COMPUTATION CENTER
Massachusetts Institute of Technology
Cambridge 39, Massachusetts

TO All CTSS Users

FROM R.C. Daley, R.J. Creasy and R.M. Graham

SUBJECT A Generalized File Structure and Input/Output System

INTRODUCTION

This paper is a proposal of a general solution to the data-handling problem existing within a multi-programming, multi-processing environment. The ideas presented here are an extension and modification of the basic philosophy expressed in CC-196, "A Master Disk Control Routine".

The current disk routine processes information in a serial manner without regard for channel configuration. The new I/O system proposed in this paper will operate all available channels in parallel. The data structure will not limit the number of channels which can be processing information simultaneously for one or many users. In other words, this I/O system provides a necessary interface for a multi-programmed supervisory system.

The I/O system can accomodate any configuration of I/O channels and/or devices. This routine will provide a machine independent and flexible means of data manipulation through a standard interface to all users.

The proposed I/O system allows a supervisory system to dump, edit or retrieve information in an incremental fashion. Proper use of this facility could eliminate bulk dumping, loading and editing of large secondary storage areas.

A restriction imposed by the current disk routine is the inability of many users to access an information file simultaneously. This has produced a necessity for a complex interlocking system, within the current version of CTSS, to allow the use of common files. The proposed I/O system will allow simultaneous usage of information files by many users, interlocked only when an attempt is made to change a file.

Another aspect of the I/O system is the modularization of all machine dependent sections. By replacement of certain modules, different strategies for particular I/O devices, or I/O device characteristics, may be changed without affecting [illegible] I/O structure.

FILE NOTATION AND STRUCTURE

The smallest piece of information which can be manipulated by the I/O system is an element. A file is an ordered sequence of elements. The file is the largest amount of information which can be manipulated by the I/O system.

Every file will have a unique name which is used to identify that file to the user. An element in a file is referenced by specifing the file name and the linear index. For example, the element "i" in file "a" is refered to as a(i). Files may be created, modified or destroyed by a user only through the use of the I/O system.

A file appears to the user to be a block of contiguous storage which may be referenced through normal sequential addressing conventions. However, the physical structure of the file is independent of the logical structure which the user experiences. The user may refer to a file only through the symbolic file name and should have no notion of where or how the file is stored. The number of elements which make up a file is arbitrary, and in fact a file may exist with no elements.

There are four basic operations for manipulating elements within files. These are, opening, closing, reading and writing. To initiate a read and/or write operation, the file must first be opened for reading and/or writing, by the user. To terminate the reading and/or writing of a file, the file must be closed.

A characteristic of all files is the mode. The mode of a file is defined as the inclusive or of the following properties. These properties and their octal values are listed below.

001. TEMPORARY- The file is automatically destroyed as it is being read.

002. SECONDARY- This property defines files which may be deleted by storage collection mechanisms in preference to other files.

004. READ-ONLY- The file can only be read. An attempt to write into a file of this property will cause an error condition.

explain?

010. WRITE-ONLY- The file can only be written. An attempt to read from a file with this property will cause an error condition. ?

020. PRIVATE- The file can only be referenced by the AUTHOR i.e. the user who created or last modified this file.

explain?

040. LINKABLE- The file may be referenced by other users, through the use of the "LINK" facility.

100. PROTECTED- The mode of the file may only be changed by the AUTHOR of the file. Any attempt by another user to change the mode of this file will result in an error condition.

If the mode of a file is zero, the file will have the same properties as a "Permanent" file in the current disk routine.

STRUCTURE OF THE I/O SYSTEM

The I/O system presents a standard machine independent interface to all users. All calls to the I/O system are directed to the basic control module of the system called the File Coordinator. The File Coordinator will then request service from particular Strategy Modules. A Strategy Module is concerned only with a certain class of information storage. The Strategy Module may in turn request service from an I/O Adapter. The I/O Adapter is a module which processes input and output requests for specific I/O devices. All calls to the I/O system requesting input or output must follow this path of control, the File Coordinator- the Strategy Module- the I/O Adapter. See Figure 1 for the basic structure.

THE FILE COORDINATOR

The main function of the File Coordinator is to keep track of all information files of all users. This includes keeping the file modes, the storage devices where files reside and the STATUS of files. The STATUS of a file may take on four values. These values are, (1) inactive, (2) open for reading, (3) open for writing and (4) open for reading and writing. In addition the File Coordinator will determine the validity of all calls to the I/O system. In conjunction with the Strategy Modules the File Coordinator will determine the number of words (elements) available to any user on all secondary storage devices. All requests for service are made by the user directly to the File Coordinator. Upon completion of a request, the user will be notified by the File Coordinator on an interrupt basis. When the user makes a valid request of the File Coordinator which requires service from a secondary storage device, the File Coordinator will call upon the Strategy Module assigned to that device.

THE STRATEGY MODULES

Each Strategy Module will be responsible for a particular storage device. This module will determine the strategy to be used in dealing with this storage device and its associated I/O Adapter. In addition the Strategy Module must be responsible for keeping track of the number of availible units of secondary storage for the device to which it is assigned. Requests are made to the Strategy Modules only through the File Coordinator. The Strategy Modules will interrupt the File Coordinator upon completion of previous requests.

THE I/O ADAPTERS

The I/O Adapter is responsible for the operation of the hardware

0

Interface to a particular device or devices. The I/O adapter will accept requests for service from Strategy Modules only. These requests will be stacked in queues to be executed whenever the associated channel becomes free. The I/O adapters will be responsible for processing all traps associated with the devices to which they are assigned. The I/O adapters will interrupt the appropriate Strategy Modules upon completion of previous requests.

queue role in Strat module

OPERATION OF THE FILE COORDINATOR

The File Coordinator may service requests from a fixed number of active users. Requests from a specific user are in the form a(I), to reference the element "I" in the user's file "a". The File Coordinator however, manipulates information by use of an implicit address of the form c(b(a(I))). This address references the element "I" in the file "a", which is specified by the file "b", which in turn is specified by the file "c". The file "c" in this case is a specific Master File Directory and the file "b" is a specific User File Directory. The user will specify "c" and "b" with one call to the I/O system. Each successive call for a(I) will then be interpreted by the I/O system as c(b(a(I))), until another call is given specifing a new "c" and "b" file pair. By treating the user file directories and the master file directories as normal information files, multiple usage of single files can be accomplished in a general manner. Figure 2 is a diagram of the file structure.

? more than one?

Figure 3 is a diagram of the proposed format for information within the master file directory and the user file directory, for implementation on the current line of IBM equipment and the new GE machine. In word 3, a 36-bit date and time specify the time the file was created or last modified. Word 4 contains an 18-bit date which specifies the date the file was last referred to and an 18-bit "AUTHOR". The AUTHOR of a file is defined as the programmer number of the user who created or last modified the file. In word 5, "MODE" is an 8-bit quantity specifying the properties of the file. "F" is a 3-bit integer which specifies the secondary storage device where the file resides. This integer is used by the File Coordinator to determine the Strategy Module responsible for this device. "NORECS" is a 15-bit integer which specifies the number of physical records contained in the file. The function of the 10-bit quantity "ILOCK" is fully described below.

"RCOUNT" is a 15-bit integer which specifies the number of elements contained in a physical record of the file. The 15-bit integer "LCOUNT" specifies the number of elements contained in the last physical record of the file. The highest element address in a file may be defined as (NORECS-1)*RCOUNT+LCOUNT. All information storage is assigned on an element (machine word) basis. The 3-bit integer "P" is the number of additional information words which are pertinent only to the Strategy Module specified by F, and are ignored by the File Coordinator.

