HP 3000 Computer Systems

INTERPROCESS COMMUNICATION

HEWLETT PACKARD
19447 PRUNERIDGE AVENUE, CUPERTINO, CA 95014

E1283

CONTENTS

SECTION 3:

MESSAGE FILES AND THE FILE SYSTEM 3-1

PREFACE

This book is a tutorial of Interprocess Communication internals and externals. It encompasses the information found in the MPE Intrinsics Manual, MPE File System Reference Manual and IPC code and reference material. The externals part of this document is meant for Hewlett Packard System Engineers or customers with an MPE and SPL background. IPC internals assumes the reader has an internal MPE and internal File System background.

Section 1 is a brief introduction to IPC, how it has evolved, where we use an IPC application. Section 2 is external IPC; how do we program IPC applications. There are two program examples at the end of this unit. One of these illustrates using soft interrupts.

The remainder of this manual is devoted to IPC internal data structures and their dynamics. Section 3 explains how IPC interacts with the MPE File System and Section 4 describes the queueing mechanisms for IPC and soft interrupts. Soft interrupt operation and soft interrupt procedures at the end of section 4 will help you to assimilate the material into a working model. For debugging information use the appendices A, B, C, and G. All of the material in this document is based on code written in MPE IV.

Larry Zeitman
MPE Technical Support
October 24, 1983

SECTION I
INTRODUCTION

Interprocess communication (IPC) is a facility of the file system which permits multiple user processes to communicate with one another in an easy and efficient manner. It is the nonpriviledged user's access to nowait I/O.[1] IPC uses message files to hold information between user processes. These message files act as first-in-first-out queues of records, with entries made by FWRITES and deletions made by FREADS.

Interprocess Communication consists of two parts. Basic IPC implements the functions as described above, and Soft Interrupts is an addition to basic IPC . The Soft Interrupt facility allows a process to trap to its own coded interrupt handler upon completing I/O.

Classically opeating systems are interrupt driven, meaning that scheduling decisions, I/O, CPU or periperal errors are all detected by the cpu by means of "interrupts". We talk of hard interrupts when either peripheral devices or hardware interrupt processing. Soft interrupts are then interrupts generated by software events. In either case the cpu is interrupted from processing and either microcode or software will cause the operating system to switch execution to an interrupt procedure. We refer to this sequence of events as a trap (also you may see trap referring to the interrupt code that executes). Soft interrupts within MPE means that when I/O completes the cpu will be interrupted, scheduling information for the process that initiated the I/O will be updated and the next time the process runs it will trap to a predetermined interrupt procedure. IPC with and without the soft interrupt feature will be explained continually throughtout this manual.

History

Prior to IPC various other system features (still available today) were used to implement process to process communication. These system features include:

File System

- o Requires users to coordinate access, using RINs or FLOCK & FUNLOCK.
- o No mechanism to queue read or write requests to a file.
- o No mechanism to implement soft interrupts.

Extra Data Segments

- o Processes must create and identify the DST to be used.
- o The DST is private to a session.
- o The user has to implement a locking scheme using RINs.
- o No mechanism to implement soft interrupts.

[1] See no-wait I/O in the MPE Intrinsics Manual.

Mail

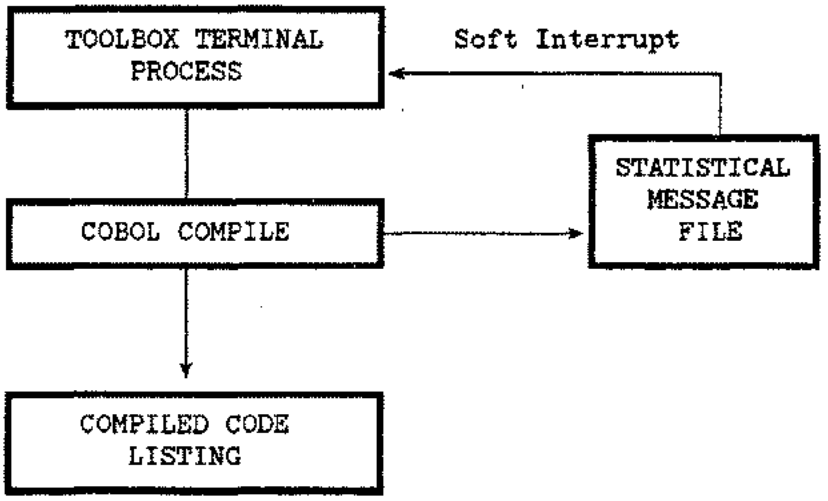
- Transfers greater than one word require the creation and deletion of an extra data segment for each transfer.
- Communication outside of the process tree is not supported.
- Mail does not have the ability to queue messages.
- Mail does not support soft interrupts.

IPC features include

- The ability to send and receive variable-length messages or records quickly and easily.
- The ability to support communication among multiple senders and receivers.
- The ability to communicate with process trees.
- The ability to queue reader and writer communication.
- The ability to perform (non-priviledged) user nowait send and receive requests.
- Support of a trap facility when a send or receive completes (software interrupts).
- Support of timeouts (timed limits to I/O inquiries).
- The ability to coordinate user communication without the need for locking and unlocking overhead.

Where Should IPC Be Used

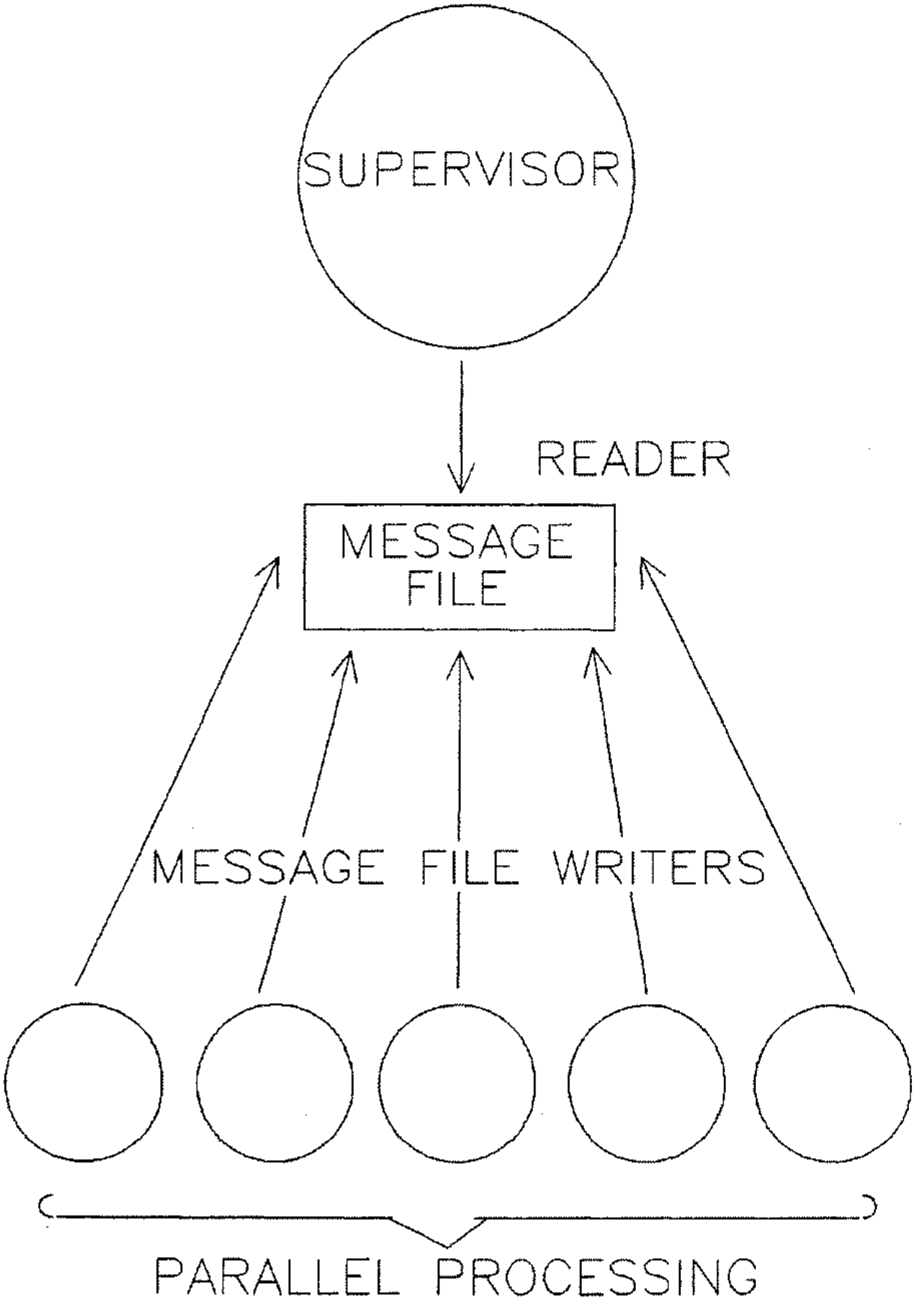
Part of IPC's power is its ability to implement parallel processing. As an example **TOOLBOX** using IPC.



The TOOLBOX terminal process edits COBOL source code. Then, a compile of that source code is initiated. Before the compile is started, the terminal process initiates a soft interrupt read to a statistical message file. The COBOL compile will actually occur in the background (as a son process). Once the compile has started, the terminal process is available to perform other tasks (e.g. write documentation, JCL). When the background compile is complete, the compiled source code is written to a special list file and the compile errors, warnings, and statistics are written to the statistical message file. Once the first record is written to the statistical message file, the pending soft interrupt read is satisfied, which interrupts the terminal process's activity to display the results of the COBOL compile.

Another useful implementation of Soft Interrupts would be in a supervisor/slave design where one/many slave processes are being managed by a supervisor process that performs other activities in parallel. The supervisor may request the status on the amount of transactions processed, whether the process is busy, or notification that the process needs to be terminated (e.g. this supervisor process could be servicing a terminal). Using IPC intrinsics the supervisor will post a read to a message file and wait for messages from slave processes. As messages are sent the supervisor can examine each one with an interrupt handler.

(See the figure on the following page)


SUPERVISOR
READER
MESSAGE FILE
MESSAGE FILE WRITERS
PARALLEL PROCESSING

EXTERNAL IPC

SECTION II

IPC Features

Message Queueing

Messages are queued when a reader tries to read from an empty message file or a writer attempts to write to a full message file.

Software Interrupts

A program may designate a trap procedure for interrupts. When I/O completes and an interrupt is generated, program control will be directed to the trap procedure. After the trap routine finishes execution, control is returned to mainline code.

Writer ID's

When a writer process opens a message file, the file system assigns a unique 16-bit identification number to the writer. Each record the process writes to the message file is associated with this number. When the writer closes the file, the ID number no longer applies to the process and may be reused. The ID is accessible via FCONTROL 46. Note that only through programming would a reader be able to receive messages from a certain writer ID.

Timeouts

A reader or a writer process may limit the length of time it will wait to be serviced. By issuing an FCONTROL 4, a process may specify the maximum number of seconds it will wait for its I/O request to be satisfied.

Copy Access

When records are read from a message file, FREAD logically deletes them as it reads. In order to copy a message file without destroying it, the file must be opened with the file copy option specified in the FOPEN Aoptions , or the ***COPY*** keyword must be specified in a :FILE command. When this option is selected the message file is treated as a standard sequential file rather than as a message file, and it may be copied safely (ie, without the logical deletion of each record).

NOTE

In order to access a message file in copy mode, a process must have exclusive access to the file.

Nondestructive Read

By issuing an FCONTROL 47, a reader may avoid deleting the record it reads, and the record will remain at the head of the message queue. This feature differs from the *copy access* feature in that it is a temporary condition. The second Fread following the Fcontrol 47 will reread the record and delete it in the usual manner.

Global Multiaccess

Global Multiaccess allows processes in the same or different process trees to open the same message file. Each file will have the same record pointer but is allowed different "f" and "a" options. The global multiaccess option may be requested in the *aoptions* of the FOPEN to the file, or by using the ***GMULTI*** keyword in a :FILE command to define the file.

NOTE

Global multiaccess is unavailable to message files when they have been opened with exclusive access in copy mode.

Appending to Variable Length Files

Variable-length files may be opened with append access. It is not necessary to have fixed-length records of the maximum possible size, so space is conserved.

Fundamental Operation

Internal IPC procedures are called by the MPE file system and the file system is envoked by the file system intrinsics. Listed below is a brief description of those intrinsics that implement an IPC operation. The intrinsics supply the options to use the basic ipc facility and soft interrupts. The default is interprocess communication without traps.

FOPEN FOPEN establishes a connection to a message file. With FOPEN, a user process identifies itself as either a reader or a writer; readers access the head of the message file and writers access the tail. Incompatible parameters that are specified with FOPEN are adjusted. For example, since messages are read or written to the file one record at a time, a multirecord parameter is not allowed and the file system will correct this parameter when the file is opened. If FOPEN is used to access a new file, a new message file is created.

FREAD Fread reads one record from the head of a message file. The record is copied to the reader's TARGET area and is logically deleted from the message queue; the next record is now at the head of the file. If a process tries to read from an empty message file that writers have opened, the file system causes it to wait until a writer process enters a record to the file. If there are no writers associated with the message file, an end-of-file indication, CCG, is returned.

NOTE

If the message file is empty and there are no writers, a read will be queued if there is either an FCONTROL 45 in effect or this is the first Fread after the reader's Fopen. See "Intrinsics - A Detailed Description page 2-11.

FWRITE FWRITE appends one record to the tail of a message file. If a process tries to write to a full message file which readers are accessing, the file system causes it to wait until a reader process has read a block of records from the file. If there are no readers associated with the message file, an end-of-file indication, CCG, is returned.

NOTE

If the message file is full and there are no readers, a write will be queued if there is either an FCONTROL 45 in effect, or this is the first Fwrite after the writer's FOPEN. See page 2-11.

FCONTROL Supplies various control functions to a process that is using a message file. These control functions permit a process to take advantage of the additional features of IPC, which are discussed on pages 2-11 and 2-12.

FCLOSE Breaks a process's connection with a message file. If the process reopens the same file later, it may do so as either a reader or a writer.

FINSTATE Enables or disables soft interrupts for the process Finstate is issued some time before the process initiates a Soft Interrupt read or write.

FINEXIT Reinstates a process's soft interrupts. FINEXIT is issued at the end of an interrupt procedure. It will reenable soft interrupts that were automatically disabled on entering the interrupt procedure.

Creating Message Files

Message files can be created in several ways. When a user process opens a new file and indicates in the *Foptions* that it will be a message file, the FOPEN intrinsic creates the new message file. In order to create a message file with the :BUILD command, use the MSG keyword. For example, to build a message file named MSG1, enter:

```
:BUILD MSG1;MSG
```

A new message file can be specified as job temporary or default to permanent and may also be defined with a :FILE command. Use the MSG keyword for a new file:

```
:FILE MSG2,NEW;MSG
```

A new message file named MSG2 is indicated.

When you perform a :LISTF,2 command, message files will be identified by an "M" in the third column of the TYP field; MSG1 is identified here:

```
FILENAME  CODE  ------------LOGICAL RECORD-----------  ----SPACE----
                  SIZE  TYP        EOF      LIMIT R/B  SECTORS #X MX

MSG1              128W  VEM          0       1031   1      258  1  8
```

Occasionally, you might create a message file and specify a certain number of records for the file to contain, only to discover that the file system has allocated more records than you requested. The reason for this is that the file system is maintaining the necessary internal structure for the message file. The file system has four basic rules for establishing this structure when the message file is created:

1. Open and close records are written to the message file every time a writer process opens or closes the file; therefore, the file system adds two records to the requested number to allow for a minimum of one open and one close operation.

2. The requested number of records is rounded up to fill an even number of blocks.

3. The file system adds an extra block to the message file for the file label to occupy. (This block is transparent to you.)

4. Each extent is the same size; that is, the file system assigns the same number of blocks to each extent.

Suppose you want to create a message file named ODDSIZE:

```
:BUILD ODDSIZE; MSG; REC=,3; DISC=51,8
```

You have specified a message file with fifty-one records, three records per block, that occupies eight extents. The file system will adjust the number of records to conform to the rules for message file structure.

1. The file system adds two records to allow for one open and one close indication; the number of records goes from 51 to 53.

2. The number of records is rounded up to 54 to provide an even number of blocks. With three records per block, 54 records will fill 18 blocks.

3. An additional block is added to the file to accommodate the file label. Now the file contains 19 blocks.

4. The eight extents must all be the same size, so the number of blocks is increased from 19 to 24. Each extent now contains three blocks.

Of the 24 blocks in ODDSIZE, 23 are data blocks and one contains the file label, which is invisible to the user. With three records per block, 23 blocks contain a total of 69 data records.

NOTE

In addition to adjusting the number of blocks in a message file, the file system adds a certain amount of space to each block for "overhead". Six bytes will be added to each record, and four bytes will be added for each block.

A listf ODDSIZE,2 now looks like:

```
FILENAME  CODE  ------------LOGICAL RECORD------------  ----SPACE----
                    SIZE   TYP       EOF      LIMIT R/B   SECTORS #X MX

ODDSIZE             128W   VBM         0         69   3        12  1  8
```

Intrinsics – A Detailed Description

Read this section carefully. Intrinsics manipulating message files do not act in a standard file system fashion.

FOPEN

Foptions: (2:3) *File Type.* Determines the type of the file to create for a new file. If the file is old, this field is ignored.

000 - Ordinary file

001 - KSAM file

010 - Relative I/O file

100 - Circular file

110 - Message file

NOTE

The Default Designator FOPTION, bits 10 through 12, offers several choices for default file designators. Any value used other than 0 for "filename" will override the File Type field

(8:2) *Record format.* Message files are always internally formatted as variable-length record files and these bits will be changed by the file system. However, a message file can appear as a fixed length file to an opener. There is no difference for a writer, but a reader will have the portion of this target area which exceeds the record filled with blanks (for an ASCII file) or zeroes (for a binary file).

00 - Fixed

01 - Variable (default)

Aoptions: (3:1) *File Copy.* This feature permits a message file to be treated as a standard sequential file, so it can be copied by logical record or physical block to another file. Setting this bit on causes all the remaining file parameters to have their normal defaults.

0 - The file will be accessed in its native mode; that is, a message file will be treated as a message file.

1 - The file is to be treated as a standard, sequential file with variable-length records. This allows nondestructive reading of an old message file at either the logical record or physical block level. Only block level access is permitted if the file is opened with write access. These blocks are checked for proper message file format to prevent incorrectly formatted data from being written to the message file while it is unprotected.

NOTE

In order to access a message file in copy mode, a process must have exclusive access to the file.

(5:2) *Multiaccess mode.* This feature permits processes located in different jobs or sessions to open the same file.

00 - No multiaccess. The file system changes this value to "2" to allow global multiaccess.

01 - Only intra-job multi-access allowed; this is the same as specifying the MULTI option in a command.

10 - Inter-job multi-access allowed; this is the same as specifying the GMULTI option in a :FILE

11 - Undefined. If this is specified, the FOPEN will be rejected with an error code of 40: ACCESS VIOLATION.

(7:1) *Inhibit buffering.* For message files, the file system turns this bit off. Readers may open a message file with NOBUF if they are in copy mode; this determines whether they will be accessing the file record by record or block by block. Writers must open message files with NOBUF if they are in copy mode; they will access the file block by block.

0 - read by logical record

1 - read by physical block

(8:2) *Exclusive.* The values for this field are the same for any disc file, but they have different meanings for the readers and writers of a message file.

BIT SETTING	USER VALUE	MEANING
01	EXCLUSIVE	One reader, one writer
10	SEMI	One reader, multiple writers
11	SHARE	Multiple readers and writers
00	Default	One reader, one writer (converts to a 01)

NOTE

Multiple accessors to one message file need not be in the same process tree

(11:1) *Multirecord.* For message files, the file system sets this bit to 0.

(12:4) *Access type.* These bits specify whether the user will be a reader or a writer process.

0000 - READ access only. The FWRITE intrinsic cannot reference this file. This access type requires both read and write access capability to the file. In the case where the writer is waiting for room to write to the message file a read to file will free the writer's wait by giving free space back to the file. The reader process will also serve the writer by performing the write itself, thereby necessitating read and write access. A process that has opened a file with this access type is a "reader."

0001 - WRITE access only. If this is the first accessor to the file and the process has write access capability, then the file's contents are purged. If this is not the first accessor to the file, the file system sets this access type to APPEND. The FREAD intrinsic cannot reference this file. A process that has opened a file with this access type is a "writer."

0010 - WRITE SAVE access. The file system sets this to APPEND access.

0011 - APPEND access only. The FREAD intrinsic cannot reference this file. This access type requires append capability to the file. A process that has opened a file with this access type is a "writer."

Device: This field is relevant only if this is a new file. The DEVICE field must either be omitted or specify a disc; specification of any device other than a disc opens the device. When this occurs, the file is no longer a message file.

Numbuffers: (0:11) Ignored.

(11:5) Value between 2 and 31; default is 2. This parameter must not exceed the physical record capacity of the file.

Filesize: The number of records is rounded up to completely fill the last block and to make the last extent the same size as the other extents. Two additional records are included for the open and close records (see page 2-5 to reference the rules the file system uses to allocate the number of records in a file).**

**See the MPE Intrinsic Manual - Accessing and Altering Files.

FCONTROL

A few controlcodes deal specifically with IPC. Those not mentioned here are either invalid or not used with IPC applications.

CONTROLCODE	PARAM	DESCRIPTION
4	integer	Set timeout interval. This applies to both FREADs and FWRITEs. The timeout will be armed at the beginning of the I/O request and cleared when the I/O completes. PARAM specifies the length of the timeout in seconds. A value of zero disables time-outs on the file. This function will remain enabled until explicitly canceled.
6	-	Write End-Of-File. Used only to verify the state of the file by writing out the file label and buffer area to disc; this ensures that the message file can survive system crashes. No actual EOF is written.
43	-	Abort NOWAIT I/O. A CCG condition code is returned if an outstanding I/O operation has completed. An IOWAIT must be issued to finish the request.
45	TRUE	Enable extended wait. Permits a reader to wait on an empty file that is not currently opened by any writer, or a writer to wait on a full file that has no reader. This FCONTROL will remain in effect until FCONTROL 45 is issued with a PARAM value of FALSE.
	FALSE	Disable extended wait. Specifies that when an FREAD encounters an empty file that has no writer, or an FWRITE encounters a full file that has no reader, it will return an end-of-file condition. (Default.)
46	TRUE	Enable reading the writer's ID.
	FALSE	Disable reading the writer's ID. Only data is read to the reader's target area (default).

CONTROLCODE	PARAM	DESCRIPTION
47	TRUE	Nondestructive read. The next FREAD by this reader will not delete the record. Subsequent FREADs will be unaffected and will logically delete records as they are read.
	FALSE	The next FREAD by this reader will delete the record (default).
48	PARM	Arm/disarm soft interrupts. Param contains the external-type label (plabel) of your interrupt procedure. In SPL it is passed as a parameter by placing an "at" sign (@) before the procedure name. If the value of PARAM is 0, the interrupt mechanism is disabled for this file.

FCHECK

FCHECK will return: **151 CURRENT RECORD WAS LAST RECORD WRITTEN BEFORE SYSTEM CRASHED.**

when a record is read following system startup and the "crash bit" has been turned on the record's header. Refer to the information on message files and their header records page xx (3-4)

FGETINFO

The value returned in RECSIZE will indicate the user's data record size, and the value returned in EOF will indicate the number of data records, unless an FCONTROL 46 is in effect. Besides data records in a message file there are also two word of header information for each data record written and open and close records for each writer that opens and closes the file. An FCONTROL 46 will return a value in RECSIZE the size of the user's data records, including the two word data record header. The number of records returned in EOF will include open, close and and data records. (See Message Files and header records page 3-2 for further reference.)

The value returned in BLKSIZE reflects the actual blocksize of the file. When the file is created, the blocksize is computed by the following algorithm:

BLOCKSIZE:=((RECORDSIZE+3)*BLOCKING FACTOR)+2

where RECORDSIZE and BLOCKSIZE are in words. For example, with a recordsize of 100 words and a blocking factor of 10, the blocksize would be 1032 words.

FFILEINFO

Three values for ITEMVALUE are specifically for use with IPC:

Item #	Type	Description
34	integer	The current number of writers.
35	integer	The current number of readers.
49	logical	Plabel of the user's soft interrupt procedure. A value of zero implies that soft interrupts are not being used.

FINSTATE

A True/False value turns on or off soft interrupts for the process.

FINEXIT

Once the interrupt handler begins execution, file system interrupts are turned off. FINEXIT is to be placed at the end of the interrupt routine to reinstate soft interrupts (logical value true).

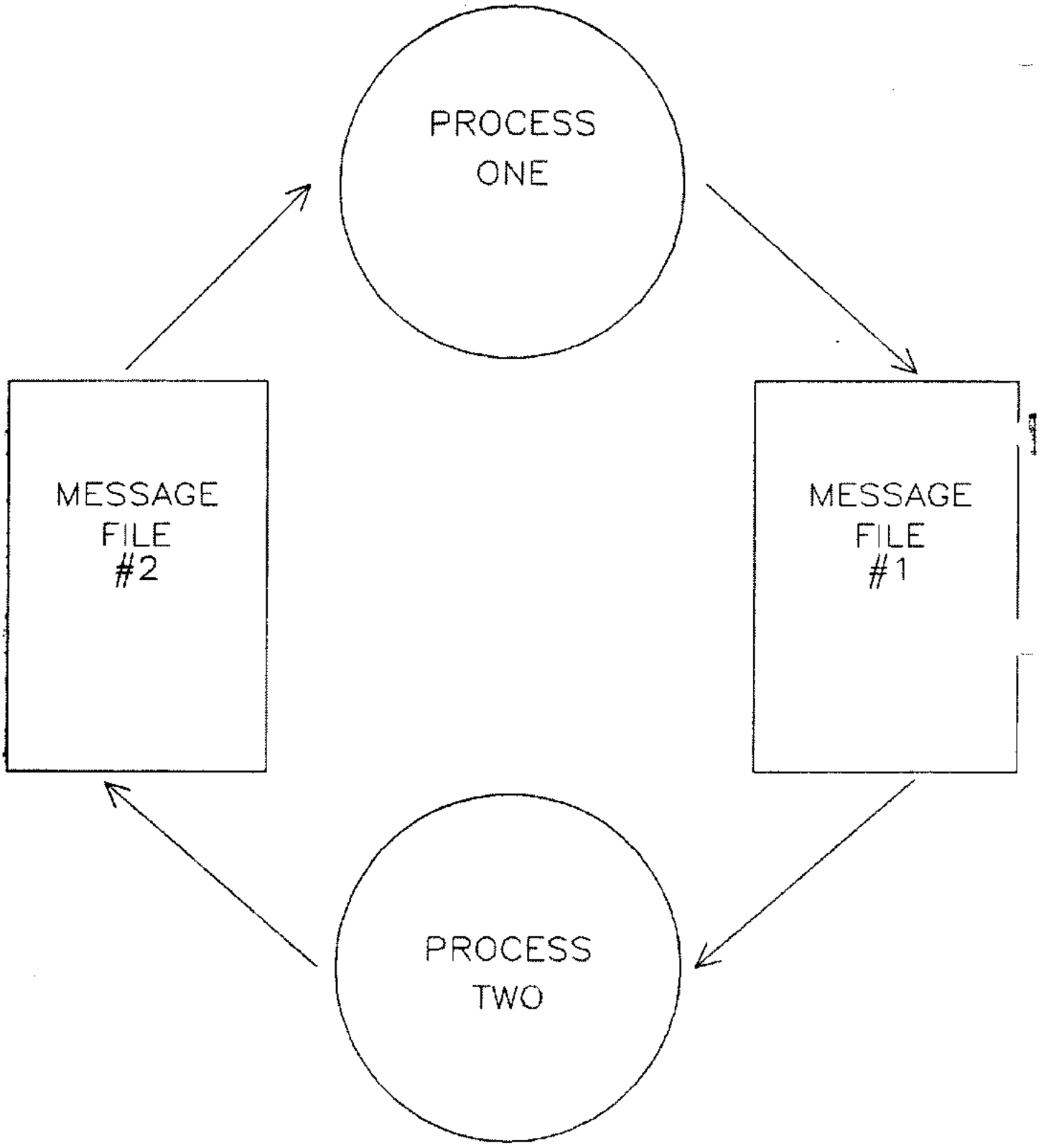
Intrinsics Not Allowed

Certain intrinsics are not allowed for message files. (The FSETMODE intrinsic is permitted, but ignored.) Disallowed intrinsics are listed below:

FPOINT	FREADDIR	FDELETE
FREADSEEK	FSPACE	
FUPDATE	FWRITEDIR	

Examples Using Message Files (Basic IPC)

In this example of IPC there are two processes. The first process reads data from a data file and passes the data to a second process via MSGFIL1 (refer to picture below). After process one finishes, it hangs a read on MSGFIL2 (which is empty) waiting for acknowledgement from process two that it has processed all the data. When process one receives the message from process 2 by completing a read, it terminates.


PROCESS
ONE
MESSAGE
FILE
#2
MESSAGE
FILE
#1
PROCESS
TWO

PROCESS ONE

```
$CONTROL USLINIT

<< Purpose: >>
<< Read data from a data file and send to another process. >>

BEGIN
 LOGICAL EOF := FALSE;
 INTEGER DATA'FILE, PIN, IN'FILE, OUT'FILE,ERR;
 INTEGER LEN:=80
 LOGICAL TRUE:=1;
 BYTE ARRAY IN'FILE'NAME (0:8) := "MSGFILE2 ";
 BYTE ARRAY OUT'FILE'NAME (0:8) := "MSGFILE1 ";
 BYTE ARRAY DATA'FILE'NAME (0:8) := "DATAFILE";
 ARRAY MESSAGE (0:39);

 INTRINSIC  CREATEPROCESS, FCLOSE, FOPEN, FREAD, FWRITE,
            QUITPROG, FCONTROL;


  << Create entries for the message files in the directory.       >>
  << Note that IN'FILE'NAME ("MSGFILE2") is opened with FOPTIONs >>
  << %30004:  this indicates a new ASCII message file.           >>


 IN'FILE := FOPEN (IN'FILE'NAME, %30004);
 IF < THEN QUITPROG (1);
 FCLOSE (IN'FILE, 2, 0);      << Save file as session temporary. >>
 IF < THEN QUITPROG (2);

  << Note that OUT'FILE'NAME ("MSGFILE1") is also opened with FOPTIONs %30004>>

 OUT'FILE := FOPEN (OUT'FILE'NAME, %30004);
 IF < THEN QUITPROG (3);
 FCLOSE (OUT'FILE, 2, 0);     << Save file as session temporary. >>
 IF < THEN QUITPROG (4);


  <<Opening the message file we will write to.                    >>
  <<Note that IN'FILE'NAME ("MSGFILE1") is opened with FOPTIONs   >>
  <<%106 and AOPTIONs %1100: %106 indicates an old temporary     >>
  << ASCII file and %1100 indicates a reader process with        >>
  << exclusive access and multiaccess capability.  MSGFILE1      >>
  << has already been designated as a message file.  Since       >>
  << only one reader and one writer process will be accessing    >>
  << the message file, exclusive access mode is specified.       >>
```

```
IN'FILE := FOPEN (IN'FILE'NAME, %106, %1100);
IF < THEN QUITPROG (7);
FCONTROL (IN'FILE,45,TRUE);


 << Open message file that we will read from .                          >>
 << Note that OUT'FILE'NAME ("MSGFILE2") is opened with FOPTIONs        >>
 << %106 and AOPTIONs %1101:  %106 indicates an old temporary           >>
 << ASCII file and %1101 indicates a writer process with                >>
 << exclusive access and multiaccess capability.  MSGFILE2 has          >>
 << already been designated as a message file.  Since only              >>
 << one reader and one writer process will be accessing the             >>
 << message file, exclusive access mode is specified.                   >>


OUT'FILE := FOPEN (OUT'FILE'NAME, %106, %1101);
IF < THEN QUITPROG (8);
FCONTROL (OUT'FILE,45,TRUE);


 << Open data input file.                                               >>
 << Note that DATA'FILE'NAME ("DATA") is opened with FOPTIONs %3        >>
 << and AOPTIONs 0:  %3 indicates an old permanent or temporary         >>
 << file and 0 indicates read only access.  The file system             >>
 << will change the FOPTIONs to specify an ASCII file.                  >>
DATA'FILE := FOPEN (DATA'FILE'NAME, %3, 0);
IF <> THEN QUITPROG (9);
WHILE NOT EOF DO BEGIN
 LEN := FREAD (DATA'FILE, MESSAGE, -80);
 IF < THEN QUITPROG (10);
 IF > THEN BEGIN
          EOF:=TRUE;
          MESSAGE:= "FOUND EOF IN DATA FILE";
          WRITE (OUT'FILE,MESSAGE,-LEN,0);
         END<<IF>>;
 ELSE  BEGIN
  FWRITE (OUT'FILE, MESSAGE, -LEN, 0);
  IF <> THEN QUITPROG (11);
 END;
END << WHILE >>;

FCLOSE (OUT'FILE, 4, 0);        << No more data to send:  EOF >>
IF < THEN QUITPROG (12);

 FREAD (IN'FILE, MESSAGE, 1);      << Wait for writer to          >>
 IF <> THEN QUITPROG (13);         << to process this information >>
```

```
  << Because we have issued an FCONTROL 45 against MSGFIL2   >>
  << process one will terminate normally. If the second      >>
  << process ;however, closes without communicating to       >>
  << process one then we will end in error                   >>


  FCLOSE (IN'FILE, 4, 0);
  IF < THEN QUITPROG (14);
END.
```

PROCESS TWO

```
$CONTROL USLINIT

<< Purpose: >>
<< Receive data from other process and print it. >>

BEGIN
 LOGICAL EOF := FALSE, TRUE:=TRUE;
 INTEGER LEN, IN'FILE, OUT'FILE;
 BYTE ARRAY IN'FILE'NAME (0:8) := "MSGFILE1 ";
 BYTE ARRAY OUT'FILE'NAME (0:8) := "MSGFILE2 ";
 ARRAY MESSAGE (0:39);
 INTRINSIC FCLOSE, FOPEN, FREAD, FWRITE, QUITPROG, PRINT FCONTROL;

  << Open message file from other process.                          >>
  << Note that IN'FILE'NAME ("MSGFILE1") is opened with FOPTIONs    >>
  << %106 and AOPTIONs %1100:  %106 indicates an old temporary      >>
  << ASCII file and %1100 indicates a reader process with           >>
  << exclusive access and multiaccess capability.  MSGFILE1         >>
  << has already been designated as a message file.  Since          >>
  << only one reader and one writer process will be accessing       >>
  << the message file, exclusive access mode is specified.          >>

 IN'FILE := FOPEN (IN'FILE'NAME, %106, %1100);
 IF < THEN QUITPROG (13);
 FCONTROL (IN'FILE,%45,TRUE);

  << Open message file to other process.                            >>
  << Note that OUT'FILE'NAME ("MSGFILE2") is opened with FOPTIONs   >>
  << %106 and AOPTIONs %1101:  %106 indicates an old temporary      >>
  << ASCII file and %1101 indicates a writer process with           >>
  << exclusive access and multiaccess capability.  MSGFILE2         >>
  << has already been designated as a message file.  Since only     >>
  << one reader and one writer process will be accessing the        >>
  << message file, exclusive access mode is specified.              >>

 OUT'FILE := FOPEN (OUT'FILE'NAME, %106, %1101);
 IF < THEN QUITPROG (14);
  MESSAGE:="ERROR"
 WHILE NOT EOF DO BEGIN
  LEN := FREAD (IN'FILE, MESSAGE, -80);
  IF < THEN QUITPROG (15);
```

```
 IF > THEN EOF:= TRUE
      ELSE PRINT (MESSAGE, -LEN, 0);
END  << WHILE >>;

 << Because an FCONTROL 45 has not been issued a CCL is generated >>
 << when process two tries to read past the EOF.                   >>

 <<Now signal other process we are done.>>

Message:="INFORMATION RECEIVED"
FWRITE(OUT'FILE,MESSAGE,-LEN,0)
FCLOSE (OUT'FILE, 4, 0);
IF < THEN QUITPROG (16);
FCLOSE (IN'FILE, 4, 0);
END. <<Procecss Two>>
```

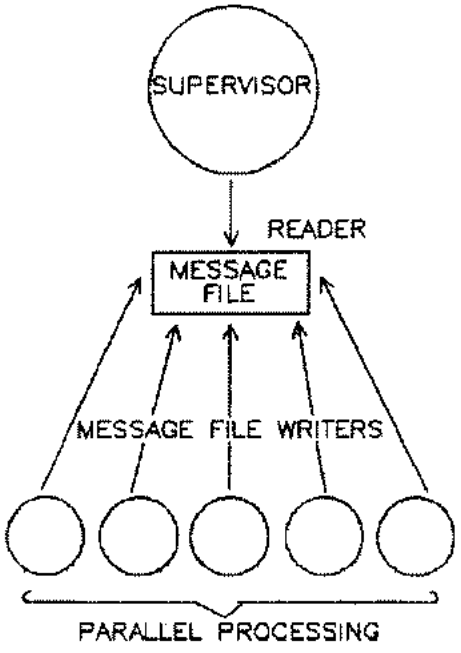
Software Interrupts

Software Interrupts is an enhancement to BASIC IPC. Soft Interrupts allow the same extended wait conditions when reading or writing to a message file as BASIC IPC. When the I/O completes your program is interrupted and executes a predetermined interrupt procedure. The interrupt handler will complete the read or write and return your program to the line that was executing when the interrupt occurred.

NOTE

Soft Interrupts cannot be implemented in Cobol.

Typically a soft interrupt application is one in which a master process serves as a controller to parallel processes. One message file is used for "system" communication where the master is a reader and the slave processes are writers to the message file. The master process will usually wait for communication from other processes. It waits by hanging a read on the empty message file. It will be "soft interrupted" by one of the slave processes writing to the message file. The master process parses the command it received in the message and processes the command with a subroutine. The master process replies if necessary through the individual message file of one of its slaves.



OPERATION

Initially, software interrupts are disabled for your program. To enable soft interrupts, use the FINTSTATE intrinsic with a value of TRUE, as follows:

```
VALUE:=FINTSTATE(TRUE);
```

The FINTSTATE intrinsic with a value of FALSE will inhibit soft interrupts and VALUE is the old state of FINTSTATE. MPE will inhibit soft interrupts just before entering an interrupt procedure. This is done to prevent unwanted nesting of the interrupt procedures. Using the FINTEXIT intrinsic also with a value of true at the end of the interrupt procedure, it will re-enable interrupts just before it exits.

```
PROCEDURE INTERRUPTPROC (FILENUM);
 VALUE FILENUM;
 INTEGER FILENUM;
  BEGIN
    .
    .
    .
  FINTEXIT (TRUE);
  END;
```

NOTE

Software interrupts are automatically inhibited just before a CONTROL-Y trap procedure. The trap procedure may elect to allow soft interrupts by calling the FINTSTATE intrinsic. If it does not call FINTSTATE, the RESETCONTROL intrinsic will restore the process' interrupt state to its pre-CONTROL-Y value.

When you have enabled software interrupts for your program, you must also "arm" soft interrupts for a particular file by specifying the interrupt procedure's plabel in an FCONTROL 48.

