MASSACHUSETTS INSTITUTE OF TECHNOLOGY

PROJECT MAC

Artificial Intelligence
Memo. No. 161

MAC-M-377
June 1968

ITS 1.4 Reference Manual

Donald E. Eastlake III

This reference manual is intended for those who have some knowledge of PDP-6 machine language and are either interested in the ITS 1.4 monitor for its own sake, or who wish to write machine language programs to run under it. It should be remembered that the Project MAC, AI Group, PDP-6 installation is undergoing continuous software and hardware developments. Please direct all corrections, additions, or comments to the author.

0. Notation Used in this Manual 1
0.1 Numbers 1
0.2 Character Objects and Strings 1
0.3 Bit Positions and Intervals 1
0.4 Internal References 2

1. Introduction to and Goals of ITS 3
1.1 Preview of Section 2 4
1.2 Preview of Sections 3 and 4 5
1.3 Preview of Section 5 5
1.4 Preview of Section 6 6
1.5 Preview of Section 7 6

2. Getting the Attention of and Transmitting Commands to ITS 8
2.1 Control Z 8
2.2 Trapping Instructions 9
2.2.1 UUO's 10
2.2.2 Non-UUO Trapping Instructions 11

3. The General Transaction of Input-Output 12
3.1 Philosophy and Organization 12
3.2 The .OPEN UUO 13
3.2.1 File Directories 14
3.3 The Translation Table 15
3.3.1 The .TRANS UUO 15
3.3.2 The .TRANAD UUO 16
3.3.3 The .TRNDL UUO 17
3.4 The .IOT UUO 17
3.4.1 The End of File Condition 18
3.4.2 The Device Full Condition 19
3.5 The .FDELE UUO 20
3.6 The .STATUS UUO 21
3.7 The Input-Output Channel Push Down Facility 23
3.7.1 The .IOPUSH and .IOPOP UUO's 23
3.7.2 The .IOPDL UUO 24
3.8 Miscellaneous Uniform Input-Output Related System Calls 24
3.8.1 The .CLOSE UUO 24
3.8.2 The .RESET UUO 25
3.8.3 The .ITYIC UUO 25
3.8.4 The .WSNAME UUO 25
3.8.5 The .ACCESS UUO 26

4. Properties of Particular Devices 27
4.1 The UTn Devices 27
4.1.1 The .UDISMT UUO 28
4.1.2 The .ASSIGN and .DESIGN UUO's 28
4.1.3 The .UBLAT UUO 29
4.2 The DDn, SYS, and COM Devices 29
4.3 The CLO and CLU Devices 30
4.4 The NUL Device 30
4.5 The Tnm and TTY Devices 30
4.5.1 The .ATTY and .DTTY UUO's 32

4.5.2 The .LISTEN UUO 33
4.6 The LPT Device 34
4.7 The PLT Device 34
4.8 The PTP and PTR Devices 35
4.9 The IMX Device 35
4.10 The OMX Device 37
4.11 The DIS and IDS Devices 38
4.11.1 The .DSTART and .DSTRTL UUO's 41
4.11.2 The .LTPEN UUO 42
4.11.3 The .DCLOSE UUO 43
4.11.4 The .DSTOP UUO 43
4.12 The USR Device 44
4.13 The VID and NVD Devices 45
4.14 The COD Device 46

5. The Procedure 47
5.1 Philosophy and Organization 47
5.2 User Interrupts 48
5.2.1 The .SETMSK and .SETM2 UUO's 51
5.2.2 The .DISMISS UUO 51
5.2.3 The .UPISET UUO 52
5.3 The Core Allocator 52
5.3.1 The .CORE UUO 53
5.4 The Scheduler 53
5.5 The .SUSET UUO 54

6. Procedure Interaction 55
6.1 The Hierarchial Organization of Procedures 55
6.1.1 Procedure Creation 57
6.1.2 Attaching Disowned Procedure Trees 57
6.1.3 The .UTRAN UUO 58
6.2 Controls over an Immediately Inferior Procedure 59
6.2.1 The .UCLOSE UUO 59
6.2.2 The .DISOWN UUO 59
6.2.3 The .USET UUO 60
6.3 Communication to an Immediately Superior Procedure 61
6.3.1 The .VALUE UUO 61
6.3.2 The .BREAK UUO 61

7. Miscellaneous System Calls and Features 62
7.1 The .LOGIN and .LOGOUT UUO's 62
7.2 Nonstandard Devices 63
7.2.1 The .RDSW UUO 63
7.2.2 The .RD760 and .WR760 UUO's 63
7.2.3 The .RD710 UUO 63
7.2.4 The .ORGI and .ORGO UUO's 64
7.3 Various Times and the Date 64
7.3.1 The .RDTIME UUO 64
7.3.2 The .RTIME UUO 64
7.3.3 The .RDATE UUO 65
7.4 The MAR Feature 65
7.4.1 The .WMAR UUO 66
7.5 The System Job 67

7.5.1 The .SUPSET UUO 68
7.5.2 The System Going Down Feature 68
7.6 Examining ITS 69
7.6.1 The .GETSYS UUO 69
7.6.2 The .GETLOC UUO 70
7.6.3 The .RSYSI UUO 70
7.7 The One Proceed Feature 70
7.8 Miscellaneous 71
7.8.1 The .SLEEP UUO 71
7.8.2 The .GENSYM UUO 71
7.8.3 The .IOTLSR UUO 72
7.8.4 The .SETLOC UUO 72

References 73

A. System Calls in Numeric Order 74

B. Available Symbolic Devices in Alphabetic Order 76

C. System Variables per User 77

D. Further Sources of Information on ITS 79

0. Notation Used in this Manual

0.1 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. Numbers that are written alphabetically are decimal.