ILOCK is used to allow multiple users to access the same file simultaneously. If a file is in read status, ILOCK also contains a count of the number of users currently reading from that file. If a request is made to modify a file, the high order bit of ILOCK is set to 1. When the number of users reading from the file drops to zero, any user who wishes to modify that file will be allowed to procede. During the time that ILOCK indicates that a modification to a file is on request or in progress, no new users will be allowed to reference that file.

If user "A" wishes to reference a file contained in some other user's file directory (user "B"), he can accomplish this by means of a "LINKED" file. A LINKED file is defined in a user's file directory as a file with a device specification of zero (F=0). When user "A" references a file which is linked to user "B", the MODE of the corresponding file directory entry for user "B" must contain the LINKABLE property.

If a file in a user's file directory is a LINKED file (F=0), MODE, RCOUNT, NORECS and ILOCK are ignored. In this case P will be two and words 7 and 8 of the file entry will contain the problem and the programmer number of the user to which the link is made. A file may be linked in this manner through the file directories of several users. The last entry must be a normal file directory entry which defines the file in a normal manner. Once this linking operation is completed, the file will be treated as a normal file. This operation will be repeated every time a user attempts to <u>open</u> a LINKED file. hopefully not every time he uses a file

The user may refer to his file directory as a file of the name "U.F.D. (FILE)" which is defined in his file directory as a normal file in READ-ONLY mode. The Master File Directory is defined as a User File Directory by the name "M.F.D. FILE" in the Master File Directory. This file is also referred to as "U.F.D. FILE" within the Master File Directory. The I/O system will never allow the Master File Directory to be deleted, regardless of which name is used to reference it.

USER INTERFACE TO THE I/O SYSTEM

The following calls form the interface between the user and I/O system. Some calls are of a control nature and as such may only be available to the supervisory system (marked by *) or a small class of privileged users (marked by +).

The following call is provided to declare a file open for subsequent reading and/or writing.

```
    OPEN.($STATUS$,$A$,$B$,MODE,DEVICE)
```

STATUS specifies which subsequent operation is to be performed on the file A,B. R indicates reading, W indicates writing and RW indicates that the file may be read or written. Elements in a file of temporary mode, which is in read status, are deleted as they are read or skipped over. If a user wishes to ramdomly address elements in a temporary file, the file should be opened for reading and writing or placed in permanent mode. If STATUS is W (indicating the file is to be written only) and the file A,B does not already exist, a new file with the name A,B will be created. Only when a new file is being created are MODE and DEVICE pertinent. MODE is an integer (1-7) specifing the mode of the file to be created, as previously described. If MODE is zero or not specified, a permanent file will be created. DEVICE is an integer which specifies on which storage medium the file A,B is to be written. The following is a list of the available storage devices and their integer equivalents.

1. Core Storage
2. High-speed Drum
3. Low-speed Drum
4. Disk
5. Tape

If DEVICE is zero or not specified, the I/O system will assign a device for the user.

The following call is provided for reading information from a file which has been opened for reading (R or RW).

```
    RDFILE.($A$,$B$,L,E(I)...E(J),EOF,EOFCT)
```

This call will read into the array E(I)...E(J), starting from the relative location L, from the file A,B. If L is zero reading will begin at the word following the last word read from the file. If L is zero on the first call to RDFILE, reading will begin at the first word in the file. If the end of the file is encountered before the specified array has been filled, control will be transferred to the statement with the label EOF and the number of words that will be transmitted will be returned as the value of the integer variable EOFCT.

To transmit the array E(I)...E(J) to the file A,B which has been opened for writing (W or RW), the following call is provided.

```
    WRFILE.($A$,$B$,L,E(I)...E(J))
```

Writing will begin at the relative location L in the file A,B. If L is zero, writing will begin at the location following the last word written into the file. If L is zero on the first call to WRFILE, writing will begin at the location following the last word of the file. The integer L must not exceed the current length of the file.

To check whether a previous read or write operation on the file A,B has been completed, the following call is provided.

```
CHECK.($A$,$B$,FINISH)
```

If the previous operation on the file has been completed, the I/O system will return control to the statement with the label FINISH. In addition the I/O system may interrupt the supervisory system on completion of previous requests.

To assign buffer storage to a file which has been opened for reading and/or writing, the following call is provided.

```
ASSIGN.($A$,$B$,Y(I)...Y(J))
```

This call will cause the core storage area specified by Y(I)...Y(J) to be assigned as utility buffer storage for the file A,B. When the file A,B is closed, this buffer storage will automatically be released from use by the I/O system. This buffer will be used by the I/O system to collect fragments of information which are smaller than the physical record size of the device on which the file resides. The amount of buffer storage required will be determined by the Strategy Module assigned to that device. On the new GE machine, this call will be eliminated and the I/O system will automatically assign all necessary buffer storage.

To terminate the reading and/or writing of a file the user is provided the following call.

```
CLOSE.($A$,$B$)
```

This call will place the file A,B into inactive status. If no file name is specified, all files for the user that are currently in active status will be closed.

To change the name and/or the mode of a file the following call is provided.

```
CHFILE.($A$,$B$,MODE,$C$,$D$)
```

This call will rename the file A,B to C,D changing the mode to the properties specified by the octal integer MODE. If any of the parameters MODE, C, D are zero or left unspecified, the corresponding file information will remain unchanged. If the current mode of the file contains the PROTECTED or PRIVATE property, only the AUTHOR of the file will be permitted the use of this call.

To delete the file A,B, the user is provided the following call.

```
DELETE.($A$,$B$)
```

If the file A.B has the PROTECTED property, the file can not be deleted by this call. by author? see p2

To obtain information concerning the file A.B, the user is provided the following call.

```
FSTAT.(Y(I)...Y(J),$A$,$B$)
```

Upon return from this call the array "Y" will contain the following information, in integer form.

```
  Y(I)= Length of file (no. of words)
Y(I+1)= Mode of file (octal integer)
Y(I+2)= Status of file (1-4)
Y(I+3)= Availability of file (1-3), see below
Y(I+4)= Device on which file resides (1-5)
Y(I+5)= Address of next word to be read from file
Y(I+6)= Address of next word to be written into file
```

Since it is possible for many users to access a single file at the same time, it may be useful to the user to know the availability of a file. The availability of a file may be described as one of the following three conditions.

1. File not currently in use
2. File being read (ILOCK)
3. File being modified (ILOCK)

To declare an area of core storage, a logical tape unit, or other device to be a file the following call is provided.

```
DCLARE.(DEVICE,Q,$A$,$B$,MODE)
```

If DEVICE is 1 (core storage), Q will be the array declaration in the form, Y(I)...Y(J). If DEVICE is 5 (tape), Q will be an integer specifing the logical tape number.

To determine the number of words (elements) allotted and used for a particular storage device, the user is provided with the following call.

```
STORGE.(DEVICE,ALLOT,USED)
```

Upon return from this call ALLOT will contain the number of words the user has been allotted for this DEVICE, and USED will contain the total number of words the user has used on this device. Both quantities will be returned in integer form.

The I/O system references user files through a file directory for a specific problem-programmer number pair. The I/O system is set to operate on this file directory for a specific user with the following call.

```
* ATTACH.($PROBNO$,$PROGNO$)
```

All subsequent calls to the I/O system will refer to files contained in the user file directory specified by $PROBNO$ and $PROGNO$, until another call to ATTACH. is given.