FCONTROL (File Number, 48, @Procedure Name)

Calling FCONTROL 48 with a parameter of zero will disarm the software interrupt mechanism so that the file would be accessed by Basic IPC procedures.

NOTE

The FFILEINFO intrinsic may be used to return the plabel of the interrupt handler. FFILEINFO 49 will return the plabel as an integer value: if it returns a value of zero, no interrupt handler has been armed.

After the program interrupt has occurred, an IODONTWAIT must be issued against the file to complete the request. Your interrupt handling procedure should issue the IODONTWAIT intrinsic. IODONTWAIT will allow completion operations for the associated I/O or return a zero for an incomplete I/O operation. Multiple pending soft interrupts are queued in a FIFO manner. Open files with a soft interrupt pending (read or write) cannot be closed.

NOTE

The message file and the interrupt must occur on the same system. Soft interrupt use is restricted to message file and may not be used with remote files.

No more than one uncompleted FREAD or FWRITE may be outstanding from one process for one particular message file. Any additional FREAD's or FWRITE's will result in an unequal condition code being returned. An uncompleted FREAD or FWRITE request; however, may be aborted by issuing an FCONTROL 43 (abort nowait I/O).

An interrupt will not occur while you are executing within operating system code; that is, while you are processing an MPE intrinsic or procedure. Exceptions to this are the PAUSE, PAUSEX, and IOWAIT intrinsics that will allow the processing of a soft interrupt. MPE reinvokes these intrinsics when the interrupt procedure has completed. If the PAUSE intrinsic is soft interrupted two minutes into its overall time, after the soft interrupt has finished processing, it will be reinitialized to its original time. Pausex, however, was created to be used for soft interrrupts. It will retain the amount of time it has used and continue after an interrupt, the remaining unused interval. If the PAUSEX was issued for five minutes and was interrupted again two minutes after it had begun, PAUSEX would continue to pause after the interrupt procedure for the remaining three minutes.

The validity of an interrupt procedure depends on the code domain and executing mode (privileged or non-privileged) of your code. It must be compatible with the code domain and the mode (privileged or non-privileged) of your interrupt procedure. The code domains are:

```
PROG   (User program)
GSL    (Group SL)
PSL    (Public SL)
SSL    (System SL, non-MPE segments)
MPESSL (System SL, MPE segments)
```

When the code of the caller is:	The plabel:
Non-privileged in PROG, GSL, or PSL.	Must be non-privileged in PROG, GSL, or PSL.
Privileged in PROG, GSL, or PSL.	May be privileged or non-privileged in PROG, GSL, or PSL.
Privileged or non-privileged PROG,GSL,PSL	May be in any non-MPESSL segment.

Soft Interrupt Example

The following is a skeleton example of a program which is using soft interrupts. The reader process sets up the soft interrupt and then initiates a read on an empty message file. That read request will be queued. The writer process once created, writes to the message file and releases the reader. The reader continuing to execute mainline code will be interrupted and process the interrupt with procedure INTERRUPTPROC. Once INTERRUPTPROC completes, the reader will continue at the interrupted instruction in mainline code.

READER PROGRAM

```
$CONTROL
BEGIN
ARRAY BUFFER(0;79);
INTEGER TRUE:=1, PLABEL, FILENUM, TCOUNT;
INTRINSIC FINSTATE, FCONTROL, FINTEXIT, INDONTWAIT,
FOPEN, FREAD;
PROCEDURE INTERRUPTPROC(FILENUM);  <----------STEP 1
  VALUE FILENUM;
  INTEGER FILENUM;
  BEGIN
    IODONTWAIT(FILENUM,BUFFER,TCOUNT);  <----------STEP 2
    PRINT BUFFER TO DISPLAY ON THE TERMINAL;
    FREAD(FILENUM,BUFFER,TCOUNT);  <----------STEP 3
    FINTEXIT(TRUE);  <----------STEP 4
    END;

BEGIN  <<INITIALIZATION>>
  FINSTATE(TRUE);  <----------STEP 5
  FILENUM:=FOPEN(MSGFILE);  <----------STEP 6
  FCONTROL(FILENUM,48,PLABEL);  <----------STEP 7
  FREAD(FILENUM,BUFFER,TCOUNT);  <----------STEP 8
  CREATE(WRITER PROCESS);
       <<END OF INITIALIZATION>>

LOOP:  <<BEGIN MAIN LINE>>

  PERFORM OTHER PROCESSING;  <----------STEP 9

  END; <<END OF MAIN LINE>>
END.
```

INTERRUPT PROCEDURE FUNCTIONS:

STEP 1 This is the beginning of the interrupt procedure which handles soft interrupts. Once the READ has started to complete this procedure will be executed.

NOTE

Once the soft interrupt has occurred, the read has not actually transferred any data. A call to IOWAIT or IODONTWAIT is required to complete the read.

STEP 2 IODONTWAIT or IOWAIT is needed to complete the read.

STEP 3 Call FREAD again to initiate another soft interrupt read.

STEP 4 FINTEXIT will re-enable soft interrupts before exiting the interrupt procedure (possibly to process a pending soft interrupt).

MAIN LINE FUNCTIONS:

STEP 5 FINSTATE(TRUE); will enable soft interrupts to occur at the program level (only needs to be initiated once).

STEP 6 FOPEN the new file with following options:

A. Message file

B. Reader

NOTE

The first message file opener must open the new message file twice. When the message file is closed for the first time, it will be put in the directory at that time. This would be needed only when more than one process will be accessing the message file.

STEP 7 FCONTROL 48: This will enable soft interrupts for the message file specified in "FILENUM."

STEP 8 This FREAD for FILENUM will initialize the first soft interrupted read.

STEP 9 The reader process is free to perform other necessary processing.

WRITER PROGRAM

```
$CONTROL
BEGIN
ARRAY BUFFER(0:79):=" A SOFT INTERRUPT OCCURED";
INTEGER FILENUM;
INTRINSIC FOPEN,FWRITE
BEGIN
  FILENUM:=FOPEN(MSGFILE);      <-------------------STEP 1
LOOP:
  PAUSE(5 MINUTES);      <-------------------------STEP 2
  FWRITE(FILENUM,BUFFER,TCOUNT,%60);      <-------STEP 3
  GOTO LOOP;
  END;
END.
```

STEP 1 FOPEN the file with following options:

A. Message file

B. Writer

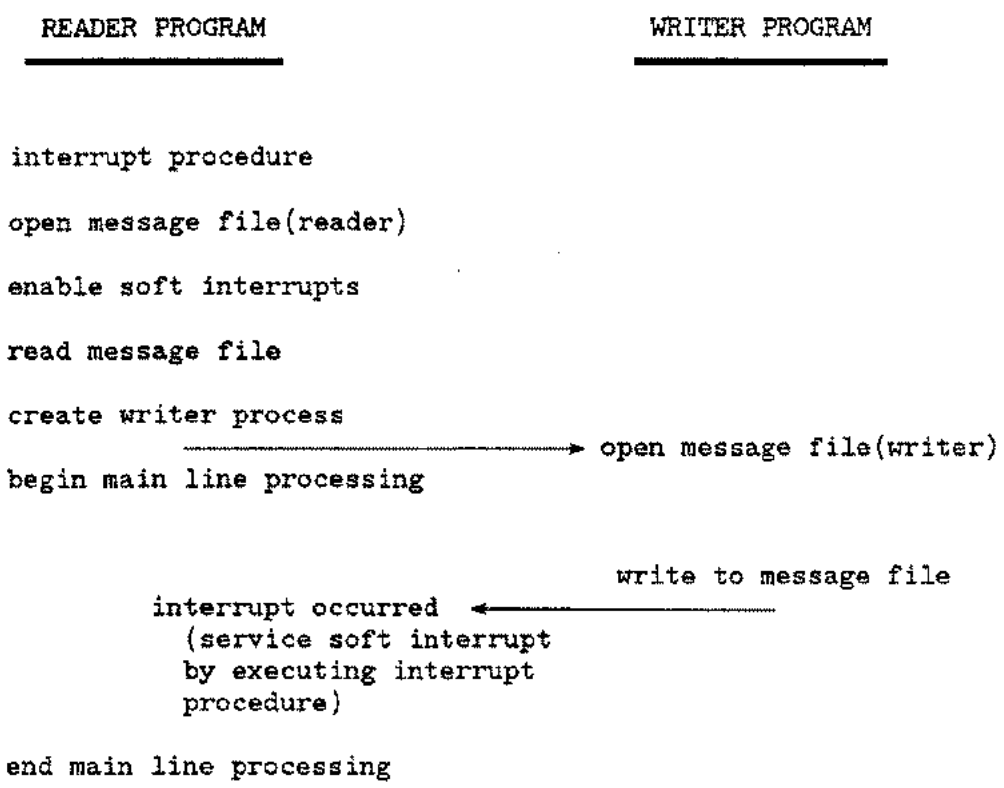
STEP 2 PAUSE for five minutes which will insure that the read posted by the reader program will be interrupted by a soft interrupt.

NOTE

The pause is not needed in an actual application design. It is used only to delay the soft interrupt from occurring the instant the soft interrupt read is posted.

STEP 3 Write to the message file causing the soft interrupt to occur on the reader process.

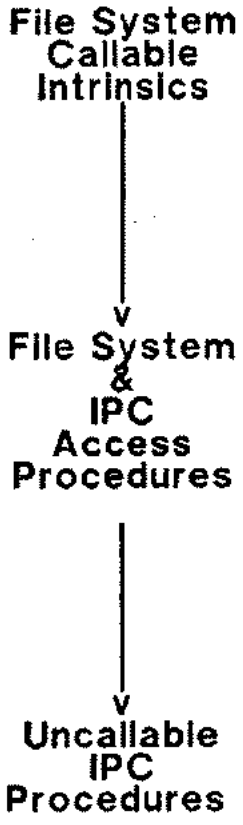
FUNCTIONAL FLOW



The READER program executes main line code until the writer program writes a message to the message file. Every five minutes the WRITER program writes a record and the READER program soft interrupts, reading the message.

MESSAGE FILES AND THE FILE SYSTEM

SECTION III



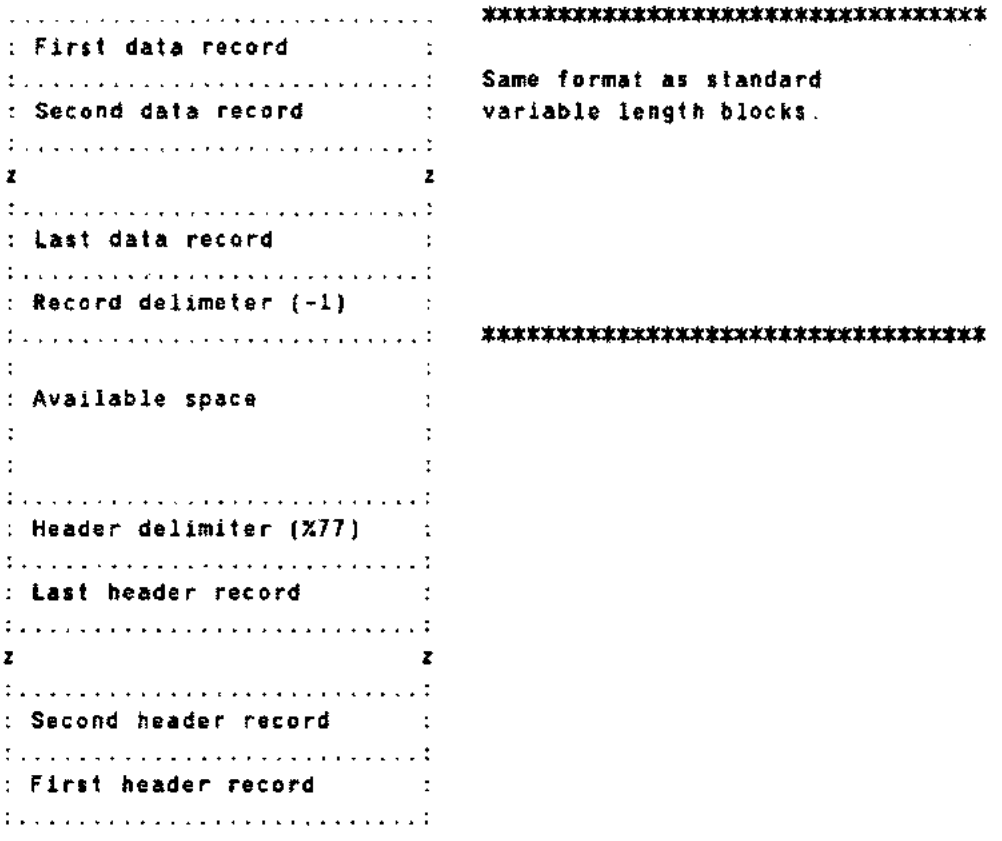
IPC PROCEDURES ARE ACCESSED THROUGH STANDARD FILE SYSTEM INTRINSICS

IPC was written sharing as much MPE File System code as possible ergo it uses many standard file system intrinsics with either added options or specially defined for maipulating message files. This section is essentially an explanation of how IPC works in conjunction with MPE's file system. IPC data structures and dynamics are added to the fundamental file system descriptions.

MESSAGE FILES

Message files since they are the fundamental building blocks of IPC. They are the communication device between processes. Message files differ from standard files, because within the file block the data records lie opposite from the header information.

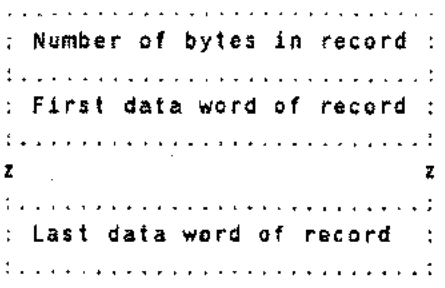
FILE BLOCK



Data is written at the top of the block while its header information is written at the bottom. A negative one delimits data records and a %77 delimits the headers.

DATA RECORDS

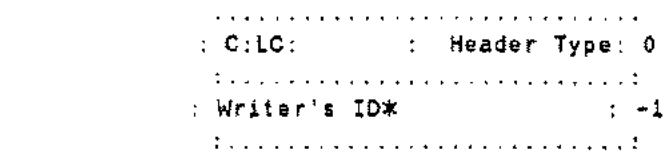
Data records in a message file are always variable length records regardless of how you may have opened the file. Each record has a one word header that prefixes the record with a byte count of the records length.



Record Format

HEADER RECORDS

The header records are built from the bottom of the block toward the top. There are three types of header records and they are written by a writer upon different events. An "Open" header record is written with the first write to the file. The "close" header record is written when an FCLOSE is issued, and a "Data" header record is written each time the writer writes to a message file. Header information is two words long. The type of header record is contained in the first word (8:15).



Header Format

```
C  (0:1)- Set  on if this was the last record written before
          the system crashed. An FCHECK issued will return:

          "Current Record  Was Last Record Written Before System Crash


LC (1:1)- Valid only for close headers.  Set to one if
          this was the last writer to close the file.


Type(8:8)- 0 data
           1 open
           2 close
```

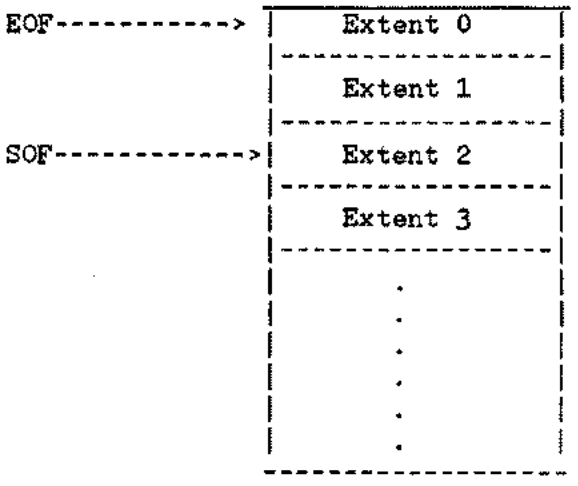
*Recall that the writer's ID may be read with an FCONTROL 46.

FILE EXTENTS

Observe the message file below and note that the Start Of File is greater than the End Of File. Message files are circular files. Data written to a message file can wrap-around the file as we use or reuse allocated extents; therefore the figure below would not be an unusual one to see.

File extents are composed of a number of blocks. Extent zero also contains space for the file label and user labels in the exact same format as a standard file. Starting with block zero, sufficient blocks are allocated to the file label/user labels to satisfy their space requirements. The start of file moves into the extent map of a File Control Block[2] as records are read; hence records are logically deleted. The EOF moves when records are written.

Extents outside of the SOF/EOF range may not exist. They are deleted at close time when there are no more writers accessing the file.

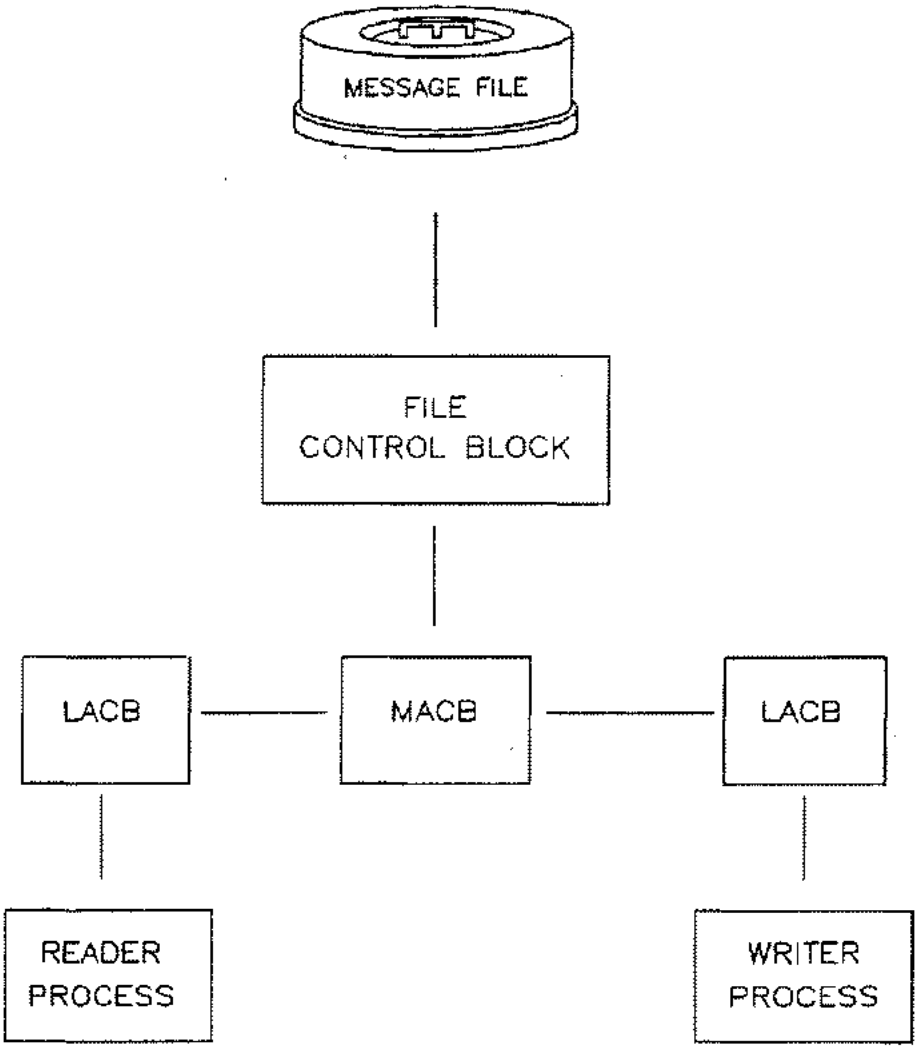


[2]See File Control Block in next section page 3-15.

CONTROL BLOCKS

File access is coordinated by control blocks. Control blocks contain information pertaining to a file. Control blocks are dynamic; they are created when the file is opened (FOPEN); they are consulted when a file is accessed (FREAD, FWRITE); and pointers to these control blocks are deleted when file access is terminated (FCLOSE). A message file has three control blocks (excluding control block extensions): a **Logical Access Control Block** and its extension, **Message Access Control Block** (corresponding to a standard file's physical access control block) and its extension, and **File Control Block.**

WHAT ARE THE RELATIONSHIPS BETWEEN FILE --- CONTROL BLOCKS --- PROCESS

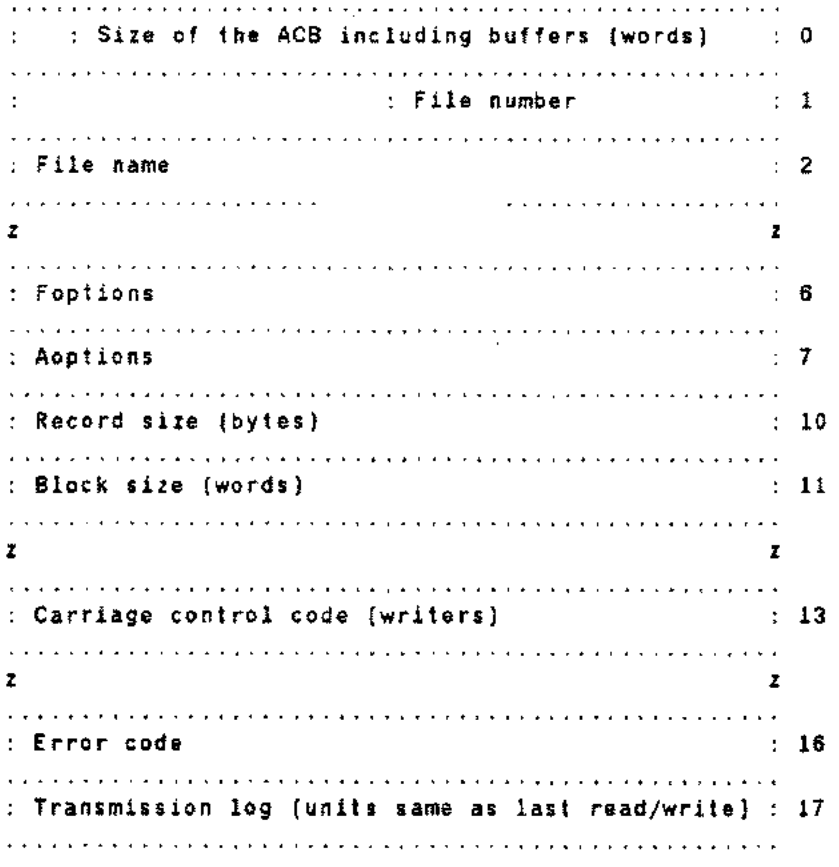


The figure above shows how we logically look at the message file and its control blocks. Message files have multi - access capability which permits the sharing of the access to a file. Each user shares the same SOF and EOF. File access information which is unique and local to each individual accessor is contained in the LACB. File access information which is common to each set of accessors is contained in the MACB. The MACB contains an FCB vector[3], a pointer to the FCB and the FCB contains the file's extent map.

[3]See vector structure page 3-21.

Logical Access Control Block

The logical access control block contains the foptions and aoptions used by a process to open the message file. It also contains the file number (returned by the FOPEN intrinsic), file name, record size and block size. Any file in our file system opened multi has an LACB. A message file's LACB is the same as a standard file's LACB.



Logical Access Control Block Extension

The LACB extension contains information specific to IPC and soft interrupts. Both the LACB and the LACB extension are created at FOPEN time. These control blocks are not shared with other processes but are unique to the process that has created them. These control blocks are found in the process's own stack in the PXFILE area[4] (unless that process is running its message file application NOCB or there is no room remaining in the area- then both of these control blocks will be found in extra data segments).

User's soft interrupt plabel	20
Number of seconds to wait on boundary condition	21
0:Ex:Nd:Vr:Bt:Cls :C : Carriage control	22
Reply Port (basic IPC port)	23
Writer ID	24
Control block index for nowait writer record buf	26
DST relative addr of nowait writer record buffer	27
No wait I/O resultant error code	30
No wait I/O resultant transmission log	31

[4]Where is the PXFILE area? See File Access later in this section.

Word	Field	Description
22		Accessor's local flags.
	(0:1)	O 1 - have not yet issued an FREAD/FWRITE against the file.
	(1:1)	ex 1 - extended wait mode.
	(2:1)	nd 1 - do not destroy the next record read.
	(3:1)	vr 1 - writer has not yet written his first record (ie., he is a virgin).
	(4:1)	bt 0 - transmission log should be expressed in words. 1 - " " " " " " bytes.
	(5:1)	cls - Not currently used (reserved for group IPC standard).
	(6:1)	C - No wait completion message is in LACB area.
	(8:8)	car ctl - carriage control character to be used for the writer's record (a value of one indicates no carriage control character).

Message Access Control Block

The MACB is a **Physical Access Control Block** [5] for message files. It is considered the "point of attachment" (from the process) to the disc file. It contains the "start of file" pointer for the reader/s and "end of file" pointer for the writer/s. The MACB is created by the first process to open the file. According to file system placement strategy the MACB and its extensions (they are contiguous) are placed in an extra data segment.

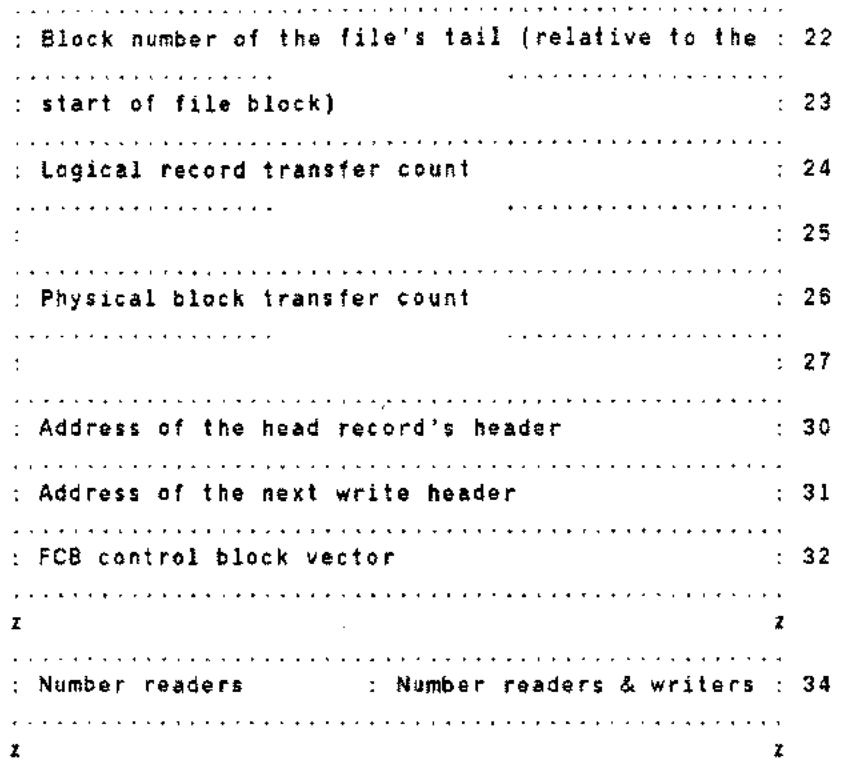
MACB Extension

This extension (contiguous with the MACB) contains IPC specific information, the location of reader and writer wait queues.[6] Beyond the extension are the message file buffers (also contiguous).

*Same fields as LACB

: Size of the ACB including buffers (words)	: 0	*
: File number	: 1	*
: File name	: 2	*
z	z	
: Foptions	: 6	*
: Aoptions	: 7	*
: Record size (bytes)	: 10	*
: Block size (words)	: 11	*
z	z	*
: Carriage control code (writers)	: 13	*
z	z	*
: Error code	: 16	*
: Transmission log (units same as last read/write)	: 17	*
: Total number of unread records (includes opens	: 20	
: and closes)	: 21	

[5]Reference MPE Tables Manual page 6-1.
[6]See Message Queue Mechanisms section 4.


Block number of the file's tail (relative to the : 22
start of file block) : 23
Logical record transfer count : 24
: 25
Physical block transfer count : 26
: 27
Address of the head record's header : 30
Address of the next write header : 31
FCB control block vector : 32
Number readers : Number readers & writers : 34
: Records per block : 36
:Wrt buf indx: : # buf - 1 : 37
Address of the head record's data : 40
Size of the buffer (words) : 41
: Logical device number : 46
:0:# rd buf : # wt buf :er :qw :m :c :d :s :f : 47
Number of max sized free records : 50
: 51
Number of free words in the current free record : 52
Address of the next write record : 53
Number of nondata records in the file : 54
: 55
# of read requests that have a claim on file : 56
Last read error : Last write error : 57

```
.....................................................
: DST number of the physical ACB                  : 60
.....................................................
: Address of the physical ACB                     : 61
.....................................................
: DST number of the logical ACB                   : 62
.....................................................
: Address of the logical ACB                      : 63
.....................................................
: DST rel address of the stack access control blk  : 64
.....................................................
: DST rel address of the DB area                  : 65
.....................................................
: MACB vector table entry address                 : 66
.....................................................
: MACB control block vector table address         : 67
.....................................................
: Target area's DST number                        : 70
.....................................................
: Reserved for calling parameters                  : 71
................                        ............
:                                                  : 72
................                        ............
:                                                  : 73
.....................................................
: Reserved for the stack marker from file system   : 74
................                        ............
: intrinsics                                       : 75
.....................................................
z                                                  z
.....................................................
: User's soft interrupt plabel                     : 100*
.....................................................
: Number of seconds to wait on boundary condition  : 101*
.....................................................
: O:Ex:Nd:Vr:Bt:Cls  :C : Carriage control         : 102*
:...................................................:
: Reply Port (basic IPC port)                      : 103*
.....................................................
: Writer ID                                        : 104*
.....................................................
: Control block index for nowait writer record buf : 105*
.....................................................
: DST relative addr of nowait writer record buffer : 106*
.....................................................
: No wait I/O resultant error code                 : 107*
.....................................................
: No wait I/O resultant transmission log           : 110*
.....................................................
```

BEGINNING OF MACB EXTENSION

Write wait queue (basic IPC port)	111
Read wait queue (basic IPC port)	112
Head record's length (bytes)	113
Head record's record type (same values as header)	114
Head record's writer ID	115
Head record's header word value	116
Max size record plus its overhead (words)	117
ACB wait queue message - contains same info as	120
the wait queue message in the Message Queue	121
Entry	122
	123
	124
Waiter's reply port, 0 if using ACB complt̃n area	125
ACB completion message area - *see Message Queue	126
Entry for completion message format	127
Waiting process' pin	130
Waiting process' file number	131
Waiting process' soft interrupt plabel	132
DST rel address of buffer one	133
DST rel address of buffer two	134
Etc.	135

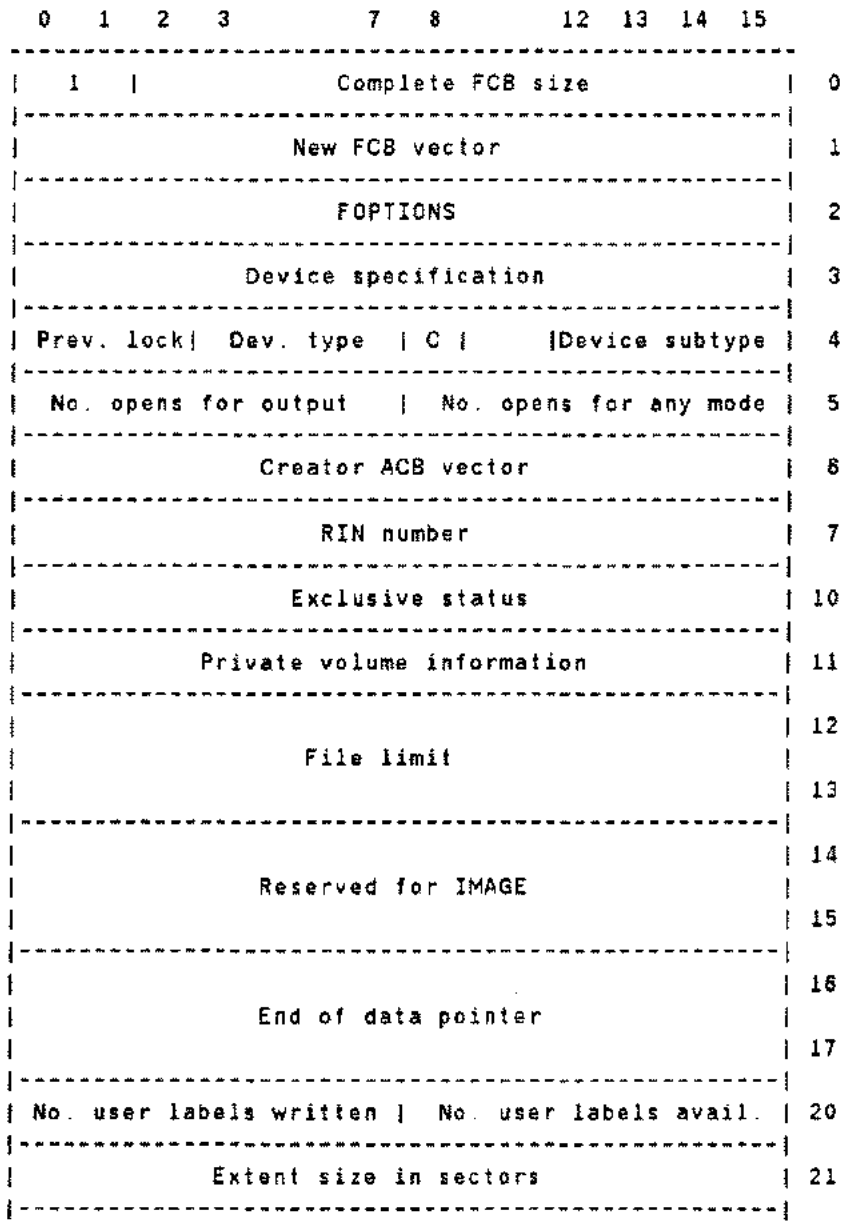
* Value is private to a particular accessor.

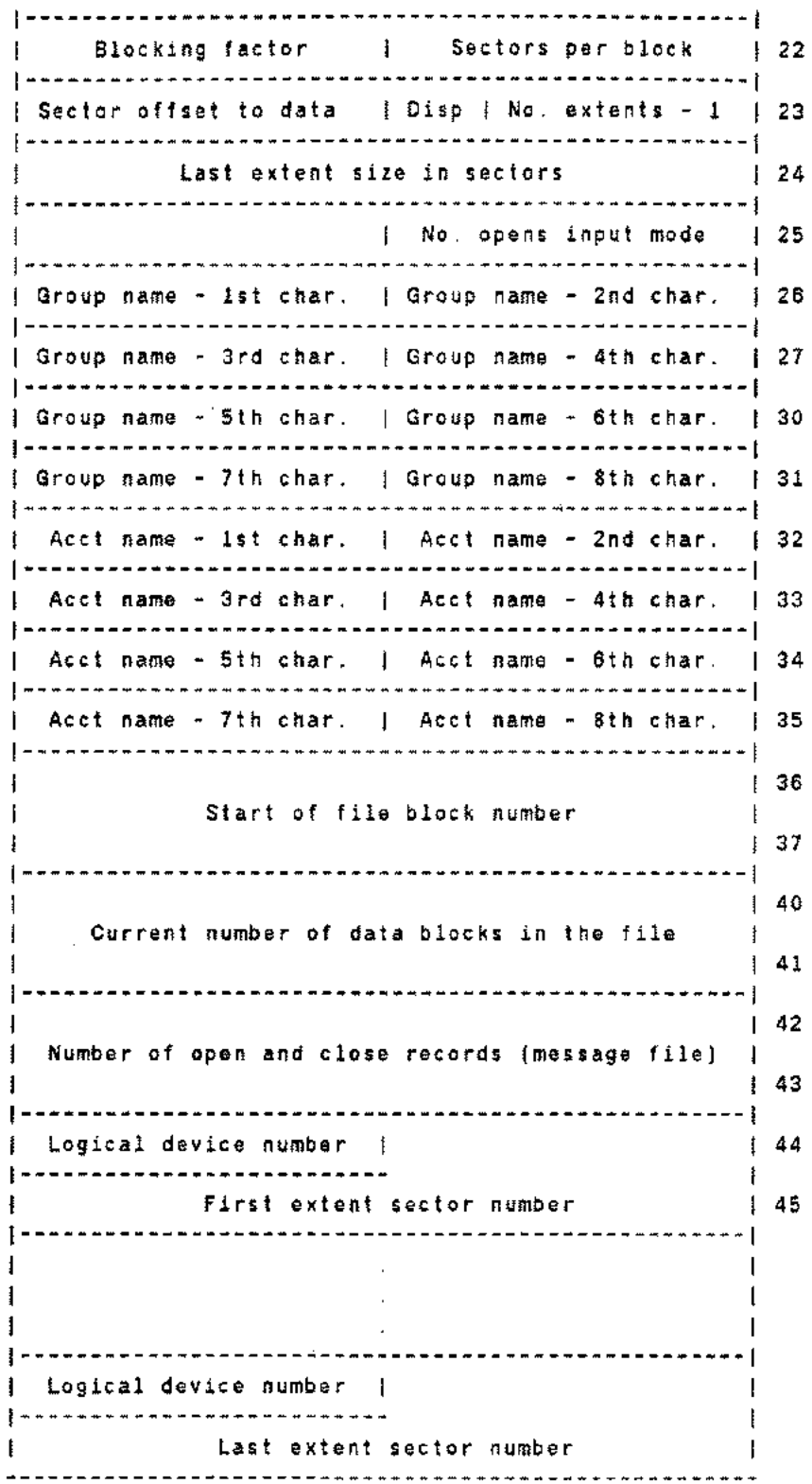
Word	Field	Description
47		File's global flags.
	(9:1)	er 1 - extended read
	(10:1)	qw 1 - one or more writers has been queued on the wait queue.
	(11:1)	m 1 - wait msg is located in the ACB
	(12:1)	c 1 - completion msg is located in the ACB
	(13:1)	d 1 - the current write buffer has dirty bit set
	(14:1)	s 1 - the start of file is block zero
	(15:1)	f 0 - the ACB buffers have not been filled
102		Accessor's local flags.
	(0:1)	0 1 - have not yet issued an FREAD/FWRITE against the file.
	(1:1)	ex 1 - extended wait mode.
	(2:1)	nd 1 - do not destroy the next record read.
	(3:1)	vr 1 - writer has not yet written his first record (ie., he is a virgin).
	(4:1)	bt 0 - transmission log should be expressed in words. 1 - " " " " " " bytes.
	(5:1)	cls - Not currently used (reserved for group IPC standard).
	(6:1)	C - No wait completion message is in LACB area.
	(8:8)	car ctl - carriage control character to be used for the writer's record (a value of one indicates no carriage control character).

File Control Block

The file control block data structure and placement strategy is the same as that for standard files. Information used by the file system is copied from the file label into the FCB. The FCB is created with the first opener of the file and when more than one process has opened the message file, the FCB will reside in a control block table in an extra data segment or Shared System Control Block Table (See MPE Tables Manual, The File System - Control Block Area). The file label and MACB have pointers to the File Control Block called the FCB vector (see "vector format" under Active File Table later in this section).

