMP builds the permanent file description word (PDWD) for the device being dumped and sets the number of catalog tracks (NCAT) and device mask (MASK) using the MSTT and single device selection (FSSD). The PDWD has the following format.



pn O if a family dump; pack name if an auxiliary device dump

eq Equipment number of a family member if a family dump; O if an auxiliary device dump

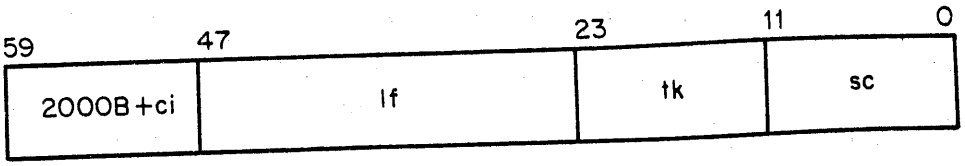
If an auxiliary device is being dumped, the pack name, unit count, and equipment mnemonic are set in the PURGF FET so that files may be properly purged if the OP=P option is selected.

File Selection

Subroutine RCE (read catalog entries), GRL (generate read list) and SDL (sort data list) are used to prepare a list of files to be read using the PFU function CTRL. Subroutine RCE reads catalog entries from the CATS file into a working buffer. These catalog entries are analyzed by GRL to build a list of files meeting the selection criteria for the dump. Subroutine SDL then sorts the list in order by track and sector to minimize mass storage positioning.

Each file is analyzed by subroutine GRL to determine if it meets the selection parameters specified for the dump. The direct only (OP=D) and indirect only (OP=I) selection

parameters are evaluated by GRL. Subroutine CSP (check selective parameters) is called to evaluate the user index (UI), permanent file name (PF), device residency (ON and TD), and date/time (OP=B,M,C,A,DT, and TM) selection parameters. For each file selected, GRL builds a read list entry in the following format.



ci	Catalog index into catalog buffer
lf	Length of file (indirect access only)
tk	First track of file
sc	First sector for indirect access files; for direct access files, bit 11 is set, bit 10 is zero for normal processing or one for a forced dump (no date checking), bits 9 through 6 are zero, and bits 5 through zero are the residency device number

For direct access files, GRL also calls subroutine CDS (check device status) to verify that the residency device for the file exists in the system (MSTT). If not, the following diagnostic is issued and the file is skipped.

```
PFDUMP - DEVICE NOT FOUND, FN=nnnnnnn,UI=uuuuuu,DN=dd.
```

A final selection process is peformed for direct access files when PFU reads the system sector for the file. For incremental dumps (OP=M) the last modification date is passed to PFU as a read function (CTRL) parameter. PFU compares that date with the last modification date in the system sector of each direct access file encountered in the read list. A system sector control word is written to the data file buffer to indicate the status of this comparison and other possible error conditions. The file is copied to the buffer only if the modification date is satisfied and no error condition is detected. PFDUMP reads the system sector control word using subroutine RPR (read PRU) and analyzes it using subroutine CFS (check file status). Error conditions indicated in the control word are identified with one of the following diagnostics.

```
DAF BUSY.
DAF ZERO LENGTH.
BAD SYSTEM SECTOR.
```

Selecting a Device to Dump

Subroutine SND (set next device) is called from the main loop of PFDUMP to select the next device to be dumped.

On the first call, SND scans the table of MST information (MSTT) for the first master device. If a master device is found, information from MSTT is used to set the master device description (MDDS), catalog description user index (PDUI), master equipment number (MAEQ), number of catalog tracks (NCAT), and the master device number (CPAR+CPDN). The files CATS, PETS, and DATA are set up for the master device using calls to the PFU function CTOL. The catalog track interlock is set according to PDUI using the CTSC option of the PCINT (process catalog interlock) macro. Status is then returned to the caller indicating the device number to be processed (-1 for an auxiliary device) or zero if no master device was found. For alternate device dumps the device number returned is the alternate device rather than the master device.

On subsequent calls, SND clears the catalog track interlock for the master device being processed. If a single device dump is not being performed, MSTT is scanned for the next master device. If another master device is found, the master device parameters are set up, the CATS, PETS, and DATA files are opened and the catalog track interlock is set for the new master device. As in the first call, status is returned, indicating the device number to be processed (-1 for an auxiliary device) or zero if no more master devices remain to be processed.

Writing the Archive File

The archive file is assigned to the control point using the LABEL macro. If the archive file has not been pre-assigned (assigned to the control point prior to the utility call), the LABEL macro requests a tape having the specified track type with the installation density. A tape, mass storage, or null equipment may be assigned to the archive file by the console operator in response to the flashing label request. The file is opened with read/no rewind and the file ID is set to 0. This same logic is followed for the verification file if one is desired (keyword V and VF). Both files are rewound unless the no rewind (NR) option is selected.

Writing and positioning of the archive file (and the archive verify file if necessary) is accomplished using the ARCHIVE macro. The type of operation to be performed is selected using the function code option of the macro. The ARCHIVE macro has the following format.

LOCATION	OPERATION	VARIABLE SUBFIELDS
	ARCHIVE	fnc,p1,p2

fnc Function code mnemonic:

WRITEW	Write words on archive file; p1 = working buffer address, p2 = word count
WRITER	Write EOR on archive file (no parameters)
WRITEF	Write EOF on archive file (no parameters)
SKIPFF	Skip archive file forward; p1 = file count to skip
FLUSH	Flush archive file buffer (no parameters)

p1 First parameter

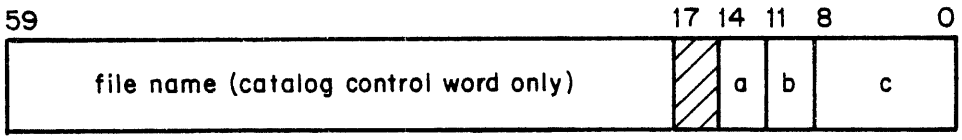
p2 Second parameter

The ARCHIVE macro sets up the appropriate parameters in registers and calls subroutine PAF (process archive file operation). PAF processes the WRITEW, WRITER, and WRITEF options of the ARCHIVE macro using the PFDUMP defined CWWRITE macro. The CWWRITE macro in turn calls subroutine CWW (control word write) which interfaces similarly to the system common deck COMCWTW but performs control word I/O rather than buffer write I/O.

The SKIPFF option is processed using the SKIPFF system macro. The FLUSH option is processed using a call to the subroutine FCW (flush buffer using control word write) which issues the WRITECW system macro if there is still data in the buffer.

Archive File Control Words

Information written to the archive file is divided into blocks. The first word of each block is a control word which identifies the number of data words in the block and the type of information contained in those data words. The format of the control words is as follows.



a One of the following:

0	Label
1	Catalog
2	Permit
3	Data
4	Not used
5	CIR (catalog image record)
7	End of dump

b One of the following:

0	Data
1	EOR
2	EOF
7	End of dump

c Number of words until next control word (excludes control word)

A symbol is defined for the more common combinations of the a and b fields for control words. These values are as follows.

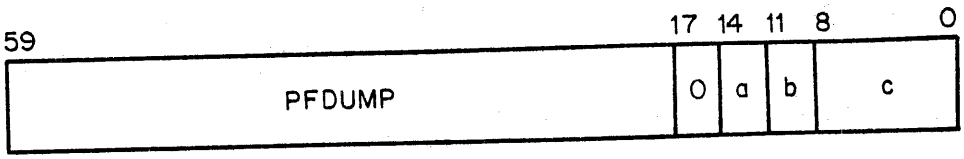
Symbol	Value (ab)	Definition
LCWC	01000	Label control word
CCWC	11000	Catalog control word
PMCW	20000	Permit control word
PRCW	21000	Permit record control word
DCWC	30000	Data control word
DRCW	31000	Data record control word
DFCW	32000	Data file control word
CICW	50000	Catalog image control word
CRCW	51000	Catalog image record control word
CFCW	52000	Catalog image file control word
EODC	77000	End-of dump control word

Archive File Label

The archive file has a data label that describes the parameters used in the PFDUMP call. The data label is generated by subroutine SLP (set label parameters) and written to the archive file by subroutine LBL (write PFDUMP archive file label) during the preset phase of PFDUMP.

Subroutine SLP copies the converted parameters frm CPAR to the data label starting at the sixth word (figure 36-8). The specified number of files (SK parameter) are skipped by subroutine LBL before the data label is written to the archive file.

The data label is written to the archive file as a record by itself and is preceded by a control word in the following format.



a Label type control word (0)
b End of label flag (1)
c Word count for label (100)

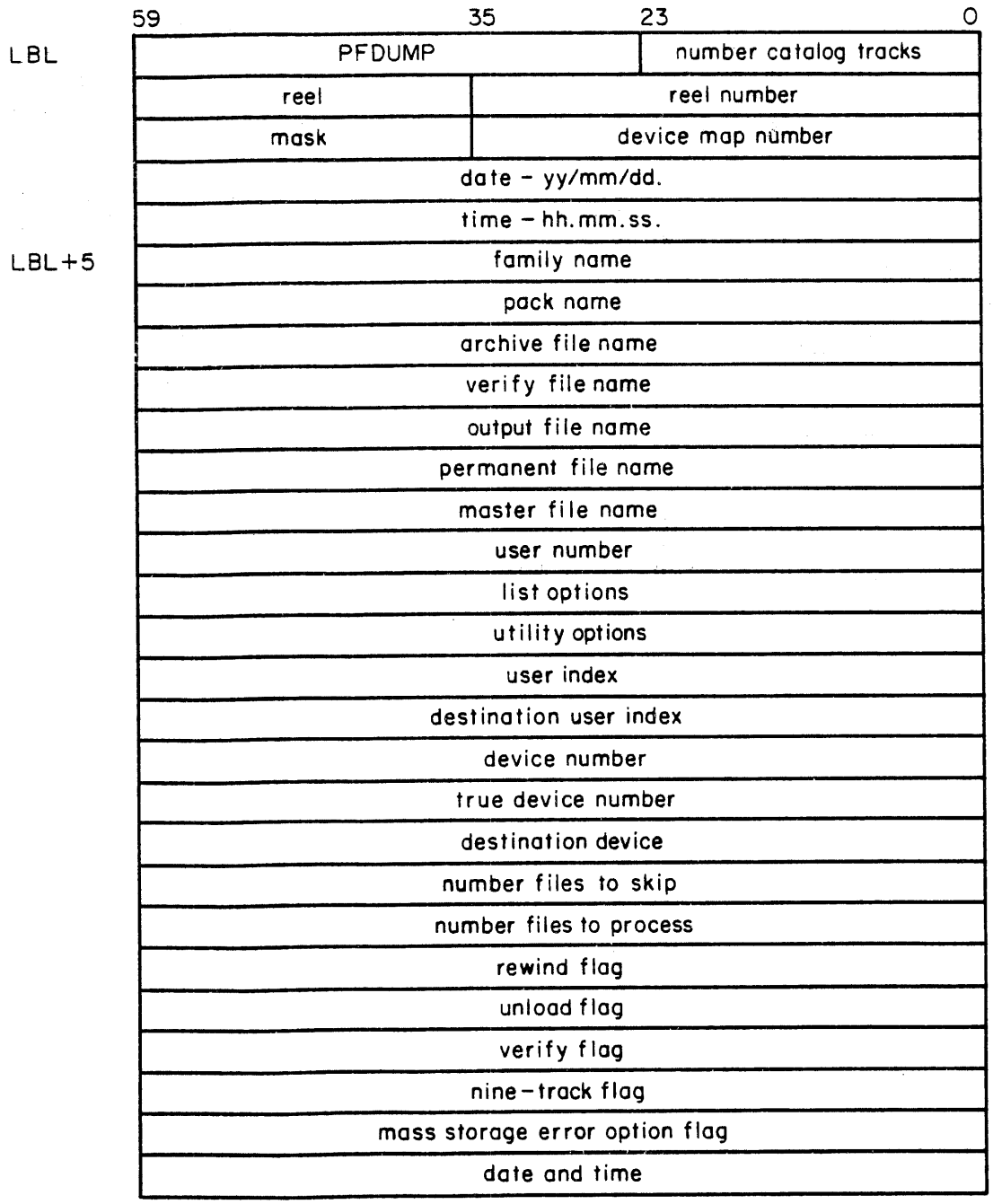


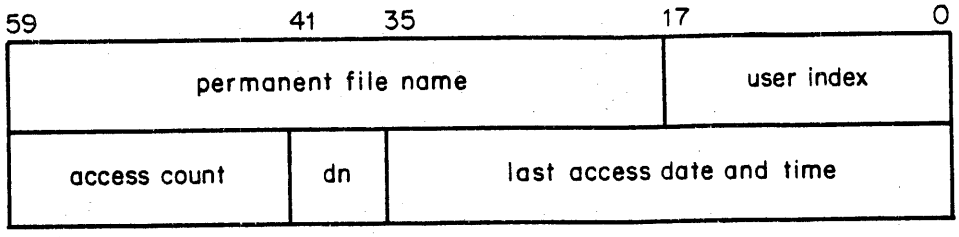
Figure 36-8. Tape Label Format

Catalog Image Record

The catalog image record (CIR) is written during the preset phase of PFDUMP if an incremental dump is being performed (OP=M, but not with OP=B). The CIR is written by subroutine CCI (create catalog image).

CCI waits for no permanent file utility interlock on a device using a call to subroutine WUC (wait PF utility interlock clear). CCI then calls PFU with the CTCT function to locate the correct catalog track, with function CTOL to open a catalog track file, and with function CTSC to interlock the catalog track.

CCI reads the catalog file generating catalog image entries of the following form using subroutine BCL (built catalog list). There is 1 entry for each file on the catalog.



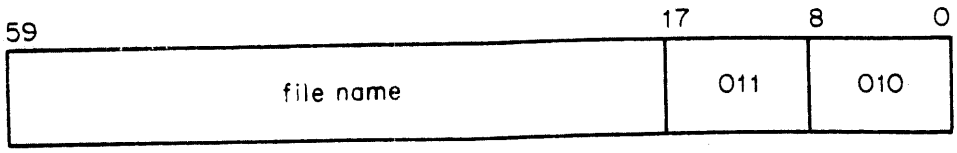
Catalog image blocks are written using a call to subroutine WIB (write image block) which writes the image block with control word on the archive file.

At the end of each catalog track, an appropriate CRCW type control word is written. The catalog track interlock is advanced via a CTAC PFU call. When all catalog tracks have been processed, PFU function CTCC is issued to clear the interlock on the last catalog track.

Writing the Permanent File

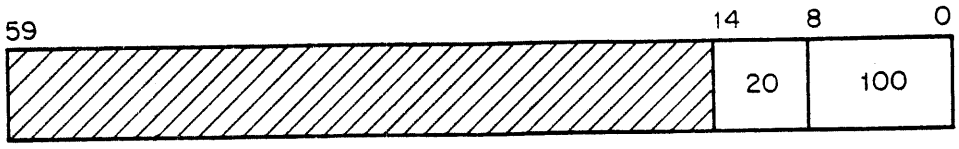
After starting the file dumping with the CTRL PFU function, the main loop of PFDUMP now begins individual file dumping to the archive file. The catalog entry, permit entries and data for each permanent file are written to the archive file as a separate logical record.

The control word for the catalog entry is built as follows with control word code CCWC.

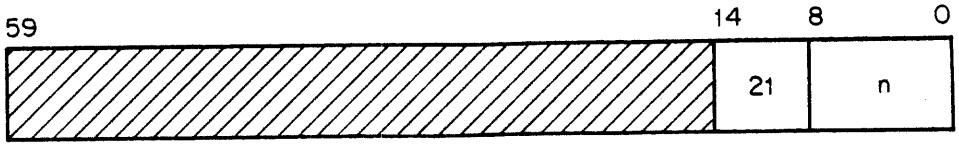


A call to subroutine WDT (write dump tape) writes the control word and the catalog entry to the archive file.

The catalog entry is then checked for a permit random index and if one is present, subroutine RPF (read permit file) is called. RPF randomly reads the permit file and calls subroutine WDT to write permit sectors to the archive file. Each permit sector is preceded by a permit control word (PMCW) in the following format.



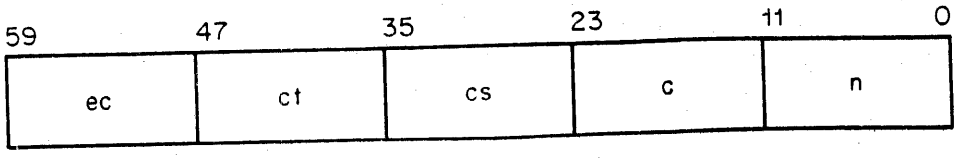
The last block of permits (may be the only block of permits for the file) is preceded by an end of record permit control word (PRCW) in the following format.



n Number of words in permit sector

The permanent file data is written using a call to subroutine WDB (write data block).

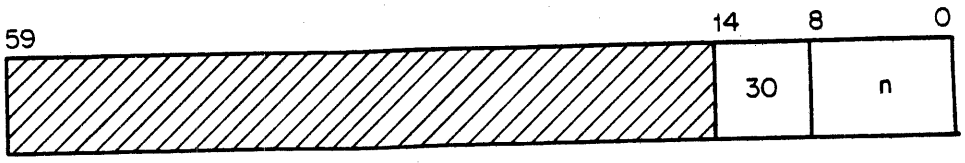
WDB calls subroutine RPR (read PRU) to retrieve the control word passed with each sector placed in the DATA buffer by the CTRL PFU function. The control word has the following format.



ec Error code
ct Current track
cs Current sector
c Control (linkage byte 1)
n Number of words of data (linkage byte 2)

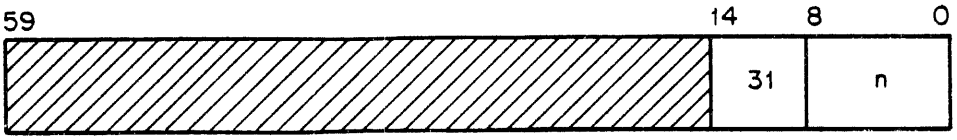
Fields c and n are the sector linkage bytes from the mass storage device. If c and n both are zero, the control word represents the end-of-information for the file. If c is zero, and n is nonzero, the control word represents an end-of-file. If c is nonzero and n is not a full PRU (=NWPR), the control word represents an end-of-record. If ec is nonzero, the control word represents bad data.

The data taken from the DATA file buffer is gathered into blocks of up to 777B words. A control word (DCWC) in the following format is appended to the front of each block before the block is written to the archive file using subroutine WDT.



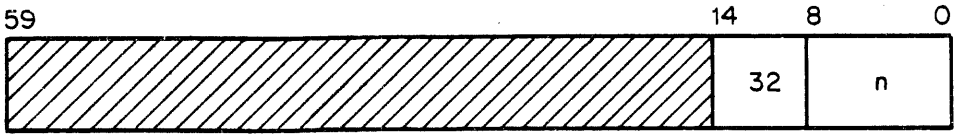
n Block word count

When an end-of-record (EOR) is detected in the DATA buffer, an end-of-record data control word (DRCW) is written. The EOR flag in the control word indicates the presence of an EOR following any data that may be associated with the control word. The format of the EOR control word is shown below.



n Number of data words in the block (preceding EOR)

End-of-files are represented on the archive file with an end-of-file data control word (DFCW) in the following format.



n Number of data words in the block (preceding EOF)

The end-of-information for the permanent file is represented by an EOR on the archive file itself.

After the EOI has been written, WDB returns to its caller.

Subroutine WDT (write data to tape) manages the writing of control words and data to the archive file. Subroutine WDT writes the specified data to the archive file using calls to subroutine WRT (write control word and data). Extra control words are added to the data written to the archive file by WDT, if necessary, to insure that no block has more than the maximum possible word count (777B).

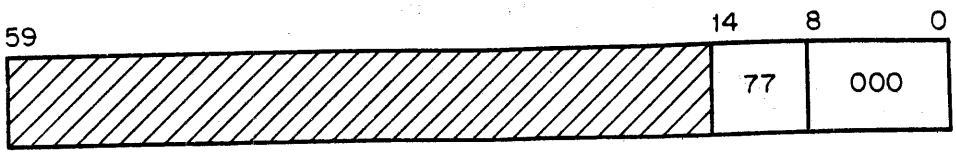
Subroutine WRT writes the control word and data using ARCHIVE WRITEW macro calls or writes an EOR using the ARCHIVE WRITER call.

Archive File Termination

Subroutine TAF (terminate archive file) is called when the dump has completed from the main loop or from AEP (about end processor).

Subroutine TAF closes the archive file if open (archive file dump label has been written) and returns it (and also the archive verify file if necessary) unless suppressed by the no unload option.

The archive file is closed by writing an end of dump control word (EODC), an end-of-record and an end-of-file. The control word is formatted as follows.



Unless suppressed by the no unload option (NU) an extra end-of-file is also written to the archive file. The archive file buffer is flushed using an ARCHIVE FLUSH macro call.

Purge After Dump

The purge after dump option (OP=P) cannot be done while dumping since, if for some reason the dump would abort, the files so purged would be lost. Therefore as files are dumped, the first word of the catalog entry is written on a file called PURGE. At the end of a successful dump, subroutine PGF (purge files that were dumped) is called.

For each entry in the PURGE file, subroutine PFG puts the file name and user index in a FET (PURGF), displays a message indicating the file name and user index being purged, uses the SETPFP macro to set the proper user index in the control point area, and calls PFM with the PURGE macro to purge the file.

Interlocking

PFDUMP conforms to the permanent file interlock mechanisms of permanent file activity, permanent file utility active and catalog track interlock.

PFDUMP requests that the permanent file activity count be incremented during the preset phase by subroutine SAC (set PF activity count). Subroutine SAC issues the CTIA PFU function to increment the permanent file activity count. The address ACFL is passed to PFU with the CTIA function as a status flag to identify when the activity count has been incremented. This flag is used during abort cleanup so that the activity count may be decremented if necessary. The following informative message is displayed on the K display and at message line 2 while PFU is being called to increment the activity count.

SETTING PF ACTIVITY COUNT

The permanent file activity count is decremented when PFDUMP completes or is aborted. Subroutine CAC (clear PF activity count) issues the CTDA PFU function to decrement the activity count only if the contents of ACFL indicates that the activity count has been incremented. The following informative message is displayed on the K display and at message line 2 while PFU is being called to decrement the activity count. The flag ACFL is cleared by PFU when the activity count is actually decremented.

CLEARING PF ACTIVITY COUNT.

PFDUMP interlocks the catalog track by using macro PCINT. Macro PCINT calls subroutine PCI (process catalog track interlock) with the PFU function to be used in interlocking the catalog track. The functions used are: CTSC to set the interlock; CTAC to advance the interlock to the next catalog track; and CTCC to clear the interlock. PCI displays the following informative message on the K display and at message line 2 while PFU is performing the specified interlocking operation.

