MASSACHUSETTS INSTITUTE OF TECHNOLOGY
ARTIFICIAL INTELLIGENCE LABORATORY

Artificial Intelligence
Memo No. 161A
(Revised 161, June 1968)

MAC-M-377
July 1969

ITS 1.5 REFERENCE MANUAL

D. Eastlake*, R. Greenblatt, J. Holloway,
T. Knight, S. Nelson

* Author of this memo

This reference manual consists of two parts. The first (sections 1 through 6) is intended for those who are either interested in the ITS 1.5 time sharing monitor for its own sake or who wish to write machine language programs to run under it. Some knowledge of PDP-6 (or PDP-10) machine language is useful in reading this part. The second part (sections 7, 8, and 9) describes three programs that run under ITS. The first program (DDT) is a modified machine language debugging program that also replaces the "monitor command" level (where the user is typing directly at the monitor) present in most time-sharing systems. The remaining two (PEEK and LOCK) are a status display and a miscellaneous utility program. It should be remembered that the McCulloch Laboratory PDP-6 and PDP-10 installation is undergoing continuous software and hardware development which may rapidly outdate this manual.

* * *

Work reported herein was supported by the Warren McCulloch Laboratory, an M.I.T. research program sponsored by the Advanced Research Projects Agency of the Department of Defense under Office of Naval Research contract number N00014-70-A-0362-0002. Reproduction of this document, in whole or in part, is permitted for any purpose of the United States Government.

0. Introduction

The reader will find below in section 0.1 a few notes on tne notation used in this manual. This is followed in section 0.2 by fairly extensive introductory and overview material on the Incompatable Time Sharing system monitor (ITS), tne body of the information about which appears in sections 1 through 6. Then in section 0.3 will be found a brief preview of the information appearing in sections 7, 8, and 9 about the three programs DDT, PEEK, and LOCK respectively. The first of these is a modified machine language debugging program that replaces the "monitor comand" level (where the user is typing directly at the monitor) present in most time sharing systems. The remaining two are of a status display and miscellaneous utility nature. These three sections are each fairly self-contained as opposed to any particular section of the six on ITS which may be highly interdependent with other ITS sections. Finally, in section 0.4, appears a summary of the appendices to this manual.

It should be remembered that the Project MAC, AI Group, PDP-6 and PDP-10 installation is undergoing continuous software and hardware development that may rapidly outdate this manual. Please direct all corrections, additions or comments to the author at:

Donald E. Eastlake III,
Room 915,
545 Technology Square,
Cambridge, Mass. 02139

0.1 Notation Used in this Manual

0.1.1 Numbers

All numbers written in arabic numerals are octal except as follows: (1) those part of a name, such as a type "35" teletype, (2) those part of the meta-structure of this manual (a page or section number for example), (3) or those that are floating point, which will always have more than one digit to the right of the decimal point such as 3.14159. For the meaning of numbers which appear to be floating point with one digit to the right of the decimal point and which are not section numbers, see section 0.1.3 below. Numbers that are written alphabetically are decimal.

0.1.2 Character Objects and Strings

Character objects and strings are usually surrounded by quotation marks (""") and are relatively clear from context. Non-graphic characters typed by holding down the control key of a teletype and striking a graphic are represented by the graphic

preceeded by a "^" as in ^A for "control-A". The special character alt.-mode (or escape or prefix) is represented by "$".

0.1.3 Bit Positions and Intervals

Bit positions in thirty-six bit words (word length on the PDP-6 and PDP-10) are represented by the concatenation of a digit from 1 through 4, a ".", and a digit from 1 through 9. The first digit is a quadrant number starting from the right or least arithmetically significant part of the word. The second digit is a bit number in the quadrant, also starting from the right or least significant bit. Thus 1.1 is the lowest bit and 4.9 the sign bit.

Bit intervals are represented by the concatenation of an included starting bit position, a "-", and an included terminating position. Thus 3.3-3.1 is the lowest octal digit in the left half of a word.

0.1.4 Internal References

Most internal references in the text of this manual are of the following form: [App x], [Ref x], and [Sec x]. These refer, respectively, to the x'th appendix, the x'th item in the section of this manual entitled "References", and the x'th numbered section of this manual. Some internal references are written out in less

formality and combined forms are allowed, such as the following, whose meaning should be evident: [Sec x, y], [App x; Ref x, y], etc.

0.1.5 Other Conventions

UUO's ("UnUsed" Operations [Ref 1]) that trap to the system [Sec 1.2.1] to request action by it are frequently refered to as "system calls". (This should not be confused with the .CALL subclass of system calls.) Individually they are written in all capital letters with a preceding period. The reason for this is that the MIDAS [Ref 3] assembler has just these character strings (actually the first six characters including the period) pre-defined as symbols. In pronouncing them, the period should be ignored.

The use of the system calls described in this manual is frequently illustrated by a short excerpt of pseudo-assembly-language. In such examples, the tag CALL will appear on the line of the system call in question. Frequently set-up instructions precede the call to emphasize certain of its properties. They should not be taken too literally.

0.2 Introduction to ITS

The Incompatible Time Sharing system (ITS) is a control program tailored to the Project MAC Artificial Intelligence (AI) Group PDP-6

and PDP-10 installation. It was designed in an attempt to provide the following potential advantages of a time sharing system:

(1) More than one person can perform tasks not requiring all available computer resources with seeming simultaneity; one person can similarly perform several tasks.

(2) Simple device independent input-output facilities should relieve the user of having to worry about hardware interface routines.

(3) Jobs which would otherwise require excessive continuous blocks of time can be performed automatically in small parts over long periods of time at reduced priority.

(4) Various debugging and other facilities can be designed-in using the mechanisms necessary to protect user programs from each other.

To achieve these advantages and the particular goals of Project MAC's AI Group, which require a high level of sophisticated service to a limited number of users, ITS is designed with the following properties:

(1) ITS sits in control of the PDP-6 performing most input-output for and allocating machine resources among various user programs.

(2) All user programs reside in core storage so it is possible to switch between programs with great rapidity and have sufficiently short quanta to allow character response without undue inefficiency. (Swapping of programs may soon be added but it is expected that only dormant programs will be swapped out.) A user may have many programs running for him "simultaneously".

(3) ITS has a minimal direct user command facility to assure users that they can maintain control over their programs. For the utmost in flexibility almost all system actions are the result of systems calls executed by the user's programs.

(4) ITS attempts to provide not only simple standardized input-output and other system calls but also much more complex and specialized calls which reduce overhead or enable the use of special devices and features.

MIDAS, a machine language assembler with macro facilities [Ref 3, 10], was chosen for writing ITS because of the resultant efficiency, flexibility, ease in writing, and ease in debugging. For information more detailed or more recent than this manual consult the sources listed in Appendix F.

0.2.1 Preview of Section 1

The user of ITS normally commands a hierarchy of programs in machine storage organized in an inverted tree. The top procedure [Sec 7] is initially loaded for him by the system when he informs it of his presence by typing a ^Z on an idle console [Sec 1.1.1]. All programs may have inferior procedures which they control. Almost all commands to the system are machine instructions [App A, B; Ref 1], which trap to the monitor, executed for the user by his programs [Sec 1.2]. The only exception is that he may interrupt the superior procedure of the program with which he is conversing by typing ^Z [Sec 1.2.1]. By this method the user can return to his top procedure at which point ^Z's typed on his console are ignored by the system.

0.2.2 Preview of Sections 2 and 3

The majority of the code in the ITS monitor is devoted to input-output. It has been organized with such goals in mind as flexibility, speed, and generality. Simple device-independent system calls applicable to all devices where they are meaningful, more complex but efficient calls applicable to most devices, and specialized system calls enabling nearly full use of some special devices are all available. Most devices are referenced symbolically [App C] and procedures may cause symbolic translations such that an inferior procedure referencing a particular device will in fact get a different device. Input-output may be done a character or word per system call or, for less overhead per character or word, an arbitrary size block of words may be transfered into or out of the user program's core with one system call. The user program is not required to have any buffers in its core image. Many special features relate to consoles or special devices on the PDP-6 such as eyes, arms and the DEC 340 display.

0.2.3 Preview of Section 4

A feature of the ITS monitor not frequently found in time sharing systems and contributing greatly to its flexibility and generality is that user programs may receive software implemented

interrupts in much the same way that ITS receives hardware interrupts. The system is so designed that the user is rarely involuntarily uninterruptable for more than a few nundred microseconds. If the interrupt is dismissed in a normal manner the non-interrupt portion of the program can proceed even from the "middle" of a system call taking an arbitrary amount of time.

The ITS core allocator keeps a record of the status of each 2000 word block of memory and the 200 word sub-blocks into which some of these are divided. It allocates and de-allocates memory for and at the request of procedures and system input-output routines. Most of its complexity is due to the lack of paging on the PDP-6 which makes it sometimes necessary to "shuffle" memory to provide sufficient contiguous space for a procedure.

A scheduling algorithm is used which tries primarily to equalize machine time used by each active procedure tree (console) and secondarily to equalize machine time among those procedures in each tree. The system variables related to a particular procedure [App D] used by the scheduler and other parts of ITS are kept in system memory and do not impinge on the user's core image.

0.2.4 Preview of Section 5

User programs operating within the environment provided by ITS are organized (as mentioned in section 0.2.2) as inverted tree

hierarchies. Superior procedures can retrieve and store words in their inferior procedures as if they were input-output devices. There also exists a large set of special system calls for controlling procedures immediately beneath them and several special ways that procedures may communicate with their superior procedure. Buffered communication between arbitrary pairs of procedures treating each other as input-output devices is also provided [Sec 3.4.3].

0.2.5 Preview of Section 6

A large number of miscellaneous system calls and features exist for a variety of purposes. Some provide easy input-output to devices which do not fit the forms of standard ITS devices. Others provide the use of certain hardware modifications to the PDP-6 which enable the simulation of certain features of manual control of the PDP-6 such as address stop [Ref 1]. The remainder are such minor but necessary functions as login and logout.

A special procedure, known as the system job, exits under ITS. It performs various low priority functions for the system and can check constant portions of ITS against a copy in an attempt to detect some forms of system or hardware failure.

0.3 Introduction to DDT, PEEK, and LOCK

These three programs all run under ITS but are more closely related to it than most other systems programs. The editor (TECO), assembler (MIDAS [Ref 3]), list processing language (LISP), etc. used by the Project MAC AI Group have also been modified to run under ITS but have very similar non-time-sharing versions. PEEK and LOCK would generally be meaningless outside ITS and while there is a non-time-sharing DDT, the version for ITS has been very extensively modified.

0.3.1 Preview of DDT

DDT originated as a general machine language debugging program on the Digital Equipment Corporation PDP-1 computer. When the hierarchal organization of procedures [Sec 5] and logical simplicity of procedure origination of almost all system actions [Sec 1.2] had been decided upon, the need for a conversational "monitor command" program was recognized. An extensively modified DDT, with commands relating to various ITS features and modifications so as to be able to control multiple procedures was the obvious answer. For historic reasons a top level modified DDT under ITS is known as a "HACTRN".

0.3.2 Preview of PEEK and LOCK

The first of these utility programs, PEEK, is described in section 8 of this manual. It provides periodically updated displays or printouts of various aspects of the time sharing system's status.

The second and smaller program, LOCK, is described in section 9 of this manual. It performs a variety of miscellaneous functions, some related to testing or debugging ITS.

DDT, described in section 7, is the normal means by which the user may load and transfer control to these programs. Note that the commands P, Q, and ? have the same meaning for both programs. The ? command causes each of them to print a list of their commands with short explanations.

0.4 Preview of Appendices

The first three appendices to this manual are for cross-reference and index usage (although important cross-references are frequently included in the text [Sec 0.1.4]). Appendices A and B list system calls [Sec 0.1.5, 1.2.1] in numeric and alphabetic order, respectively, with the number of the manual section most relevant to the call. Appendix C lists symbolicly referenceable input-output devices in alphabetic order with the section or sections most relevant to them.

Appendix D is nearly a direct extract from the source listing of ITS. It is the initial "dummy" image of the user variable area, one of which exists for each procedure [Sec 4, 5] in a system.

Appendix E, included for completeness, lists certain system calls that are being phased out along with tneir replacements and short descriptions of their functions.

Appendix F lists persons to consult for further information on ITS.

The last appendix, denominated G, gives a single console session with ITS as an illustration in which the self documenting features of DDT, PEEK, and LOCK are exercised.

1. Transmitting Commands to ITS

See also section 0.2.1.

1.1 Control Z

The only item of input-output data recognized by ITS directly as a command is the character ^Z when received from an idle teletype or from an active console, that is a teletype in control of a procedure tree [Sec 5]. A teletype is a member of the class of devices that may be consoles, currently either a GE Datanet 760 terminal or a KSR 35/37. Teletypes may also be in use as ordinary devices (as Thm instead of TTY [Sec 3.2.1]) in which case ITS ignores ^Z's typed on them.

1.1.1 Control Z on an Idle Teletype

If the teletype is idle, ^Z normally causes the teletype to become a console and loads and starts as its top level procedure [Sec 5.1, 7] the dump file named "@ HACTRN" on device SYS, or if not found there from device UT2 (see section 2.1 for a discussion of file and device names). The JNAME of a thus initiated procedure is HACTRN and its UNAME and SNAME are set to the value minus one [Sec 2.2.3, 3.1; App D]. This will not happen if there is insufficient memory

available or if too many people type ^Z simultaneously. ITS will echo a ^Z typed on an idle teletype if and only if it succeeded in starting a top procedure. If a failure to start a top procedure is due to lack of memory, a console free message will be typed out [Sec 6.5].

1.1.2 Control Z on an Active Console

The effect of ^Z in the second case, that of an active console, is intended to be such that the user at the console may cause control of his console to revert to higher level procedures and ultimatly to his top level procedure even in the face of hostile inferiors.

A ^Z typed on an active console is ignored by ITS if the console is being controlled by its top level procedure. Otherwise it sets a flag associated with the console so that it may not be assigned downward in the procedure tree [Sec 3.2.1.1]. This flag is cleared only by typing any character other than ^Z on the console. ITS also sets the ^Z interrupt bit, a class one interrupt [Sec 4.2], for the procedure controlling the console and possibly for an arbitrary number of procedures, each the immediate superior of the last, extending upward in the tree from the procedure controlling the console. The condition for each step upward in the procedure tree is that either the superior to the procedure currently having its ^Z interrupt bit set is stopped or the current procedure was found to

already have its ^Z bit on.

Action on ^Z interrupt bits set will be taken in less than one quantum time when the scheduler [Sec 4.4] next runs.

1.2 Trapping Instructions

With the exception of ^Z all commands to ITS are given by instructions which trap to the monitor. All valid commands are from a class of instructions called UUO's which are characterized by an initial octal digit of 0. These are discussed in section 1.2.1 and listed in Appendices A and B. Section 1.2.2 concerns all other trapping instructions. It is a useful characteristic of the PDP-6 and PDP-10 that an identical effective address calculation is performed on all words fetched as instructions regardless of their operation code or legality [Ref 1]. Thus indexing and indirect addressing are available on all trapping instructions. Interrupts to the monitor as a result of conditions encountered in the execution of an instruction (such as memory protection violation, AR overflow, PDL overflow, etc) rather than the type of instruction are discussed in section 4.2.

1.2.1 UUO's

The execution as instructions of words with an operation code [Ref 1] of from 000 to 077 normally results in the instruction word being stored in absolute location 40 modified by having had its effective address computed and stored in its address field and its indirect and index fields cleared. The instruction in absolute location 41 is then executed. Since all ITS system calls utilize only UUO's 040 through 047 the Project MAC AI Group's PDP-6 has been modified so that UUO's 001 through 037 and 050 through 077 will trap, as described above, but directly to the user program's relocated core image with no extra overhead. UUO 000 will also appear to trap to the user but this occurs via monitor simulation that checks to see if relocated location 41 directly addresses (indexing or indirect addressing ignored) a location between 20 and the location 6 less than the top of the user's core image inclusive. If so, a JSR to that location is simulated. If not, the user's illegal instruction interrupt bit, a class two interrupt [Sec 4.2], is turned on and a schedule [Sec 4.4] immediately performed.

Complete lists of system calls appear in numeric order in Appendix A and in alphabetic order in Appendix B. However, a list classified by operation code is included here:

UUO	Symbol	Description
040	.IOT	Executed for each item or block of data transmitted between a user program and a symbolic input-output device [Sec 2.3].
041	.OPEN	Used to initialize input-output between a device and a program [Sec 2.2].
042	.OPER	A class of system calls further decoded by the value of their effective address.
043	.CALL	A class of system calls further decoded by the value of their accumulator field.
044	.USET	Executed by procedures to examine and set some of the system variables associated with their inferior procedures [Sec 5.2.3].
045	.BREAK	Executed by a procedure to signal its superior [Sec 5.3.2].
046	.STATUS	Used to ascertain the status of an input-output device, channel, or transfer [Sec 2.5].
047	.ACCESS	Used in input-output to randomly addressable devices [Sec 2.7.4].

ITS system calls were designed to be numerically rather than symbolically decoded to decrease their cumbersomeness and increase monitor efficiency.

1.2.2 Non-UUO Trapping Instructions

The only other trapping instructions are those whose first octal digit is 7 and the instruction JRST with the 10 or 4 bit on in its accumulator field [Ref 1]. The first of these two includes all instructions which effect hardware input-output and most of those affecting the state of the PDP-6 instruction processor (including user/executive mode). In the second the 10 and 4 bits respectively dismiss a hardware interrupt and stop the PDP-6. The execution of

any of these trapping instructions causes an illegal instruction interrupt to the user (a class two interrupt [Sec 4.2]) unless the user's procedure is in iot-user mode [Sec 6.8.3]. The remaining way to affect the state of the processor is a JRST instruction with the 2 bit on in the accumulator field; however, in user mode this will not effect the state of the user, special or iot-user [Sec 6.8.3] modes except that it may turn off iot-user mode.

user core image input-output is negligible in comparison to any significant processing of the data. Using system buffers has the following advantages:

(1) The user may be interrupted, swapped out, moved in core, or otherwise molested during an .IOT [Sec 2.3] as it is the software transfer of data between system buffers and his core.

(2) Real input-output transfers may proceed without consideration of the state of the user program they are occuring for as they are into or out of system controlled memory.

(3) Higher memory efficiency is obtained due to the dynamic nature of major file device buffers. Even for devices with fixed buffers, such as the line printer, only one large buffer need exist regardless of how many programs that could use the device are in the system.

(4) User programs need not concern themselves with the size of physical blocks on devices.

Some devices do not fit this framework very well and system calls relating to them are included in section 6. Miscellaneous calls and special features related to particular symbolic devices are, however, included under the device in section 3.

2.2 The .OPEN UUO

```
CALL:       .OPEN CHNUM,FILNAM
            ;error return
            ;normal return

FILNAM:     MODE,,(SIXBIT /DEV/)
            SIXBIT /FILNM1/
            SIXBIT /FILNM2/
```

Input-output on a particular channel is initialized by executing a .OPEN. The channel is numerically specified by the accumulator field of the .OPEN (CHNUM in the illustration above) and any previous transfer of that channel is terminated as if by a .CLOSE [Sec 2.7.1]. If the user is interrupted [Sec 4.2] during a .OPEN the channel may have been closed and not yet reopened. The effective address of the .OPEN should point to the first word of a block of three words in the user's core image which specify a device, mode, and file. The second two words specify the two file names (FILNM1 and FILNM2 in the above illustration). As a convention, the file names are usually thought of as up to six left-justified sixbit characters rather than thirty-six bit quantities. Some devices ignore the supplied file names. Other devices augment the file names by the procedure's system name as explained in section 2.2.3 below. These properties are indicated in the device descriptions in section 3. The first word has the three character sixbit device name in its right half (DEV in the above illustration) and the direction and mode in its left. Immediatly below are the standard mode bits from this left half.

Bit(s)	Meaning
4.9	Always ignored.
4.9-3.4	Device dependent but using zero should be safe.
3.3	1=>Image mode. 0=>ASCII mode, alias character mode.
3.2	1=>Block mode. 0=>Unit mode.
3.1	1=>Output. 0=>Input.

The device and file specified will be altered by entries in the translation table [Sec 2.2.2] made by either the procedure or any of its superior procedures. Because of this translation and the symbolic nature of .OPEN, it is one of the slower ITS system calls.

If the .OPEN is successful it will skip the immediately following instruction. If not successful it will not skip and will set a "most recent channel in error" system variable associated with the procedure to the channel number on which the .OPEN was attempted. The reason for the failure may be ascertained with a .STATUS [Sec 2.5] or via the ERR device [Sec 3.4.4]. Zero file names are not allowed on many devices and attempts to use them will cause the .OPEN to fail. Any .OPEN closes previously opened devices on that channel whether or not the .OPEN is successful.

2.2.1 File Directories

For devices with true file structure a file directory of the device may be read by trying to input the file with the name ".FILE.

(DIR)". If character mode is used, a readable file directory terminated by a ^L and ^C results. If this pseudo-file is read from the DSK, DKn, or Pnm device, only files with the system name [Sec 2.2.3] of the reading procedure will be listed. Both block and unit modes are available [Sec 2.3]. For non-file devices character input of this file name will yield the string "NON-DIRECTORY DEVICE". Attempted binary input will fail (the .OPEN will not skip) except for the UTn devices [Sec 3.1.2] and the DSK family of devices [Sec 3.1.1, 3.1.3] where an actual device dependent binary of the directory will be read. For the DSK family of devices a pseudo-file "M.F.D. (FILE)" is also recognized by the system and yields a list of the system names [Sec 2.2.3] for which directories exist on the disk. In general files may be written with these pseudo-file names and then renamed [Sec 2.4] so their contents can be read back.

In an ITS-produced readable file directory, an "*" by a file name indicates that the file is inaccessable. This may be due to the fact that it is just then being written or it may have been deleted while open for reading and will vanish when closed.