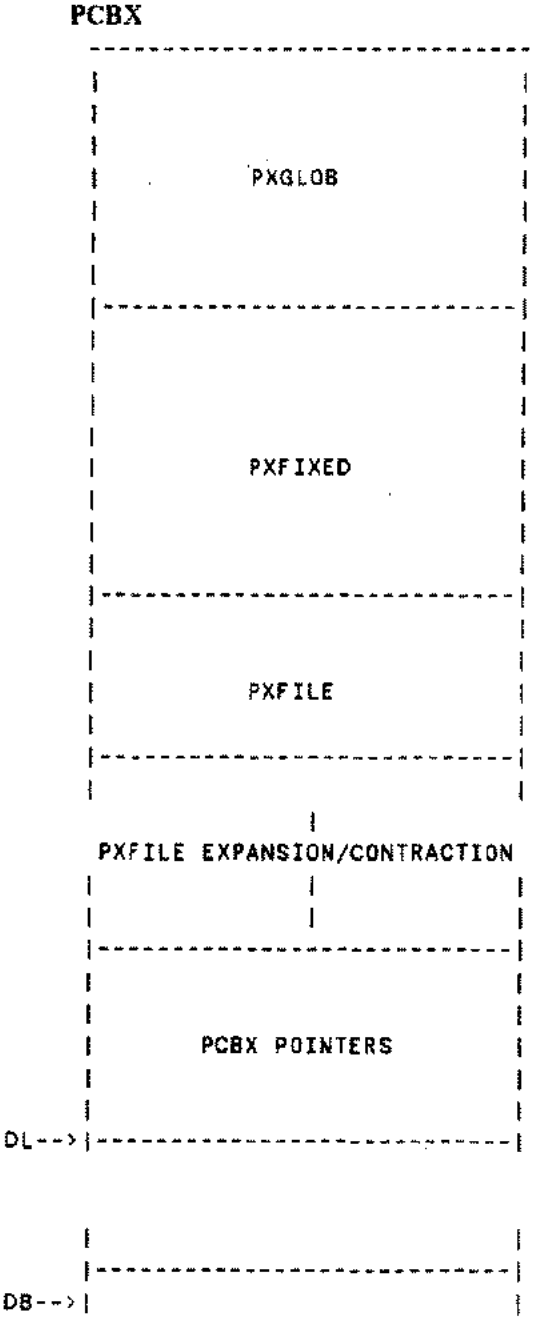
The FCB has the following format:




Blocking factor
Sectors per block
22
Sector offset to data
Disp
No. extents - 1
23
Last extent size in sectors
24
No. opens input mode
25
Group name - 1st char.
Group name - 2nd char.
26
Group name - 3rd char.
Group name - 4th char.
27
Group name - 5th char.
Group name - 6th char.
30
Group name - 7th char.
Group name - 8th char.
31
Acct name - 1st char.
Acct name - 2nd char.
32
Acct name - 3rd char.
Acct name - 4th char.
33
Acct name - 5th char.
Acct name - 6th char.
34
Acct name - 7th char.
Acct name - 8th char.
35
Start of file block number
36
37
Current number of data blocks in the file
40
41
Number of open and close records (message file)
42
43
Logical device number
44
First extent sector number
45
Logical device number
Last extent sector number

Control Block Access

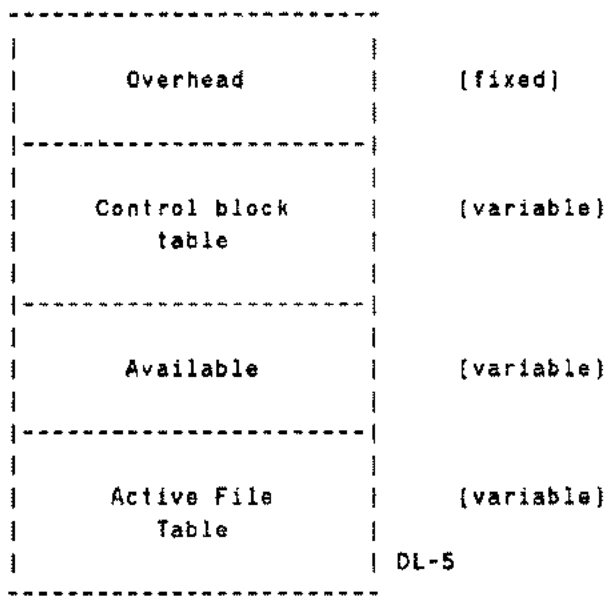
This is PCBX area (Process Control Block Extension) that sits below DL in the process's stack. The PXFILE area lies within the PCBX area and control blocks are contained in the PXFILE area. The PCBX is used to hold process specific information that can be swapped out with the stack.



PXFILE

The PXFILE area is a contiguous, expandable block of storage that is managed by the file system.

The overall structure of the PXFILE area is:



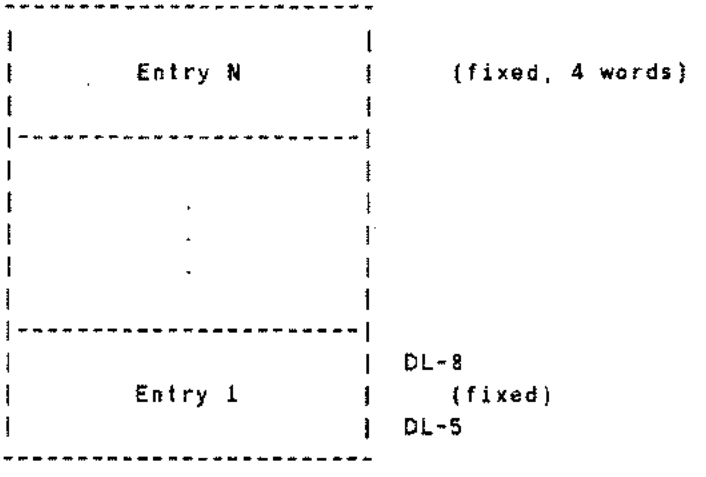
Available Block

The part of the PXFILE area labeled "Available" is used to provide space when the Control Block Table or the Active File Table is expanded. If the Available area is exhausted, the entire PXFILE area is expanded, the AFT is relocated and new space is added to the Available Block. Currently the PXFILE area is only expanded; it is never contracted.

Active File Table (AFT)

The Active (or Available) File Table contains information used by the file system to coursely characterize file access and, most importantly, to give the location of the file's control blocks.

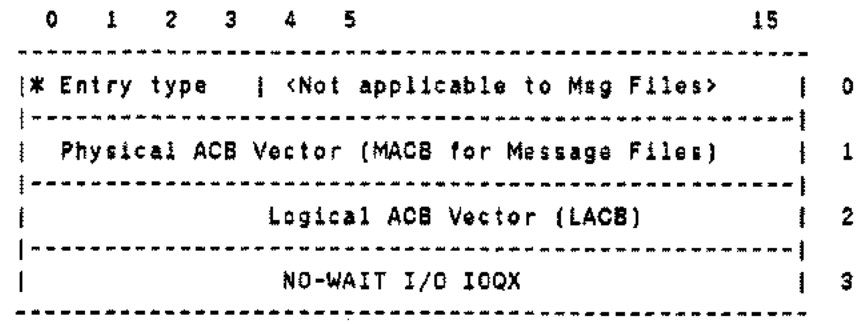
The overall structure of the AFT is:



The AFT is negatively indexed by file number: the entry at DL-5 corresponds to file number 1, the entry at DL-9 corresponds to file number 2[7], etc.

[7]For sessions, file #1 and #2, refer to $STDIN and $STDLIST respectively

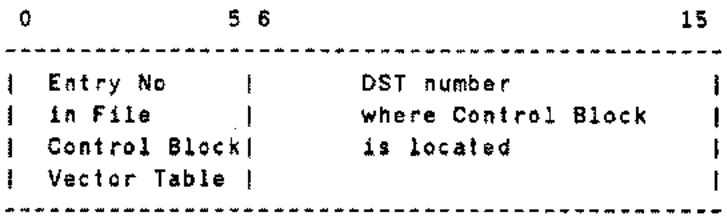
The structure of a file system AFT entry is:



*** Message Files have an entry type %10.**

The file system will find file control blocks (e.g. during an FREAD or FWRITE) by negatively addressing into the Available File Table and using the appropriate vector to locate a control block.

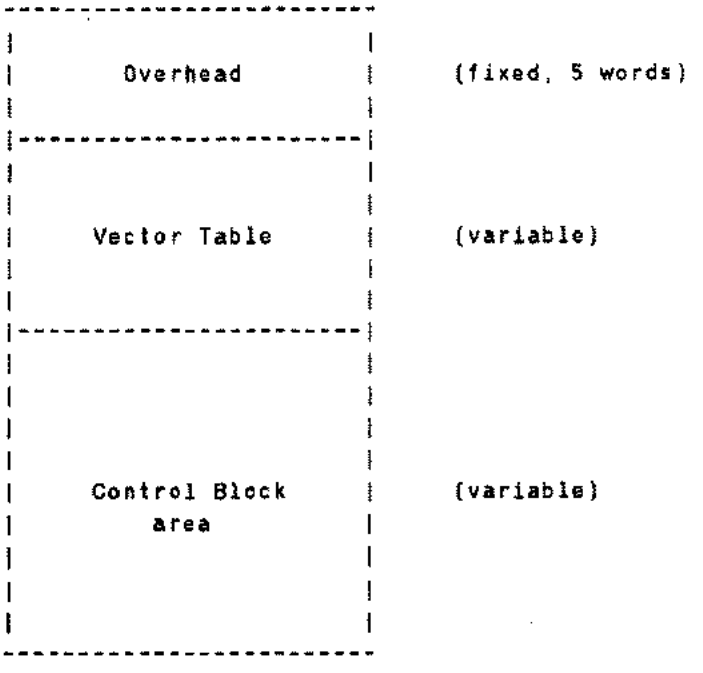
The format for a vector; LACB, PACB or FCB (found in the PACB vector) is:



Control Block Table

The control block table holds either one control block or multiple control blocks. A control block table may either reside in the process's stack in the PCBX area or in an extra data segment (see file system narrative in the MPE Tables Manual File System; section 3.2 File Control Block Table).

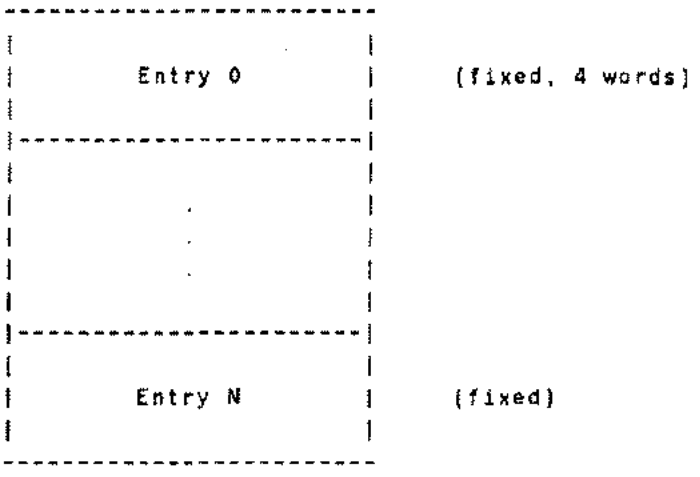
The overall structure of a control block table is:



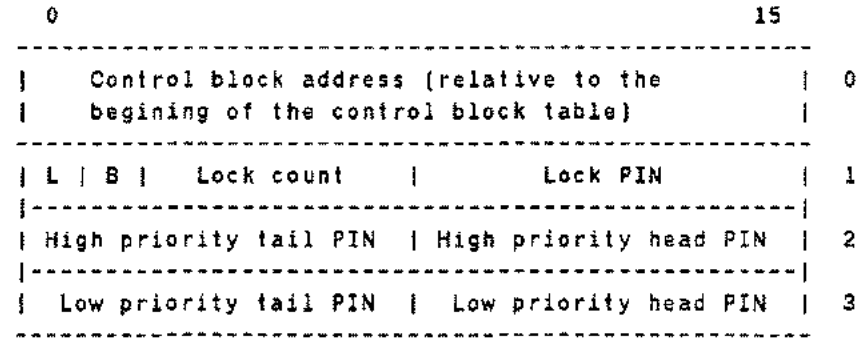
Vector Table

Vectors in the AFT point to vector table entries in the Control Block Table. Vector Table entries contain the addresses of the control blocks (**addresses are relative to the beginning of the Control Block Table) contained in the control block table. The file system also uses the fields in a vector table entry to lock or unlock control blocks in the control block table while they are being accessed.

The overall structure of the vector table is:



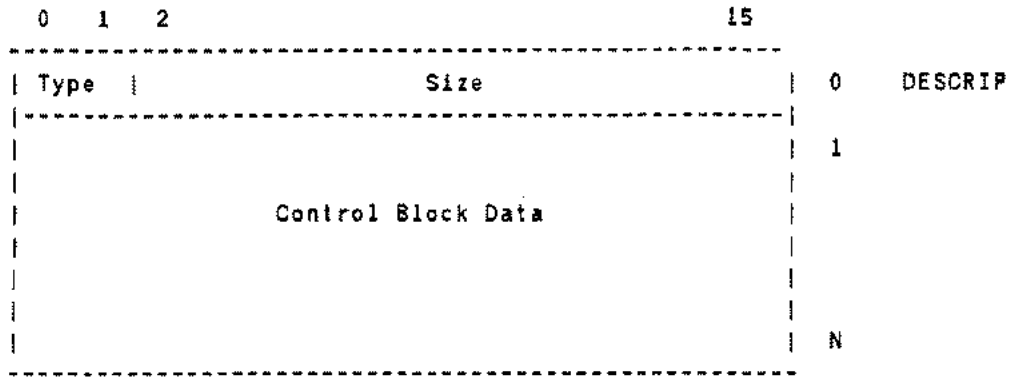
(An unused vector table entry will have zeroes in all the words of the entry.)
The structure of a vector table entry is:



The lock pin field will have the pin number of the process that has locked the control block. Control blocks are locked while they are being accessed so no other processes may write to them. The High Priority head and tail pin in word two are the head and tail of processes that are waiting to access that control block and are impeded. The remaining links for this queue are the NIMP pin (next impeded process) and PIMP pin (previous impeded pin) found in the PCB entry for that process.

Control Block Area

The part labeled CONTROL BLOCK AREA contains the control blocks used by the file system. To facilitate storage management, all control blocks have the same overall structure:



DESCRIP — This is the first word of a control block; the format is common for all control blocks.

SIZE — This is the size (in words) of the control block. The size includes the descriptor word.

TYPE — This is the type number of the control block. There are four types of control blocks:

0 - Garbage
1 - FCB
2 - PACB
3 - LACB

When a control block table is created, the initial control blocks are completely allocated to a single control block area of type garbage. When space is requested for a new control block, the control blocks are scanned (using a first fit algorithm) for a garbage control block that is as large as the size requested. The space for the new control block is taken from this garbage control block and the space remaining belongs to the new garbage control block size.

If space is requested and no garbage control block is large enough to contain the new control block, then the control block area and control block table are expanded by a sufficient amount. If expansion is not possible, some other control block table must be used.

REVIEW

When a file system intrinsic is executed, the file access procedures need to get at file information. Information is sought by the file number which tells the file system to:

1. Negatively address into the Active File Table (4 x File Number).

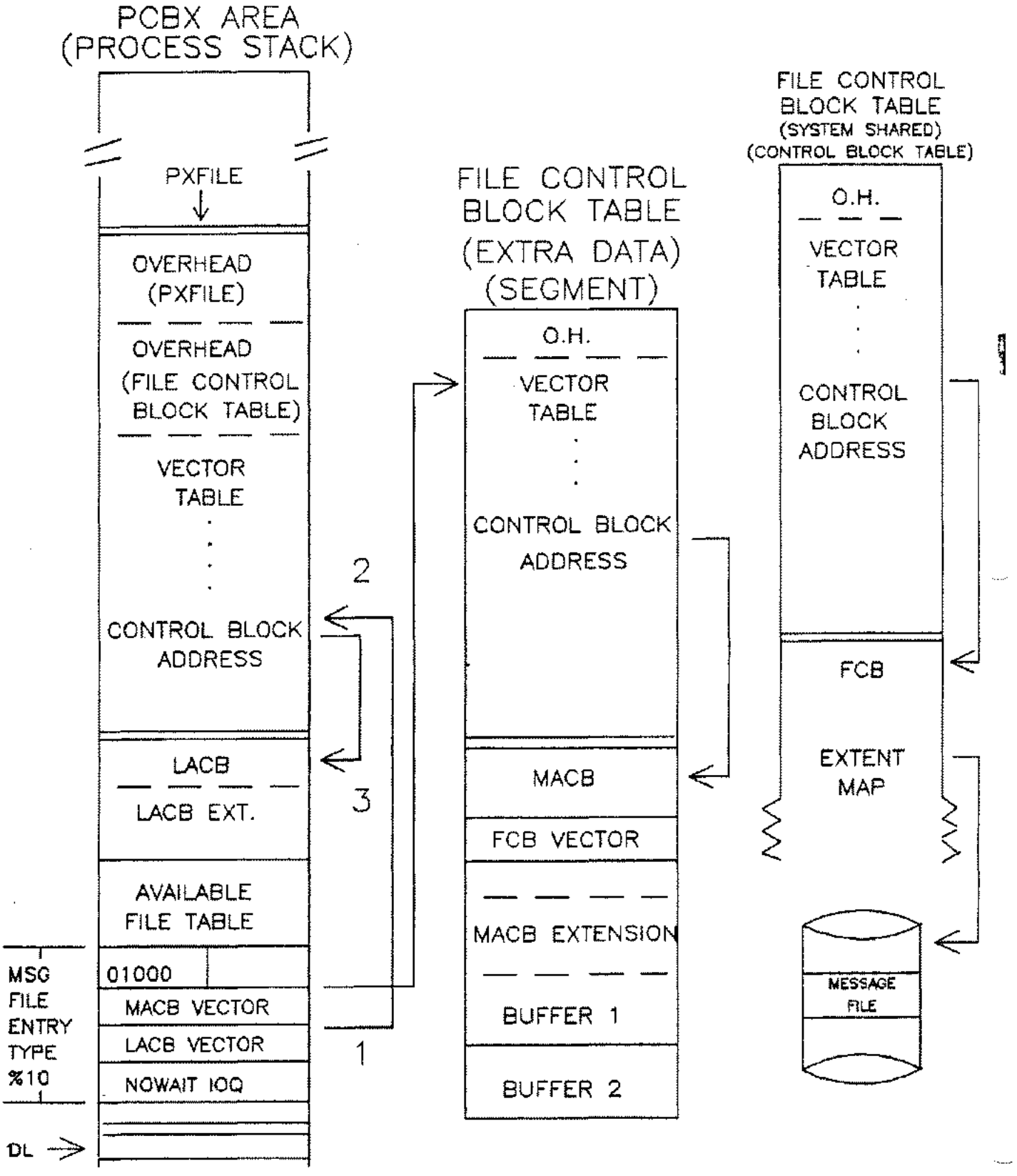
2. Find the vector it needs for either an LACB or PACB (MACB) in the AFT entry. The vector will give a DST# where the vector table is contained and an index or entry into the vector table.

3. The vector table entry is an address relative to the beginning of the Control Block Table where the control block may be found.[8]

(see picture on the following page)

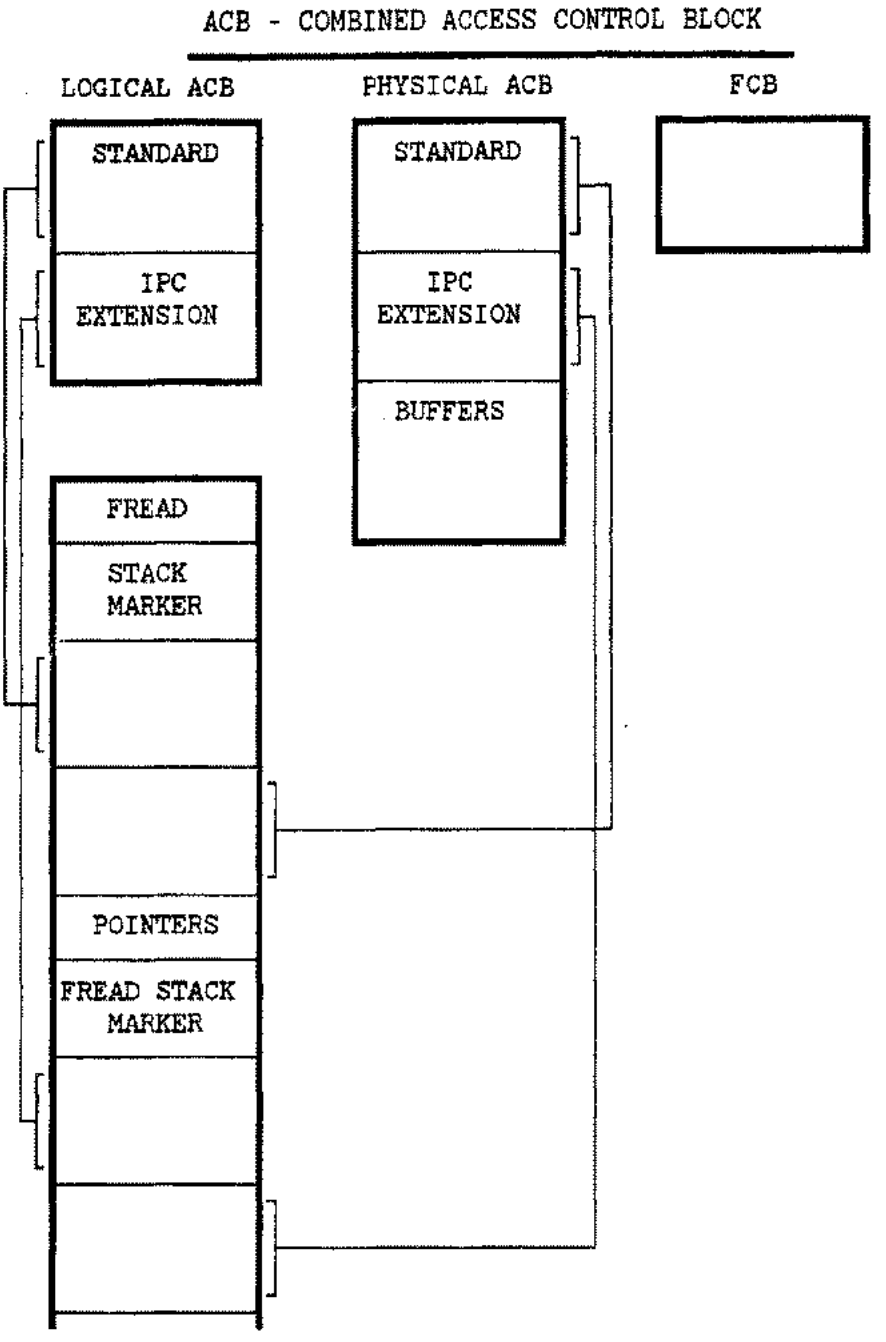
[8]See Appendix B for a dump example of finding control blocks.

CONTROL BLOCK ACCESS

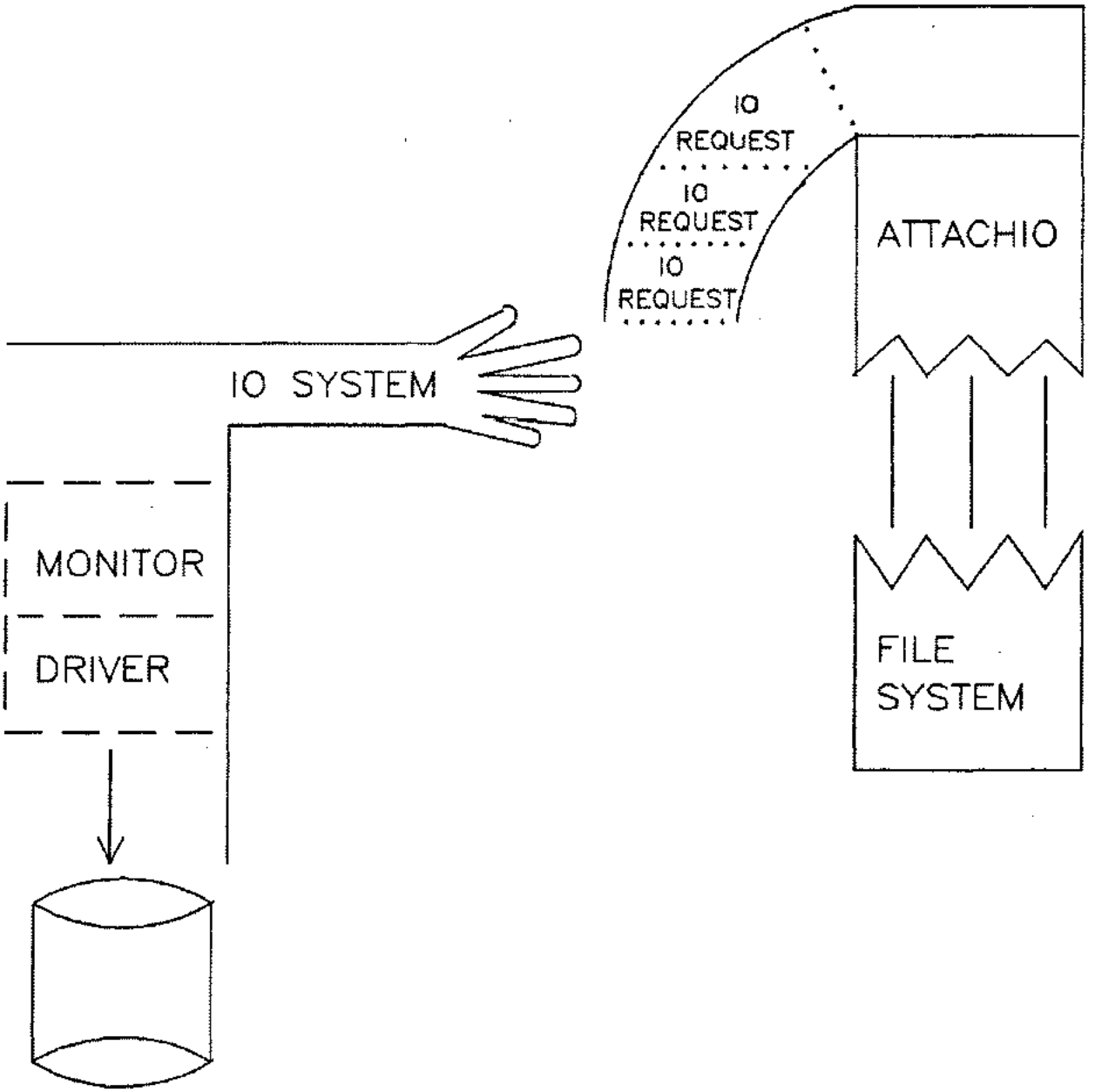


File Access

To read or write to a message file, the file system must be able to obtain the information in the file's control blocks. Instead of doing several exchange DB's to get at different data segments that contain control blocks, the FREAD and FWRITE intrinsics lay the control blocks on the stack. You may have noticed already that the MACB contains the LACB (words 0-%17) and LACB extension (words %100-%110). When the MACB is laid on the stack it is overlayed by these two other control blocks unique to the caller of that intrinsic. We refer to these control blocks when they are on the stack as the **Combined Access Control Block.**


ACB - COMBINED ACCESS CONTROL BLOCK
LOGICAL ACB
PHYSICAL ACB
FCB
STANDARD
IPC EXTENSION
STANDARD
IPC EXTENSION
BUFFERS
FREAD
STACK MARKER
POINTERS
FREAD STACK MARKER

When the control blocks are on the stack and accessible, then a call to the I/O system can be made for the physical I/O to be performed. Attatchio is the procedure that bridges the file sytem and the I/O system. Some of the parameters it passes are function: read or write etc.; a byte count to be transferred; and a destination or source of the transfer. The I/O system creates a request for the I/O and puts the request on the IOQ until the driver can complete the request.



Buffer Management and Anticipatory Reading

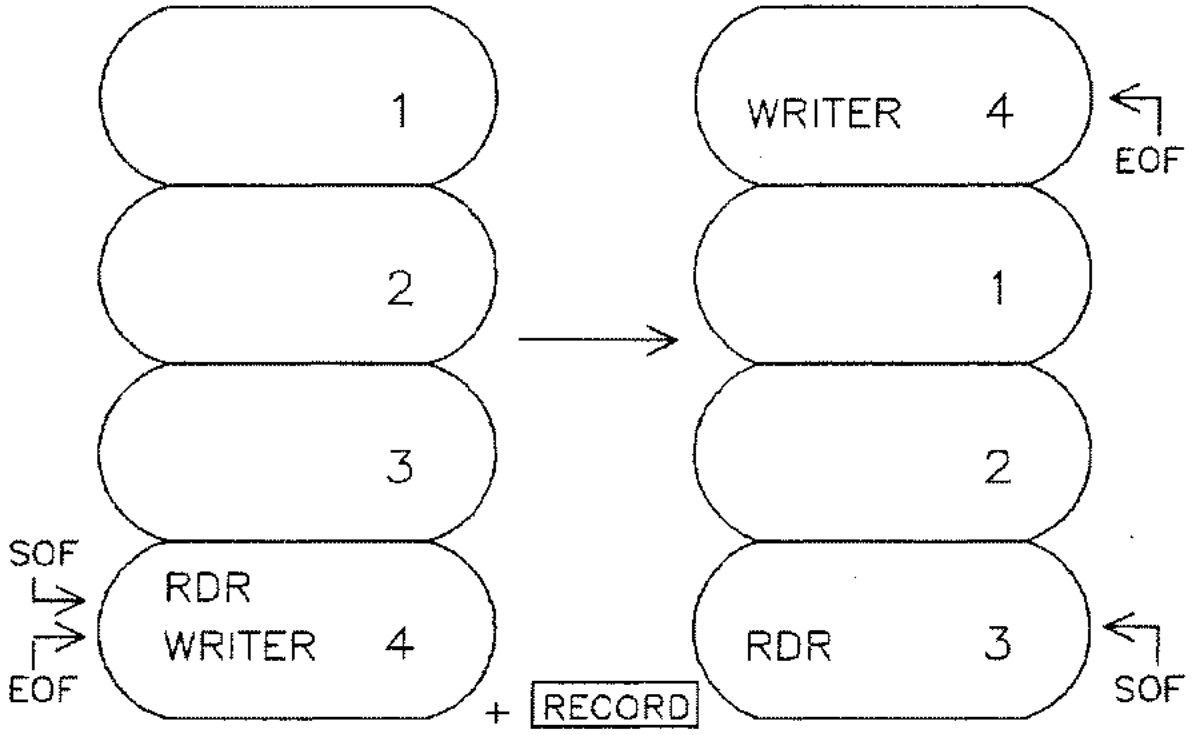
Message files support anticipatory reads. When the record number is passed in an FREAD; the file system uses the information in the complete ACB to calculate what block contains the record[9] and calculates which buffer to use for that block . The file system then makes a request for the transfer of the block containing that record. When the process "waits" for the record (with IOWAIT or IODONTWAIT) then the record is transferred from the buffer to the target DST. When a process writes to a message file the buffer is not "flushed" -- records are not actually transferred to the disc file until the buffer becomes filled.

Both reader and writer buffers are contained in the same pool of buffers physically attached to the end of the MACB. Reader buffers are at the "head" of the queue whereas writer buffers comprise the "tail" of the queue. Buffer management runs in one of two modes.

9**Note that variable length record files have a blocking factor of one record per block.

COUPLED MODE

When all active, unread records may be contained in the buffers buffer management is said to be in coupled mode.[10] A spent read buffer[11] is added to the tail of the write buffer list. Writers write to the tail buffer in the list. If this happens to be shared with a reader, then no disc I/O is initiated when the writer has filled the buffer. Otherwise when the buffer is full and only used by the writer on the next write, the buffer is flushed, copied to disc, and then the record is written. Note that if all the accessors were to close the file then all data would be written to the disc. When the number of active records can no longer be held in the buffers, uncoupled mode is entered.



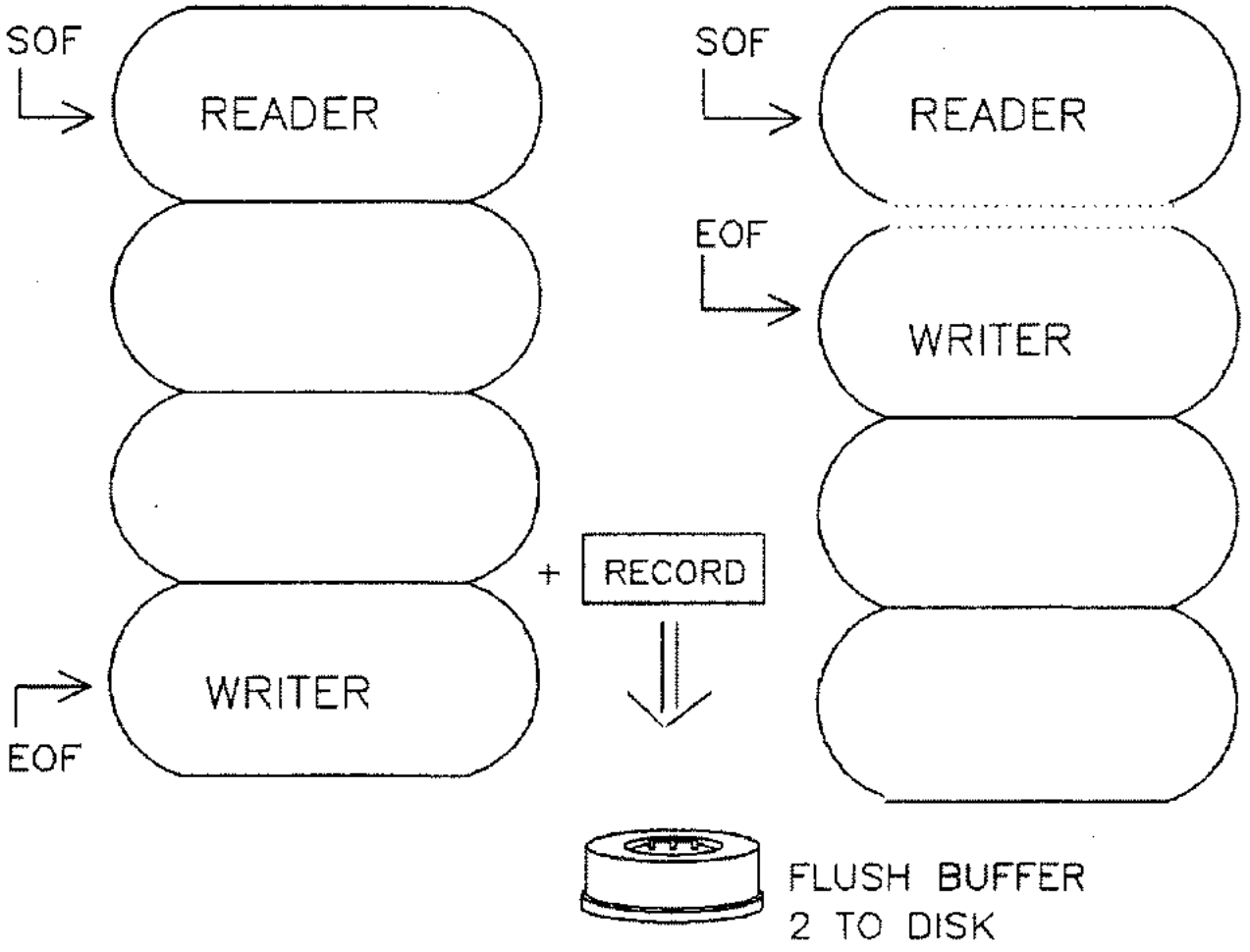
ALL ACTIVE RECORDS MAY BE CONTAINED IN THE BUFFERS

[10]Note that coupled mode is guaranteed with FOPEN.

[11]Remember that records are logically deleted after having been read. A spent read buffer is a block (physical record) in which all records have been deleted.

UNCOUPLED MODE

The active records cannot be contained in the buffer area. In uncoupled mode readers and writers use their own buffers independently. The writers will circularly use the buffers not being used by the reader and flush each buffer when it's necessary to add a record. Note that if the reader reads sufficient records such that the active records may be contained in the buffers, buffer management automatically reverts to coupled mode.



MESSAGE QUEUE MECHANISMS

SECTION IV

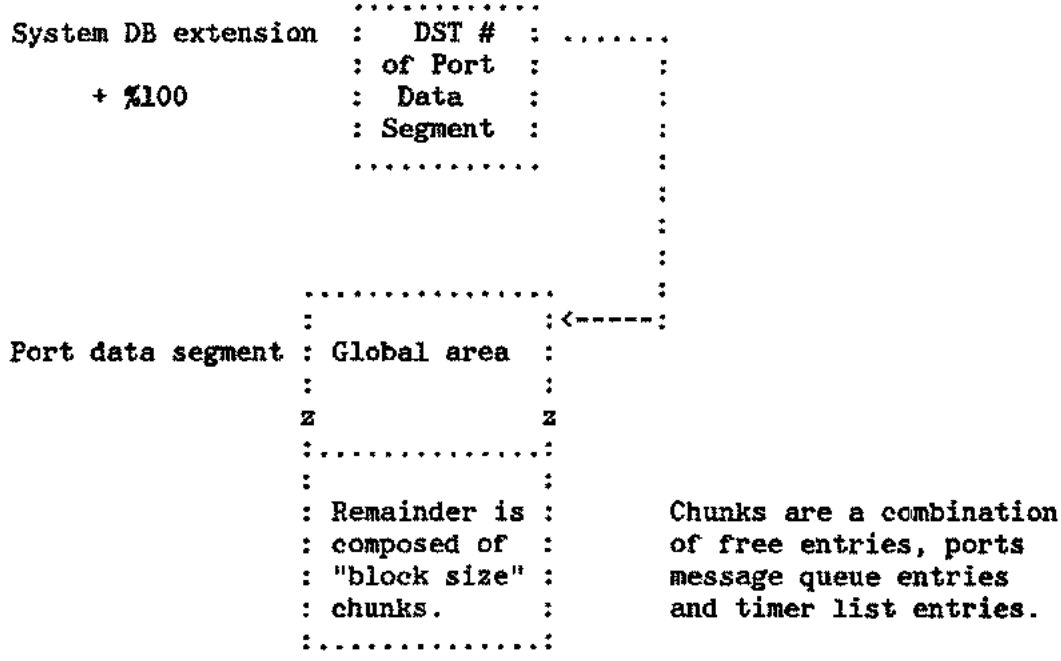
Port Data Segment

How does IPC maintain and control readers and writers that must wait for I/O after encountering a boundary condition writing to a full message file or reading from an empty message file? It uses queueing mechanism established in a data segment called the **Port Data Segment**. (Note that soft interrupts uses an additional incore message facility shared with system process communication) The port data segment is known as the heart of IPC. It contains:

1) **A Reader Wait Queue and Writer Wait Queue** for each message file that's open, to queue readers and writers waiting on boundary conditions.

2) **Return/Reply Ports** to inform readers or writers of completed I/O.

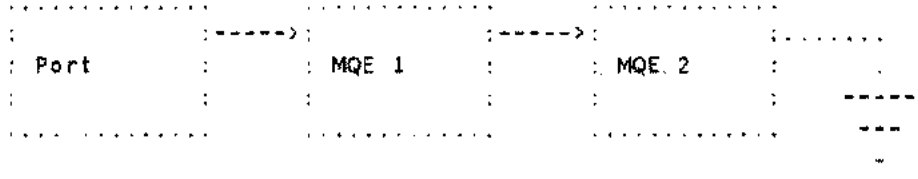
3) **Timer List Entries** to implement time-outs for read and write requests.