0.2 Character Objects and Strings

Character objects and strings are usually surrounded by quotation marks (""") and are relatively clear from context. Non-graphic characters created by holding down the control shift of a teletype and striking a graphic are represented by the graphic preceeded by a "^" as in ^A for "control-A".

0.3 Bit Positions and Intervals

Bit positions in thirty-six bit words 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.

0.4 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. Other internal references are written out in less formality and forms whose meaning should be evident, such as the following, are allowed: [Sec x, y], [App x; Ref x, y], etc.

1. Introduction to and Goals of ITS

The Incompatible Time Sharing system (ITS) is a control program tailored to the Project MAC Artificial Intelligence (AI) Group PDP-6 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) Each user has better control over hardware and greater ease in debugging software due to appropriate designed-in features.

(3) Jobs which would otherwise require excessive blocks of time can be performed automatically in small parts over long periods of time when the computer would otherwise be idle.

(4) Programs can more easily be made independent of specific input and output devices.

It was desired to achieve these goals without the usual penalties of time sharing systems such as rancid response and massive inefficiency and their attendant evils such as required break characters, arbitrary built in command levels of dubious virtue, and severe core restrictions due to a bloated monitor.

For these and other reasons ITS was 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. (The coming installation of additional disk storage will enable the swapping of programs but it is expected that only dormant programs will be swapped out.)

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

(4) ITS attempts to provide simple standardized input-output and other system calls as well as 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 beyond or more recent than this manual consult the sources listed in Appendix D.

1.1 Preview of Section 2

The user of ITS normally commands a hierarchy of programs in machine storage organized in an inverted tree. The top procedure is initially loaded for him by the system when he types a ^Z on an idle console. All programs may have inferior procedures which they control. Almost all commands to the system are machine instructions [App A; Ref 1], which trap to the monitor, executed for the user by his programs. The only exception is that he may interrupt the superior procedure of the program with which he is conversing by typing ^Z. 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.

1.2 Preview of Sections 3 and 4

The majority of the code for 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 are available where meaningful, and a continuum to the specialized system calls necessary to make nearly full use of some specialized devices exists. Most devices are referenced symbolically [App B] 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 or the DEC 340 display.

1.3 Preview of Section 5

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 hundred 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.

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 C] used by the scheduler and other parts of ITS are kept in system memory and do not impinge on the user's core image.

1.4 Preview of Section 6

User programs operating within the environment provided by ITS are organized (as mentioned in 1.2) 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.

1.5 Preview of Section 7

A large number of miscellaneous system calls and features exist for a variety of purposes. Many 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. 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 checks constant portions of ITS against a copy in an attempt to detect some forms of system or hardware failure.

2. Getting the Attention of and Transmitting Commands to ITS

2.1 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 (a member of the class of devices that may be consoles, currently either a GE terminal or a KSR 35/37) or from an active console, that is a teletype in control of a procedure tree. Teletypes may also be in use as ordinary devices (as Tnm instead of TTY [Sec 4.5]) in which case ITS ignores ^Z's typed on them.

If the teletype was idle, ^Z normally causes the teletype to become a console and loads and starts as its top level procedure [Sec 6.1] the dump file named "@ HACTRN" on device SYS, or if not found there from device UT2 (see section 3.1 for a discussion of file and device names). The JNAME of a thus initiated procedure is HTN and its UNAME and SNAME are set to the value minus one [Sec 3.8.4, 6.1]. This will not happen if there is insufficient memory available, if the system is known to be going down soon [Sec 7.5.2], 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.

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 4.5.1]. 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 5.2], 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 [Sec 5.2] set will be taken in less than one quantum time when the scheduler [Sec 6.5] next runs.

2.2 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 2.2.1 and listed in appendix A. Section 2.2.2 concerns all other trapping instructions. It is a useful characteristic of the PDP-6 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 mentioned in section 5.2.

2.2.1 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, 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 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 5.2], is turned on.

A complete list of system calls in numeric order appears in appendix A. 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 3.3].
041	.OPEN	Used to initialize input-output between a device and a program [Sec 3.2]:
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 6.2.3].
045	.BREAK	Executed by a procedure to set the procedure's .BREAK interrupt bit, a class one interrupt [Sec 5.2], thus stopping it and interrupting its superior porcedure [Sec 6.3.2].

046 .STATUS Used to ascertain the status of an input-output device, channel, or transfer [Sec 3.5].
047 .ACCESS Used in input-output to randomly addressable devices [Sec 3.8.5].

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

2.2.2 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, see section 3.2). 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 7.8.3] modes except that it may turn off iot-user mode.