WAITING FOR CATALOG INTERLOCK.

PFDUMP tests the permanent file utility interlock using a call to subroutine WUC (wait for PF utility interlock clear). This is done to insure that any PF utility activity (PFLOAD, CMS) has completed on a device before PFDUMP processes it. Subroutine WUC issues PFU function CTTU to retrieve the permanent file utility interlock status. The following informative message is displayed on the K display and at message line 2 until the permanent file utility interlock on the device is clear.

WAIT FOR PF UTILITY ON xx.

Error Processing

Error Processing fits into two categories: parameter processing, which includes missing files, device, or user; and errors in reading and writing permanent file archive data. This discussion is limited to the latter case.

Reading Catalog Entries

PFDUMP reads catalog entries (subroutine RCE) with CIO error processing selected (ERP1$ assembly option). If an error status is returned while reading the catalog file, subroutine RCE resets the bad data in the circular buffer and sets the OUT pointer to point to the bad data.

Subroutine PCE (process catalog error) is called from the main loop and from subroutine CCI (create catalog image) to process an error detected by PCE. Subroutine PCE calls subroutine GFP (get file parameters) which retrieves the equipment, track, and sector in error (using the STATUS macro). Subroutine PCE

issues an error diagnostic message, showing the equipment, track, and sector of the error. Additional messages are issued for each catalog entry in the sector affected by the mass storage error. These messages identify the file names and user indexes of files affected by the error.

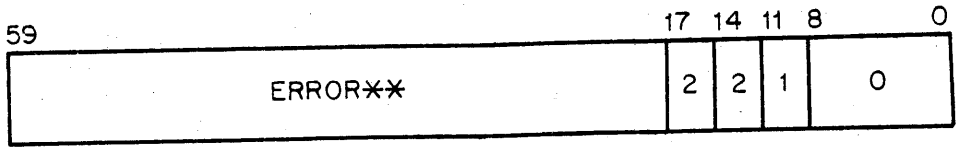
If the error occurs in the last catalog track sector or the error reported is wrong sector read, an end of catalog track status is returned to the caller. Otherwise, subroutine PCE reinitiates the read on the catalog file and processing may continue.

An error idle status is set on the equipment in error using a call to subroutine SEI (set error idle) which issues a CTEI PFU function.

Reading Permit Entries

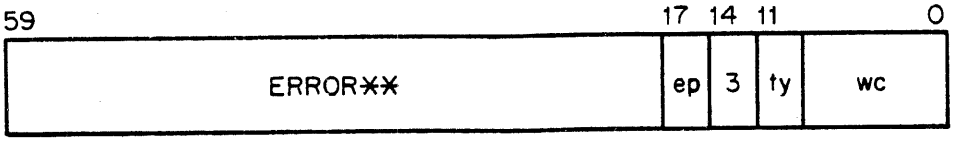
If subroutine RPF (read permit file) detects an error, it calls subroutine PPE (process permit errors). Subroutine PPE calls subroutine GFP to obtain the location (equipment, track, and sector) of the error. Subroutine PPE issues an appropriate error diagnostic and calls SEI to set the error idle status on the device.

A permit read error control word with the following format is written to the archive file in place of the bad sector.



Dumping is continued with the data for the file.

Subrouting WDB (write data block) interrogates the control word passed with the data by PFU. If a bad sector is detected, it is written to the archive file as a block by itself. A read error data control word in the following format is written with the block so that the data may be identified as being bad.



ep Error processing status:

1 E0 not specified
2 E0 specified

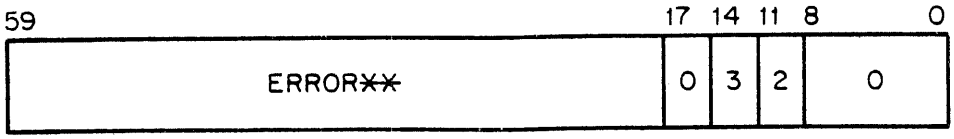
ty Block type

0 Full or EOI block
1 EOR block
2 EOF block

wc Block word count

Subroutine IDM (issue data messages) is called to issue an error diagnostic identifying the file and location (equipment, track, and sector) of the error. If the dump can be continued (not fatal error and not EOI), the E0 selection is checked. If E0 has not been specified, the remaining data in the file is processed. Otherwise subroutine SFD (skip file data) is called to properly position the data file to the next permanent file.

If direct access file length errors are detected by WDB, the appropriate too short or too long diagnostic is issued, a length error control word is written on the archive file, and the error idle status is set using a call to SEI. Processing is continued with the next file. The control word (LGCW) is formatted as follows.



Writing the Archive/Verify File

Subroutine PAE (process archive file error) is called by subroutine PAF (process archive file operation) when an unrecovered error status is returned by the CWWRITE call. PAE waits for the FET to complete and then clears the error status. The K display message

```
UNRECOVERED PARITY ERROR -
     ENTER K.GO - CONTINUE.
           K.END - ABORT DUMP.
```

is displayed and PAE calls the keyboard processor KIP (from PFS) to wait for operator action.

If END. is selected the following message is issued and the dump is aborted.

```
WPE UNRECOVERED - ABORT.
```

If GO is selected, the current reel is closed with the CLOSER/UNLOAD function. If the close operation is successful (no unrecovered parity error), the dump is continued on the new reel. If the close is unsuccessful, however, the following message is issued and the dump is aborted.

```
WPE UNRECOVERED - ABORT.
```

PFLOAD UTILITY

PFLOAD is a permanent file utility that loads files from an archive file onto a permanent file device. The following paragraphs describe PFLOAD options.

A noninitial load option (OP=N) verifies that no indirect access files exist on a device before the load is allowed. A replace option (OP=R) causes files from the archive file to replace those already on the device being loaded. The normal load (OP=R not specified) causes files from the archive file to be ignored if they are already present on the device being loaded.

An incremental load is performed by reading the catalog image record (CIR) from the most recent incremental PFDUMP (OP=M). Each item in the CIR represents a file active in the permanent file system at the time of the incremental (OP=M) dump. As the archive file is processed, if a match is found in the CIR, the file is a candidate for loading and the entry in the CIR is cleared. Files not found in the CIR are skipped since they have either already been loaded or they were purged after the archive file being processed was written.

Direct access files are loaded onto the device number on which they resided when dumped unless that device no longer exists or the file is not allowed on the device (user index is less than AUIMX and secondary mask bit corresponding to the user index is

not set). If the file cannot be loaded on the original device, it is loaded on the destination device (DD keyword) if one is specified and if the file is allowed to reside on that device. If the file cannot be loaded on either device, a diagnostic message is issued and the file is skipped. A direct access file is considered to have resided on the master device being loaded if it originally resided on its own master device.

Tape errors may be encountered when loading a file. These errors are processed by setting the length of the file to the amount of the file processed at the time of the error and continuing with the next file.

A flowchart of PFLOAD main loop is shown in figure 36-9.

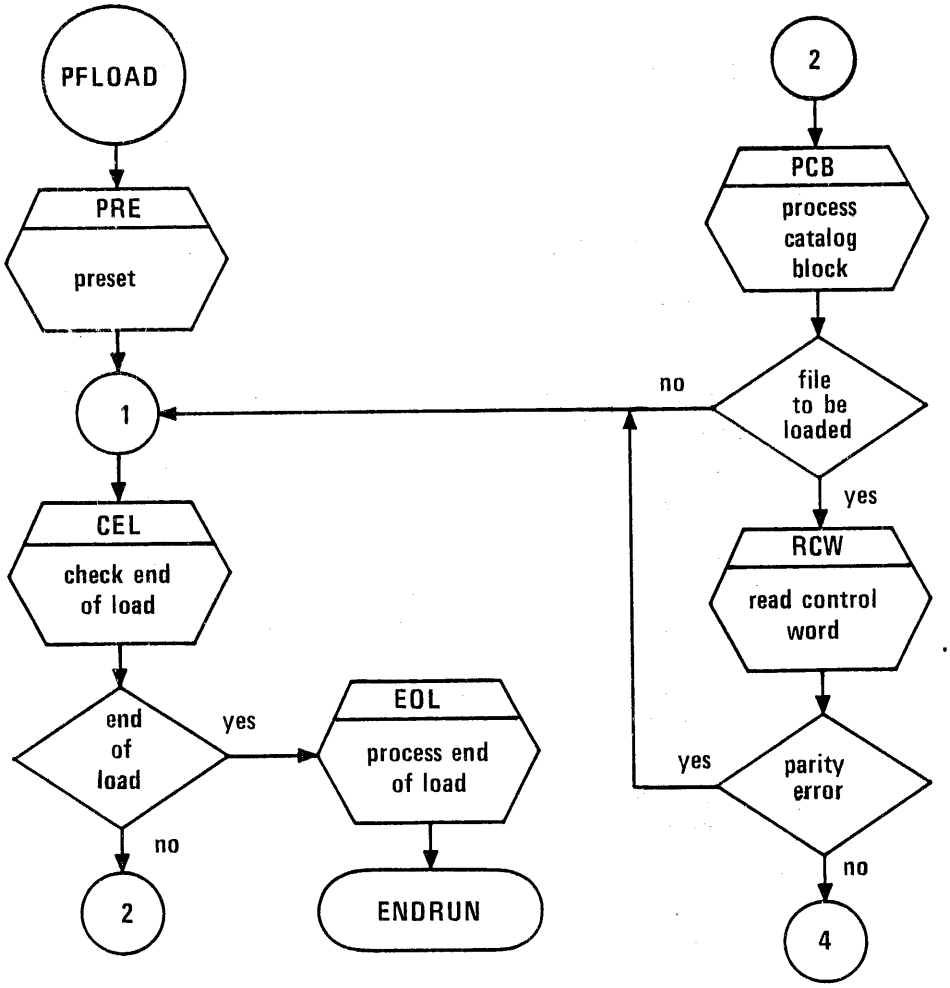


Figure 36-9. PFLOAD.

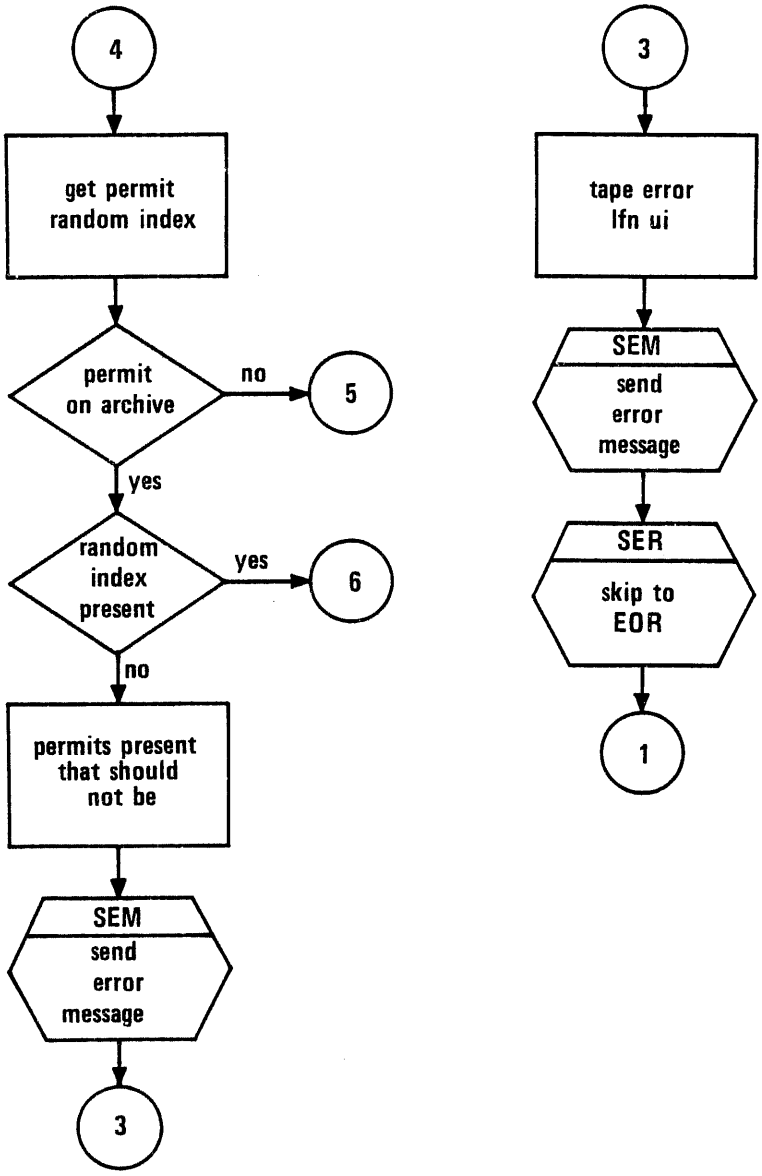


Figure 36-9. PFLOAD (Continued)

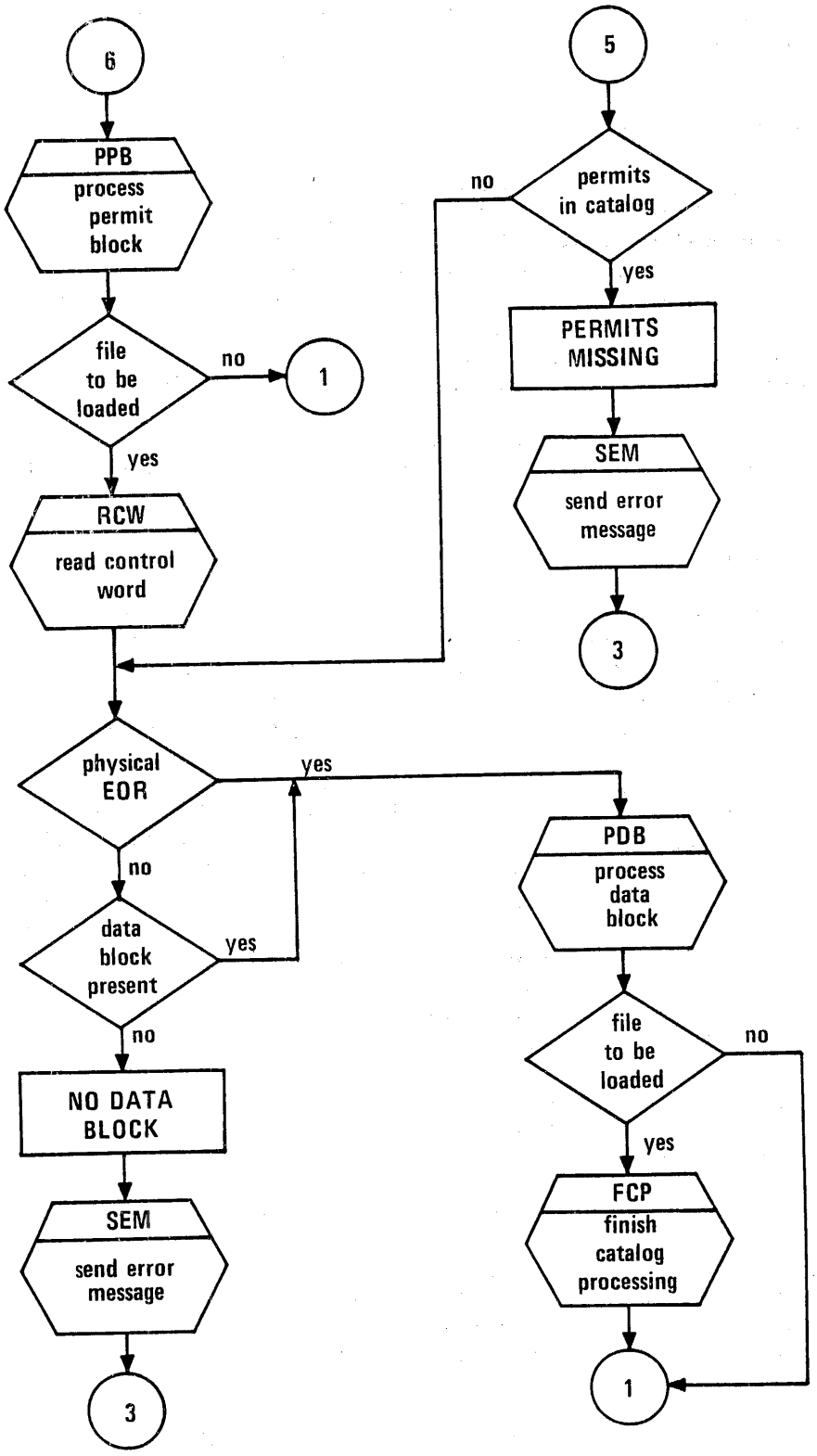


Figure 36-9. PFLOAD (Continued)

This discussion of PFLOAD concentrates on these areas: loading the file, assigning the archive file, transferring the file to mass storage, interlocking, activating PFU, and error processing.

Loading the File

PFLOAD performs the same device selection mechanism as does PFDUMP. Subroutines BMT (build mass storage tables), SBS (set boolean selection), SSP (set selection parameters), and SMP (set master device parameters) are virtually the same for both routines and therefore will not be discussed in detail here.

The boolean file selection variables are identical to those in PFDUMP, but PFLOAD adds two file selection boolean expressions:

DSTD	Display true device
DSDI	Display device for destination user index

File Selection

Subroutine PCB (process catalog buffer) is called to read the catalog entry and determine if the file is to be loaded. Subroutine PCB calls subroutine CSP (check selective parameters) which determines if the file meets specified date, user index, permanent file name, and alternate device options.

If the file is to be loaded, the catalog image is checked by subroutine CCI (check catalog image) to determine if the file is to be loaded during an incremental load. The catalog image record file (CIRF) is read to determine if the file exists on the CIR. If a match is found, the entry is cleared, and the load file status is returned. The load file status is also returned if the load is not an incremental load. This mechanism means that if an incremental load is being performed, only those files which are found in the CIR can be loaded and once a file is loaded it is removed from the CIR.

If the file is still a candidate to be loaded, subroutine PDI (process destination index) is called to process the destination user index. If a destination index has been specified, it is written into the catalog entry and the device number is set to the corresponding device.

The access type of the file (direct/indirect access) is then checked to determine if the file satisfies any specified access type selection parameter (OP=I or D). If the access type selection is satisfied, the file continues to be a candidate for loading.

If the candidate file is a direct access file, PCB determines whether the residency device is available and if so, whether the

device is still a valid residency device for the user (that is, correct secondary mask bit set for the user index). If a valid residency device is not found, validation of the destination device (DD keyword) is done. If no DD is specified or it is not valid for the user, the following diagnostic is issued and the file is no longer a candidate for loading.

ALTERNATE DEVICE NOT FOUND.

If the DD device is valid it is put into the catalog and becomes the new residency device for the file.

PCB now sets up master device parameters using a call to subroutine SMD (set master device parameters). If the file is on the same master device and catalog track as the previously loaded file, SMD requests the permanent file utility interlock using a call to subroutine SIN (set PF utility interlock) and activates PFU using a call to subroutine AUP (activate utility processor).

If the file is on the same master device but a different catalog track, SMD checks to ensure outstanding PFU activity has ceased. If PFU is not active, it is activated on the new catalog track using the SIN/AUP sequence described above. If active, subroutines FCP (finish catalog processing) and PCT (position on catalog track) are called to complete processing of the old catalog track and to establish processing on the new catalog track. If the file is on a different master device, SMD gets the catalog parameters for that device using a CTCT PFU call, sets up a CATS, PETS, and DATA FST entry for the master device and opens these files using CTOP PFU calls. SMD then executes the SIN/AUP sequence to set the utility interlock and activate PFU on the new device.

Subroutine PCB then calls subroutine SFF (search for file) to determine if the file already exists on the permanent file device. If it does, the file is no longer a candidate for loading unless the replace option is specified (OP=R). If the replace option is specified, the file is purged using subroutine PGF (purge file). If the purge is successful, the file continues to be a candidate for loading.

The candidate file is now ready to be loaded. Subroutine MCE (move catalog entry) is called to move the catalog entry to the CATS buffer. Subroutine PCB exits to the caller with the load file flag set.

If the file is not a candidate for loading, PCB sets the skip to EOR (SKER) flag and returns with the load flag clear.

Permits Processing

The next archive file control word is read using a call to RCW (read control word). The control word and catalog entry permit random index are verified to ensure that permits are present

when the catalog so indicates and not present otherwise. The following diagnostics are used to report permit mismatches between the catalog entry and archive file.

PERMITS PRESENT THAT SHOULD NOT BE.

PERMITS MISSING.

Subroutine PPB (process permit block) reads permit entries from the archive file and writes them to the permit file using a call to subroutine WPR (write PRU to PFU circular buffer). Subroutine PPB sets the random index of the permit entries in the catalog entry and sets permit linkage in the permit entry before calling WPR. If the archive file permit block is empty, the permit random index is cleared from the catalog entry and no writing is done. Subroutine PPB returns to the caller with the load file status set unless file loading is suppressed because of a tape error.

If the archive file control word indicates that there was an error while dumping the permit entries, subroutine SCE (set catalog error code) is called to indicate that the permits were in error. Processing of the permit entries then continues normally.

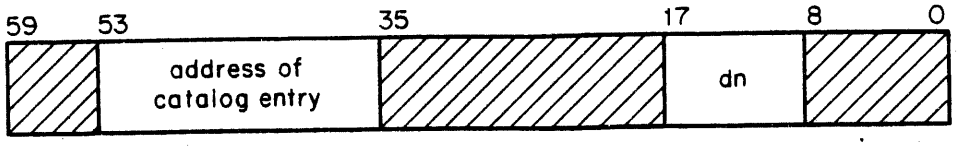
If the permits cannot be loaded, an appropriate diagnostic is issued, the skip to EOR flag (SKER) is set, and subroutine PPB returns to the caller with a file not to be loaded status.

Data Processing

The next control word is read from the archive file using RCW. If a nondata type block is present, the following diagnostic is issued and processing continues with the next file.

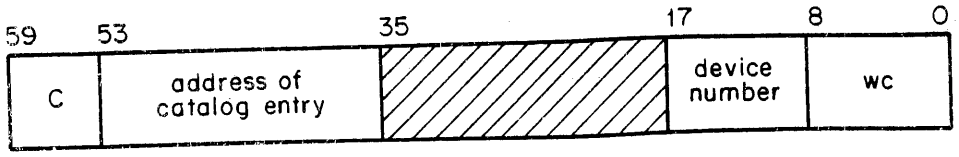
NO DATA BLOCK.