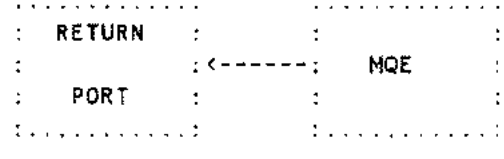
In this section the port data segment and its components are generally characterized. Figures of the data structures follow and then the port data segment operation is explained.

Wait Queues A wait queue is comprised of a "port" which is header information for the queue, and "Message Queue Entries" (a record attached to the port which signifies a process is waiting on an I/O boundary condition). There are two queues, a "reader" and a "writer" queue, for each message file. These queues are created when the first reader or writer opens the file. Reader and writer queue pointers are at words %110 and %111 in the message file's ACB.

Wait Queue with two outstanding messages

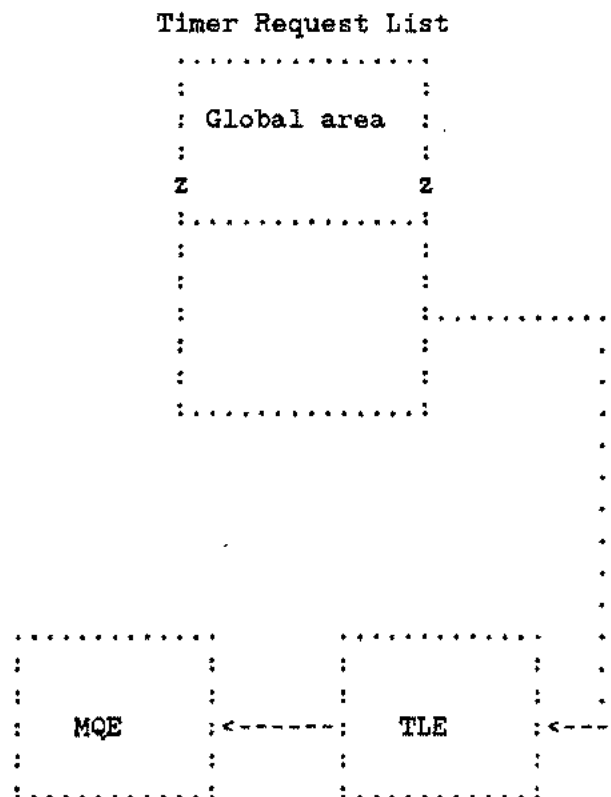


Reply Ports Each reader and writer has a "reply port." Reply ports inform their owners or "managers" their I/O requests have been satisfied. Reply ports receive the completed read/write requests by linking a message queue entry to the port. (The current limitation of IOWAIT is one request outstanding, so the reply ports will have a maximum queue of one entry). The reply port is created at FOPEN time and the process that creates and owns that port is referred to as the "ports manager."

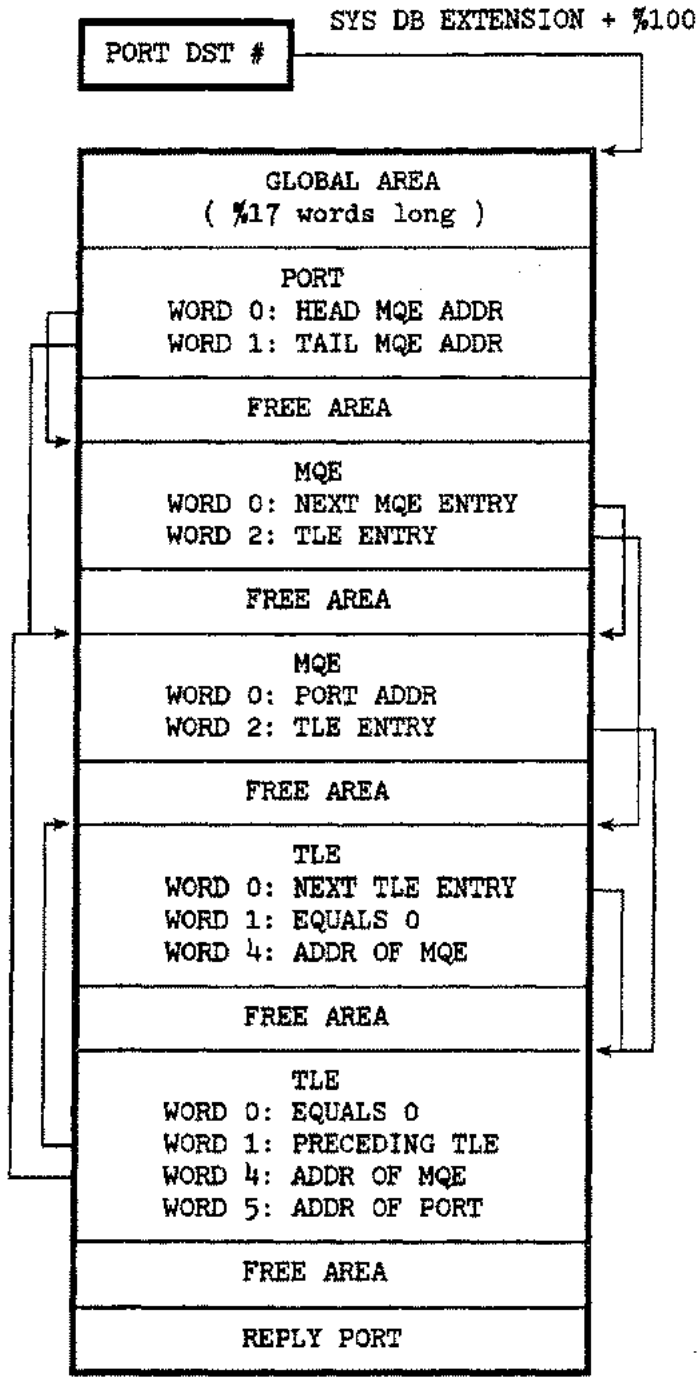


Timer List Entries

Timer List Entries (TLE) are created when the process sets a time-out interval with an FCONTROL 4. The TLE with the smallest remaining interval has an entry in the Timer Request List (TRL) and also heads a linked list of TRLs throughout the port data segment. UCOP checks the timer request list for events that time-out. When UCOP discovers a message file read or write time out, the message queue entry would be detached from its port and an EOF condition would be sent to the user. The next TLE (being the smallest time-out) would take its place in the system Timer Request List. They are also linked to the MQE's instead of having the MQE's and TLE's being contained in the same entry. This scheme was decided upon because it was thought that TLE's would not be widely used, thus saving the overall size of the MQE entry.



Port Data Segment



MQE: Message Queue Entry
TLE: Timer List Entry

PORT DATA SEGMENT GLOBAL AREA

The PDS global area is 17 octal words in length. It contains global information to manage the wait queues, reply ports and time list entries.

* Processes may be impeded waiting for free blocks in the Port Data Segment. These processes will form a linked list with the head and tail PIN number at words %10 and %11 in the port data segment global area, and PIMP PIN (previous impeded process) and NIMP PIN fields (next impeded process) in the PCB. * The TLE (the smallest time-out) address is in word %112. It heads a thread of TLE's through the Port Data Segment. The first TLE is also an entry in the system Timer Request List. That entry number is kept in word %13.

0	Data segment number of this port data segment	0
1	Block size in words	1
2	Total number of blocks	2
3	Maximum number of blocks	3
4	Current number of free blocks	4
5	Number of open ports	5
6	Head of free list	6
7	Tail of free list	7
10	Head of impeded process list	8
11	Tail of impeded process list	9
12	Head of timeout thread (TQE address)	10
13	TRLX of timeout	11
14	Value returned by TIMER intrinsic when	12
15	Timeout was initiated.	13
16	Head of Port List	14
17	Not Used	15

PORTS

All entries in the Port Data Segment (ports and message queue entries and Timer List Entries) are ten octal words long. Ports contain overhead information to maintain queues. You will find a port for each wait queue that is open. And there is a reply or return for each message file opener.

NOTE

There is one exception to this rule. In semi-exclusive mode there may be at most one reader, and in exclusive mode there may be only one writer. The generality of a wait queue is not needed. Instead when the process attempts to read from the empty file or write to a full file the reply port number is placed directly into the MACB. If the process has opened the file with wait, then no reply port is used.

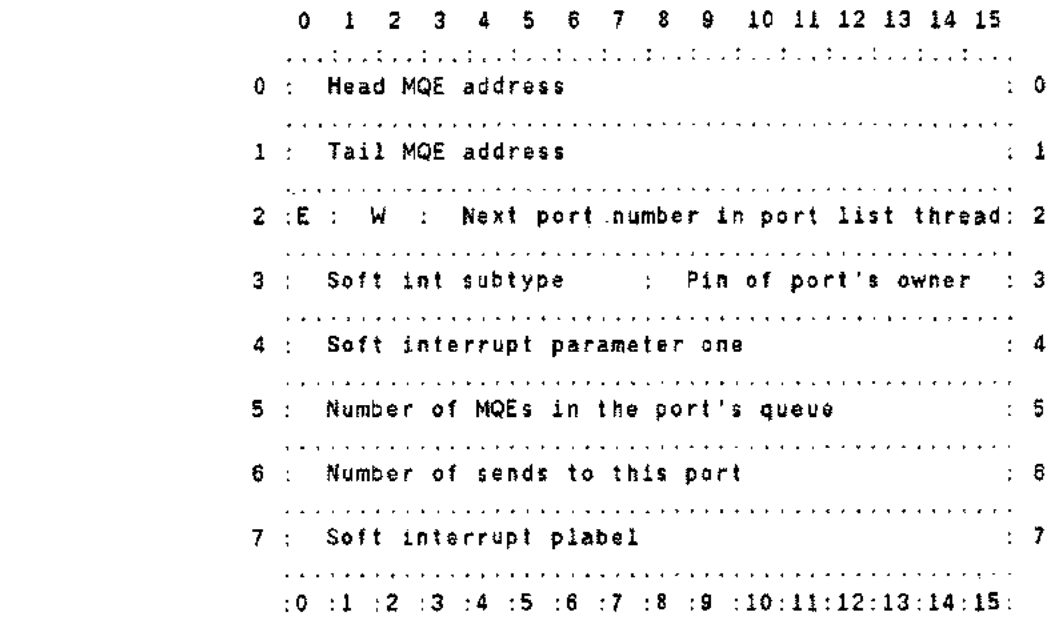
Information contained in a port is shown in the figure on the next page. Head and tail MQE pointers are used for access to queues. Soft interrupt subtype is always "1". The port manager (word 3 (8:8)) is the process that opens the port. Word 6, the number of sends, is an accumulated number of MQEs that have been attached to the port. And the plabel for the soft interrupt procedure is written when passed by an FCONTROL 48.

```
                .....................................
** PLABEL       :I/E:  :  STT ENTRY NO  :CST/CSTX NO :
                .....................................

                 I/E: Internal or External Bit
```

When soft interrupts are armed with an FCONTROL 45, IPC procedure FCONTROL turns on the **Enable Wake Up** bit (word %2 (0:1).[11] When the port is accessed, the **Enabled Wake Type** bit (word 2(1:1)) tells IPC procedures the action to take when a message is received.

[11]Note: This bit is also toggled by another IPC procedure - LONGWAIT).

PORT



W Type word 2(1:1)

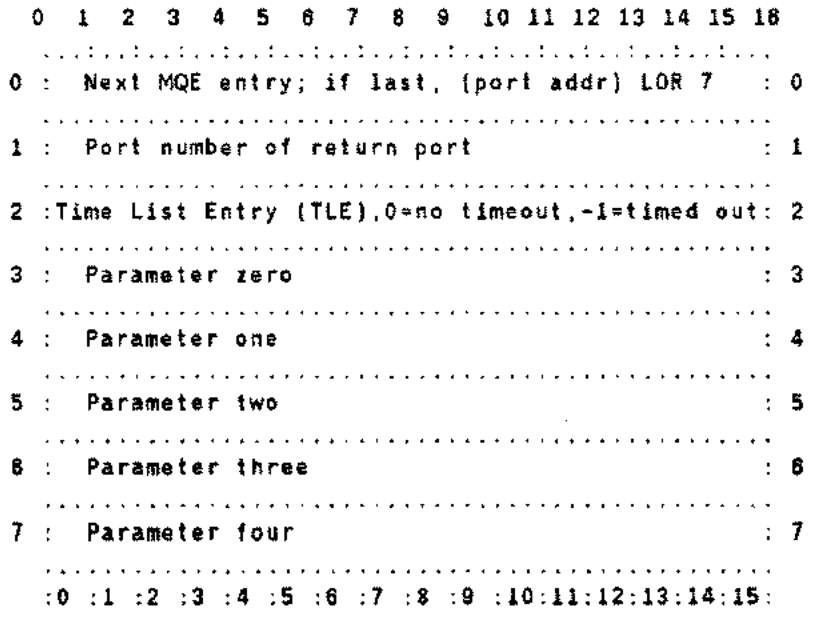
0 - Awaken on port

1 - User soft interrupt

2 - System soft interrupt

MESSAGE QUEUE ENTRIES

The Message Queue Entry records wait information when attached to the reader or writer wait queues: is there a timeout, the return port number, writers ID, DST number of the buffer, etc. When attached to a reply or return port an MQE stores the number of bytes transmitted or an error code for an incomplete read or write. See the parameter explanations on the following page for information contained in a wait message or completion message.



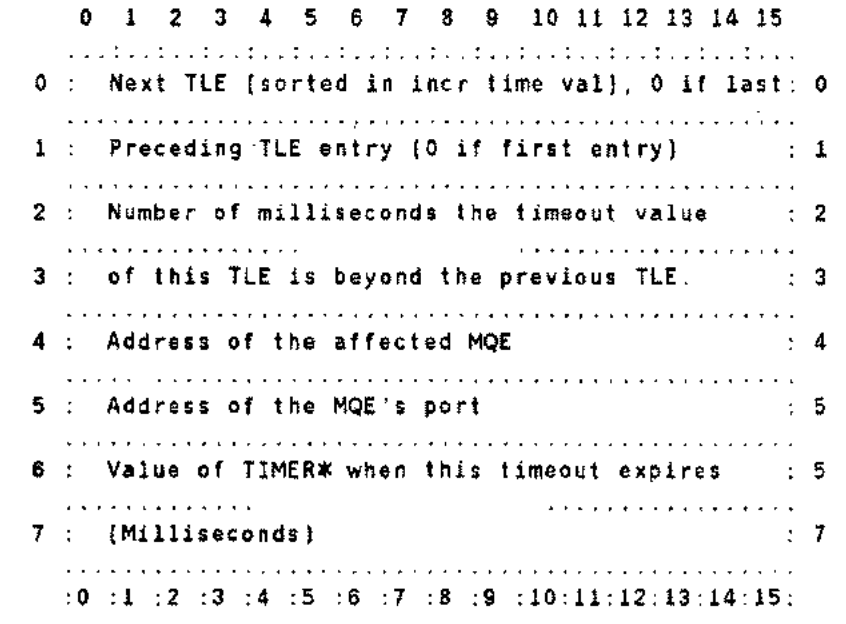
Timer entry definitions

0 - No timeout
1 - Timeout expired
2 - TLE address for a pending timeout

```
Wait Message

parm#
  0  -  WRITER ID
  1  -  LOCAL FLAGS (differ with each accessor)
        (0:1) - accessor just opened file
        (1:1) - will wait on boundary condition if no symbiotic process
        (3:1) - writer has not written a record
        (4:1) - transmission log in bytes
        (8:1) - carriage control code
  2  -  DST# of data buffer
  3  -  Address of data buffer (DST relative)
  4  -  Length of data buffer in bytes


  Completion Message

  0  -  Resultant error code
  1  -  Resultant transmission log in bytes
```

TIMER LIST ENTRY

Below is a Time List Entry. Its use and operation has been explained on page 4-3. The captions inside the picture should be sufficient to explain the TLE fields.



* TIMER is an internal MPE intrinsic that returns the time in milliseconds.

HOW DO WE FIND THE PORT DATA SEGMENT?

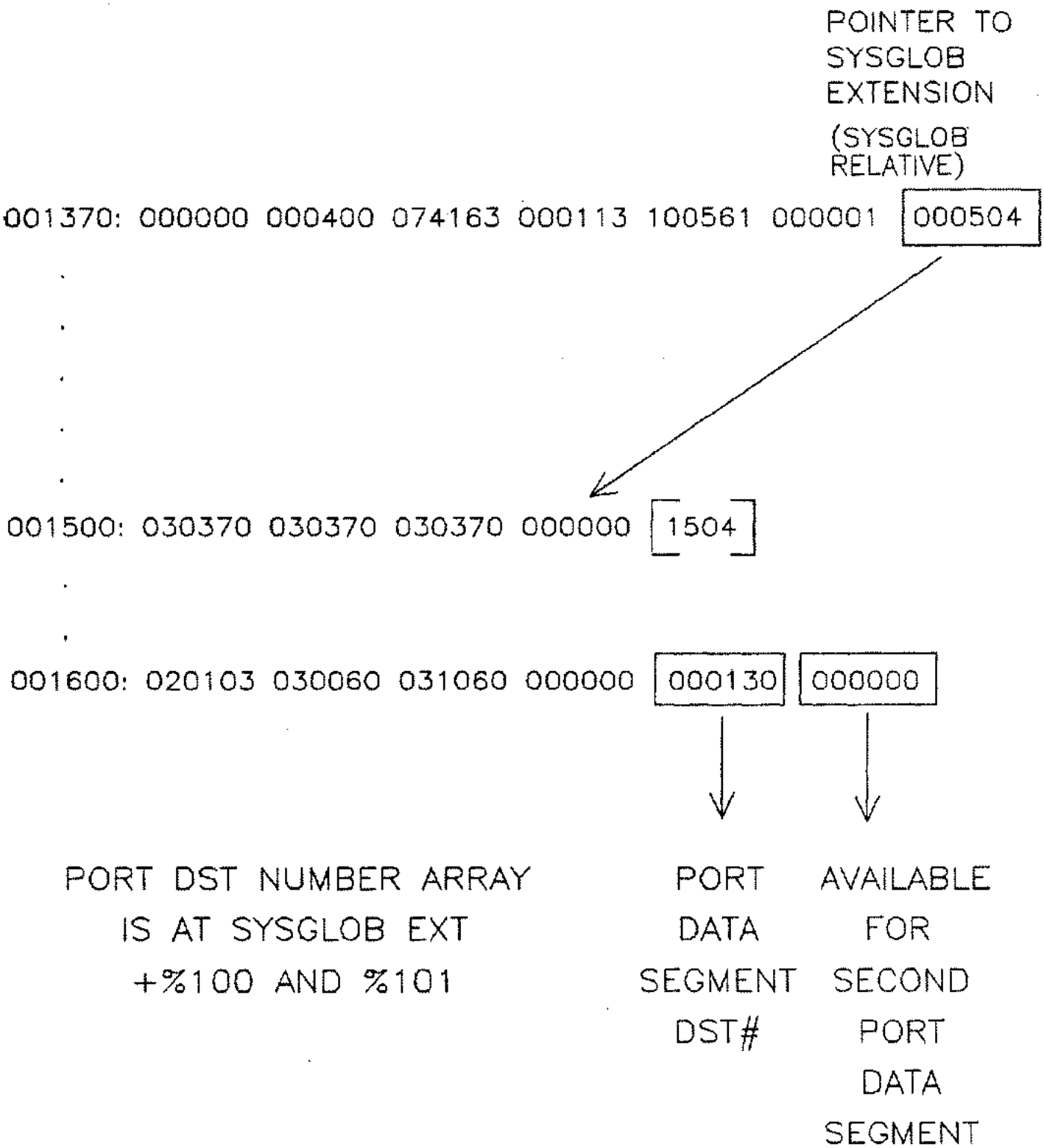
The DST number of the port data segment is found at Sysglob Extended +%100. The address of sysglob extension is located at %1377 (sysglob +%377). This is a sysglob relative address. Add it to %1000 and you will be at the beginning of the extension area. One-hundred and one-hundred-and-one octal words into the extension area is called the **DST Number Array**. The DST Number Array is simply two words, the first of which contains the DST number of the existing port data segment and the following word is reserved for a possible second port data segment.

PORT DST NUMBER ARRAY

Located in System DB Extension Area.

64	Port data segment number	64
65	Reserved for a second port segment	65

MEMORY DUMP


POINTER TO SYSGLOB EXTENSION (SYSGLOB RELATIVE)
001370: 000000 000400 074163 000113 100561 000001 000504
001500: 030370 030370 030370 000000 1504
001600: 020103 030060 031060 000000 000130 000000
PORT DST NUMBER ARRAY IS AT SYSGLOB EXT +%100 AND %101
PORT DATA SEGMENT DST#
AVAILABLE FOR SECOND PORT DATA SEGMENT

Port Data Segment Operation

This section explains Basic IPC operation and the fundamental operation of the port data segment. Soft interrupts and its message facility is explained later in this section. Below is a description of the port data segment operation a program will FOPEN a file, read from an empty message file or writing to a full message file.

FOPEN

When the message file is opened a reply port is initialized for the opener. If it is the first time the file is opened both reader and writer wait queues are created by establishing a port for each queue and write pointers to those ports in words %110 and %111 of the message ACB.

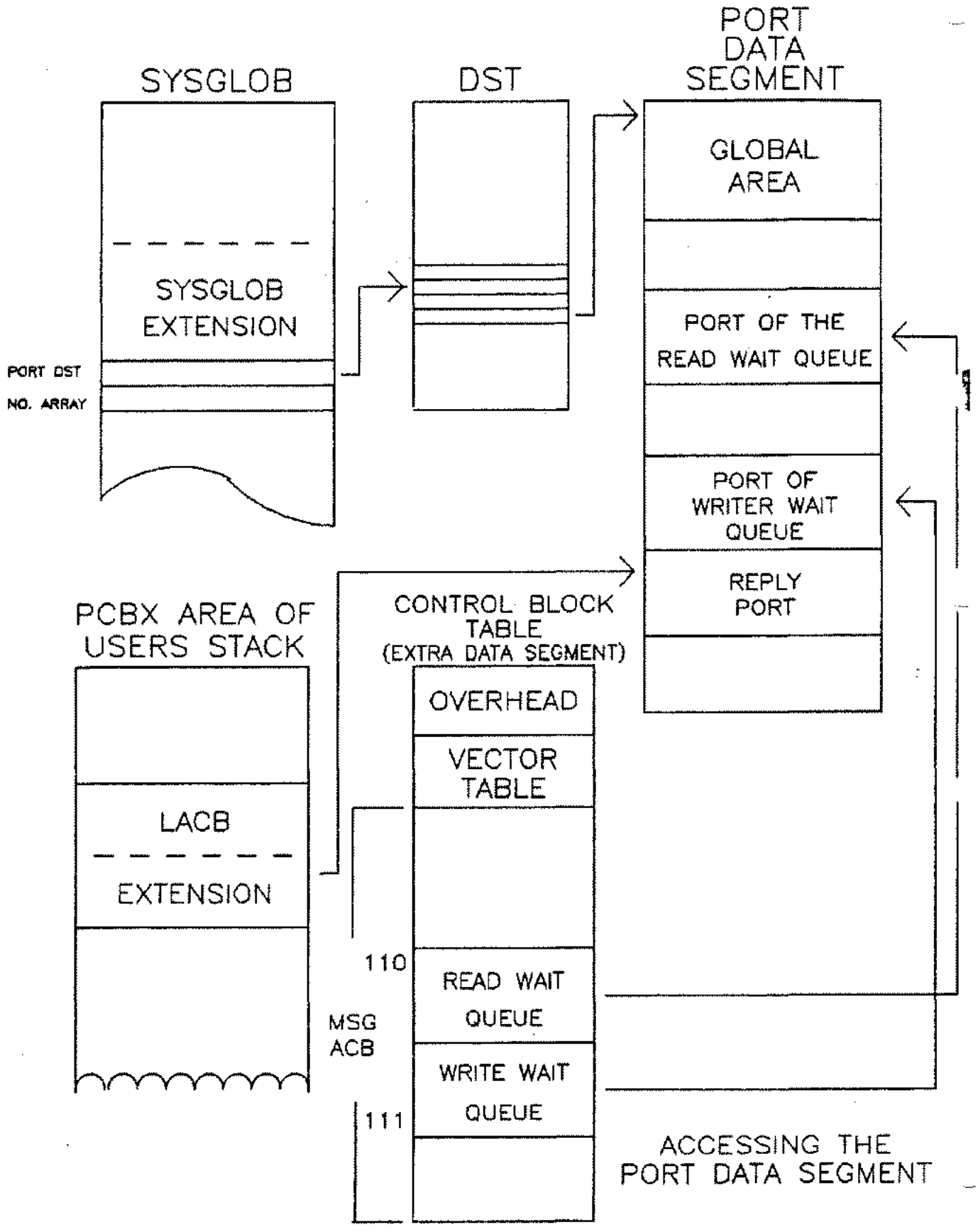
Pointers to ports have the format shown below. The port index refers to the word to use in the DST number array. Since we have only one port data segment at this time the index is always "0". The remaining part of the word indexes into the port data segment by port number.

PORT NUMBER

```
 0  1  2  3  4  5  6  7  8  9  10 11 12 13 14 15 16
.................................................
:Port index :  Port data segment relative addr/8   :
.................................................
```

The address of the openers reply port is in the third word of the openers LACB extension. There is one small exception, however, if the opener is using waited I/O, a reply port is never built.[13]

[13]Most customers will use nowait I/O in IPC applications because of its throughput advantage.


PORT
DATA
SEGMENT
SYSGLOB
DST
GLOBAL
AREA
SYSGLOB
EXTENSION
PORT OF THE
READ WAIT QUEUE
PORT DST
NO. ARRAY
PORT OF
WRITER WAIT
QUEUE
REPLY
PORT
PCBX AREA OF
USERS STACK
CONTROL BLOCK
TABLE
(EXTRA DATA SEGMENT)
OVERHEAD
VECTOR
TABLE
LACB
EXTENSION
110
READ WAIT
QUEUE
MSG
ACB
WRITE WAIT
QUEUE
111
ACCESSING THE
PORT DATA SEGMENT

WRITING

The usual course of a write is to 1) insure that there is room for the record, 2) write the record (usually just a data move to an ACB buffer), and, 3) if a reader is waiting send a successful message to the readers reply port.

The following exceptions might occur:

A. The first write.

The writers first write after an FOPEN consists of writing the open record, the data record and allocating space for the closed record.

B. Impeded writer.

When the file is first opened the amount of free file space (in max-size records) is calculated. Each write to the file first subtracts its record size and record overhead from the free space. If the record will not fit into the file, then the writer is *waited if:

1) a reader has opened the file, or

2) this is the writers first write after the open, or

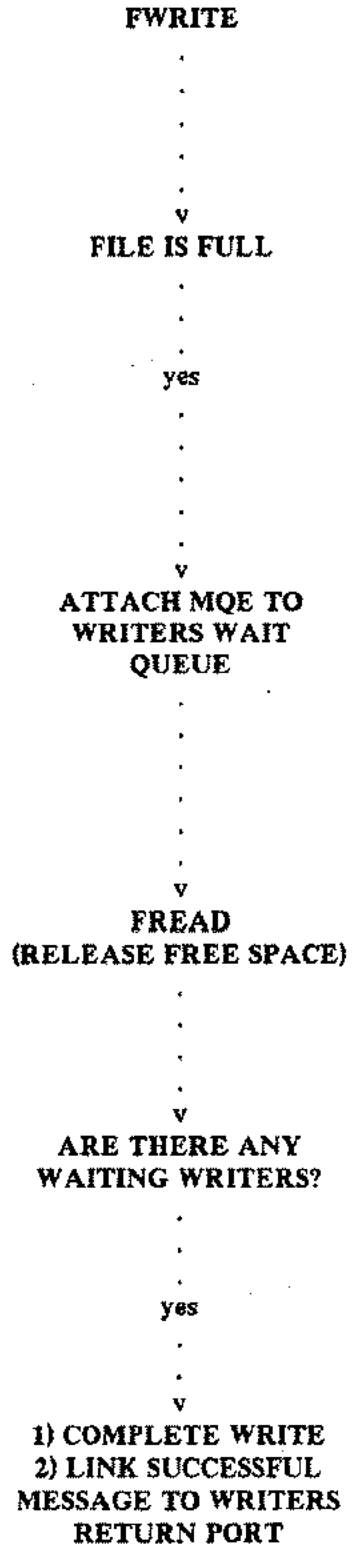
3) the writer has specified extended wait (FCONTROL 45).

* Waiting consists of placing a message on the wait queue. Eventually a reader returns free space (a block's worth at a time). The actual write is performed by the reader at this time. This is to:

1) Expedite the no-wait writer's data, and

2) insure the released writers will fill the file in the same order as they were freed (any other sequence may result in the writers running out of file space).

When the transfer is completed a successful message is attached to the writer's reply port.


FWRITE
v
FILE IS FULL
yes
v
ATTACH MQE TO
WRITERS WAIT
QUEUE
v
FREAD
(RELEASE FREE SPACE)
v
ARE THERE ANY
WAITING WRITERS?
yes
v
1) COMPLETE WRITE
2) LINK SUCCESSFUL
MESSAGE TO WRITERS
RETURN PORT

READING

Most read requests read a record and if the next record is in another block, issue an anticipating read of the block from the disc.

The following exceptions may occur:

A) Empty File

The reader decrements a counting semaphore (number of records in the file) when the semaphore goes to "0" there are no more records left. The reader is waited if:

1) One or more writers has opened the file

2) Or this is the readers first read to the file after the open

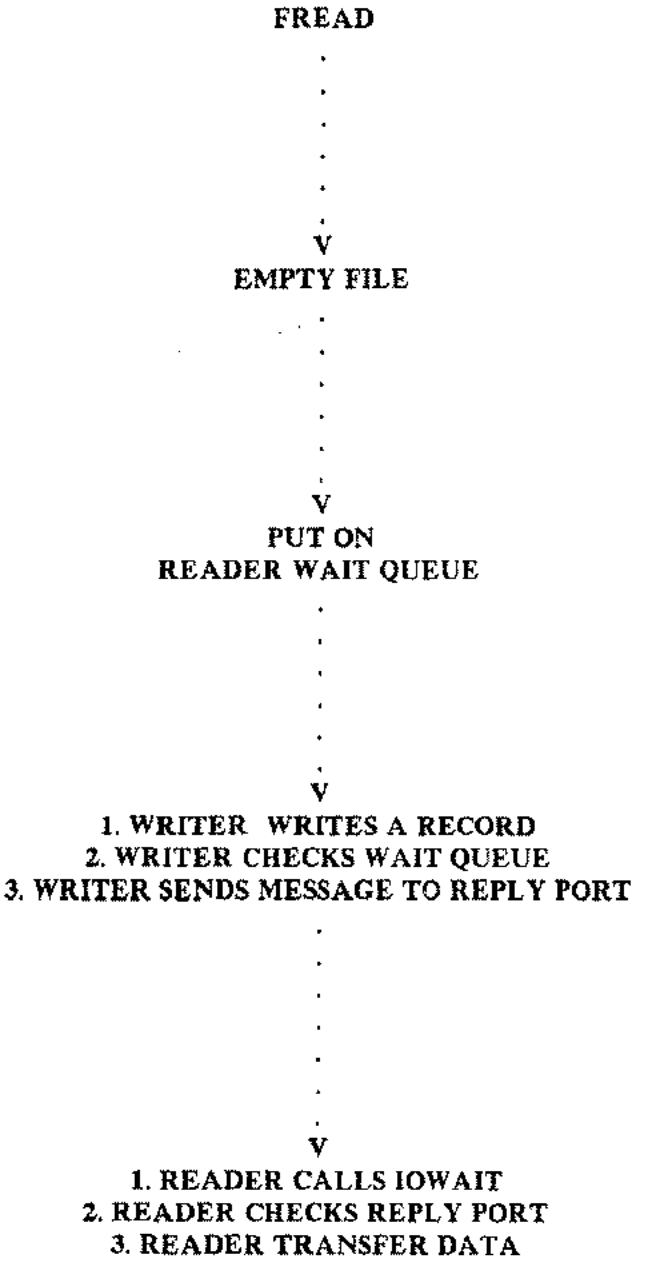
3) Or reader has specified extended wait (FCONTROL 45). [NOTE: Do not confuse this with extended read (FCONTROL 46)]

Again, parallel with writing to a message file *waiting consists of placing a message on the reader's wait queue. Each time a writer writes a record the writer checks the reader wait queue. When the writer discovers an impeded reader it will send a successful message to the reader's reply port. The reader then (or at I/O wait time) performs the data transfers.

Waited readers, unlike waited writers, perform their own data movement upon being liberated. They can do this because they are freed with a claim on any record in the file -- not a particular one. Thus it's permissible for other readers to issues FREAD/IOWAITs between another reader's FREAD and IOWAIT. The only constraint is that a one record claim be set aside for the first reader when the process is liberated from the wait queue.

B) Extended Read Mode (FCONTROL 46)

Possible open/close records must be logically deleted before the actual read to the users target area. If the reader must wait due to an empty file, then when the reader is awakened by the writer this process must be repeated until the actual read can commence. To detect an empty file the reader decrements a counting semaphore. When the semaphore goes to "0" there are no more records left, hence an empty file. If not in extended read then only data records count in the counting semaphore. Extended read causes all records (data, open, and close) to be counted.

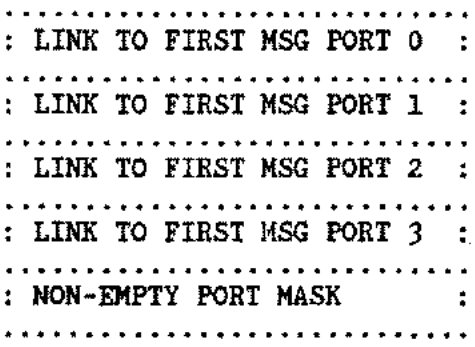

FREAD
V
EMPTY FILE
V
PUT ON
READER WAIT QUEUE
V
1. WRITER WRITES A RECORD
2. WRITER CHECKS WAIT QUEUE
3. WRITER SENDS MESSAGE TO REPLY PORT
V
1. READER CALLS IOWAIT
2. READER CHECKS REPLY PORT
3. READER TRANSFER DATA

Soft Interrupts

Soft interrupts not only uses the port data segment for queueing I/O requests but additionally shares a memory resident message facility with system processes to implement interrupts and interrupt handling. IPC uses this same facility to queue interrupts. Since this message facility is memory-resident, it is fast and can be accessed while on the ICS. There are three data structures that make up the in-core message facility, the Message Harbor Table, Primary Message Tables, and the Secondary Message Table.

MESSAGE HARBOR TABLE

The Message Harbor Table is DST %71. Each process running on the system including users and system process has an entry of 5 words in the Message Harbor Table. The entry is known as the process's "Message Harbor." The first four words are links or addresses into either the Primary Message Table or Secondary Message Table respectively where system and user process messages are queued. IPC uses word 1 in the process's Message Harbor and it heads a linked list of interrupts in the Secondary Message Table. The fifth word in a process's message harbor is a port mask of the ports being used, hence bit 1 is set when an interrupt is queued.

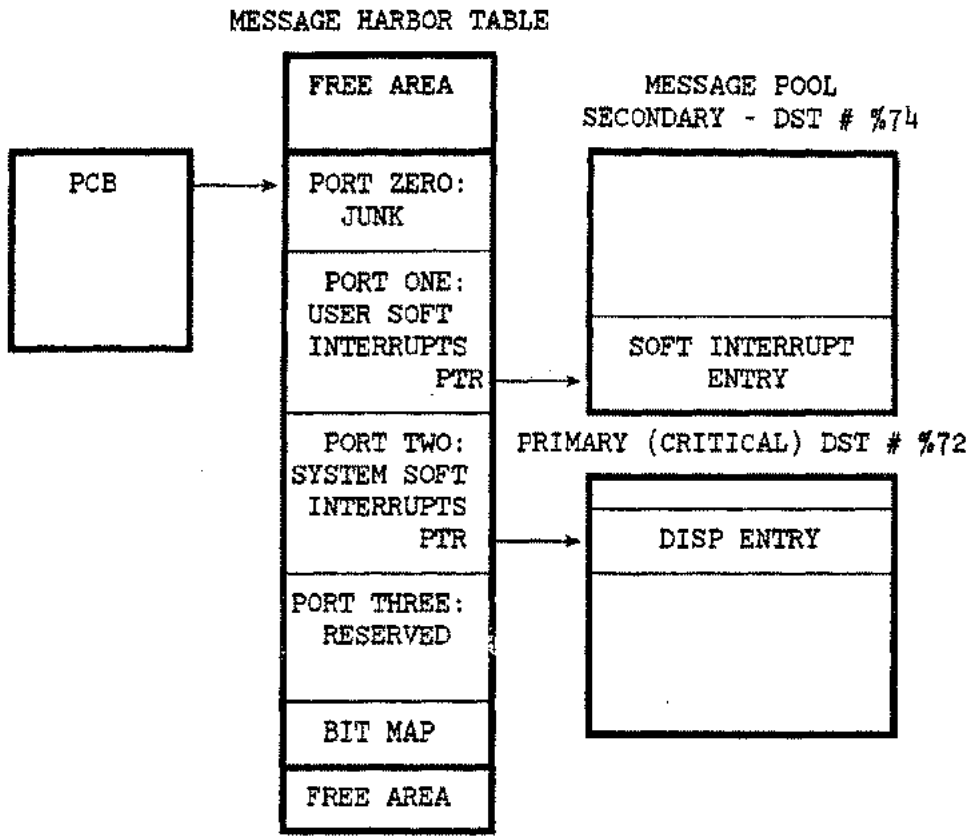


First Message Queue Link (0:1) = 1 ==> Next Message in Secondary Message Table
(1:15) = Index of next message in appropriate table

PRIMARY AND SECONDARY MESSAGE TABLES

The primary message table is DST %72 and is used to queue system messages. The secondary message table is at DST %74 and is used to queue software interrupts in an FIFO manner. The messages are posted by the process that relieves the boundary condition and PCB wait flags are set. (i.e. The writer writing to an empty message file or the reader that frees space). Later when the dispatcher is going to launch the waited process it will note the PCB wait flags. The PLABEL in the secondary message table is used to launch this process into its interrupt handler.