3. The General Transaction of Input-Output

3.1 Philosophy and Organization

Standard ITS input-output has been organized with such goals as symbolic device independence, no required buffering in the user's core image, simple unit (character, word, vidisector point, etc.) mode transfers available along with block at a time transfers which reduce overhead by reducing the number of system calls. Each procedure has associated with it 20 input-output channels (but see sections 3.7, and 4.12). Each of these transfers in one direction and mode between the program and a file on a device with which the channel was associated by the execution of a .OPEN [Sec 3.2]. Items are then transmitted over the channel by the execution of .IOT's [Sec 3.4]. This association may be terminated by a .CLOSE [Sec 3.9.1] or another .OPEN on the same channel.

All data transfers directly to or from symbolic ITS devices [App B] (except VID, NVD, and some modes of IMX) are effected to or from buffers in system memory. In the case of major file devices (UTn and DDn) these buffers are dynamically allocated from free memory by ITS [Sec 5.3]. The additional overhead required to transmit data between a user's core image and the system's buffer as compared with a system having direct 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 3.4] 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 core storage usage 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 7. Miscellaneous calls related to particular symbolic devices are, however, included under the device in section 4.

3.2 The .OPEN UUO

Input-output on a particular channel is initialized by executing a .OPEN. The channel is numerically specified by the accumulator field of the .OPEN and any previous transfer of that channel is terminated as if by a .CLOSE [Sec 3.8.1]. If the user is interrupted [Sec 5.2] 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. As a convention, the file names are usually thought of as up to six, left justified, sixbit characters rather than thirty-six bit quantities. Zero file names are not allowed on many devices and attempts to use them will cause the .OPEN to fail. The first word has the three character device name in its right half and the direction and mode in its left.

Immediatly below are the standard mode bits from this left half.

Bit(s)	Meaning
4.9	1=>Delete or rename [Sec 3.5] rest of bits ignored. 0=>Open.
4.8-3.4	Device dependent.
3.3	1=>Image mode. 0=>ASCII 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 3.3] 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 the reason for the failure may be ascertained with a .STATUS [Sec 3.6]. Any .OPEN closes previously opened devices on that channel whether or not the .OPEN is successful.

Files may be deleted and renamed with a .OPEN but this is usually done with a .FDELE as explained in section 3.5.

3.2.1 File Directories

For devices with true file structure (UTn, DDn, CLU, CLO, COM, and SYS), a file directory of the device may be read by trying to input the file ".FILE. (DIR)". If character mode is used, a readable file directory terminated by a ^L and ^C results while image mode yields an actual binary of the directory which is device dependent. Both block and unit modes are available. For other devices character input of this file name will yield the string "NON-DIRECTORY DEVICE". Attempted binary input will fail (the .OPEN will not skip).

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. An "!" by a file name indicates that it relates to a person at that time logged in.

3.3 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, and (5) the from and to device and pair of file names, where "*"s in from positions match any device or file name and "*"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 and no further attempt will be made to translate.

3.3.1 The .TRANS UUO

This system call (.CALL 2,) is expected to point to a three word block in the same manner as a .OPEN [Sec 3.2]. 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.

3.3.2 The .TRANAD UUO

This system call (.CALL 3,) makes new entries in the translation table. It is assumed to point at an eight word block as follows:

```
UNAME   ;must be the UNAME of procedure making and subject to entry
JNAME   ;JNAME for procedure subject to entry
ATMIO,,DEV1 ;4.9 = 0 => retranslate, 1 => atomic
        ;3.2 = 1 => output, 3.1 = 1 => input
        ;DEV1 = from device
FN11    ;from file names
FN12
,,DEV2  ;to device
FN21    ;to file names
FN22
```

Before trying to insert a new entry in the translation table, this system call simulates a .TRNDL pointing at the same block. If a .TRANAD is successful in creating an entry it skips.

3.3.3 The .TRNDL UUO

This system call (.CALL 7,) 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. It skips if successful in deleting an entry. 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 .TRNDL 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.

3.4 The .IOT UUO

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 that has been set up for transfer in a particular mode between the procedure and a particular device by a successful .OPEN [Sec 3.2]. If the channel has not been .OPEN'ed the procedure will receive an input-output channel error interrupt (a class two interrupt [Sec 5.2]). 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 effective address points to a word to be written from or 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 3.4.1) 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. 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 (namely Tnm, TTY, COD, LPT, PTP, PLT, and PTR) 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 3.4.1].

3.4.1 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 3.8.1] and further .IOT's will cause input-output channel errors (class two interrupts [Sec 5.2]).

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 4.1] 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 UUO (.OPER 50). This channel number will be replaced with the end of file character for that channel by the execution of the .EOFC.

3.4.2 The Device Full Condition

Attempted output on a full file structured device (UTn, DDn) results in an input-output channel error (a class two interrupt [Sec 5.2]) 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 procedures interrupt routine or some other procedure deletes material from the full device.

3.5 The .FDELE UUO

The system call .FDELE (.CALL 11,) 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:

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

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

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

A .FDELE is internally treated as a .OPEN pointing at a block with the sign bit on in the first word [Sec 3.2] except that it does not also result in a .CLOSE being executed on a channel. A .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 on channel zero [Sec 3.6].