2.2.2 The Translation Table

Entries in the translation table may be made or deleted by any procedure and have effect only at .OPEN time for the procedure making the entry and its inferiors. The entry consists of the following:

(1) the UNAME and JNAME of the procedure making the entry, (2) the JNAME of the procedure it is to apply to ("*" means all JNAME's), (3) whether it is to apply to input, output, or both, (4) whether its resultant is to be considered final or be retranslated, (5) the from device and pair of file names, where "*"s in from positions match any device or file name, and (6) the to device and pair of file names, where "*"s in to positions indicate no substitution.

At .OPEN time all entries in the table are examined to see if the device and file names of the .OPEN and the UNAME and JNAME of the procedure executing the .OPEN match them. If so and the procedure which made the entry is the same or superior to the procedure executing the .OPEN, the substitution for device and file names specified by the entry is made and unless prohibited by the entry this process is repeated. Should eight successful and sequential translations be made the .OPEN will fail leaving a "too many translations" indication [Sec 2.5] and no further attempt will be made to translate.

2.2.2.1 The .TRANS UUO

```
CALL:       .TRANS FILNAM
            ;error return
            ;normal return

FILNAM:     MODE,,(SIXBIT /DEV/)
            SIXBIT /FILNM1/
            SIXBIT /FILNM2/
```

This system call is expected to point to a three word block in the same manner as a .OPEN [Sec 2.2]. It normally skips and returns with the block replaced by its translation (the block pointed to by a .OPEN is never changed by the execution of the .OPEN). If too many levels of translation are required it will not skip and the block will be unchanged.

2.2.2.2 The .TRANAD UUO

```
CALL:       .TRANAD TBLOCK
            ;error return
            ;normal return

TBLOCK:     UNAME           ;UNAME of procedure making & subject to
entry
            JNAME           ;JNAME for procedure subject to entry
            ATMIO,,DEV1     ;4.9 = 0 => retranslate
                            ;4.9 = 1 => atomic, don't retranslate
                    ;3.2 = 1 => applies to output .OPEN's
                    ;3.1 = 1 => applies to input .OPEN's
                    ;DEV1 = from device
            FN11    ;from file names
            FN12
            ,,DEV2  ;to device
            FN21    ;to file names
            FN22
```

This system call makes new entries in the translation table. All information for the entry is specified by an eight word block pointed to by the UUO (TBLOCK in the illustration above). Before trying to insert a new entry in the translation table, this system

call simulates a .TRANDL pointing at the same block to eliminate redundant entries. If a .TRANAD is successful in creating an entry it skips. If not, because the first entry in the block is incorrect or the translation table is full, it returns without skipping.

2.2.2.3 The .TRANDL UUO

This system call deletes entries in the translation table and normally points at an eight word block identical to the one which was used by the corresponding .TRANAD. See section 2.2.2.2 immediately above for the contents of this normal block. It may, if appropriate, modify the ATMIO bits of an entry for both input and output to delete one or the other. It skips if successful in deleting an entry or removing one of the directions to which it applies. If it effects no change in the translation table, it returns without skipping. It also has a special form where the first word of the block is zero (not a possible UNAME). In this case all entries previously made by the procedure doing the .TRANDL relating to the JNAME in the second word of the block are deleted. This special form always skips and does not examine the remaining six words of the block.

2.2.3 System Names

In order to distinguish the files of different users of ITS on shared devices with common file directories a third file name is used. This file name is refered to as a procedure's "system name" or SNAME.

In readin, writing, deleting, etc. files on the DSK and related devices [Sec 3.1.1] and on the core link devices [Sec 3.4.3], ITS obtains a third file name from the SNAME [App D] associated with the procedure. This variable is set, when the procedure is initially created, to the common UNAME [App D] of the procedure tree [Sec 5.1] the procedure is in. This is normally the name the user has logged in as [Sec 6.1] so his procedures will initially refer to his files. However, a procedure may examine and change its own SNAME by means of an .SUSET [Sec 4.5] and may examine and change the SNAME of any of its inferiors with a .USET [Sec 5.2.3].

The system name facility is also used to simulate certain pseudo-devices as explained in sections 3.1.3 and 3.4.5.

2.3 The .IOT UUO

```
;unit mode
CALL:       .IOT CHNUM,DATLOC
            ;return

DATLOC:     ;place char or word read into or written from


;block mode
CALL:       .IOT CHNUM,PNTR

PNTR:       -LENGTH,,DATLOC  ;pointer, modified during transfer

DATLOC:     BLOCK LENGTH     ;block read into or written from
```

All actual data transfers to or from devices being handled via input-output channels are initiated by .IOT's. The accumulator field of a .IOT should contain the number of an input-output channel (CHNUM in the above illustrations) that has been set up for transfer in a particular mode between the procedure and a particular device by a successful .OPEN [Sec 2.2]. If the channel has not been .OPEN'ed the procedure will receive an input-output channel error interrupt (a class two interrupt [Sec 4.2]). Also the number of the channel on which the .IOT was attempted will be remembered as the most recent erroneous channel for use by the ERR device [Sec 3.4.4]. The effective address of a .IOT is used somewhat differently depending on whether the channel is open in unit or block mode.

For unit mode (the first of the two illustrations above) the effective address points to a word to be written from or read into.

For character unit modes the character is right justified and the rest of the word is ignored on output and zeroed (except as mentioned in section 2.3.1) on input. Some devices, which are classified as input, use the contents of the effective address as an argument to determine what they will store back (see devices IMX, VID, and NVD).

Block modes always read or write a contiguous block of words. The contents of the effective address of the .IOT is interpreted as having minus the length of the block in its left half and the address of the first word in its right half (see second illustration above). Block mode normally treats each word in the block identically to the corresponding unit mode except that block character modes pack five characters of seven bits per word left adjusted. The block pointer word pointed to by the .IOT is advanced as the transfer progresses so that if it completes the left half will be zero and the right half will point to one greater than the last word processed. If a procedure is interrupted out of a block mode .IOT the pointer word will reflect the progress of the transfer when it was interrupted.

Block character input-output from some devices which are naturally one character at a time is limited to blocks of length 77777 words or less and the top three bits of the pointer word are used during the transfer as a character count.

When a .IOT returns to the user without causing an input-output channel error, the transfer it requested will have been completed for unit mode input or output or block mode output. For block mode

input, the count in the pointer word will indicate the amount transfered which may be less than that requested if an end of file is reached [Sec 2.3.1].

2.3.1 The End of File Condition

For devices on which the end of file while reading condition is meaningful it is signaled to the user in various ways depending on the transfer mode. For character devices in the character at a time mode a ^C will be read with the left half of the word read into set to minus one. In block character mode the last word will be filled out with ^C's. Usually the pointer word will not have counted out as the end of file did not occur on a block boundary. For word devices, the user must normally determine the logical end of file from the data being read although physical end of file is detectable in block mode by the lack of advancement of the pointer word. On some devices, attempts to read beyond an end of file will cause the input-output channel to be automatically .CLOSE'ed [Sec 2.7.1] and further .IOT's will cause input-output channel errors (class two interrupts [Sec 4.2]).

When packed characters are being read from a word device the physical block is normally filled out with ^C's but some older files on DEC tape [Sec 3.1.2] are filled out with the character whose ASCII value is 141 ('a"). The end of file character for a particular file

can be determined after the file has been opened by putting the number of the input-output channel it is open on in the accumulator specified by an .EOFC. This channel number will be replaced with the end of file character for that channel by the execution of the .EOFC. See following illustration:

```
                MOVEI AC,CHNUM
CALL:           .EOFC AC,       ;replaces CHNUM in AC with eof character
                ;return
```

2.3.2 The Device Full Condition

Attempted output on a full file structured device (UTn, DSK) results in an input-output channel error (a class two interrupt [Sec 4.2]) for the outputing procedure. This interrupt will occur for the .IOT that failed in such a way that it may be successfully resumed if the procedure's interrupt routine or some other procedure deletes material from the full device. The channel on which the error occured is saved by ITS for the ERR device.

2.3.3 Non-Recoverable Data Errors

ITS normally encounters non-recoverable data errors when reading into its buffer for the user asynchronously with any UUO's by the user. After a fixed number of attempts it accepts the possibly erroneous data which it will later give to the user if he reads far enough. It also sets a flag related to the channel so that the next .IOT performed on it will give an input-output channel error interrupt. The purpose of this flag is to avoid giving the user a channel error interrupt with no indication of which channel it is on. The user's program may return to the .IOT and continue reading the file after handling the interrupt.

2.4 The .FDELE UUO

The system call .FDELE is used for deleting and renaming files. It points at a five word block which, for the three types of .FDELE's, contain the following:

```
CALL:       .FDELE FBLOCK
            ;error return
            ;normal return
```

```
;file delete
FBLOCK:     -,,DEV          ;device on which to find file
            FN1             ;names of file to be deleted
            FN2
            0               ;zero
            -               ;unexamined


;normal file rename
FBLOCK:     -,,DEV          ;device on which to find file
            FN11            ;old file names
            FN12
            FN21            ;new file names
            FN22


;rename of file while open for writing
FBLOCK:     -               ;unexamined
            0               ;zero
            CHNUM           ;channel on which open
            FN1             ;new file names
            FN2
```

An .FDELE on a device that is not truly file structured has no effect but skips as do all successful effective .FDELE's. If a .FDELE does not skip the reason for its failure may be determined with a .STATUS as though the .FDELE had been a failing .OPEN on channel zero [Sec 2.5].

2.5 The .STATUS UUO

```
CALL:       .STATUS CHNUM,DATLOC
            ;return

DATLOC:     ;location status information stored in by call
```

The accumulator field of this system call is the number of an input-output channel (CHNUM in the above illustration) whose status word replaces the contents of the effective address (DATLOC in the above illustration) of the UUO. This status word is described in detail by the tables below; however, its general composition is as follows: the left half relates to the channel and is set by failing .OPEN's, failing .FDELE's and input-output channel errors; the right half relates to the device, if any, open on the channel and the state of the transfer between it and the user.

Bit(s)	Meaning
1.1-1.6	ITS physical device code (see table below).
1.7-1.9	Mode in which device was opened [Sec 2.2].
2.1	1 => buffering capacity empty.
2.2	1 => buffering capacity full.
2.3-2.9	Device dependent.
3.1-3.6	Set by failing .OPEN's and .FDELE's (see table below).
3.7-3.9	Set by interpreted display [Sec 3.4.1] input-output channel errors (see table below).
4.1-4.5	Set by non-display input-output channel errors (see table below).
4.6-4.9	Always zero.

The folowing is a table of ITS device codes (note that the 40 bit

indicates that the file names used are significant and the 20 bit indicates a software device):

Code	Symbol	Device
1	Tnm	Teletype (KSR 35/37).
2	Tnm	GE display and keyboard.
3	LPT	Line printer.
4	VID	Old vidisector ("TVB").
5	NVD	New vidisector ("TVC").
6	PLT	Calcomp plotter.
7	PTP	Paper tape punch.
10	IMX	Input multiplexor (A-D).
11	OMX	Output multiplexor (D-A).
12	PTR	Paper tape reader.
13	DIS	DEC 340 display.
14	IDS	Interpreted display.
15	COD	Morse code transmitter.
16	COD	Morse code receiver.
21	NUL	Null device.
41	UTn	DEC tape.
43	DSK	2311 disks.
60	USR	A not immediately inferior procedure.
61	USR	An inferior procedure.
62	CLx	Core link device.
63	-	The directory device [Sec 2.2.1].
64	USR	PDP-10 device.

The following is a list of the codes left by failing .OPEN's and .FDELE's:

Code	Reason
1	No such device.
2	Wrong direction.
3	Too many translations.
4	File not found.
5	Directory full.
6	Device full.
7	Device not ready.
10	Device not available.

11	Illegal file name.
12	Mode not available.
13	File already exists (attempted rename).
14	Bad channel number (attempted open rename).
15	Link depth limit exceeded (DSK).
16	Pack not mounted (Pnm).
17	Directory not now available.
20	User doesn't exist (DSK or USR).
21	Local device only.

The following are the error codes left by the 340 interpretive display routines:

Code	Meaning
1	Illegal scope mode.
2	Scope hung.
3	More than 2000 words scope buffer.
4	Memory protection violation.
5	Illegal scope operation.
6	Memory protection violation on PDL pointer.
7	Illegal parameter set.

The following are the non-display input-output channel error codes:

Code	Meaning
3	Non-recoverable data error on read [Sec 2.3.3].
4	Non-existant sub-device (such as IMX channel [Sec 3.3.3]).
5	Over .IOPOP [Sec 2.6.1].
6	Over .IOPUSH [Sec 2.6.1].
7	Channel does not have a procedure open on it.
8	Channel not open.
9	Device full[Sec 2.3.2].

2.6 The Input-Output Channel Push Down Facility

For greater flexibility in input-output than that available with the basic input-output channel system [Sec 2.1], a facility is provided whereby a limited number of input-output transactions may be stored for later resumption. Meanwhile the channel they were being effected on may be otherwise utilized.

2.6.1 The .IOPUSH and .IOPOP UUO's

```
CALL:     .IOPUSH CHNUM,   ;push channel number CHNUM
          ;return

CALL:     .IOPOP CHNUM,    ;pop channel number CHNUM
          ;return
```

These instructions treat input-output channels as the PUSH and POP instructions [Ref 1] do memory locations. A channel may be .IOPUSH'ed regardless of whether it is open or not and any transfer in progress on it will be inaccessable until the slot occupied by the information pushed is .IOPOP'ed (possibly by .IOPDL [Sec 2.6.2]) into a channel (possibly different from that from which it was .IOPUSH'ed). The error status and .ACCESS [Sec 2.7.4] pointer for a channel are also correctly stored and restored. Executing a .IOPOP into a channel first .CLOSE's the channel. Each procedure's

input-output channel push down list [App D] has space for eight entries.

2.6.2 The .IOPDL UUO

```
CALL:       .IOPDL          ;reset input-output channel pdl
            ;return
```

This system call takes no arguments and is intended to reset a procedure's input-output channel push down list. Entries on a procedure's input-output channel push down list actually contain the number of the channel that was pushed to produce the entry. This information is used only by .IOPDL whose execution is equivalent to the number of .IOPOP's [Sec 2.6.1] equal to the number of entries in the procedure's input-output push down list and which .IOPOP each entry back into the channel from which it was .IOPUSH'ed.

2.7 Miscellaneous Input-Output Related System Calls

The folowing system calls relate to most or all devices handled via input-output channels.

2.7.1 The .CLOSE UUO

```
CALL:       .CLOSE CHNUM,   ;close input-output channel
            ;return
```

This system call closes the input-output channel whose number (CHNUM in the above illustration) is in its accumulator field. It has no effect if the channel is not open. Executing a .IOT on the channel without reopening it will result in an input-output channel error. For devices with true file structure it is at close time that a file with the same name as one being written is deleted.

2.7.2 The .RESET UUO

```
CALL:       .RESET CHNUM,   ;reset input-output channel
            ;return
```

This system call is intended to reset system buffer pointers so that buffered data on an input-output channel (specified by the accumulator field as with CHNUM in the above illustration) will be ignored. It is currently implemented for the following devices: TNM, TTY, LPT, PLT, PTR, PTP, TVC, and USR. For the USR device it has special effects [Sec 3.4.2].

2.7.3 The .ITYIC UUO

```
        MOVEI AC,CHNUM  ;set up for channel number CHNUM
CALL:   .ITYIC AC,      ;get interrupt character
        ;error return   ;none available
        ;normal return  ;got one
```

It is frequently desirable to examine the characters of an incoming stream at the interrupt [Sec 4.2] level of a procedure (for "quit" features, etc.). This system call provides the facility to read these characters from channels on which devices are open that will provide appropriate interrupts (TTY and Tnm). The accumulator referred to by an .ITYIC should have an appropriate channel number placed in it (see illustration above). The execution of an .ITYIC will then replace this number with the character read and skip. If no character is available the .ITYIC will return without skipping and if an improper channel is specified the procedure will receive an illegal instruction interrupt.

2.7.4 The .ACCESS UUO

```
CALL:   .ACCESS CHNUM,LOC  ;set channel pointer to value LOC
        ;return
```

This system call is intended to be used in accessing randomly

addressable devices. A pointer is associated with each input-output channel which is set to the effective address of a .ACCESS executed with the channel's number (CHNUM in the above illustration) in its accumulator field. This pointer is set to zero whenever the channel is closed. Currently this pointer influences only the USR device [Sec 3.4.2] but it is hoped that random access will be added to the disk [Sec 3.1.1] routines soon.

3. Properties of Particular Devices

Properties and system calls related to particular devices handled via input-output channels are discussed below. Lower case letters in a device name indicate that there are several devices distinguished by numeric digits in their place. The phrase "all standard modes" in descriptions below implies that all combinations of ASCII/image and unit/block modes are available and that ASCII mode implies seven bit ASCII characters and image mode thirty-six bit binary words.

3.1 File Structured Devices

3.1.1 The DSK, DKn, and Pnm Devices

These symbolic devices represent IBM 2311 units attached to the PDP-6. These disks are characterized by interchangeable disk packs. A single logical directory is maintained by the system for the entire disk volume. Files are identified by two file names [Sec 2.2] and a system name [Sec 2.2.3]. If a file is being read, which of the above symbolic device names is used is entirely immaterial. There are some special aspects to reading directories from th ese devices as explained in section 2.2.1.

If a file is being written, the DSK device will physically write

it on a particular drive whose number is assembled into the system. In contrast the DSn device will write the file on the n'th drive. The Pnm device will try to write on a pack with a particular two digit number.

The actual directory for these devices consists of a master directory for each drive and a user directory for each system name that has been used to write files on the disk. The maximum number of user directories is one hundred. The maximum number of files per user is not fixed as the area in a users directory, used to describe the location of the blocks each file consists of is dynamically allocated. When as is occasionally necessary a garbage collection, to compact the information in a user directory, occurs, the system job [Sec 6.5] prints out a message on its teletype. This warning was added when the garbage collector was less reliable than it now is.

The master directory for a drive is read in the first time the drive is referenced. A copy is written back out whenever the disks are idle and the master directory has been changed since it was last written out. User directories are read in when first referenced. If a currently nonexistant user directory is referenced to write it will be created. If referenced to read, a special open lose [Sec 2.5] will indicate this to the user. User directories are copied out whenever the disks are idle and they have been changed since last

copied out. A user directory in memory may be erased if no files are open in it, a current copy has been written out, and a core request provokes the core allocator [Sec 4.3] to examine the status of an area of memory including the buffer.

The IBM 2311 interface transfers information directly to memory in units of 2000 words. Dynamically allocated buffers in system memory are used for input and output. The capacity of each pack is about thirty times that of a reel of DEC tape [Sec 3.1.2].

3.1.1.1 Links

```
CALL:       .FDELE FBLOCK
            ;error return
            ;normal return

FBLOCK:     200000,,[SIXBIT /DSK/]
            SIXBIT /FFNAM1/
            SIXBIT /FFNAM2/
            SIXBIT /TFNAM1/
            SIXBIT /TFNAM2/
            SIXBIT /TSNAME/
```

A speical feature of the disk device is that users may establisn symbolic links. If the call illustrated above has been executed, then an attempt to read the file FFNAM1 FFNAM2 with the system name [Sec 2.2.3] that was being used by the procedure that established the link will actually refer to file TFNAM1 TFNAM2 with the system name TSNAME. A link may refer to another link and so on for up to eight

levels. Attempting to write or delete through a link writes on top of or deletes the link rather that the file it refers to.

3.1.2 The UTn Devices

The devices UT1, UT2, UT3 and UT4 represent the four DEC tape drives on the PDP-6. They are true file structured devices under ITS with all standard modes available. Data is dynamically buffered by ITS for both input and output. The file directory of a tape is read in by the system and retained when a drive is referenced and there is no directory being retained for it. An updated directory will be written out on a particular tape whenever the directory has been changed, no files are open on the tape, and no data transfers are in progress on any drive. Files on UTn do not have a system name [Sec 2.2.3] associated with them. Directories can be excised from core only by the .UDISMT UUO.

3.1.2.1 The .UDISMT UUO

```
        MOVEI AC,TAPEN
CALL:   .UDISMT AC,
        ;error return
        ;normal return
```

This system call, if successful, causes a DEC tape's file

directory to be excised from core and the tape to be physically demounted. The drive number must be in the accumulator specified by the UUO. Manually removing tapes or switching drive numbers of drives for which ITS is retaining a directory may result in out-of-date file directories left on tapes and directories for one tape being written on another.

A .UDISMT skips if and only if successful. It will fail if any files are open on the tape. If a file is opened, deleted, or renamed on a tape while the tape is being dismounted by ITS the dismount will be aborted and the tape retained.

3.1.2.2 The .ASSIGN and .DESIGN UUO's

```
        MOVEI AC,TAPEN
CALL:   .ASSIGN AC,
        ;error return
        ;normal return
```

```
        MOVEI AC,TAPEN
CALL:   .DESIGN AC,
        ;error return
        ;normal return
```

These system calls are to enable a user to protect against other users accidently referencing his DEC tape to write or delete a file or rename [Sec 3.1.2.4] the tape. Both specify the drive numbers as

the contents of their specified accumulator. The .ASSIGN UUO skips if successful and results in attempts to write on the designated drive failing unless the writing procedure has the same system name [Sec 2.2.3] as the UNAME of the procedure that executed the .ASSIGN. For a procedure in a disowned tree [Sec 5.2.2], an .ASSIGN will be treated as an illegal instruction. An .ASSIGN will fail only if the drive is already assigned. The .DESIGN UUO skips if successful and causes the drive to be unassigned. It fails if the drive is already unassigned or assigned to a different user. A drive may also be .DESIGN'ed by a .UDISMT [Sec 3.1.2.1] and the .ASSIGN'ing of a drive does not protect against other users dismounting it.

3.1.2.3 The .UBLAT UUO

```
                MOVEI AC,TAPEN
CALL:           .UBLAT AC,
                ;error return
                ;normal return
```

This system call is intended for reading non-MAC format [Ref 4] (such as DEC format) DEC tapes. The contents of the specified accumulator must be a drive number for which ITS is not retaining a directory. If a directory is already being retained, the .UBLAT will fail and return without skipping. (The user might try first .UDISMT'ing the drive.)