WORD 0: NEXT MESSAGE IN QUEUE LINK
WORD 1: SOFT INTERRUPT MESSAGE WORD ZERO
WORD 2: SOFT INTERRUPT MESSAGE WORD ONE
WORD 3: INTERRUPT HANDLER'S PLABEL
WORD 4: SOFT INTERRUPT SUBTYPE


MESSAGE HARBOR TABLE
FREE AREA
PCB
PORT ZERO:
JUNK
PORT ONE:
USER SOFT
INTERRUPTS
PTR
PORT TWO:
SYSTEM SOFT
INTERRUPTS
PTR
PORT THREE:
RESERVED
BIT MAP
FREE AREA
MESSAGE POOL
SECONDARY - DST # %74
SOFT INTERRUPT
ENTRY
PRIMARY (CRITICAL) DST # %72
DISP ENTRY

Soft Interrupt Operation

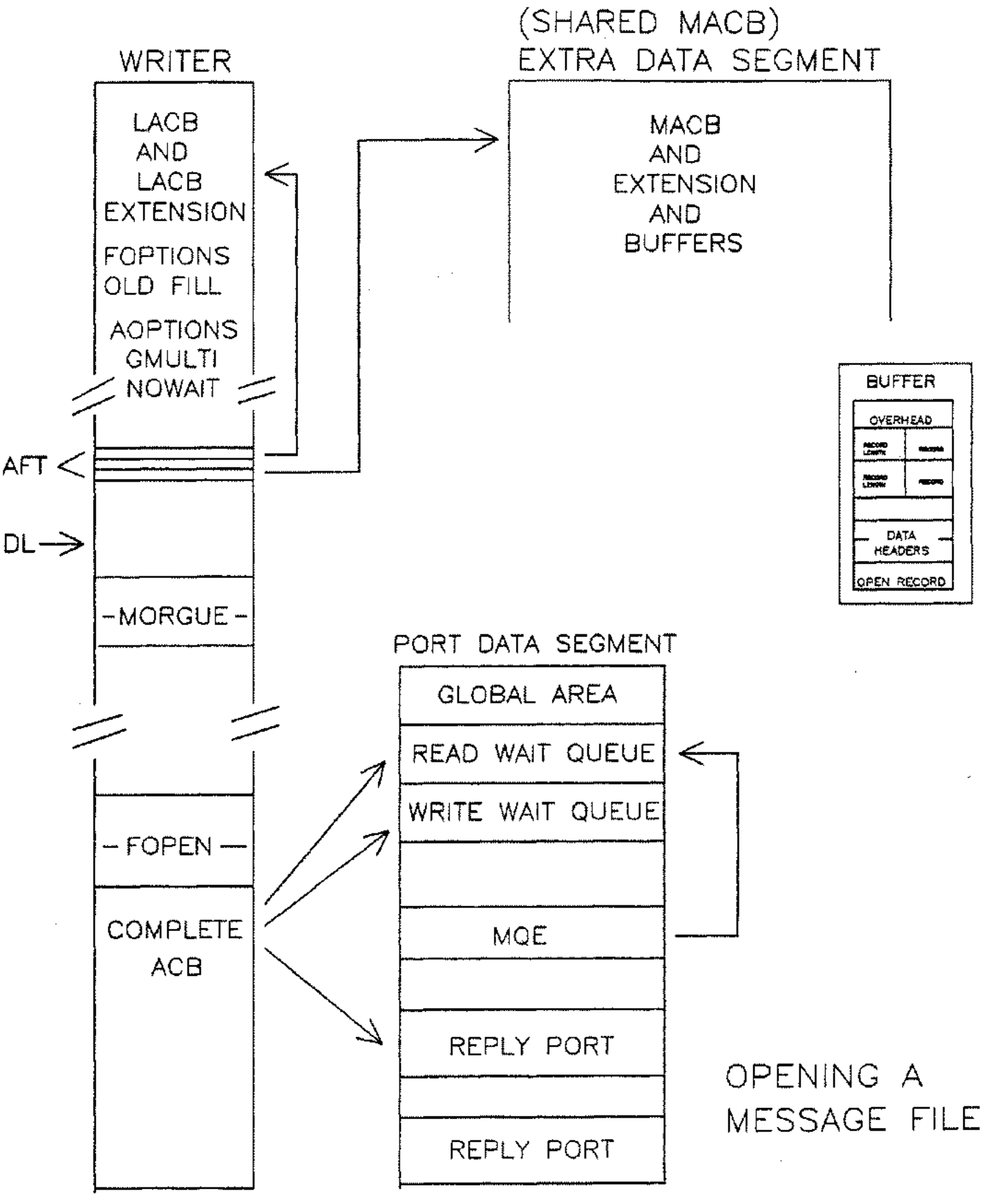
In this example of a soft interrupt operation a reader has already opened and written to a message file. The reader has enabled soft interrupts with an FINSTATE and allows extended wait with an FCONTROL 45. The reader is still executing but has queued a request on the reader wait queue. We are going to follow the writer until the read completes.

OPENING THE MESSAGE FILE

1. Open an old message file with write, Gmulti access and no-wait attributes.

2. The writer is assigned a 16 bit Writer's ID and a return port is allocated in the port data segment. This process's reply port is located via word 3 of its AFT entry and the reply port points back to the process in the right byte of word 3 - "PIN of the Owner". A writer and reader wait queue have already been established by the first opener of the message file (a reader in this case). Pointers to the reader and writer wait queues are found in the shared Message ACB.

3. A Logical Access Control Block and Logical Access Control Block Extension are created in the writer's own stack. These control blocks will contain local variables. A buffered message access control block and its extension and a File Control Block already exist in extra data segments. Again the process's Available File Table contains pointers to locate the MACB and LACB.


(SHARED MACB)
EXTRA DATA SEGMENT
WRITER
LACB
AND
LACB
EXTENSION
FOPTIONS
OLD FILL
AOPTIONS
GMULTI
NOWAIT
MACB
AND
EXTENSION
AND
BUFFERS
BUFFER
OVERHEAD
DATA
HEADERS
OPEN RECORD
AFT
DL
-MORGUE-
PORT DATA SEGMENT
GLOBAL AREA
READ WAIT QUEUE
WRITE WAIT QUEUE
-FOPEN-
COMPLETE
ACB
MQE
REPLY PORT
REPLY PORT
OPENING A
MESSAGE FILE

AN FWRITE IS INVOKED

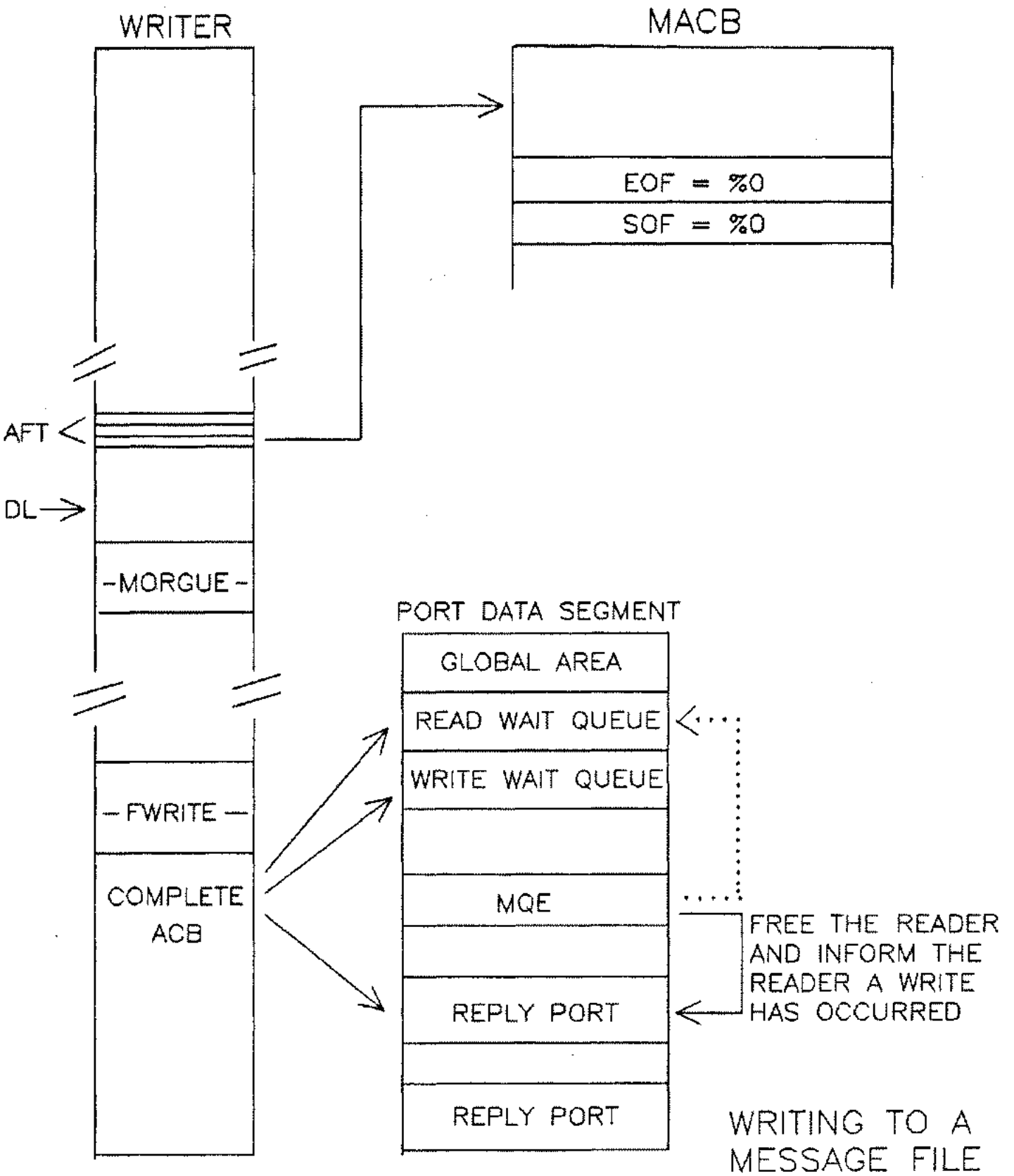
1. After an Fwrite is issued, procedure LOC'ACB locates an LACB and MACB via an AFT entry and locks these control blocks with the locking mechanism in the vector table entry for each of these control blocks. The LACB and its extension are laid on the stack.

2. LOC'ACB also places an image of these control blocks on the user's stack.

3. IPC procedure FCWRITE is invoked and the MACB and MACB extension and local parameters are placed on the stack. The complete ACB now lies on the stack.

4. FCWRITE performs the write to the target buffer. Keep in mind data and its header information is written. When the buffer becomes full the actual I/O takes place.

5. Still within the writer process, the reader wait queue is checked for waiting readers (subroutine FREEREADER).


WRITER
MACB
EOF = %0
SOF = %0
AFT
DL
-MORGUE-
PORT DATA SEGMENT
GLOBAL AREA
READ WAIT QUEUE
WRITE WAIT QUEUE
-FWRITE-
COMPLETE ACB
MQE
REPLY PORT
REPLY PORT
FREE THE READER AND INFORM THE READER A WRITE HAS OCCURRED
WRITING TO A MESSAGE FILE

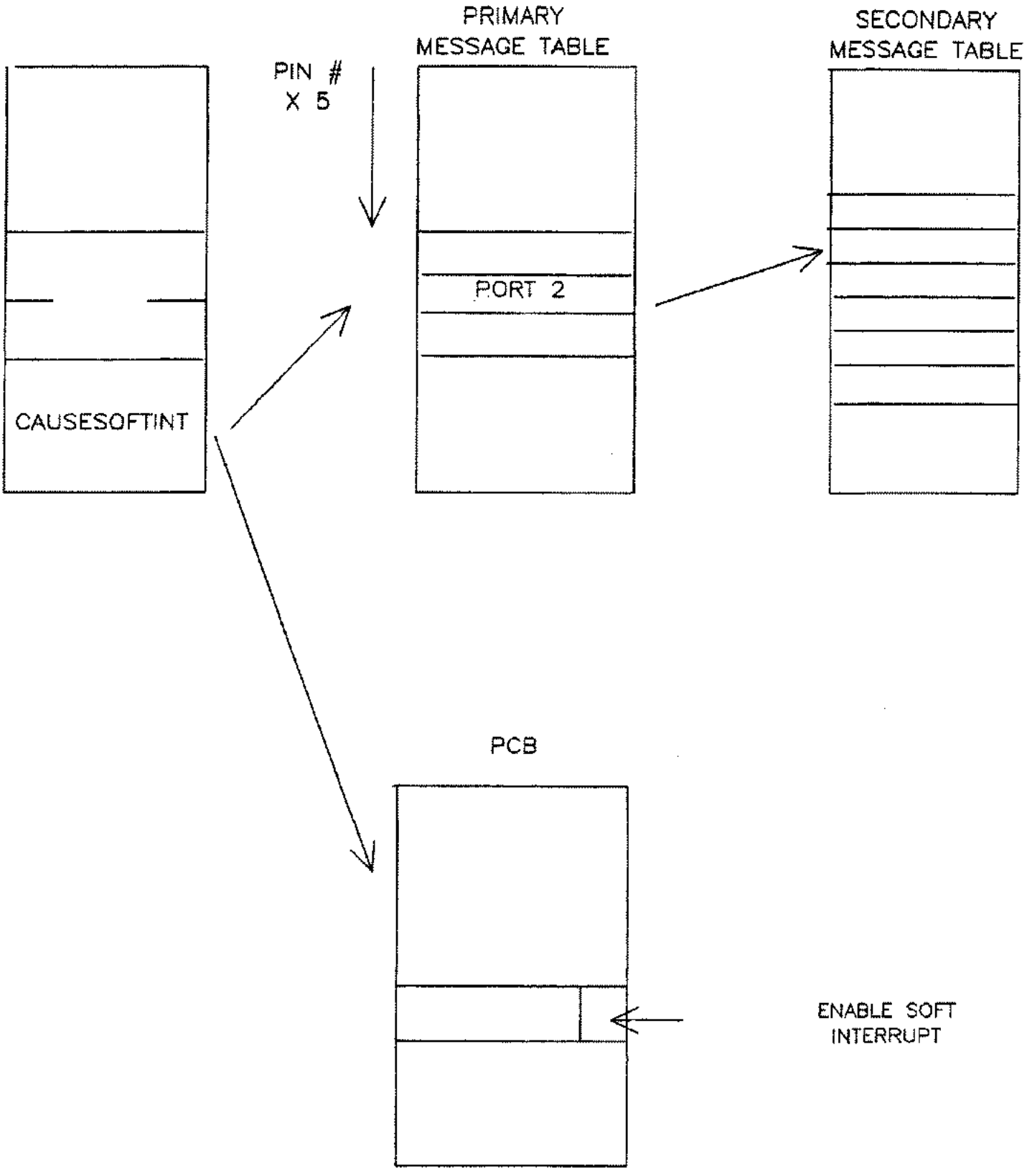
THE SOFT INTERRUPT OCCURS

The reader wait queue is examined for waiting readers. An MQE is found linked to port of the reader wait queue. This reader process has enabled soft interrupts with an FINSTATE intrinsic. The "enable wake up" bit in the reply port is set (word 2 (0:1)) and the "allow soft interrupts" bit in the reader's PCB is turned on (PCB 13 (7:1)). The writer will make the necessary arrangements for the reader to be soft interrupted.

The writer process finds the address of the reply port from the MQE that was taken from the reader wait queue. The reply port gives us the information that identifies who the reader is (PIN number in word 3 (8:8)) and also contains the PLABEL (word 7) of the reader's soft interrupt routine. With this information the writer calls procedure CAUSESOFTINT of KERNELC and passes the data as parameters to the procedure The plabel must specify code which is within the interrupted process's code address space.

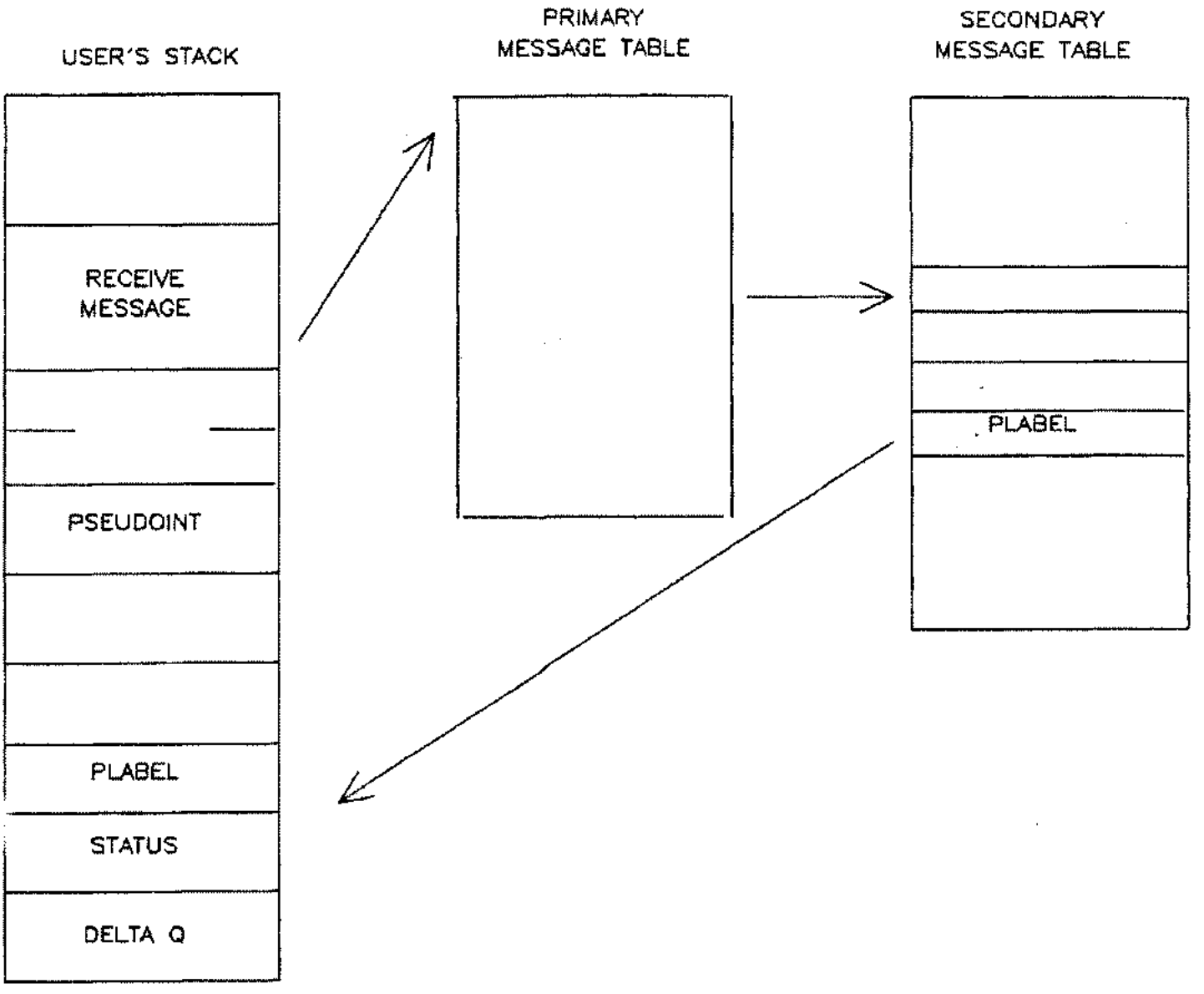
CAUSESOFTINT first checks to see that the target process is healthy (not in soft or hard kill mode). It then sends a message to the process's appropriate soft interrupt port. That port in the secondary message table DST %21 is found by indexing into the process's Message Harbor within DST %71. How far do we index - pin number x 5 (each entry is 5 words long + base of DST %71. The second word in the process's message harbor points to the beginning of the process's queue in the secondary message table where CAUSESOFTINT will queue its interrupt request.

Since the reader we are dealing with is not critical, with a SIR, or impeded, or waiting, CAUSESOFTINT will set the Soft Interrupt bit PCB 9 (9:1). Since the process is not waiting, it must already be on the dispatch queue so that no further action is needed until the dispatcher and procedure PSEUDOINT are called to service the interrupt.


PRIMARY
MESSAGE TABLE
SECONDARY
MESSAGE TABLE
PIN #
X 5
PORT 2
CAUSESOFTINT
PCB
ENABLE SOFT
INTERRUPT

The dispatcher scans the PCB of this process before it tries to launch the process. Since the soft interrupt bit is set; the dispatcher writes a stack marker on the process's stack to enter procedure PSEUDOINT. Then the process is launched. PSEUDOINT checks that the user has soft interrupts enabled (FINSTATE). If not, PSEUDOINT returns to the interrupted code and when the user reenables soft interrupts with either FINSTATE or FINTEXT, the process checks for any pending soft interrupts.

Next it is determined if the interrupt can be executed right away. That is, the process cannot be executing in system code or the user interrupt procedure must be privileged if we are interrupting privileged user code. If these two conditions are satisfied, then PSEUDOINT retrieves the message from the secondary message table, disables user soft interrupts, builds a stack marker for the interrupt handler and exits through it.


USER'S STACK
PRIMARY
MESSAGE TABLE
SECONDARY
MESSAGE TABLE
RECEIVE
MESSAGE
PSEUDOINT
PLABEL
STATUS
DELTA Q
PLABEL

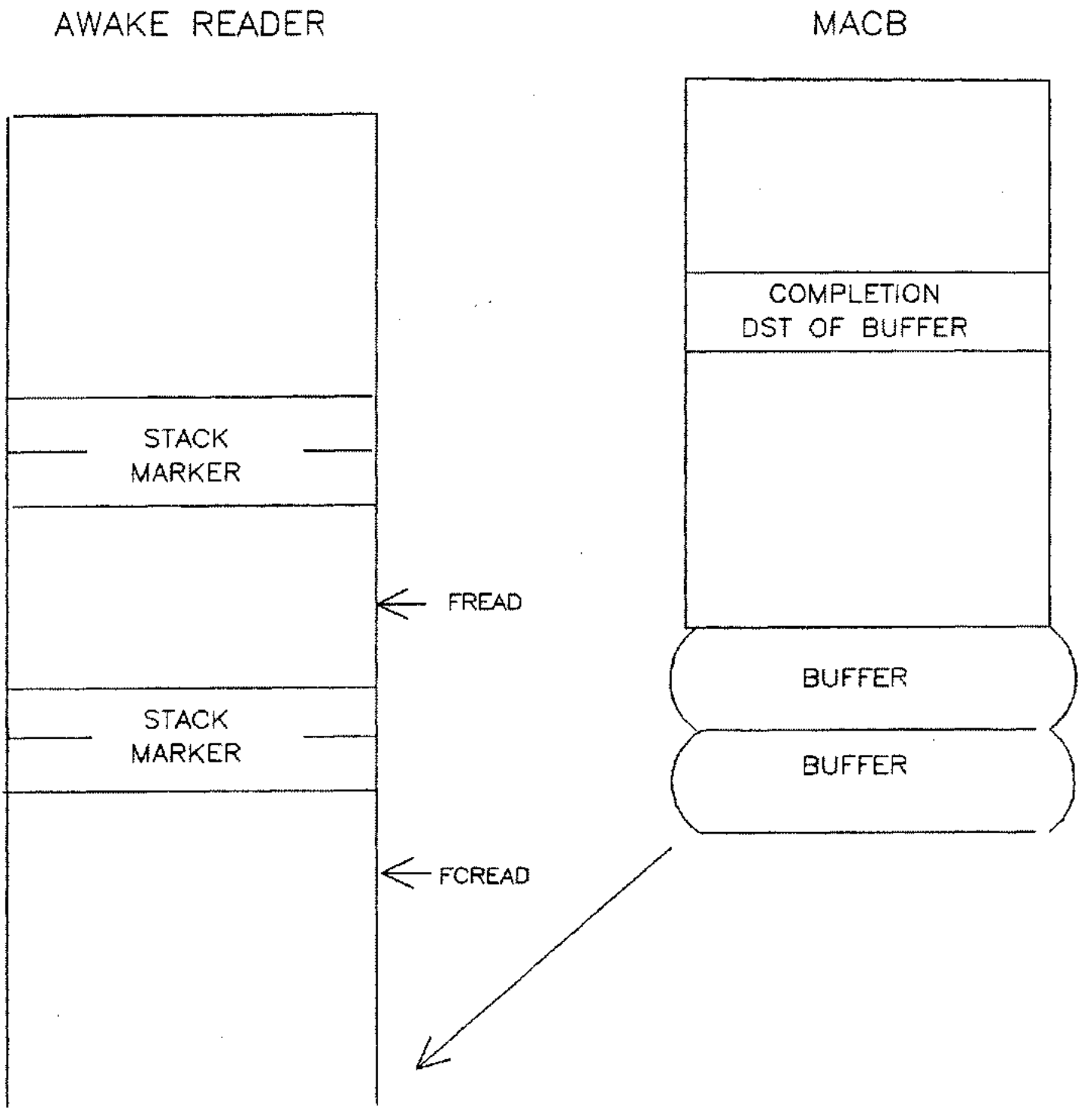
DELAY OF SOFT INTERRUPTS

If the code was not compatible the delayed soft interrupt bit in the process's PCB is set again and bit 0 of the P-register save word is set to 1 in the first eligible stack marker. This will cause a bounds violation trap to ININ, which will note a soft interrupt is pending and will invoke PSEUDOINT thereby delaying the soft interrupt.

If the process is waiting but is not critical. It may have a SIR or be impeded on some other event on which the "delay soft interrupt bit" (PCB 0(8:1)) is set. When the process is ready to launch the dispatcher will notice this bit and launch the process into PSEUDOINT so the interrupt will be served.

If the process has some combination of being critical, is with a sir or is impeded at the time the soft interrupt occurs, then the delayed soft interrupt bit and the psuedo-interrupt overflow bits (PCB 0(4:1)) are set in the PCB. The dispatcher will not have to be recalled because after inhibiting conditions are removed, RELSIR and RESETCRITICAL (called by the awakening process) both look at the PSEUDOINT overflow flag in the PCB and call PSEUDOINT if the flag is set.

Upon entering the interrupt handler routine, soft interrupts are turned off. The soft interrupt routine calls IODONTWAIT to complete the I/O, does whatever soft interrupt processing it wishes to do, and then FINEXIT will reinstate soft interrupts and normal processing resumes.


AWAKE READER
MACB
COMPLETION
DST OF BUFFER
STACK
MARKER
FREAD
BUFFER
STACK
MARKER
BUFFER
FCREAD

TIMEOUTS

A timed I/O request (initiated by issuing an FCONTROL4) upon timing out will return a condition code of CCL (error 22 is also set which can be retrieved by FCHECK). A timeout request will have created an MQE and a Timer List Entry in the port data segment. These two entries will have pointers to one another. All timer list entries in the port data segment will be linked to one another in ascending order (word 0) and the head TLE (smallest remaining interval) has an entry in the system's Timer Request List (DST %23). When the interval passes, Tick (internal interrupt procedure) updates the entries in the TRL, hence a timeout will be recognized on the ICS. Since the port data segment is not memory resident and there is no mechanism for waiting while on the ICS, timeouts of I/O requests for message files are handled by procedure OLD'TICK. TICK will set bit 12 in sysglob cell %121. This will inform UCOP that a timeout has occurred on a port data segment. OLD'TICK awakes UCOP. UCOP may wait (if necessary) for the port data segment to be brought into memory and does the necessary processing for the timeout. UCOP will send a message to the reply port and awaken the process from its timer wait.

CLOSING A MESSAGE FILE

1. All outstanding requests are cancelled.
2. For readers, any outstanding claim to a record is returned.
3. If this is the last writer, then any reader waiting process (not in extended wait mode) is satisfied with a CCG.
4. A close record is written.
5. The reply port and record buffer are deleted.
6. If this is the last writer, then spent extents are deleted.

Procedural Flow of a Soft Interrupt

INITIATING A SOFT INTERRUPT READ

USER CODE

```
FREAD(FILENUM,TARGET,TCOUNT);

  << FILENUM = filenumber returned from fopen >>
  << TARGET = buffer which will receive data >>
  << TCOUNT = number of bytes/words to read >>
```

FILEIO

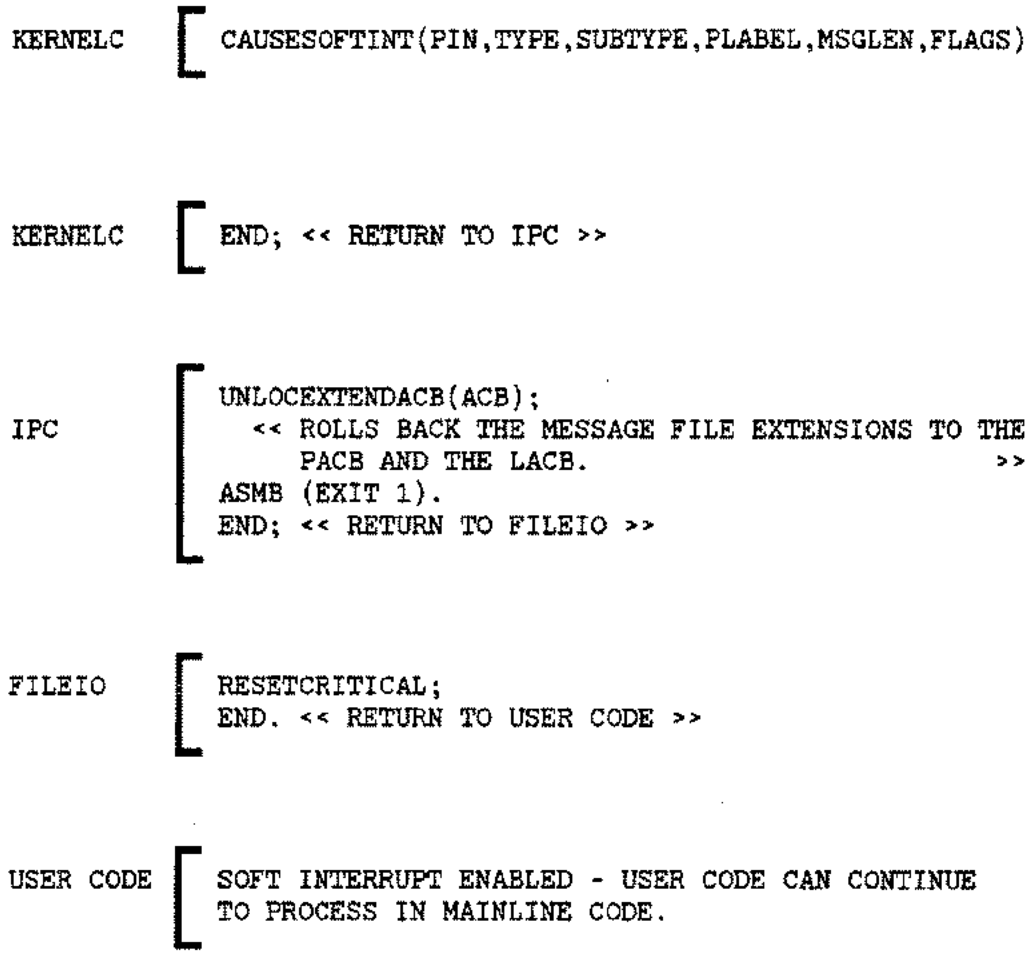
```
INTEGER PROCEDURE FREAD(FILENUM,TARGET,TCOUNT);

  ERRORON; << CHECKER - MODULE 69 >>
    << TURN TRAPS OFF >>

  SETCRITICAL; << KERNELC - MODULE 92 >>
    << SET CRITICAL IN PCB >>

  LOC'ACB(*,11,FILENUM,UMODE);
    << IF INVALID FILE NUMBER - RETURN CCL >>

  CASE FTYPE  << FTYPE 8 = A MESSGE FILE >>

    IF IOQX <> 0 THEN RETURN CCL
      << NOWAIT I/O PENDING ALREADY - AFT WORD 3 >>

    FCREAD(ACB'NOWAIT,TARGET,TCOUNT);

      << ACB'NOWAIT = AOPTIONS.(4:1) NOWAIT I/O
         MODE >>
```

IPC

```
FCREAD
```

```
FCREAD(FUNCTION,ADDR,TCOUNT);

  INITIALIZE; << COMBINE LACB AND PACB AT Q+ AREA >>

  PUTWAITQUEUE(ACB,0,0,LENGTH);
    << THIS PROCEDURE PLACES A WAIT MESSAGE INTO THE
       APPROPRIATE READERS WAIT QUEUE. THIS IS
```

```
      EITHER THE ACB RESIDENT QUEUE OR THE PORT
      READERS WAIT QUEUE.

      ALGORITHM FOR DETERMINING THE PROPER QUEUE:

      1. THE READER WAIT QUEUE MUST BE USED IF:
         A. MORE THAN ONE ACCESSOR CAN QUEUE UP.
         B. THE TIMEOUT VALUE IS SPECIFIED (ONLY
            AVAILABLE ON THE PORT WAIT QUEUE).

  NOTE: IF THE PORT WAIT QUEUE IS USED, THEN THE
        REPLY PORT MUST ALSO BE USED. THE REPLY
        PORT MUST BE USED FOR NOWAIT I/O SO THAT
IPC     IOWAIT/IODONTWAIT CAN EFFICIENTLY CHECK
        THE REQUEST'S STATUS.

FCPREPAFT(FILENUM,SOFT'INT'PEND = -2);
  << CHANGE THE MESSAGE FILE AFT ENTRY - WORD
     THREE (PORT NUMBER WORD FOR NOWAIT I/O)>>

READEXIT(ERROR);
  PUTMYCOMPLTMSG(ACB,ERRORCODE,0);
    << SEND A MESSAGE TO OWN LACB >>
    CAUSESOFTINT(0,USER'INT,FILE'SOFT'INT,
                 SOFTINTPLABEL,2,0);

      << 0 = PIN NUMBER OF TARGET PROCESS - A
         ZERO IMPLIES OWN PROCESS>>
      << USER'INT:=0 = TYPE OF SOFT INTERRUPT
         0 - INTERRUPTS PROCESSED ONLY WHEN USER
          INTERRUPTS ARE ENABLED AND THE PROCESS
          IS EXECUTING IN USER CODE (SEG %3XX).
         1 - INTERRUPTS PROCESSED IN CURRENT STATE
          UNLESS THE TARGET PROCESS IS WAITING, HAS
          SIR, IMPEDED, OR CRITICAL.
         NOTE: THE ABOVE CONDITIONS ONLY DELAY THE
         SOFT INTERRUPT. ALL TYPE ONES ARE
         SERVICED BEFORE TYPE ZEROES.
      << FILE'SOFT'INT:= 1 = SUBTYPE, TYPE OF
         PRE-PROCESSING TO BE DONE BY PSEUDOINT
         PROCEDURE. A ZERO IMPLIES NO
         PREPROCESSING .
      << SOFTINTPLABEL = THE INTERRUPT HANDLER
         PLABEL>>
      << 2 = NUMBER OF WORDS IN THE MESSAGE >>
      << 0 = LOGICAL FLAGS;
         (0:14) RESERVED FOR FUTURE USE
         (0:1) 0 = IF APPROPRIATE PLACE PROCESS
                   ON THE READY LIST
               1 = DO NOT PUT ON READY LIST
```


KERNELC
CAUSESOFTINT(PIN,TYPE,SUBTYPE,PLABEL,MSGLEN,FLAGS)
KERNELC
END; << RETURN TO IPC >>
IPC
UNLOCEXTENDACB(ACB);
<< ROLLS BACK THE MESSAGE FILE EXTENSIONS TO THE
PACB AND THE LACB. >>
ASMB (EXIT 1).
END; << RETURN TO FILEIO >>
FILEIO
RESETCRITICAL;
END. << RETURN TO USER CODE >>
USER CODE
SOFT INTERRUPT ENABLED - USER CODE CAN CONTINUE
TO PROCESS IN MAINLINE CODE.

COMPLETING THE SOFT INTERRUPT READ

USER CODE

```
FWRITE(FILENUM,TARGET,TCOUNT,CONTROL);
  << FILENUM = FILENUMBER RETURNED FROM FOPEN >>
  << TARGET = BUFFER WHICH CONTAINS THE DATA >>
  << TCOUNT = NUMBER OF BYTES/WORDS TO WRITE >>
  << CONTROL = CARRIAGE CONTROL CODE >>
```

FILEIO

```
PROCEDURE FWRITE(FILENUM,TARGET,TCOUNT,CONTROL);

  ERRORON; << CHECKER - MODULE 69 >>
    TURN TRAPS OFF;

  SETCRITICAL; << KERNELC - MODULE 92 >>
    SET CRITICAL IN PCB;

  LOC'ACB(*,9,FILENUM,UMODE);
    << IF INVALID FILE NUMBER - RETURN CCL >>

  CASE FTYPE  << FTYPE 8 = A MESSGE FILE >>

  IF IOQX <> 0 BEGIN << NO-WAIT I/O PENDING
                        RETURN CCL >>

    FCWRITE(ACB'NOWAIT,TARGET,TCOUNT);
      << ACB'NOWAIT = AOPTIONS.(4:1) NOWAIT I/O >>
```

```
PROCEDURE FCWRITE(FUNCTION,ADDR,TCOUNT);
  <<FUNCTION = 1 COMPLETE WRITE >>

  LOCEXTENDACB(ACB'LOC,AATARGETDST,ADDRESS);
    << BUILD CACB FROM LACB/PACB >>

  PUTRECORD(ACB,RECTYPE,CALLERDST,TARGET,LENGTH);
    << PUTS WRIT'ERS DATA INTO A WRITE BUFFER >>

    FREEREADER(ACB);
      << INFORMS A WAITING READER A WRITE HAS
         OCCURRED >>

    GETWAITQUEUE(ACB,ABREADQUEUE,MSG,DELETE'MSG);
      << GET WAIT QUEUE MESSAGE FROM ACB OR PORT
         WAIT QUEUE >>

    PUTCOMPLMSG(ACB,PORT,ERROR,TLOG,ID);
      << PLACES A COMPLETION MESSAGE INTO THE
         APPROPRIATE AREA ACB COMPLETION AREA OR
         THE REPLY PORT >>

    WRITEXIT;
      << RETURNS FROM THE ACCESS PROCEDURE >>

      UNLOCEXTNEDACB(ACB);
        << ROLL BACK THE LACB/PACB >>
      ASMB(EXIT 1);
```

```
UNLOC'ACB(9,0)
  << RELEASE ACB - COPY ACB TO LACB/PACB >>

IF SO <> 0 THEN AWAKE(",PORTWAKE,0); ELSE DEL

TOS:=ACB'ERROR
```

FILEIO

USER CODE

```
WRITE COMPLETE
```

PCB FIELDS USED BY SOFTWARE INTERRUPTS

APPENDIX A

Field Name	Location	Description
SPCBCritSir	PCB.(2:2)	Nonzero if the process is critical or with SIR.
SPCBIOvrFlag	PCB.(4:1)	Set when the execution of the soft interrupt must be postponed due to the process's s's being critical, with sir or impeded. Tells the appropriate kernel procedure to enter PSEUDOINT.
SPCBDelaySoft	PCB.(8:1)	Set when the execution of the soft interrupt must be delayed - critical impeded, with sir, system code, or privileged code. This bit tells the bounds violation procedure in ININ that a soft interrupt is pending. The dispatcher ignores this bit.
SPCBWaitField	PCB(4).(0:12)	Indicates which events (if any) the user process is waiting on. Does not include system events such as memory wait.
SPCBImpede	PCB(4).(12:1)	Set when the process is impeded.
SPCBPSim	PCB(8).(0:3)	Current value of process's pseudo interrupt state.
SPCBWakeSoft	PCB(8).(3:1)	Set on when the process will accept the soft interrupt even though it is waiting on other events.
SPCBSoftInt	PCB(9),(9:1)	Set when the dispatcher (and PSEUDOINT) should be aware of a pending soft interrupt.
SPCBPIFlags	PCB(9).(10:6)	Specifies which of the six, independent psuedo interrupts may be pending against the process.

SPCBAllowSoft	PCB(13).(7:1)	On implies that the process will process user soft interrupts. A zero value postpones processing of user soft interrupts (but not system soft interrupts). This bit is managed by the user through FINSTATE and FINTEXIT.

MEMORY DUMP EXAMPLE

APPENDIX B

Summary

PCB 55 has opened the file MSG1.
An FCONTROL 48 is enabled.
Pin 55 has initiated a soft interrupt read.
PCB 55 has not enabled soft interrups (FINSTATE) at the program level.

PCB 54 has opened the file and written a record.
It has triggered a soft interrupt on message file MSG1.
A soft interrupt is now pending on PCB 55.
Port 110 has a message pending soft interrupt.
The Message Harbor contains the pending soft interrupt read
that will not be processed because PCB 55 does not have interrupts
enabled at the program level (FINSTATE).