Subroutine PDB (process data block) is called to process the data. PDB sets up a control word in location LWRD that has the following format.



dn 0 for indirect access file;
4000 for a direct access file on an auxiliary device;
device number for a direct access file on a family device

The data is read from the tape according to the archive file control word. Subroutine PDB updates LWRD with additional control information as follows.



c 0 if normal; 1 if EOF (DLEF); 2 if EOI (DLEI)
wc Word count

PDB now calls subroutine WPR to move the data to the DATA FET buffer. PDB continues until all data has been transferred from the archive file. Once the EOI has been sent to PFU through WPR, PDB returns to the caller with a load file status.

If an archive file control word indicates a file truncation error, the following diagnostic is issued, the skip to EOR flag is set, the catalog is updated to reflect the true length, and control is returned to the caller with load file status set.

FILE TRUNCATED filenam userin.

If an archive file control word is a read error control word, PDB sets the data error status in the catalog entry. If bad file processing is suppressed (EO selected) or the error control word is a suppressed file control word (EO specified when dumping), subroutine DIF (delete incomplete file) is called to relinquish unneeded mass storage. Subroutine PDB returns to the caller with file not to be loaded status and skip to EOR (SKER) set. If loading of the file is not suppressed, additional data for the file is processed as if no error condition was present in the control word.

When a tape error or abnormal condition is sensed on the archive tape, an appropriate diagnostic is issued, skip to EOR (SKER) is set, an EOI sector is written and the catalog is updated to reflect the true length of the file. The load file status is set and control is returned to the caller.

Catalog

After the data has been processed, subroutine FCP (finish catalog processing) is called to complete the loading of the file. FCP advances the CATS IN pointer and performs CATS buffer writing as required using calls to FCW (force catalog write).

End-Of-Load

Subroutine CEL (check for end of load) is called to advance to the next file and determine if the load has finished. Subroutine CEL checks the SKER (SKIP to EOR) flag and calls subroutines DUP (drop utility processor), CIN (clear utility interlock), and SER (skip to EOR) if the flag is set.

If an individual file was selected to be loaded (PF keyword) and that file was loaded, CEL sets the load complete status and returns to the caller. If this is not the case, the incremental load flag is checked. If this is an incremental load and the catalog image record is now all zero (all incremental files processed), the load complete status is set and control returns to the caller.

If this is not the case, the next archive file control word is read. If the control word is a catalog control word, CEL returns to the caller with the load not complete status. If the control word is a label control word, subroutines DUP and CIN are called and subroutine PAL is called to process the archive file label. CEL proceeds by reading the next control word and interrogating it as above.

If an EOR is detected on the archive file, CEL reads the next control word and interrogates it as above.

If an EOF is detected on the archive file, and no more files are to be processed (N keyword met or not specified), it is treated as an archive file EOI. If more files are to be processed, subroutines DUP and CIN are called, archive file reading is reinitialized and subroutine PAL is called. Subroutine SCI (skip catalog image) is called to properly position the archive file. The next control word is read and interrogated.

When the archive file EOI is detected, end of load status is returned to the caller unless an incremental load is being performed and more files remain on the catalog image record. When more files remain for an incremental load, subroutine ART (archive file transition) is called to process operator intervention. Depending upon operator action, either end of load status is returned by subroutine CEL or the first control word of the next archive tape is analyzed to determine the next action of subroutine CEL.

The operator may choose to end the load, list files remaining on the catalog image record or proceed to the next tape of an incremental load set. Subroutine LFS (list files on catalog image record) is called by subroutine ART to list files if the list option is selected by the operator. Subroutine AAF (assign archive file) is called to request the next tape for the incremental load if that option is selected by the operator.

If the control word is an end-of-archive file control word, and no more files are to be processed (N keyword not specified or has been met), the end of incremental load test is performed.

If the archive file is not positioned at an EOF/EOI, it is so positioned using a SKIPFF macro. If now positioned at an EOI, the end of incremental load test is performed; otherwise, the condition is treated as an EOF control word.

If a valid control word is not recognized or a parity error status is returned from RCW, a diagnostic is issued, subroutines DUP and CIN are called, a skip to EOR is performed (call to SER) and the next control word is interrogated.

The main loop of PFLOAD either continues with processing of the next file or calls subroutine EOL (end of load processing) and performs an ENDRUN, depending upon the status returned by subroutine CEL.

Subroutine EOL drops PFU using subroutine DUP, clears the utility interlock using subroutine CIN, returns the load scratch files (CATS, PETS, DATA, and so on), issues appropriate end-of-load messages, and restores the user permanent file parameters to the control point area using subroutine RUP (restore user permanent file parameters). The archive file is then positioned to skip any remaining file count (N parameter) if the NR parameter is specified or is returned if not specified.

Archive File Assignment

Subroutine AAF (assign archive file) unloads the current archive file (expected on first call to AAF), does a LABEL request for the next archive file, positions it as required by NR and SF keywords, initiates file reading using the READCW macro, and calls subroutine PAL (process archive file label) to check the label.

If good label status is returned by subroutine PAL, subroutine AAF returns to its caller. If not, the following message is displayed and the LABEL request is repeated.

```
PFLOAD - LABEL BAD, ASSIGN NEW TAPE.
```

PAL reads the first control word from the archive file (using RCW) and if it is a label control word, reads the remaining label data.

The K display is updated from the date, time, and mask fields in the label, an output file message containing the date and time from the label is generated, the archive file is positioned to EOR using subroutine SER, and good label status is returned to the caller.

If a bad label is detected (control word is incorrect, label identifier is incorrect, or the label read is incomplete), an SER (skip to EOR) is performed, and a bad label status is returned to the caller.

Transferring Files to Mass Storage

Once the permanent file utility processor (PFU) has begun processing the CTLM (load main loop) function, data transfer operations may be done through the circular (or FET) buffers associated with the PETS and DATA files. Subroutine WPR (write PRU to PFU circular buffer) moves data from a working buffer to the desired circular buffer. WPR is a general subroutine which is passed the address and size of the working buffer and the address of the FET.

WPR examines the sector counter in FET word FTSC. If bit 11 of the sector counter is set, PFLOAD has reached sector threshold during an earlier request and the buffer needs to be flushed (emptied) before proceeding. Subroutine FUB (flush utility buffer) is called to issue PFU write requests until the buffer has been emptied. The sector counter is reset to zero and processing is continued.

Once sector threshold has been eliminated, subroutine WPR checks to see if data is to be transferred. If no data is present in the working buffer, WPR returns to the caller. If data is present, subroutine WPR determines if the data to be transferred will fit in the circular buffer. If there is insufficient room in the circular buffer and the buffer is not busy, a write request is issued. If the buffer is busy or after a request is issued, subroutine WPR waits in periodic recall for buffer space to become available.

Once space is available, the data is moved from the working buffer to the circular buffer. The sector counter is incremented and subroutine WPR determines if it is necessary to issue a write request. If the buffer is at least half full a write request is issued before returning to the caller.

The permit and data buffers are also flushed under force catalog write and delete incomplete file situations by subroutines FDB (flush data buffer) and FPB (flush permit buffer). These two routines call FUB with the appropriate DATA or PETS FET address.

The catalog file (CATS) is written by subroutine FCP (finish catalog processing) using a call to subroutine FCW (force catalog write). Subroutine FCW first calls subroutines FDB and FPB to flush the DATA and PETS buffers. A write function is then issued on the CATS FET and subroutine FCW waits in RECALL for the function to be completed by PFU. Subroutine FCW then rewinds the FET buffer, clears the sector counter (FTSC), clears words in the sector (NWIS) and returns to the caller.

The PETS and DATA buffers are also flushed when an incomplete file is deleted. Subroutine DIF (delete incomplete file) is called by subroutine MCE to flush a partially loaded file, thus ensuring that the previous file has been properly terminated before processing a new file. DIF is also called during bad file processing by PDB (that is, E0 specified). If the incomplete file flag (IPFF) is set on entry, subroutine DIF calls subroutines DDS (delete data space) and DPS (delete permit space) and clears the incomplete file flag.

Subroutine DDS calls subroutine, FDB to flush the data file buffer and then checks the files catalog entry. If there is no user index or the user index is WEUI (write error user index) or if there is no first track specified, DDS returns to the caller. Otherwise, the first track is cleared from the catalog entry, the drop track and flaw parameters are set in FTPM of the DATA FET and the position file request is set in the DATA FET for processing by PFU.

Subroutine DPS calls subroutine FPB to flush the permit buffer and then resets the permit random index pointer in RICT to that found in the catalog entry. A position file operation is requested in the PETS FET with the new random index set in FTPM. An EOI control word is set in the working buffer and WPR is called to terminate the permit chain.

Interlocking

PFLOAD uses the permanent file utility interlock to prohibit any other permanent file usage of a device while loading is being performed on that device.

Subroutine SIN (set PF utility interlock) is used to request PFU to obtain the PF utility interlock. If the interlock is not currently set, SIN displays the following message while issuing a CTSU PFU function to set the interlock.

```
SETTING UTILITY INTERLOCK.
```

The address UIFL is passed to PFU with the CTSU function as a status flag to identify when the utility interlock has been set. This flag is used during abort cleanup so that the utility interlock may be cleared if necessary.

Subroutine CIN (clear PF utility interlock) is used to clear the utility interlock. If UIFL is nonzero, subroutine CIN displays the following message while issuing a CTCU PFU function to clear the interlock.

```
CLEARING UTILITY INTERLOCK.
```

UIFL is cleared by PFU when the utility interlock is cleared.

Activating PFU for Loading

Subroutine AUP (activate utility processor) is used to request PFU to initiate communication with PFLOAD for permanent file loading. All loading operations are performed by the CTLM PFU function which loops until PFLOAD directs it to complete and drop.

If PFU is already active (PAFL is nonzero), subroutine AUP returns to the caller. If PFU is not active, subroutine AUP sets the master equipment and noninitial flags in word FTDW of the DATA FET, sets the device number and FET addresses into word FTPM of the CATS FET, sets the DATA and PETS FETS busy, clears the sector count (FTSC) in each FET, and clears the data state (FTDL) word in the DATA FET.

Subroutine AUP then issues the CTLM PFU function and goes into recall on the DATA FET. PFU sets the DATA FET complete when load initialization is complete. Subroutine AUP then calls subroutine PCT (position on catalog track) to determine what files already are present on the catalog track. Finally the current EOF position of the permits file is set in RICT from the random index returned by PFU in word FTSC of the PETS FET.

Subroutine DUP (drop utility processor) is used to drop PFU. If PFU is active (PAFL is nonzero), subroutine DUP forces an EOI write on the catalog track using a call to FCP (finish catalog processing). When the DATA FET is complete, subroutine DUP sets a 10B request in the FET (drop PFU) and waits for PFU to complete the DATA FET and drop from the PFLOAD control point. Once PFU has dropped, control is returned to the caller.

Error Processing

This section discusses error processing for the archive file and for the mass storage devices being loaded.

Reading the Archive File

The archive file is read in control word mode (READCW), typically through the use of the CWREAD macro defined in PFLOAD. This macro calls subroutine CWR (control word read words) to transfer data in a manner similar to READW.

Errors Reading Control Words

If after issuing a CWREAD to obtain an archive file control word, an archive file error status is returned, subroutine RCW returns a status that indicates a parity error. The caller of subroutine RCW must then process the error appropriately. The action typically taken in this error case is to issue an appropriate diagnostic and attempt to process the next file by doing a skip to EOR (SER) request.

Similarly, when reading data from the tape, if a parity error occurs, an appropriate diagnostic is issued, a skip to EOR is done, and the file not loaded.

Writing the Permanent File

Subroutine PCE (process catalog errors) is called to handle both read and write catalog errors. Subroutine PCE is called by subroutines FCW (write) and PCT (read).

Subroutine PCE requests PFU to drop if it is active. Once PFU has dropped, subroutine SEI is called to set the error idle status for the device. Subroutine SEI (set error idle) issues

the CTEI PFU function to set the error idle status on the device being processed. The utility interloack is then cleared using a call to subroutine CIN. Subroutine PCE then generates a collection of error diagnostics identifying the failing device and the catalog entries in the sector in error.

Subroutine PWE (permit write error processor) is called to handle errors from permit writes. Subroutine PWE requests PFU to drop if it is active. Once PFU has dropped, subroutines SEI and CIN are called to set the error idle status and drop the utility interlock.

Subroutine PWE then issues appropriate diagnostic messages identifying the error condition and the affected file.

Subroutine DWE (data write error processor) is called to handle errors that occur when writing the data to mass storage. PFU returns the location of the error (FST at time of error) in word FTPM of the DATA FET. The equipment, track and sector are extracted from the FST into MPEQ, MPTK, and MPSC for use in error message formatting. The file name (MPFN), user index (MPUI), and device number (MPDN) are also set up for error message formatting. An appropriate diagnostic is then issued from this information.

Subroutine DWE calls subroutine SEC (skip through EOI control word) to read the DATA file until the circular buffer is empty or an EOI is detected.

The sector count returned by subroutine SEC is used to update the file length in the catalog to include the bad sector but nothing following it. The sector count is also used to adjust the sector counter for the DATA FET to reflect the amount of data removed from the circular buffer.

If the file is indirect access, the user index in the catalog entry is changed to a special write error user index. This allows the catalog entry to be kept in the permanent file system as a pointer to the bad file. For a direct access file, the user index is cleared (creating a direct access hole), the appropriate parameters are set in word FTPM of the DATA FET and a PFU file position function is issued to drop the track chain and flaw the bad track.

Subroutine SCE (set catalog error code) is called by subroutines PPB and PDB to indicate that an error has occurred in the permits or data for the file. These error conditions are stored in the catalog entry word FCEC. The use of these error codes allows a file with errors to be loaded into the permanent file system without the danger of the file's owner being unaware of the errors.

INTRODUCTION

The Interactive Facility (IAF) is a subsystem that provides support for interactive processing from remote terminals communicating through the Network Access Method (NAM). The subsystem consists of the following CPU and PP programs.

Program	Description
IAFEX	Time-sharing executive initialization routine. This routine is loaded at 40000B relative to control point 1 when the operator types IAF. It initializes tables and pointers and loads IAFEX1 and IAFEX4.
IAFEX1	Time-sharing executive processor. This is the main routine that processes I/O for the remote terminals. It cracks and processes commands, and makes requests to dump source input to disk and refill output buffers from disk.
IAFEX2	Performs an optional dump upon termination and loads IAFEX3.
IAFEX3	Time-sharing executive termination routine. This routine is executed after an abnormal condition is detected or when the operator terminates IAF with 1.STOP.
IAFEX4	Interfaces IAFEX1 with NAM.
1TA	Auxiliary function processor. This routine processes functions for IAF which require PP action.
1TN	Communicates between IAF and the NOS stimulator (checkout/test).
1TO	Terminal input/output. Called by IAF to perform terminal I/O requiring disk accesses.

The relationship between the various system routines and subsystem routines is shown in figure 37-1.

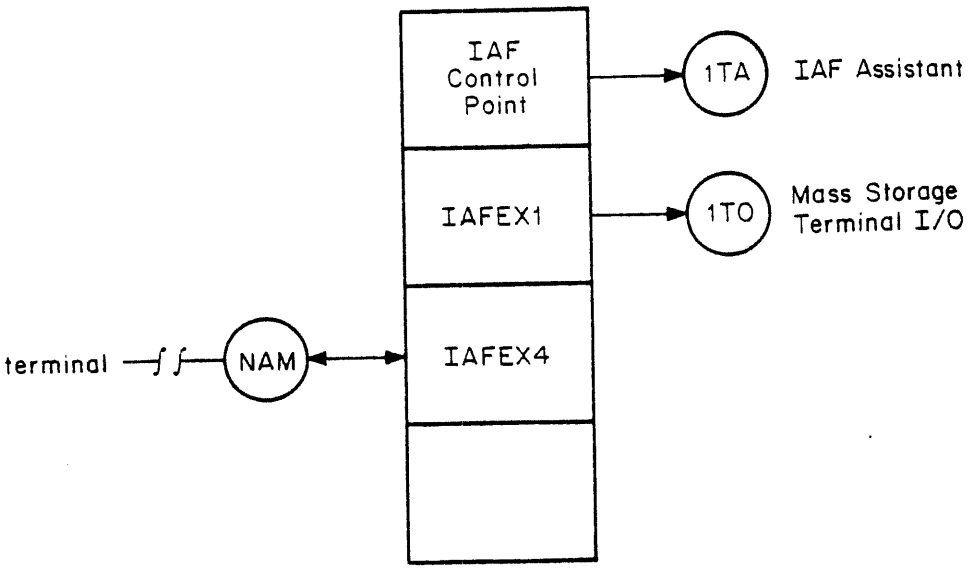


Figure 37-1. IAF Interactive Subsystem

TERMINAL OPERATION

The flow of data to or from a terminal and a mass storage device is shown in figure 37-2. The terminal user enters source statements. These statements are read from NAM by IAFEX4, converted to the proper internal representation, and stored in pots (a pot is an eight-word buffer) by IAFEX4. Whenever IAFEX4 has filled VIPL pots (defined in common deck COMSREM) it issues a dump pot request. IAFEX initiates the routine DMP (local to IAFEX1) which calls 1TO. In the interim IAFEX4 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.

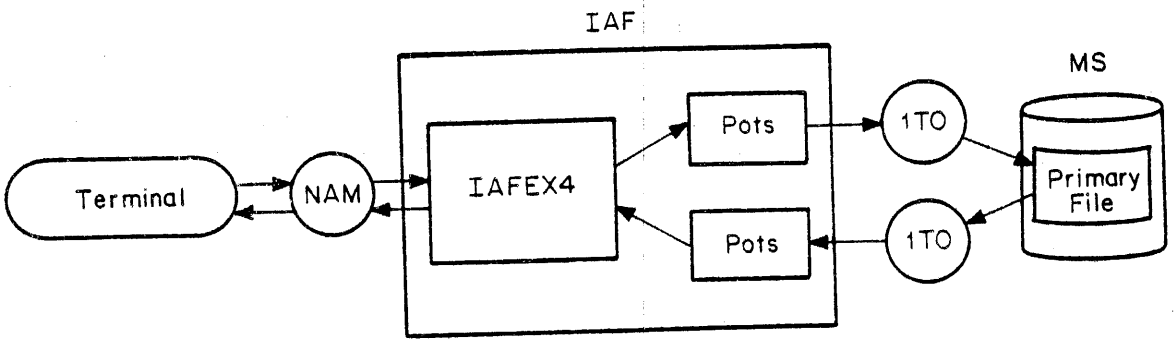


Figure 37-2. Terminal Mass Storage Data Flow

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, 100B 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 the terminals whose files were sorted to active mode and IAF then processes the command that indirectly caused the sort.

TERMINAL JOB INITIATION

Refer to figure 37-3 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. IAF 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 37-15).

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 only 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.

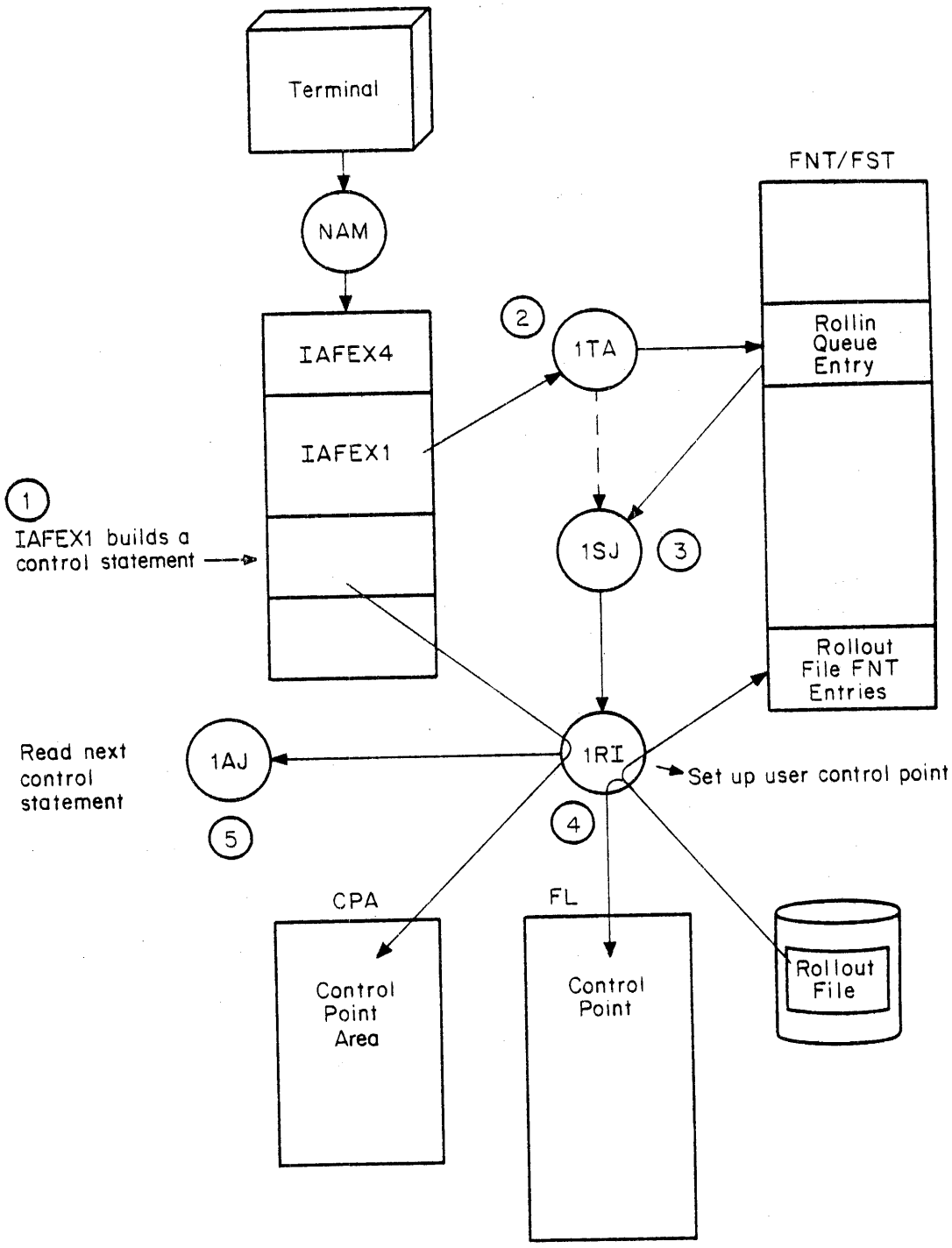


Figure 37-3. Terminal Job Initiation

TERMINAL JOB INTERACTION - OUTPUT

Refer to figure 37-4 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 saves the first sector of output in 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 IAF that output is available for the terminal by issuing monitor function TSEM.