In order to update all pertinent information concerning the user, his files and his file directory on permanent secondary storage, the following call is provided.

```
    UPDATE.
```

This call will cause any new information which, until now, has been kept for the user in core storage to be written out on permanent secondary storage. This procedure will provide the user an important backstop against possible system failure.

To create a link to the file A,B contained in another user's file directory, defined by $PROBNO$,$PROGNO$, the following call is provided.

```
    LINK.($A$,$B$,$PROBNO$,PROGNO$)
```

This call will cause a LINKED file (F=0) of the name A,B to be entered in the user's file directory. The process of chaining through file directories to find the original file A,B will not occur until the user references this file. All references to the file A,B, by this user, (with the exception of the calls RDFILE and WRFILE) will initiate this linking operation. The call,

```
    UNLINK.($A$,$B$)
```

will cause the linked file A,B to be deleted from this user's file directory only. If the file A,B is not a LINKED file, an error condition will result.

The following call is provided to allow the dynamic allotment of storage areas.

```
  * ALLOT.(DEVICE,WORDS)
```

WORDS is an integer specifing the number of words of storage the user will be allowed to use on the storage medium specified by DEVICE.

The I/O system, though capable of keeping information for many users, can only operate for one user at a time. The user to which all subsequent calls to the I/O system will refer, is specified with the following call.

```
  * SETUSR.(USERNO)
```

USERNO specifies one of several active users and is in integer form. The I/O system can be set to operate for another user only through another call to SETUSR.

To set the I/O system to reflect all interrupts to a supervisory system, the following call is provided.

```
 * SETRAP.(FUNCT,)
```

FUNCT. is the name of a internal or external function to be executed by the I/O system in the event of an interrupt or unusual error condition. The I/O system will reflect the interrupt to the supervisory system by means of the statement, EXECUTE FUNCT.(DEVICE,CODE). CODE will be a word or an array of information defining the cause for the interrupt.

The following calls are designed to make the I/O system compatible with CTSS as currently implemented on the IBM 7094.

```
 * USTAT.(Y(I)...Y(J))
 * USAVE.(Z(I)...Z(J))
 * URSTOR.(Z(I)...Z(J))
```

The routine USTAT. assigns an area of protected storage specified by the supervisory system. This storage area will be used by the I/O system to store information pertinent to a specific user's active files. This information is of a critical nature and should not be accessible by the user. However, when a user is dumped or restored by the supervisory system, this information must also be dumped and restored. USAVE. is a routine which packs the information contained in Y(I)...Y(J) which has been specified by USTAT., into the array Z(I)...Z(J). This provides a means of saving the status of all active files for the current user. URSTOR. will reconstruct the active file status table in the array Y(I)...Y(J) from the array Z(I)...Z(J). The calls USAVE. and URSTOR. are necessary for implementation of the SAVE, RESTOR and RESUME commands.

The following call is provided to initialize the I/O system and must also be the first call to the I/O system.

```
 * IOINIT.(CPRINT,ERRLOC)
```

CPRINT is the function name of a general purpose print or type routine. ERRLOC is the label of a statement which will be transferred to when an error occurs for which no error return has been provided.

The following call is provided to isolate the implementation of the current two-core RPQ on the IBM 7094. This call will be eliminated on the new GE machine.

```
 * SETAB.(CALLER,BUFFER,MEMORY)
```

This call is used to specify the memory containing the calling program (CALLER), the memory containing the buffer storage (BUFFER), and the memory to and from which all subsequent I/O will be directed (MEMORY). The integer value "1" specifies memory "A" and the value "2" specifies memory "B".

In all the preceding calls to the I/O system, an additional two parameters may be placed at the end of the calling sequence.

The first of these parameters is taken to be the label of a statement to be transfered to in case of an error. The second parameter is taken to be an integer variable name. The I/O system will store an error code in the corresponding location in the event of an error. A description of these error codes will be available at a later time.

PROPOSED I/O STRATEGIES FOR [illegible]

The following sections describe proposed strategies for the current line of IBM equipment. The point should be stressed that these strategies are independent of the file structure. Strategy Modules may be changed or replaced without modification to the user interface, the File Coordinator or any other part of the I/O system.

1301/1302 DISK AND 7320 DRUM STRATEGY

The file directory entry for a 1301, 1302 or a 7320 file will contain pointers to the first and last "P" tracks. P will be greater than or equal to the number of disk/drum channels which may operated simultaneously. For example, if P is two the seventh and eighth words of the file entry will contain pointers to the first and last two tracks of the file. For a file of this type, RCOUNT will be the number of data words in a single track. NORECS will be the total number of tracks in the file and LCOUNT will be the number of data words in the last track.

Each track in a file of this type will contain chain address pointers to the following and preceding "P" tracks. In addition each track will contain a label in the following form.

```
        PZE TRAKNO,,LCOUNT
```

TRAKNO is a track sequence number. LCOUNT will be non-zero only in the last track of a file and will contain the count of the number of data words in that track. This count must match the value of LCOUNT in the user file directory for that file.

Tracks are assigned in a manner similar to that described in memo CC-196 (Disk Control Routine). All track usage tables will be files contained as entries in the Master File Directory. The file which defines the usage of disk tracks will be referred to as "DISK USAGE". The track usage file for the 7320 drum will be referred to as "DRUM USAGE". Whenever possible, successive tracks of a file will be assigned to separate channels. This procedure will allow all available disk/drum channels to operate on a file in parallel.

1301/1302 DISK AND 7320 DRUM I/O ADAPTER

The disk and 7320 drum Strategy Modules will of course be separate but may share common subroutines including the disk/drum I/O adapter. The disk/drum Strategy Modules will provide calls to the disk/drum I/O adapter specifing only logical track addresses. The I/O adapter will be responsible for determining the actual channels which must be used. The adapter will place all requests into a request queue and return. The trap processor for the disk/drum I/O adapter will empty the request queue on completion of previous requests for that channel. If a request is made requiring a channel not already in operation, a trap will be simulated for that channel. If a request of the user to the disk/drum Strategy Module cannot be immediately accepted by the I/O adapter, the request will be ignored. At this time the disk/drum Strategy Module will effect a return to the supervisory system by means of a trap, simulated through the File Coordinator.

7320A HIGH SPEED DRUM STRATEGY

Storage on the high speed drum will be assigned by sectors (blocks of 2048 words). These sectors will be assigned in the same manner as disk or drum tracks. Sector assignments will be kept in a file which is referred to as "SECTOR USAGE" through the Master File Directory. Each sector is subdivided into two groups of 1024 words each. A user writing a file on the high speed drum need only provide a single buffer of 1024 words. The reason for assigning high speed drum storage by sectors rather than by groups is purely for efficiency.

In a file directory entry for a file stored on the high speed drum, RCOUNT will be 2045 and P will be 1. The seventh word of the file directory entry will contain the sector number of the first and the last sector of the file. Each sector in the file will contain chain address pointers to the following and preceding sectors in the file. In addition, each sector will contain a label as described in the disk/drum Strategy Module.

7320A HIGH SPEED DRUM I/O ADAPTER

The high speed drum Strategy Module will provide calls to its I/O adapter specifing logical sector numbers only. The I/O adapter will convert the logical sector numbers to coresponding logical and physical drum areas with absolute addresses. The high speed drum I/O adapter will operate on request queues in the same fashion as the disk/drum I/O adapter.

TAPE STRATEGY MODULE

Magnetic tapes will be treated as secondary storage in the same manner as disks or drums. Only one file can be recorded on a single tape. However, a single file may consist of more than one tape. The first physical record of a tape file will be a BCD header label of five words. The first two words of the header record are the original file name, the next two words contain the date and time when the file was initiated, and the fifth word will be the tape sequence number. The first tape of a file will always have the sequence number "1". All subsequent records on the tape will be in blocked binary format.