3.6 The .STATUS UUO

The accumulator field of this system call is the number of an input-output channel whose status word replaces the contents of the effective address 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 chanel 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.
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 4.11] 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	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 (XSR 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.
21	NUL	Null device.
41	UTn	DEC tape.
42	DDn	Data-disk.
60	USR	A not immediately inferior procedure.
61	USR	An inferior procedure.
62	CLU	Core link use device.
63	CLO	Core link open device.
64	-	The directory device [Sec 3.2.1].

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).

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
4	Non-existant sub-device (such as IMX channel [Sec 4.9]).
5	Over .IOPOP [Sec 3.7.2].
6	Over .IOPUSH [Sec 3.7.2].
7	Channel does not have a procedure open on it.
8	Channel not open.
9	Device full.

3.7 The Input-Output Channel Push Down Facility

For greater flexibility in input-output than that available with the basic input-output channel system [Sec 3.1], a facility is provided whereby a limited number of input-ouput transactions may be stored for later resumption. Meanwhile the channel they were being effected on may be otherwise utilized.

3.7.1 The .IOPUSH and .IOPOP UUO's

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 into a channel (possibly different). The error status and .ACCESS [Sec 3.8.5] pointer for a channel are also correctly stored and restored. Executing a .IOPOP into a channel first closes the channel. Each procedure's input-output channel push down list [App C] has space for eight entries.

3.7.2 The .IOPDL UUO

This system call (.OPER 57) is intended to reset a procedures 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 3.7.1] equal to the number of entries in the procedure's input-output push down list and which .IOPOP each enty back into the channel from which it was .IOPUSH'ed.

3.8 Miscellaneous Uniform Input-Output Related System Calls

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

3.8.1 The .CLOSE UUO

This system call (.OPER 7) closes the input-output channel whose number is in its accumulator field. 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.

3.8.2 The .RESET UUO

This system call (.OPER 37) is intended to reset system buffer pointers so that buffered data on an input-output channel (specified by the accumulator field) will be ignored. It has not yet been implemented for DEC tape or disk transfers and is also ignored by devices for which it is not meaningful (IMX, OMX, VID, and NVD).

3.8.3 The .ITYIC UUO

It is frequently desirable to examine the characters of an incoming stream at the interrupt [Sec 5.2] level of a procedure (for "quit" features, etc.). This system call (.OPER 60) provides the facility to read these characters from channels on which devices that will provide appropriate interrupts (TTY, Tnm, CLU, CLO) are open. The accumulator reffered to by an .ITYIC should have an appropriate channel number placed in it. The execution of an .ITYIC will then replace this number with the character read and skip. If no character is available the .ITYIC will not skip and if an inproper channel is specified the procedure will receive an illegal instruction interrupt.

3.8.4 The .WSNAME UUO

On some devices (namely DDn, CLO and CLU) files are actually identified by three file names. Two of these are specified in the normal way [Sec 3.2] and the third is derived from the current system name [App C] associated with the procedure referencing the file. If the file is being written this will be written as its third file name and if being read it must be the same as the third file name. Each procedure's system name is set to the UNAME of the superior procedure that creates it at the time of its creation [Sec 6.1.1]. .WSNAME (.OPER 35) sets a procedure's current system name from the accumulator specified except that if that accumulator contains zero the procedure's SNAME will be unchanged. A procedure may read its SNAME with a .SUSET [Sec 5.5].

3.8.5 The .ACCESS UUO

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 in its accumulator field. Currently this pointer influences only the USR device [Sec 4.12].

4. 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, unit/block, and input/output modes are available and that ASCII mode implies seven bit ASCII characters and image mode thirty-six bit binary words.

4.1 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 3.8.4] associated with them. Directories can be excised from core only by the .UDISMT UUO.

4.1.1 The .UDISMT UUO

This system call (.OPER 22), 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.

4.1.2 The .ASSIGN and .DESIGN UUO's

These system calls are to enable a user to protect against other users accidently referencing his DEC tape to write or delete a file. Both specify the drive numbers as the contents of their specified accumulators. 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 3.8.4] as the UNAME of the procedure that executed the .ASSIGN. For a procedure in a disowned tree [Sec 6.2.2], a .ASSIGN will be treated as an illegal instruction. A .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 4.1.1] and the .ASSIGN'ing of a drive does not protect against other users dismounting it.

4.1.3 The .UBLAT UUO

This system call (.OPER 56) 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. Instead of attempting to read in the directory from the tape mounted on the specified drive, ITS internally marks the drive so that this directory 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 4.1.1] it.

4.2 The DDn, SYS, and COM Devices

The devices DD0 and DD1 are two Data Disk M-6 disk drives. They are truly file structured with system names [Sec 3.8.4] and have all standard modes. Data is dynamically buffered by ITS [Sec 5.3] for both input and output. There file directories are read in the first time the drives are referenced and are written out whenever they have been changed and the disks are otherwise idle.

The device SYS is used for storage of systems programs and is currently those files on device DD0 with system name SYS. Reading or writing device SYS causes one to reference DD0 as though one's system name were momentarily SYS. The device COM is used for commonly used user files. In a manner similar to device SYS it is DD1 with system name COMMON.