4. IAFEX1 assigns the data pots to IAFEX4 to the terminal. IAFEX4 continues to ask IAFEX1 for additional output and IAFEX1 in turn calls 1TO until all output has been transferred.

5. After all output is transferred, IAFEX1 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 37-5 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.

3. 1TO issues any available output and issues monitor function TSEM to inform IAFEX1 of the completion of its processing.

4. IAFEX1 calls IAFEX4 to send any output and/or issue the input prompt character (a question mark).

5. IAFEX4 translates the data as required and places it in pots.

6. When the end-of-line is sensed, IAFEX1 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.

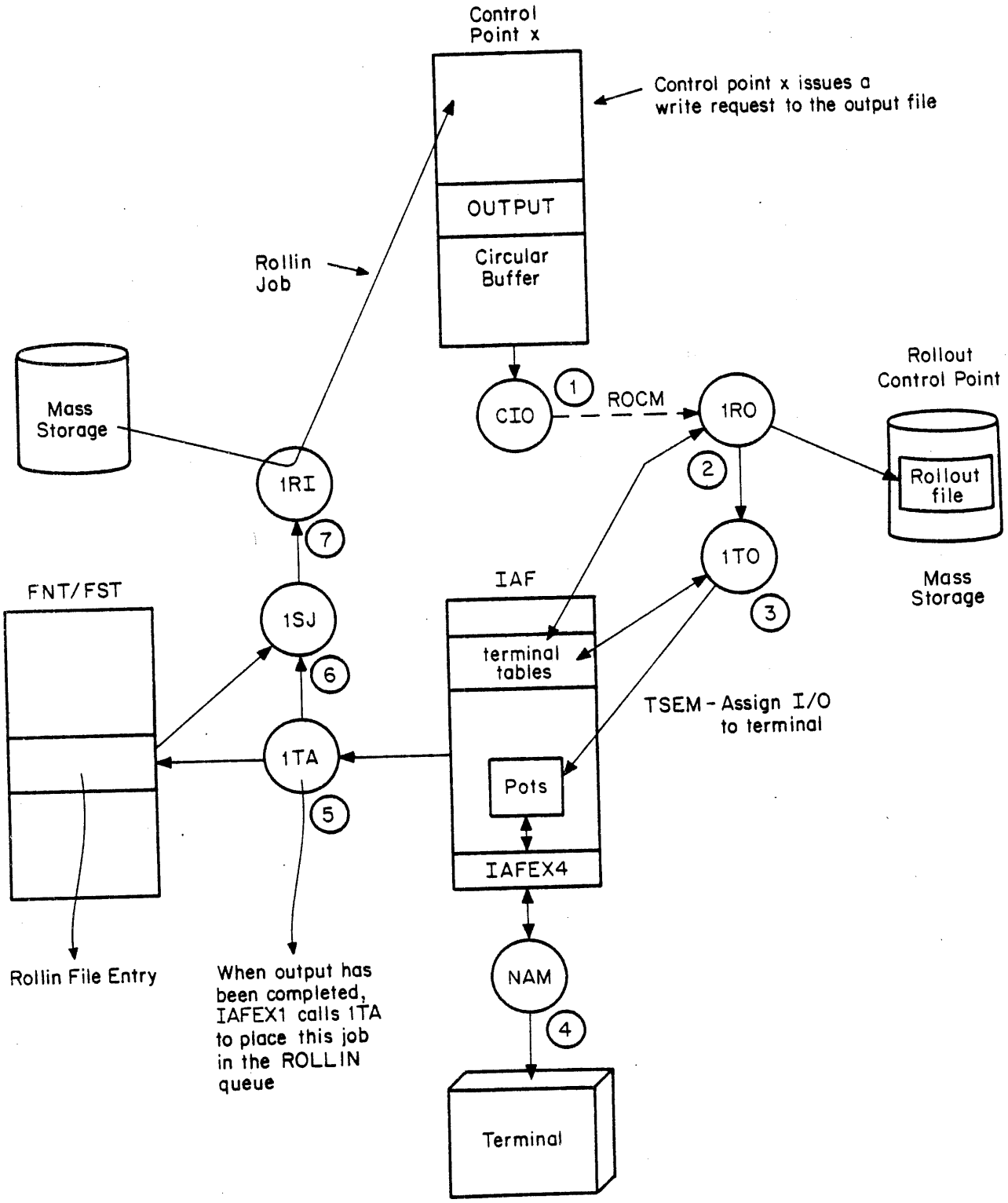


Figure 37-4. Terminal Job Interaction (Output)

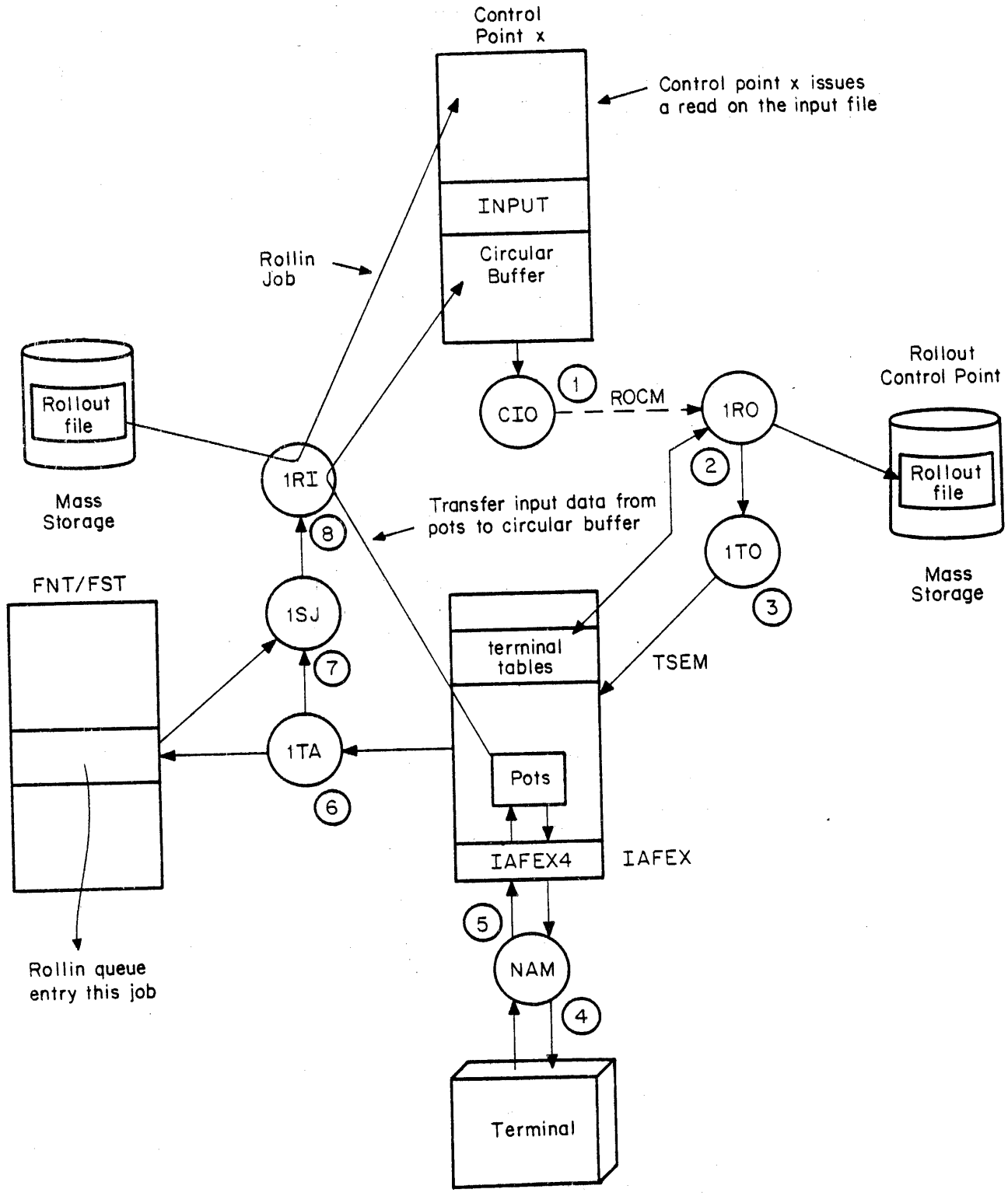


Figure 37-5. Terminal Job Interaction (Input)

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 IAFEX 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 IAFEX. 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.
/
```

IAFEX INITIALIZATION

Basically, IAFEX initializes tables and pointers, then loads IAFEX1 and IAFEX4 and starts IAFEX1, 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
1TA	Auxiliary function processor
1TN	Stimulated terminal driver

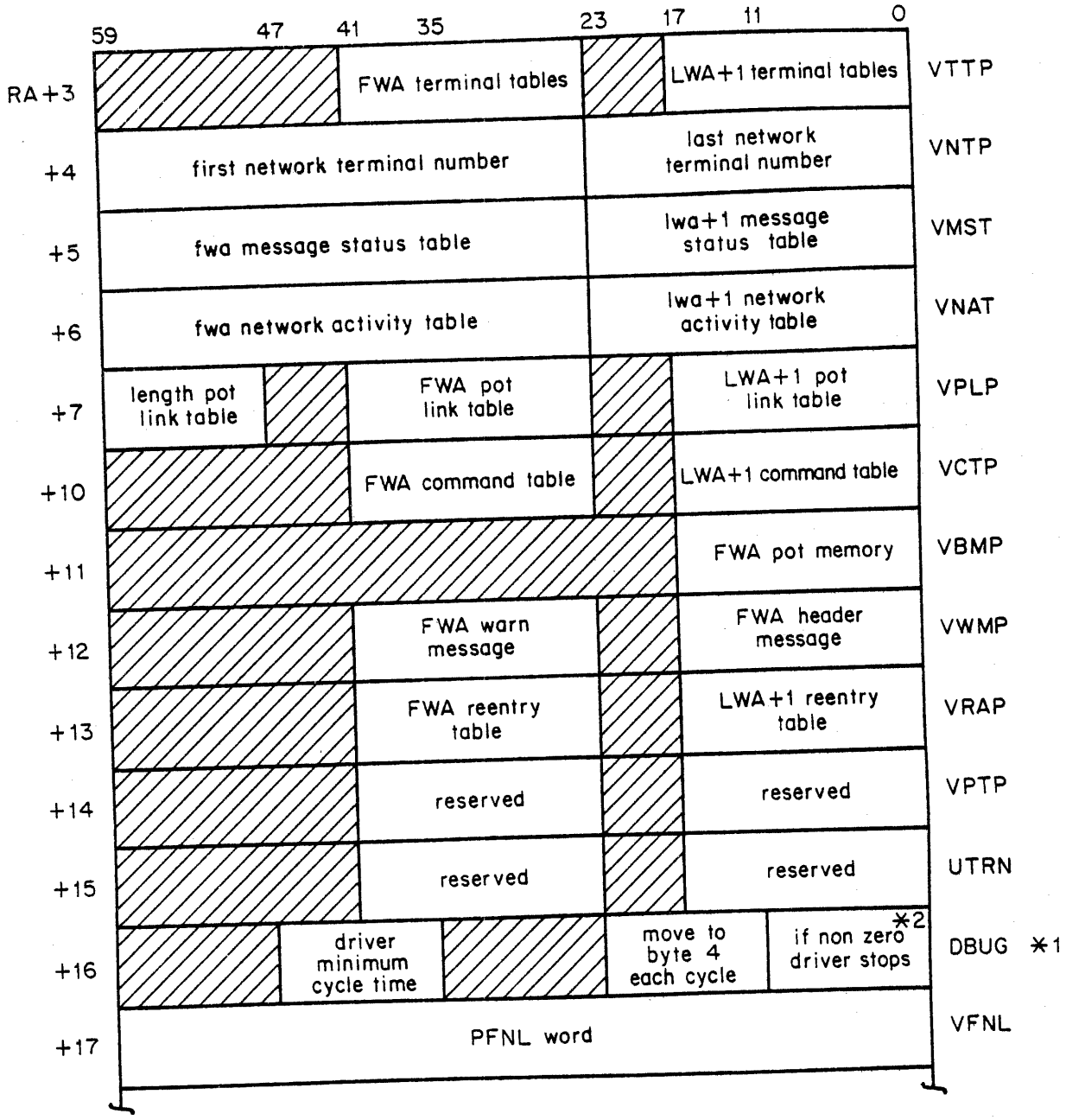
When the operator types IAFffff.*, DSD calls 1DS which generates an input FNT/FST entry of a special type. Routine 1SJ recognizes this entry during its scheduling process, assigns the entry to the proper control point, and calls 1SI into a PP. Routine 1SI calls a procedure file named IAFffff*, an indirect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