In a file directory entry for a tape file, RCOUNT will be 256 and P will be one. The seventh word of the file directory entry will contain an internal tape address known to the I/O and supervisory systems only. Other information in the file directory entry has the same meaning as described in the disk and drum Strategy Modules.

Each data record will contain 256 information words and in addition will contain a control word in the following form.

```
        PZE RECNO,,LCOUNT
```

RECNO will be the record sequence number. LCOUNT will be non-zero only in the last record of a file and will be the count of the number of words in that record. This word count must match the value of LCOUNT in the file directory entry for that file.

If a file consists of more than one physical tape, the physical tapes will be terminated by an end of file followed by a control record, written in BCD. This record will contain information, such as the next tape sequence number to be used.

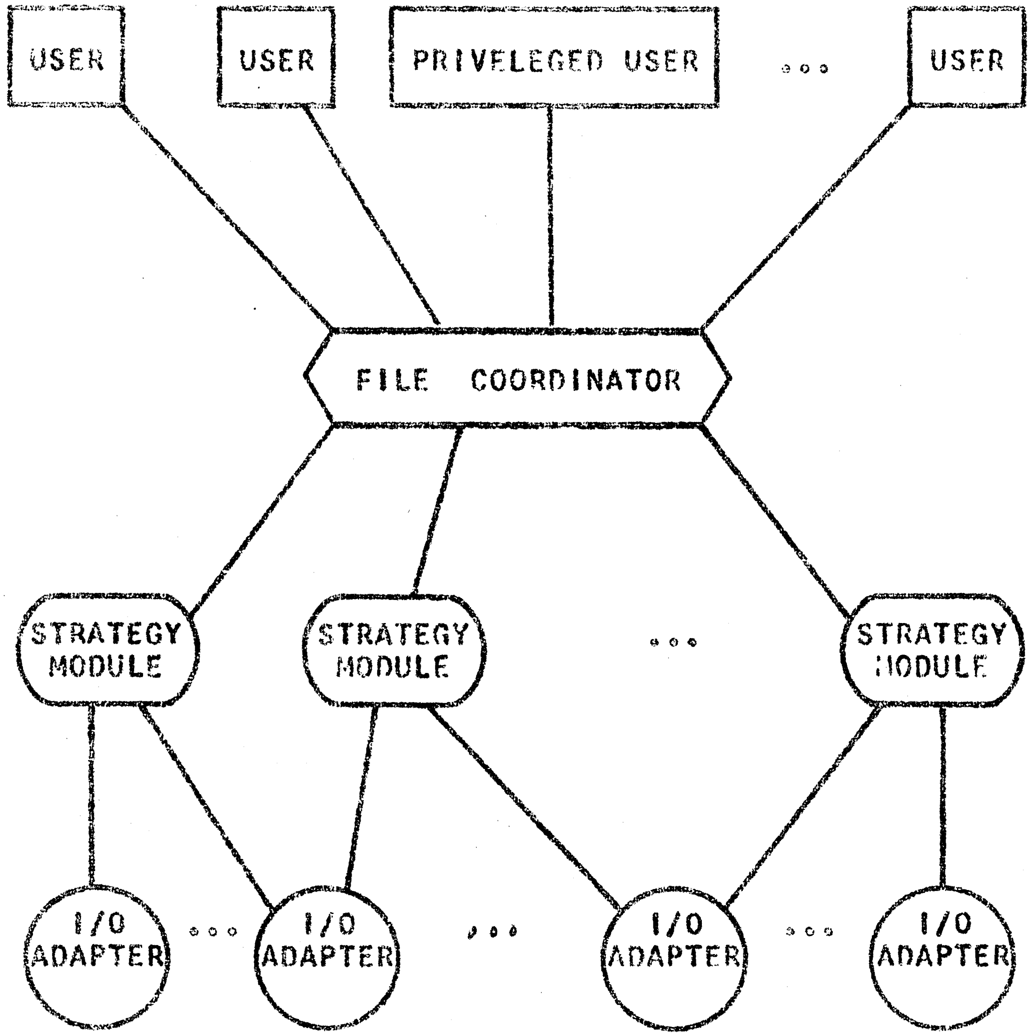
The I/O adapter for the tape Strategy Module will operate on request queues in the same manner as the disk and drum I/O adapters.

INCREMENTAL DUMPING LOADING AND EDITING OF PERMANENT SECONDARY STORAGE

Due to the generality of the I/O system it is possible to incrementally dump areas of permanent secondary storage. Since the date that a file was created or modified is available, it becomes necessary to dump only those files which have been created or modified since the last incremental dump. This procedure would eliminate the dumping of more redundant information than absolutely necessary. Since the user's file directory is a normal information file, the date and time last modified may be used to indicate the last time a file in that file directory was created or modified.

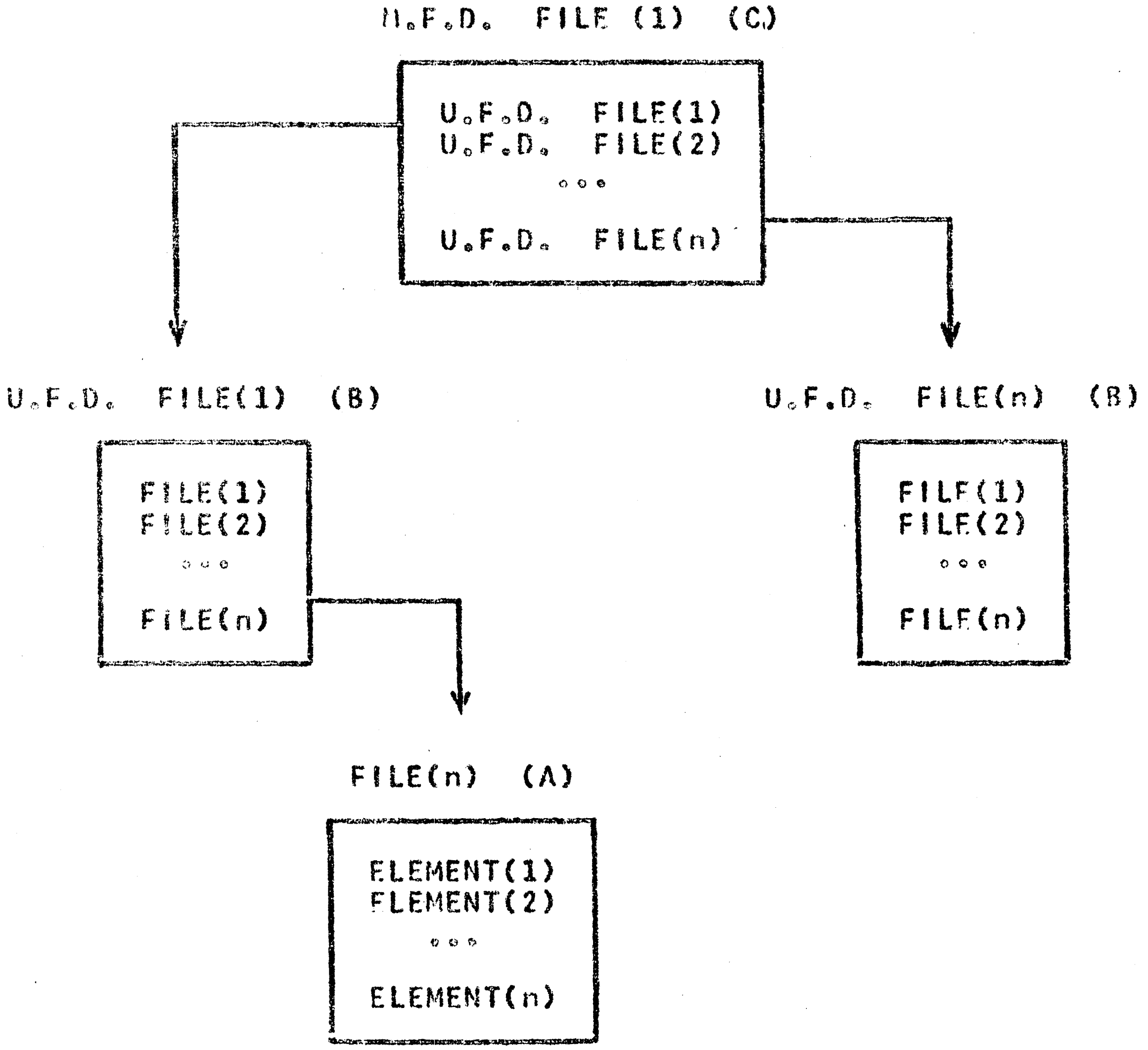
Periodic incremental dumps may be taken for specific users by the LOGOUT command, or may be taken for all users by a time-sharable background job. Naturally, this background job could be run in the absence of CTSS. Information may be retrieved from these dump tapes, again, by either a CTSS command or a background job.