4.3 The CLO and CLU Devices

These internal devices are used for buffered interprocedure communication. One procedure must open a file on device CLO (core link open) and the second must then open a file with the same names (including system name [Sec 3.8.4]), mode, and the opposite direction on device CLU (core link use). All standard modes are available. For character mode transfers, interrupt level processing is available [Sec 3.8.3, 5.2]. Each link has a 200 word buffer associated with it.

4.4 The NUL Device

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

4.5 The Tnm and TTY Devices

Devices T00 through T04 represent teletype lines to the PDP-6 with KSR 35/37's on the other end. Devices T15 through T20 (the two digits are treated as an octal number) are GE Datanet 760 terminals with a character only display and keyboard. (Devices T05 through T14 may soon be available for more teletype and teletype-like hardware.) 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 3.2] has special meanings.

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 T00 opened as a console (TTY) and that actually has controle of it [Sec 4.5.1] can always seize the DEC 340 display [Sec 4.11].

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 that the character whose value is 33 is printed as a "$", other characters with a value less than 40 that are not format effectors or bell are printed as an "^" followed by the character with an ASCII value 100 greater, and 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) a "_", as displayed by the GE terminal, is used instead of an "^" 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 3.2] 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 2.1], is ignored by ITS. Procedures may enable interrupts (class three) due to non-empty input buffers [Sec 5.2] and also examine the input characters at interrupt time with .ITYIC [Sec 3.8.3].

Image mode input is characterized by no ITS supplied echo and no modification of the character codes unless open mode [Sec 3.2] bit 3.5 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 effecting input are obtained from the first input .OPEN [Sec 3.2] 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 3.4] being executed terminates as on an end of file [Sec 3.4.2].

Any input or output .IOT or .OPEN executed on a TTY device while it is actually being controlled by another procedure [Sec 4.5.1] will hang until the executing procedure regains the device.

4.5.1 The .ATTY and .DTTY UUO's

These system calls (.OPER 11 and .OPER 12 respectively) 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 on which an immediate inferior procedure is open [Sec 4.12]. 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.

4.5.2 The .LISTEN UUO

This system call (.OPER 2) is usable only on the console a procedure is associated with. It will hang if control of the console [Sec 4.5.1] 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 6.2.2] zero will be returned.

4.6 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 3.2] 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. A fixed 1000 word buffer in ITS is used for output to this device.

4.7 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 4.6], 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 from right to left as follows: (1) pen up, (2) pen down, (3) plus x, (4) minus x, (5) plus y, (6) minus y.

4.8 The PTP and PTR Devices

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 4.6, 4.7] 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 3.2] 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 3.4] executed.

4.9 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 3.2] 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 multiplex channel of the same number. If it is desired to read in values converted at each .IOT [Sec 3.4] 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
0-32	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	New arm shoulder.
47	New arm shoulder.
50	New arm joint 1A.
51	New arm joint 1B.
52	New arm joint 2A.
53	New arm joint 2B.
54	New arm joint 3A.
55	New arm joint 3B.
56	New arm joint 4A.
57	New arm joint 4B.
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.
114-137	Unused.
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.
150-165	Unused.
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.

4.10 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 3.2] 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 effects 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	NVD iris.
33	NVD focus.
34	NVD mirror.
35-53	Unused.
54	New arm shoulder.
55	New arm shoulder.
56	New arm joint 1A.
57	New arm joint 1B.
60	New arm joint 2A.
61	New arm joint 2B.
62	New arm joint 3A.
63	New arm joint 3B.
64	New arm joint 4A.
65	New arm joint 4B.
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.

4.11 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 users core image [Sec 4.11.1].

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 devices 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 4.5]. 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 4.4] output. .CLOSE'ing [Sec 3.8.1] an input-output channel on which device DIS or IDS are open is the same as executing a .DCLOSE [Sec 4.11.3].

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 3.2] 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 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". Almost all error conditions encountered during this "execution" cause interrupts to the user [Sec 5.2]. 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 the .IOT'ing [Sec 3.4] of each starting location to the IDS device is normally terminated 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	See below.	2.3-2.9 parameter to set from 4.9-3.1.
5	PUSH.	4.9-3.1 address.
6	POP.	4.9-3.1 address.
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.

4.11.1 The .DSTART and .DSTRTL UUO's

These UUO's (.CALL 10, and .CALL 12, respectively) 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 3.4].

4.11.2 The .LTPEN UUO

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 5.2].

This system call (.CALL 14,) 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 of 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 .LPEN, (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.

4.11.3 The .DCLOSE UUO

This system call (.OPER 15) 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.

4.11.4 The .DSTOP UUO

This system call (.OPER 16) 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.

4.12 The USR Device

The USR device actually represents procedures. A successful .OPEN [Sec 3.2] executed on device USR will either create a new procedure [Sec 6.1.1], attach a disowned procedure tree [Sec 6.1.2], or simply associate with the input-output channel on which it was executed an existant procedure. 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 6.2.1] but will be entirely unaffected by a .CLOSE [Sec 3.7.1] 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 6.1.3] is assigned by ITS to a procedure. The number of inferior procedures a procedure may have is currently limited to eight because that is the allowed number of unique inferior procedure interrupt bits [Sec 5.2].