```
*RETURN PROCEDURE FILE IAFEX.
RETURN,IAFEX.
WHILE,TRUE,LOOP.
IAFEX.
IAFEX2.
SKIP,LOOP.
EXIT.
IAFEX2.
ENDIF,LOOP.
ENDW,LOOP.
```

Initialization consists of allocating tables, establishing the pointers shown in figure 37-6 and the constants shown in table 37-1.

---

* The characters ffff are optional; if required, installation personnel must supply the one to four alphanumeric characters to be used.



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

Figure 37-6. Pointer Addresses

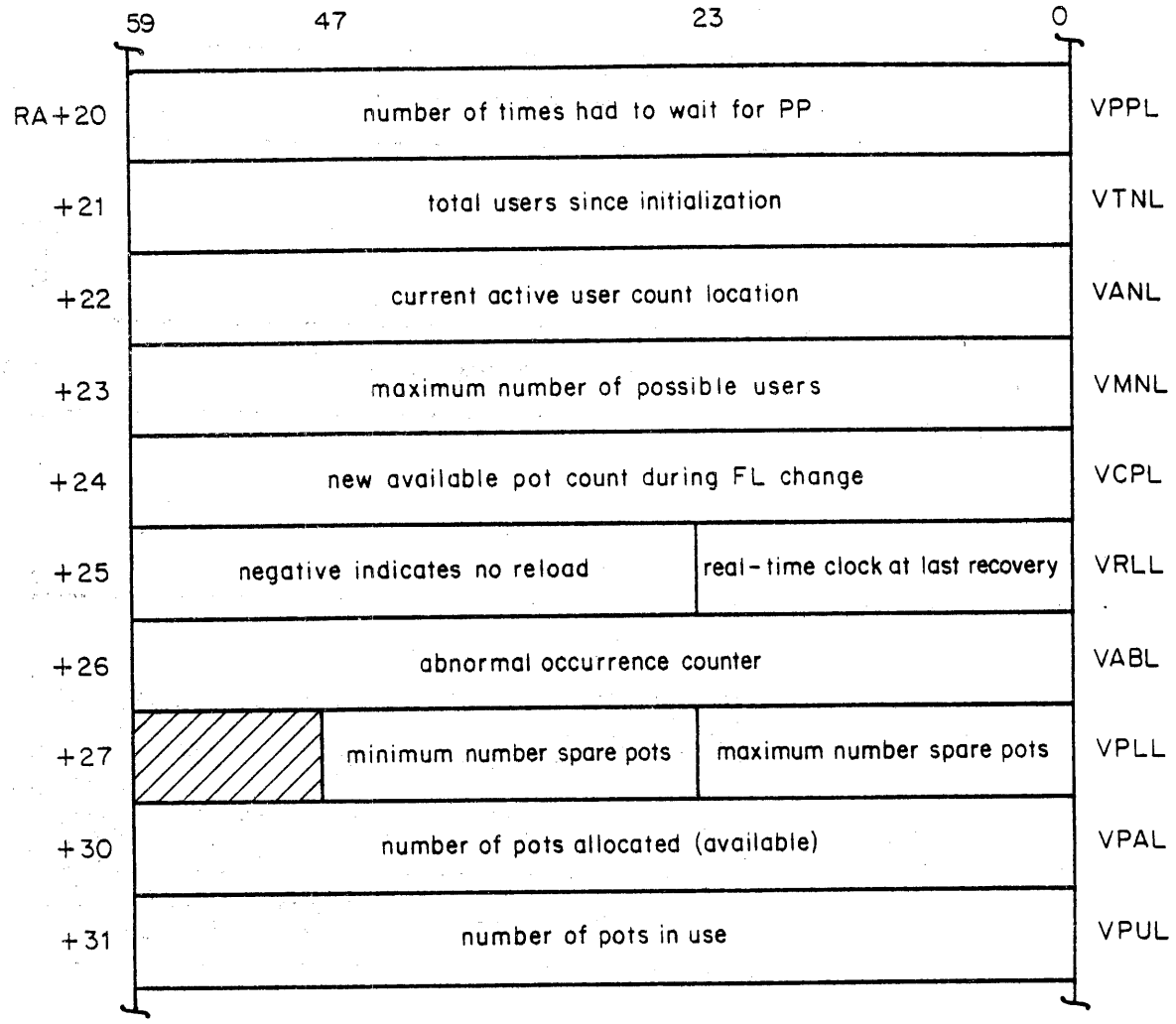


Figure 37-6. Pointer Addresses (Continued)

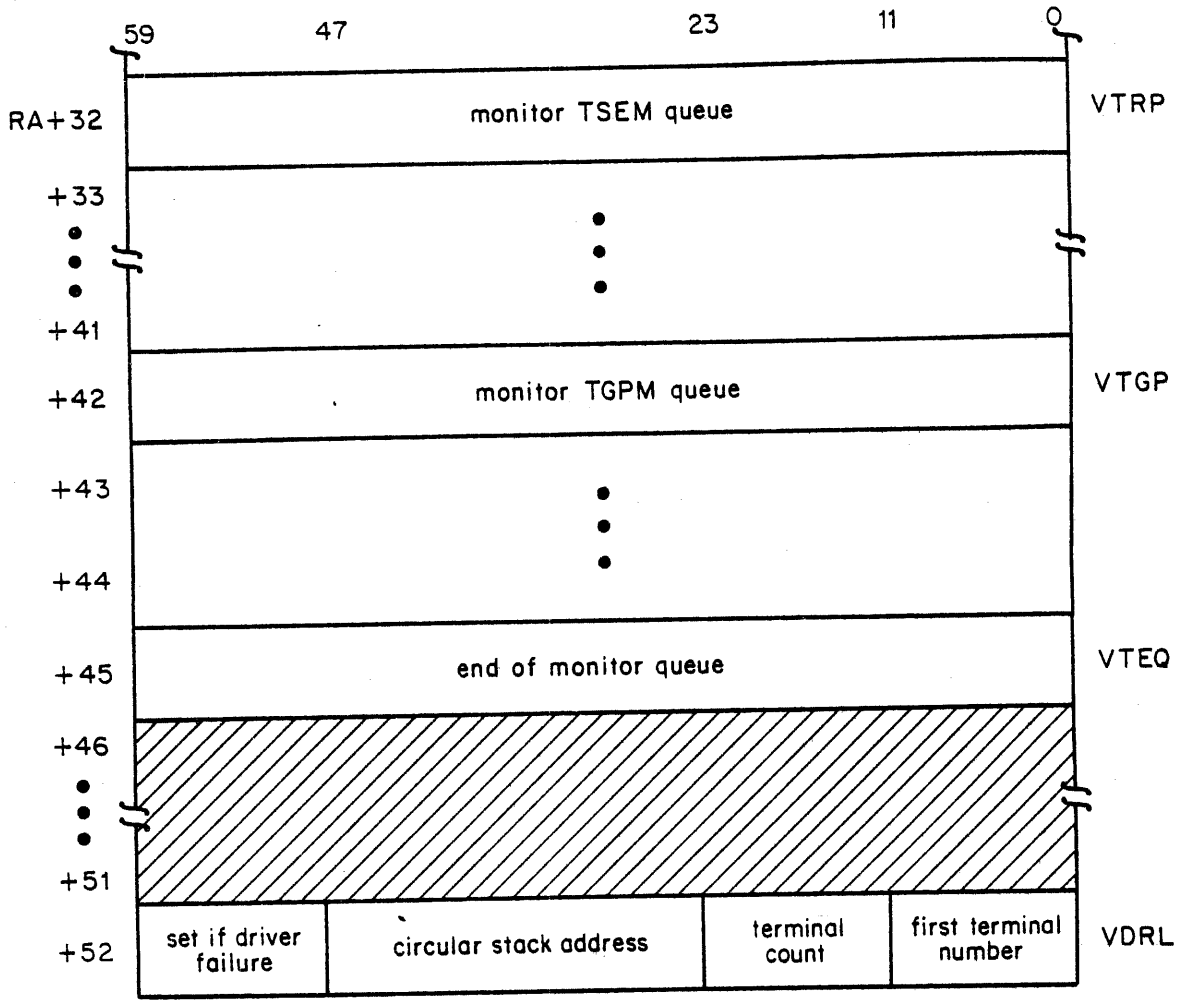


Figure 37-6. Pointer Addresses (Continued)

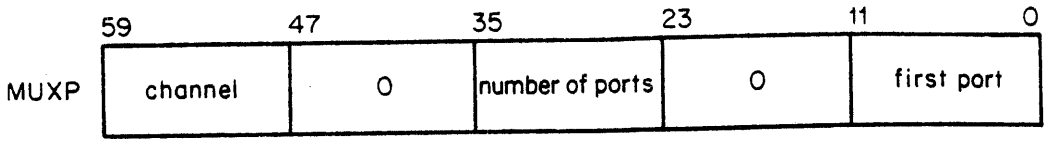
TABLE 37-1. IAFEX CONSTANTS

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	IAF 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	10D	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	20D	Network terminal SALVARE file time-out value (minutes)
VOPL	4	Number of pots issued for multiplexer
VRBL	--	Reserved
VSBL	--	Reserved
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
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	1024D	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

TABLE 37-1. IAFEX CONSTANTS
(Continued)

Constant	Value	Description
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	--	Reserved
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 stimulated multiplexer in the EST which is on plus an entry for the network interface.



After initializing the tables, IAFEX modifies addresses in IAFEX1 and IAFEX4 code which use the increment instruction operation definitions. Next, each terminal table entry is set to complete status by setting VR0T equal to 3 in each entry. Next, the warning message address VWMP is set to the normal header. Next, IAFEX calls 1TA to search for time-sharing jobs in the system. The jobs searched for are TX0T and MT0T type. The count of such jobs is returned in a pseudo terminal table for IAFEX. If the count is nonzero, IAFEX aborts with the message: IAFEX 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), IAFEX is complete and control is transferred to IAFEX1.

IAFEX1 - MAIN PROGRAM

IAFEX1 is the main program that controls and coordinates the time-sharing subsystem. This program is driven by the following queues.

- Request entering IAFEX:

Queue	Description
Driver request	Requests from 1TN and IAFEX4
Monitor request	Requests from other PPs
Monitor pot request	Requests from other PPs for pots

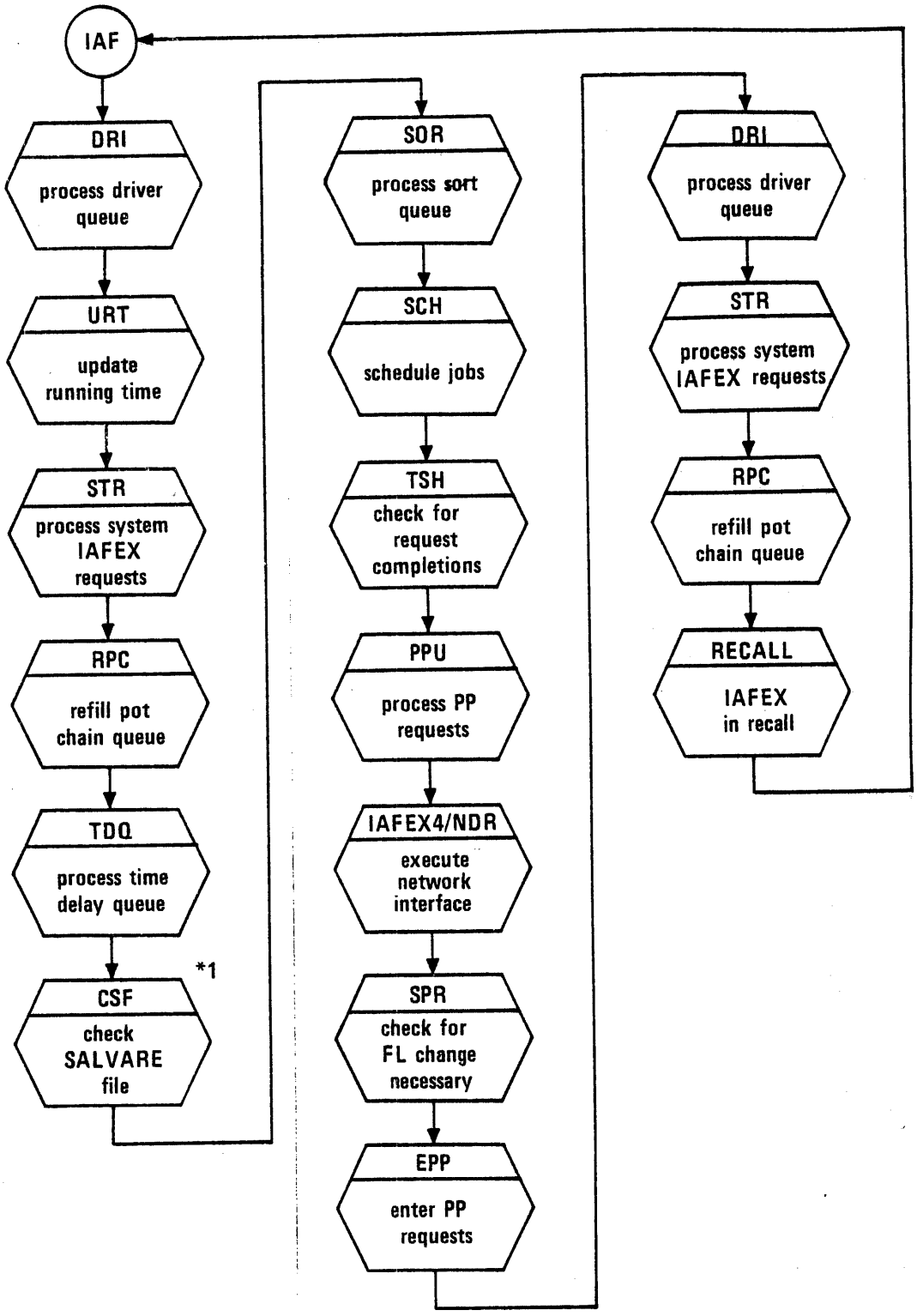
- Internal control:

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 IAFEX:

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

These queues are scanned by the IAFEX1 control loop which is defined in the flowchart of figure 37-7.



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

Figure 37-7. IAFEX1 Control Loop

Every 3 minutes the SALVARE file entries are checked and if the time is over VNTO 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 VNTO minutes of system recovery or the SALVARE file entry will be eliminated. The SALVARE file is discussed further under SALVARE - IAFEX Recovery File in this section.

The relationship between processing modules of IAFEX1 is shown in figure 37-8.

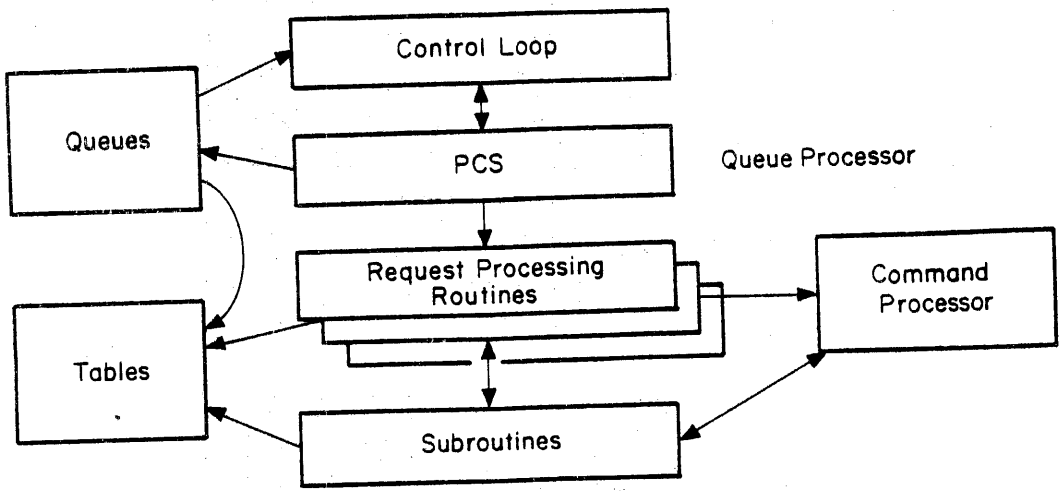


Figure 37-8. IAFEX1 Processing Modules

In general, all tables in IAFEX 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 IAFEX 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 IAFEX will never be moved. Thus, no SYSEDITs which change the size of CMR can be run while IAFEX is running. The IAFEX1 memory layout is shown in figure 37-9.

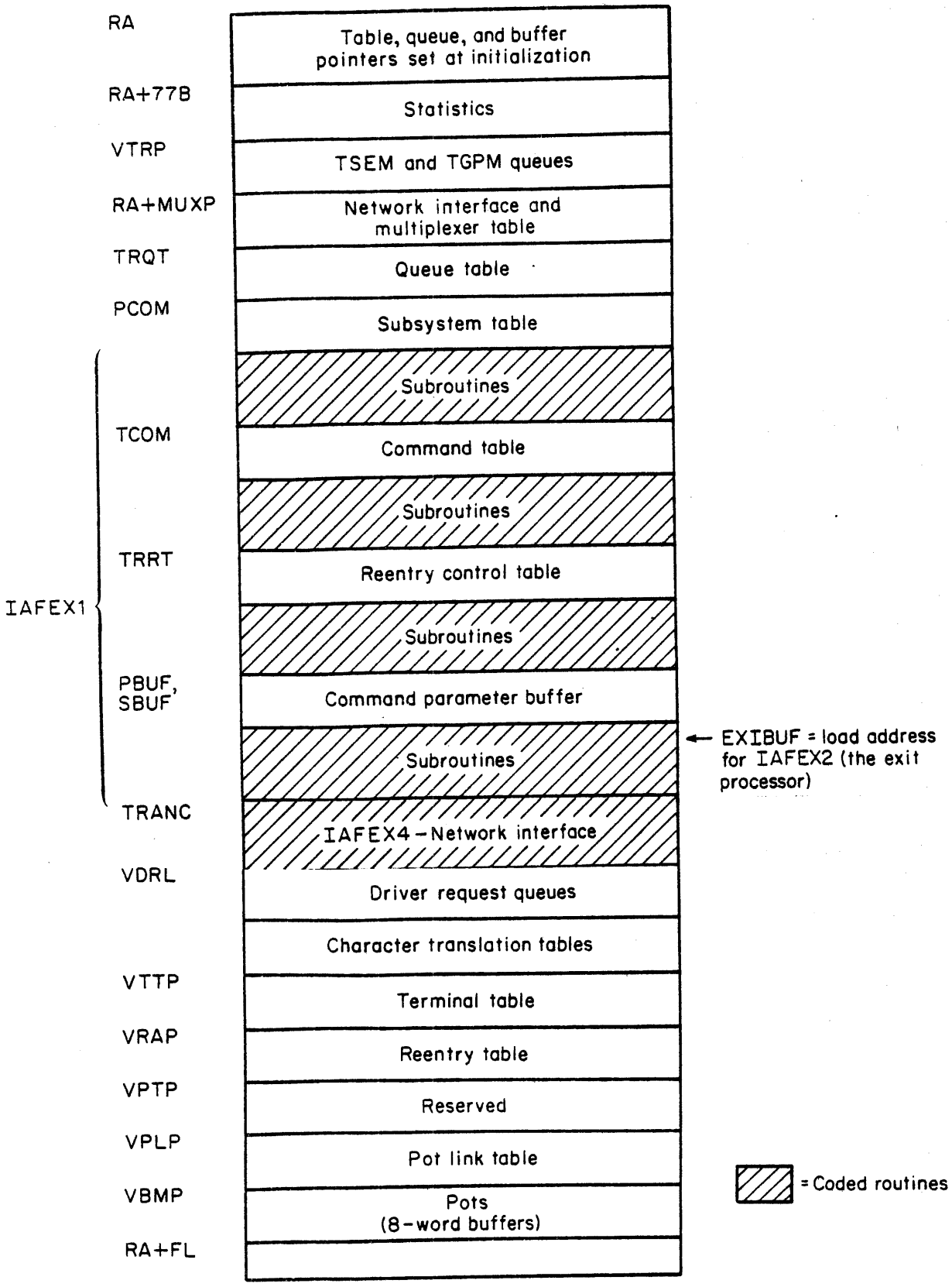


Figure 37-9. IAFEX1 Memory Map

DRIVER REQUEST QUEUE(S)

Driver requests are passed to IAFEX1 via the driver request queue which are circular stacks as shown in figure 37-10.

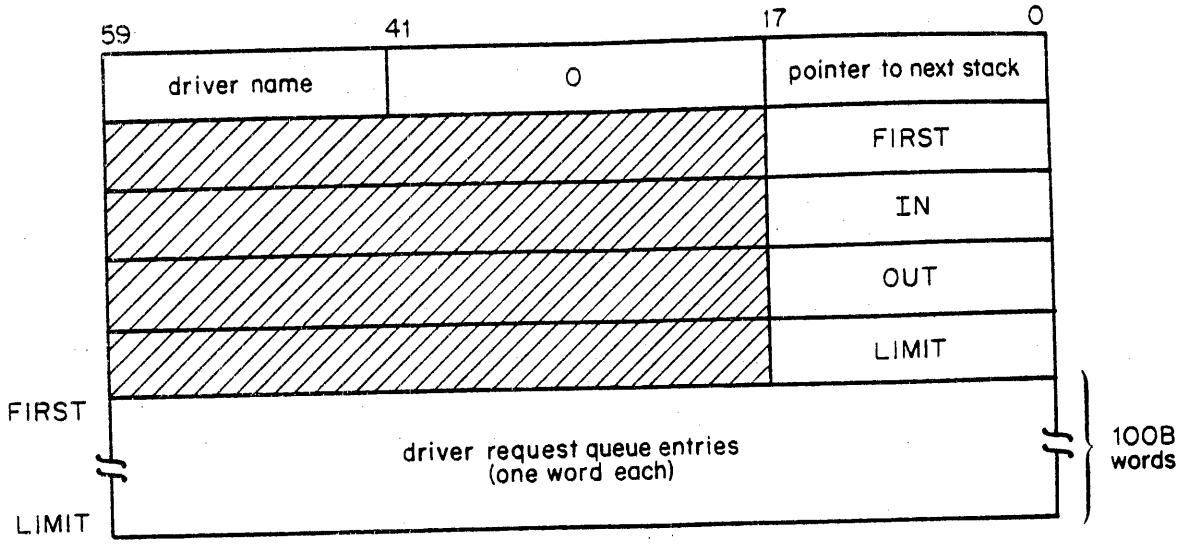
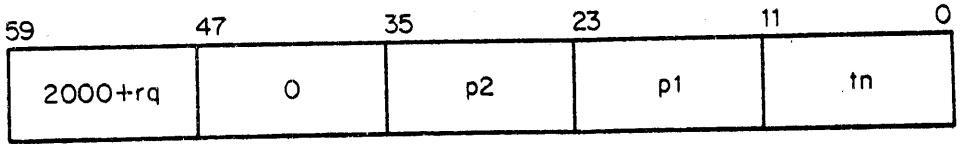


Figure 37-10. Driver Request Queue Stack

Driver request queue entries are placed in a circular stack by the driver.* The IN pointer is updated by the driver when an entry is placed into the queue. IAFEX1 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. The entries are one word as follows:



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

*The driver is either IAFEX4 when no STIMULA-type terminals are being used or IAFEX4 and/or 1TD if STIMULA is in use.

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
```

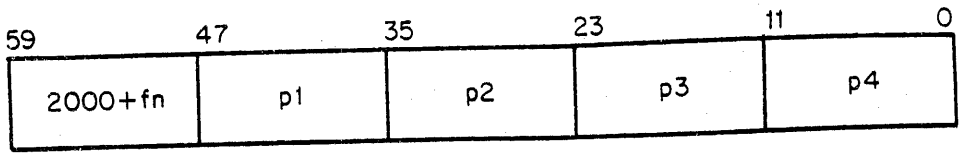
The result is that B7 contains the request number and X1 contains the parameters and terminal number (that is, the lower 48 bits). A list of request numbers (request codes) is maintained in common deck COMSTDR and are listed in table 37-2.

TABLE 37-2. DRIVER REQUEST NUMBERS (ISSUED TO IAFEX1)

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	Reserved
14	MTN	Reserved
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	Reserved
22	SSC	Reserved
23	TTI	Reserved
24	EMO	Reserved

MONITOR REQUEST QUEUE(S)

PP requests for IAFEX 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 37-3.

TABLE 37-3. 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	Reserved
2007	VSBS	Set subsystem

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

In general, IAFEX 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) in the output data except when network ASCII data has been generated, since this mode has a character count in the data header.

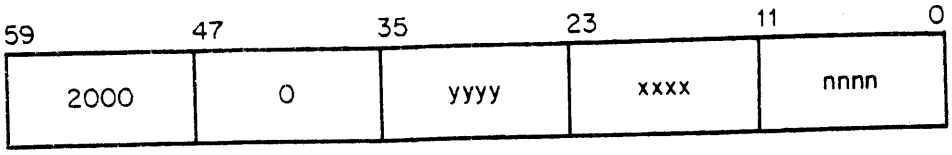
NOTE

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 IAFEX in a 3-word queue similar to TSEM requests.

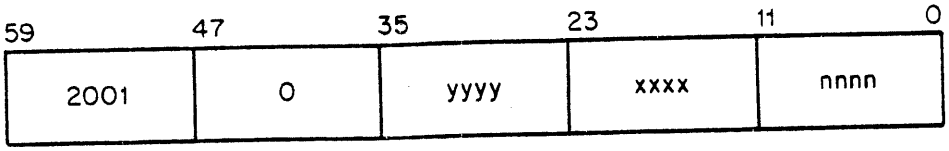
The parameters for the various functions are defined as follows.

VDPO - Drop Pots (IAFEX1 Routine DRT)



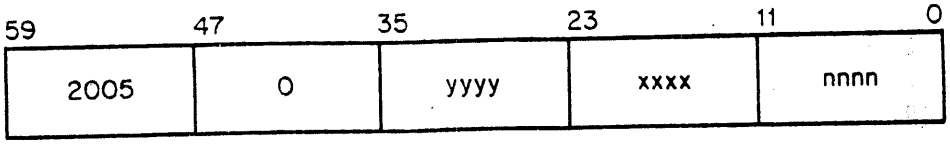
yyyy	Last pot to be dropped
xxxx	First pot to be dropped
nnnn	Terminal number

VASO - Assign Output (IAFEX1 Routine ASO)



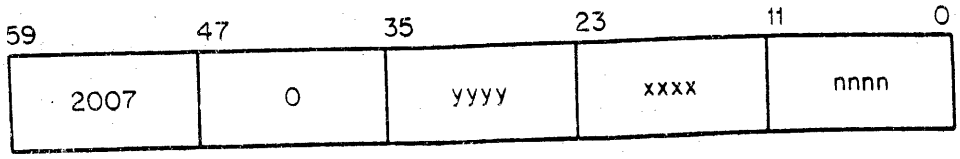
yyyy	Last pot of output
xxxx	First pot of output
nnnn	Terminal number

VSCS - Set Character Set Mode (IAFEX1 Routine SCS)



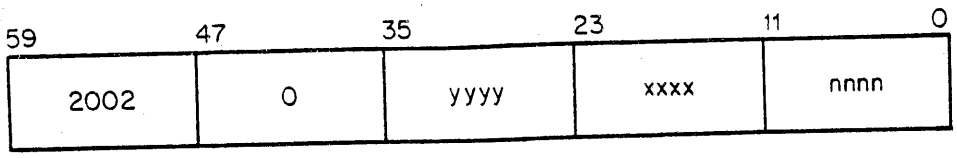
yyyy	Last pot containing mode
xxxx	First pot containing mode
nnnn	Terminal number

VSBS - Set Subsystem (IAFEX1 Routine SBS)



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

VMSG - Assign Message (IAFEX1 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

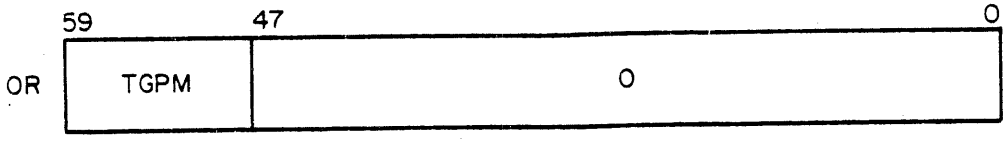
VMSG is used by DSD to process the DIAL and WARN operator commands.

VSDT and VCDT TSEM Requests

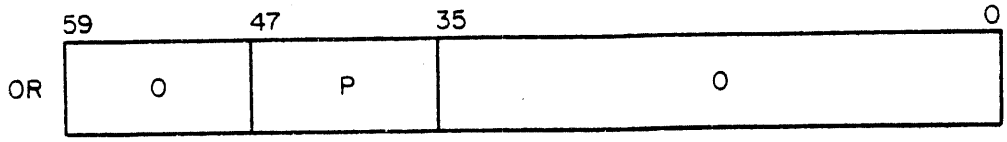
When a terminal user initiates a CPU program, he may terminate that program with termination or user break 2 (UB2) sequence (refer to the IAF Reference Manual). If the user wishes to disable/enable this function he can use the DISTC macro described in section 12 of the NOS Reference Manual, volume 2. 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 IAFEX 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 IAFEX 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 IAFEX.

During IAFEX'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.

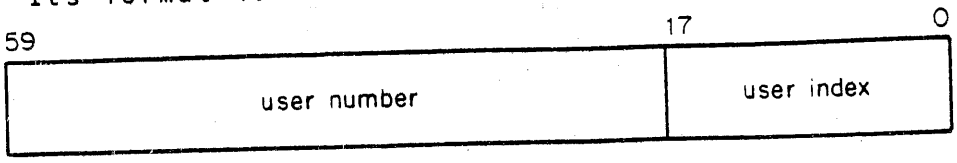
The TGPM is used by 1TO, which requests pots for flushing a TXOT type job's OUTPUT file. Another user is DSD, who must get a pot chain for the WARN and DIAL messages.

TERMINAL TABLE

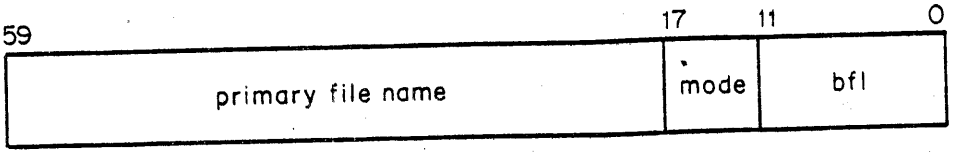
The terminal table contains an eight-word entry for each possible active user. Each entry contains the current status of each connection. These eight-word entries are structured in such a way so as to minimize interlocks between IAFEX1 and the various FP routines which read and write them.

IAFEX4 and the network terminal processing routines of 1TO use terminal tables VFST, VDPT, and VCHT to maintain terminal operations parameters.

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



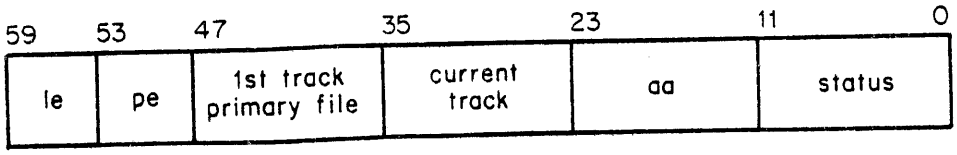
Word 1 (VFNT) is written by IAFEX, 1RO, and 1TA and read by 1RI, 1TA, IAFEX, 1RO, and the driver. Its format is as follows.



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 IAFEX, 1RO, 1TA, 1RI, and 1TO and is read by IAFEX, 1RO, 1TA, 1RI, and 1TO.