If successful, the .UBLAT skips. Instead of attempting to read in the directory from the tape mounted on the specified drive, ITS internally marks the drive so that its contents may not be changed (no writes, deletes, or renames). At this point a successful .OPEN of the tape may be executed with any file name and all blocks of the tape will be read consecutively. Thus the tape may only be referenced to read the file containing the entire contents or to .UDISMT [Sec 3.1.1] it.

3.1.2.4 The .UTNAM UUO

```
        MOVE AC,[(SIXBIT /111/),,TAPEN]
CALL:   .UTNAM AC,
        ;error return
        ;normal return
```

This system call changes the three character tape name of a particular DEC tape. The tape drive number must be specified in the right half of the specified accumulator. The call will fail and return without skipping if there is no such drive or is the drive is in .UBLAT mode [Sec 3.1.2.3] or .ASSIGN'ed [Sec 3.1.2.2] to another user. Otherwise it modifies the tapes directory (reading it in if necessary) so as to have a tape name corresponding to the left half of the accumulator specified by the call and skips on return.

3.1.2.5 The .UINIT UUO

```
        MOVE AC,TAPEN
CALL:   .UINIT AC,
        ;error return
        ;normal return
```

This system call initializes the directory for the DEC tape on the drive whose number is the contents of the accumulator it specifies. The tape must have been .ASSIGN'ed [Sec 3.1.2.2] to the user doing the .UINIT. If successful the .UINIT skips on returning. If the specified drive is nonexistent or the tape on it not .ASSIGN'ed to the user doing the .UINIT, it returns without skipping.

3.1.3 The COM and SYS Devices

The device SYS is used for storage of systems programs and the "message of the day" [Sec 7.1.1]. It is currently those files on device DSK with system name SYS. Reading or writing device SYS causes one to reference DSK as though one's system name were momentarily SYS. The device COM is used for commonly used user files and the mail feature [Sec 7.2.1, 7.2.2]. In a manner similar to device SYS it is DSK with system name COMMON.

3.2 Unit Character Oriented Devices

3.2.1 The Tnm and TTY Devices

Device T00 is the console teletype. Devices T11 through T14 (the two digits are treated as an octal number) are GE Datanet 760 terminals with a character only display and keyboard. Devices T01 through T10 are available for more teletype and teletype-like hardware and are read and written through the Knight Teletype Kludge. Device TTY for a particular procedure is the console controling the procedure tree it is in. All input-output to these devices is in ASCII characters and the "image" mode [Sec 2.2] has special meanings. Further detail on which Tnm is "where" is given in the following table (as of July 14, 1969):

Tnm	Location
T0	Main console (by PDP-6).
T1	Advanced Remote Display Station.
T2	Outside line (or 1479 depending on switch).
T3	Robot console.
T4	PDP-10 console.
T5	System console (by line printer).
T6	Inside line 1425.
T7	Inside line 1474 (or outside depending on switch) 15cps.
T10	Spare.
T11	GE console by plotter.
T12	GE console on 8th floor.
T13	GE console by air conditioners.
T14	GE console by ham rig (in far corner).
T15up	Nonexistent.

A procedure may not open its console by refering to it as Tnm for the corresponding n and m. Only one procedure at a time may have a teletype open as a device rather than a console. The procedure with TOO opened as a console (TTY) and that actually has control of it [Sec 3.2.1.1] can always seize the DEC 340 display [Sec 3.4.1].

Image mode output to KSR 35/37's is very straight forward and just sends the character to the teletype unmodified. ASCII mode differs only in the following: (1) the character whose value is 33 is printed as a "$"; (2) other characters with a value less than 40 that are not format effectors or bell are printed as a "^" followed by the character with an ASCII value 100 greater; (3) tab is simulated with spaces; (4) the "delete" character doesn't type out and (5) new line is simulated for a carriage return. Output to GE terminals is the same as ASCII mode output to KSR 35/37's except for the following: (1) all line feeds (12) are ignored, (2) a ^L (form feed) clears the screen, (3) an automatic new line is inserted if the right margin is touched, (4) an "⌐", as displayed by the GE terminal, is used instead of a "^" prefix, (5) output beyond the bottom of the output area wraps around and overwrites from the top of the output area which does not include the bottom three lines unless the first input .OPEN [Sec 2.2] had the 3.4 mode bit on, (6) finally, if the output is in image mode only, the character ^T will allow overwritting starting from the top of the output area without clearing the screen or wrapping off the bottom. For all the above output modes, adding

block mode only affects individual character processing by causing ^C's that are output to be ignored.

Input to keyboards (which are all full duplex) is buffered by ITS in relatively small fixed size buffers. Striking a key when this input buffer is full results only in a ^G (bell) (or "?" on a GE terminal) being output as echo. Any input when the keyboard is not in use, except ^Z [Sec 1.1.1], is ignored by ITS. Procedures may enable interrupts (class three) due to non-empty input buffers [Sec 4.2] and also examine the input characters at interrupt time with .ITYIC [Sec 2.7.3].

Image mode input is characterized by no ITS-supplied echo and no modification of the character codes unless open mode [Sec 2.2] bit 3.5 ("old mode") is also on, in which case lower case KSR 37 characters are transformed to upper case KSR 35 characters.

ASCII mode input provides echo in approximately the same manner as outputing the characters in ASCII mode. Procedure output has higher priority than echo output. On GE terminals input is normally echoed in the bottom three lines unless the 3.4 open mode bit was on, in which case characters are echoed where output is appearing except that a few characters are not echoed at all. In ASCII mode input, the characters whose values are 175 and 176 are input and echoed as if they were the character whose value is 33. Having open mode bit 3.5 on has the same effect for ASCII mode input as it did for image mode input. In general, all of the various mode bits affecting input

are obtained from the first input .OPEN [Sec 2.2] executed by a procedure for a particular teletype. For all the above input modes, adding block mode results in input until the block input pointer runs out or until a ^C is typed at which point the .IOT [Sec 2.3] being executed terminates as on an end of file [Sec 2.3.2].

Any input or output TTY .OPEN by a procedure that has never had control of a console will fail. Any input or output .IOT or .OPEN executed on a TTY device by a procedure that has had control of the device while it is actually being controlled by another procedure [Sec 3.5.1] will hang until the executing procedure regains the device.

The device dependent .STATUS [Sec 2.5] bits for the teletype devices are as follows:

Bit	Meaning
2.3	Channel open in "DDT mode" (3.4 mode bit).
2.4	A console (TTY device) open on this channel.
2.5-2.9	If channel open for output: Current line number if a GE console.
2.5-2.9	If channel open for input:
2.5	Indicates some characters have been seen at interrupt and not main program level.
2.8	Teletype is at the 340 [Sec 3.4.1] or a 340 slave.
2.9	Teletype is local, not dial in.

3.2.1.1 The .ATTY and .DTTY UUO's

```
CALL:       .ATTY CHNUM,
            ;error return
            ;normal return

CALL:       .DTTY CHNUM,
            ;error return
            ;normal return
```

These system calls enable procedures in a single console controlled procedure tree to transfer control of their console between each other. The execution of either by a procedure that does not have control of the console, because it was taken away by a higher procedure's execution of a .DTTY, will hang until it regains control. The accumulator field of each must specify a channel (CHNUM in the above illustration) on which an immediate inferior procedure is open [Sec 3.4.2]. An .ATTY will pass control of the console to this open procedure unless the procedure executing the .ATTY has control of the teletype by taking it from some deeper inferior along the same procedure tree branch in which case it reverts to the procedure from which it came. An .ATTY will hang instead of doing the above if the last character typed on the console was a ^Z. A .DTTY retreives control of the console from some inferior to the procedure executing it.

Both of these calls skip if successful. An .ATTY will fail to skip only if no inferior is open on the channel it specifies. A .DTTY will fail to skip if no inferior is open on the channel it

specifies or if the procedure executing it never had control of the console or never gave control away. All other conditions blocking the success of either system call cause them to hang.

3.2.1.2 The .LISTEN UUO

```
CALL:       .LISTEN AC,
            ;return
```

This system call is usable only on the console a procedure is associated with. It will hang if control of the console [Sec 3.2.1.1] has been taken away from the procedure executing it and if all procedure output has not been typed. It then returns with the number of buffered input characters in the specified accumulator. If the procedure has never had control of the console or is in a disowned tree [Sec 5.2.2] zero will be returned.

3.2.1.3 The Dial Feature

A feature is available under ITS that allows procedures to dial calls on the two dataphone lines now available. Caution should be exercised in using this feature if one is using the system over one of these lines. The computer may "hang-up" in a more literal sense than usual.

```
.           MOVE AC,[440600+DIALN,,DLBUF]
CALL:       .DIAL AC,
            ;error return
            ;normal return

DLBUF:      010201,,020202
            100506,,101000
```

The .DIAL call is used to initially associate a procedure with a dialer and to actually request dialing on the dataphone line associated with the dialer. The accumulator specified by the call, AC in the above example, should contain a byte pointer directly addressing a byte within the users core image. Indirect addressing and indexing are ignored and in fact the index field is used to specify the dialer number, DIALN in the above example. This corresponds to a teletype as follows:

Dialer	Teletype
0	T07
1	T06

The .DIAL will have no effect and return without skipping if any of the following conditions hold: (1) a nonexistant dialer is specified, (2) the teletype associated with the dialer is in use by a procedure other than the dialing procedure, or (3) the associated teletype is not in use but the dialer is assigned (by the execution

of a .DIAL not yet followed by a .HANGUP (see below)) to a procedure other than the dialing procedure. If the .DIAL skips in returning it will have set up variables in the system so that dialing will proceed asynchronously with the procedure's execution. The byte string, which specifies what is to be dialed, can not be more than five words long. These words are transfered to the system by the .DIAL and dialing will be unaffected by the user's subsequent modification of his image of these words. However a later .DIAL without an intervening .DIALW or .HANGUP (see below) will simply overwrite these words and may cause garbled dialing.

As dialing proceeds, each byte is examined in succession. A zero byte indicates the end of bytes to be processed. A byte whose value is between one and ten causes a similar number of dial pulses to be sent. Interdigital pauses are automatically inserted after each digit. Bytes with a value larger than ten cause a pause in dialing of as many tenths of a second as the bytes value minus ten.

A special means is provided to send the "break" signal. If the accumulator specified by a .DIAL is zero except possibly for the index field (where the dialer number is specified), then a "break" will be sent after a .DIALW has been simulated to avoid garbling any dialing in progress.

```
CALL:       .DIALW DIALN,
            ;error return
            ;normal return
```

This call refers to the dialer specified by its accumulator field (DIALN in the above example). It has no effect and returns without skipping if the dialer has not been assigned to the .DIALW'ing procedure by a successful .DIAL (see above). Otherwise it returns and skips after requested dialing by the dialer is complete.

```
CALL:       .HANGUP DIALN,
            ;error return
            ;normal return
```

This call refers to the dialer specified by its accumulator field (DIALN in the above example). It has no effect and returns without skipping if the dialer has not been assigned to the .HANGUP'ing procedure by a successful .DIAL (see above). Otherwise it returns and skips after hanging up the line for at least three seconds and freeing the dialer so it is no longer assigned to any procedure.

3.2.2 The LPT Device

A six hundred line per minute Data Products, Inc. line printer with one hundred and twenty print positions and sixty four printing

characters is available for character output as device LPT. The ASCII/image mode bit [Sec 2.2] is ignored in .OPEN's on device LPT but either single ASCII characters or blocks of packed characters may be output as indicated by the block/unit mode bit. The following "transformations" are made to the output text: (1) lower case characters are made upper case, (2) characters beyond the one hundred and twentieth print postion are ignored, (3) format effectors (except for vertical tab) are simulated as for a model 35 teletype, thus overprinting can be accomplished by using carriage return without line feed, (4) characters with an ASCII value below 40 other than the simulated format effectors and character 33, which is printed as a "$", are printed as a "^" followed by the character whose ASCII value is 100 greater.

Only one procedure may have the LPT device open at one time but it may be open on more than one channel. Procedures in disowned trees are blocked from opening the LPT device but it will not be taken away from a procedure as the procedure is disowned [Sec 5.2.2]. (Disowned procedures may use the TPL [Sec 3.4.5] device.) A fixed 1000 word buffer in ITS is used for output to this device.

The device dependent .STATUS [Sec 2.5] bits for the LPT device have in the 2.2-2.9 field the current character position in the print line. This count starts at zero at the left margin.

3.2.3 The PLT Device

A CalComp model 565 plotter [Ref 9] is available via "character" output to device PLT. Available with the same modal restrictions as the LPT device [Sec 3.2.2], it is also similar to that device in that it may only be open by one procedure at a time but may be open on more than one channel. A fixed 200 word area is used in ITS to buffer output.

Only the right most six bits of characters output to the PLT device have effect. They are as follows:

Bit	Effect if a one
1.1	drum down
1.2	drum up
1.3	carriage right
1.4	carriage left
1.5	pen up
1.6	pen down

3.2.4 The PTP and PTR Device

The PTP and PTR devices represent, respectively, the sixty three and a third character per second paper tape punch and four hundred character per second photoelectric paper tape reader on the PDP-6 [Ref 1]. They are similar to the LPT and PLT devices [Sec 3.2.2, 3.2.3] in that only one procedure may have each open at one time but that procedure may have the device open on more than one channel

possibly in different modes. Also output or input is buffered in ITS in fixed areas of 20 and 100 words respectively. These devices differ from the LPT and PLT devices in that all standard modes [Sec 4] are available and the following special mode. If in a successful .OPEN for one of these devices, the 3.4 mode bit [Sec 2.2] is on then the 3.3 mode bit is ignored, the 3.2 mode bit must indicate unit mode, and all eight paper tape channels may be written or read from or into the eight rightmost bits of the word addressed by each .IOT [Sec 2.3] executed.

3.2.4.1 The .FEED UUO

```
CALL:       .FEED CHNUM,
            ;error return
            ;normal return
```

This system call checks to see if the PTP device is open on the channel it specifies (CHNUM above). If not, it returns without skipping. If so, it causes one line of blank tape to be punched and skips.

3.2.5 The COD Device

This is a character output device that may be used in block or unit mode but not any form of image mode and not by more than one

procedure at a time. It causes transmission of low power FM Morse code at a frequency just off the bottom of the FM broadcast band. Characters for which there is no standard Morse code (such as line feed) are ignored except that the speed is set by characters whose value is greater than 137 to approximately, seventy five divided by the difference between the character value and 137, words per minute. It tries to interrupt on the input-output channel it is open on when the fixed buffer in ITS it uses is almost empty.

3.3 Robotics Oriented Devices

3.3.1 The NVD, TVC, and VID Devices

The VID and NVD devices are somewhat similar to the IMX device [Sec 3.9] in that they may be open by any number of procedures, have block and unit modes, and "functional" word input where the quantity read is a function of the contents. They differ in that the ASCII/image .OPEN mode bit [Sec 2.2] is ignored and they represent vidissectors. The initial contents of words input into should be the coordinates of a point in the field of view of the particular vidissector at which it is desired to know the light intensity. This contents will be replaced by a function of said intensity. The exact form of these words and the use of any device dependent .OPEN mode bits [Sec 2.2] is listed in the table below.

Item	VID	NVD
X	2.3-1.1	4.5-3.2
Y	4.3-3.1	2.5-1.2
value	1.8-1.1 (reciprocal of intensity)	
		4.9-3.3 (normalized floating reciprocal intensity)
		2.1-1.1 (integer logarithm reciprocal intensity)
(too dark)	400	3.1 (overflow) 3.2 (dim cut off)
dead	-1	4000000
modes	3.6-3.4 (340 intensity)	
		3.5-3.4 (confidence level)
		3.8-3.6 (dim cut off level)

The VID device is of little use as it is a lower quality subset of the NVD device. The deflection signals for the VID device come from the DEC 340 display [Sec 3.4.1] whose use will be degraded by using device VID.

The TVC device is the same as the NVD device except that it should be opened on two channels, one for output and one for input. Words with coordinates in them are output on one channel and stored in a fixed length buffer in system memory. There they are replaced by ITS, at its interrupt level, with the response word shown above (value). They may be read, in the same order as output, on the input channel. Unlike most devices, a block mode .IOT outputting to the TVC device when its buffer is full will not hang up but return with the block .IOT pointer word pointing at the first word not

transfered.

3.3.1.1 The .VSCAN and .VSTST UUO's

(Some of this section is taken from Reference 13.)

```
CALL:       .VSCAN PTABLE
            ;return

PTABLE:     WBIT,,VCONO      ;+0
            -LENGTH,,ARRAY   ;+1
            XRES,,YRES       ;+2
            R1               ;+3
            R2               ;+4
            C1               ;+5
            R3               ;+6
            R4               ;+7
            C2               ;+10
            P1               ;+11
            P2               ;+12

ARRAY:      BLOCK LENGTH
```

This system call allows the user to read the light intensity at an array of points with low overhead and, if desired, overlapped computation.

The effective address of a .VSCAN should point at an eleven word block of parameters defining the points to be scanned, the locations to be read into, the mode they are to be read in, and whether the call should hang until the scan is through. The effect of the parameters, in order, is described below.

The first word has in its right half the control bits for the vidisector as in section 3.3.1 for NVD. The sign bit, if a one, causes the .VSCAN to hang until the scan is over. If a zero, computation by the .VSCAN'ing user may procede while the scanning procedes at ITS's interrupt level.

The second word should be similar to a block mode .IOT [Sec 2.3] pointer word. Its right half should point to the beginning of an array to be read into in the response format of NVD in section 3.3.1. Its left half should be a negative count equal to or larger, in magnitude, than the number of points being scanned. (If in the above illustration either PTABLE+12 or ARRAY+LENGTH-1 were beyond the users memory bound, the .VSCAN would be treated as an illegal instruction.)

The third word has in its left and right halves, as eighteen bit integers, the "X" and "Y" resolution or number of points to be scanned in each direction.

The remaining nine parameters, which are fixed point quantities with the binary point between bits 2.9 and 3.1, define where the XRES by YRES points lie in vidissector coordinates according to the following ALGOL program:

```
FOR Y1<-0 STEP 1 UNTIL YRES-1 DO
BEGIN
        Y2<-(2*Y1+1)/(2*YRES)
FOR X1<-0 STEP 1 UNTIL XRES-1 DO
BEGIN
        X2<-(2*X1+1)/(2*XRES)
TEMP<-P1*X2+P2*Y2+1
```

```
X<-(R1*X2+R2*Y2+C1)/TEMP
Y<-(R3*X2+R4*Y2+C2)/TEMP
ARRAY(Y1*XRES+X1)<-SCAN(X,Y)
END
END
```

A .VSCAN will hang up until the NVD device is free. This device is then seized, not to be released until the scan is over or aborted. A scan is only aborted if the procedure it is being executed for is reset [Sec 3.4.2] or its core size reduced so as to exclude the array being read into (ARRAY in the above illustration) or by .VSTST.

```
        MOVSI AC,1      ;or 0 or 400000
CALL:   .VSTST AC,
        ;return
```

This system call allows later control of and sensing by a procedure that has overlapped computation with a .VSCAN. It is illegal if executed when another procedure is .VSCAN'ing. If the contents of the specified accumulator is positive, the .VSTST hangs till the scan is over. If the contents of the specified accumulator is negative, any scan in progress is aborted. If the contents of the specified accumulator is zero, the user's location then being stored into by the scan, if any, replaces it.

3.3.2 The OMX Device

A multiplexed digital to analog converter on the PDP-6 is available via word output to the OMX device [Ref 5, 6]. Block and unit modes are available and the "ASCII/image" mode bit [Sec 2.2] has a special significance explained below. Each word output should have the desired channel number in bits 4.9-4.4 and the value to be converted in bits 4.3-3.1. Thus channel numbers range from zero to 77 and values from zero to 7777. These output channels decay exponentially with some large time constant so ITS stores the current desired value for each channel and outputs them every half second as long as the OMX device is open by some procedure. Output in "ASCII" mode immediately affects the channel as well as storing a value to be output periodically. Slightly less overhead occurs in "image" mode which only stores a value in ITS and may have no effect for up to half a second. Any number of users may have the OMX device open, possibly on more than one channel and possibly in different modes. Current OMX channel assignments are as follows:

Channel	Effect
0-31	Unused.
32	MVD iris.
33	MVD focus.
34	MVD mirror.
35-55	Unused.
56	TVB pan.
57	TVB tilt.
60	MA3 hand extend.

61 MA3 hand rotate.
62 MA3 hand grasp.
63 Cannon lens zoom (MA3 hand finger #1).
64 Cannon lens focus (MA3 hand finger #2).
65 MA3 hand tilt.
66 Alles hand grip.
67 Alles hand tilt.
70 Alles hand extend.
71 Alles hand rotate.
72 AMF arm wrist roll.
73 AMF arm wrist yaw.
74 TDR horizontal.
75 AMF arm horizontal.
76 AMF arm vertical.
77 AMF arm swing.

3.3.2.1 The .ARMOVE and .ARMOFF UUO's

(Some of this section is taken from Reference 13.)

```
        MOVE AC,[-LENGTH,,ARMBLK]
CALL:   .ARMOVE AC,
        ;normal return
        ;test successful return

ARMBLK: BLOCK LENGTH
```

This system call allows one procedure at at time to exercise special control over several digital to analog multiplexor channels [Sec 3.3.2]. It provides acceleration and velocity limiting, software limit stops, and conversion from joint number to multiplexor channel.

To avoid certain timing problems, the transmission of commands to

the multiplexor control routines in ITS is noninterruptable (but for a limited length of time as only a limited number of commands may be transmitted). Each command is a single word in a block pointed at by the contents of the accumulator specified in the call. The format of these words is as follows:

Bits	Significance
4.9-4.4	joint number
4.3-3.7	command
3.6	unused
3.5	indirect
3.4-3.1	index
2.9-1.1	address or operand

The currently available joint addresses are as follows:

Address	Significance
0	AMF swing
1	AMF vertical
2	AMF horizontal
3	AMF yaw (inoperative)
4	Alles hand tilt
5	Alles hand grip
6	Alles hand rotate
7	Alles hand extend
10	AMF roll (inoperative)

The currently available commands are as follows:

Command	Significance
0	set destination
1	set velocity limit
2	test magnitude of position error
3	test magnitude of velocity

The operand is computed by adding the contents of the specified index register (if any) to the right half of the command. The result is masked to 18 bits. If the indirect bit is a one, the right half of the word addressed is used as the operand. No further indexing or indirecting is interpreted.