The 3.3 mode bit [Sec 3.2] is ignored in .OPEN's on device USR but both block and unit modes are avaiable. 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 3.8.5] 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 5.3] will result in a class two interrupt [Sec 5.2] but in a similar attempt to write (possible only for an immediate inferior) ITS will attempt to extend the procedures core image and an interrupt will result only if more core is unobtainable.

Executing a .RESET [Sec 3.8.2] 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.

4.13 The VID and NVD Devices

These devices are somewhat similar to the IMX device [Sec 4.9] 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 3.2] 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 3.2] 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 deflection signals for the VID device come from the DEC 340 display [Sec 4.11] whose use will be degraded by using device VID.

4.14 The COD Device

This is a charcter 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.

5. The Procedure

5.1 Philosophy and Organization

The use of ITS is entirely procedure oriented. Almost all system actions are caused by trapping instructions [Sec 2.2; App A] executed by user programs that are hierarchially organized [Sec 6.1]. A large number of variables [App C; Sec 5.5] are 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 or time or until they become unrunnable (whichever comes sooner) in an order determined by the scheduler [Sec 5.4]. The scheduler also handles software interrupts to the user [Sec 5.2, 5.4]. When in execution, user programs run with the PDP-6 in user mode [Sec 2.2.2} which enables protection and relocation hardware that ITS sets in acordance with the length [Sec 5.3] 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.

5.2 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 5.2.1] and may also re-enable interrupts without immediately or ever returning to the location his program was interrupted from.

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

The particular bit used in an immediately superior procedure's second interrupt request word to interrupt it when a fatal error occurs in its inferior is the bit assigned by ITS to that inferior at the time it was created

[Sec 6.1.3].

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 HACTRN will be loaded (as in section 2.1) and all inferior procedures will be lost. If the procedure tops a disowned tree [Sec 6.2.2] it will be immediately stopped and when next attached [Sec 6.1.2] by a job tree the attaching procedure will be appropriately interrupted.

ITS determines that it should actually interrupt a procedure at schedule [Sec 5.4], .DISMISS [Sec 5.2.2], or .UPISET [Sec 5.2.3] time if there are corresponding bits on in one or both of the procedure's interrupt request and mask words. ITS examines the contents of the procedure's location 42 to see if its direct address (indexing and indirect addressing ignored) is between 20 and 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 the second word was 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 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 5.2.2].

The following is a list of word one interrupt bits:

Bit(s)	Class	Condition
1.1	3	Character typed at console [Sec 4.5.1].
1.2	1	^Z typed [Sec 2.1].
1.3	1	Bad location 42.
1.4	3	AR overflow [Ref 1].
1.5	2	Display list memory protection violation [Sec 4.11].
1.6	2	Illegal instruction.
1.7	3	System going down in 5 minutes [Sec 7.5.2].
1.8	1	.VALUE executed [Sec 6.3.1]. (Note that a value retrieving .USET is available [Sec 6.2.3].)
1.9	2	Input-output channel error. For more exact cause of interrupt use the .STATUS UUO [Sec 3.6].
2.1	2	Memory protection violation on reference to inferior procedure [Sec 4.12].
2.2	1	.BREAK executed [Sec 6.3.2].
2.3	1	Return from single instruction proceed [Sec 7.7].
2.4	3	Slow clock (every half second).
2.5	2	Memory protection violation [Ref 1].
2.6	2	MAR interrupt [Sec 7.4].
2.7	3	Light pen [Sec 4.11].
2.8	3	PDL overflow [Ref 1].
2.9-4.8	-	Not used.
4.9	-	Indicates rest of bits to be interprited as below.

In the second word of interrupt requests, bits 3.1-3.8 represent inferior procedures [Sec 6.1.3] and bits 1.1-2.7 represents input-output channels [Sec 3.1] 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 4.5] input or character mode core link [Sec 4.3] input on the channel. These buffered characters may be examined at the interrupt level with .ITYIC [Sec 3.8.3].

5.2.1 The .SETMSK and .SETM2 UUO's

These system calls (.OPER 4 and .OPER 5 respectively) provide the mechanism by which the user can set his interrupt enable mask words. These mask words contain zero whenever a procedure is created or has a .RESET executed on it [Sec 3.8.2, 4.12]. Since the second mask word will not be useful to many simple programs, the first of these calls is provided to set only the first word form the accumulator it specifies. The second will set both mask words from the specified accumulator and the following location.

A procedure may examine its interrupt mask with the .SUSET UUO [Sec 5.5]. In general a .SUSET will not retrieve what a procedure attemptes to set in its first mask word as bits 1.2, 1.3, 1.8, 2.2, and 2.3 (the class one interrupts) will not appear set.

5.2.2 The .DISMISS UUO

This system call (.CALL 1,) provides the normal means to return to an interrupted routine. It simultaneously re-enables interrupts and transfers to the location directly addressed by the contents of its effective address which 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 woris he will be immediately re-interrupted.