VFST is used by 1TO to maintain status information between output operations involving an output line that has been split across pot chains. This word uses the VROT interlock; that is, when VROT is interlocked, 1TO may write word VFST. Its format is as follows.



le	List file equipment number
pe	Primary file equipment number

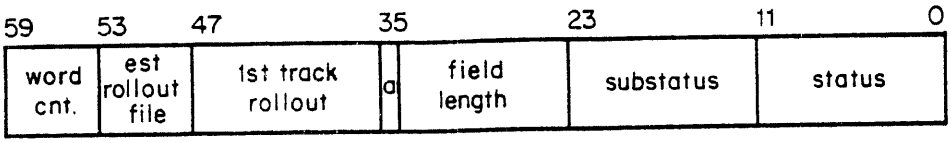
aa One of the following:

- List or primary file current sector
- Control statement pot pointer during job scheduling
- Accounting pot pointer during logout

status Output status bits:

Bit	Description
11-6	Unused
5	Line continuation
4	7400 escape character
3	7600 escape character
2	Binary
1	Transparent
0	Extended

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



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

substatus:

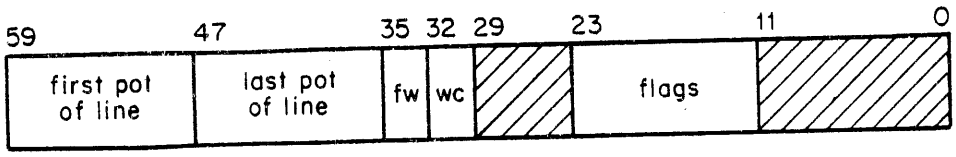
File list if 0 (list with EOR and EOF if 1)

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
IAFEX 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
Special system job	5	1
List	6	1
Multi-terminal	7	1
Suspended	9	1
Not used	10	
Error on last operation	11	1

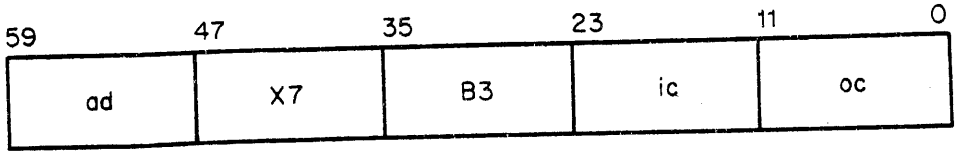
Word 4 (VDPT) is used by IAFEX4 and the network processing routines of 1TO to maintain terminal operations parameters. Its format is as follows.



fw	First word of first pot
wc	Last pot word count
flags	Each bit defined as follows:

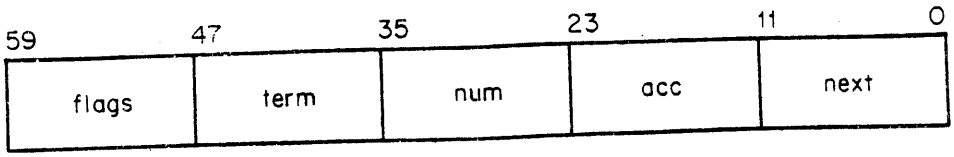
Bit	Description
23-17	Unused
16	Source input initiated
15	Binary input
14	Transparent input
13-12	Unused

Word 5 (VCHT) is also used by IAFEX4 and the network processing routines of 1TO to maintain terminal operations parameters. Its format is as follows.



ad	Reentry address relative to NDR
X7	12-bit X7 parameter
B3	B3 parameter; assumed to be pot pointer
ic	Input character count
oc	Output character count

Word 6 (VDCT) is written and read by IAFEX1 and IAFEX4. 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	----	Reserved
5	----	Reserved
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 IAFEX1 byte 4

term Terminal control information as follows:

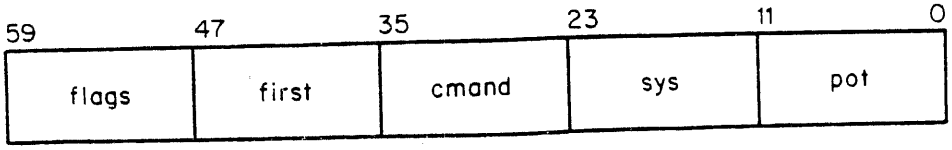
Bit	Value	Meaning
11-3	--	Not Used
2-0	0-7	First word of output line in pot

num In AUTO mode, the line number increment

acc Access control flags (lower 12 bits of access word defined in VALIDUs file for this user). Refer to the NOS Installation Handbook for procedures to establish the access word. Refer to section 20 for a description of the bits currently defined for the access control word.

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 IAFEX1 and is read by IAFEX1, IAFEX4, IAFEX4, 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		
		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 37-4 is a summary of the terminal table entry.

TABLE 37-4. TERMINAL TABLE ENTRY SUMMARY

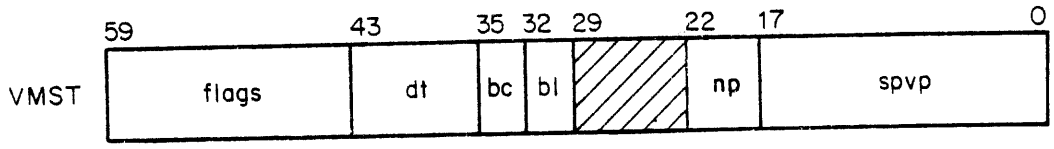
Name	Word	Written by	Read by
VUIT	0	IAFEX, 1TA	IAFEX, 1TA
VFNT	1	IAFEX, 1RO, 1TA	IAFEX, 1RI, 1RO, 1TA, IAFEX4
VFST	2	IAFEX, 1RI, 1RO, 1TA, 1TO	IAFEX, 1RI, 1RO, 1TO, 1TA
VROT	3	IAFEX, 1RI, 1RO, 1TO, 1TA	IAFEX, 1RI, 1RO, 1TO, 1TA, IAFEX4
VDPT	4	IAFEX4	IAFEX, IAFEX4
VCHT	5	IAFEX4	IAFEX4
VDCT	6	IAFEX, IAFEX4	IAFEX, IAFEX4
VSTT	7	IAFEX	IAFEX, 1RI, 1RO, 1TA, IAFEX4, 1TO, DSD

In table 37-4, the name IAFEX refers to any of the overlays comprising IAFEX except IAFEX4. Any routine which writes a word also is assumed to read that word.

NETWORK TABLES

IAFEX4 uses two dynamic tables in addition to the nonnetwork tables. These tables, the message status table (VMST) and the network activity table (VNAT), are allocated only if network tables are defined.

The message status table VMST contains network terminal control information and supervisory message pointers. Its format is as follows.



flags Each bit defined as follows:

Bit	Description
59	Terminal on-line
58	Suspend traffic
57	Break in progress
56	Shutdown warning sent
55	End-connection in progress
54	Data received previous cycle
53	MSG block sent (input enabled)
52	Data present on NAM (used only during break processing)

dt	Device type
bc	Unacknowledged downstream block count
bl	Application block limit
np	Number of pots allocated for output pot string
spvp	Pot pointer of pot containing supervisory message

The network activity table VNAT indicates which network terminals require service from the terminal manager. One bit is used for each network terminal, 32 bits per CM word. A terminal's activity is set when the terminal requires service by the network driver. A terminal's activity bit may be found by the following algorithm.

w = (tn - ft) / 32

w	Word location relative to start of table
tn	Terminal number
ft	First network terminal number

rb = (tn - ft) mod 32

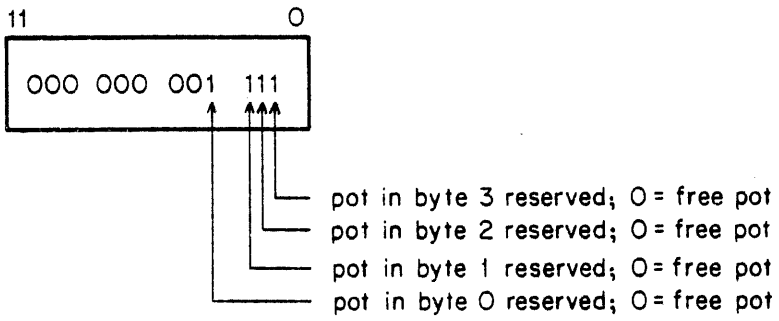
rb	Bit location relative to bit 59

POT LINK TABLE

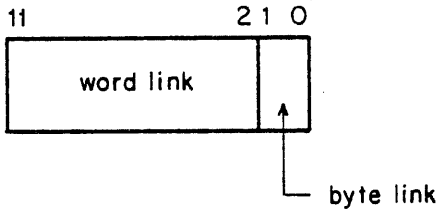
The pot link table (PLT) controls the use of pots (8-word buffers). Its layout is as follows.

	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

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.



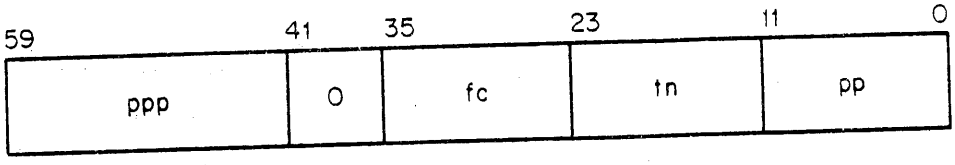
In the preceding table, pots 1-5 are reserved and comprise one chain. Pots 6 and 7 comprise another chain. Pot 10 is free. Pot 11 is the start of another chain. Reserved chains need not be contiguous.

INTERNAL QUEUES (TRQT)

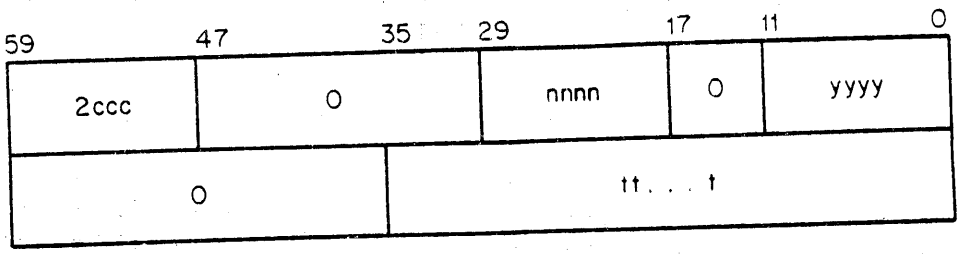
All internal queues are built at assembly time in a table of queues. This table consists of all the queues that may have requests in the reentry table. The following is a list of valid queue names in the table of queues.

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



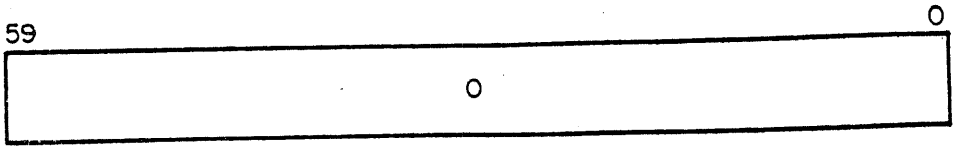
ccc	Number of entries (packed format)
nnnn	First terminal entry (index into reentry table)
yyyy	Last terminal entry (index into reentry table)
tt...t	Resource control count

NOTE

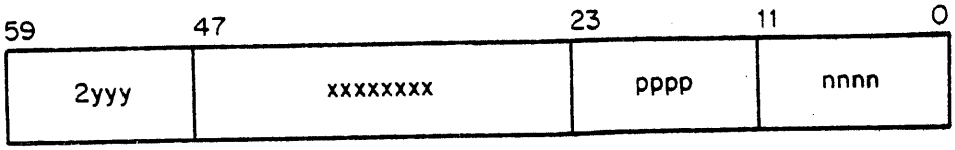
Each queue has an associated string of entries in the reentry table.

REENTRY TABLE (VRAP)

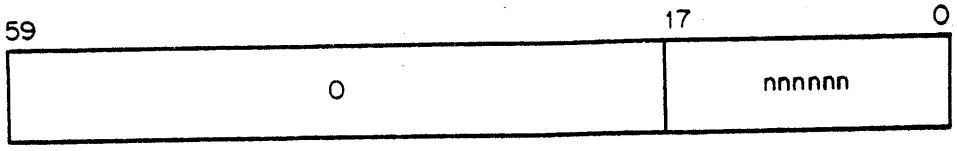
The IAFEX subroutines use the reentry table to have control returned or functions performed for them when a set of conditions are met. The table consists of one word for each terminal with one of the following formats.



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)

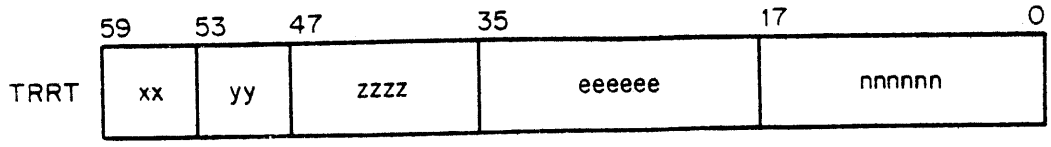


nnnnnn Pot address of stacked entries

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 IAFEX.

Figure 37-11 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.

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 IAFEX.

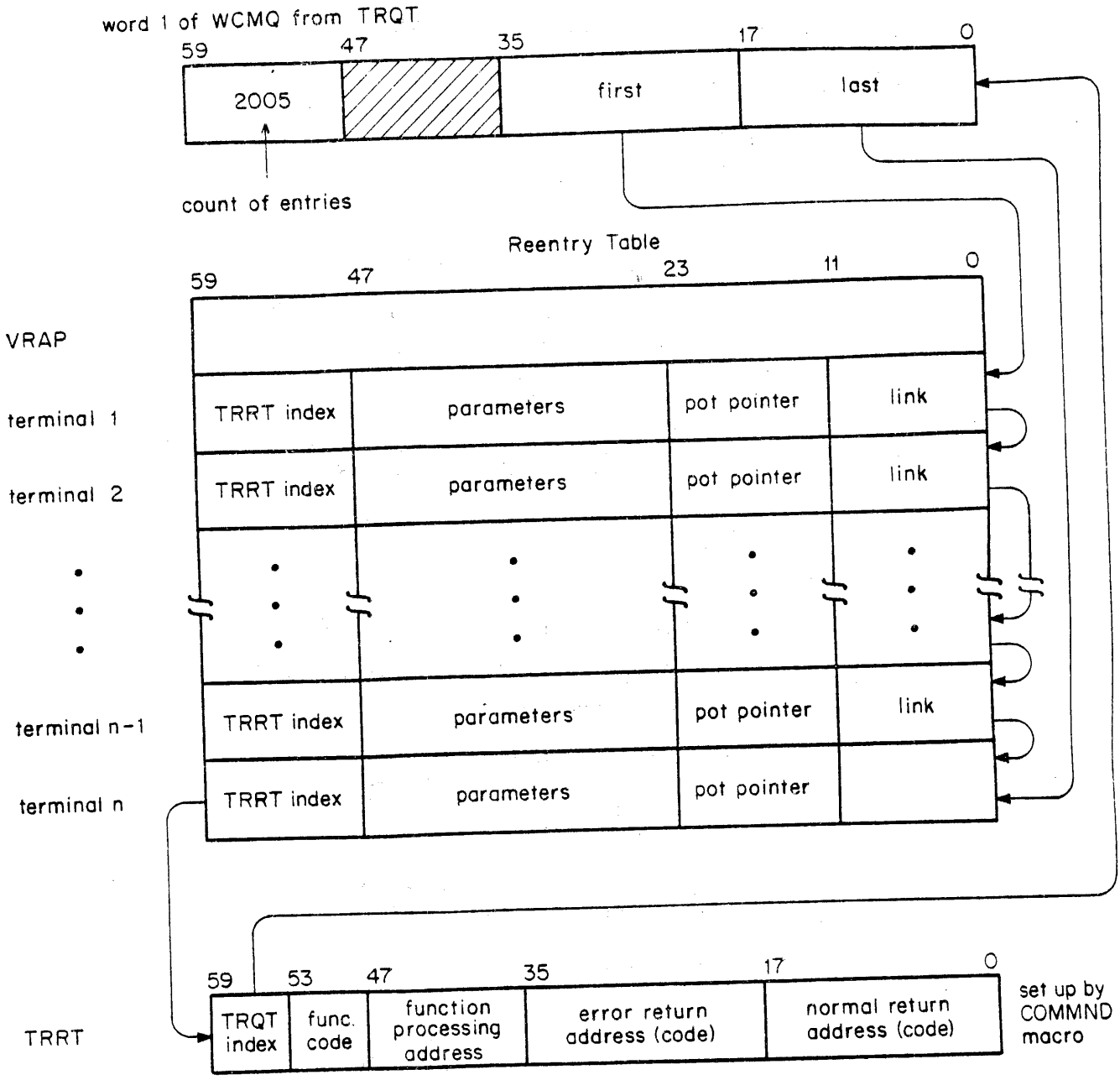


Figure 37-11. Table Relationships

IAFEX ROUTINES

The following is an outline of the subroutines comprising IAFEX1.

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

WCMQ	ITAQ
TIMQ	ITOQ
JOBQ	PFMQ
SORQ	

- IAF - Control loop; calls the following:

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

- 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	06S	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

IAFEX2 - TERMINATION OVERLAY

IAFEX2 perfoms exit processing for the time-sharing 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 IAFEX1 a jump to the abort subroutine (ABT) is executed. ABT issues the message

IAFEX 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 (IAFEX2) is found, and 1AJ has the termination routine loaded. IAFEX2 is loaded in a buffer which leaves the tables and queues untouched. Basically, IAFEX2 dumps the FL if requested and loads IAFEX3 which 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 IAFEX. Refer to the NOS Operator's Guide for more information.)

IAFEX4 - IAF/NAM INTERFACE

The IAFEX4 overlay to IAFEX provides the interface between the Network Access Method (NAM) protocols and message formats and the internal characteristics of IAF. Figure 37-12 depicts the general organization of the IAFEX4 overlay; figure 37-13 shows the organization of the entire field length. The following paragraphs describe the internal design of the interface and its relationship to the time-sharing executive and NAM.

The basic purpose of IAFEX4 is to send and receive data from terminals which are connected through NAM to IAF. In order to accomplish this for IAF, IAFEX4 provides the control and connection management for all connected terminals. The features provided are as follows.

- Receives, interprets, and sends supervisory messages
- Transforms outgoing data from internal forms to the appropriate network format
- Transforms incoming data from network format to internal format
- Manages data traffic to optimize interactive performance.

A component of NAM called the Application Interface Program (AIP) resides in the field length of IAFEX (refer to figure 37-13) and provides the data interchange between NAM and IAFEX. The details of the AIP are not described here; the reader should refer to the NAM Reference Manual for the background information required to fully understand the details and requirements of the network interface.

The functions of IAFEX4 are described in relation to the terminal functions available to the user rather than the internal organization of IAFEX4. This approach should provide a logical organization within which to describe the operation of the network interface overlay. The following functions are described.

- Connection establishment (login)
- Command line entry
- Source line entry
- Input to a running program
- Output processing
- Session termination

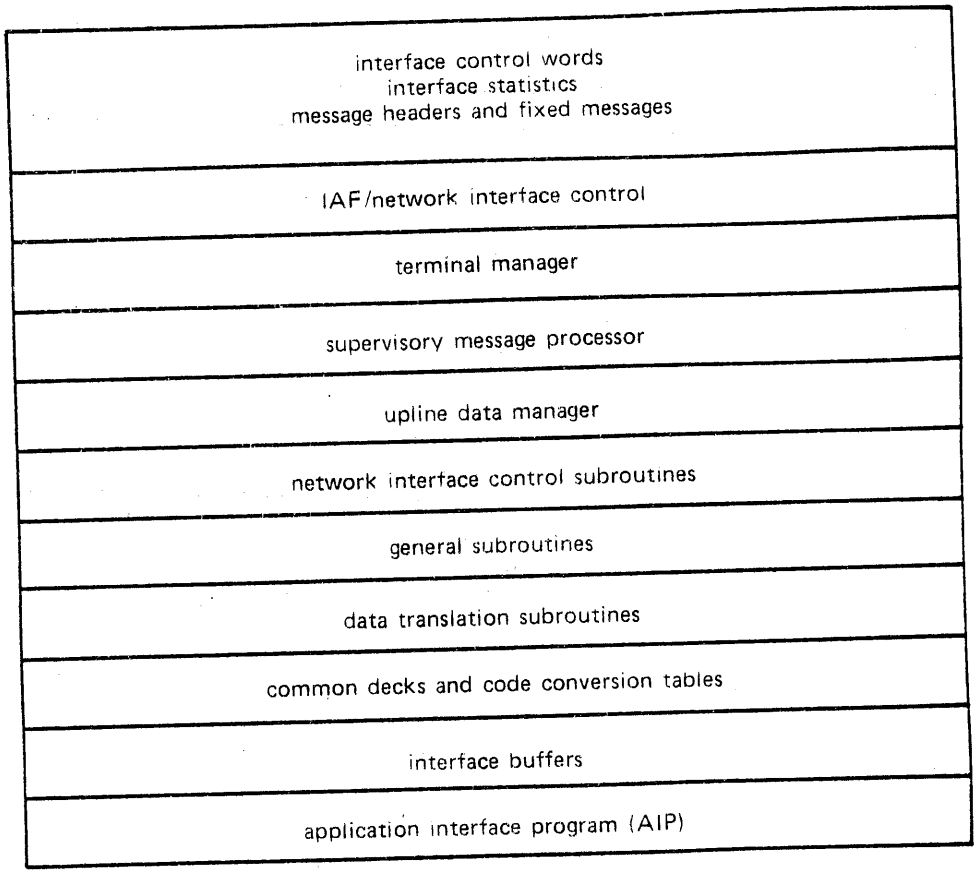


Figure 37-12. IAFEX4

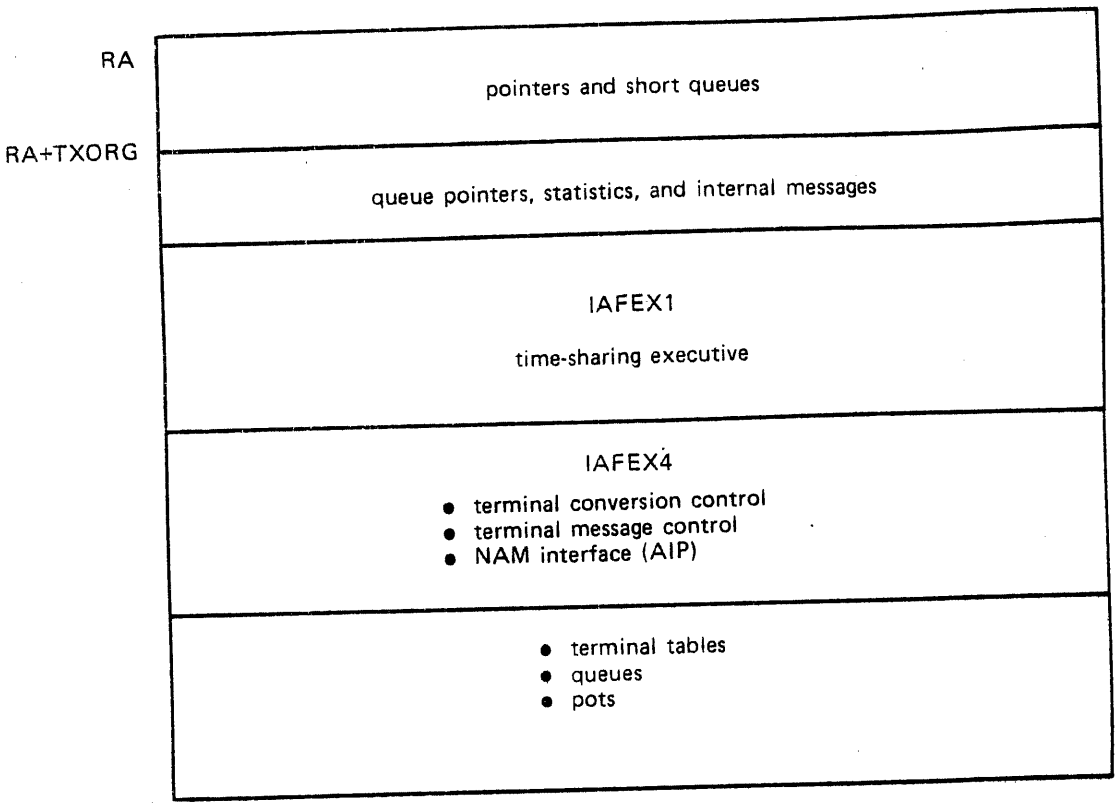


Figure 37-13. IAFEX Control Point

CONNECTION ESTABLISHMENT

Prior to connection to IAF the user completes the login dialog wth NVF (NETVAL). NAM then sends a connection request (CON/REQ) supervisory message to IAF which is interpreted by the supervisory message processor (SMP) and transferred through PCS to the connect function code processor (CON). Since there are a number of variations to the connect function, CON checks the subfunction code and determines that this is a new connection. CON then transfers control to NWC to process the new connection.