The test commands are "true" if the quantity tested exceeds the operand. If any tests are "true" the .ARMOVE will skip.

```
CALL:       .ARMOFF
            ;return
```

The multiplexor control routines are activated by the first .ARMOVE and are turned off by an .ARMOFF or killing the .ARMOVE'ing procedure.

An .ARMOVE is illegal if (1) the arm is in use by another user, (2) the block extends above the user's memory bound, (3) an indirect reference is out of bounds, (4) an unused command code or joint number is specified, or (5) the block is over 100 words long.

3.3.3 The IMX Device

A multiplexed analog to digital converter attached to the PDP-6 is available via word input from the IMX device [Ref 5, 6]. Block and unit modes are available but the "ASCII/image" mode bit [Sec 2.2] has a special significance explained below. Each word input into must initially contain a number from zero to 177. This number will be replaced with the twelve bit digitalization of the analog quantity associated with the multiplexor channel of the same number. If it is desired to read in values converted at each .IOT [Sec 2.4] time then device IMX should be opened in "ASCII" mode. Opening device IMX in image mode causes ITS to continuously read all input multiplexor channels into system core a minimum of about ten times a second. Executing a .IOT on a channel opened in this manner will result in values up to one tenth of a second old but without the overhead necessary to read in new values. Any number of procedures may have the IMX device open, possibly on different channels and possibly in different modes. Current IMX channel assignments are as follows:

Channel	Input function
0-21	Unused.
22	MA3 hand extend.
23	MA3 hand rotate.
24	MA3 hand grasp.
25	Cannon lens zoom (MA3 hand finger #1).
26	Cannon lens focus (MA3 hand finger #2).
27	MA3 hand tilt.
28-32	Unused.

33	NVD iris.
34	NVD focus.
35	NVD mirror.
36	Rhomboidal test table pot A.
37	Rhomboidal test table pot B.
40	Rhomboidal test table pot C.
41	Rhomboidal test table pot D.
42	Pot box upper left.
43	Pot box upper right.
44	Pot box lower right.
45	Pot box lower left.
46–57	Unused.
60	TDR output.
61	Alles hand tilt.
62	Alles hand extend.
63	Alles hand rotate.
64	Alles hand grasp.
65	AMF arm wrist roll.
66	AMF arm wrist yaw.
67–70	AMF arm horizontal high resolution.
71	AMF arm horizontal absolute position.
72–73	AMF arm swing high resolution.
74	AMF arm swing absolute position.
75–76	AMF arm vertical high resolution.
77	AMF arm vertical absolute position.
100–111	Unused.
112	Joy stick X.
113	Joy stick Y.
114–131	Unused.
132	TVC pot box one (manual iris conrol).
133	TVC pot box two (manual focus control).
134	TVC pot box three.
135	TVC pot box four.
136	TVC pot box five.
137	TVC pot box six.
140	New pot box pot one.
141	New pot box pot two.
142	New pot box pot three.
143	New pot box pot four.
144	New pot box pot five.
145	New pot box pot six.
146	New pot box pot seven.
147	New pot box pot eight.
150–155	Unused.
156	Pot box two, one.
157	Pot box two, two.
160	Pot box two, three.

```
161     Pot box two, four.
162     Pot box two, five.
163     Pot box two, six.
164     Pot box two, seven.
165     Pot box two, eight.
166     Joy stick console pot ten.
167     Joy stick console pot nine.
170     Joy stick console pot eight.
171     Joy stick console pot seven.
172     Joy stick console pot six.
173     Joy stick console pot five.
174     Joy stick console pot four.
175     Joy stick console pot three.
176     Joy stick console pot two.
177     Joy stick console pot one.
```

3.3.3.1 The .POTSET UUO

(some of this section is taken from Reference 13.)

```
CALL:       .POTSET PTABLE
            ;return

PTABLE:     BLOCK 4*NUMENTRIES
            0
```

This call gives the user a flexible means of controlling program parameters via the input multiplexor. A maximum of 20 simultaneous connections between pots and variables is permitted. Each may be variable fixed or floating point. If fixed point, it may be an arbitrary byte within a word. The user may specify a mapping from pot values to variable values by giving an upper and lower limit. These may be inverted to give a backward mapping. Two types of

control are provided, absolute and incremental. In absolute mode, the true pot position sets the value. This may be useful for positioning displayed objects with a joystick. The incremental mode permits a variable or set of variables to be changed slightly without causing a discontinuous jump in their values. The value is unchanged at connect time, but rotating the pot adds its scaled increment to the variable. Turning it down in the bottom third, or up in the top third of the pot's range causes a faster change so as to keep the control near center. The increase in gain is inhibited at low speeds to prevent drift due to noise.

The address of the call points to a table of pot connection (or disconnect) specifications. This table consists of blocks of 4 words, followed by a zero word. The block format is as follows:

```
Location Variable
FOO+0   Multiplexor channel,,control bits (see below)
FOO+1   Btye specification(see below),,variable address
FOO+2   Lower limit (value at pot = 0)
FOO+3   Upper limit (value at pot = 100000)
```

The control bits are as follows:

```
Bit     Meaning
2.9     1=>disconnect pot
        0=>connect pot
1.2     1=>absolute
        0=>incremental
1.1     1=>floating
        0=>fixed
```

The byte specification should be in machine byte pointer format (no index or indirect allowed) for a partial word and may be zero for a full word.

A .POTSET call is illegal if either (1) the user tries to connect a pot already in use by another user, (2) the table is more than twenty blocks long, (3) an attempt is made to connect a pot when twenty pots are already connected, or (4) the address exceeds the user's memory bound. A pot is disconnected when (1) the user disconnects it with a .POTSET, (2) the user reduces his memory bound below the address the pot controls, or (3) the job is killed.

3.4 Miscellaneous Devices

3.4.1 The DIS and IDS Devices

The DEC 340 display [Ref 1] differs significantly from all other devices usable in ITS. In order to maintain an image on the display it is necessary to repeatedly output to it a list of display words. This list must be stored in main memory and the various means of using the dispaly may be divided into those that maintain this list in system memory (discussed in this section) and those that allow the list to be maintained in the user's core image [Sec 3.4.1.1].

Only one procedure at a time may use the 340 display and in only

one of its three modes (as symbolic device DIS, as symbolic device IDS, or via the .DSTART and .DSTRTL UUO's). If the display is free, any procedure may seize it by executing a proper .OPEN on device DIS or IDS or by executing a .DSTART or .DSTRTL UUO. If the display is in use, however, a procedure will succeed in any of the above UUO's only if it is the procedure that has the display or it has control of device TOO as a console [Sec 3.5]. If the second case applies, the channels, if any, on which the other procedure had the DIS or IDS open will be modified to the corresponding (block or unit) NUL device [Sec 3.4] output. .CLOSE'ing [Sec 2.8.1] all input-output channels on which device DIS or IDS are open is the same as executing a .DCLOSE [Sec 3.4.1.3].

The DIS symbolic device allows the user to output single or packed blocks of ASCII characters in ASCII mode and single or blocks of 340 display list half words [Ref 1] in image mode. This device may be open on more than one channel at a time in possibly different modes. If the user outputs characters and then halfwords, ITS will automatically insert intervening items to cause the display to escape to paramerer mode. If the user outputs half words and then characters, ITS assumes that he has inserted items to cause the display to escape to parameter mode. A display list produced by output to the DIS device will start by setting the 340 display intensity to 7, its x and y coordinates to zero and 7000, and its scale to one larger than the 3.4-3.5 field of .OPEN mode bits [Sec

2.2] used unless that field is 3 in which case the scale is set to zero. Lines of character output that extend beyond the right edge of the screen will wrap around and continue, superimposed on their beginning, from the left. Lines of character output that extend beyond the bottom of the screen similarly continue from the top. The amount of display information created by DIS output may not exceed 2000 words. After that, further output is ignored. The stored information created by output to the DIS device may be deleted to accept new information by outputing a ^T or a ^L (form feed) character. In the latter case, this action will be delayed for a few seconds.

The IDS symbolic device allows the user to use a semi-idealized display. This device is known as the interpreted display since the user writes "instructions" to be executed by a display processor which are then interpreted by ITS simulating the processor and creating a 340 display list to actually display. The IDS device may only be opened in unit output mode and what the user outputs to it is the location, in the user core image, for the display processor to start "executing".

```
        .IOT DISCHN,PC
        ...
PC:     STRTLC          ;location to start
                ;updated during execution
```

Almost all error conditions encountered during this "execution" cause interrupts to the user [Sec 4.2]. The simulated display processor has a push down list facility with its push down list pointer in the user's location 43. The information to be sent by tne .IOT'ing [Sec 2.4] of each starting location to the IDS device is normally terminated by the display processor POPJ'ing to zero. Frequently this is made to happen by over popping by the display processor. The information stored by the IDS device is currently limited to 2000 words. If the 340 display is being run by the IDS device when a .IOT is excuted on it, the display will stop and the stored information will be over-written. If the 340 display is not being run, output from simulation is appended to that already present. Whether the 340 display is started at the end of the simulation initiated by a .IOT is determined by a display processor parameter that may be set during simulation.

Instructions for the IDS simulated display processor are thirty-six bit words interpreted as five left justified ASCII characters if bit 1.1 is zero. If bit 1.1 is a one, bits 1.4-1.6 are interpreted as the following commands:

1.4-1.6	Command	Other fields
0	Illegal.	
1	Point.	4.9-3.4 Y, 3.3-1.7 X, 1.2 intensify.
2	Rel vector.	4.9-3.4 Y, 3.3-1.7 X, 1.2 intensify.
3	Increment.	5 left justified 6 bit fields each: 1.1-1.2 length, 1.3 intensify, 1.4-1.6 direction (clockwise from vertical).

4	See below.	1.7-1.9 subcommand function.
5	Abs vector.	4.9-3.4 Y, 3.3-1.7 X, 1.2 intensify.
6	Illegal.	
7	Illegal.	

1.7-1.9	Function	Other fields
0	Illegal.	
1	PUSHJ.	4.9-3.1 address.
2	POPJ.	4.9-3.1 address.
3	JRST.	4.9-3.1 address.
4	POP.	4.9-3.1 address.
5	PUSH.	4.9-3.1 address.
6	See below.	2.3-2.9 parameter to set from 4.9-3.1.
7	Illegal.	

2.3-2.9	Parameter
0	Illegal.
1	Start mode (nonzero means start after .IOT).
2	Character scale.
3	Increment scale.
4	Vector scale.
5	All scales.
6	Coordinate of left edge of displayed area.
7	Coordinate of bottom edge of displyed area.
10-77	Illegal.

3.4.1.1 The .DSTART and .DSTRTL UUO's

```
CALL:       .DSTART DPNTR
            ;error return
            ;normal return

DPNTR:      -LENGTH,,DISLIST-1

DISLIST:    BLOCK LENGTH
```

```
CALL:       .DSTRTL PNTRL
            ;error return
            ;normal return

PNTRL:      DPNT2+1,,.+1
            DPNT2+1,,0

DPNT2:      -LENGTH,,DISLIST

DISLIST:    BLOCK LENGTH
```

These UUO's enable the user to display lists that are in his core image. The execution of either attempts to seize the 340 display and skips only if successful.

A .DSTART should point at a location that will always contain either a BLKO [Ref 1] type output pointer, if one block of data is to be displayed, or the first word of a linked list of display blocks. In this linked list, the right half of each entry points to the next entry, or, if zero, indicates the end of the list. The left half of each word should be positive and is a pointer to a BLKO pointer or is ignored if it is zero or points to a zero word.

.DSTRTL differs from .DSTART only in that, if it is used to display a list of blocks, the left halves of its linked list words are interpreted as pointing to the word after not a BLKO pointer but a regular block mode output pointer [Sec 2.4].

3.4.1.2 The .LTPEN UUO

```
            MOVE AC,[CONTROL]
            MOVEM AC,LBLOCK
CALL:       .LTPEN LBLOCK
            ;return

LBLOCK:     BLOCK 6
```

The 340 display is equiped with a light pen that causes a hardware interrupt when the light intensity it sees increases suddenly as when a spot is displayed under it. The display also stops when this occurs and its current coordinates can be read in by the PDP-6. The user may enable software interrupts (class three) to his procedures when the light pen is seen and he has the display seized [Sec 4.2].

This system call enables the user with the display seized to read information about where the light pen has been seen and the state of the current transfer to the display. Results are returned in the contents of the effective address and following five locations. The contents of the effective address also acts as an argument. If bit 4.9 is zero the rest is ignored. But if bit 4.9 is a one then bit 3.1 will cause the .LTPEN to hang until the light pen has been seen at least once and bit 1.1 will be used to set the multiple sighting mode. If this mode is on, the light pen may be seen many times while a display block is displayed once. If this

mode is off (zero) the light pen may be seen only once and ignored thereafter for each display or a display block.

The six words returned to the user are as follows: (1) the actual word DATAI'ed [Ref 1] from the 340 display at the time of the last light pen interrupt, (2) the number of times the light pen was seen since the last .LTPEN, (3) the sum of the y coordinates at the points the light pen was seen since the last .LPEN, (4) the corresponding sum of x coordinates, (5) the current linked list pointer, and (6) the current BLKO pointer.

3.4.1.3 The .DCLOSE UUO

```
CALL:       .DCLOSE
            ;return
```

This system call releases the 340 display if executed by the user who has the display seized, closing any channels on which it is open. Otherwise it does nothing.

3.4.1.4 The .DSTOP UUO

```
CALL:       .DSTOP
            ;return
```

This system call stops the 340 display, but does not release it,

if executed by the user who has the display seized. Otherwise it does nothing. It is intended for use in conjunction with the .DSTART or .DSTRTL UUO.

3.4.1.5 The .NDIS UUO

```
        MOVEI AC,69.
CALL:   .NDIS AC,
        ;error return
        ;normal return
```

This system call is intended for use in taking pictures of 340 displays. It sets a display control variable from the contents of the accumulator it specifies. This variable has no effect if it is negative but if zero it inhibits display completely and if positive it is decremented by one each time the current display block or list of blocks is displayed. This call skips unless executed by a procedure that does not have the display assigned to it.

3.4.2 The USR Device

The USR device actually represents procedures. A successful .OPEN [Sec 2.2] executed on device USR will either create a new procedure [Sec 5.1.1], attach a disowned procedure tree [Sec 5.1.2], associate with the input-output channel on which it was executed an

existant procedure or allow the user access to the PDP-10's memory [Sec 3.4.2.1]. The file names are the UNAME and JNAME of the procedure being .OPEN'ed. Only immediate inferiors may be .OPEN'ed to write into but any procedure may be examined. If an inferior procedure has been opened, it may be destroyed by a .UCLOSE [Sec 5.2.1] but will be entirely unaffected by a .CLOSE [Sec 2.7.1] executed on the channel. The same procedure may be open on more than one channel of its superior, possibly in different modes. It is at first .OPEN time that an interrupt bit [Sec 5.1.3] is assigned by ITS to a procedure. The number of inferior procedures a procedure may have is currently limited to eight.

The 3.3 mode bit [Sec 2.2] is ignored in .OPEN's on device USR but both block and unit modes are available. Executing .IOT's on channels opened on device USR results in the transfer of a thirty-six bit binary word or words between the procedures. The location in the .OPEN'ed procedure of the first word to be transfered is specified by the .ACCESS [Sec 2.7.5] pointer associated with the input-output channel on which the transfer is occuring. Each word transfered advances the appropriate pointer by one even in unit mode which ITS treats internally as blocks of length one. An attempt to read a word beyond the core allocated [Sec 4.3] will result in a class two interrupt [Sec 4.2] but in a similar attempt to write (possible only for an immediate inferior) ITS will attempt to extend the procedure's core image and an interrupt will result only if more core is

unobtainable.

Executing a .RESET [Sec 2.7.2] on a channel with an inferior procedure open on it is equivalent to executing a .UCLOSE and then a .OPEN to recreate the procedure (with reset variables, one 2000 word block of cleared core, etc.) but with less overhead.

3.4.2.1 The PDP-10

An .OPEN on device USR with a second file name of "PDP10" may be made, in all the modes allowed for regular procedures, to access the memory of the PDP-10. .IOTs attempting to read or write the PDP-10's accumulators will be ineffective and read or write the core locations shadowed by the accumulators. Attempting to reference above the PDP-10's memory will result in illegal user address interrupts. Only the procedures of one procedure tree may have the PDP-10 open at a time, possibly on more than one channel at a time in different modes.

Attempts to set or read variables for the PDP-10 "inferior" are ignored except that 40000 will be read as its memory bound. A .RESET executed on a channel on which the PDP-10 is open will clear its memory.

3.4.3 The CLA, CLI, CLO, and CLU Devices

These symbolic devices enable arbitrary pairs of procedures to talk to each other as buffered input-output devices. Each file on any of these "core link" devices represents a 200 word area, most of which is buffer, which may be simultaneously open for reading, writing, both or neither. The files are identified by two regular file names and a system name [Sec 2.2, 2.2.3]. Only one procedure at a time may have a core link open to write or read on one channel. All standard modes are available.

The CLO (Core Link Open) device may be used to open any core link file and (whether reading or writing) will create one if none exists with the name used. The CLU (Core Link Use) device is identical except that and .OPEN on it will fail if the referenced file does not already exist.

Core link "files" have some peculiar properties. When writing a core link, a procedure will hang if the buffer is full and will write an end-of-file mark outside of the data stream when it closes the channel it has the file open on. However, opening and writing again the same file name essentially pushes a new block of data into the pipeline to be removed by a reading procedure. A procedure reading a core link can not read past an end-of-file and must close and reopen the file if it suspects that there is more to follow. If a reading procedure closes a core link before encountering an end-of-file,

remaining information until the end-of-file is ignored. Core link "files" cease to exist only if empty and open for neither reading nor writing or not empty and not open for from two to four minutes. In the later case they are expunged by the system job [Sec 6.5] to unclutter the system.

The CLI (Core Link Interrupt) device may only be opened to write. The two file names specified should be the UNAME and JNAME [Sec 5.1] of a procedure in the system with the CLI interrupt [Sec 4.2] enabled. If there is no existing core link with the file names and system name used, the .OPEN will succeed in creating a link and interrupting the specified procedure. In addition ITS inserts in the created link two words of information as the start of the first file. These are the UNAME and JNAME of the .OPEN'ing procedure.

The CLA (Core Link Answer) device is to be used in response to a CLI interrupt. It may only be opened for reading and scans through all existing links for one with file names the same as the UNAME and JNAME [Sec 5.1] of the procedure doing the CLA open. If one is found not already open for reading, the .OPEN succeeds.

3.4.4 The ERR Device

This device, by a mechanism similar to that used for the "directory device" [Sec 2.2.1], allows the user to input a character string explaining whatever error indication there may be in a .STATUS

word [Sec 2.5].

The first file name specified on an .OPEN on device ERR must numerically be either 1, 2, or 3. If it is 1, the status of the last input-output channel on which there was an error will be analyzed. If it is a 2, the bottom four bits of the second file name specify which input-output channel to examine. If a 3, the second file name is treated as the .STATUS word to analyze.

3.4.5 The TPL Device

This device allows a user to output a file to be line printed [Sec 3.2.2] at some later time when the line printer is free and such print out has not been inhibited [Sec 6.5.1]. In a manner similar to the COM and SYS [Sec 3.1.3], this device refers to files on the disk with system name ".LPTR.". However, renames are ignored on the TPL device and, while reads work normally, on writes the supplied file names are ignored. Instead, the UNAME of the procedure writing is used as the first file name and a counter of how many files have been written on the TPL device in the current system run is used as the second file name.

Files written on the TPL device are printed out by the system job [Sec 6.5] when the line printer is free. A significant length of time (up to two minutes) is left between non-TPL line printer use and TPL print out and between successive TPL output to allow local users

to seize and use the line printer directly.

3.4.6 The NUL Device

This device has all standard modes and is a high speed infinite sink on output and source of zeros on input.

4. The Procedure

See also section 0.2.3.

4.1 Philosophy and Organization

The use of ITS is entirely procedure oriented. Almost all system actions are caused by trapping instructions [Sec 1.2; App A, E] executed by user programs that are hierarchially organized [Sec 5.1]. A large number of variables [App D; Sec 4.5] is associated with each procedure by the ITS system. These variables are stored in the system and do not impinge on the user's core image.

These procedures are run for short fixed lengths of time or until they become unrunnable (whichever comes sooner) in an order determined by the scheduler [Sec 4.4]. The scheduler also handles software interrupts to the user [Sec 4.2]. When in execution, user programs run with the PDP-6 in user mode [Sec 1.2.2] which enables protection and relocation hardware that ITS sets in acordance with the length [Sec 4.3] and location in absolute memory of the particular procedure's core image.

The following are among the advantages of this storage of user variables:

(1) No effort by a user's procedure to randomize its core image can effect system variables related to the procedure and thus erase evidence of system malfunction or lack of malfunction.

(2) A procedure's core image may be easily swapped out without significant reduction in information concerning it of interest to the system or the necessity to retain some portion of it.

4.2 User Interrupts

ITS provides software implemented interrupts to the user program similar to the hardware interrupts it receives. Only one level of interrupt is simulated but this makes little difference due to the ease with which the user can inhibit or enable various types of interrupt [Sec 4.2.1] and may also re-enable interrupts without immediately or ever returning to the location his program was interrupted from.

Each procedure has in system memory two interrupt enable mask words, two interrupt request words, and a flag indicating interrupt level status [App D]. Various conditions that can cause interrupts are assigned bits in these interrupt request words. These conditions are divided into three classes according to severity as follows:

(1) Class one conditions, the most severe, can not be enabled. They always cause the "interrupted" program to be stopped and its superior procedure to receive an interrupt.

(2) Class two conditions are of moderate severity. They may be enabled to interrupt the user but if they occur while not enabled or while the user's interrupt level flag indicates he is currently

processing an interrupt, he will be stopped and his superior procedure interrupted.