5.2.3 The .UPISET UUO

The user may arbitrarily inhibit or enable all interrupts to his program by means of this system call (.OPER 36) which stores the contents of the accumulator specified into his interrupt level status flag. The zero state inhibits interrupts while the minus one state enables them. If there are pending conditions in the user's interrupt request word when he enables interrupt he will be interrupted immediately. The user may examine his interrupt level flag by means of the .SUSET UUO [Sec 5.5].

5.3 The Core Allocator

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 pageing 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 7.5]), (2) a disk buffer or directory [Sec 4.2], (3) a display buffer [Sec 4.11], (4) unused, or (5) further subdivided into 200 word blocks. These 200 word blocks are used for core link device buffers [Sec 4.3] or for DEC tape buffers or directories [Sec 4.1].

A procedure may request more or less core by means of the .CORE UUO [Sec 5.3.1] and determine its memory bound without making protection violations with the .SUSET UUO [Sec 5.5]. 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 6.2.3] or .IOT on device USR [Sec 4.12] executed by its immediate superior.

5.3.1 The .CORE UUO

This system call (.CALL 6,) 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.

5.4 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 C] in the system for each procedure. It simultaneously performs those modifications of these variables necessary to provide the simulated interrupt feature [Sec 5.2].

For those procedures that have their stop word [App C] zero, their runnability is determined by a variable known as the flush instruction 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 C] loaded into ITS symbolic accumulator T. If the instruction skips then the procedure is runable 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 or its console tree has used more but the procedure less than one quarter as much as the best so far.

All disowned 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 7.5] is also given lower than normal priority.

5.5 The .SUSET UUO

Procedures may examine some of their system variables [App C] by means of this system call (.CALL 13,). It is essentially the same as the .USET UUO [Sec 6.2.3] except that no input-output channel is specified, since it is understood to refer to the variables of the procedure executing it. Also variables may only be examined and then only if the 1.3 bit is on in the variables mode

6. Procedure Interaction

6.1 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 6.2.2].

Console "controlled" trees have had their top level procedure automatically loaded by ITS [Sec 2.1]. 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 2.1], 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 [Ref 2, 8, 10]. The only restrictions deliberately inserted in it relate to logging in [Sec 7.1].

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 5.2]; (3) they are scheduled with lower priority [Sec 5.4]; (4) certain system calls [Sec 4.1.2, 6.1.2, 6.2.2, 7.8.3, 7.8.4], when executed in them, are treated as illegal instructions.

There exist a variety of means for procedures immediately above and below each other to interact [Sec 6.2, 6.3]. Procedures may also effect any procedure beneath them in the procedure tree by means of input-output translation table entries [Sec 3.3] and an arbitrary pair of procedures may converse treating each other as input-output devices [Sec 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.

6.1.1 Procedure Creation

Procedures may be created only by typing ^Z on an idle teletype [Sec 2.1] or by executing a .OPEN [Sec 3.2] on device USR [Sec 4.12]. In the second case, which is discussed here, there are strict requirments 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, some as follows: its interrupt mask words will be zero and its interrupt level status word will be minus one [Sec 5.2]; its system name [Sec 3.8.4] will be set to its UNAME; its program counter (PC) will be set to zero; its stop word [App C] will have the user control bit on.

6.1.2 Attaching Disowned Procedure Trees

The only case in which a successful .OPEN [Sec 3.2] on device USR [Sec 4.12] 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 is 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 ITS checks only to see that the UNAME of the opening procedure and the JNAME of the procedure being opened will form a unique pair. If so and the opening procedure has less than eight inferiors the open will succeed and ITS will modify the UNAME of the opened procedure to agree with the opening procedure. The UNAME's of procedures, if any, deeper in the attached tree are not modified.

6.1.3 The .UTRAN UUO

When a procedure is first created by a .OPEN, ITS assigns to it a bit for use in interrupts [Sec 5.2] 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 6.2.3], this system call (.CALL 5,) has been provided to translate from particular interrupt bit to name of inferior. The left half of 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.

6.2 Controls over an Immediately Inferior Procedure

See also section 4.5.1.

6.2.1 The .UCLOSE UUO

Procedures may be destroyed only by the execution on a .LOGOUT [Sec 7.1] or a .UCLOSE. The second of these two system calls (.OPER 10), which is discussed here, destroys the inferior procedure open on the channel specified by its accumulator field and the entire procedure subtree of which this inferior is the top procedure. .CLOSE's [Sec 3.7.1] are automatically effected on all channels of all the destroyed procedures and if any one of them is running the DEC 340 display [Sec 4.11] a .DCLOSE will be executed for it. In order to kill certain cancerous types of procedure subtrees it has been found necessary to so implement this system call such that it first stops all procedures in the subtree and then destroys them.

6.2.2 The .DISOWN UUO

This system call (.OPER 27) requires that an inferior procedure be open on the channel specified by its accumulator field. It causes this open procedure to become the top procedure of a disowned tree 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 5.2] as will a procedure trying to disown a subtree that controls the tree's console [Sec 4.5.1].