The CON/REQ message includes the family name, user number, user index, and validation information which NVF has read from the validation file. This information is moved into a pot and a pointer to that pot is placed in VDPT for 1TA so login can be completed. Terminal table words VCHT and VUIT are initialized and the terminal is set to on-line status in the message status table (VMST). Information required to control the terminal (device type and application block limit) is also stored in the VMST entry at this time.

The last step required to complete this phase of the connection process is to send a connection accepted supervisory message to NAM. This is done in NWC by calling ASV to assign the message after which NWC exits to PCSX which returns to SMP to complete processing of this message. The terminal is not yet fully logged in since 1TA has not yet been called to perform the login process.

The last step in the connection protocol is to receive and reply to the connection initialized (FC/INIT) supervisory message. Until this happens terminal dialog cannot occur. SMP interprets the FC/xxx supervisory message and transfers control through PCS to the proper processor, this time FCN. FCN selects the proper subfunction processor for the FC/INIT (CNM) and transfers control to it to process the message.

CNM first calls ASV to respond to the FC/INIT and then exits to PCS4 to make an ITA3$ queue entry in order to complete the internal login process for this terminal. The exit to PCS4 returns to PCSX which returns to SMP to complete the processing of the supervisory message.

COMMAND LINE ENTRY

When network driver main control (NDR) detects a data message from a terminal, the read data manager (RDM) is called to process the message. RDM extracts the connection number from the message and validates the number so it can be related to a terminal table entry. The connection number (ACN) and the terminal table entry number are identical. Assuming that the terminal is in a normal state and no resource limitations are currently exceeded, RDM calls PIN to process the input data.

PIN determines the correct internal format for this data and calls one of the translation routines to convert the received data from NAM to IAF format and character set. PIN calls the end-of-line processor (EIL), which in this case calls ECL to enter the command line to the executive. Pot pointers in VDPT are cleared and ERQ is called to enter a CLI request in the driver circular stack. This request causes the executive to accept and process the command line. Control now returns to RDM which indicates that this network input data has been accepted. RDM then returns to main control (NDR) which directs the next operation.

SOURCE LINE ENTRY

A source line is data received in text mode or a line of data which begins with a digit. Input which is source is to be placed on the primary file and this requires slightly different processing than command line data in the network interface.

All processing for data of this type is the same as for command line entry discussed above up to the point where EIL (process end of input line) is called. EIL detects that the input is source and calls ESL to enter the source line. Since source data is buffered prior to writing it to the primary file, ESL does not actually release the input data unless it exceeds one pot. Assuming that such a condition exists, ESL calls ERQ to make an RIN driver stack request, resets the input initiated bit, and returns control to RDM which completes processing in the same way as for command line entry.

INPUT TO A RUNNING PROGRAM

This is handled in the network interface in the same way as command line entry. The read-data bit in terminal table word VDCT forces this to occur and the command line processor in the executive directs the data to the program requesting it.

OUTPUT PROCESSING

Processing of output to terminals in the network interface begins when main control (NDR) calls the terminal manager (MGR). The terminal manager, after performing a number of tests necessary for timely processing of supervisory messages and internal functions, calls PQO to process queued output.

In the simple case where no special control bytes need be processed and there is output indicated in VDCT or VSTT, PQO formats the network message header and calls SPC to send the pot chain containing the message to NAM.

SPC checks the size of the output because the exact method of sending data to NAM depends on whether the data is in a contiguous block or fragmented in a number of linked pots.

If the data is in a contiguous space, SPC makes a NETPUT call to send the data. When the data is fragmented, SPC builds the send vector table which is sent to NAM with a NETPUTF request. NAM, using the send vector, extracts the data fragments from the pots and sends the data to the terminal. The pots holding the data must not be released until NAM indicates it has taken the data. (This is true for both NETPUT and NETPUTF transmission techniques.)

SPC now returns to PQO in order to enter a job restart (RES) request through ERQ. After the restart request has been issued, PQO returns to the manager (MGR) whih controls processing of the next terminal.

Dropping of the output pots can be done when the request is complete. Routine CKP performs this check and drops the data pots. In the case where output is not immediately accepted, cells OBSY and NBSY are set nonzero and OTPP holds the pot pointer of the still-active data chain. MGR exits upon finding NBSY nonzero since no further network accesses are possible until the condition clears. This exit also causes NDR to exit to the executive. The next time NDR is called, CKP is executed. Since NBSY, OBSY, and OTPP are still set from the previous request, the output operation is post-processed and normal network activity continues.

SESSION TERMINATION

A session can be terminated in one of the following manners.

- User logoff command
- Logoff control byte in the output
- Unexpected line failure
- Network immediate shutdown

The logoff command is detected by the IAFEX executive's command processor while the remaining conditions are detected in the network interface. A logoff of any type which generates a driver request is detected in NDR which calls TFR. Routine TFR transfer control to the disconnect processor (HUP) through PCS. HUP clears the pot pointer in VSTT and exits to ENC which ends the network connection.

A line failure or a similar condition which causes the connection with the terminal to be lost is detected by CON as a connection broken (CON/CB). CON exits to CNB which calls CUT to clean up the connection. Subroutine CUT queues an HUP request to the executive which performs the actions necessary to place the user on the recovery file. The executive in turn queues an HUP driver request which is processed in IAFEX4 by HUP which calls ENC to terminate the connection for this terminal in the same way as for the first case of logoff mentioned above.

The logoff caused by a logoff control byte is detected in PQO which calls LOF to enter an executive COF request. When the COF request is processed by the executive, it returns control to the network interface just as for the logoff command case above.

1TA IAFEX AUXILIARY ROUTINE

Routine 1TA processes functions for IAFEX which require PP action. The functions allowed are listed in table 37-5.

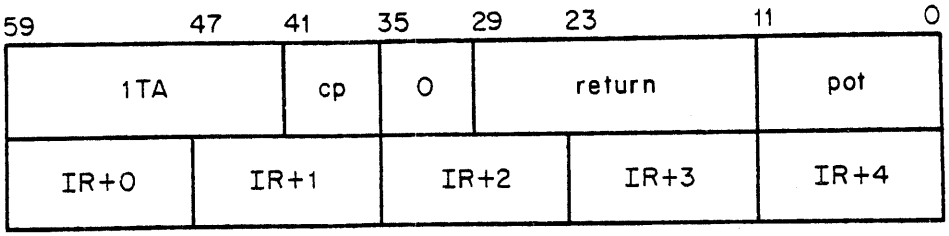
TABLE 37-5. PROCESS FUNCTIONS

Overlay Name	Function Code	Routine Name	Description
3TA	1	TFL	Adjust IAFEX 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

IAFEX calls 1TA in one of two ways.

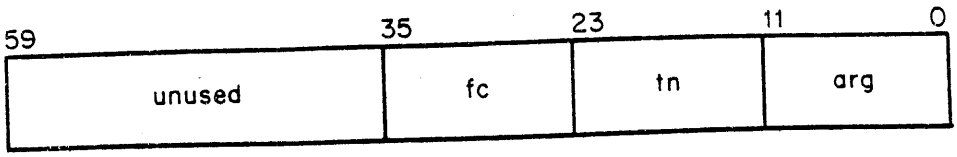
GROUP REQUEST

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:

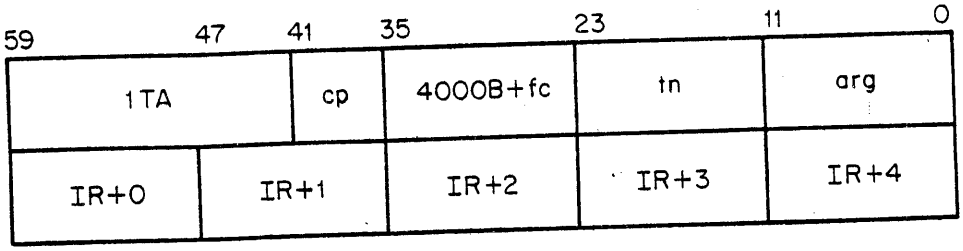


fc	Function code
tn	Terminal number
arg	Pot pointer or request type

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 IAFEX
10	Purge rollout FNT's
11	Error return

Figure 37-14 is the flowchart of the initialization, execution, and termination of the control loop for 1TA.

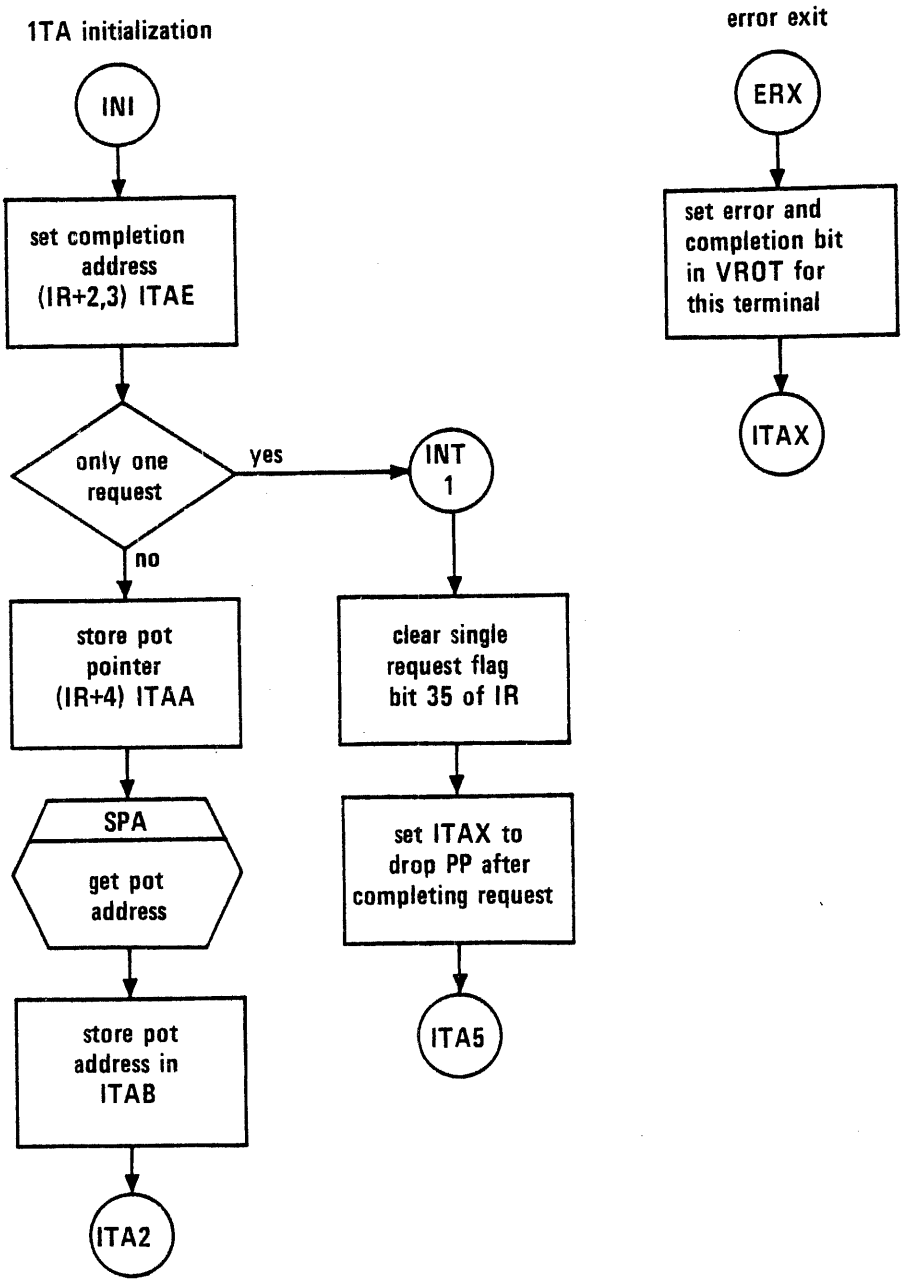


Figure 37-14. 1TA Control Loop

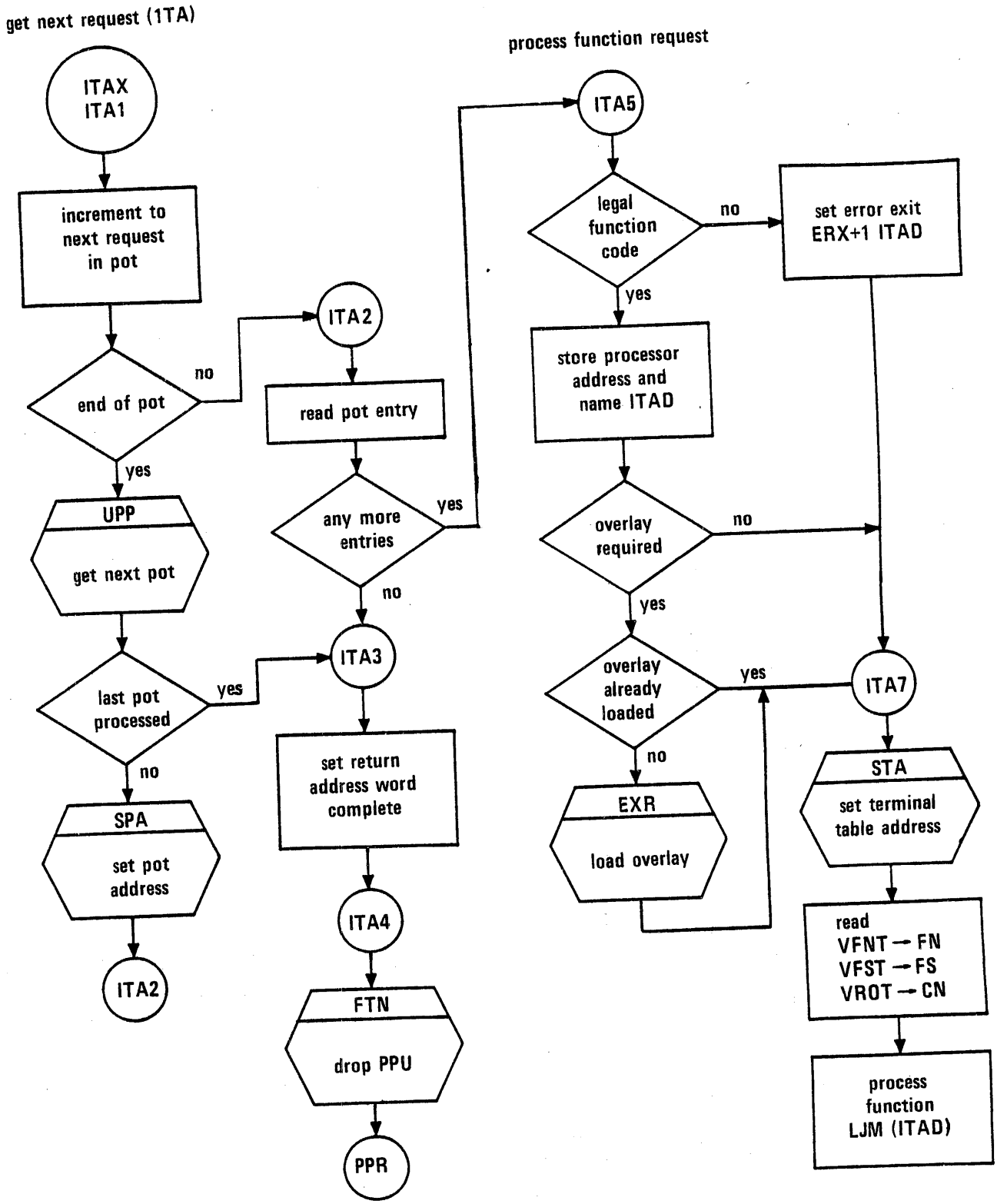


Figure 37-14. 1TA Control Loop (Continued)

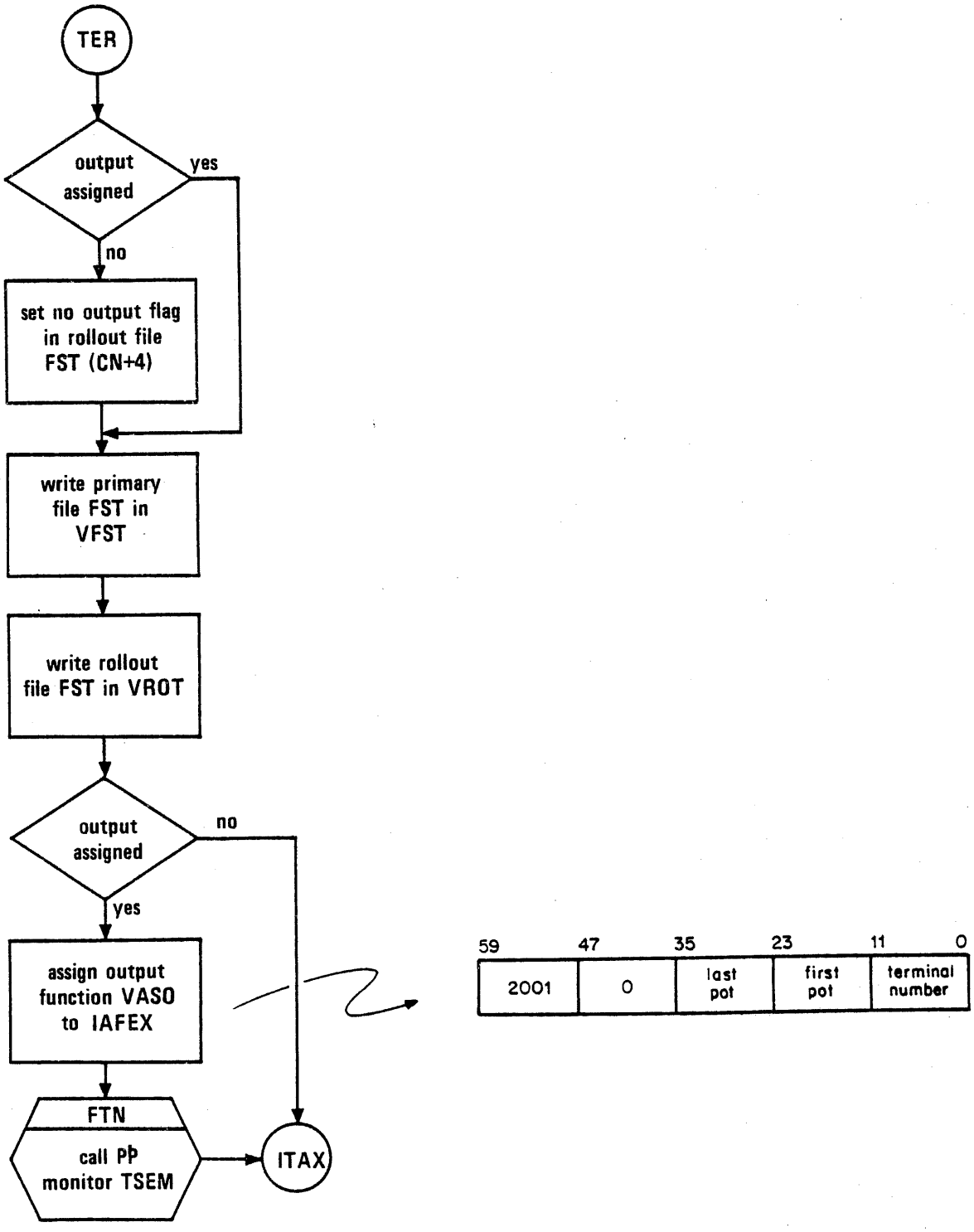


Figure 37-14. 1TA Control Loop (Continued)

Function 5 is used to create a rollout file for a time-sharing job. The format of the rollout file is given in figure 37-15.

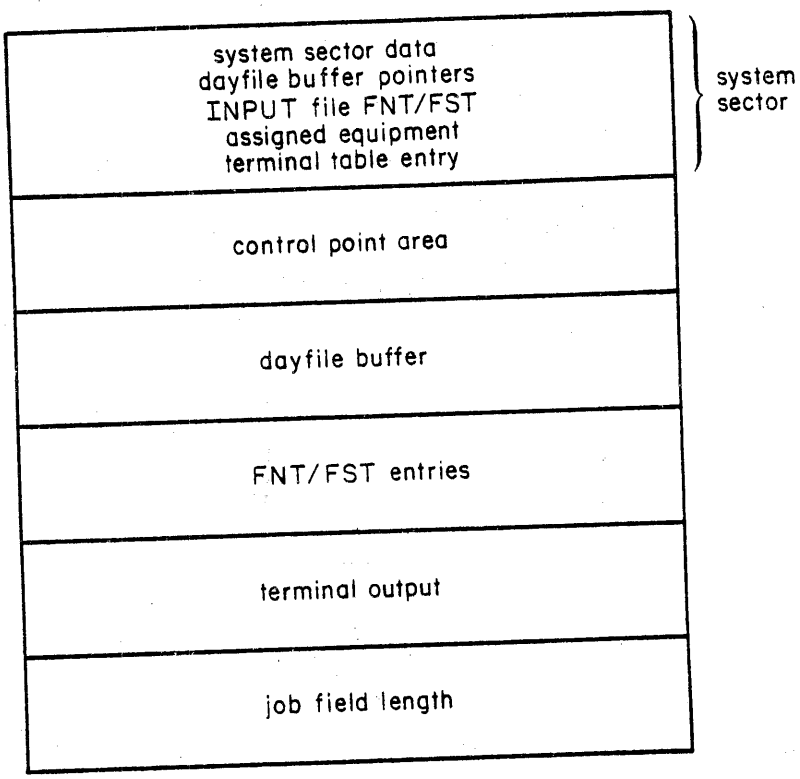


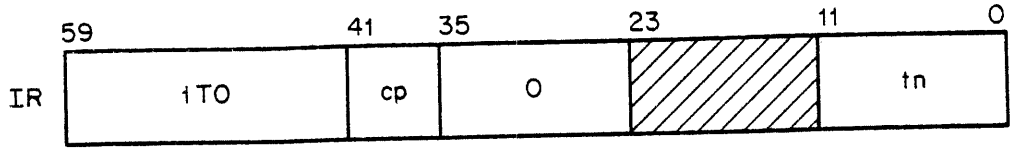
Figure 37-15. Time-Sharing Job Rollout File

1TO - TERMINAL INPUT/OUTPUT ROUTINE

Routine 1TO is called by IAFEX to process a queue of requests for terminal input and output which require disk accesses. The queue resides in pots within the IAFEX field length. The queue has been sorted by IAFEX 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.