Editing of information to or from secondary storage can be done with a time-sharable background job. Print or punch requests for the users's files could be placed in a file of the name "OFFLIN OUTPUT" for processing by this editor. This file would then be deleted by the editor. The notion of incremental dumps, loads and edits would satisfy the requirements for the continuous operation of CTSS.



I/O SYSTEM STRUCTURE

FIGURE 1



FILE STRUCTURE

FIGURE 2

MASTER FILE DIRECTORY, "M.F.D. FILE"

WORD .... CONTENTS ....

1. USER PROBLEM NUMBER (36 BITS)
2. USER PROGRAMMER NUMBER (36 BITS)
3. DATE AND TIME LAST MODIFIED (36 BITS)
4. DATE LAST USED (18 BITS), AUTHOR (18 BITS)
5. --- (8 BITS), --- (10 BITS), F (3 BITS), NORECS (15 BITS)
6. --- (3 BITS), RCOUNT (15 BITS), P (3 BITS), LCOUNT (15 BITS)
7. The next "P" words contain specific information for a file of type "F".

USER FILE DIRECTORY, "U.F.D. FILE"

WORD .... CONTENTS ....

1. FILE NAME, PART 1 (36 BITS)
2. FILE NAME, PART 2 (36 BITS)
3. DATE AND TIME LAST MODIFIED (36 BITS)
4. DATE LAST USED (18 BITS), AUTHOR (36 BITS)
5. MODE (8 BITS), ILOCK (10 BITS), F (3 BITS), NORECS (15 BITS)
6. --- (3 BITS), RCOUNT (15 BITS), P (3 BITS), LCOUNT (15 BITS)
7. The next "P" words contain specific information for a file of type "F".

18 ? [illegible]

MASTER AND USER FILE DIRECTORY FORMATS

FIGURE 3

BUFFER CONTROL MODULE:

In any call to OPEN, RDFILE, WRFILE, CHECK or CLOSE, the File Coordinator will only be responsible for determining the validity of the call. If the call proves to be valid, the File Coordinator will pass the call to the Buffer Control Module, along with a pointer to the necessary information in the Active File Status Table. In all internal calls in the I/O System, the operation EFA (assembled as NOP) is used to indicate that the address and tag of this word should be combined to form the effective address of the desired parameter. A standard subroutine (GETEFA) will be provided to compute the effective address if required. The operation PAR (assembled as TXH) is used to indicate that parameters may exist in both the address and decrement of this word and that the tag should be ignored. The following calls form the interface between the File Coordinator and the Buffer Control Module.

1. BOPEN- opens a file for subsequent reading and/or writing (called by OPEN).

```
TSX      BOPEN,4          open a file
EFA      PTR,T            .. pointer to file entry
PAR      PRIOR,,ERROR     .. file I/O priority
```

PTR,T defines the effective address which points to the file entry in the Active File Status Table. PRIOR is the location of an integer from 0-7 which defines the I/O priority of this file for as long as it remains an active file. BOPEN will be responsible for initializing that portion of the Active File Status Table for which the Buffer Control Module is responsible.

2. BASIGN- assigns a buffer to a file which has been previously opened for reading and/or writing (called by ASSIGN).

```
TSX      BASIGN,4
EFA      PTR,T
PAR      Y                C(Y)= PZE   BUFADR
```

BUFADR is the first location of the buffer to be used as necessary when reading or writing the specified file. BASIGN will only store the buffer address in the Active File Status Table and return.

3. BREAD- reads from a file which has been previously opened for reading (called by RDFILE).

```
          TSX       BREAD,4
          EFA       PTR,T
          PAR       MEMORY,,BUFFER
          PAR       RELADR,,EOFRTN
          PAR       Y,,QWAIT         C(Y)= PZE    LOC,,NWORDS
          PAR       ERROR
```

MEMORY is the location of an integer which specifies which memory unit is to be read into (1=A, 2=B). BUFFER is the location of an integer specifying in which memory the buffer resides. RELADR is the location of an integer specifying the address within the file at which reading is to begin. If C(RELADR) is zero, reading will begin at the word following the last word read from this file. NWORDS is the number of words to be transmitted from the file beginning with the address LOC. If an attempt is made to read beyond the end of the file, control will be returned to the location EOFRTN. When this occurs, the number of words that will actually be read will be returned in the AC. If an error has occurred during a previous operation on this file, the read request will be ignored and control will be returned to location ERROR. If a previous I/O request involving this file has not been completed or the Strategy Module cannot completely accept the current I/O request, the request will be ignored and control will be returned to location QWAIT.

4. BWRITE- writes into a file which has been previously opened for writing (called by WRFILE).

```
          TSX       BWRITE,4
          EFA       PTR,T
          PAR       MEMORY,,BUFFER
          PAR       RELADR,,EOFRTN
          PAR       Y,,QWAIT         C(Y)= PZE    LOC,,NWORDS
          PAR       ERROR,,NSPACE
```

RELADR is the location of an integer specifying the address within the file at which writing is to begin. If C(RELADR) is zero, writing will begin at the word following the last word writing into the file. NWORDS will be transmitted to the file starting with location LOC. If an attempt is made to write through the end of the file, control will be returned to the location EOFRTN. At this time, the number of words that will actually be written (up to the end of file) will be returned in the AC. If a user wishes to append information to the end of a file, the write operation must begin at the address following the last word in the file. If all available storage on the device being written is exhausted, the call will be ignored and control will be returned to the location NSPACE. After a normal return from BWRITE, the AC will contain the number of records, if any, that have been appended to the file.

5. BTRUNC- truncates a file which has been opened for writing (called by TRFILE).

```
TSX     BTRUNC,4
EFA     PTR,T
PAR     BUFFER,,MEMORY
PAR     RELADR,,EOFRTN
PAR     ERROR,,QWAIT
```

The file will be truncated before the relative address specified by RELADR.

6. BCHECK- checks to see if the previous I/O operation on this file has been completed (called by CHECK).

```
TSX     BCHECK,4
EFA     PTR,T
PAR     MEMORY,,BUFFER
PAR     ERROR,,FINISH
```

If the previous I/O operation has been completed, the Buffer Control Module will finish any related tasks and return to location FINISH. A normal return will indicate that the specified operation is still in progress.