(3) Class three conditions are the least severe and may also be enabled to interrupt the user. While not enabled they have no effect. If enabled and they occur while the user's interrupt level flag indicates he is free to receive interrupts he will be interrupted but if the user's flag indicates he is not free they will be stored in the interrupt request words to be presented to the user as soon as he allows unless reset with an .SUSET [Sec 4.5].

In cases mentioned above (for class one and two) when a procedure's superior is interrupted, the particular bit used to interrupt the immediately superior procedure is in its second interrupt request word. A bit is assigned by ITS to that inferior at the time it was created [Sec 5.1.3] and can be read with a .USET [Sec 5.2.3].

When conditions that would cause a procedure's superior to be interrupted occur in a top level procedure one of two things may happen. If the procedure tops a console controlled tree, a new HACTRM will be loaded (as in section 1.1) and all inferior procedures will be lost. If the procedure tops a disowned tree [Sec 5.2.2] it will be immediately stopped and when next attached [Sec 5.1.2] by a job tree the attaching procedure will be appropriately interrupted.

ITS determines that it should actually interrupt a procedure when it schedules [Sec 4.4], or the procedure executes a .DISMISS [Sec 4.2.2], or .SUSET [Sec 4.5] of the PICLR variable if there are corresponding bits on in one or both of the procedure's interrupt

request and mask words and its interrupt level flag (PICLR) indicates interrupts are enabled. ITS examines the contents of the procedure's location 42 to see if its direct address (indexing and indirect addressing ignored) is between location 20 and the location 6 less than the top of the user's core image, inclusive. If not, the user receives a class one interrupt for a bad location 42. If so, the user's appropriate interrupt word (first word has higher priority than the second), containing bits on for interrupts the user is being presented with, is placed in the word directly addressed by the user's location 42 and bit 4.9 of that location is turned on if second word requests were stored. The interrupt request word stored is then cleared to receive future interrupts. The user's interrupt level flag is also set to inhibit further interrupts. Finally a JSR [Ref 1] is simulated to a location one greater than that in which the interrupts were just stored. The normal manner to return from an interrupt is with a .DISMISS [Sec 4.2.2].

The following is a list of word one interrupt bits:

Bit(s)	Class	Condition
1.1	3	Character typed at console [Sec 3.2.1].
1.2	1	^Z typed [Sec 1.1.2].
1.3	1	Bad location 42.
1.4	3	AR overflow [Ref 1].
1.5	2	Display list memory protection violation [Sec 3.4.1].
1.6	2	Illegal instruction.
1.7	3	System going down in 5 minutes [Sec 6.6.2].
1.8	1	.VALUE executed [Sec 5.3.1].
1.9	2	Input-output channel error. This interrupt

always occurs at an .IOT [Sec 2.3] or .OPER [Sec 1.2.1] that refers, with its ac field, to the channel on which the error occured. For more exact cause of interrupt use a .STATUS [Sec 2.6] or the ERR device [Sec 3.4.4].

2.1	2	Memory protection violation on reference to inferior procedure [Sec 3.4.2].
2.2	1	.BREAK executed [Sec 5.3.2].
2.3	1	Return from single instruction proceed [Sec 6.7].
2.4	3	Slow clock (every half second).
2.5	2	Memory protection violation [Ref 1].
2.6	2	MAR interrupt [Sec 6.4].
2.7	3	Light pen [Sec 3.4.1.2].
2.8	3	PDL overflow [Ref 1].
2.9	3	CLI [Sec 3.4.3] device interrupt.
3.1-4.8	-	Not used.
4.9	-	Indicates rest of bits to be interpreted as below.

In the second word of interrupt requests, bits 3.1-3.8 represent inferior procedures [Sec 5.1.3] and bits 1.1-2.7 represents input-output channels [Sec 2.1] with 1.1 for channel zero. Interrupts occur from inferior procedures as described above for class one or two interrupts to the inferiors while an input-output channel will try to interrupt if there is buffered teletype [Sec 3.2.1] input on the channel. These buffered characters may be examined at the interrupt level with .ITYIC [Sec 2.7.3].

4.2.1 The .SETM2 UUO

A procedure can examine and modify all of the five variable words associated with interrupts to it by means of the .SUSET UUO

[Sec 4.5]. When a procedure is first created [Sec 5.1.1] or has a .RESET executed on it [Sec 3.4.2] all of these five variables are zero except PICLR which is set to minus one. For the mask (MSKST and MSKST2) and request (PIRQC and IPIR) words a one bit represents an enabled interrupt or pending request and a zero represents an inhibited interrupt or lack of request. ITS will ignore attempts by the user to enable a class one interrupt (first word mask bits 1.2, 1.3, 1.8, 2.2, and 2.3). To avoid certain timing errors, special .SUSET means (APIRQ and IPIRQ) are provided to turn on or off bits in the first request word without affecting other bits in that word.

The procedure's interrupt level flag (PICLR) can also be read or set. Its zero state indicates that interrupts are inhibited and class two interrupts become fatal. If nonzero, interrupts are enabled. ITS sets this variable to minus one when it executes a .DISMISS from a user, but when a user attempts to set it the word he stores there with .SUSET will be shifted right thirty five places. Thus the user can set his PICLR to either zero or one depending on the sign bit of the word he stores. As a result he can enable interrupts (even in the middle of his "interrupt routine") in a manner distinguishable from the automatic enablement performed by ITS or inhibit interrupts.

The above discussion also applies to examining and modifying these variables in inferiors with .USET [Sec 5.2.5].

To avoid certain timing problems there is a special JOB by which

a procedure may set both its mask words at once as follows:

```
        MOVE AC,[MASKW1]   ;set up first word of mask bits
        MOVE AC+1,[MASKW2]  ;set up second word of mask bits
CALL:   .SETM2 AC,         ;set both mask vars from AC & AC+1
        ;return
```

4.2.2 The .DISMISS UUO

```
        LOC 42
        JSR TSINT

TSINT:  0                  ;interrupt bits stored here
        0                  ;loc interrupted from stored here
        ...                ;interrupt routine
        ...
        ...
CALL:   .DISMISS TSINT+1   ;returns to where interrupted from
```

This system call provides the normal means to return to an interrupted routine. It simultaneously re-enables interrupts and transfers to the location addressed by the right half of the contents of its effective address, restoring flags from the left half. This is normally, but not necessarily, the pseudo-JSR word stored by ITS as it simulates an interrupt. If there are pending conditions in the user's interrupt request words he will be immediately re-interrupted.

4.3 The Core Allocator

The core allocator consists of a single special procedure, called the core job, that does most of the work and various routines that signal to it their interest in acquiring or releasing core.

An internal map of the status of every 2000 word block of memory exists in the system and is maintained by the core allocator. The majority of the complexity of the core allocator is not due to maintaining this table but to the actions it must sometimes take to relocate input-output buffers and other occupants of core when a procedure wishes to expand. The lack of paging on the PDP-6 makes it sometimes necessary to "shuffle" core to make a sufficient contiguous block.

Each 2000 word block must, except for two transient states, be devoted to one of the following uses: (1) part of a procedure's core image (ITS code and variables appear as core allocated for the system job [Sec 6.5]), (2) a disk buffer or directory [Sec 3.1.1], (3) a display buffer [Sec 3.4.1], (4) unused, or (5) further subdivided into 200 word blocks. These 200 word blocks are used for core link device buffers [Sec 3.4.3] or for DEC tape buffers or directories [Sec 3.1.2].

A procedure may request more or less core by means of a .CORE [Sec 4.3.1] and determine its memory bound without making protection

violations with a .SUSET [Sec 4.5]. Other than the .CORE UUO a procedure's memory bound may be affected only by its destruction, when all blocks in its core image are marked free, or a .USET [Sec 5.2.3] or .IOT on device USR [Sec 3.4.2] executed by its immediate superior.

4.3.1 The .CORE UUO

```
CALL:       .CORE NUMBLK
            ;error return
            ;normal return
```

This system call requests for the procedure executing it as many 2000 word blocks as the value of its effective address except that if this value is zero it will be treated as if it were one. If and only if the .CORE is successful it skips. Only a .CORE to acquire more core may fail. If it does and a more than one block increase was being requested, some smaller number than requested may have been obtained.

4.4 The Scheduler

The scheduler runs when a quantum time-out occurs or when the physically running procedure encounters a blocking condition or causes an interrupt by some program action (memory protection

violation, PDL overflow, etc.). The scheduler decides which procedure should run next by an examination of the variables [App D] in the system for each procedure. It simultaneously performs those modifications of these variables necessary to provide the simulated interrupt feature [Sec 4.2].

For those procedures that have their stop word (USTP [App D]) zero, their runnability is determined by a variable known as the flush instruction (FLSINS) which is set when the procedure becomes blocked. If this variable is zero the procedure is runable. If nonzero it is executed with the procedure's EPDL2 variable [App D] loaded into ITS symbolic accumulator T. If the instruction skips then the procedure is runnable and has just become unblocked. If the flush instruction does not skip the procedure is still blocked.

The next procedure to be run is picked from among the runnable procedures by a scan which considers two "resource" variables per procedure. Both of these are quantized approximations to exponential decays toward proportions of machine time. The JTMU variable is associated with a particular procedure while the UTMPTR variable pointer for that procedure points to a quantity similar to JTMU maintained for all procedures in one console controlled tree. During the scan to select the next procedure to run, a procedure will be prefered to the best so far if its console tree has used less time recently than that of the best so far or its console tree has used exactly the same amount and the procedure has used less time

recently. Exceptions to this are that a procedure is always preferred to the best so far if it has used less than one eighth as much time recently and never preferred if it has used more than eight times as much. Also the procedure, if any, that has seized the .MASTER [Sec 6.8.5] facility is given double priority and disowned procedures are given one eighth priority unless they have the line printer (device LPT) assigned to them or are running in executive mode when scheduled.

All disowned [Sec 5.1] job's UTMPTR's point to a single word which is given less resource than that for a normal console tree. The system job [Sec 6.5] and core job [Sec 4.3] have UTMPTR's to seperate words given the same priority as a normal tree.

The general properties of this scheduling system can be described as follows:

(1) Machine time is normally equally divided between consoles and then between all the procedures running for each console.

(2) Machine time used by a procedure is integrated with decay (time constant about seven seconds) so
(a) a procedure that is frequently blocked for a short period of time (compared with a quantum, currently one fifteenth of a second) should receive equal service with a pure compute job and
(b) an interactive program called on infrequently by a user is essentially guaranteed good service as it starts to think of a reply but can not, on average, use more time than a pure compute job.

4.5 The .SUSET UUO

```
CALL:       .SUSET [DIRECTION+VARIABLE,,DATLOC]
            ;return

DATLOC:     ;location read into or written form
```

Procedures may examine and modify some of their system variables [App D] by means of this system call. The contents of its effective address (a constant in the illustration above) is viewed as two half words. The right half should point at the word from which the variable is to be set or into which the variable is to be read. The left half specifies by its top bit (4.9) whether the variable is to be read (zero) or written (one) and by the value of its remaining bits which system variable. The following table lists permissible values:

Name	Value	Type	Variable
UPC	0	07	Program counter word [Ref 1].
VALUE	1	17	Contents of effective address of most recent .VALUE [Sec 5.3.1].
TTY	2	05	Status relative to console [Sec 3.2.1].
FLSINS	3	01	Blocking condition [Sec 4.4].
UNAME	4	05	UNAME [Sec 3.4.2].
JNAME	5	07	JNAME [Sec 3.4.2].
MASK	6	17	Interrupt mask word [Sec 4.2].
USTP	7	03	Stop word (only bit 4.7 can be written) [Sec 4.4].
PIRQC	10	17	First interrupt request word [Sec 4.2].
INTB	11	01	Bit used to interrupt superior [Sec 5.1.3].
MEMT	12	07	Memory bound [Sec 4.3], first location it would be illegal to reference.

SV40	13	05	Last executed UUO trapping to system [Sec 1.2.1].
IPIRQ	14	17	Interrupt request word (written by an IORM).
APIRQ	15	17	Interrupt request word (written by an ANDCAM).
SNAME	16	17	System name [Sec 2.2.3].
PICLR	17	17	Interrupt level status word [Sec 4.2].
MARA	20	17	MAR register and control bits [Sec 6.4].
MARPC	21	17	Program counter at last MAR interrupt [Sec 6.4].
UUOH	22	05	Program counter word at last system UUO.
UIND	23	05	User index.
RUNT	24	05	Total run time in 4.069 microsecond units.
MSK2	25	17	Interrupt mask word two [Sec 4.2].
IFPIR	26	17	Interrupt request word two [Sec 4.2].
APRC	27	05	Disowned and processor cono status.
SV60	30	05	Saved absolute location 60.
	31-77	00	Unused.
IOC	100+N	01	Input-output channel word for channel N.
IOS	120+N	01	Status word for input-output channel N.
IOP	140+N	01	Nth word of input-output channel push down list [Sec 2.6], two words per entry.

In the above table the 1.3 bit on in the type column implies that the variable can be read with .SUSET and the 1.4 bit implies it can be written with .SUSET. The other type bits relate to the .USET UUO [Sec 5.2.3].

Pre-defined symbols are available in the MIDAS assembler consisting of a "." followed by an "R" for read or a "S" for set followed by the string in the name column of the above table for all its entries.

5. Procedure Interaction

See also section 6.2.4 and 8.4.2.

5.1 The Hierarchial Organization of Procedures

All procedures operating under ITS are members of a tree structure recorded in the system by associating with each a pointer to its immediate superior. If this pointer is minus one it indicates that the procedure tops a tree. These trees may be of two types, those under the control of a console and those that are disowned [Sec 5.2.2].

Console "controlled" trees have had their top level procedure automatically loaded by ITS [Sec 1.1]. Since procedures may control their inferiors and the most that the user can do in appealing for aid when confronted with recalcitrant programs is to cause his console to converse with higher level procedures [Sec 1.1.2], he is actually at the mercy of this top level program. It is currently a DDT expanded by the addition of new commands related to time sharing [Sec 7]. About the only restrictions deliberately inserted in it relate to logging in [Sec 6.1].

Disowned procedure trees differ from console controlled trees only in the following ways: (1) they have no console associated with them; (2) class one interrupts in a top level procedure are handled

differently [Sec 4.2]; (3) they are scheduled with lower priority [Sec 4.4]; (4) certain system calls, when executed in them, are treated as illegal instructions; (5) certain input-output devices are not accessible by them.

There exist a variety of means for procedures immediately above and below each other to interact [Sec 5.2, 5.3]. Procedures may also affect any procedure beneath them in the procedure tree by means of input-output translation table entries [Sec 2.2.2] and an arbitrary pair of procedures may converse treating each other as input-output devices [Sec 3.4.3].

Among the advantages of the above system of procedure organization are the following:

(1) The highest level command processor (the top level procedure of a console controlled tree) may vary arbitrarily from user to user. In actual operation it is normal to see HACTRN's whose inferior procedure variables and symbol tables much exceed their code.

(2) New versions of the highest level command processor may be introduced simply by writing a file on the disk with no interruption to system continuity.

(3) New versions of the highest level command processor may be easily and safely debugged by loading them as inferior procedures. In this position they can exercise all their normal functions except logging in and out.

(4) It is not neccessary to make every effort to remove all bugs from the highest level command processor for fear that the system may be destroyed by them, a consideration often inhibiting more complex and powerful command languages. A fatal error simply results in the program being reloaded.

(5) Users have great generality in creating command levels of their

own.

(6) Multi-task programs may be organized in a consistent manner.

Items 1, 2, and 4 above argue to various extents against pure procedures as does their inherent speed inefficiency. Having to maintain separate copies of the highest level command processor for each user is not that significant a disadvantage if one considers the amount of each copy's core image unique to each user and that while not in actual use they could be swapped out of the system.

5.1.1 Procedure Creation

Procedures may be created only by typing ^Z on an idle teletype [Sec 1.1.1] or by executing a .OPEN [Sec 2.2] on device USR [Sec 3.4.2]. In the second case, which is discussed here, there are strict requirements on the "file name" used in the .OPEN. The first file name, which will be the UNAME of the created procedure, must be the same as the UNAME of the procedure executing the .OPEN. The second file name, which will be the JNAME of the created procedure, must be different from the JNAME of all jobs then in existence that have the UNAME being used. A procedure may have a maximum of eight directly inferior procedures. A newly created procedure will have one 2000 word block of cleared memory and all of its system variables initialized.

5.1.2 Attaching Disowned Procedure Trees

The only case in which a successful .OPEN [Sec 2.2] on device USR [Sec 3.4.2] may be executed to provide an inferior procedure (as opposed to opening a procedure for examination only) with a first file name different from the UNAME of the procedure executing the .OPEN is that in which the top procedure of a disowned tree is being opened. This type of .OPEN will be treated as an illegal instruction if executed by a procedure in a disowned tree. If not and the opening procedure has less than eight inferiors the open will succeed and ITS will modify the UNAME of the opened procedure and its inferior tree to agree with the opening procedure. It will then take each UNAME JNAME pair in the attached tree and make it unique by incrementing the JNAME part if necessary.

5.1.3 The .UTRAN UUO

```
CALL:       .UTRAN FBLOCK
            ;error return   ;no inferior found for int bit
            ;normal return  ;inferior found

FBLOCK:     INTBIT,,
            0               ;UNAME stored here by call
            0               ;JNAME stored here by call
```

When a procedure is first created by a .OPEN, ITS assigns to it

a bit for use in interrupts [Sec 4.2] to its superior procedure different from that for any other inferior procedure its superior may have. Since inferior procedures can only be directly controlled when open on input-output channels, the loss of the name of an inferior by its superior while it was not open would result in complete loss of communication. Although procedures that have inferiors normally keep close track of their names and interrupt bits as read by .USET's [Sec 5.2.3], this system call has been provided to translate from particular interrupt bit to name of inferior. The left half of the contents of the effective address must have only the appropriate interrupt bit on. If a corresponding inferior procedure is found, the .UTRAN skips and the UNAME and JNAME of the inferior are stored in the two words after the word containing the interrupt bit. If no corresponding inferior is found the .UTRAN will fail to skip and the two locations after the location it effectively addresses will be unmodified.

5.2 Controls over an Immediately Inferior Procedure

See also section 3.2.1.1.

5.2.1 The .UCLOSE UUO

```
CALL:       .UCLOSE CHNUM,
            ;return
```

Procedures may be destroyed only by the execution of a .LOGOUT [Sec 6.1] or a .UCLOSE. The second of these two system calls, which is discussed here, destroys the inferior procedure open on the channel specified by its accumulator field (CHNUM in the above illustration) and the entire procedure subtree of which this inferior is the top procedure. If an inferior is not open on the channel, the .UCLOSE'ing procedure will receive an input-output channel error interrupt. .CLOSE's [Sec 2.7.1] are automatically effected on all channels of all the destroyed procedures and any system resources they claim are surrendered (by possibly simulating a .DCLOSE, .HANGUP, and/or .MASTER). In order to kill certain cancerous types of procedure subtrees it has been found necessary to implement this system call such that it first stops all procedures in the subtree and then destroys them.

5.2.2 The .DISOWN UUO

```
CALL:       .DISOWN CHNUM,
            ;return
```

This system call requires that an inferior procedure be open on the channel specified by its accumulator field (CHNUM in the above illustration) or the .DISOWN'ing procedure will receive an input-output channel error interrupt. It causes this open procedure to become the top procedure of a disowned tree [Sec 5.1] and in the process closes all channels on which the disowning procedure has it open. A procedure in a disowned tree that tries to execute this system call will be unsuccessful and receive an illegal instruction interrupt [Sec 4.2] as will a procedure trying to disown a subtree that controls the tree's console [Sec 3.2.1.1].

5.2.3 The .USET UUO

```
CALL:       .USET,[DIRECTION+VARIABLE,,DATLOC]
            ;return
DATLOC:     ;location read into or set from
```

This system call was created to allow a procedure to set and read many of the system variables [App D] associated with its inferiors and to read variables associated with other procedures. When executed its accumulator field must contain the number of a channel (CHNUM in the above illustration) on which a procedure is

open or the .USET'ing procedure will receive an input-output channel error interrupt. Otherwise its is very similar to .SUSET [Sec 4.5]. Reading is allowed if the 1.1 bit is on in the type column of the table in section 4.5. Setting is allowed for immediate inferiors if the 1.2 bit is on in that table.

5.3 Communication to an Immediately Superior Procedure

5.3.1 The .VALUE UUO

```
CALL:       .VALUE ARGLOC
            ;return if restarted
ARGLOC:     ARG             ;trasmitted to superior procedure
```

A procedure executing this system call receives a class one interrupt [Sec 4.2] stopping it and interrupting its superior procedure. The contents of the effective address is saved in a system variable associated with the procedure [App D] so its superior can easily retreive it with a .USET [Sec 5.2.3].

5.3.2 The .BREAK UUO

```
CALL:       .BREAK ARG1,ARG2
            ;return if restarted
```

The accumulator field and effective address of this UUO are ignored by ITS. A procedure executing it receives a class one interrupt [Sec 4.2] stopping it and interrupting its superior procedure.

5.4 The .GUN UUO

```
            MOVEI AC,USRINDEX
CALL:       .GUN AC,
            ;failure return
            ;normal return
```

This system call, intended for use as a last resort, allows any procedure in a console controlled tree to expunge another tree from the system. The procedure specifies the tree to be .GUN'ed down by the user index (see section 5.2, mode N) of its top procedure. This is used instead of the normally unique UNAME JNAME pair so that easy reference may be had to procedures not logged in [Sec 6.1]. The contents of the accumulator specified by a .GUN must be the user index (can be read with .USET) of a top level [Sec 5.1] procedure other than the system job [Sec 5.5] or core job [Sec 4.3]. If the procedure executing the .GUN is in a disowned tree it will receive an

illegal instruction interrupt.

The .GUN skips if successful at which point the system job will have been informed of the tree it is desired to expunge, which action the system job will, in due course, perform, also typing out an appropriate message. If the above restrictions on the specified user index are not met the .GUN will return without skipping.

6. Miscellaneous System Calls and Features

See also section 0.2.5.