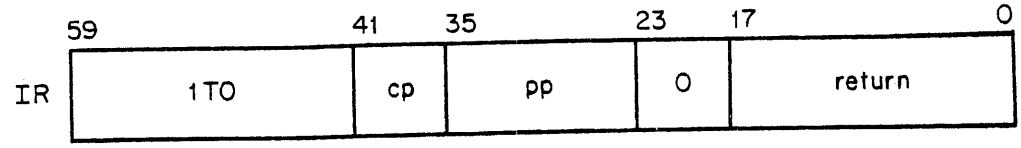
Routine 1TO is also called by 1RO to handle the first buffer of data on a rollout file. This data is passed to 1TO in a PP buffer. Routine 1TO dumps the PP buffer into pots and makes a VASO request to IAFEX for that terminal.

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



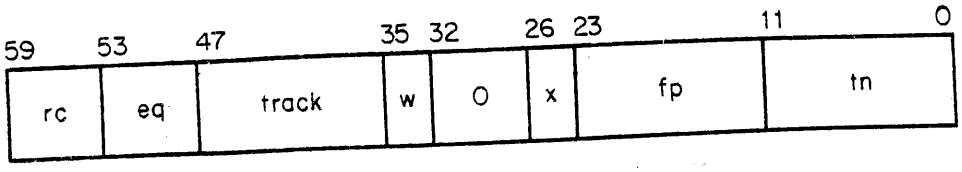
cp	IAFEX control point number
tn	Terminal number

The input register when called by IAFEX is as follows.



cp	IAFEX 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 37-16) 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)

1TO Initialization - PRS

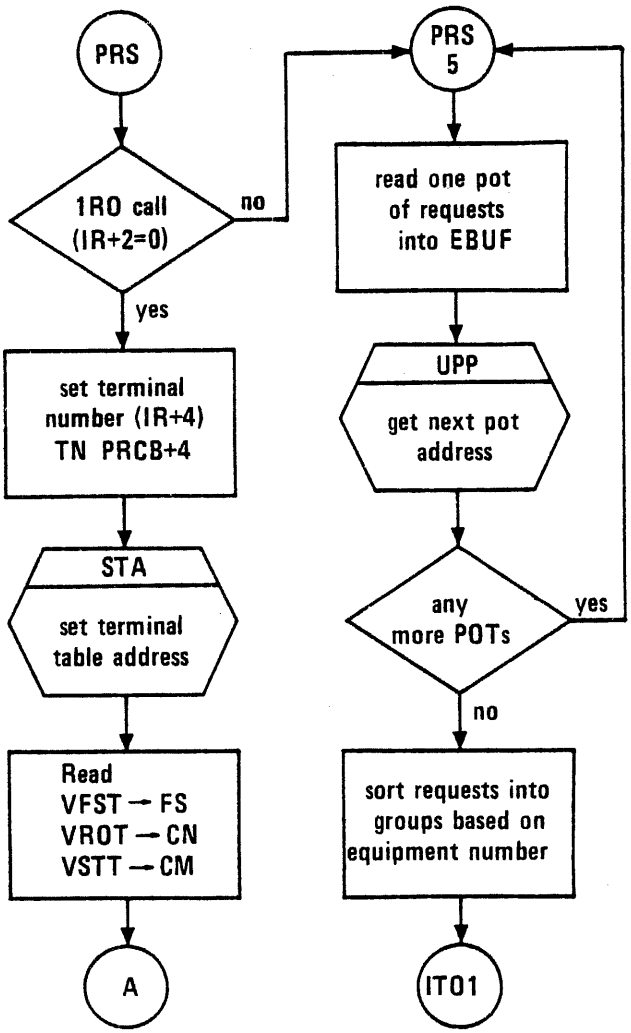


Figure 37-16. 1TO I/O Routine

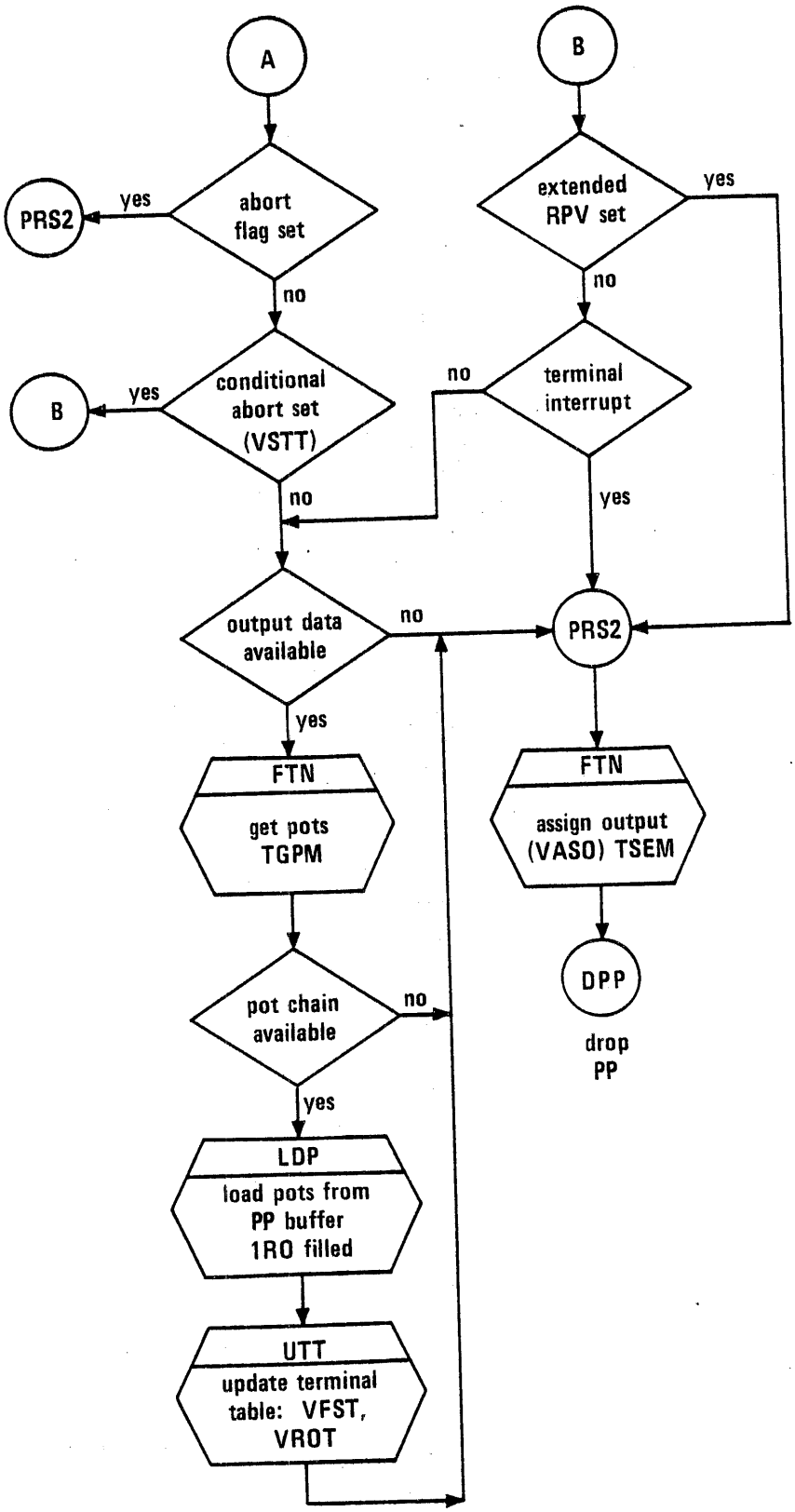


Figure 37-16. 1TO I/O Routine (Continued)

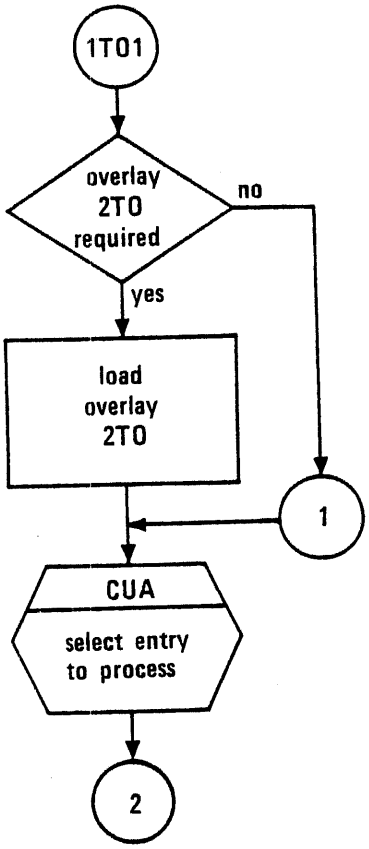


Figure 37-16. 1TO I/O Routine (Continued)

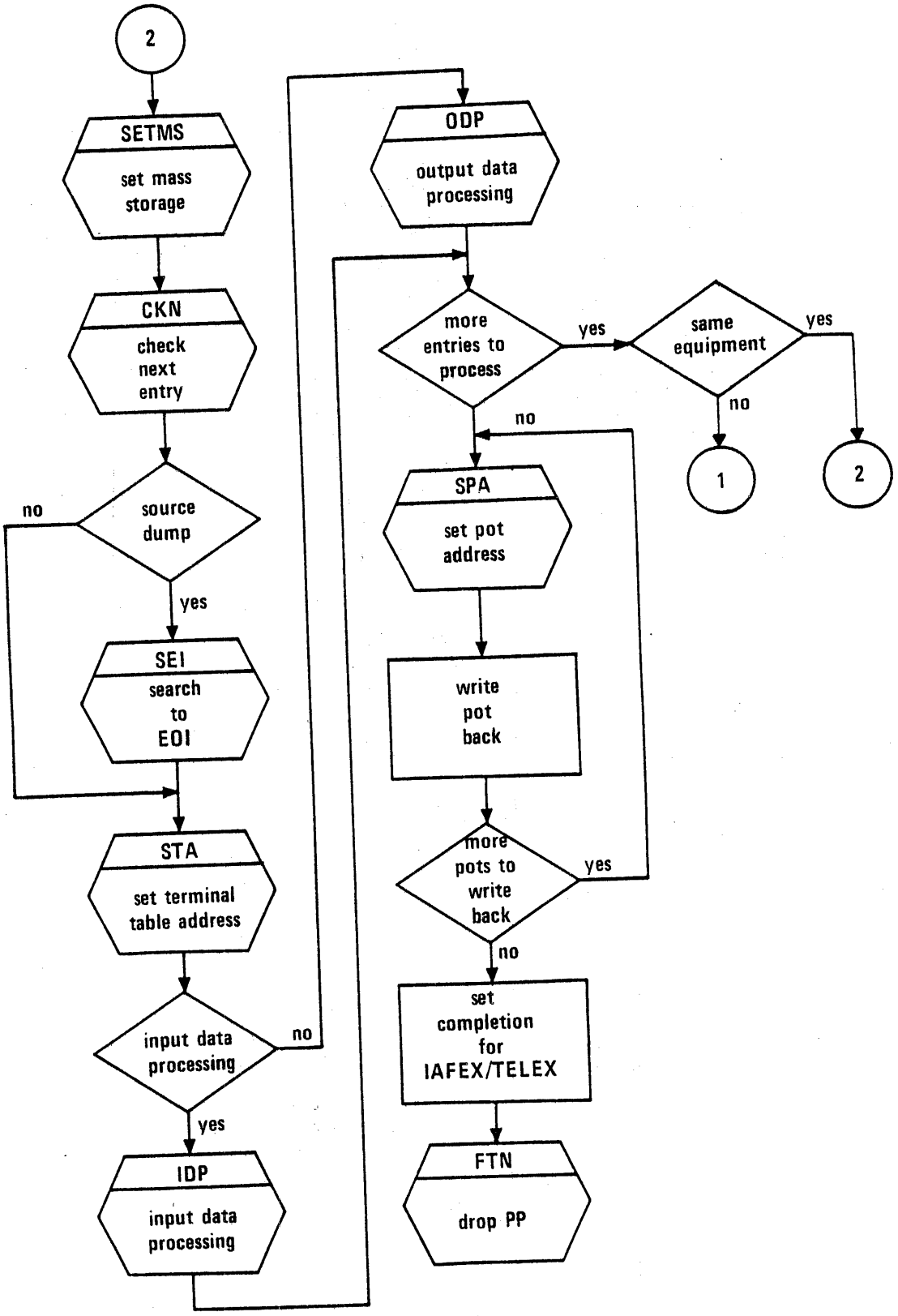


Figure 37-16. 1TO I/O Routine (Continued)

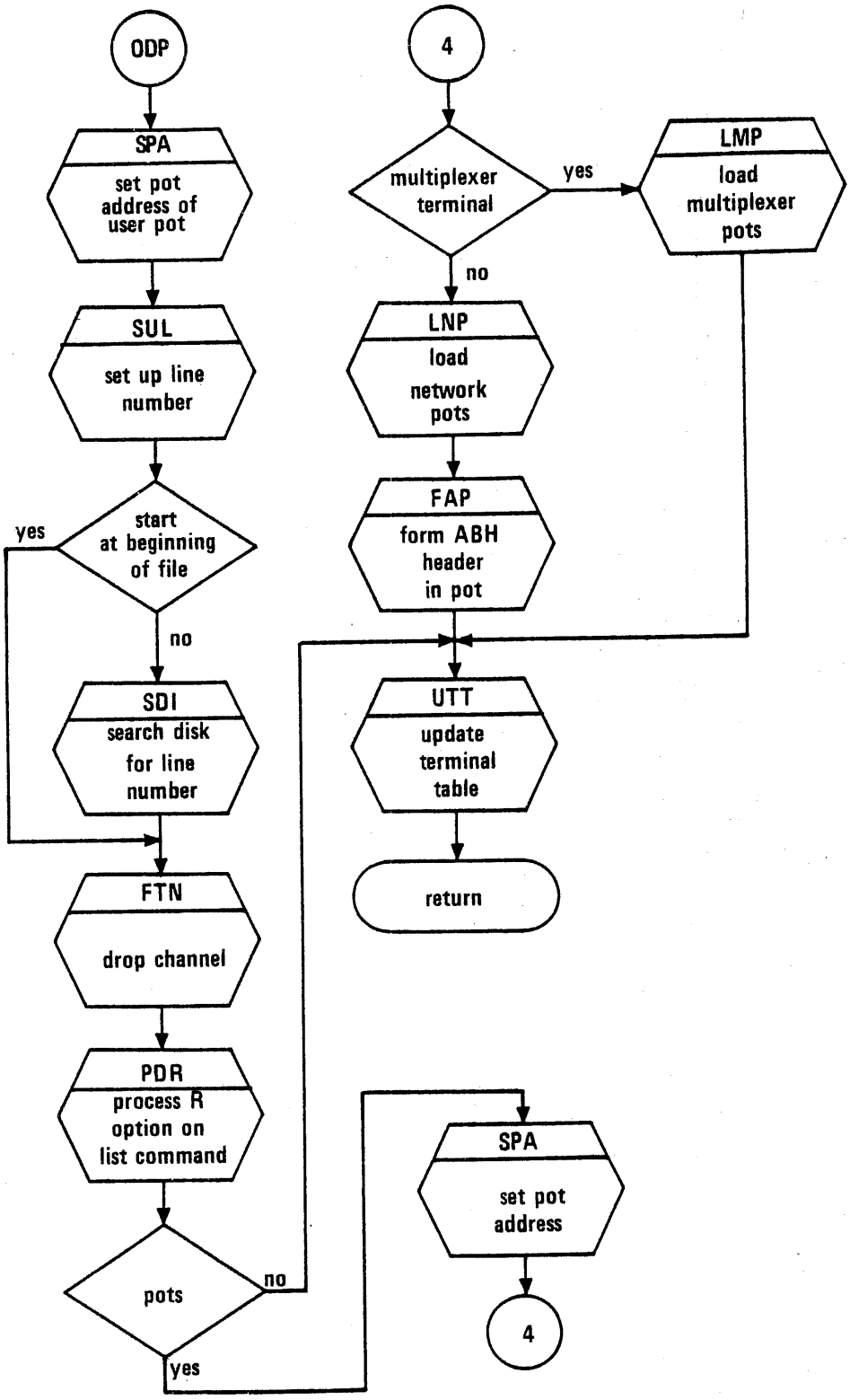


Figure 37-16. 1TO I/O Routine (Continued)

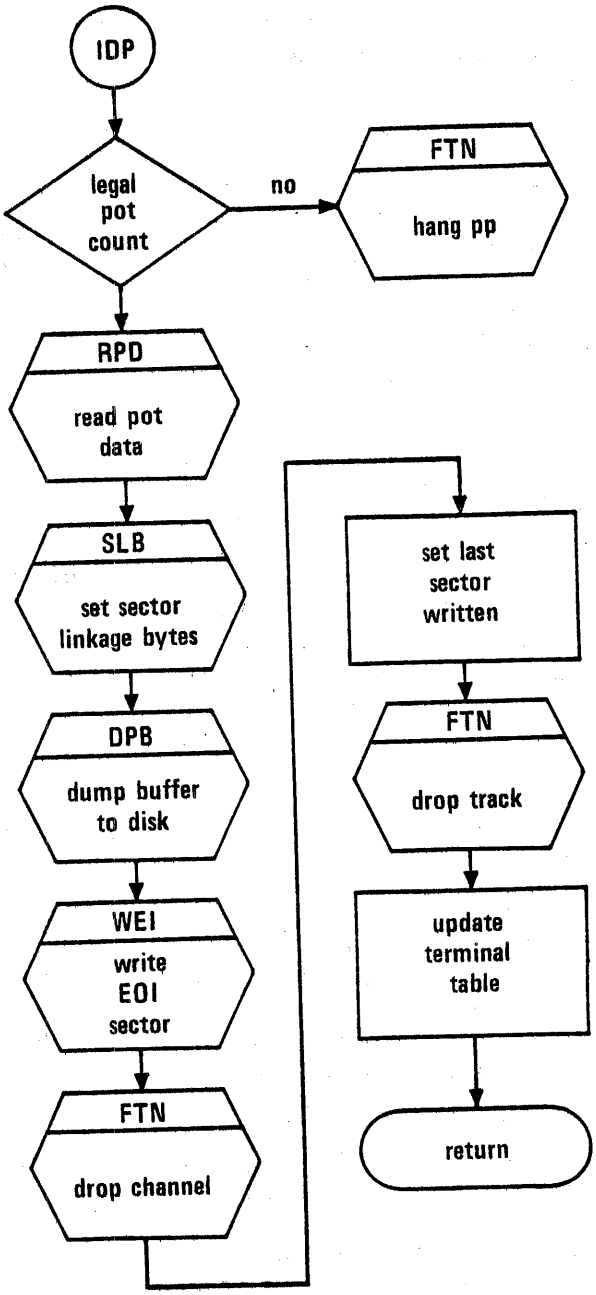


Figure 37-16. 1TO I/O Routine (Continued)

ADDITIONAL CONSIDERATIONS

The NETWORK or SIMFILE file specifies the number of network terminals IAFEX will initialize its tables to hold.

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

SALVARE - IAFEX 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 IAFEX. ISF assures that the file length does not exceed one logical track on the residence device.

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

During operation in IAFEX1, 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 VNTO minutes old. This call is made about evey 3 minutes.

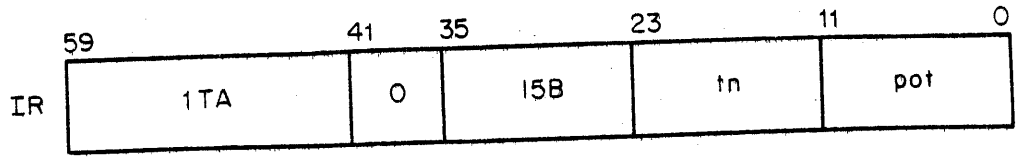
Routine 1TA is a combination of functions to perform for IAFEX. The important functions associated with the SALVARE are as follows.

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

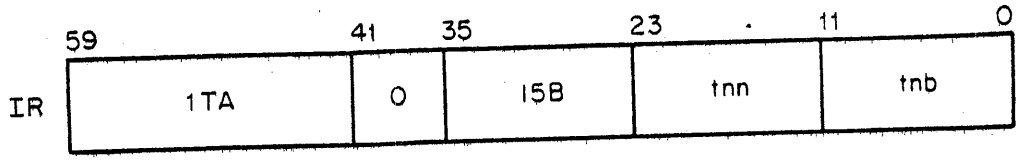
Since the SALVARE file is checked about every 3 minutes and entries more than VNTO minutes old are eliminated, then:

- A user that wishes to be recovered after losing contact must attempt recovery within VNTO 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.

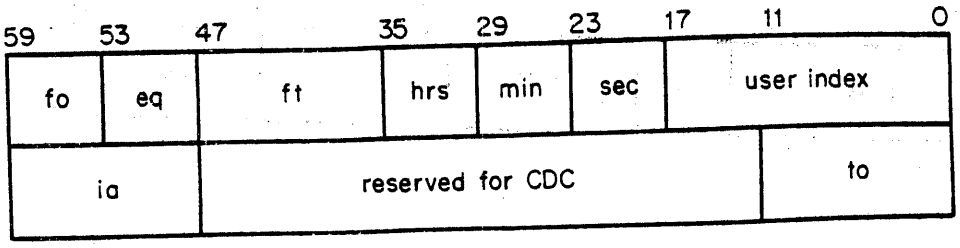


tnn	Terminal number now
tnb	Terminal number before

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 VNTO 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

COMMENT SHEET

MANUAL TITLE: CDC NOS Version 1 Internal Maintenance Specification, Volume 3

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