7. BCLOSE- finish any I/O operation on the specified file so that the file may be returned to inactive status (called by CLOSE).

```
TSX     BCLOSE,4
EFA     PTR,T
PAR     MEMORY,,BUFFER
PAR     ERROR,,QWAIT
```

If at this time all related I/O operation have been completed, the Buffer Control Module will complete any of its related tasks. If it is necessary to initiate any new I/O at this time, the I/O should be started and control returned to QWAIT. A normal return from BCLOSE will indicate that all modifications to the file have been completed and the file may be safely removed from active status.

The Buffer Control Module will initiate and control all I/O operations by giving the appropriate calls to the specified Strategy Module. All possible error conditions should be checked before any new I/O is initiated. Whenever possible, reading and writing should be done directly in and out of the user's memory. The buffer should only be used when a partial record is involved. For example, assume the record size is 10 and the user wishes to read 53 words from the beginning of the file. The first 5 records will be read directly into the user's memory. The sixth record will be read into the buffer associated with that file. The remaining 3 words will be copied from the buffer when the user calls CHECK or attempts to initiate new I/O involving this file. If the user now wishes to read the next 53 words from the file, the first 7 words may be copied directly from the buffer. the Buffer Control Module will also provide a label for every record it reads, writes or rewrites. This label will consist of a word which contains the record sequence number in the address portion. This word (or label) will be recorded as the first word of every record in the file. In addition, the label of the last record in the file will contain in the decrement the number of words in this record.

STRATEGY MODULES:

The following calls form the interface between the File Coordinator or the Buffer Control Module and the Strategy Module for the device "F".

1. OPENF- initializes a file for subsequent reading and/or writing (called by BOPEN).

```
TSX      OPENF,4
EFA      PTR,T
PAR      ERROR
```

2. QTEST- checks to see if there is room in the specified queue to process the specified number of requests (called by BREAD, BWRITE, BCLOSE).

```
TSX      QTEST,4
EFA      PTR,T
PAR      REQCT,,FULRTN
```

REQCT is the location of an integer which specifies the number of requests needed. If the queue specified by the file I/O priority cannot accept this number of requests, control will be returned to the location FULRTN.

3. READF- reads from a file starting from a specific record in the file (called by BREAD).

```
TSX      READF,4
EFA      PTR,T
PAR      LABEL,,IOLIST
```

LABEL is the location of a word containing the record number of the first record to be read. This LABEL must match the label of the record to be read. If successive records are to be read with a single call, the record labels must be in ascending order and sequenced by ones. IOLIST is the location of a list of I/O commands in the following form.

```
IOLIST ION      ,,N
       IOP      A,M,N
       IOD
```

ION (PON) is used to skip N words in the record and procede to the next command in the list. IOP (PTW) reads or writes N words starting from the location A in the memory unit specified by M (1= memory A, 2= memory B). After completion, IOP will procede to the next command in the list. IOD (PZE) is used to terminate the list.

4. REWRTF- rewrites successive records in a file starting with the record specified by the address of LABEL (called by BWRITE, BCLOSE).

```
TSX      REWRTF,4
EFA      PTR,T
PAR      LABEL,,IOLIST
```

The record labels will be verified and incremented in the same manner as with READF.

5. WRITEF- appends successive records to a file (called by BWRITE, BCLOSE).

```
TSX      WRITEF,4
EFA      PTR,T
PAR      LABEL,,IOLIST
```

The contents of LABEL will be recorded as the record label of the first record. Successive records labels will be sequenced by ones starting with C(LABEL)+1.

> NOTE: When calling either WRITEF or REWRTF, the decrement of LABEL will specify the word count of the last record to be written. If the decrement of LABEL is zero, the Strategy Module will assume that this record will be followed by another and will provide for any necessary chaining.

6. DFILEF- deletes successive records from a file starting with the record specified by the address of RECNUM (called by DELETE, BTRUNC).

```
TSX      DFILEF,4
EFA      PTR,T
PAR      RECNUM,,QWAIT
```

If the Strategy Module cannot accept this call at the present time, control will be returned to location QWAIT. Delete requests will automatically be placed in the lowest priority queue. Once a delete request has been accepted by the Strategy Module, the corresponding entry in the Active File Status Table is no longer required.

The Strategy Module will maintain priority queues for all I/O requests and will supply the necessary calls to the appropriate I/O Adapter to execute the requests. The interface between the Strategy Modules and their I/O Adapters is defined by the nature of the I/O device and cannot be specified by a single set of calls.

ACTIVE FILE STATUS TABLE:

All information concerning the status of an active file is passed through an entry in the Active File Status Table. The format of this entry is described below. The numbers in "()" indicate the number of binary bits assigned to the function. The character "-" is used to indicate an unused bit position.

1. PROBNO(36) (set by File Coordinator)
2. PROGNO(36) (set by File Coordinator)
3. FNAME1(36) (set by File Coordinator)
4. FNAME2(36) (set by File Coordinator)
5. DAYTIM(36) (set by File Coordinator)
6. DATELU(18), AUTHOR(18) (set by File Coordinator)
7. MODE(8), ILOCK(10), F(3), RCOUNT(15)
8. ~~-, W(1), R(1)~~, NORECS(15), P(3), LCOUNT(15)
9. POINTR(36)
10. ---, REDREC(15), ---, REDWRD(15)
11. ---, WRTREC(15), ---, WRTWRD(15)
12. -, CHNG(1), PRIME(1), BUFREC(15), ---, BUFADR(15)
13. ---, WINDEX(15), ---, DINDEX(15)
14. -, DR(1), DW(1), DCOUNT(15), ---, DADDRS(15)
15. PRIOR(3), IOTASK, EFLAG(3), CURREC(15)
16. Reserved for Strategy Module use
17. Reserved for Strategy Module use

The following list describes the functions of the variables listed above. The initials in parenthesis indicate which modules may modify that variable. FC is the File Coordinator, BCM is the Buffer Control Module and SM is the Strategy Module. The module whose initials are listed first will be the module that is responsible for initializing the variable.

PROBNO- (FC), user problem number in bcd

PROGNO- (FC), user programmer number in bcd

FNAME1- (FC), file name, part 1

FNAME2- (FC), file name, part 2

DAYTIM- (fc), date and time file was created or last modified

DATELU- (FC), date file was last used

AUTHOR- (FC), programmer no. of user who last modified this file (in binary)

MODE- (FC), mode of file

ILOCK- (FC), used in file interlock mechanism (see CC-241)

F- (FC), device (1-3) on which file resides

RCOUNT- (FC), number of words per record for this device

W- (FC), non-zero if file open for writing

R- (FC), non-zero if file open for reading

NORECS- (FC,BCM), number of records in this file

P- (FC), reserved for compatibility with future systems

LCOUNT- (FC,BCM), number of words in last record of file

POINTR- (FC,SM), pointer to beginning of file (interpreted by SM)

REDREC- (BCM), record no. of record containing the next word to be read from the file

REDWRD- (BCM), address of word within REDREC to be read next

WRTREC- (BCM), record no. of record containing the next address to be written

WRTWRD- (BCM), address within WRTREC to be written next

CHNG- (BCM), non-zero if contents of buffer differ from that of corresponding record

PRIME- (BCM), non-zero if contents of buffer represent a complete file record

BUFREC- (BCM), record no. of record contained in the buffer

BUFADR- (BCM), address of file buffer

WINDEX- (BCM), no. of words written into the buffer (CHNG=1 and PRIME=1)