Message File: MSG1

PCB 55:

```
OPENS FILE READ, EXCLUSIVE          (12:4,8:2)
FCONTROL 48 ENABLED                 (LACB EXTENSION WORD 0)
SOFT INTERRUPT PENDING READ         (IOQX = -2 OR %177776)
FILE NUMBER = 11                    (LACB)
REPLY PORT =110                     (LACB IPC EXTENSION WORD 3)
   AWAKEN THE PROCESS               (0:1)
   GENERATE USER SOFT INTERRUPT     (1:2)
   NO MQE
PACB DST NUMBER = 124               (LACB)
WRITE WAIT QUEUE =100               (PACB)
READ WAIT QUEUE =70                 (PACB)
PCB INT ENABLED @ PROG LEVEL        (PCB (13).(7:1)
ACCESSORS LOCAL FLAGS = 000000      (LACB IPC EXTENSION WORD 1)
FILE'S LOCAL FLAGS =004217          (PACB)
```

PCB 54

```
OPEN FILE:  WRITE,EXCLUSIVE         (12:4,8:2)
REPLY PORT = 120                    (LACB IPC EXTENSION WORD 3)
WRITER ID = 1                       (LACB IPC EXTENSION WORD 4)
ACCESSOR'S LOCAL FLAGS = 004001     (LACB IPC EXTENSION WORD 2)
```

```
FILE'S GLOBAL FLAGS =004217          (PACB)
FILE NUMBER = 10                     (LACB)
PACB DST NUMBER = 124                (LACB)
WRITE WAIT QUEUE = 100               (PACB)
READ WAIT QUEUE =70                  (PACB)
```

```
HP3000 III MEMORY DUMPC.00.02 OF SYS VER C  UPDATE  01  FIX 00  DUMP TIME 10/16/81,  4:01PM
(C) HEWLETT-PACKARD CO. 1980

                                        ******    REGISTERS    ******

*********************************************************************************************************************************
*  DATA SEGMENT       *  CODE SEGMENT  *  MISCELLANEOUS  *  STATUS = 141074   *  CPX2 = 040001                          * MICRO REGS
*********************************************************************************************************************************
*                     *                *                 *                    *                                         *
*  DB BANK  =       0 *  PB  =  070120 *  X    = 177756  *  MODE        = PRIV *  RUN/HALT  = RUN     EXEC SW   = OFF  *  0 = 177534
*  DB       = 001000  *  P   =  073000 *  CIR  = 030020  *  INTERRUPTS  =  ON  *  SYS DUMP  =  ON     INC ADDR  = OFF  *  1 = 152033
*  S BANK   =       0 *  PL  =  116503 *  CPX1 = 000030  *  TRAPS       = OFF  *  COLD LOAD =  ON     DEC ADDR  = OFF  *  2 = 022351
*  DL       = 177777  *PBBANK=       0 *  MSIZE =      4 *  STACK OP    = LEFT *  LOAD REG  = OFF     INHIBIT          *  3 = 000000
*                     *                *                 *                    *                      AUTO REG  = OFF  *
*  Q        = 014514  *(P-PB)=  002660 *                 *  OVERFLOW    = OFF  *  LOAD ADDR = OFF                     *  4 = 000000
*  S        = 014576  *                *                 *  CARRY       = OFF  *  LOAD MEM  = OFF                     *  5 = 141433
*  Z        = 015512  *                *                 *  COND CODE   = CCE  *  DISP MEM  = OFF                     *  6 = 027443
*  Z BANK   =       0 *                *                 *  SEGMENT #   =  74  *  SNGL INST = OFF                     *  7 = 000000
*********************************************************************************************************************************

PAUSE INSTRUCTION IN CIR

                                        ******    FIXED LOW MEMORY

CODE SEGMENT TABLE POINTER                   007754
EXTENDED CODE SEGMENT TABLE POINTER          012440
DATA SEGMENT TABLE POINTER                   006314
PROCESS CONTROL BLOCK BASE                   013014
CURRENT PCB POINTER                          000000
INTERRUPT STACK BASE                         014514
INTERRUPT STACK LIMIT                        015512
INTERRUPT MASK                               000000
```

HP3000 III MEMORY DUMPC.00.02 OF SYS VER C UPDATE 01 FIX 08 DUMP TIME 10/16/11, 4:01PM
(C) HEWLETT-PACKARD CO. 1980

****** DST TABLE ******

SEGMENT NUMBER	SEGMENT DESCRIPTION	REFERENCE BIT	SEGMENT LENGTH	ABSOLUTE ADDRESS	BANK/ /LDEV	DISC ADDRESS	DCV	ROC	IMI	STK	MOD	FWI#	SYS	CRE$	WD	VM ALLOC
1	(CODE SEGMENT TABLE)	OFF	1400	007754	0								S	C		0
2	(DATA SEGMENT TABLE)	OFF	1440	008314	0								S	C		0
3	(PROCESS CONTROL BLOCK)	OFF	1400	013014	0								S	C		0
4	(CST EXTENSION)	OFF	1440	011354	0								S	C		0
5	(SYSTEM GLOBAL AREA)	OFF	200	001000	0								S	C		0
6	(FIXED LOW CORE)	ON	2000	000000	0								S	C		0
7	(INTERRUPT CONTROL STACK)	OFF	1100	014414	0								S	C		0
10	(SYSTEM BUFFERS)	OFF	2020	022704	0								S	C		0
11	(UCOP REQUEST QUEUE)	ON	104	134423	0								S			1
12	(PROCESS-PROCESS COMMUNICATION TABLE)	OFF	140	017223	1								S			1
13	(I/O QUEUE)	OFF	1030	015514	0								S	C		0
14	(TERMINAL BUFFERS)	OFF	2010	001704	0								S	C		0
15	(LOGICAL-PHYSICAL DEVICE TABLE)	ON	170	037510	0								S	C		0
16	(LOGICAL DEVICE AND CLASS TABLE)	ON	1204	061423	3								S			2
17	(DRIVER LINKAGE TABLE)	OFF	40	000134	0								S	C		0
20	(I/O RESOURCE TABLES)	OFF	24	000174	0								S	C		0
21	(DISK FREE SPACE)	OFF	20000	040023	1								S			21
22	(LOADER SEGMENT TABLE)	ON	3144	174423	1								S			14
23	(TIMER REQUEST LIST)	OFF	204	000450	0								S	C		0
24	(DIRECTORY)	ON	2000	007623	1								S			3
25	(DIRECTORY SPACE)	ON	600	151423	0								S			1
26	(RIN TABLE)	OFF	1304	036423	1								S			0
27	(SWAPTABLE)	OFF	7250	024724	0								S	C		0
30	(JOB PROCESS COUNT)	ON	20	000654	0								S	C		0
31	(JOB MASTER TABLE)	ON	200	174023	1								S			14
32	(TAPE LABEL TABLE)	OFF	1750		1	17074	D						S			2
33	(LOG TABLE)	OFF	170		1	3148	D						S			0
34	(REPLY INFORMATION TABLE)	OFF	2000		1	18304	D						S			3
35	(VOLUME TABLE)	ON	34	063423	3								S			1
36	(BREAKPOINT TABLE)	OFF	674		1	17170	D						S			1
37	(LOG BUFFER 1)	OFF	400		1	17174	D						S			1
40	(LOG BUFFER 2)	OFF	400		1	17200	D						S			1
41	(LOG ID TABLE)	OFF	150		1	3144	D						S			0
42	(ASSOCIATION TABLE)	OFF	640		1	17120	D						S			1
43	(CST BLOCK)	OFF	44	000220	0								S	C		0
44	(JOB CUTOFF TABLE)	OFF	74	000674	0								S	C		0
45	(SYSTEM JIT)	OFF	100	082423	3								S			1
46	(SPECIAL REQUEST TABLE)	OFF	144	034174	0								S	C		0
47	(VIRTUAL DISK SPACE TABLE)	OFF	164	035320	0								S	C		0
51	(ARSBM TABLE)	OFF	44	000404	0								S	C		0
52	(ILT)	OFF	1020	021654	0								S	C		0
53	(SIR TABLE)	OFF	170	037700	0								S	C		0
54	(FILE MULTI-ACCESS VECTOR)	ON	200	177423	3								S			2
55	(INPUT DEVICE DIRECTORY)	ON	400	114223	1								S			40
56	(OUTPUT DEVICE DIRECTORY)	OFF	400	062023	3								S			40
57	(WELCOME MESSAGE #1)	OFF	1750		1	17036	D						S			2

HP3000 III MEMORY DUMPC.00.02 OF SYS VER C UPDATE 01 FIX 08 DUMP TIME 10/18/81, 4:01PM
(C) HEWLETT-PACKARD CO. 1980

****** DST TABLE ******

SEGMENT NUMBER	SEGMENT DESCRIPTION	REFERENCE BIT	SEGMENT LENGTH	ABSOLUTE ADDRESS	BANK/ /LDEV	DISC ADDRESS	DCV	ROC	IHI	STK	MOD	FWIP	SYS	CRES	WD	VM ALLOC
60	(WELCOME MESSAGE #2)	OFF	1750		1	17044	D						S			2
61	(CS SYSTEM SEGMENT)	OFF	1220		1	16170	D						S			2
62	(JOB-PROCESS CROSS REFERENCE)	ON	60	177623	2								S			1
63	(SYSTEM JDT)	OFF	34		1	16340	D						S			1
64	(COMMAND INTERPRETER LOG-ON DST)	OFF	1000		1	17054	D						S			4
65	(MOUNTED VOLUME TAB.)	OFF	520		1	17124	D						S			1
66	(PRI. VOL. USER TABLE)	OFF	200		1	17130	D						S			10
67	(AVAILABLE REGION LIST)	OFF	2004	035504	0								S	C		0
70	(DISC REQUEST TABLE)	OFF	3120	016544	0								S	C		0
71	(MSG HBR TABLE)	OFF	360	034340	0								S	C		0
72	(PRIMARY MSG TABLE)	OFF	200	034720	0								S	C		0
73	(MEASUREMENT INFO TABLE)	OFF	120	000264	0								S	C		0
74	(SECONDARY MSG TABLE)	OFF	200	035120	0								S	C		0
75		ON	3244	047423	3					S			S			7
76		OFF	3674		1	16134	D			S			S			7
77		OFF	3604		1	17204	D			S			S			7
100		OFF	13144		1	17240	D			S			S			16
101		OFF	2554		1	17330	D			S			S			6
102		ON	2310	175423	0					S			S			13
103		OFF	2280		1	17434	D			S			S			6
104		OFF	4770		1	17464	D			S			S			13
105		ON	6704	050223	2					S			S			43
106		OFF	4720		1	17754	D			S			S			17
107		OFF	4010		1	20150	D									5
110		OFF	10374		1	20514	D			S						37
111		ON	204	075023	3											1
112		ON	1640	172623	0											12
113		OFF	1404		1	20264	D									2
114		OFF	5324		1	20404	D			S						22
115		ON	104	177623	1											1
116		ON	300	075423	3											5
117		ON	100	165223	3											1
120		ON	2520	127223	2											3
121		ON	500	076023	3											1
122		OFF	2520		1	20324	D									3
123		OFF	404	174623	0		D	R								1
124		ON	610	132623	3											1
125		ON	500	151623	2											1
126		ON	610	012023	1											1
127		OFF	2520		1	20344	D									3
130		ON	460	030023	2											1
131		ON	7640	141623	2											10
132		ON	14164	067423	1					S						54
133		ON	11764	165423	2					S						54
134		ON	610	175623	3											1
135		ON	610	061423	3											1

Dump Example

```
HP3000 III MEMORY DUMPC.GG.02 OF SYS VER C  UPDATE  01  FIX 08  DUMP TIME 10/16/81,  4:01PM
(C) HEWLETT-PACKARD CO. 1980

                                  ******   DST TABLE   ******

SEGMENT                                   REFERENCE  SEGMENT   ABSOLUTE   BANK/   DISC      D R I S H F S C W   VN
NUMBER   SEGMENT DESCRIPTION              BIT        LENGTH    ADDRESS    /LDEV   ADDRESS   C O M T O W Y R D   ALLOC
------   ------------------------         ---        ------    --------   -----   -------   V C I K D P S S     -----
    137                                   ON          644      173022       1                                      71
```

PCB

* ONLY ONE PROCESS RUNNING

* CI SON WAIT PCB 16

* PCB 54 IN PAUSE

* PCB 55 IN PAUSE

HP3000 III MEMORY DUMPC.00.02 OF SYS VER C UPDATE 01 FIX 09 DUMP TIME 10/16/81, 4:01PM

****** PROCESS CONTROL BLOCK (1ST HALF) ******

W A I T S T A T E

DATA -SEGMENTS-- / -FAMILY TREE-- / ------WAKEMASK------

PIN	XDS	ADS	STK	OVALC	FTHR PIN	SON PIN	BRO PIN	QA	M	RG	RL	MA	BIO	IO	UCOP	JUNK	TIMER	MSG	SON	FATHR	IMP	SIR	TMOUT	MEM
1			108			2		S								J								
2		A	75		1		3									J								
3			76		1		4							I			T	M						
4			77		1		5	F								J								
5			100		1		6	F													I			
6			101		1		7	F								J								
7	11		102		1	18	10	F								J								
10			103		1		11									J								M
11			104		1		12							I		J								
12			105		1		14									J								
14			114		1			S											S	F				
18			110		7	54		S											S					
54			132		18	55		F									T							
55			133		54			F									T							

------EVENTFLAGS------ / -PSEUDO INTERRUPTS-- / --MISC---

PIN	M	RG	RL	MA	BIO	IO	UCOP	JUNK	TIMER	MSG	SON	FATHR	IMP	SIR	TMOUT	MEM	PSIM	HK	SK	ST	HB	CY	BK	RITBK	PIOVR	PTYPE	CRIT	HSIR
1																	NORM									SYST		
2																	NORM									SYST		
3																	NORM									SYST		
4																	NORM									SYST		
5																	NORM									SYST	C	
6																	NORM									SYST		
7																	NORM									SYSTU		
10																	NORM									SYST		
11																	NORM									SYST		
12																	NORM									SYST		
14																	NORM									SYST		
18																	NORM									DRAIN		
54																	NORM									USONM		
55																	NORM									USER		

HP3000 III MEMORY DUMPC.00.02 OF SYS VER C UPDATE 01 FIX 09 DUMP TIME 10/18/81, 4:01PM
(C) HEWLETT-PACKARD CO. 1980

****** PROCESS CONTROL BLOCK (2ND HALF) ******

----- SCHEDULING INFORMATION -----																						---RESOURCES---					LIFE/DEATH			---------- MISCELLANEOUS ----------						
PIN	NQPIN	PQPIN	DISPQ	LQ	CQ	DQ	EQ	INTER	CORE	PRI	HIPRI	USEDQ	TRW	SW	LW	HP	PC	IPEXP	HSPRI	SAR	SOV	CRIT	HSIR	PREV IMPD PIN	NEXT IMPD PIN	SC	LIVE	DEAD	FAC	BMS	PPC	PCST	PBXPTR	SLLPTR	BPT LNK	SYSTEM PROC NAME
1				L						61																	L			SNF	NUL		10	24063		PROGEN
2				L						62																	L			SNF	NUL	CST 41		23731		SYSIO
3				L						170																	L			SNF	NUL	CST 37		23743		IOMESS
4				L						62																	L			SNF	NUL		1	23755		LOG
5				L						175												C					L			SNF	NUL		2	23767		MEMLOG
6				L						175																	L			SNF	NUL		3	24001		
7				L						175																	L			SNF	NUL		4	24013		UCOP
10				L						12										S							L			SNF	NUL		5	24025		PFAIL
11				L						175																	L			SNF	NUL		6	24037		DEVREC
12				L						216																	L			SNF	NUL		7	24051		LOAD
14				L						230																	L			SNF	NUL			24551		
16					C			I		230					L											S	L			SNF	NUL			25023		
54					C			I		230					L											S	L		F	SNF	NUL		15	31171		
55					C			I		230					L											S	L			SNF	NUL		27	31532		

60 ENTRYS
41 UNASSIGNED ENTRYS
17 ASSIGNED ENTRYS

```
HP2000 III MEMORY DUMPC.00.02 OF SYS VER C  UPDATE  01  FIX 00  DUMP TIME 10/16/81,  4:01PM
(C) HEWLETT-PACKARD CO. 1980

                    ******    PCBX AND STACK MARKERS FOR DST 132  (PCB 54 )    ******

SEG REL  SEG REL    JMAT      JPCNT     JOB INPUT   JOB OUTPUT  JDT DST   JIT DST
DL       DB         INDEX     INDEX     LOG DEV #   LOG DEV #   INDEX     INDEX     JOB TYPE   DUPLICAT  INTERACT  INIT Q   JCUT INDEX
001044   001200       1         2          20          20       126110      115       052        YES       YES      000302       0

ADDRESS    BANK       X        DELTA P   STATUS    DELTA Q     SEGMENT

077236       1      177756     020003    101074    000012       74
077224       1      000021     010630    141074    000014       74
077210       1      006337     001725    142430    000020       30
077170       1      000041     004074    062304    000007      304   USER SEGMENT
077161       1      000041     000207    062304    000004      304   USER SEGMENT
077155       1      000000     000053    062304    000024      304   USER SEGMENT
077131       1      000000     000000    140041    000004       41

                    ******    PCBX AND STACK MARKERS FOR DST 133  (PCB 55 )    ******

SEG REL  SEG REL    JMAT      JPCNT     JOB INPUT   JOB OUTPUT  JDT DST   JIT DST
DL       DB         INDEX     INDEX     LOG DEV #   LOG DEV #   INDEX     INDEX     JOB TYPE   DUPLICAT  INTERACT  INIT Q   JCUT INDEX
000044   001000       1         2          20          20       126118      115       052        YES       YES      000302       0

ADDRESS    BANK       X        DELTA P   STATUS    DELTA Q     SEGMENT

175036       2      177756     020003    101074    000012       74
175024       2      170520     010630    141074    000014       74
175010       2      006337     001725    142430    000020       30
174770       2      000041     004074    062304    000007      304   USER SEGMENT
174761       2      000041     000207    062304    000004      304   USER SEGMENT
174755       2      000000     000053    062304    000024      304   USER SEGMENT
174731       2      000000     000000    140041    000004       41
```

```
HP3000 III MEMORY DUMPC.00.02 OF SYS VER C  UPDATE  01  FIX 00  DUMP TIME 10/16/81,  4:01PM
(C) HEWLETT-PACKARD CO. 1980

                                        ******   SIR TABLE    ******

NO LOCKED SIRS

                                        ******   MONITOR TABLE    ******

LOCATION    PIN    EVENT                                    PIN    EVENT                                    PIN    EVENT
40735         0 QUIESCE   031171 000200 122236               0 QUIESCE   031171 004000 122236               0 QUIESCE   031171 004000 122234
40721         0 QUIESCE   031171 004000 122234               0 SIODMEXIT 001000 060000 133201               0 SPECIALRQ 000121 000003 000000
40705         0 INTERRUPT 001132 000000 043177               0 QUIESCE   031171 002000 122234              54 SIODMEXIT 001500 060413 003151
40471        54 SPECIALRQ 000121 141100 000001              54 SIODMEXIT 001000 060000 003133               0 SIODMEXIT 001000 060000 133126
40455         0 SPECIALRQ 000132 000023 000000               0 INTERRUPT 001132 000000 043125               0 QUIESCE   031171 004000 122234
40441        54 SIODMEXIT 001420 060413 001052              54 SPECIALRQ 000132 001520 000001               0 SIODMEXIT 001000 060000 133032
40425         0 SIODONE   000126 016024 000000               0 INTERRUPT 001132 000000 043030               0 SWAPIN    000054 100000 000000
40411         0 SIODMEXIT 001260 060413 002770               0 SEGIO     000128 016024 000001               0 DEALLOCM  000000 000001 013023
40575         0 ALLOCMEM  000004 000001 012021               0 FETCHSEG  000126 000054 000003               0 QUIESCE   031171 000001 122234
40561        54 QONSEG    000128 031171 000017               0 SIODMEXIT 001000 060000 132747               0 SPECIALRQ 000132 000023 000000
40545         0 INTERRUPT 001132 000000 042745               0 QUIESCE   031171 004000 122234              54 SIODMEXIT 001300 060413 002733
40531        54 SPECIALRQ 000132 005100 000001               0 SIODMEXIT 001000 060000 132721               0 SPECIALRQ 000132 000023 000000
40515         0 INTERRUPT 001132 000000 042717               0 QUIESCE   031171 004000 122234              54 SIODMEXIT 001340 060413 132863
40501        54 SIODMEXIT 001340 060413 002882              54 SPECIALRQ 000132 043480 000001               0 QUIESCE   031171 004000 122234
40469         0 QUIESCE   031171 004000 122234               0 DEALLOCM  000000 000001 107023               0 DEALLOCM  000000 000001 104023
40451         0 QUIESCE   031171 004000 122232               0 SIODMEXIT 001000 060000 132732               0 SPECIALRQ 000121 000003 000000
40435         0 INTERRUPT 001132 000000 042330               0 DEALLOCM  000000 000001 107023               0 DEALLOCM  000000 000001 104023
40421         0 QUIESCE   031171 002000 122232              54 SIODMEXIT 001040 060413 002272              54 SPECIALRQ 000121 177780 000001
40405         0 DEALLOCM  000000 000001 107023               0 DEALLOCM  000000 000001 104023               0 QUIESCE   031171 004000 122232
40371         0 DEALLOCM  000000 000001 107023               0 DEALLOCM  000000 000001 104023               0 QUIESCE   031171 004000 122232
40355        54 SIODMEXIT 001000 060000 001852               0 SIODMEXIT 001000 060000 131447               0 SPECIALRQ 000132 000023 000000
40341         0 INTERRUPT 001132 000000 041645               0 DEALLOCM  000000 000001 107023               0 DEALLOCM  000000 000001 104023
40325         0 GOARBAGE  000002 000200 000028               0 ALLOCMEM  000028 000000 151423               0 QUIESCE   031171 004000 122232
40311        54 SIODMEXIT 001200 060413 001813              54 SPECIALRQ 000132 001520 000001               0 SIODMEXIT 001000 060000 131573
40275         0 SIODONE   000132 016144 000000               0 INTERRUPT 001134 000000 041571               0 SIODMEXIT 001400 060413 131546
40261         0 SEGIO     000122 016144 000001               0 SIODONE   000132 015724 100000               0 INTERRUPT 001134 000000 041543
40245         0 SWAPIN    000054 100000 000000               0 DEALLOCM  000000 000001 104023               0 ALLOCMEM  000002 000001 067423
40231         0 DEALLOCM  000002 000001 077023               0 MAKEDC    040060 000000 000000               0 DEALLOCM  000002 000001 071023
40215         0 MAKEDC    104402 000000 000000               0 FETCHSEG  000132 000054 000003               0 SIODMEXIT 001160 060413 001503
40201         0 SEGIO     000132 015724 100001               0 DEALLOCM  000000 000002 013023               0 QUIESCE   031171 000001 122232
40165        54 SPECIALRQ 000132 000011 000200               0 SIODMEXIT 001000 060000 131472               0 SIODONE   000134 015704 000000
40151         0 INTERRUPT 001132 000000 041471               0 SWAPIN    000054 100000 000000               0 SIODMEXIT 001140 060413 001472
40139         0 SEGIO     000134 015706 000001               0 DEALLOCM  000000 000003 178023               0 ALLOCMEM  000004 000003 175823
40121         0 FETCHSEG  000134 000054 000003               0 QUIESCE   031171 000001 122232              54 QONSEG    000176 031171 000017
40105         0 SIODMEXIT 001000 060000 131412               0 SPECIALRQ 000132 000022 000000               0 INTERRUPT 001132 000000 041410
42085         0 QUIESCE   031171 004000 122232              54 SIODMEXIT 001220 060413 001278              54 SPECIALRQ 000132 005100 000001
42051         0 SIODMEXIT 001000 060000 131384               0 SPECIALRQ 000132 000023 000000               0 INTERRUPT 001132 000000 041382
42035         0 QUIESCE   031171 004000 122232              54 SIODMEXIT 001240 060413 131330              54 SIODMEXIT 001240 060413 001327
42021        54 SPECIALRQ 000132 043480 000001               0 QUIESCE   031171 004000 122230               0 SIODMEXIT 001000 060000 131201
42005         0 SPECIALRQ 000121 000003 000000               0 INTERRUPT 001132 000000 041177               0 QUIESCE   031171 002000 122230
41771        54 SIODMEXIT 001480 060413 001142              54 SPECIALRQ 000121 177780 000001               0 QUIESCE   031171 004000 122230
41755        54 SIODMEXIT 001060 060000 131103              54 SPECIALRQ 000124 000023 000000              54 INTERRUPT 001132 000000 041102
41741        54 SIODMEXIT 001120 060413 001085              54 SPECIALRQ 000124 000200 000001               0 QUIESCE   031171 004000 122230
41725        54 SIODMEXIT 001000 060000 000755               0 SIODMEXIT 001000 060000 130750               0 SPECIALRQ 000132 000023 000000
```

Dump Example

```
HP3000 III MEMORY DUMPC.09.02 OF SYS VER C  UPDATE  01  FIX 09  DUMP TIME 10/16/81,  4:01PM
(C) HEWLETT-PACKARD CO. 1980

41716      0 INTERRUPT 001132 000000 040747      0 QUIESCE   031171 004000 122230     54 SIODMEXIT 001100 080413 000744
41675     54 SPECIALRQ 000132 141040 000001      0 SIODMEXIT 001000 080000 130722      0 SPECIALRQ 000132 000023 000000
41661      0 INTERRUPT 001132 000000 040720      0 QUIESCE   031171 004000 122230     54 SIODMEXIT 001440 080413 000705
41645     54 SPECIALRQ 000132 064560 000001      0 SIODMEXIT 001000 080000 130674      0 SPECIALRQ 000132 000023 000000
41631      0 INTERRUPT 001132 000000 040672      0 QUIESCE   031171 004000 122230     54 SIODMEXIT 001100 080413 130641
41615     54 SIODMEXIT 001100 080413 000842     54 SPECIALRQ 000132 041460 000001      0 QUIESCE   031171 004000 122230
41601      0 SIODMEXIT 001000 080000 130484      0 SPECIALRQ 000121 000003 000000      0 INTERRUPT 001132 000000 040682
41585      0 QUIESCE   031171 002000 122230     54 SIODMEXIT 001040 080413 000428     54 SPECIALRQ 000121 177700 000001
41551      0 QUIESCE   031171 004000 122230      0 QUIESCE   024013 000400 140175      0 SIODMEXIT 001000 080000 131124
41535      0 SIODONE   000031 015564 000000      0 INTERRUPT 001132 000000 021132      0 SWAPIN    000007 100000 000000
41521      0 SIODMEXIT 001020 080413 001105      0 SEGIO     000031 015564 000001      0 ALLOCMEM  000002 000001 174023
41505      0 FETCHSEG  000031 000007 000001      0 QUIESCE   024013 000001 140175      7 QQWSEG    000031 024013 000023
41471      0 SIODMEXIT 001000 080000 131077      0 SIODONE   102001 020124 000000      0 INTERRUPT 001132 000000 021075
41455      0 SWAPIN    000007 100000 000000      0 SIODMEXIT 001360 080413 001031      0 SEGIO     102001 020124 000001
41441      0 ALLOCMEM  000012 000003 142623      0 FETCHSEG  102001 000007 000003      0 QUIESCE   024013 000001 140175
41425      7 QQWSEG    102001 024013 000030      0 SWAPIN    000007 140000 000000      0 DEALLOCM  000000 000000 174823
41411      0 FETCHSEG  000102 000007 000001      0 FETCHSEG  000011 000007 000001      0 QUIESCE   031532 002000 122234
41375      0 SIODMEXIT 001000 080000 125405      0 SPECIALRQ 000125 000003 000000      0 INTERRUPT 001132 000000 013403
41361      0 QUIESCE   031532 002000 122234     55 SIODMEXIT 001030 080413 005371     55 SPECIALRQ 000125 000000 000001
41345      0 SIODMEXIT 001000 080000 135202      0 SPECIALRQ 000125 000003 000000      0 INTERRUPT 001132 000000 013300
41331      0 QUIESCE   031532 002000 122234     55 SIODMEXIT 001760 080413 005244     55 SPECIALRQ 000125 141100 000001
41315     55 SIODMEXIT 001000 080000 005227      0 SIODMEXIT 001000 080000 135223      0 SPECIALRQ 000131 000023 000000
41301      0 INTERRUPT 001132 000000 015221      0 QUIESCE   031532 004000 122234     55 SIODMEXIT 001740 080413 005173
41265     55 SPECIALRQ 000133 001520 000001      0 SIODMEXIT 001000 080000 135152      0 SIODONE   000124 020824 000000
41251      0 INTERRUPT 001132 000000 015150      0 SWAPIN    000055 100000 000000      0 SIODMEXIT 001000 080413 005113
41235      0 SEGIO     000124 020824 000001      0 DEALLOCM  000000 000003 133823      0 ALLOCMEM  000004 000063 132823
41221      0 FETCHSEG  000124 000055 000003      0 QUIESCE   031532 000001 122234     55 QQWSEG   000124 031532 000017
41205      0 SIODMEXIT 001000 080000 135072      0 SPECIALRQ 000133 000023 000000      0 INTERRUPT 001132 000000 015070
41171      0 QUIESCE   031532 004000 122234     55 SIODMEXIT 001720 080413 005056     55 SPECIALRQ 000133 005100 000001
41155      0 SIODMEXIT 001000 080000 135044      0 SPECIALRQ 000133 000023 000000      0 INTERRUPT 001132 000000 013042
41141      0 QUIESCE   031532 004000 122234     55 SIODMEXIT 001000 080413 005004     55 SPECIALRQ 000133 043460 000001
41125      0 QUIESCE   031171 000200 122230     54 SIODMEXIT 001000 080000 134715     54 SPECIALRQ 000125 000003 000000
41111     54 INTERRUPT 001132 000000 014713      0 QUIESCE   031532 002000 122232     55 SIODMEXIT 001880 080413 004857
41075     55 SPECIALRQ 000125 000000 000001      0 QUIESCE   031171 004000 122230      0 QUIESCE   031532 000000 122232
41061     55 SIODMEXIT 001000 080000 004524      0 SIODMEXIT 001000 080000 134520      0 SPECIALRQ 000133 000023 000000
41045      0 INTERRUPT 001132 000000 014516      0 QUIESCE   031532 004000 122232     55 SIODMEXIT 001020 080413 004451
41031     55 SPECIALRQ 000133 001520 000001      0 SIODMEXIT 001000 080000 134431      0 SIODONE   000137 020444 000000
41015      0 INTERRUPT 001132 000000 014427      0 SWAPIN    000055 100000 000000      0 SIODMEXIT 001700 080413 004430
41001      0 SEGIO     000137 020444 000001      0 DEALLOCM  000000 000001 174023      0 ALLOCMEM  000004 000001 173023
40765      0 FETCHSEG  000137 000055 000003      0 QUIESCE   031532 000001 122232     55 QQWSEG   000137 031532 000011
40751      0 SIODMEXIT 001000 080000 134380      0 SIODONE   000135 020404 000000      0 INTERRUPT 001132 000000 014358
```

HP3000 III MEMORY DUMPC.00.02 OF SYS VER C UPDATE 01 FIX 09 DUMP TIME 10/16/81, 4:01PM

****** TIMER REQUEST LIST ******

FREE LIST POINTER	000034
NUMBER OF ENTRYS	000040
ENTRY SIZE	4
TRACE WORD	010020
QUANTUM/100MS	000000
POINTER TO MOST ACTIVE REQ	000024
DATE	10/16/81, 4:01PM

ENTRY	REQUEST STATUS	TYPE OF REQUEST	POINTER TO NEXT REQUEST	REQUEST POINTER	TIME TO SERVICE REQ IN FRONT (SEC/10)
14	ACTIVE	DELAY	0	PCBB IX = 000120	15532
20	ACTIVE	202TURNAROUND	14	DITP = 001300	6
24	ACTIVE	202TURNAROUND	30	DITP = 001320	9483
30	ACTIVE	HANGUP	20	DITP = 000001	107
34	INACTIVE	HANGUP	40	DITP = 000000	0
40	INACTIVE	HANGUP	44	DITP = 000000	0
44	INACTIVE	HANGUP	50	DITP = 000000	0
50	INACTIVE	HANGUP	54	DITP = 000000	0
54	INACTIVE	HANGUP	60	DITP = 000000	0
60	INACTIVE	HANGUP	64	DITP = 000000	0
64	INACTIVE	HANGUP	70	DITP = 000000	0
70	INACTIVE	HANGUP	74	DITP = 000000	0
74	INACTIVE	HANGUP	100	DITP = 000000	0
100	INACTIVE	HANGUP	104	DITP = 000000	0
104	INACTIVE	HANGUP	110	DITP = 000000	0
110	INACTIVE	HANGUP	114	DITP = 000000	0
114	INACTIVE	HANGUP	120	DITP = 000000	0
120	INACTIVE	HANGUP	124	DITP = 000000	0
124	INACTIVE	HANGUP	130	DITP = 000000	0
130	INACTIVE	HANGUP	134	DITP = 000000	0
134	INACTIVE	HANGUP	140	DITP = 000000	0
140	INACTIVE	HANGUP	144	DITP = 000000	0
144	INACTIVE	HANGUP	150	DITP = 000000	0
150	INACTIVE	HANGUP	154	DITP = 000000	0
154	INACTIVE	HANGUP	160	DITP = 000000	0
160	INACTIVE	HANGUP	164	DITP = 000000	0
164	INACTIVE	HANGUP	170	DITP = 000000	0
170	INACTIVE	HANGUP	174	DITP = 000000	0
174	INACTIVE	HANGUP	200	DITP = 000000	0

SYSTEM GLOBAL PORT DATA SEGMENT

* FIND %1377 SYSTEM GLOBAL (WHICH IS ADDRESSED TO SYSTEM GLOBAL EXTENSION)

* ADD %1000 TO IT, SYSGLOB

* PORT DATA SEGMENT IN SYSTEM GLOBAL EXTENSION + %100

  SYSTEM GLOBAL EXTENSION CELL + %1100 = DST # FOR PORT DATA SEGMENT.

```
HP3000 III MEMORY DUMPC.00.02 OF SYS VER C  UPDATE  01  FIX 00  DUMP TIME 10/18/81,  4:01PM                BANK   0
(C) HEWLETT-PACKARD CO. 1980

000404: 070000 040000 000000 000000 000000 040000 000000 000000  000414: 000000 000000 000000 000000 000000 000000 000000 000000
000424: 000000 000000 000000 000000 000000 000000 000000 000000  000434: 000000 000000 000000 000000 000000 000000 000000 000000
000444: 000000 000000 000000 000000


$$$$$$$$ DST   23 (TIMER REQUEST LIST) $$$$$$$$
000450: 000024 020004 010020 000000 177747 001580 050380 121441  000460: 000024 000000 000000 000000 112000 000120 000000 038254
000470: 124014 001300 000000 000008 124030 001320 000000 023207  000500: 120020 000001 000000 000153 000040 000000 000000 000000
000510: 000044 000000 000000 000000 000050 000000 000000 000000  000520: 000054 000000 000000 000000 000060 000000 000000 000000
000530: 000084 000000 000000 000000 000070 000000 000000 000000  000540: 000074 000000 000000 000000 000100 000000 000000 000000
000550: 000104 000000 000000 000000 000110 000000 000000 000000  000560: 000114 000000 000000 000000 000120 000000 000000 000000
000570: 000124 000000 000000 000000 000130 000000 000000 000000  000600: 000134 000000 000000 000000 000140 000000 000000 000000
000610: 000144 000000 000000 000000 000150 000000 000000 000000  000620: 000154 000000 000000 000000 000160 000000 000000 000000
000630: 000164 000000 000000 000000 000170 000000 000000 000000  000640: 000174 000000 000000 000000 000200 000000 000000 000000

000650: 000000 000000 000000 000000


$$$$$$$$ DST   30 (JOB PROCESS COUNT) $$$$$$$$
000654: 011422 000378 177378 177378 177378 177378 177378 177378  000664: 177378 177378 177377 000000 000000 000000 000000 000000


$$$$$$$$ DST   44 (JOB CUTOFF TABLE) $$$$$$$$
000674: 000003 011403 000071 000006 000000 000000 000011 000000  000704: 000000 000014 000000 000000 000017 000000 000000 000022
000714: 000000 000000 000025 000000 000000 000030 002000 000000  000724: 000032 000000 000000 000035 000000 000000 000041 000000
000734: 000000 000044 000000 000000 000047 000000 000000 000052  000744: 000000 000000 000055 000000 000000 000000 000000 000000
000754: 000063 000000 000000 000066 000000 000000 000071 000000  000764: 000000 000000 000000 000000 000004 000001 030370 030370
000774: 000000 030370 030370 030370


$$$$$$$$ DST    5 (SYSTEM GLOBAL AREA) $$$$$$$$


001000: 000000 005754 005314 012014 177404 014514 021704 013514  001010: 038510 000000 177450 177874 038700 177654 000704 037071
001020: 000000 128023 001560 010733 000000 023724 034320 000004  001030: 000000 015544 001440 003040 000000 003724 000000 000000
001040: 034340 034360 013174 000000 034504 000121 000000 000003  001050: 100001 177220 000400 010001 000400 177174 177202 177210
001060: 000000 000000 000000 000000 000000 000000 000000 000000  001070: 000000 000026 000000 000001 000024 018015 000111 000000
001100: 000001 172000 000000 000000 000800 040000 000077 075000  001110: 002280 077777 000004 000107 030111 030071 020103 000000
001120: 000170 000000 103440 007232 000438 002340 000400 008100  001130: 000000 000136 177773 102113 000000 101117 000000 000000
001140: 104118 000020 000000 000160 000200 000220 000000 000000  001150: 000100 000240 000040 000040 000120 100441 000000 101415
001160: 001864 104054 101554 001245 002343 111054 004708 007703  001170: 000000 000000 002037 000040 000002 000252 000000 000008

001200: 000000 000000 000000 000000 001777 000284 000000 000001  001210: 000000 000000 000000 000000 000001 000000 000000 000000
001220: 000104 000000 000000 000001 000001 031344 002001 000787  001230: 000400 000200 000000 001300 000713 048517 051122 044523
001240: 020040 044120 042440 020040 020040 000000 000000 000000  001250: 000000 000000 000000 000000 000000 000000 000000 000000
001260: 000000 177284 000000 000000 000000 000001 104223 000000  001270: 000000 000000 000000 000000 000000 000000 000000 000000
```