6.1 The .LOGIN and .LOGOUT UUO's

These two system calls are the only ones that are effective only at the top level.

```
CALL:       .LOGOUT
            ;return if not top level
```

.LOGOUT has no effect and simply returns from the system to the instruction following if not executed in a top level procedure. It causes the procedure tree in whose top procedure it is executed in to be expunged from the system. It works for both console controlled trees and disowned trees. Since the normal means to create [Sec 5.1.1] and destroy [Sec 5.2.1] a procedure under ITS are calls in a superior procedure, the special case of the top level procedure must be handled differently as it is by ^Z [Sec 1.1.1] and .LOGOUT.

```
            MOVE AC,[SIXBIT /NAME/]
CALL:       .LOGIN AC,
            ;error return
            ;normal return
```

.LOGIN is not restricted to top level procedures by ITS but rather by HACTRN, the top level procedure ITS automatically loads [Sec 7]. A .LOGIN represents an attempt by a procedure to set its UNAME to the contents of the accumulator specified. This is allowed by ITS only if the procedure's current UNAME is minus one (as the UNAME of initial HACTRN's is set) and the desired UNAME is neither zero, minus one, nor the UNAME of an extant console controlled tree. The success of a .LOGIN is signalled by its skipping. HACTRN will not open inferior procedures for a user until he has successfully logged in at which point his inferior procedures will have UNAME's dooming future .LOGIN's in that tree to failure.

6.2 Nonstandard Devices

6.2.1 The .RDSW UUO

```
CALL:       .RDSW AC,
            ;return
```

This system call reads (DATAI's [Ref 1]) the contents of the data switches on the PDP-6's console into the specified accumulator.

6.2.2 The .RD760 and .WR760 UUO's

```
CALL:       .RD760 AC,
            ;return

            MOVE AC,[BITS]
CALL:       .WR760 AC,
            ;return
```

These system calls respectively read the data register (DATAI) of the device whose PDP-6 device number is 760 into the specified accumulator and output (DATAO) from the specified accumulator to the data register of the device [Ref 1]. The bits in the data register of this device are usually used for temporary direct control or sense hookups. Currently the bottom six bits control lights on the hand of the AMF arm [Sec 3.3.2].

6.2.3 The .RD710 UUO

```
CALL:       .RD710 AC,
            ;return
```

The real time clock, which ITS currently uses to time user quanta, has, as well as an interval timing counter, a clock counter which is neither set or stopped by ITS. This system call reads into

the specified accumulator a word whose bottom twenty four bits are this clock counter. It is incremented once every 4.069 microseconds.

6.2.4 The .RD500 UUO

```
CALL:       .RD500 AC,
            ;return
```

This call reads (DATAI's) into the specified accumulator from device 500. This device is purported to be or become yet another kind of clock. See Paul DeCoriolis for more information.

6.3 Various Times and the Date

6.3.1 The .RDTIME UUO

```
CALL:       .RDTIME AC,
            ;return
```

This sytem call reads into the specified accumulator the internal system time which is a word set to zero when a system is first loaded and is incremented every thirtieth of a second thereafter. This quantity is useful in conjunction with .SLEEP [Sec 6.8.1].

6.3.2 The .RTIME UUO

CALL: .RTIME AC,
;return

After the system has determined real (24 hour system) time, it maintains this internally as a number in units of one half second since the last midnight. This system call reads into the specified accumulator in sixbit characters this time in pairs of digits, reading from the left, hours, minutes, and seconds. If the time has not yet been determined, minus one will be read.

6.3.3 The .RDATE UUO

CALL: .RDATE AC,
;return

After the system has determined the calendar date, it maintains it internally as a word with six sixbit characters in it. These characters are all numeric. The first two are the last two digits of the year, the next two are the month number, and the last two digits are the day of the month. This system call reads this word into the specified accumulator. If the date has not yet been determined,

minus one will be read. In the current implementation, the date being determined implies that the time [Sec 6.3.2] has been determined. The date is correctly incremented every midnight.

6.4 The MAR Feature

In order to simulate the address stop facility of the PDP-6 in a time shared mode, it was decided to introduce a new hardware register, the MAR register, to cause interrupts when a particular location was addressed by the computer. To increase the flexibility of this feature, three control bits were added which further define the interrupt condition given that a memory address match has occured.

Among the system variables [App D] kept by ITS for each procedure are its MAR register contents and control bits. These are loaded into the hardware as the user is started. Since the MAR feature is intended primarily as a debugging aid, it causes a class two interrupt [Sec 4.2] and a .USET [Sec 5.2.3] is used to set it from DDT [Sec 7] more frequently than a procedure sets its own with an .SUSET [Sec 4.5].

.USET and .SUSET take the right half of the word set from as the new address to be put in the MAR register and the low three bits of the left half as control bits as follows:

Bit	Meaning (one state),
3.3	Interrupt if data read or written is negative.
3.2	Interrupt if data read or written is positive.
3.1	Interrupt only on a write.

At this time, attempts to differentiate read cycles on the basis of sign will fail as, due to a timing error, the data always appears positive. If a write cycle is interrupted on, the protected location will have been modified when the interrupt is processed.

When an MAR interrupt occurs, further MAR interrupts are disabled until the MAR is again set. A .SUSET or .USET may be used to read the word stored by the JSR at the system's processor interrupt channel location when the last MAR interrupt occured. Although, for their own safety, some parts of executive code inhibit MAR interrupts, it is quite possible for the user to cause some system calls to interrupt to the user, on reference to his core image where his MAR is pointing. The program counter word stored by the JSR at the user's location 42 [Sec 4.2] will be the location in his core image of the offending system call. The word read by a .SUSET or .USET will be an executive program counter word showing the absolute location at which the system call first attempted to reference the user's protected location.

6.5 The System Job

There are various tasks ITS would like to perform, most of which include input-output or waiting indefinite periods of time, which would be inconvenient to implement directly due to the procedure originated orientation of almost all system routines. Because of this a procedure was synthesized, known as the system job, that is actually a part of the monitor and always runs in executive mode. To aid in debugging ITS a feature was added to the system job so that when it had no other tasks to perform it would compare constant parts of ITS against a copy it makes initially [Sec 6.5.1]. The following are some of the other tasks the system job performs:

(1) Does .CORE's [Sec 4.3.1] for ITS to make room for more blocks of user variables or for the system job's copy of constant parts of ITS.

(2) Prints messages on the system console whenever an effective .SETLOC, .GUN, .IOTLSR, .LOGIN, .LOGOUT, or .OPEN for output or .FDELE on device SYS is executed.

(3) Prints messages on the system console when a parity error, unrequested change in teletype priority interrupt channel allocation, memory excessively full condition, or garbage collection of DSK user directory, is detected.

(4) Initially determines the 24 hour time and calendar date [Sec 6.3.2, 6.3.3].

(5) Prints a message on all teletypes when the system is first loaded.

(6) Outputs a "CONSOLE FREE" message on any teletype when it is no longer open by any procedure.

(7) Various tasks related to the sytem going down feature [Sec 3.5.2] and TPL device [Sec 3.4.5].

3.5.1 The .SUPSET UUO

```
        MOVEI AC,2
CALL:   .SUPSET AC,
        ;return
```

The system job is partially controlled by ITS through a word in system memory. Bits to the left of 2.8 in this system word represent requests for some transient action. In the right half the bottom two bits (1.1-1.2) are currently used. Bit 1.1 controls the system checking feature. If this bit is on, the system job makes a copy of the constant parts of ITS and proceeds to spend its spare time checking the areas it copied against the copy. If a discrepancy occurs, a message is printed and the copy corrected. If this bit is off, the system job will destroy any copy it may have, freeing that memory and most of the fraction of the machine time the system job uses when in checking mode.

Bit 1.2 affects the TPL device [Sec 3.4.5]. If it is on, TPL use of the LPT device is inhibited. Furthermore if a TPL file is being printed when this bit is turned on, it will be deleted, output will be aborted, and the LPT device freed.

The .SUPSET system call enables the user to control the system

checking bit by XOR'ing [Ref 1] bits 1.1-2.8 of the specified accumulator into the system job control word and returning the resulting value in the specified accumulator. It also types out a message on the system job console like the message for .SETLOC.

6.5.2 The System Going Down Feature

To provide an orderly means for the termination of the operation of ITS, a special feature has been added. It is activated by setting absolute location 37 negative (either with the PDP-6 console keys [Ref 1] or a .SETLOC [Sec 3.8.4]) or using .SHUTDN [Sec 6.5.2.1]. After a negative location 37 is noticed, all procedures for which it is enabled will receive a system going down interrupt [Sec 4.2] and all free teletypes have an appropriate message typed on them. Then a five minute timer is started and new console trees started [Sec 1.1.1] will be given a system going down interrupt right after they are created.

When the timer times out or when the number of console controlled trees in which successful .LOGIN's [Sec 6.1] have been executed goes to zero, all trees are forceably logged out and a message typed on all teletypes. All DEC tapes [Sec 3.1.2] for which the system is retaining a directory are flapped.

6.5.2.1 The .SHUTDN UUO

```
            MOVE AC,[TIME]
CALL:       .SHUTDN AC,
            ;error return
            ;normal return
```

This system call, which is illegal for procedures in a disowned tree, is a request for the system to go down [Sec 6.5.2] in as many thirtieths of a second as the contents of the specified accumulator. If an amount less than five minutes is specified, it will be treated as though it had been five minutes. If the system is already going down in less time than specified, this call will be ignored and return without skipping.

If the system is going down in a longer time than specified or not going down at all the call will succeed and skip. It will cause a system going down interrupt to all enabled procedures and start a timer for the specified length of time. New console trees created thereafter [Sec 1.1.1] will also receive a system going down interrupt when created. If the time till system death passes through five minutes, all enabled procedures will be interrupted again.

The conditions for and actions on actual death are as in section 6.5.2.

6.5.2.2 The .DIETIM UUO

```
CALL:       .DIETIM AC,
            ;return
```

This system call sets its specified accumulator to minus one if the then running ITS is not going down [Sec 6.5.2]. If it is, the accumulator is set to the maximum time till system death in thirtieths of a second.

6.6 Examining ITS

6.6.1 The .GETSYS UUO

```
            MOVE AC,[-LENGTH,,DATLOC]
            MOVE AC+1,[SIXBIT /AREA/]
CALL:       .GETSYS AC,
            ;error return
            ;normal return

DATLOC:     BLOCK LENGTH    ;area read into
```

This sytem call enables a procedure to receive copies of various internal monitor areas in such a way that the data in those areas must be consistent. The specified accumulator contains a pointer word as for a block mode .IOT [Sec 2.3]. If the left half of the accumulator is zero, the block is treated as running to the end of

the users core image. The location following the specified accumulator contains, in left justified sixbit, the area requested as follows:

Sixbit	Area
TRANS	Translation table [Sec 2.2.2].
GETS	List of things which can be requested with .GETSYS.
DEVS	List of available symbolic input-output devices [App C].
MEMORY	Memory allocation tables [Sec 4.3].
UTAPE	DEC tape routine variables [Sec 3.1.2].
CLINK	Core link variables [Sec 3.4.3].
CALLS	List of symbols in squoze [Ref 3] for system calls [App A] and .SUSET [Sec 4.5] left halves and their values.
DSYMS	Symbol table for ITS.
IMPX	Area input multiplexor cycled into [Sec 3.3.3].
OMPX	Area output multiplexor cycled from [Sec 3.3.2].
USERS	User variables for all procedures [App D].
USER	User variables for particular user specified by UNAME and JNAME in locations two and three greater than the specified accumulator [App D]. If user not found location containing UNAME will be zeroed.

Interrupts are inhibited to a sufficient extent that the data obtained can not be inconsistent due to the areas modification while being transfered to the user. A successful .GETSYS will skip. One that fails, due to non-recognition of its sixbit word or insufficient memory specified by its pointer word, will not skip and in the former case the sixbit word will be zeroed.

6.6.2 The .GETLOC UUO

```
            MOVE AC,[ABSLOC,,DATLOC]
CALL:       .GETLOC AC,
            ;return

DATLOC:     0               ;contents of ABSLOC stored here
```

This system call simply transfers the contents of the absolute location pointed to by the left half of the specfied accumulator to the user's relative location pointed to by the right half. If the right half points beyond the user's memory bound he will receive an illegal instruction interrupt [Sec 4.2].

6.6.3 The .RSYSI UUO

```
CALL:       .RSYSI AC,
            ;return
```

This sytem call reads into the specified accumulator the sixbit representation of the second file name of the symbolic file that was assembled to produce the running version of ITS.

6.6.4 The .EVAL UUO

```
        MOVE AC,[SQUOZE sym]
CALL:   .EVAL AC,
        ;error return
        ;normal return
```

The .EVAL system call allows a user to easily look up symbols in high core DDT's table of system symbols. The "squoze" code [Ref 3] for the symbol should be put in the accumulator specified. If the symbol is not found, the call will return without skipping. If it is found its value will be placed in the specified accumulator and the call will skip on returning.

6.7 The One Proceed Feature

The Project MAC AI Group PDP-6 has been modified so that there exists a mode in which it executes one instruction and then traps. This mode is properly stored when an interrupt occurs and is restored by JRST 2, [Ref 1]. For users it causes a class one interrupt and is expected to be used by superior (such as DDT [Sec 7]) procedures by .USET'ing [Sec 5.2.3] the 3.9 bit of their inferior's program counter word. There is coding in ITS such that all system calls will appear to be single instructions using this mode. While in this mode a user interrupt may be serviced and the user's status, including this mode,

will be restored by a normal .DISMISS [Sec 4.2.2].

6.8 Miscellaneous

6.8.1 The .SLEEP UUO

```
        MOVE AC,[TIME]
CALL:   .SLEEP AC,
        ;return eventually
```

This system call allows a procedure to stop running, while remaining interruptable, by either specifying a length of time to remain quiescent or a particular time to wake up. If the specified accumulator contains a positive number it is considered the length of time, in thirtieths of a second, that the procedure wishes to sleep. If the specified accumulator contains a negative number then the procedure will sleep until the magnitude of system time (as read by .RDTIME [Sec 6.3.1]) is greater than the magnitude of this negative number. When a .SLEEP returns, the accumulator specified will be zero, but if a program is interrupted out of either type of .SLEEP the accumulator will be seen to contain the appropriate negative number.

6.8.2 The .GENSYM UUO

```
CALL:       .GENSYM AC,
            ;return
```

This system call returns in the specified accumulator a valid sixbit file-name-like symbol that will be unique among those returned by .GENSYM's for a particular system run.

6.8.3 The .IOTLSR UUO

```
            MOVSI AC,400000  ;or 0 to release
CALL:       .IOTLSR AC,
            ;return
```

Procedures run by ITS in IOT-user mode may execute hardware input-output instructions which would otherwise be ineffective and trap to the monitor. This system call sets the IOT-user status of the procedure executing it from the sign bit of the specified accumulator. If this bit is a one the procedure will attempt to enter IOT-user mode. If it is not currently in IOT-user mode it will enter it and a message will be output by the system job [Sec 6.5] unless the procedure is in a disowned tree in which case it will receive an illegal instruction interrupt [Sec 4.2]. If this bit is a zero, the procedure will not be in IOT-user mode after the .IOTLSR.

6.8.4 The .SETLOC and .IFSET JUO's

```
            MOVE AC,[DATLOC,,ABSLOC]
CALL:       .SETLOC AC,
            ;return

DATLOC:     ...             ;data from here written in ABSLOC
```

The .SETLOC system call treats the specified accumulator as two half word pointers. It takes the contents of the relative user location pointed to by the left half and places it in the absolute location pointed to by the right half and causes the system job to output an appropriate message [Sec 6.5] unless the procedure executing it is in a disowned tree in which case the procedure will receive an illegal instruction interrupt [Sec 4.2].

```
            MOVE AC,[DATBLK,,ABSLOC]
CALL:       .IFSET AC,
            ;error return
            ;normal return

DATBLK:     TESTWD          ;tested against c(ABSLOC)
            SETWRD          ;set into ABSLOC if test succeeds
```

The .IFSET system call is very similar to .SETLOC. The left half of its accumulator, however, points to a two word block. It first compares the first of these two words against the specified absolute

location. If they are not equal, it returns immediately without skipping. If they are equal, it proceeds as for .SETLOC but sets from the second word of the block and skips on returning.

Going to the extra effort to use .IFSET is much safer if you think there is some chance that the absolute location you are setting may be moved by the core allocator.

6.8.5 The .MASTER UUO

```
        MOVSI AC,400000  ;or 0 to release
CALL:   .MASTER AC,
        ;error return
        ;normal return
```

ITS has a special feature whereby one procedure can receive preferential treatment. It will be scheduled [Sec 4.4] with twice the priority of a normal procedure (runnining in competion with a second running job, it would reach an equilibrium where it was using twice as much machine time) and is given preference in the use of the DEC 340 display [Sec 3.4.1].

To request this status a procedure should execute a .MASTER with the 4.9 bit of the specified accumulator a one. It will succeed, and the .MASTER will skip, if no procedure has the facility seized or if

the procedure executing the .MASTER has control [Sec 3.2.1.1] of teletype T00 (the "main console" in front of the 340 display). If the later case, the facility may be removed from anotner procedure having it. A failing .MASTER returns without skipping.

The facility may be released by executing a .MASTER with the specified accumulator having bit 4.9 zero. This always skips even if the procedure does not have the facility to release.

6.8.6 The .REDEF UUO

```
            MOVEI AC,SYMBLK
CALL:       .REDEF AC,
            ;error return
            ;normal return

SYMBLK:     SQUOZE BITS,sym
            VALUE
```

This system call allows the user to define, redefine, and delete system symbols from high core DDT's symbol table. It is illegal for a procedure in a disowned procedure tree. The accumulator specified by it should contain a pointer to a two word block. The first word in this block should be the "squoze" code [Ref 3] for the symbol to be effected. The second word is the new value for a define (the symbol is not currently in the table) or redefine (the symbol currently is in the table). A delete request is indicated by having

all four squoze flag bits (BITS in the above illustration) on. The call skips if successful. It can fail only on an attempt to define a new symbol when there is no more room above the time sharing system.

7. DDT

See also section 0.3.1.

7.1 Introduction

DDT is a general purpose machine language debugging program that has been extensively modified to act as a supervisory procedure in ITS. It is automatically loaded by ITS [Sec 1.1.1] when a user informs ITS of his presence at an idle teletype. All further requests of the system by the user, until he logs out, are by means of trapping instructions [Sec 1.2] executed for the user, possibly as a result of characters he types in, by DDT or procedures inferior [Sec 5.1] to and under the control of DDT. The current binary of DDT exists as the file "@ HACTRN" on device SYS [Sec 3.1.3]. For historic reasons, a top level DDT is known as a HACTRN (pronounced hack-tran).

The initial command structure of DDT provided single character debugging commands of limited mnemonicity. This structure was badly strained by the introduction of time-sharing related commands. Then a new set of multi-letter commands was introduced. This set was expanded with the thought of providing all the most frequently used commands that did not relate to machine language debugging. Since a beginning or transient user might wish to use only these mnemonic

commands, a mode switch was added to DDT. There is a "monitor mode" in which only the multi-letter commands are available besides a few interrupt level commands that are unaffected by mode. In "DDT mode" all of the single character debugging commands are also available.

Section 7.2 below gives more information on monitor mode and describes almost all monitor mode commands. Section 7.3 describes the few interrupt mode commands in HACTRN and section 7.4 explains how DDT tells the user about error conditions. The information in these three sections should be more than enough to enable extensive use of ITS, particularly the conversational and interpretive programs available under it (such as TECO, LISP and STRING).

The remaining section (7.5) gives information on finding out about DDT mode commands. These would be important to someone developing a machine language program.

7.1.1 When DDT Starts

When a new copy of DDT is started, it types out certain useful information. If output is to a GE Datanet 760 console, it first clears the screen. Then it types out "DDT.xxx." where xxx is the version number of the DDT so that bugs can be correctly matched to version. Then, if ITS is in the system going down mode [Sec 3.5.2], a message indicating its remaining life span is output. Finally, if a file named "SYSTEM MAIL" is found on device SYS [Sec 3.1.3], it is

typed out.

7.2 Monitor Mode Commands

When in monitor mode [Sec 7.1], DDT indicates its readiness to accept a command by typing out a ":". At this point the user can type a series of letters followed by a break character (normally space or carriage return). DDT looks up this symbol (or its first six letters if longer) in a table of monitor mode commands. A user may also type a comment that will be ignored by DDT in or before this string by beginning and ending it with an "^$" (alt.-mode). No "^$"s may be embedded in the comment. If the symbol is found, control is transferred to a routine associated with it which may perform immediate actions or wait for more information to be typed by the user. At any time during the typing of a command, the user can abort and start over by typing a rubout.

If the string (say "llll") after the ":" is not found in DDT's command table, it looks for a file called "TS llll" on device SYS [Sec 3.1.3]. If not found an error comment [Sec 7.4] to that effect is output. If found, an inferior procedure is created, this file loaded into the inferior core image, and it is then started. The effect is as though the following [Sec 7.2] were typed

```
:JOB 1111
:LOAD SYS: TS 1111
:START
```

except that symbols for the program are not loaded into DDT from the file and the normal effects of :LOAD on the current file and device names in DDT is absent.

7.2.1 Logging In and Out

:LOGIN uname

This command takes as its argument the "uname" that the user wishes to log in with [Sec 6.2]. A "?" will be typed out and the command will be ineffective if someone is already logged in with that name. For ease in accessing his disk files, a user should use a consistent UNAME.

At the time the user logs in, DDT looks for a file named "uname MAIL" on device COM [Sec 3.1.3]. If not found, no action is taken by DDT. If found, DDT types it out, deletes the file "uname OMAIL", if any, on device COM, and renames the "uname MAIL" file as "uname OMAIL".

:LOGOUT

This command expunges all of the user's procedures from the system including the user's MACTRN [Sec 6.1]. The teletype the user

was typing on will become free and a message to that effect will be typed out on it by the system job [Sec 6.5]. DDT also deletes the file "uname CMAIL" on device COM at this time (where "uname" is the name the user logged in as).