DINDEX- (BCM), index within buffer of words to be transmitted to or from user's memory before initiating new I/O for this file

DR- (BCM), non-zero when necessary to move words from buffer to user's memory before initiating additional I/O for this file

DW- (BCM), non-zero when necessary to move words from user's memory to buffer in order to complete the previous I/O request for this file

DCOUNT- (BCM), no. of words to move if DR or DW is non-zero

DADDRS- (BCM), address in user's memory to start moving to or from when DR or DW is non-zero

PRIOR- (BCM), file I/O priority (0-7)

IOTASK- (SM), no. of sub-tasks to be completed to complete previous I/O request for this file

EFLAG- (SM), non-zero if error during previous request (PERROR=1, FERROR=2)

CURREC- (SM), record currently in process

PROGRAMMING STAFF NOTE 42

FROM: R.C. Daley

SUBJECT: User Interface to the File I/O System

DATE: JAN. 22, 1965

The following calls form the interface between the user and the I/O system. Some calls are of a control nature and as such may only be available to the supervisory system (marked by *) or a small number of privileged users (marked by +). The parameter "-0" may be used in any calling sequence to specify a missing parameter.

UPDMFD- is used to place a new user in the MFD.

```
+    UPDMFD.($PROBNO$,$PROGNO$)
```

Error codes:

03. User already in M.F.D.
04. Machine or System error

DELMFD- is used to remove a user from the MFD.

```
+    DELMFD.($PROBNO$,$PROGNO$)
```

Error codes:

03. User not found in M.F.D.

ATTACH- is used to attach a user to the U.F.D. specified by 'PROBNO' and 'PROGNO'.

```
+    ATTACH.($PROBNO$,$PROGNO$)
```

Error codes:

03. User not found in M.F.D.
04. Machine or System error

UPDATE- is used to update all pertinent information concerning the user currently attached.

```
UPDATE.
```

Error codes:

03. Machine or System error

SETPRI- is used to assign priorities to certian I/O tasks which would otherwise be processed in the order in which they were received.

```
SETPRI.(PRIOR)
```

PRIOR is an integer from 1-7. The higher the value of PRIOR, the lower the priority. When files are opened for reading and/or writing, they will be assigned the priority set by the last call to SETPRI. If there was no previous call to SETPRI, all files will be treated with equal priority.

Error codes:

Only the standard error codes, see below

OPEN- is used to declare a file open for subsequent reading and/or writing.

```
OPEN.($STATUS$,$NAME1$,$NAME2$,MODE,DEVICE)
```

Error codes:

03. File is already in active status
04. Too many active files
05. $STATUS$ is illegal
06. 'LINKED' file not found
07. File to which link is made is not 'LINKABLE'
08. File in 'PRIVATE' mode
09. Attempt to write a 'READ-ONLY' file
10. Attempt to read a 'WRITE-ONLY' file
11. Machine or System error
12. File not found in U.F.D.
13. Illegal device specified
14. No space allotted for this device
15. Space exhausted for this device
16. File currently being restored from tape
17. Input/Output error, see codes below

BUFFER- is used to assign a buffer for use in reading or writing an active file (must be used after call to OPEN). This call will be eliminated on the GE 635.

```
BUFFER.($NAME1$,$NAME2$,BUFF(450)...450)
```

Error codes:

03. File is not an active file
04. previous I/o out of bounds (membnd changed)
05. Buffer too small
06. Input/Output error, see codes below

RDFILE- is used to read from a file which has been opened for reading.

```
RDFILE.($NAME1$,$NAME2$,RELLOC,A(N)...N,EOF,EOFCT)
```

Error codes:

03. File is not an active file
04. File is not in read status
05. No buffer assigned to this file
06. Previous I/O out of bounds (MEMBND changed)
07. Input/Output error, see codes below

WRFILE- is used to write into a file which has been opened for writing.

```
WRFILE.($NAME1$,$NAME2$,RELLOC,A(N)...N,EOF,EOFCT)
```

Error codes:

03. File is not an active file
04. File is not in write status
05. No buffer assigned to this file
06. Allotted space exhausted for this device
07. Previous I/O out of bounds (MEMBND changed)
08. Input/Output error, see codes below

TRFILE- is used to truncate a file which has been previously opened for writing.

```
TRFILE.($NAME1$,$NAME2$,RELLOC)
```

The file will be truncated immediatly before the relative address RELLOC.

Error codes:

03. File is not an active file
04. File is not in write status
05. No buffer assigned to this file
06. Previous I/O out of bounds (MEMBND changed)
07. RELLOC larger than file length
08. Input/Output error, see codes below

FCHECK- is used to check if a previous read or write on a file has been completed.

```
FCHECK.($NAME1$,$NAME2$,FINISH)
```

Error codes:

03. File is not an active file
04. Previous I/O out of bounds (MEMBND changed)
05. Input/Output error, see codes below

CLOSE- is used to close an active file and return it to inactive status.

```
CLOSE.($NAME1$,$NAME2$)
```

If NAME1 is ALL and NAME2 is not specified, all active files will be closed.

Error codes:

03. File is not an active file
04. Previous I/O out of bounds (MEMBND changed)
05. Input/Output error, see codes below
06. Machine or System error

RESETF- is used to remove all active files from active status when the user's core image is no longer available. This call will normally only be used by the supervisory system (CTSS).

```
RESETF.
```

Error codes:

03. Machine or System error

CHFILE- is used to change the name and/or mode of a file.

```
CHFILE.($OLDNM1$,$OLDNM2$,NEWMOD,$NEWNM1$,$NEWNM2$)
```

Error codes:

03. Attempt to change M.F.D. or U.F.D. file
04. File not found in U.F.D.
05. 'LINKED' file not found
06. File to which link is made is not 'LINKABLE'
07. Attempt to change 'PRIVATE' file
08. Attempt to change 'PROTECTED' file of another user
09. Temporary file would overflow space allotted for device
10. File already exists with name 'NEWNM1 NEWNM2'
11. Machine or System error
12. file in active status

DELFIL- is used to delete a file.

```
DELFIL.($NAME1$,$NAME2$)
```

Error codes:

03. File not found in U.F.D.
04. 'LINKED' file not found
05. file to which link is made is not 'LINKABLE'
06. File is 'PROTECTED'
07. Machine or System error
08. file in active status

FSTATE- is used to determine the present status of an active or inactive file.

```
FSTATE.($NAME1$,$NAME2$,A(8)...8)
```

Upon return from this call the array "A" will contain the following information.

A(8)= length of file
A(7)= MODE of file
A(6)= STATUS of file (1-4)
A(5)= DEVICE on which file resides (1-3)
A(4)= Address of next word to be read from file
A(3)= Address of next word to be written into file
A(2)= Date and time file was created or last modified
A(1)= Date file was last referred to and 'AUTHOR' of file

Error codes:

03. File not found in U.F.D.
04. 'LINKED' file not found

05. File to which link is made is not 'LINKABLE'

MOVFIL- is used to move a file from the current user's file directory to the file directory specified by 'PROBNO PROGNO'.

```
+    MOVFIL.($NAME1$,$NAME2$,$PROBNO$,$PROGNO$)
```

Upon return from this call, the file will no longer exist in the current user's file directory.

Error codes:

03. File not found in current U.F.D.
04. File is a 'LINKED' file
05. File is 'PROTECTED'
06. File already exists in 'PROBNO PROGNO'
07. Machine or System error

SETFIL- is used by the file load and retrieval systems to create an entry in a file directory with a specific date and time.

```
+    SETFIL.($NAME1$,$NAME2$,DAYTIM,DATELU,MODE,DEVICE)
```

DAYTIM is the date and time to be used as the date and time last modified. DATELU contains the date last used and the 'AUTHOR' of the file.