Dump Example

```
HP3000 III MEMORY DUMPC.00.02 OF SYS VER C  UPDATE  0I  FIX 00  DUMP TIME 10/18/81,  4:01PM                BANK  . 0
(C) HEWLETT-PACKARD CO. 1980

001300: 000000 000000 000005 000000 004402 000020 003400 000020  001310: 002001 000020 001403 000020 002404 000020 000000 000000
001320: 000000 000000 000000 112132 000000 000000 000000 000000  001330: 000000 000000 000000 000310 000310 104130 110530 106130
001340: 100527 105532 101134 100534 100531 000000 000001 000000  001350: 000000 000000 023855 000000 001750 000001 000800 000000
001360: 000000 000003 177777 000140 000000 000000 000000 000000  001370: 000000 000401 137734 000113 100547 101144 000000 000504

001400: 000000 000000 000000 000000 000000 000000 000000 000000  001410: 177526 000000 034172 000000 000000 000000 000000 000000
001420: 000005 000012 000047 000360 000000 000000 000000 000000  001430: 000000 000000 000000 000000 000000 000000 000000 000000
001440: 030370 030370 030370 030370 030370 030370 030370 030370  001450: 030370 030370 030370 030370 030370 030370 030370 030370
LINES 001460 - 001477 SAME AS ABOVE

001500: 030370 030370 030370 030370 000000 000000 126023 000001  001510: 000030 000200 000000 001550 041504 000764 000000 000000
001520: 000000 000000 000000 000000 000000 000000 000000 000000  001530: 000000 000000 000000 000000 000000 000000 000000 000000
LINES 001540 - 001557 SAME AS ABOVE

001560: 000000 000000 000000 000000 105472 005254 104472 002676  001570: 005262 108072 000000 000000 000000 000000 000131 102552
001600: 020103 030111 030071 000000 000000 000000 000000 000000  001610: 000000 000000 000000 000000 000000 000000 000000 000000
001620: 000000 000000 000000 000000 000000 000000 000000 000000  001630: 000000 000000 000000 000000 000000 000000 000000 000000
LINES 001640 - 001677 SAME AS ABOVE

001700: 000000 000000 000000 000000
```

%1577

%1504 BEGIN SYS GLOB EXTENSION

%1604 PORT DATA SEGMENT NUMBER = 137

```
$$$$$$$$ DST   14 (TERMINAL BUFFERS) $$$$$$$$
001704(000000): 040070 000020 001750 001730 004400 000000 000000 005078  000030 005040 020040 020465  001704:@8....p.....h.....  15
001720(000014): 020040 020040 020113 046117 047105 008412 020040 020081  033040 020040 020040 051473  001720:      KLONE..  18    S.
001734(000030): 000050 042508 020111 030415 005040 020040 030467 020040  020040 020120 040525 051505  001734:.(EF K1..    17      PAUSE
001750(000044): 020081 030015 005040 020040 000070 030470 020040 020040  020103 008412 002440 020040  001750:18..      $18    C.
001764(000060): 030471 020040 020040 020117 050105 047040 046523 043481  000110 035501 036461 008412  001764:19        OPEN MSG1.H.A+1..
002000(000074): 020040 020062 030040 020040 020040 053522 044524 042415  005040 020040 031081 020040  002000:    20      WRITE..    21
002014(000110): 000130 020040 020103 008412 020040 020062 031040 020040  020040 047520 042518 020115  002014:.X    C..    22      OPEN M
002030(000124): 051507 031415 005040 020040 000150 002462 031440 020040  020040 051505 052111 047124  002030:SG3...    .h.23      SETINT
002044(000140): 020115 051507 031415 005040 020040 031064 020040 020040  020040 000170 020103 047516  002044: MSG3..    24    .x CONTR
002060(000154): 047514 020115 051507 031473 041475 032073 050075 030480  030060 008412 020040 020062  002060:OL MSG3;C=4 P=1000..    2
002074(000170): 000210 032440 020040 020040 051105 040504 020115 051507  031415 005040 020040 002482  002074:..S     READ MSG3..    2
002110(000204): 033040 020040 020040 041415 000230 005040 020040 031067  020040 020040 020117 050105  002110:6       C.....  27      OPE
002124(000220): 047040 046523 043484 008412 020040 020062 034040 020040  020040 000250 020040 051505  002124:N MSG4..    28    .. SETI
002140(000234): 047124 020115 051507 032015 005040 020040 031071 020040  020040 020122 042501 042940  002140:NT MSG4..    28      READ
002154(000250): 000270 046523 043484 008412 020040 020062 002460 020040  020040 020120 040525 051505  002154:..MSG4..    3.0     PAUSE
002170(000264): 020081 030080 030015 005040 000310 020040 031461 020040  020040 020105 047506 008412  002170:1000..      21    EOF..
002204(000300): 027480 044524 027483 030060 030040 020108 051111 028040  030060 000330 045440 052040  002204:/OIT/3000  FRI..X T 18
002220(000314): 028040 030471 034061 028040 020063 035065 034440 050115  008412 024103 024440 044105  002220:. 1981.  3:59 PM..(C) HE
002234(000330): 000250 048481 008412 052055 050101 041513 040522 042040  041517 027040 030471 002470  002234:..MI..T-PACKARD CO. 19.I
002250(000344): 030015 005080 030040 030060 000410 048481 020101 048122  042501 042131 020105 054111  002250:0..00 00..M1 ALREADY EXI
002264(000360): 051324 051440 028440 051105 051520 047516 042040 054505  000530 020015 005075 020120  002264:STS - RESPOND YE.X ..= P
002300(000374): 052522 042505 020115 051507 032015 005040 020040 030461  020040 002440 020040 041125  002300:URGE MSG4      11      BU
002314(000410): 000430 051440 052117 020120 052522 043505 020117 048104  020101 047104 020113 042905  002314:..S TO PURGE OLD AND KEE
002330(000424): 050040 047105 053415 005120 000450 052522 043505 020117  048104 037460 030060 030060  002330:P NEW  P.[URGE OLD?00000
002344(000440): 030440 030460 033483 030064 008412 042101 030462 032453  000470 054720 027481 031000  002344:1  107304  DA125= BY ./12
002360(000454): 029060 030060 030064 032440 030060 030060 030064 020061  033467 033467 033440 030060  002360: 000047 000004 177777 00
002374(000470): 000519 027480 020040 034840 020040 020040 041125 044514  042040 046523 043483 035515  002374:.H/O  I     BUILD MSG3;M
002410(000504): 051507 008412 020081 032467 000370 042440 020040 034440  020040 020040 043111 046105  002410:SG    177   E     8     FILE
002424(000520): 020115 051507 031454 047514 042015 005040 020040 030460  000550 042530 044524 046523  002424: MSG3 OLD..    10 hEXITMS
002440(000534): 042484 035515 051507 008412 020040 020081 031040 020040  020040 043111 046105 020115  002440:G4:MSG..    12      FILE M
002454(000550): 000570 008412 032054 047514 042015 005057 030060 020060  030060 030060 032040 030060  002454:..x..4.OLD../00 000004 00
```

```
*SPCBDELAYSOFT   PCB (8:1) 000100 = 0 000 000 001 000 000
                                    0 1   4   7   10  13
NOT SET

PROCESS IS CRITICAL, HOLDING SIR, IMPEDED

*SPCBWAKESOFT    PCB 8.(3:1) 172400 = 1 111 010 100 000 000
                                      0 1   4   7   10  13

SET                           172000 = 1 111 010 000 000 000
PROCESS ALLOWS SOFT INT               0 1   4   7   10  13
EVEN IF WAITING ON OTHER EVENTS

*SPCBSOFTINT    PCB 9.(9:1) 100200 = 1 000 000 010 000 000
                                     0 1   4   7   10  13
NOT SET                      100000 = 1 000 000 000 000 000
                                     0 1   4   7   10  13
SET TO INFORM DISP/PSUEDOINT
SHOULD BE AWARE OF SOFT INT

*SPCBALLOWSOFT    PCB 13.(7:1) 02220 = 0 010 010 010 011 000
                                       0 1   4   7   10  13

NOT SET

SET, THE PROCESS WILL PROCESS
SOFT INT (FINSTATE)
```

Dump Example

```
HP3000 III MEMORY DUMPC.00.02 OF SYS VER C  UPDATE  01  FIX 00  DUMP TIME 10/15/81,  4:01PM                BANK   0
(C) HEWLETT-PACKARD CO. 1980

012414: 100540 000000 000401 041376 100244 000000 000401 041411  012424: 100570 000000 000401 041417 100460 000000 000401 041433
012434: 100506 000000 000401 041445



012440: 000004 125252 000001 000000 021808 000000 000003 152423  012450: 021153 000000 000001 037223 021051 000000 000000 152423
012460: 021465 000000 000002 030823 000004 125252 000001 000000  012470: 021808 000000 000001 080223 021152 000000 000000 144423
012500: 021051 000000 000002 047023 021465 000000 000002 075823  012510: 100000 001140 000000 000000 100000 001144 000000 000000
012520: 100000 001150 000000 000000 100000 001154 000000 000000  012530: 100000 001160 000000 000000 100000 001164 000000 000000
012540: 100000 001170 000000 000000 100000 001174 000000 000000  012550: 100000 001200 000000 000000 100000 001204 000000 000000
012560: 100000 001210 000000 000000 100000 001214 000000 000000  012570: 100000 001220 000000 000000 100000 001224 000000 000000
012600: 100000 001230 000000 000000 100000 001234 000000 000000  012610: 100000 001240 000000 000000 100000 001244 000000 000000
012620: 100000 001250 000000 000000 100000 001254 000000 000000  012630: 100000 001260 000000 000000 100000 001264 000000 000000

012640: 100000 001270 000000 000000 100000 001274 000000 000000  012650: 100000 001300 000000 000000 100000 001304 000000 000000
012660: 100000 001310 000000 000000 100000 001314 000000 000000  012670: 100000 001320 000000 000000 100000 001324 000000 000000
012700: 100000 001330 000000 000000 100000 001334 000000 000000  012710: 100000 001340 000000 000000 100000 001344 000000 000000
012720: 100000 001350 000000 000000 100000 001354 000000 000000  012730: 100000 001360 000000 000000 100000 001364 000000 000000
012740: 100000 001370 000000 000000 100000 001374 000000 000000  012750: 100000 001400 000000 000000 100000 001404 000000 000000
012760: 100000 001410 000000 000000 100000 001414 000000 000000  012770: 100000 001420 000000 000000 100000 001424 000000 000000
013000: 100000 001430 000000 000000 100000 001434 000000 000000  013010: 100000 000000 000000 000000


DUCB $$$$$$$$ DST    3 (PROCESS CONTROL BLOCK)$$$$$$$$
HEAD 013014: 000057 000270 000041 001440 001240 000017 000000 000000  013024: 000000 000000 000000 000000 000000 000000 000000 000000
     013034: 000000 024083 000000 004300 000400 000002 000000 000000  013044: 164000 101000 000000 000000 000010 040041 000000 000000
     013054: 000000 023731 100000 003640 000400 000400 001400 000000  013064: 160000 101000 000000 040041 000000 040042 000000 000000
     013074: 000000 023743 000000 003700 002200 000400 002000 000000  013104: 160000 101000 000000 040037 000000 040170 000000 000000
     013114: 000000 023755 000000 003740 000400 000400 002400 000000  013124: 162000 101000 000000 000000 000001 040062 000000 000000
     013134: 020000 023787 000000 004000 000010 000400 003000 000000  013144: 162000 101000 000000 000000 000002 040175 000000 000000
     013154: 000000 024001 000000 004040 000400 000400 003400 000000  013164: 162000 101000 000000 000000 000003 040175 000000 000000
     013174: 000000 024013 000440 004100 000400 000418 004000 000000  013204: 162000 101400 000000 000000 000004 040175 000000 000000

     013214: 100000 024025 000000 004140 000401 000400 004400 000000  013224: 160000 101000 000000 000000 000005 040012 000000 000000
     013234: 000000 024037 000000 004200 002400 000400 005000 000000  013244: 160000 101000 000000 000000 000006 040175 000000 000000
     013254: 000000 024051 000000 004240 000400 000400 005800 000000  013264: 160000 101000 000000 000000 000007 040216 000000 000000
     013274: 000000 000320 000440 004400 001000 000400 006400 000000  013304: 162000 001000 000000 000000 000000 040230 000000 177777
     013314: 000100 024551 000000 004600 000060 000400 006800 000000  013324: 164000 101000 000000 000000 000000 040230 000000 000000
     013334: 000000 000360 000440 004420 001000 003400 000000 000000  013344: 162000 000400 000000 000000 000000 022230 000000 177777
PCB  013354: 000100 025021 000000 004420 000040 001452 000000 000000  013364: 164000 100400 000000 000000 000000 022230 000000 000000
16   013374: 000000 000400 000440 005060 001000 003400 000000 000000  013404: 162000 000400 000000 000000 000000 010310 000000 177777

     013414: 000000 000440 000440 005500 001000 007000 000000 000000  013424: 162400 000200 000000 040044 000015 022230 000000 177777
     013434: 000000 000460 000440 005500 001000 007000 000000 000000  013444: 164400 000200 000000 000000 000027 022230 000000 177777
     013454: 000000 000420 000440 005200 001000 010400 000000 000000  013464: 162400 000000 000000 000000 000022 022230 000000 177777
     013474: 000000 000520 000440 005500 001000 007000 000000 000000  013504: 162400 000200 000000 000000 000011 022230 000000 177777
     013514: 000000 000560 000440 005500 001000 007000 000000 000000  013524: 164400 000200 000000 000000 000015 022230 000000 177777
     013534: 000000 000500 000440 005540 001000 012000 000000 000000  013544: 162400 000000 000000 040044 000011 022230 000000 177777
     013554: 000000 000600 000440 005500 001000 007000 000000 000000  013564: 164400 000200 000000 040044 000027 022230 000000 177777
     013574: 000000 000540 000440 005240 001000 013000 000000 000000  013604: 162400 000000 000000 040044 000022 022230 000000 177777

     013614: 000000 000520 000440 005500 001000 007000 000000 000000  013624: 162400 000200 000000 000000 000015 022230 000000 177777
     013634: 000000 000640 000440 005500 001000 007000 000000 000000  013644: 162400 000200 000000 000000 000020 022230 000000 177777
```

```
HP3000 III MEMORY DUMPC.00.02 OF SYS VER C  UPDATE  0I  FIX 0@  DUMP TIME 10/18/81,  4:01PM                BANK   0
(C) HEWLETT-PACKARD CO. 1980

013654: 000000 000000 000000 000000 000000 000000 000000 000000   013664: 000000 000000 000000 000000 000000 000000 000000 177777
013674: 000000 000700 000440 005500 001000 007000 000000 000000   013704: 182400 000200 000000 000000 000013 022230 000000 177777
013714: 000000 000740 000440 005500 001000 007000 000000 000000   013724: 182400 000200 000000 000000 000020 022230 000000 177777
013734: 000000 000760 000440 005500 001000 007000 000000 000000   013744: 184400 000200 000000 000000 000015 022230 000000 177777
013754: 000100 000720 000000 005280 000040 018400 000000 000000   013764: 042400 000000 000000 000000 000027 022230 000000 177777
013774: 000000 001000 000440 005500 001000 007000 000000 000000   014004: 182400 000200 000000 000000 000013 022230 000000 177777

014014: 000000 001040 000440 005500 001000 007000 000000 000000   014024: 182400 000200 000000 000000 000011 022230 000000 177777
014034: 000100 001120 000000 005520 000040 007000 000000 000000   014044: 043400 000200 000000 000000 000015 022230 000000 177777
014054: 000100 001020 000000 005380 000040 020400 000000 000000   014064: 043000 000000 000000 000000 000027 022230 000000 177777
014074: 000010 001220 000440 005500 001000 007000 000000 000000   014104: 184400 000200 000000 000000 000015 022232 000000 177777
014114: 000100 001180 000000 005220 000040 021400 000000 000000   014124: 043000 020000 000000 000000 000027 022230 000000 177777
014134: 000010 001100 000440 005540 001000 021400 000000 000000   014144: 182400 000000 000000 040044 000011 022232 000000 177777
014154: 000100 001080 000000 005020 000040 021400 000000 000000   014164: 043000 020000 000000 000000 000027 022230 000000 177777
014174: 000000 001140 000440 005200 001000 021400 000000 000000   014204: 182400 000000 000000 040044 000011 022230 000000 177777

014214: 000000 001260 000440 005500 001000 007000 000000 000000   014224: 184400 000200 000000 000000 000015 022230 000000 177777
014234: 000100 001200 000000 005360 000040 024000 000000 000000   014244: 053400 000000 000000 000000 000027 022230 000000 177777
014254: 000000 000000 000440 005500 001000 007000 000000 000000   014264: 184400 000200 000000 000000 000015 022230 000000 177777
014274: 000000 001240 000440 005040 001000 025000 000000 000000   014304: 182400 000000 000000 040044 000011 022230 000000 177777
014314: 000100 001140 000000 005420 000040 007000 000000 000000   014324: 172400 100200 000000 000000 000015 022230 000000 000000
014334: 000100 001140 000000 005360 000040 020000 000000 000000   014344: 172000 100000 000000 000000 000027 022230 000000 000000
014354: 100000 001280 001320 000000 000000 000000 000000 000000   014364: 000000 000000 000000 000000 000000 000000 000000 177777
014374: 100000 000280 001240 000000 000000 000000 000000 000000   014404: 000000 000000 000000 000000 000000 000000 000000 177777


$$$$$$$$ DST    7 (INTERRUPT CONTROL STACK) $$$$$$$$
014414: 000000 000000 000000 000000 000000 000000 000000 000000   014424: 000000 000000 000000 000080 000000 000000 000000 000054
014434: 000001 000000 000000 000000 001750 001750 000143 000144   014444: 000204 000454 000000 000340 000312 000230 000375 000256
014454: 000310 000000 000000 000000 000000 000000 000000 000000   014464: 000000 000000 000000 000000 000000 000000 000000 000000
014474: 000132 100074 023133 177777 000000 000874 001300 087422   014504: 012580 177844 008415 000001 070822 000000 001512 100074
014514: 000000 000000 001000 000007 013314 006064 000230 000000   014524: 013314 000000 014746 101033 000000 000000 024812 000000
014534: 000006 000001 087423 002714 000000 000001 002676 000000   014544: 001200 000000 040037 000400 000000 013314 000000 001580
014554: 042454 001580 041504 000000 000754 000235 000000 000704   014564: 000000 001300 000000 000302 001000 040000 000000 000003
014574: 005105 100433 000032 177756 002688 141074 100068 000000   014604: 001000 000003 000000 000000 003034 000354 000007 000362

014614: 014746 101033 000014 022270 100033 000017 001300 000200   014624: 000000 015283 025228 101033 000007 013314 004000 000000
014634: 001000 013314 000001 000000 013314 017277 100074 000013   014644: 000000 000000 000230 000000 177777 000235 000000 000000
014654: 000303 001000 000000 133201 000003 005105 100432 000031   014664: 000000 000000 000302 000000 013887 000002 000121 000003
014674: 000000 000507 014133 103074 000020 000000 000011 000002   014704: 000000 013704 020664 000000 000000 012023 000128 000000
014714: 000521 012764 100074 000015 001300 002714 000000 000001   014724: 012008 000054 000000 010000 000000 031171 014208 103074
014734: 000015 013314 000001 000000 001000 102074 012000 000001   014744: 013022 000000 005177 000000 177134 020664 000000 000012
014754: 018024 017178 000001 017201 000001 002418 000203 001280   014764: 000412 002770 000003 005105 102033 000031 000000 000000
014774: 000303 000000 013775 170000 012023 002360 141155 000001   015004: 020664 177777 000004 000043 000011 000000 000000 000000

015014: 020664 000000 002714 020711 177777 177775 002360 141155   015024: 000031 020864 000004 000064 021711 040000 000004 000007
015034: 024854 100033 000012 087423 000038 177777 007254 004814   015044: 144474 000066 000000 000000 067406 000000 000000 000000
015054: 002714 020711 177777 177775 002380 141155 000031 020864   015064: 000004 000004 021711 040000 000004 000007 024854 100033
015074: 000012 000000 000000 000000 041401 000613 000000 000002   015104: 000015 000002 000015 000000 000001 104200 000001 107023
015114: 000000 003177 011651 100474 000031 000002 000002 000001   015124: 107376 000001 107401 000001 107401 000001 107377 000001
015134: 107023 000002 000000 000013 012225 100074 000022 000000   015144: 157004 000002 006006 000003 082408 000001 000001 020694
015154: 000000 000000 020711 177777 177775 002360 141155 000031   015184: 020884 000004 000004 021711 040000 000004 000007 024854
015174: 100033 000012 000012 000000 000000 000000 000000 000000   015204: 000000 000000 000000 000000 000000 000000 000000 000000
```

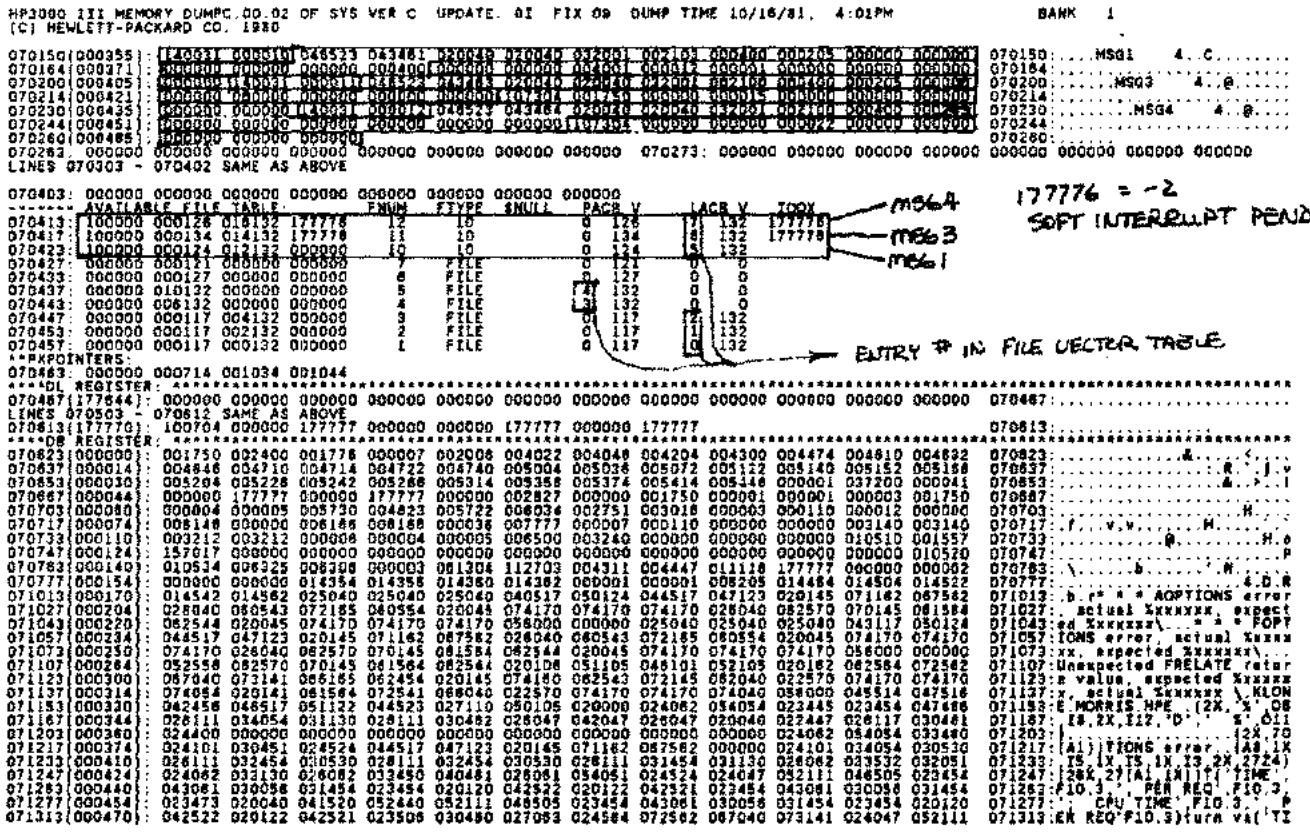
```
MESSAGE HBR TABLE: (5 *55) + 34340 = 34701
                   (ENTRY LENGTH *PIN #) + MSG HBR TABLE BASE

WORD 1:  JUNK PORT

WORD 1:  USER SOFT INTERRUPTS,
         (0:1) = 1 = NEXT MESSAGE IN SECONDARY MESSAGE TABLE
         (1:15) = 12 = INDEX OF ENTRY IN TABLE

WORD 2:  SYSTEM SOFT INTERRUPTS

WORD 3:  RESERVED

WORD 4:  BIT MAP = 040000 = 0100 000 000 000 000

         BIT ONE BEING SET CORRELATES WITH AN ENTRY EXISTING FOR
         WORD 1 = USER SOFT INTERRUPTS


NOTE:  ENTRY EXISTS FOR EACH PCB BUT NOT PRINTED BECAUSE ONLY
       ONE ENTRY IS NON-ZERO
```

```
HP3000 III MEMORY DUMPC.GG.02 OF SYS VER C  UPDATE  01  FIX 09  DUMP TIME 10/16/81,  4:01PM              BANK   0
(C) HEWLETT-PACKARD CO. 1980


$$$$$$$$ DST   46 (SPECIAL REQUEST TABLE) $$$$$$$$
034174: 000023 000005 000023 000024 000017 000000 000012 001575  034204: 000800 000644 000000 000017 002375 000200 000440 000000
034214: 000000 001035 000200 000640 000000 000031 001375 000200  034224: 000644 000000 000038 001435 000200 000844 000000 000043
034234: 002375 000200 000440 000000 000050 002575 000000 000000  034244: 000000 000055 003028 000200 000440 000000 000082 004226
034254: 000000 000000 000000 000087 004088 000200 000640 000000  034264: 000074 004266 000000 000000 000000 000101 004268 000000
034274: 000000 000000 000108 001375 000200 000844 000000 000113  034304: 001435 000200 000844 000000 000120 002375 000200 000440
034314: 000000 000125 002575 000000 000000 000000 000132 002775  034324: 000000 000000 000000 000137 001375 000200 000844 000000
034334: 000005 001435 000200 000664


$$$$$$$$ DST   71 (MSG HDR TABLE) $$$$$$$$
034340: 000000 000000 000000 000000 000000 000000 000000 000000  034350: 000000 000000 000000 000000 000000 000000 000000 000000
LINES 034360 - 034677 SAME AS ABOVE

034700: 000000 000000 100012 000000 000000 040000 000000 000000  034710: 000000 000000 000000 000000 000000 000000 000000 000000


$$$$$$$$ DST   72 (PRIMARY MSG TABLE) $$$$$$$$
034720: 000030 000005 000030 000005 000170 100000 000012 013314  034730: 000000 000000 100000 000017 000005 000000 000000 100000
034740: 000024 000012 000000 000000 100000 000031 000017 000080  034750: 000000 100000 000038 000024 000000 000000 100000 000043
034760: 000031 000000 000000 100000 000050 000038 000000 000000  034770: 100000 000055 000843 000000 000000 100000 000082 000050
035000: 000000 000000 100000 000087 000055 000000 000000 100000  035010: 000074 000062 000000 000000 100000 000101 000087 000000
035020: 000000 100000 000108 000074 000000 000000 100000 000113  035030: 000101 000000 000000 100000 000120 000108 000000 000000
035040: 100000 000125 000113 000000 000000 100000 000132 000120  035050: 000000 000000 100000 000137 000125 000000 000000 100000
035060: 000144 000132 000000 000000 100000 000151 000137 000000  035070: 000000 100000 000158 000144 000000 000000 100000 000183
035100: 000151 000000 000000 100000 000170 000158 000000 000000  035110: 100000 000000 000183 000000 000000 000000 000000 000000
        OVERHEAD


$$$$$$$$ DST   74 (SECONDARY MSG TABLE) $$$$$$$$
035120: 000030 000005 000030 000005 000170 000000 000000 177777  035130: 107304 000001 100005 000004 000010 107304 000001 100000
035140: 000024 000012 000000 000000 100000 000031 000017 000080  035150: 000000 100000 000038 000024 000000 000000 100000 000043
035160: 000031 000000 000000 100000 000050 000038 000000 000000  035170: 100000 000055 000843 000000 000000 100000 000082 000050
035200: 000000 000000 100000 000087 000055 000000 000000 100000  035210: 000074 000062 000000 000000 100000 000101 000087 000000
035220: 000000 100000 000108 000074 000000 000000 100000 000113  035230: 000101 000000 000000 100000 000120 000108 000000 000000
035240: 100000 000125 000113 000000 000000 100000 000132 000120  035250: 000000 000000 100000 000137 000125 000000 000000 100000
035260: 000144 000132 000000 000000 100000 000151 000137 000000  035270: 000000 100000 000158 000144 000000 000000 100000 000183
035300: 000151 000000 000000 100000 000170 000158 000000 000000  035310: 100000 000000 000183 000000 000000 000000 000000 000000
        FIRST WORD OF FREE ENTRIES


$$$$$$$$ DST   47 (VIRTUAL DISK SPACE TABLE) $$$$$$$$
035320: 000184 000001 000020 001000 000004 000020 001402 001347  035330: 000000 000000 000000 000000 000000 000000 000000 000000
035340: 000020 000001 000000 018100 000000 012000 002400 001402  035350: 000121 001500 001347 000000 000000 000000 000000 000000
035360: 000000 000000 000000 000000 000000 000000 000000 000000  035370: 000000 000000 000000 000000 000000 000000 000000 000000
035400: 000000 000000 000000 000000 000000 000000 003777 177777  035410: 177770 000000 000000 000000 000000 000000 000000 000000
```

```
HP3000 III MEMORY DUMPC.00.02 OF SYS VER C  UPDATE  OI  FIX 09  DUMP TIME 10/16/81,  4:01PM                  BANK   1
(C) HEWLETT-PACKARD CO. 1980

011033(001210): 020040 040440 020040 020040 020040 001364 000043 048517 047123 042040 020040 001368  011033:  A          ..$MONSD
011047(001224): 000050 000050 000050 000050 000050 000050 000050 000050 000050 000050 000050 000050  011047:.(.(.(.(.(.(.(.(.(.(.(.(
LINES 011063 - 011206 SAME AS ABOVE
011207(001384): 000050 000050 000050 000050 000050 000050 000050 000050 000050 000050 000050 000000  011207:.(.(.(.(.(.(.(.(.(.(.(.
011223(001400): 000000 000000 000000 000000 000000 000000 000000 000000 000000 000000 000000 000000  011223:........................
LINES 011237 - 011412 SAME AS ABOVE
011413(001570): 000000 000000 000000 000000 000000 000000 000000 120141 000000 000000 000000 021601  011413:...............a......#
011427(001604): 001350 048517 051122 044523 020040 020040 020040 020040 020040 020040 020040 020040  011427:..MORRIS
011443(001620): 020040 020040 020040 020040 020040 020040 020040 020040 020040 020040 020040 020040  011443:
LINES 011457 - 011816 SAME AS ABOVE
011817(001774): 020040 020040 020040 000000 000150 000025 000001 000000 000001 000035 000004 002421  011817:       ...h............
011833(002010): 103001 000027 000001 001023 000004 007375 102001 000035 000000 034340 000000 034340  011833:.....................8...8.
011847(002024): 000000 034720 000000 035120 000000 034725 000005 000000 000005 000012 000003 000000  011847:..9...:P..8..............
011863(002040): 177777 025261 000000 000000 000125 002400 000000 000000 000165 000050 000002 000000  011863:..*.......U.........u.(...
011677(002054): 000000 000000 000000 000000 000132 011412 000001 000001 041762 000000 000000 000000  011677:.........Z......C.......
011713(002070): 000000 000000 000000 000000 000000 001200 000000 000000 020040 000000 000020 000000  011713:........................
011727(002104): 000002 000000 000000 000000 000000 000020 000001 000000 100132 006740 000000 001200  011727:..................Z......
011743(002120): 000001 041755 000001 176600 005247 140077 000115 000041 177777 002714 000001 177134  011743:..C.........?.M.!........\
011757(002134): 016544 000000 000132 177777 000007 010687 140032 000015 000000 140074 015544 015545  011757:.d...Z.................<.d r
011773(002150): 001000                                                                                011773:..
011774(002151): 000780 000011 100000 000011                                                           011774:.........

012000: 100000 000004 000000 110001 018024 100000 028054 000000  012010: 000000 000004 100000 000000 000128 000000 000000 000400
012020: 020720 000000 000000

                                 WORD 111/112   WORD 47
$$$$$$$$ DST  128 $$$$$$$$
012023(000000): 000604 000128 110001 000000 000000 000015 000005 000000 000000 000004 000000 000000  012023:....V...................
012037(000014): 000000 100567 000012 048523 043404 020040 020040 032001 002100 000400 000205 000000  012037:....w..MSG4     4..@.......
012053(000030): 000000 000000 000000 000000 000000 000000 000000 000000 000000 000000 000000 000000  012053:........................
012067(000044): 000000 000388 000368 030112 000000 000401 000000 000001 000001 000182 000215 000000  012067:.......0J...............r....
012103(000060): 000024 000001 001000 037401 032221 000000 002007 000000 000182 000000 000000 000001  012103:.......?...............r....
012117(000074): 000000 177777 177777 177777 177777 177777 177777 177777 177777 177777 177777 177777  012117:........................
012133(000110): 177777 177777 177777 177777 177777 000000 000000 000000 000000 000000 000000 000000  012133:........................
012147(000124): 000000 000000 000021 000020 000000 000000 000000 000000 000203 000000 100000 000000  012147:........................
012163(000140): 000000 177400 000000 000054 000012 107304 000000 000000 000182 000377 000000 000000  012163:....................r.....
012177(000154): 000000 000000 000000 000000 000000 000000 000000 000000 000000 000000 000000 000000  012177:........................
LINES 012213 - 012822 SAME AS ABOVE
012823(000800): 000000 000000 000000 000000 032040 030488 028040 030471 031401 043604 152700 000100  012823:.........T 18, 183.G....^
012837(000814): 022050 145514 025001 051435 041804 031017 043604 043604 020218 043604 170004 010201  012837:$(.L*.S.C.2.G.G.  .G....
012853(000830): 140004 000220 040514 048040 021003 020243 141526 000800 051435 043604 022402 052604  012853:.....ALL *.....T..S.G.%.W
012867(000844): 041604 031017 043604 152700 000100 022051 141205 021002 051032 031401 140003 041604  012867:C.2.G.....@$)..*.R.3...C.
012703(000860): 031017 140170 043804 152700 000100 022043 145524 025001 051436 041804 031017 043604  012703:2..xG.....@$#.T*.S.C.2.G.
012717(000874): 043604 020221 043804 002100 051445 000700 043604 041445 021015 161441 141205 021045  012717:G.  .G..@S%..G.CX2..!..^
012733(000710): 051032 031401 000071 043604 071445 053604 043604 152700 000100 022040 141503 041804  012733:R.3..9G.sXW.G.....@$ .CC.
012747(000724): 031017 043804 152700 000100 022057 141204 021002 051032 031401 041804 031017 043804  012747:2.G.....@!/..*.R.3.C.2.G
012763(000740): 152700 000100 022043 141204 021002 051032 031401 041604 031017                       012763:....@$#..*.R.3.C.2.
012774(000751): 043804 000004 100000 000004                                                           012774:G.........

013000: 020000 000021 000700 000000 000000 000000 000000 000000  013010: 021045 000020 000000 043804 000000 053804 043804 152700
013020: 000100 022040 141503

$$$$$$$$ AVAILABLE AREA $$$$$$$$
```

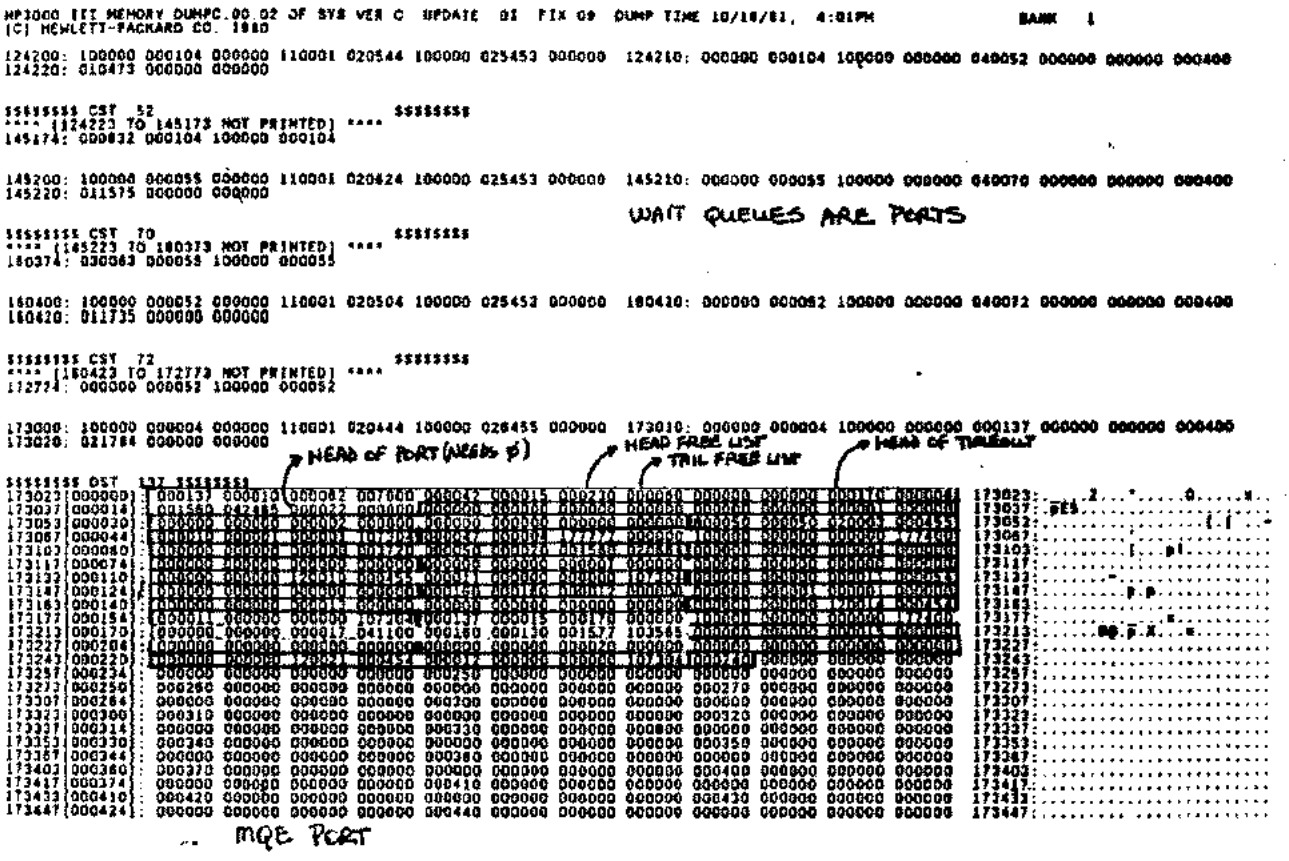
```
HP3000 III MEMORY DUMPC.00.02 OF SYS VER C  UPDATE  01  FIX 09  DUMP TIME 10/16/81,  4:01PM            BANK   1
(C) HEWLETT-PACKARD CO. 1980

                    [****** PCBX AND STACK MARKERS FOR DST 132  (PCB 54 )     ******]

SEQ REL   SEQ REL   JMAT     JPCNT    JOB INPUT  JOB OUTPUT  JDT DST  JIT DST
DL        DB        INDEX    INDEX    LOG DEV #  LOG DEV #   INDEX    INDEX    JOB TYPE  DUPLICAT  INTERACT  INIT Q   JOUT INDEX
001044    001200        1        2       20          20      126116      115   #S2          YES       YES   006302        0

ADDRESS   BANK      X        DELTA P   STATUS   DELTA Q    SEGMENT

077236     1      177750     020003    101074    000012     74
077224     1      000021     016636    141074    000014     74
077210     1      006337     001725    142436    000020     38
077170     1      000041     004074    062304    000007    304  USER SEGMENT
077162     1      000041     000207    062304    000004    304  USER SEGMENT
077155     1      000000     000053    062304    000024    304  USER SEGMENT
077131     1      000000     000000    140041    000004     41

$$$$$$$$ DST  132 $$$$$$$$
******PCBX: ************************************************************************************************
***PXGLOBAL:
067423: 001044 001200 173407 000424 001024 126116 016115 000000
***PXFIXED:
067433: 000120 006415 012560 006302 000134 000612 000000 000004   067443: 000000 000000 000000 000000 000304 051040 000000 000000
067453: 000000 000000 112703 040002 054000 000000 000000 010714   067463: 000000 005255 000344 002202 000000 000000 000000 000000
067473: 000000 000000 000000 000001 000000 000000 000000 000000   067503: 000000 000000 000000 004641 005255 000000 000000 000000
067513: 000000 000000 000000 000000 000000 000000 000000 000000   067523: 000000 000000 000000 000000 000000 000004 000000 000000
067533: 000000 000000 000000 000000 000000 000000 000000 000000   067543: 000000 000000 000000 000000 000000 000000 000000 000000
***PXFILE: (ZERO TABLE ENTRIES ARE NOT PRINTED)
067553: 000710 000000 000000 000000 000000 000050 000000 000000   067563: 000000 000000 000000 000000 000000 000000 000000 000000
067573: 000470 000132 000100 000000 000000
------- FILE VECTOR TABLE:         ENTRY  ADDRESS  LOCK  BRK  LOCK COUNT/PIN   HIPRI TAIL  HIPRI HEAD  LOPRI TAIL  LOPRI HEAD
067600: 000106 100454 000000 000000   0     106    LOCK             1  54
067604: 000126 100454 000000 000000   1     126    LOCK             1  54
067610: 000146 100454 000000 000000   2     146    LOCK             1  54
067614: 000166 000000 000000 000000   3     166                     0   0
067620: 000246 000000 000000 000000   4     246                     0   0
067624: 000355 100454 000000 000000   5     355    LOCK             1  54
067630: 000406 100454 000000 000000   6     406    LOCK             1  54
067634: 000437 100454 000000 000000   7     437    LOCK             1  54   --> ADDR INTO CONTROL BLOCK
------- CONTROL BLOCKS:
067700(000105): 000001 140020 000000 022123 052104 044516 020040 002244 001700 000120 000050 000000  067700:......$STDIN .....P.(
067714(000121): 000430 000020 000000 000000 177773 140020 000002 022123 052104 046111 051524 002414  067714:..............$STDLIS
067730(000135): 001701 000121 000051 000000 000000 000000 000000 000000 177766 110020 000003 041517  067730:...Q.)...................
067744(000151): 046515 040516 042040 000255 001400 000120 000050 000000 000000 000022 000000 000000  067744:MMAND .....P.(.......
067760(000165): 000000 100060 000004 041501 052101 046117 043440 000001 000420 000120 001200 000000  067760:...0..CATALOG .....P .
067774(000201): 000000 004000 000000 000000 000000 000000 000000 000000 000000 000000 000000 000000  067774:........................
070010(000215): 000000 177777 177777 002112 000000 000000 000000 000020 000000 000000 000000 000000  070010:.........J..............
070024(000231): 000000 000001 000000 037001 000000 000000 000000 000000 000000 000000 000000 000000  070024:.......>................
070040(000245): 000000 100060 000005 040517 047103 040524 020040 000005 000420 000120 001200 000000  070040:...0..MONCAT .....P.
070054(000261): 000000 004000 000000 000000 001200 000000 000040 000000 000000 000000 000220 000000  070054:........................
070070(000275): 000011 177777 177777 006112 000000 000000 000001 000020 000000 000000 000000 000000  070070:.........J..............
070104(000311): 000000 000001 001000 077401 000000 000000 000000 000000 000000 000000 000000 000000  070104:........................
070120(000325): 000000 000027 000010 045514 047516 042440 020040 002001 000767 000400 000200 000000  070120:.......KLONE ..........
070134(000341): 000000 004000 000000 000000 006300 000000 000033 000000 000000 000000 000033 000000  070134:........................
```