7.2.2 Inter User Communication

:MAIL uname string...^C

This command takes as its first argument the user name of someone who uses ITS and as its second argument a string terminated by a "^C". The command prefixes the string to the file "uname MAIL" on device COM [Sec 3.1.3] if such a file exists. If no such file exists, one is created with the string as its contents. This file is presented to the specified user when he logs in [Sec 7.2.1]. The user can rubout individual characters while typing the string or cancel the command by typing enough rubouts.

:BUG string...^C

This command is intended for use in reporting bugs in systems software. It is identical to ":MAIL SYS string...^C".

:SEND uname string...^C

This command takes arguments like :MAIL above but attempts to send the message string supplied to the user specified in a direct

manner so that, if he is logged in, his HACTRN will type the string out on his console. If he is not logged in, LDT will type out "(MAIL)" and perform the actions specified under :MAIL above.

:GAG number

This command enables a user to gag messages that other users may attempt to send to him with :SEND (above). It takes a numeric argument that is bit decoded as per the table below.

Bit	Source
1.1	Other user's inferiors.
1.2	Other user's HACTRN's.
1.3	Your inferiors.
1.4	Your HACTRN (yourself).

If a bit is on it inhibits messages from the source listed. If off, messages are not inhibited. The initial state is :GAG 1.

The current implementation gives one no way to tell if a message sent was gagged by the intended receiver.

7.2.3 Inferior Procedures

:JOB {jname}

This command may be followed by an optional job name [Sec 5.1]. If there is an inferior procedure with this JNAME it will be selected by HACTRN as its "current job" (possibly attaching a disowned tree

[Sec 5.1.2]). Many commands in DDT affect the current job only as this changes infrequently and one would not want to respecify it with each command. If there is no inferior with the specified name, one will be created [Sec 5.1.1] and selected as the current job.

The job names SYS and PDP10 are special. The first allows the user limited access to absolute core. The second allows full access to the core image of the PDP-10 [Sec 3.4.2.1] to one user.

If no argument is supplied to :JOB it attempts to select some job different from the current one and type out "jname$J" to inform the user it has selected job "jname". If there is no current or other job, nothing is typed out. If the current job is the only inferior of HACTRN it will be reselected. If other inferiors exist one is selected in such a way that all inferiors are cycled through by successive :JOB's with no argument.

:KILL

This command obliterates [Sec 5.2.1] the current job and all its inferiors if it is an immediate inferior. If not an immediate inferior the variable space for it in HACTRN is reclaimed and logical pointers to it obliterated. In either case a :JOB with no argument is then simulated to minimize the time during which there is no current job.

:LISTJ

This command lists all of the inferiors of which HACTRN is aware. An "*" is printed before the current job, if any. After each job will be a character indicating its state as follows:

Character	Status
-	Just created or loaded.
R	Running.
P	Procedable from random interrupt.
Bn	Procedable from breakpoint n [Sec 7.5].
W	Interrupt/message to HACTRN pending.

After this letter is the user index of the procedure (see section 8.2, mode N). An example of :LISTJ output appears in Appendix G.

:DISOWN

This command disowns [Sec 5.2.2] the current job, deselecting it, expunging all information about it in HACTRN,and simulating a :JOB with no argument.

:LOAD {file specification}

This command attempts to load a file into the current job. HACTRN has a current device, system name [Sec 2.2.3], and pair of file names which it will use if nothing is specified after a command taking a file argument. This current file is initially device DSK, file name "@ BIN", and the system name the user logged in as [Sec 7.2.1]. Various fields can appear in the file specification

This command lists all of the inferiors of which HACTRN is aware. An "*" is printed before the current job, if any. After each job will be a character indicating its state as follows:

Character	Status
-	Just created or loaded.
R	Running.
P	Procedable from random interrupt.
Bn	Procedable from breakpoint n [Sec 7.5].
W	Interrupt/message to HACTRN pending.

After this letter is the user index of the procedure (see section 8.2, mode N). An example of :LISTJ output appears in Appendix G.

:DISOWN

This command disowns [Sec 5.2.2] the current job, deselecting it, expunging all information about it in HACTRN,and simulating a :JOB with no argument.

:LOAD {file specification}

This command attempts to load a file into the current job. HACTRN has a current device, system name [Sec 2.2.3], and pair of file names which it will use if nothing is specified after a command taking a file argument. This current file is initially device DSK, file name "@ BIN", and the system name the user logged in as [Sec 7.2.1]. Various fields can appear in the file specification

distinguished by their terminating character. A "dev:" field sets the device to "dev". An "sname;" sets the system name to "sname". A field terminated by a space or carriage return (the later indicating the end of the file specification) is treated differently depending on how many of this type of field have been seen. The first, if any, such becomes the first file name. The second, if any, becomes the second file name. A third, if supplied, becomes the device name (as if followed by a ":") and indicates the end of the file specification.

:DUMP {file specification}

This command dumps the current job as a file optionally specified as in :LOAD above.

:START {number}

This command starts the current job, giving it control of the user's console [Sec 3.2.1.1]. If no argument is supplied it is started at the last place it was started with a :START. If not :START'ed before then at the assembled in starting address that was read in as the program was loaded. If no starting address was supplied a "?" is typed out.

If a numeric argument is supplied, the job is started at the specified address which is stored away for future :START's without argument.

:CONTIN{UE}

This restarts the current job if it is stopped and gives it control of the user's console. It can be used to continue after returning to HACTRN with a "^Z" [Sec 1.1.2].

:PROCED

This command restarts the current job if it is stopped but leaves HACTRN in control of the user's console. Using this command the user can easily get several inferiors running at the same time.

7.2.4 Files and Input-Output

:PRINT {file specification}

This command treats a file as ASCII characters and prints it out. It accepts an optional file specification as described under :LOAD [Sec 7.2.3] above.

:DELETE {file specification}

This command deletes [Sec 2.4] a file. It takes an optional file specification as described under :LOAD [Sec 7.2.3] above.

:LISTF {device name}

This command lists the files on HACTRN's current device or the

device specified as an argument [Sec 2.2.1].

:FLAP number

This command must be followed by the number of a DEC tape drive. It causes the tape on the specified drive to be physically demounted and its directory to be excised from ITS [Sec 3.1.2.1].

:LINK {file name pair} sname; fnam1 fnam2

This command creates a link [Sec 3.1.1.1] under HACTRN's current system name to the file "fname1 fnam2" under system name "sname". The link's file name is also "fnam1 fnam2" unless the optional file names shown above are supplied.

7.2.5 The "Control-P" Feature

DDT has a feature which is designed to reduce the need to repeatedly type the same or very similar file names at different systems programs. For historic reasons this is called a "Control-P" feature. This feature is implemented using a block of four locations in DDT. These are deposited in the core image of a newly loaded inferior by the execution of certain DDT mode [Sec 7.5] commands and certain of the commands explained below. The locations set in an inferior and their significance are as follows:

Location	Significance
50	First file name.
51	New second file name.
52	Current second file name.
53	Old second file name.

:CR

This command reads in a file name as for :LOAD [Sec 7.2.3]. It puts the first file name read in the DDT word to be deposited in location 50. It puts the second file name in the DDT word to be deposited in location 52 and the second file name incremented in the word to be deposited in location 51. It clears the word to be deposited in 53 and finally loads, deposits in 50 through 53, and starts a copy of TECO, the normal ITS editing program, as for a :TECO [Sec 7.2] except for the depositions in 50 through 53.

:ED

This command increments the file name in the location to be put in 51 and then puts the incremented file name back after moving what was there into the location to be put in 52 and similarly moving that word into the location to be put in 53. It then loads, deposits in, and starts a copy of TECO, the normal ITS editor, as for a :TECO [Sec 7.2] except for the depositions in 50 through 53.

:ST

This command simply reads a file name and does the same thing

with it as :CR (described above) except it does not load or start a TECO.

7.2.6 Miscellaneous

:?

This command lists all important monitor mode commands with a short explanation of each. An example is shown in Appendix G.

:DDT

This command switches HACTRN to DDT mode [Sec 7.1].

:MON

This command switches HACTRN to monitor mode [Sec 7.1].

:XFILE {file specification}

This command reads the file, optionally specified as for :LOAD [Sec 7.2.3] above, and, treating it as ASCII, executes it as HACTRN commands.

:SYMLOD {file specification}

Mostly useful in conjunction with DDT mode [Sec 7.5] commands, this command loads symbols only for the current job. It takes an optional file specification as for :LOAD [Sec 7.2.3] above.

:ERR

Mostly useful in conjunction with DDT mode [Sec 7.5] commands, this command interprets the last quantity typed out by HACTRN as a .STATUS [Sec 2.5] word using the ERR device [Sec 3.4.4].

7.3 Interrupt Level and Context Free Commands

There are six single character commands to HACTRN that are interpreted at its interrupt level or at its top level in a manner independent of and not affecting the context in which they occur. They are presented in alphabetic order below. Two provide the user with the ability to "silence" any particular output of his HACTRN and abort it from a hung state. The remaining four control where HACTRN's output goes.

Character Effect

^B Attempts to seize the line printer [Sec 3.2.2]. If successful, causes HACTRN output to be printed.

^E Releases the line printer [Sec 3.2.2] and stops HACTRN output from being printed.

^G Interpreted at the the interrupt level, this command causes HACTRN to abort from whatever it is doing, type out "QUIT?", and wait for new commands. This command should only be used when necessary (ie HACTRN seems to be permanently hung up) as there is almost no protection from interrupting out of embarrasing places where variables are out of phase, etc. All buffered type in and out is also reset [Sec 2.7.2].

^S This charcter is interpreted at both interrupt and top level. It inhibits MACTRN output, also doing a type out reset [Sec 2.7.2], between these two interpretations. Thus "^S" is a much less drastic step than "^G" (above) to be used when the only trouble is that MACTRN is being somewhat long winded. It re-enables output when MACTRN gets around to it at the top level.

^V This command causes MACTRN output to be typed. It is the initial state.

^W This command stops MACTRN output from being typed.

7.4 Error Comments

DDT used to have two error comments, "U" and "?". The first of these is typed out when an undefined symbol is evaluated, usually when it is terminated by a non-symbol-constituent. The second was typed out for all other errors. It has been significantly augmented.

Errors in attempting to open, rename, or delete a file are indicated by the comment produced by the ERR device [Sec 3.4.4] followed by the symbolic name of the device and a question mark. In the case of an input-output error interrupt from an .IOT or .OPER, a special check is made to see if the error is device full on the input-output channel on which dumps are written. If so, the partial dump is deleted and an appropriate message output. Otherwise, the following error comment is output.

n, m, r, IOC?

where n is the error number, m the input-output channel on which

it occured, and p the location of the offending .IOT or .OPER. For other errors indicated by interrupts the following is output:

n, p, INT?

where n is a number with the interrupt bits on and p is the location interrupted from.

Other error comment listed in the following table:

Output	Meaning
UML?	Interrupt from unknown inferior.
TMJ?	Attempt to have too many inferiors.
CKS?	Checksum error on load.
ECF?	Unexpected end-of-file on read.
CFT?	Unable to assign console to inferior.
COR?	Can't get more core.
JOB?	No current job.
UNF?	DEC tape unflappable.
DSN?	Disown failed.
TMS?	Too many undefined symbols.
ILUUO?	Illegal UUO executed in DDT.
LOGIN?	You are not logged in.

7.5 DDT Mode Commands

Monitor mode commands may also be used in DDT mode. It is simply necessary for the user to type the initial ":". For further information on DDT mode commands, references 2, 7, and 8 of this memo are recommended. Also one might try next year's edition of this memo.

8. PEEK

8.1 Introduction

PEEK is a utility program operating under ITS. It enables a user to monitor a variety of aspects of the time sharing system by providing periodically updated display output or periodic character string output to teletype or line printer. Commands to PEEK are single letters sometimes preceded by numeric arguments. When initially started, PEEK is in N, or normal, mode [Sec 8.2]. If the user is on a GE console [Sec 3.2.1] PEEK will initially start outputing by typing (which, on a GE console, is effectively displaying). If the user is on a teletype, PEEK will try to seize the 340 display [Sec 3.4.1] and either display on the 340 or type its output depending on whether it succeeds or fails, respectively. This may be disconcerting if you are at a teletype not in view of the 340 because it will appear that the PEEK is just sitting there. This can be rectified with the ^N command [Sec 8.3].

The first line of all pages output by PEEK is as follows:

ITS xxx PEEK yyy mm/dd/yy hh:mm:ss

where xxx is the version number of the system in use [Sec 6.6.3], yyy is the version number of the PEEK in use (so bugs can be matched to

exact version), and mm/dd/yy is the date and hh:mm:ss the time according to ITS [Sec 6.3.2, 6.3.3]. If any core or disk parity errors have occured during the current system run, a second line is displayed listing the number of each. The last line of all pages displayed on the 340 by PEEK is a list of mode commands which can be given by light-penning [Sec 3.4.1.2] them with effects identical to typing them.

All characters typed at PEEK are interpreted at its interrupt level and have effect, if any, immediately. Characters that are not commands, digits, or ^Z are completely ignored. Digits are accumulated as an octal number until a non-digit is typed at which time the accumulated number is made available (if the non-digit is a command taking a numeric argument) and reset.

The amount of memory occupied by PEEK varies with mode but the present minimum is three memory blocks (a total of 6000 words).

8.2 Modes Available

The particular subset of the time sharing system's status being output by PEEK is controlled by its mode. Initially PEEK is in H mode but it may be changed to any of those listed below by typing the letter indicated or light penning [Sec 3.4.1.2] it if PEEK is displaying on the 340. Unless otherwise specified the output repetition rate for these modes is the standard rate initially set to

5 seconds delay after end of page output before starting on the next update. This is variable by the user with the Z command [Sec 8.4].

D This mode outputs system supplied directories [Sec 2.2.1] for the disk device [Sec 3.1.1]. The directory output will list only those files for the system name [Sec 2.2.3] of the PEEK. If the PEEK is running as an inferior of DDT [Sec 7], this system name may be easily changed by a DDT command. If output is to the 340 display, the user directory is preceded by a list of extant user directories which may be individually light penned to change PEEK's system names and select a different directory to display.

G This mode lists the sixbit symbols recognized by .GETSYS [Sec 6.6.1] in the system in use. Output is updated after a ten second delay.

H Only available with the 340 display, this mode outputs a graph of memory with different memory users [Sec 4.5] on different vertical levels and location represented by horizontal coordinate.

I The symbolic devices implemented in the system in use [App A, E] are listed by this mode. Output is updated after a ten second delay.

M This mode is somewhat reminiscent of H mode. It lists all memory users and the amount of memory they are using in decreasing order. An "*" is also output by stopped jobs as are the characters for special resources being used by any job (see table below).

N This is the most frequently used mode of PEEK. It tries to give a snapshot of the status of all user jobs in the system. Different amounts of information are displayed depending on the particular output device in use, but in all cases the following additional information is output:

(1) Above the list of user jobs, the number of free blocks (each 2000 words) of memory (MEMFR), the number of areas that are allocated for user variable [App D] (USRHI), and the number of user programs that were runnable last schedule time (RNABLU);
(2) Below the list of user jobs, the number of blocks of memory occupied by user programs (USR MEM) (the sum of this figure and MEMFR is not constant due to dynamic IO buffers), the total percentage of machine time being used by the listed programs (USR

TIM) (less than one hundred due to scheduling and other overhead), the amount of time that the running system has been up (SYSTIM), and the total amount of machine time used by users that have logged out (LOUTIM).

Between the two sets of information described above, there is one line of information for each procedure in the system topped by a title line which provides a heading for each column of information. The first item on the line for any particular procedure will be the user index of the procedure, a number that uniquely refers to it (see mode V). The second item will be either the UNAME for top level procedures and the JNAME for inferiors or the UNAME, JNAME, and SNAME for each procedure depending on the line width of the output device. This second item also portrays the existant procedure tree structure in that each procedure not at the top level is an inferior of the nearest procedure above it whose name is not indented as far.

Other items include an exponential approximation to the percentage of machine time the procedure has been receiving, the number of core blocks it is occupying, a column labeled "TTY" that has a "T" concatenated with the console number opposite procedures controlling a console, various letters immediately after the "TTY" column for system resources controled by the procedure (see the table below), and a column labeled "STATUS". The character string opposite a procedure in the STATUS column should be interpreted as follows: (1) a leading "*" indicates that the procedure is processing a software interrupt [Sec 4.2], (2) RUN indicates that the program is running in user mode, (3) a pair of numbers separated by an "!" indicates that the procedure is stopped, (4) most other strings represent a system call (with S used for the very common .SLEEP call) or input-output device and type of transfer; (4A) if preceded by a "+" the procedure is running in executive mode performing the call or input-output; (4B) if not it is hung on the call or input-output.

T The contents of the system translation table [Sec 2.2.2] is output.

U This mode displays the status of the DEC tape drives [Sec 3.1.2] and of open DEC tape files. For the normal user, the most useful information per tape is the drive number in the leftmost column and the DIR variable in the next to the rightmost (UTASS) column. If the DIR variable is minus one, ITS is not retaining a directory for that drive and a tape mounted on it could be removed manually. Otherwise it is the absolute location of the directory. The per open file information includes the names of the procedure using the file, its direction, and the number of buffers (200 words each) currently being used for the file by the system.

V This mode command should be proceeded by the user index (see mode N above) of some user. It displays many of his user variables [App D] and the channel word and status word for each of his input-output channels [Sec 2.1, 2.5]. If output is to the 340 or line printer, the contents of the selected user's accumulators are also output.

X This mode outputs the digitalization of all the multiplexed analog to digital input channels [Sec 3.3.3] and the value being output on each of the multiplexed digital to analog channels [Sec 3.3.2].

Y This mode outputs system supplied [Sec 2.2.1] directories for DEC tapes. It should be proceeded by a drive number.

? This mode outputs a list and brief explanation of PEEK's commands. An example of its use is included in Appendix G. Output is updated after a ten second delay.

The following are the system resource letters used in modes N and M above:

C Indicates that the procedure is now or was the last to use the core allocator.

D Indicates that the procedure has seized control of the DEC 340 display.

F Indicates that the procedure has control of the COD device.

I This letter can be present for more than one procedure. It indicates that the procedure it appears with is in "IOT-user" mode [Sec 6.8.3].

L Indicates that the procedure has control of the line printer.

M Indicates that the procedure has seized the .MASTER [Sec 6.8.5] facility.

P Indicates that the procedure has control of the plotter.

R Indicates that the procedure has control of the paper tape reader.

T Indicates that the procedure has control of the paper tape punch.

10 Indicates that the user has the PDP-10 core image open on device USR [Sec 4.3.2.1].

8.3 IO Control

PEEK will output to only one device at a time. The initial device is chosen as explained in section 8.1. All of the following four commands for changing PEEK's output device also cause PEEK to immediately restart output for its current mode [Sec 8.2]:

^B This switches output to the line printer if it is available. If not available the output device will be unchanged.

^E This command will terminate PEEK output to the line printer and cause it to either type or display its output.

^N This command terminates 340 diplay output, starts typed output, and sets a flag in PEEK that is cleared only by the ^Y command. This flag inhibits PEEK's attempts to seize the 340 as described in section 8.1.

^Y PEEK will attempt to seize the 340 display for output on receipt of this command. If it is unsuccesful, output will revert to typing. In any case the flag mentioned under ^N above is cleared.

8.4 Miscellaneous

The following are miscellaneous PEEK commands:

P If the PEEK is running under DDT [Sec 7], this command will return control of the users console to DDT but leave PEEK running and, if the 340 display or line printer is selected, producing output. It uses .VALUE to return the string "^P" as a command to DDT.

Q If the PEEK is running under DDT [Sec 7], this command will destroy the PEEK it is typed at and return control of the users console to DDT. It uses .VALUE to return the string "$^X." as a command to DDT.

Z This command should be proceeded by a small number. It sets the standard update delay to as many seconds [Sec 8.2].

! This command will probably go away when swapping is added to the system. Under normal circumstances it will cause the procedure tree in which is found the PEEK it is typed at to be replaced by a single newly loaded PEEK at the top level. This saves memory if several people want to leave a PEEK at a single console running for their joint enlightenment. A top level PEEK will commit suicide if a ^Z is typed at it.

9. LOCK

9.1 Introduction

LOCK is a utility program operating under ITS performing a multitude of infrequently required tasks. Its name derives from the fact that the function it was originally written for was to "lock" and "unlock" teletypes (see the + and - commands [Sec 9.2]). Most of LOCK's commands are single characters.

In contrast to PEEK [Sec 8], which interprets commands at the interrupt level, commands to LOCK are read at its main program level. Thus commands will in general have no effect before previous commands have been completed. In fact, some commands cause a change in the state of LOCK such that following letters are interpreted as arguments.

LOCK occupies only one block (2000 words) of memory except after the "T" command.

9.2 Commands

The following are the current LOCK commands:

+ This command should be proceeded by a teletype number [Sec 3.2.1]. The LOCK will try to open the specified teletype as an input-output device. It will be succesful only if the teletype is not in use. If it is successful, the teletype will become impervious to ^Z's so no user can log in on it, LOCK will type out an appropriate message on the now locked teletype, and LOCK will type a "Y" to the LOCK user. If the LOCK fails in opening the teletype it will type an "L" at the LOCK user.

- This command should be proceeded by a teletype number. If LOCK has the specified teletype locked (see "+" above) it will unlock it and type out an "*". Otherwise it will type a "?".

_ This command should be preceded by the octal code for a character. LOCK will type out the character represented in image mode [Sec 3.2.1]. This is useful for determining the reaction of a console to various codes.

G This command should be preceded by the user index [Sec 3.2, see mode M] of the top procedure in a tree that the user wishes to obliterate. It must be followed by the characters "UN". If the user deviates from this sequence, a "?" will be typed and LOCK will listen for a new command. If the user does not deviate the system job [Sec 6.5] forceably logs out [Sec 5.4] the tree and types out a message on the system job teletype.

I A series of magic characters must be typed at a model 37 teletype to allow its use in full duplex, the mode assumed by most programs operating under ITS. This command causes LOCK to type the requisite characters in image mode [Sec 3.2.1].