Error codes:

03. Illegal device number
04. Machine or System error

LINK- is used to create a link to a file contained in another user's file directory.

```
+    LINK.($NAME1$,$NAME2$,$PROBNO$,$PROGNO$)
```

Error codes:

03. Machine or System error
04. 'PROBNO PROGNO' not found in M.F.D.

UNLINK- is used to delete the association set up by LINK.

```
     UNLINK.($NAME1$,$NAME2$)
```

Error codes:

03. File not found in U.F.D.
04. File is not a 'LINKED' file
05. Machine or System error

ALLOT- is used to set the number of records allotted for and used on a particular DEVICE.

```
+    ALLOT.(DEVICE,ALLOT,USED)
```

Normally USED is not specified. The parameter USED should only be used to correct an error in the number of records used.

Error codes:

03. Illegal device specified

STORGE- is used to determine the number of records allotted and used for a particular DEVICE.

```
     STORGE.(DEVICE,ALLOT,USED)
```

Error codes:

03. Illegal DEVICE specified
04. Machine or System error

The following calls concern TAPE files only.

MOUNT- is used to direct the file system to mount a set of reels on the unit specified by the logical tape drive UNITNO.

```
     MOUNT.(CHANNO,UNITNO,MESSAG(20)...20)
```

CHANNO specifies the number of the channel to be used. If CHANNO is zero or not specified, the file system will select a channel for the user. This call must be used prior to reading or writing a tape file. The array MESSAG is a BCD comment that will be sent to the console operator with the mounting directions.

Error codes:

03. No tape available on specified channel

UMOUNT- is used to unmount a set of reels and free the corresponding tape drive for other use.

```
UMOUNT.(UNITNO,MESSAG(20)...20)
```

Error codes:

03. Tape file currently in use

VERIFY- is used to verify the label of a tape file after it has been mounted but before it may be opened for reading or writing.

```
VERIFY.(UNITNO,LABEL(4)...4)
```

Error codes:

03. Label is incorrect, try again up to five times
04. Label is unreadable
05. Tape file does not exist
06. Tape file cannot be mounted at this time (operations)

LABEL- is used to write a label on a new tape file before it may be opened for writing.

```
LABEL.(UNITNO,LABEL(4)...4)
```

Error codes:

03. Tape will not write
04. Tape file does not exist
05. Tape file cannot be mounted at this time (operations)

TAPFIL- is used to inform the file system that a file exists or is to be created on the set of reels specified by UNITNO.

```
TAPFIL.($NAME1$,$NAME2$,UNITNO,FILENO)
```

FILENO is used to specify which file on the set of reels specified by UNITNO. If a user wishes to add a file to the end of a set of reels, he may specify a FILENO of zero. When this file is opened for writing, the tape strategy module will assign the file number automatically. This procedure may be used to add a file to the end of a set of reels when the number of files is unknown.

Error codes:

03. Machine or System error

ERROR PROCEDURE:

In all calls to the file system, an additional 2 parameters may be added to the end of the calling sequence. The first of these parameters is taken to be tha label of a statement to be transferred to in case of an error. The second is taken to an integer variable in which the file system will store the error code. In addition, the following call is provided to obtain more specific information about an error condition.

IODIAG.(A(7)...7)

Upon return from this call, the array "A" will contain the following information.

A(7)= Location of call causing the error
A(6)= BCD name of entry resulting in error
A(5)= Error code
A(4)= Input/Output error code (1-7)
A(3)= NAME1 of file involved in error
A(2)= NAME2 of file involved in error
A(1)= Location in file system where error was found

STANDARD ERROR CODES:

001. Illegal calling sequence or Protection violation
002. Unauthorized use of priveleged call
100. Error reading or writing U.F.D or M.F.D.
101. U.F.D. or M.F.D. not found, Machine error

INPUT/OUTPUT ERROR CODES:

1. Parity error reading or writing file
2. Fatal error reading or writing file, cannot continue
3. Available space exhausted on this device
4. Tape file not mounted or not available

SUPERVISOR ENTRIES TO FILE SYSTEM:

SETUSR- is used to set the I/O system to operate for one of several active users (DUSER1= CTSS, DUSER2= current CTSS user).

*   SETUSR.(DUSER,RCODE,AUTHNO,LIMITS,RELLOC,PRIOR)

RCODE is the user restriction code and is described later. PRIOR is an integer from 1-7 which specifies the user's I/O priority. The higher the value of PRIOR, the lower the priority. AUTHNO is the programmer no. (in BCD) of the user who is about to use the file system. AUTHNO is used to determine the authorship of files in 'PRIVATE' or 'PROTECTED' mode. LIMITS is the user protection bounds and RELLOC is the user relocation. All of the above parameters to SETUSR are optional.

Error codes:

03. Illegal user number

SETRAP- is used to set the supervisory interrupt procedure.

*   SETRAP.(IFUNCT.)

The I/O system will reflect interrupts to the supervisory system by means of the following call.

```
EXECUTE IFUNCT.(USERNO,ICODE,IR4,ILC,INFO(N)...N)
```

ICODE is the interrupt code. The following interrupt codes have been asigned.

1. User attempting to initiate I/O
2. I/O task initiated
3. I/O task completed
4. File interlocked
5. File no longer interlocked
6. User I/O queue full or waiting on I/O

USTAT- is used to assign an area of protected storage to be used by the I/O system in servicing the current user.

*   USTAT.(Y(N)...N,Q1...Q1L,Q2...Q2L, ..., QN...QNL)

The array 'Y' will be used by the file system to store all information pertaining to a particular user of the file system. Qi specifies storage for queueing all i/o requests for the device 'i'. If QiL is zero, all attempts to use the device 'i' will be rejected.

USAVE- is used to save the status of all active files for the current user.

*   USAVE.(COUNT,Z(M)...M)

Upon return from this call, the contents of COUNT will contain the number of words saved in the array 'Z'.

Error codes:

03. The array 'Z' is too small

IOINIT- is used to initialize the I/O system.

*   IOINIT.(ERRLOC,DATLOC,TIMLOC,ENBLOC)

IOSTOP- is used to terminate all I/O for the user specified by 'USERNO'.

*   IOSTOP.(USERNO)

If USERNO is zero, all I/O currently in process will be terminated.

IOSTRT- is used to restart I/O processing after a call to IOSTOP.

*   IOSTRT.(USERNO)

SETAB- is used to set the I/O system to operate on the correct memory units (1=A, 2=B).

*   SETAB.(CALLER,BUFFER,MEMORY)

This call is used to specify the memory containing the calling program (CALLER), the memory containing the buffer storage (BUFFER), and the memory to which all subsequent I/O will be directed. If MEMORY, BUFFER or CALLER are negative, all references to the specified memory (1 or 2) will be checked for protection mode violations.

I/O DEVICES:

1. LOW-SPEED DRUM
2. DISK
3. TAPE

RESTRICTION CODES:

The LOGIN command will set the low-order 6 octal digits of the user restriction code.

00000001	User may use common files.
00000002	User may use restricted calls to the I/O system.
00000004	User may modify "PROTECTED" file of other users.
00000010	User may refer to "PRIVATE" files of other users.
00000020	User may modify the supervisory and I/O systems.
01000000	User is Background system.
02000000	User is Foreground.
04000000	User is FIB.
10000000	User is incremental dumper
20000000	User is priveleged command