Dump Example



```
DST 137

PORT:  HEAD THREAD (%16 OVERHEAD) = 220      22
                            NEXT = 210      21
                                   200      20
                                   150      15
                                   140      14
                                   130      13

                                   120      12
                                   110      11
                                   100      10
                                    70       7
                                    40       4
                                    30       3
                                    20       2
                                    00

PORT 40:  MQE = 50 (TAIL/HEAD SAME)
          PORT MGR PIN = 55
          SOFT INT SUBTYPE = 1
          SOFT INT PLABEL = 107304

 MQE 50:  NEXT MQE = 47  (40 LOR 7)
          RETURN PORT = 4
          TIME LIST ENTRY = 1, TIME OUT OCCURED

PORT 130:  MQE = 160 (TAIL/HEAD SAME)
           RETURN PORT = 15
           TIME LIST ENTRY = 170

           TLE 170:  NEXT TLE 0 IF LAST
```

Dump Example



```
HP3000 III MEMORY DUMPC.00.02 OF SYS VER C  UPDATE  01  FIX 06  DUMP TIME 10/18/81,  4:01PM                BANK   2
(C) HEWLETT-PACKARD CO. 1980

151763(000140): 020040 020040 020040 020040 020040 020040 020040 020040 020040 030060 030061 030060 030060  151763:                00010000
151777(000154): 042517 043040 020040 020040 020040 020040 020040 020040 020040 020040 020040 020040 020040  151777:EOF
152013(000170): 020040 020040 020040 020040 020040 020040 020040 020040 020040 020040 020040 020040 020040  152013:
LINES 152027 - 152042 SAME AS ABOVE
152043(000220): 030080 030061 030480 030080 030060 030060 030060 030060 030060 030060 030060 030060 030060  152041:0001100000000000000000000
152057(000234): 030060 030060 030060 030060 030060 030060 030060 030060 030060 030060 030060 030060 030060  152057:000000000000000000000000
LINES 152073 - 152106 SAME AS ABOVE
152107(000284): 030060 030060 030060 030060 030060 030060 030060 030060 030060 000000 000000 000000 000000  152107:000000000000000000........
152123(000300): 177777 177777 000000 000000 000000 000000 000000 000000 000000 000000 000000 000000 000000  152123:........................
152137(000314): 000000 000000 000000 000000 000000 000000 000000 000000 000000 000000 000000 000000 000000  152137:........................
LINES 152153 - 152372 SAME AS ABOVE
152373(000550): 000000                                                                                       152373:..
152374(000551): 000000 000003 100000 000003                                                                  152374:........

152400: 100000 000038 000000 110001 013744 100000 026054 000000  152410: 000000 000038 100000 000000 040137 000000 000000 000400
152420: 031405 000000 000000

$$$$$$$$ CST 137                                $$$$$$$$
**** (152423 TO 181773 NOT PRINTED) ****
181774: 051407 000038 100000 000038

182000: 100000 000018 000000 110001 018124 100000 026054 000000  182010: 000000 000018 100000 000000 040014 000000 000000 000400
182020: 004050 000000 000000

$$$$$$$$ CST  14                                $$$$$$$$
**** (182023 TO 185373 NOT PRINTED) ****
185374: 051401 000018 100000 000018

185400: 100000 000051 000000 110001 017404 140000 028455 000000  185410: 000000 000051 100000 000000 000132 000000 000000 000400
185420: 021504 000000 000000

                    [******    PCBX AND STACK MARKERS FOR DST 133  (PCB 55 )    ******]

SEG REL  SEG REL    JMAT      JPCNT     JOB INPUT    JOB OUTPUT   JDT DST   JIT DST                                          JCUT
DL       DB         INDEX     INDEX     LOG DEV #    LOG DEV #    INDEX     INDEX     JOB TYPE   DUPLICAT  INTERACT  INIT Q   INDEX
000044   001000       1         2          20           20        126118      115       BS2        YES       YES     006302      0

ADDRESS    BANK        X        DELTA P   STATUS     DELTA Q     SEGMENT

175038       2       177758     020003    101074     000012       74
175024       2       170520     018638    141074     000014       74
175010       2       006337     001725    142438     000020       36
174770       2       000041     004074    062304     000007      204   USER SEGMENT
174781       2       000041     000207    062304     000004      204   USER SEGMENT
174755       2       000000     000053    062304     000024      304   USER SEGMENT
174731       2       000000     000000    140041     000004       41
```

```
HP3000 III MEMORY DUMPC.00.02 OF SYS VER C  UPDATE  01  FIX 08  DUMP TIME 10/16/81,  4:01PM                 BANK   2
(C) HEWLETT-PACKARD CO. 1980

$$$$$$$$ DST  133 $$$$$$$$
******PCBX: *****************************************************************************************************
***PXGLOBAL:
165423: 000044 001000 173407 000424 001024 126110 010115 000000
***PXFIXED:
165433: 000120 000415 010500 006302 000134 000013 000000 000004   165443: 000000 000000 000000 000000 000304 050640 000000 000000
165453: 000000 000000 000000 040002 054000 000000 000000 010714   165463: 000000 001132 000344 000666 000000 000000 000000 000000
165473: 000000 000000 000000 000001 000000 000000 000000 000000   165503: 000000 000000 000000 001132 001132 000000 000000 000000
165513: 000000 000000 000000 000000 000000 000000 000000 000000   165523: 000000 000000 000000 000000 000000 000005 000000 000000
165533: 000000 000000 000000 000000 000000 000000 000000 000000   165543: 000000 000000 000000 000000 000000 000000 000000 000000
***PXFILE: (ZERO TABLE ENTRIES ARE NOT PRINTED)
165553: 000510 000000 000000 000000 000000 000044 000000 000000   165563: 000000 000000 000000 000000 000000 000000 000000 000000
165573: 000410 000131 000100 000000 000000
------- FILE VECTOR TABLE:                  ENTRY   ADDRESS   LOCK  BRK  LOCK COUNT/PIN    HIPRI TAIL  HIPRI HEAD  LOPRI TAIL  LOPRI HEAD
165600: 000106 100455 000000 000000           0        106    LOCK                1  55
165604: 000126 100455 000000 000000           1        126    LOCK                1  55
165610: 000146 100455 000000 000000           2        146    LOCK                1  55
165614: 000166 000000 000000 000000           3        166                        0   0
165620: 000246 000000 000000 000000           4        246                        0   0
165624: 000326 100455 000000 000000           5        326    LOCK                1  55
165630: 000357 100455 000000 000000           6        357    LOCK                1  55
------- CONTROL BLOCKS:
165700(000105): 000001 140020 000001 022123 052104 044516 020040 002244 001700 000120 000050 000000   165700:......$STDIN  .....P.(..
165714(000121): 000000 000020 000000 000000 000000 140020 000002 022123 052104 040111 051524 002014   165714:...............$STDLIST..
165730(000135): 001701 000121 000051 000000 000000 000000 000000 000000 000000 140020 000003 041517   165730:...O.)...................CO
165744(000151): 046515 040516 042040 000255 001400 000120 000050 000000 000000 000022 000000 000000   165744:MMAND  .....P.(..........
165760(000165): 000000 100060 000004 041501 052101 040117 043440 000001 000420 000120 001200 000000   165760:...O..CATALOG  .....P....
165774(000201): 000000 004000 000000 000000 000000 000000 000000 000000 000000 000000 000000 000000   165774:........................
166010(000215): 000000 177777 177777 002112 000000 000000 000000 000020 000000 000000 000000 000000   166010:........J...............
166024(000231): 000000 000001 000000 037001 000000 000000 000000 000000 000000 000000 000000 000000   166024:.......>................
166040(000245): 000000 100060 000005 046517 047103 040524 020040 000005 000420 000120 001200 000000   166040:...O..MONCAT  .....P....
166054(000261): 000000 004000 000000 000000 000000 000000 000000 000000 000000 000000 000000 000000   166054:........................
166070(000275): 000000 177777 177777 000112 000000 000000 000000 000020 000000 000000 000000 000000   166070:........J...............
166104(000311): 000000 000001 001000 077401 000000 000000 000000 000000 000000 000000 000000 000000,  166104:........................
166120(000325): 000000 140031 000010 046523 043462 020040 020040 032001 002100 000400 000205 000000,  166120:......MSG2    4..@......
166134(000341): 000000 000000 000000 000000 000000 107304 000002 000000 000004 000000 000000 000000   166134:........................
166150(000355): 000000 000000 140031 000011 046523 043461 020040 020040 032001 002100 000400 000205   166150:........MSG1    4..@....
166164(000371): 000000 000000 000000 000000 000000 000000 107304 000000 000000 000011 000000 000000   166164:........................
166200(000405): 000000 000000 000000                                                                  166200:......
166203: 000000 000000 000000 000000 000000 000000 000000 000000   166213: 000000 000000 000000 000000
------- AVAILABLE FILE TABLE:          FNUM   FTYPE  FNULL    PACB V     LACB V    IOQX
166217: 100000 000124 014133 177776     11     10              0  124     6  133   177776   -- MSG 1
166223: 100000 000135 012133 177774     10     10              0  135     5  133   177774   -- MSG 2
166227: 000000 000125 000000 000000      7     FILE            0  125     0    0
166233: 000000 000120 000000 000000      6     FILE            0  120     0    0
166237: 000000 010133 000000 000000      5     FILE            4  133     0    0
166243: 000000 006133 000000 000000      4     FILE            3  133     0    0
166247: 000000 000117 004133 000000      3     FILE            0  117     2  133
166253: 000000 000117 002133 000000      2     FILE            0  117     1  133
166257: 000000 000117 000133 000000      1     FILE            0  117     0  133
**PXPOINTERS:
166263: 000000 000514 000634 000644
****DL REGISTER: *************************************************************************************************
166267(177644): 000000 000000 000000 000000 000000 000000 000000 000000 000000 000000 000000 000000   166267:........................
LINES 166303 - 166412 SAME AS ABOVE
166413(177770): 100704 000000 177777 000000 000000 177777 000000 177777                              166413:................
```

```
HP3000 III MEMORY DUMPC.00.02 OF SYS VER C  UPDATE  01  FIX 08  DUMP TIME 10/10/81,  4:01PM              BANK   3
(C) HEWLETT-PACKARD CO. 1980

050228(000137): 177777 001200 001200 023730 100074 000007 000000 000000 001000 013214 000001 000001  050228:......'..4..............
050242(000153): 001003 024124 100074 000012 000000 000000 000230 000000 177777 000000 000000 000000  050242:..(T.<..................
050256(000167): 000000 000000 000000 000000 000000 000000 000000 000000 000000 000000 000000 000000  050256:........................
LINES 050272 - 052661 SAME AS ABOVE
052682(002573): 000000 000000 000000 000000 000000                                                052682:..........

052667: 000000 000000 000000 000000 000000 000000 000000 000000  052677: 000000 000000 000000 000000 000000 000000 000000 000000
052707: 000000 000000 000000 000000 000000 000000 000000 000000  052717: 000000 000000 000000 000000 000402 000000 000000 000401
052727: 036735 000000 000000 000000 000000 000000 000000 000000  052737: 000000 000000 000000 000000 000000 000000 000000 000000
052747: 000000 000000 000000 000000 000000 000000 000000 000000  052757: 000000 000000 000000 000000 000000 000000 000000 000000
052767: 000000 000000 000000 000000 000000
052774: 000000 000016 100000 000016

053000: 100000 000032 000000 110001 017344 100000 024050 000000  053010: 000000 000032 100000 000000 040031 000000 000000 000400
053020: 007475 000000 000000

$$$$$$$$ CST  31                    $$$$$$$$
**** ( 53023 TO  61373 NOT PRINTED) ****
061374: 041605 000032 100000 000032

061400: 100000 000004 000000 110001 020404 100000 020455 000000  061410: 000000 000004 100000 000000 000135 000000 000000 000400
061420: 020400 000000 000000
                          overhead
$$$$$$$$ DST  135 $$$$$$$$
061423(000000): 000004 000135 140004 000000 000000 000015 000000 000000 000000 000004 113032 000003  061423:...1....................
061437(000014): 000000 100537 000010 046573 043402 020040 020040 032001 002100 000400 000205 000000  061437:...w..MS02    4..@......
061453(000030): 000000 000000 000000 000000 000000 000000 000000 000000 000000 000000 000000 000000  061453:........................
061467(000044): 000000 000386 000386 022112 000000 000401 000000 000001 000001 000162 000215 000000  061467:.......$J.............r....
061503(000060): 000010 000001 001000 037401 004201 000000 002007 000000 000162 000000 000000 000001  061503:.......?..............r.....
061517(000074): 000000 177777 177777 177777 177777 177777 177777 177777 177777 177777 177777 177777  061517:........................
061533(000110): 177777 177777 177777 177777 177777 000000 000000 000000 000000 000000 000000 000000  061533:........................
061547(000124): 000000 000000 000001 000002 000000 000000 000000 000000 000203 000000 000000 000000  061547:........................
061563(000140): 000000 000000 000000 000000 000000 000000 000000 000000 000162 000377 000000 000000  061563:...................r....
061577(000154): 000000 000000 000000 000000 000000 000000 000000 000000 000000 000000 000000 000000  061577:........................
LINES 061613 - 062222 SAME AS ABOVE
062223(000600): 000000 000000 000000 000000 000000 000000 000000 000000 046517 051122 044523 020040  062223:................MORRIS
062237(000614): 044120 042440 020040 020040 000000 000000 000000 000000 000000 000000 000401 030735  062237:HPE     ..............=.
062253(000630): 000000 000000 000000 000000 000000 000000 000000 000000 000000 000000 000000 000000  062253:........................
062267(000644): 000000 000000 040052 000000 002001 177772 000011 000401 000100 000001 000000 000000  062267:....@*............@.....
062303(000660): 000000 000002 000000 000000 000000 000002 000000 000041 000441 020402 000041 000001  062303:.................!.!!..!..
062317(000674): 050125 041040 020040 020040 051531 051440 020040 020040 000000 000000 000000 000000  062317:PUB     SYS     ........
062333(000710): 000000 000000 000401 031201 000401 031242 000401 031303 000000 000000 000704 177773  062333:......2...2...2..0......
062347(000724): 000011 000401 000128 000000 000000 000000 000000 000103 000000 000000 000000 000103  062347:.....V..........C.......C
062363(000740): 001033 000050 000404 016000 000050 000000 046517 051122 044523                    062363:...(.....(..MORRIS
062374(000751): 020040 000004 100000 000004                                                       062374:  ......

062400: 100000 000001 000000 110001 016484 100000 005012 000000  062410: 000000 000001 100000 000000 000045 000000 000000 000400
062420: 016334 000000 000000

$$$$$$$$ DST   45 (SYSTEM JIT) $$$$$$$$
```

Dump Example

```
HP3000 III MEMORY DUMPC.00.02 OF SYS VER C  UPDATE  01  FIX 00  DUMP TIME 10/15/81,  4:01PM                BANK   3
(C) HEWLETT-PACKARD CO. 1980

$$$$$$$$ CST 101                          $$$$$$$$
**** (104623 TO 115173 NOT PRINTED) ****
115174: 000600 000042 100000 000042

115200: 100000 000056 000000 110001 016584 100000 000401 000000  115210: 000000 000086 100000 000000 040077 000000 000000 000400
115220: 012347 000000 000000

$$$$$$$$ CST  77                          $$$$$$$$
**** (115223 TO 132573 NOT PRINTED) ****
132574: 001100 000086 100000 000088

132600: 100000 000004 000000 110001 020624 100000 028455 000000  132610: 000000 000004 100000 000000 000124 000000 000000 000400
132620: 020710 000000 000000

$$$$$$$$ DST  124 $$$$$$$$
132623(000000): 000604 000124 140004 000000 000000 000015 000000 000040 000000 000004 012724 001736  132623:...T.....................
132637(000014): 0001341100567 000010 048523 043461 020040 020040 032001 002100 000400 000205 000000  132637:.\.W..MSGi    4..C.....
132653(000035): 000000 000000 000000 000000 000400 000000 000002 000000 000001 000000 000001 000000  132653:........................
132667(000044): 000001 000368 000001 024112 000000 000402 000000 000001 000401 000162 000215 000000  132667:.....(J.............r....
132703(000060): 000014 000001 001000 037401 004217 000000 002004 000000 000600 000000 000001 000401  132703:.....T...................
132717(000074): 000000 077777 177777 177777 177777 177777 177777 177777 177777 177777 177777 177777  132717:........................
132733(000110): 177777 177777 177777 177777 177777 177777 000000 000000 000000 000000 000000 000000  132733:........................
132747(000124): 000000 000000 000010 000007 000000 000000 000000 000000 000203 000000 100000 000000  132747:........................
132763(000140): 000000 177400 000000 000055 000011 107304 000000 177400 000162 000377 000000 000002  132763:.....-.........r.....
132777(000154): 000000 000000 000000 000000 000000 000000 177777 000000 000000 000000 000000 000000  132777:........................
133013(000170): 000000 000000 000000 000000 000000 000000 000000 000000 000000 000000 000000 000000  133013:........................
LINES 133027 - 133202 SAME AS ABOVE
133203(000360): 000000 000000 000000 000000 000077 000001 000001 040120 000005 000000 000000 000401  133203:.........?....@P........
133217(000374): 036277 000000 000000 000400 002108 044514 046004 001111 042003 000514 002403 042522  133217:<.........FILL..ID..L..ER
133233(000410): 051000 002403 041503 043403 000522 002403 042522 051000 001401 046405 001505 051122  133233:R...CCG..R..ERR...M..ERR
133247(000424): 000105 002403 047105 053405 001505 051122 000111 002402 041503 043403 000504 002403  133247:.E..NEW..ERR.I..CCG..D..
133263(000440): 042522 051000 002403 041503 043408 002127 040511 052005 001505 051122 000105 002403  133263:ERR...CCG..WAIT..ERR.E...
133277(000454): 041503 043405 001505 051122 000003 002402 041503 043408 002108 044514 046003 000514  133277:CCG..ERR....CCG..FILL..L
133313(000470): 002403 042522 051000 002403 041503 043406 002114 044523 052005 001526 040514 002403  133313:...ERR...CCG..LIST..VAL..
133327(000504): 042522 051000 002403 041503 046003 000511 002403 044516 052006 002102 054524 042405  133327:ERR...CCL..I..INT..BYTE..
133343(000520): 001504 041114 004006 051524 051111 047107 000111 003004 050122 047507 003004 044516  133343:.DBL..STRING.I..PROG..IN
133357(000534): 042117 000005 003004 050122 000000 000000 000000 000000 000000 000000 000000 000000  133357:FD....PR..................
133373(000550): 000000 000000 000000 000000 000000 000000 000000 000000 000000 000000 000000 000000  133373:........................
LINES 133407 - 133422 SAME AS ABOVE
133423(000600): 177777 000077 000001 000000 000000 000000 000000 000000 020040 020040 020040 020040  133423:...?.....................
133437(000614): 020040 020040 020040 020040 020040 020040 020040 020040 020040 030064 030065 032460  133437:                  04055000
133453(000630): 032460 020105 047104 020117 043040 050122 047507 051101 046440 020040 020040 020040  133453:50 END OF PROGRAM
133467(000644): 020040 020040 020040 020040 020040 020040 020040 020040 020040 020040 020040 020040  133467:
LINES 133503 - 133516 SAME AS ABOVE
133517(000674): 030064 030065 032460 030060 032461 020105 047104 020117 043040 050122 042520 040522  133517:0405700051 END OF PREPAR
133533(000710): 042440 020040 020040 020040 020040 020040 020040 020040 020040 020040 020040 020040  133533:E
133547(000724): 020040 020040 020040 020040 020040 020040 020040 020040 020040 020040 020040 020040  133547:
133563(000740): 020040 020040 020040 020040 030064 030065 034460 030060 032462                        133563:         0405900052
133574(000751): 020105 000004 100000 000004                                                            133574: E.......

133600: 020000 000003 054523 000000 000000 174623 000000 000000  133610: 020040 000003 000000 020040 000000 020040 020040 020040
```

```
HP3000 III MEMORY DUMPC.00.02 OF SYS VER C  UPDATE  01  FIX 00  DUMP TIME 10/16/81,  4:01PM              BANK   3
(C) HEWLETT-PACKARD CO. 1980

$$$$$$$$ DST  117 $$$$$$$$
165223(000000): 000074 000117 140004 000000 000000 000014 000000 000000 000000 000003 037417 004300   165223:..t.O.....................?...
165237(000014): 100060 000003 041517 046515 040516 042040 000255 001400 000120 000050 000000 000000   165237:.0...COMMAND.....P.(.....
165253(000030): 000000 000000 000000 177766 000000 004520 000000 000000 000000 004520 000000 004520   165253:..............P.........P...P
165267(000044): 177777 177777 000000 000000 002410 010001 030001 000000 000000 000000 000000 000004   165267:..............0...........
165303(000060): 000016 001400 037024 000000 000000 000000 000000 000000 000000 000000 000000 000000   165303:....}.....................
165317(000074): 047408 037777 061074 141503 000004 004543 102033 000011 000020 000001 177134 000000   165317:G.?.b<.C...e.........\...
165333(000110): 002250 000000 000001 022057 100433 000010 000000 020220 000000 002404 000203 001460   165333:.....$/.....  .........0
165347(000124): 000000 177134 020220 000000 000013 017700 000014 000000 000303 000000 002404 000303   165347:....\ ...................
165363(000140): 001600 080413 003645 000003 000576 102033 000031 000014 000000                        165363:..a.....~.........
165374(000151): 000303 000001 100000 000001                                                           165374:.........

165400: 100000 000041 000000 110001 020244 100000 021443 000000  165410: 000000 000041 100000 000000 040040 000000 000000 000460
165420: 010075 000000 000000

$$$$$$$$ CST  40                    $$$$$$$$
**** (165423 TO 175573 NOT PRINTED) ****
175574: 141202 000041 100000 000041

175600: 100000 000004 000000 110001 015704 100000 026054 000000  175610: 000000 000004 100000 000000 000134 000000 000000 000400
175620: 020714 000000 000000

$$$$$$$$ DST  134 $$$$$$$$
175623(000000): 000004 000134 140004 000000 000000 000015 000000 000000 000000 000004 000000 000000   175623:...\.....................
175637(000014): 000000 100587 000011 046523 041461 020040 020040 032001 002100 000400 000205 000000   175637:....w...HSQ3    4..@.....
175653(000030): 000000 000000 000000 000000 000000 000000 000000 000000 000000 000000 000000 000000   175653:........................
175667(000044): 000000 000388 000386 026112 000000 000401 000000 000001 000001 000182 000215 000000   175667:.......J.............f....
175703(000060): 000020 000001 001000 037401 004201 000000 002007 000000 000182 000000 000000 000001   175703:.......?.........f.......
175717(000074): 000000 177777 177777 177777 177777 177777 177777 177777 177777 177777 177777 177777   175717:........................
175733(000110): 177777 177777 177777 177777 177777 000000 000000 000000 000000 000000 000000 000000   175733:........................
175747(000124): 000000 000000 000014 000013 000000 000000 000000 000000 000202 000000 000000 000000   175747:........................
175763(000140): 000000 000000 000000 000000 000000 000000 000000 000000 000000 000182 000377 000000   175763:..................f.....
175777(000154): 000000 000000 000000 000000 000000 000000 000000 000000 000000 000000 000000 000000   175777:........................
LINES 176013 - 176422 SAME AS ABOVE
176423(000300): 000000 000000 000000 000000 000415 000415 000415 000415 000020 000000 000000 000000   176423:........................
176437(000314): 000000 000000 000000 000000 000830 000000 000000 000000 000000 000000 000000 000000   176437:........................
176453(000330): 000000 000000 000000 000000 000000 000000 000000 000000 177777 177777 177777 177777   176453:.0......................
176467(000344): 000000 000000 000000 000000 000260 000000 000000 000000 000000 000000 000000 000000   176467:........................
176503(000360): 000270 000000 000000 000000 000000 000000 000000 000000 000300 000000 000000 000000   176503:........................
176517(000374): 000000 000000 000000 000000 000310 000000 000000 000000 000000 000000 000000 000000   176517:........................
176533(000710): 000320 000000 000000 000000 000000 000000 000000 000000 000320 000000 000000 000000   176533:........................
176547(000724): 000000 000000 000000 000000 000340 000000 000000 000000 000000 000000 000000 000000   176547:........................
176563(000740): 000350 000000 000000 000000 000000 000000 000000 000000 000360                        176563:.....................
176574(000751): 000000 000004 100000 000004                                                           176574:.........

176600: 020000 000003 000000 000000 000000 000000 000000 174623  176610: 000000 000003 000000 000000 000000 000000 000000 000400
176620: 021080 000000 000000

$$$$$$$$ AVAILABLE AREA $$$$$$$$
**** (176623 TO 177373 NOT PRINTED) ****
```

MMSTAT DEFINITIONS

APPENDIX C

If problems arise using soft interrupts, the monitor facility can be used to analyze the specific steps that have occurred when using soft interrupts. By enabling MMSTAT (soft interrupts) before the execution of the application, specific information can be logged to tape. Once the application has executed the logging tape can be analyzed to assist in determining the existing problem. The MPE Tables Manual contains the definitions of the entries in the printed log file.

Octal Value	Event Type	Parameter 0	Parameter 1	Parameter 2
62	Open	Port Number	Port DST No.	Flags Parameter
63	Receive Completion	Port Number	MQE Address 15:1 Waitspc	Return Port
64	Send	Port Number	MQE Address 15:1 Q Type	Return Port
65	Change Status	Port Number	0 = Enable 1 = Disable	Head MQE Address
66	Abort	Port Number	Parameter Zero	Return Port
67	Close	Port Number	Port DST	# Open Ports Left
70	Expand	Port DST No.	# Expanded Blocks	Total # of Blocks
71	Timeout Expired	Port Number	MQE Address	Return Port

Octal Value	Event Type	Parameter 1	Parameter 2
72/0	Read Init	# free rec	
72/1	Read Compl	(0:8) error, (8:8) ID	No of Records
72/2	Write Init	(0:8) # rec, (8:8) ID	No of Free Records
72/3	Write Compl	(0:8) error, (8:8) ID	No of Free Records
72/4	Control	(0:8) error, (8:8) ID	(0:4) func, (4:12) parameters
72/5	EOF	(0:8) error, (8:8) ID	No of Records
72/6	Open	(0:8) error, (8:8) ID	No of Records
72/7	Close	(8:8) #Free, (8:8) ID	No of Records
72/10	Initiation	0	(0:8) fix, (8:8) update
73/0	Put Record	(0:8) error, (8:8) ID	(0:3) record type, (3:13) no of records
73/1	Delete Rec	(0:8) error, (8:8) ID	(0:3) record type, (3:13) no of records
73/2	Delete Block	Start of file block #	End of file block #

Notes:

1. The aa/bb notation in the "octal value" column denotes type/subtype. Type is the actual MMSTAT event number. Subtype is (0:4) of parameter 0.

2. Several items can possibly exceed their fields, in that case the bits beyond the field are lost. These items are number of records, number of free records, start of file, and end of file.

3. Parameter word zero has a common format for all the MMSTAT events.

Field	Description
(0:4)	Event's subtype
(4:2)	File's state 0 = empty 1 = partially full

2 = only a fraction of a free record is left
3 = completely full

(6:1) Nonzero indicates that there is one or more waiting readers

(7:1) Nonzero indicates that there is one or more waiting writers

(11:1) Nonzero indicates that the write has a carriage control character

(12:4) Flags local to the accessor
(12:1) - the accessor has done no FREADs/FWRITEs
(13:1) - extended wait
(14:1) - nondestructive read
(15:1) - writer has not written any records

Soft Interrupt MMSTAT Entry Forms

MMSTAT tracing for the following types is enabled if the "monitoring" has been turned on by the MON command.

Type	Number	Word 0	Word 1	Word 2
Cause Soft Interrupt	240	(0:4) level (4:2) type (6:2) stype (8:8) pin	Msg Word One	Msg Word two
Pseudoint	241	(0:8) type (8:8) stype	Msg Word One	Msg Word Two
Build Stack	242	Plabel	P-Reg Word of Prev Marker	Status Word of Prev Marker
Change State	243	(14:1) old state (15:1) new state	User's P Register	User's Status Register
Timeout	244	completion type 0 timeout 1 no trlx 2 control y soft int	16 msb of timeout (ms)	16 lsb of timeout (ms)

Where:

Level — The current status of the interrupt/process

0 process is dying (hard kill or soft kill)

1 other soft interrupts are pending

2 user interrupts are disabled

3 process is impeded, critical, and/or with SIR

4 interrupt was against own process

5 process was either waiting or ready to execute

MPE MEMORY RESIDENT MESSAGE FACILITY

APPENDIX D

The memory resident message facility of MPE IV addresses the need for an efficient, simple, and uniform message method for system code to send short status-type messages to processes.

Each process is created with a message harbor which supports a set of message ports which are private to that process. There is a maximum of four ports per harbor in the initial implementation. This limit can be easily extended when new ports are required.

Any system code, even code running on the ICS, can send a message to any port of any process. The destination process's PIN must be known, and a priority conventions on port number and message formats must be established. The caller of SENDMSG may optionally specify that the destination process be awakened from a message wait.

The caller of SENDMSG specifies whether the message is to be buffered in the primary message table or the secondary message table. When the secondary table is specified, if the pool of secondary entries is exhausted, the calling process is queued for a message table entry and blocked until one becomes available. Use of the primary message table is reserved for code running on the ICS or during critical sections (Pdisabled or Disabled intervals) in which it is not possible to release control of the processor to queue for a free message table entry. If the primary table is specified and no free entries are available, the SENDMSG crashes the system.

Messages can be of any length up to the configured maximum. Message length is specified in the call to SENDMSG and RECEIVEMSG. In the initial implementation, messages are limited to four words in length. This maximum can be easily increased if the need arises.

By calling PORTSTATUS, a process may at any time determine whether a specified port is non-empty or obtain the port number of the most urgent non-empty port (lowest numerical port number = most urgent port).

By calling RECEIVEMSG, a process may received the message at the head of the specified message port. This receive is optionally nondestructive.

A process can wait on a message wait, or on a combination of a message wait and other wait types.

Message Intrinsics

A. Procedure SENDMSG (Destpin, Destport, Msglength, Flags);
Value Destpin, Destport, Msglength,Flags;
Integer Destpin, Destport, Msglength;
Option Priviledged, Uncallable;
Logical Flags;

Destpin, Destport and Msglength has better be within range and reasonable (process and port exist), since SENDMSG checks and will crash if the parameters are bad.

The caller of SENDMSG stacks the message contents before calling the procedure. SENDMSG expects the first message word to be at Q-7-Msglength, and the last message word at Q-8. The message contents at Q-8 to Q-7-Msglength are deleted from top of stack by the exit from SENDMSG to the caller.

Flags.(1:1) = 1 ==> Wake-up destintation process from a message wait
(0:1) = 1 ==> Place message in secondary message table

Return CC=CCG if process was already awake else CC=CCE.

B. Logical Procedure PORTSTATUS (Portnumber);
Value port number;
Integer port number;
Option Priviledged, Uncallable;

When supplied a valid port number, PORTSTATUS returns a true value if the port is non-empty and a false value if the port is empty.

When passed a -1 as port number parameter, PORTSTATUS returns the portnumber of the process's most urgent non-empty port (the smaller the number, the more urgent the port).

If all ports are empty, PORTSTATUS returns CC=CCE. If at least one port is non-empty, PORTSTATUS returns CC-CCG.

C. Procedure RECEIVEMSG (Portnum, Msglength,Flags);
Value Portnum,Msglength, Flags;
Integer Portnum, Msglength;
Option Priviledged, Uncallable;

Portnum and Msglength had better be within range or else its suddendeath time.

The caller of RECEIVEMSG does an ADD S Msglength to make space for the message contents. RECEIVEMSG stores the message contents into Q-8, Q-9,.....Q-7-Msglength.

```
Q-7-Msglength contains the first word of the message.
Flags.(0:1) = 1 ==> do not release message from head of
                    port's message queue (non-destructive
                    read)

Return CC=CCG if port was empty, else CC:=CCE.
```

FILE SYSTEM BASIC IPC DEFINITIONS

APPENDIX E

The objective of this set of uncallable procedures is to provide a simple IPC mechanism to support the IPC file access procedures. It enables one process to send short, control messages to another processes.

General Behavior

FCPORTOPEN Procedure

The heart of this mechanism is the port. A process desiring to receive messages would first open (create) a port. This process is termed the "port manager". When the port is created, a port number is returned to the opener. Since the port number value cannot be known in advance, potential senders need some method of obtaining the port number from the port manager.

Both the ports and the messages are contained in a single disc resident data segment. There can be a total of over 3,500 open ports and outstanding messages. Thus neither ports nor message blocks are scarce resources.

FCPORTSEND Procedure

This procedure sends a 0-to 5-word message to a port. Optionally a timeout value may be specified which will limit the duration the message will remain attached to the port. Expiration of the time-out causes the message to be deleted from the target port's queue and placed on the sender's reply port (specified by the sender in the FCPORTSEND procedure call).

FCPORTRECEIVE

Reads and deletes the head message from a port. The sender's return port number is also given to the receiver, enabling him to send a reply message.

FCPORTCLOSE

Demolishes the port.

IPC File's Use of This Mechanism

All open message files have two ports open for the file (read wait queue and write wait queue), plus one port per accessor (reply port). Their use is described in the following.

Reader and Writer Wait Queues

When an empty message file is accessed by more than one reader (share), then there must be a way of having the readers' FREADs satisfied in the same order that they were issued. That is, there must be a queue of waiting readers. The IPC procedures acccomplish this by dedicating a basic IPC port as a "read wait queue". Whenever a reader's request is stalled because the file is

empty, a message is sent to the read wait queue. Subsequent FREADs by other processes will queue up behind the first reader in an FIFO manner. An FWRITE will take the first entry from the wait queue and send a "read may be done" message to the reader's reply port. In a like manner multiple writers will queue on the write wait queue when the file is full.

Completion Notification for NoWait I/O

The IOWAIT intrinsic waits for a message to be sent to the reply port(s) of the specified user files.

Timeouts

When an accessor encounters a boundary condition (e.g. a reader accesses an empty file), it may specify that the condition must be satisfied in x seconds (FCONTROL 4). To this end the IPC access procedures merely issue the FCPORTSEND to the wait queue with the user's timeout value specified. The timeout will tear the message from the wait queue and place it on the accessor's reply port.

SYSTEM FAILURES

APPENDIX F

BIPC - MODULE 65

System Failure 690

UGLYPORTPROBLEM is the equate used to describe system failure 690. System failure 690 can occur from 21 different code statements. The main description of this sys- tem failure is a problem which occurred during accessing a port. This could be an incorrect Port DST number, incorrect head entry, or an inconsistent number of MQE's between procedures.

IPC - MODULE 66

System Failure 495

UGLYMSGACCESS is the equate used to describe system failure 495. Calls to suddendeath using UGLYMSGACCESS can occur from 13 different locations. The causes of this system failure, buff- er addresses incorrect, or a wait queue address is incorrect.

KERNALC - MODULE 92

System Failure 4

This system failure is common throughout KERNALC. Because this system failure is common in KERNELC it is redefined within the soft interrupt procedures within KERNELC (which means the crossreference separates calls to suddendeath specific to soft interrupts). Causes of System Failure 4 are calling parameters incorrect or problems with messages (don't exist, etc.).