K This command must be immediately followed by the characters "ILL" (see G above) and should be preceded by a number. It will activate the system going down feature [Sec 6.5.2] with a time limit of five or the number preceeding the command minutes, whichever is larger.

O When it receives this command, LOCK will permanently go into a mode where characters typed at it are echoed as their ASCII values in octal. To ultimately escape from LOCK in this mode, the user should use ^Z [Sec 1.1.2].

P If LOCK is running under a DDT [Sec 7], this command causes control of a user's console to revert to it. The command uses .VALUE to return a null string.

S If LOCK is running under a DDT [Sec 7], this command

causes control of a user's console to revert to his LDT and his LOCK to be destroyed. Note that this will unlock all teletypes previously locked (see "+" above) by the destroyed LOCK as destroying a procedure closes all its channels. It uses .VALUE to return the string ":KILL".

S This command complements the run status of the system checking feature of the system job [Sec 6.5.1]. It will type out the resultant state as either "START" or "STOP".

T This command causes complex and nonterminating gyrations on the part of the LOCK it is typed at and the procedures the LOCK will create as a result of this command. Its purpose is to test the core allocator [Sec 4.3] and other components of the ITS system. Its casual use is not recommended.

? This command causes LOCK to type out an up to date list of its commands with a brief explanation of each. An example appears in Appendix G.

References

1. Digital Equipment Corporation
Programmed-Data-Processor-6 Handbook (F-85)
August 1964, DEC, Maynard, Massachusetts.

2. Digital Equipment Corporation
DDT (DEC-6-0-UP-DDT-UM-FP-ACT00),
April 1965, DEC, Maynard, Massachusetts.

3. Peter R. Samson
MIDAS (MAC-M-279, AI-90)
October 1965, Project MAC memo.

4. Peter R. Samson
MAC PDP-6 DEC-tape File Structure (MAC-M-249, AI-82)
Project MAC memo.

5. Michael D. Beeler
[untitled] (AI-128)
March 1967, MAC AI group memo.

6. Michael D. Beeler
IO Test (AI-144)
October 1967, MAC AI group memo.

7. Digital Equipment Corporation
DDT-10 (DEC-10-CDDB-D)
June 1968, DEC, Maynard, Massachusetts.

8. Thomas F. Knight
A Multiple Procedure DDT (AI-147)
January 1968, MAC AI group memo.

9. John T. Holloway
Output to the PDP-6 Calcomp Plotter (AI-104)
August 1966, MAC AI group memo.

10. Donald E. Eastlake III
PDP-6 Software Update (Revised AI-118)
June 1967, MAC AI group memo.

11. Donald E. Eastlake III
ITS 1.4 Reference Manual (MAC-C-377, AI-181)
June 1968, MAC memo.

12. Donald E. Eastlake III
PEEK and LOCK (MAC-M-367, AI-169)
November 1968, MAC memo.

13. Stewart Nelson, Michael Levitt
Robot Utility Functions (AI-172)
February 1969, MAC AI Group memo.

Appendix
A. System Calls in Numeric Order
(correct as of ITS version 628)

Name	Number	Section
.IOT	UUO 040	2.3
.CPEK	UUO 041	2.2
.OPER	UUO 042	1.2.1
.ITYI	.OPER 1	App E
.LISTE	.OPER 2	3.2.1.2
.SLEEP	.OPER 3	6.8.1
.SETMS	.OPER 4	App E
.SETM2	.OPER 5	4.2.1
.LOGIN	.OPER 6	6.1
.CLOSE	.OPER 7	2.7.1
.UCLOS	.OPER 10	5.2.1
.ATTY	.OPER 11	3.2.1.1
.DTTY	.OPER 12	3.2.1.1
.IOPUS	.OPER 13	2.6.1
.IOPOP	.OPER 14	2.6.1
.DCLOS	.OPER 15	3.4.1.3
.DSTOP	.OPER 16	3.4.1.4
.RDTIM	.OPER 17	6.3.1
.RDSW	.OPER 20	6.2.1
.GUN	.OPER 21	5.4
.UDISM	.OPER 22	3.1.2.1
.GETSY	.OPER 23	6.6.1
.RD500	.OPER 24	6.2.4
.GETLO	.OPER 25	6.6.2
.SETLO	.OPER 26	6.8.4
.DISOW	.OPER 27	5.2.2
.RD760	.OPER 30	6.2.2
.WR760	.OPER 31	6.2.2
.GENSY	.OPER 32	6.8.2
.LOGOU	.OPER 33	6.1
.GSNAM	.OPER 34	App E
.WSNAM	.OPER 35	App E
.UPISE	.OPER 36	App E
.RESET	.OPER 37	2.7.3
.ARMOV	.OPER 40	3.3.2.1
.WMAR	.OPER 41	App E
.RRTIM	.OPER 42	App E
.ASSIG	.OPER 43	3.1.2.2
.DESIG	.OPER 44	3.1.2.2
.RTIME	.OPER 45	6.3.2
.RDATE	.OPER 46	6.3.3

.RD710	.OPER 47	6.2.3
.EOFC	.OPER 50	2.3.1
.IOTLS	.OPER 51	6.6.3
.RSYSI	.OPER 52	6.6.3
.SUPSE	.OPER 53	6.5.1
.UBLAT	.OPER 56	3.1.2.3
.IOPDL	.OPER 57	2.6.2
.ITYIC	.OPER 60	2.7.3
.MASTE	.OPER 61	6.8.5
.VSTST	.OPER 62	3.3.1.1
.DIAL	.OPER 63	3.2.1.3
.DIALW	.OPER 64	3.2.1.3
.HANGU	.OPER 65	3.2.1.3
.DIETI	.OPER 66	6.5.2.2
.SHUTD	.OPER 67	6.5.2.1
.ARMOF	.OPER 70	6.6.2.1
.MDIS	.OPER 71	3.4.1.5
.FEED	.OPER 72	3.2.4.1
.EVAL	.OPER 73	6.6.4
.REDEF	.OPER 74	6.8.6
.ITSET	.OPER 75	6.8.4
.UTNAM	.OPER 76	3.1.2.4
.UINIT	.OPER 77	3.1.2.5
.CALL	UUO 043	1.2.1
.DISMI	.CALL 1,	4.2.2
.TRANS	.CALL 2,	2.2.2.1
.TRANA	.CALL 3,	2.2.2.2
.VALUE	.CALL 4,	5.3.1
.UTRAN	.CALL 5,	5.1.3
.CORE	.CALL 6,	4.3.1
.TRAND	.CALL 7,	2.2.2.3
.DSTAR	.CALL 10,	3.4.1.1
.FDELE	.CALL 11,	2.4
.DSTRT	.CALL 12,	3.4.1.1
.SUSET	.CALL 13,	4.5
.LTPEN	.CALL 14,	3.4.1.2
.VSCAN	.CALL 15,	3.3.1.1
.POTSE	.CALL 16,	3.3.3.1
.USET	UUO 044	5.2.3
.BREAK	UUO 045	5.3.2
.STATU	UUO 046	2.5
.ACCES	UUO 047	2.7.4

Appendix
E. System Calls in Alphabetic Order
(correct as of ITS version 528)

Name	Number	Section
.ACCES	UUO 047	2.7.4
.ARMOF	.OPER 70	3.3.2.1
.ARMOV	.OPER 40	3.3.2.1
.ASSIG	.OPER 43	3.1.2.2
.ATTY	.OPER 11	3.2.1.1
.BREAK	UUO 045	5.3.2
.CALL	UUO 043	1.2.1
.CLOSE	.OPER 7	2.7.1
.CORE	.CALL 6,	4.3.1
.DCLOS	.OPER 15	3.4.1.3
.DESIG	.OPER 44	3.1.2.2
.DIAL	.OPER 63	3.2.1.3
.DIALW	.OPER 64	3.2.1.3
.DIETI	.OPER 66	6.5.2.2
.DISMI	.CALL 1,	4.2.2
.DISOW	.OPER 27	5.2.2
.DSTAR	.CALL 10,	3.4.1.1
.DSTOP	.OPER 16	3.4.1.4
.DSTRT	.CALL 12,	3.4.1.1
.DTTY	.OPER 12	3.2.1.1
.EOFC	.OPER 50	2.3.1
.EVAL	.OPER 75	6.6.4
.FDELE	.CALL 11,	2.4
.FEED	.OPER 72	3.2.4.1
.GENSY	.OPER 32	6.8.2
.GETLO	.OPER 25	6.6.2
.GETSY	.OPER 26	6.6.1
.GSNAM	.OPER 34	App E
.GUN	.OPER 21	5.4
.HANGU	.OPER 65	3.2.1.3
.IFSET	.OPER 75	6.8.4
.IOPDL	.OPER 57	2.6.2
.IOPOP	.OPER 14	2.6.1
.IOPUS	.OPER 13	2.6.1
.IOT	UUO 040	2.3
.IOTLS	.OPER 51	6.8.3
.ITYI	.OPER 1	App E
.ITYIC	.OPER 60	2.7.3
.LISTE	.OPER 2	3.2.1.2
.LOGIN	.OPER 5	6.1
.LOGOU	.OPER 36	6.1

.LTPEK	.CALL 14,	3.4.1.2
.MASTE	.OPER 61	6.8.5
.NDIS	.OPER 71	3.4.1.5
.OPEN	UUO 041	2.2
.OPER	UUO 042	1.2.1
.POTSE	.CALL 15,	3.3.3.1
.RD500	.OPER 24	6.2.4
.RD710	.OPER 47	6.2.3
.RD760	.OPER 30	6.2.2
.RDATE	.OPER 46	6.3.3
.RDSW	.OPER 20	6.2.1
.RDTIM	.OPER 17	6.3.1
.REDEF	.OPER 74	6.8.6
.RESET	.OPER 37	2.7.3
.RPTIM	.OPER 42	App E
.RSYSI	.OPER 52	6.6.3
.RTIME	.OPER 45	6.3.2
.SETLO	.OPER 26	6.8.4
.SETM2	.OPER 5	4.2.1
.SETMS	.OPER 4	App E
.SHUTD	.OPER 67	6.5.2.1
.SLEEP	.OPER 3	6.8.1
.STATU	UUO 046	2.5
.SUPSE	.OPER 53	6.5.1
.SUSET	.CALL 13,	4.5
.TRANA	.CALL 3,	2.2.2.2
.TRAND	.CALL 7,	2.2.2.3
.TRANS	.CALL 2,	2.2.2.1
.UBLAT	.OPER 56	3.1.2.3
.UCLOS	.OPER 10	5.2.1
.UDISM	.OPER 22	3.1.2.1
.UINIT	.OPER 77	3.1.2.5
.UPISE	.OPER 36	App E
.USET	UUO 044	5.2.3
.UTNAM	.OPER 76	3.1.2.4
.UTRAN	.CALL 5,	5.1.3
.VALUE	.CALL 4,	5.3.1
.VSCAN	.CALL 15,	3.3.1.1
.VSTST	.OPER 62	3.3.1.1
.WMAR	.OPER 41	App E
.WR760	.OPER 31	6.2.2
.WSNAM	.OPER 35	App E

Appendix
C. Available Symbolic Devices in Alphabetic Order
(correct as of ITS version 828)

Device	Type	Section
CLA	5	3.4.3
CLI	3	3.4.3
CLO	7	3.4.3
CLU	7	3.4.3
COD	6	3.2.5
COM	7	3.1.3
DIS	2	3.4.1
DKn	7	3.1.1
DSK	7	3.1.1
ERR	4	3.4.4
IDS	2	3.4.1
IMX	4	3.3.3
LPT	2	3.2.2
NUL	6	3.4.6
NVD	6	3.3.1
OMX	2	3.3.2
PLT	2	3.2.3
Pnm	7	3.1.1
PTP	2	3.2.4
PTR	4	3.2.4
SYS	7	3.1.3
Tnm	6	3.2.1
TPL	7	3.4.5
TTY	6	3.2.1
TVC	6	3.3.1
USR	6	3.4.2
UTn	7	3.1.2
VID	6	3.3.1

Appendix
D. System Variables per User
(correct as of ITS version 528)

```
ICCHNM:     REPEAT 20,0 ;input-output channel assignment words
                        ;right half, index into IOTTB, CLSTB,
etc.
                        ;left half, device dependent
SIOCHN:     BLOCK LUIOP ;input-output channel push down list
SIOCP:      SIOCHN-1 ;pointer into input-output push down list
IOCHST:     BLOCK 20 ;input-output channel status for channels at
ICCHNM
                        ;3.1-4.9 input-output status
                        ;1.1-2.9 .ACCESS pointer

UPC:        0       ;stores program counter when not running
UPR:        0       ;user protection and relocation registers
MEMTOP:     0       ;six less than first illegal location
RMEMT:      0       ;first illegal location
CORRQ:      -1      ;request to core job
                    ;1.1-1.8 number of blocks, 3.1-4.8 user
requested for
                    ;4.9=0 means request present, 4.9=1 means no
request

AC0S:       BLOCK 15 ;swap out accumulators
AC15S:      0       ;commonly referenced swap out accumulator
AC16S:      0       ; "
AC17S:      0       ; "

UUO:        ;the following three locations swapped in & out of
                    ;UEXIT, UUOH, etc with user
SUEXIT:     JRST 2,@UUOH ;user UUO exit instruction
SUUOH:      0       ;contents of @41 (absolute)
SAC17P:     MOVEM U, ;accumulator 17 in user's shadow memory

SV40:       0       ;last interrupting UUO
                    ;(contents of 40 when user out)
SV60:       0       ;contents of 60 when user out

AC14P:      0       ;points to accumulator 14 in shadow memory
AC15P:      0       ; " " " " 15 " " "
AC16P:      0       ; " " " " 16 " " "
40P:        0       ;points to user's relative location 40
41P:        0       ; " " " " 41
```

```
APRC:       APRCHN  ;right half is cono to processor
                    ;4.9 implies user is in a disowned tree
                    ;4.8 implies user being .UCLOS'ed
                    ;4.7 user has been cored zero
                    ;4.6 core request pending on this user
                    ;4.1-4.4 number of last channel on which an
                                    ;error occured

USTP:       0       ;stop word ;zero implies runnable
                    ;4.9 stopped to be moved by core allocator
                    ;4.8 general core allocator stop bit
                    ;4.7 controlled by superior procedure
                    ;4.6 special bit, see URMEMT & UMEMEX
                    ;1.1-2.9 count of transient reasons to be
                                    ;stopped

PIRQC:      0       ;pending first word interrupts
MSKST:      0       ;interrupt enable mask for PIRQC
IFPIR:      0       ;second word of interupt requests
                    ;3.8-3.1 inferior procedure interrupts
                    ;2.7-1.1 input-output channel interrupts
                    ;1.1 for channel 1, etc
MSKST2:     0       ;second word interrupt enable mask
PICLR:      0       ;interrupt level status flag

SUPPRO:     0       ;pointer and interrupt bit to superior procedure
                    ;-1 for top level procedure

FLSINS:     0       ;blocking condition instruction
                    ;zero implies runnable
UFINAL:     0       ;nonzero implies finalization required to
                                    ;interrupt
RPCL:       0       ;0 implies no RPCLosering in progress
                    ;0,,n RPLSR'ing user n
                    ;-1,,n being RPCLSR'ed by user n

UNAME:      0       ;user name
;This word for each procedure is copied from the UNAME
;of the procedure that creates it (see USR device).
;For an initial top level procedure it is -1 but is
;modified by .LOGIN.  It is the same for all procedures
;in a tree.

JNAME:      0       ;job name
;Each logged in procedure has a unique UNAME, JNAME pair.
;The JNAME of an initial top level job is "HACTRN".  For other
;jobs it is the second file name specified in the .OPEN on the
```

```
;USR device that creates them.

USYSNM:     0       ;current system name
;This variable is initially set equal to the procedure's UNAME.
;It can be read or set by the procedure or its superior (.SUSET,
                                    ;.USET).
;It is used as the third file name on certain shared devices.
USYSN1:     0       ;system name used inside disk routines
;This variable is normally equal to USYSNM but see devices COM, SYS,
                                    ;and TPL.

TTYTBL:     0       ;console assignment for this procedure

UQUAN:      0       ;quantum time
UTRNTM:     0       ;total run time
UTMPTR:     0       ;pointer to procedure tree resource word
JTMU:       0       ;procedure resource word

IOTLSR:     0       ;4.9 a one implies iot-user mode
                    ;4.5-4.3 MAR conditions
                    ;2.9-1.1 contents of MAR register
UMARPC:     0       ;program counter at last MAR interrupt
VALUE:      0       ;contents of effective address of last .VALUE
UOUT:       0       ;4.9=1 => user not completely in core
                    ;4.8 in transit in
                    ;4.7 in transit out
                    ;4.6 ref to disk in progress
                    ;4.5 back in core but needs to be blessed before
                                    ;starting
                    ;1.1-1.9 core blocks needed for swap in

USRPDL:     -LUPDL,,UPDL-1 ;push down list pointer for system calls
UPDL:       BLOCK LUPDL-2 ;push down list area for system calls
EPDL1:      0       ;link depth for 2311 routines
EPDL:       0       ;temporarily saves accumulator B for ACORE
EPDL2:      0       ;temporarily saves accumulator T for FLSINS
```

Appendix
E. Relics of ITS's Past

The following system calls, while still effective as of ITS version 523, can be expected to ultimately disappear. They are redundant, more specialized, older versions of their replacements.

Name	Number	Replacement	Function
.ITYI	.OPER 1	.ITYIC	Read interrupt level characters from open TTY device, if any.
.SETMS	.OPER 4	.SUSET	Set own user variable MSKST.
.GSNAM	.OPER 34	.SUSET	Read own user variable USYSNM.
.WSNAM	.OPER 35	.SUSET	Set own user variable USYSNM.
.UPISE	.OPER 36	.SUSET	Set own user variable PICLR.
.WMAR	.OPER 41	.SUSET	Set own user variable IOTLSR.
.RRTIM	.OPER 42	.SUSET	Read own user variable UTRNTM.

Appendix
F. Further Sources of Information on ITS

Information further than or more recent than that contained in this manual may usually be obtained by consulting the symbolic listings of ITS, DDT, PEEK, or LOCK. The following persons may also be helpful:

Donald E. Eastlake III

Jerry S. Freiberg

Richard D. Greenblatt

Thomas F. Knight

John T. Holloway

Stewart B. Nelson

Frederick H. G. Wright II

Appendix G. Illustrative Console Output

The following pages are a sample console session with ITS illustrating several features documented in this memo.

```
↑Z
  DDT.198.
:LOGIN DE
:PEEK!

↑N
    ITS 530   PEEK 142    07/14/69    00:22:15
PARITY ERRORS:  DISK =    11    CORE =     0
MEMFR=257 USRHI=11                 RNABLU=2
I= U-JNAME  STATUS TTY    CORE %TIM
 0 SYS      TYO    T5      25   0%
 1 CORE     UUO    ?        0   1%
 2 DE       S      >        6   3%
 3  PEEK    +GETSY T4   C   5   3%
10 MS     EOAMIC  RUN DISOWN  20 91%
USR MEM= 60  USR TIM= 97%
STIME = 7:03:22  LOUTIM = 0


?
    ITS 530   PEEK 142    07/14/69    00:22:58
MODE CONTROL:
  D  DISK DIRECTORY
  G  AVAILABLE .GETSYS'S
  H  MEMORY GRAPH (340 ONLY)
  I  AVAILABLE IO DEVICES
  M  MEMORY USE LIST
  N  NORMAL MODE
  T  TRANSLATION ENTRIES
  U  DEC TAPE STATUS
  V  SINGLE USER (PRECEDE BY INDEX)
  X  MULTIPLEXORS MODE
  Y  DEC TAPE DIRECTORY (PRECEDE BY #)
  ?  EXPLANATION MODE
IO CONTROL:
  ↑B USE LINE PRINTER
  ↑E STOP USING LINE PRINTER
  ↑Y USE 340
  ↑N STOP USING 340
OTHER:
  P  PROCEED BUT RETURN TTY TO DDT
  Q  QUIT
  Z  SET DOZE IN SECONDS
  !  STAND ALONE

Q

$↑X.
```

```
:LOCK!

 LOCK.39

←?

+  LOCK TTY (PRECEDE BY #)
-  UNLOCK TTY (")
←  OUTPUT CHARACTER WITH ASCII VALUE # (")
GUN  KILL USER WITH # INDEX (")
I  INITIALIZE A MODEL 37 TTY
KILL  SYSTEM DOWN IN # MINUTES (")
O  OUTPUT CHARACTERS IN OCTAL
P  RETURN TO DDT
Q  VALRET AN $↑X.
S  COMPLEMENT RUN STATUS OF SYS JOB
T  TEST CORE ALLOCATOR
?  LIST COMMANDS

←Q

$↑X.
:?
? = LIST MOST COMMANDS
DISOWN = DISOWN CURRENT JOB
LOGOUT = AUTO-EXPUNGE
SYMLOD = LOAD SYMBOLS ONLY
XFILE = EXECUTE FILE AS DDT COMMANDS
CR = ENTER TECO AND CREATE FILE
ED = ENTER TECO AND EDIT
LOAD = LOAD FROM FOLLOWING FILE INTO CURRENT JOB
DUMP = DUMP INTO " " FROM " "
KILL = KILL CURRENT JOB
LOGIN = LOGIN AS FOLLOWING NAME
SEND = SEND MESSAGE
GAG = CONTROL RECEIPT OF MESSAGES
BUG = DOCUMENT BUG
MAIL = ADD TO USER'S MAIL FILE
JOB = CREATE OR SELECT JOB
LISTJ = LIST JOBS
FLAP = FLAP DECTAPE, FOLLOW BY DRIVE #
LISTF = LIST FILES
DELETE = DELETE FILE
PRINT = PRINT FILE
LINK = CREATE LINK
ERR = IOC ERROR STATUS
DDT = ENTER DDT MODE
MON = ENTER MONITOR MODE
START = START INFERIOR
CONTIN = CONTINUE GIVING INF. TTY
PROCED = PROCEDE INF., LEAVE TTY WITH DDT
WALLP = ↑B TO SPECIFIED FILE
```

```
:LOGOUT
ITS 530 CONSOLE 4 FREE. 00:25:57
```