6.2.3 The .USET UUO

In order to allow a procedure to set and read many of the system variables [App C] associated with its inferiors, this system call was created. When executed its accumulator field must contain the number of a channel on which an inferior procedure is open. The contents of its effective address is viewed as two half words with the right half pointing to the location from which the system variable is to be set or into which the system variable is to be read. The left half specifies by its top bit (4.9) whether the system variable is to be read (zero) or written (one) and by the value of its remaining bits which system variable. The following table lists permissable values:

Name	Value	Type	Variable
UPC	0	7	Program counter word.
VALUE	1	7	Contents of effective address of most recent .VALUE [Sec 6.3.1].
TTY	2	1	Status relative to console [Sec 4.5].
FLSINS	3	1	Blocking condition [Sec 5.4].
UNAME	4	5	UNAME [Sec 4.12].
JNAME	5	7	JNAME [Sec 4.12].
MASK	6	7	Interrupt mask word [Sec 5.2].
USTP	7	3	Stop word (only bit 4.7 can be written).
PIRQC	10	7	Interrupt request word [Sec 5.2].
INTB	11	1	Bit used to interrupt superior [Sec 6.1.3].
MEMT	12	7	Memory bound [Sec 5.3], first location it would be illegal to reference.
SV40	13	5	Last executed UUO trapping to system.
IPIRQ	14	7	Interrupt request word (written by an IORM).
APIRQ	15	7	Interrupt request word (written by an ANDCAM).
SNAME	16	7	System name [Sec 3.7.4].
PICLR	17	7	Interrupt level status word [Sec 5.2].
MARA	20	7	MAR register and control bits [Sec 7.4.2].
MARPC	21	7	Program counter at last MAR interrupt [Sec 7.4.1].
UUOH	22	5	Program counter word at last system UUO.
UIND	23	1	User index.
RUNT	24	5	Total run time in 4.069 microsecond units.
MSK2	25	7	Interrupt mask word two [Sec 5.2].
IFPIR	26	7	Interrupt request word two [Ref 5.2].
	27-77	0	Unused.

IOC	100+N	1	Input-output channel word for channel N.
IOS	120+N	1	Status word for input-output channel N.
IOP	140+N	1	Nth word of input-output channel push down list.

In the "type" column above, bit 1.1 on indicates that the variable may be read, bit 1.2 that it may be written, and bit 1.3 that it may be read with as .SUSET [Sec 5.5].

6.3 Communication to an Immediately Superior Procedure

6.3.1 The .VALUE UUO

A procedure executing this system call (.CALL 4,) receives a class one interrupt [Sec 5.2] stopping it and interrupting its superior procedure. The contents of the effective address is saved in a system variable associated with the procedure [App C] so its superior can easily retreive it with a .USET [Sec 6.2.3].

6.3.2 The .BREAK UUO

The accumulator field and effective address of this UUO are ignored by ITS. A procedure executing it receives a class one interrupt [Sec 5.2] stopping it and interrupting its superior procedure.

7. Miscellaneous System Calls and Features

7.1 The .LOGIN and .LOGOUT UUO's

These two system calls (.OPER 6 and .OPER 33) are the only ones that are effective at only the 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 6.1.1] and destroy [Sec 6.2.1] 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 2.1] and .LOGOUT.

.LOGIN is not restricted to top level procedures by ITS but rather by HACTRN, the top level procedure ITS automatically loads [Sec 2.1]. A .LOGIN represents an attempt by a procedure to set its UNAME to the contents of the accumultor 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 nor minus one. 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.

7.2 Nonstandard Devices

7.2.1 The .RDSW UUO

This system call (.OPER 20) reads (DATAI's [Ref 1]) the contents of the data switches on the PDP-6's console into the specified accumulator.

7.2.2 The .RD760 and .WR760 UUO's

These system calls (.OPER 30 and .OPER 31) respectively read the control register (CONI) of the device whose PDP-6 device number is 760 into the specified accumulator and output (CONO) all but the bottom three bits from the specified accumulator to the control register of the device [Ref 1]. The bits in the control register of this device are usually used for termporary direct control or sense hookups.

7.2.3 The .RD710 UUO

The real time clock which ITS currently uses to time user quantums has, as well as an interval timing counter, a clock counter which is neither set or stopped by ITS. This system call (.OPER 47) reads into the specified accumulator a word whole bottom twenty four bits are this clock counter. It is incremented once every 4.069 microseconds.

7.2.4 The .ORGI and .ORGO UUO's

These system calls (.OPER 54 and .OPER 55) input or output from their specfified accumulator and the two following locations from and to what will be an electric organ keyboard and tone generators.

7.3 Various Times and the Date

7.3.1 The .RDTIME UUO

This sytem call (.OPER 17) 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.

7.3.2 The .RTIME UUO

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 (.OPER 45) 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.

7.3.3 The .RDATE UUO

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 (.OPER 46) 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 7.3.2] has been determined. The date is correctly incremented every midnight.

7.4 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 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 5.2] and the .USET's [Sec 6.2.3] corresponding to the following UUO is used more frequently than it is.

When an MAR interrupt occurs, further MAR interrupts are disabled until the MAR is again set.

7.4.1 The .WMAR UUO

This system call (.OPER 41) takes the right half of the specified accumulator as the new address to be put in the MAR register and the low three bits of the left half as conrol 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.

A .SUSET [Sec 5.5] 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 5.2} will be the location in his core image of the offending system call. The word read by a .SUSET will be an executive program counter word showing the absolute location at which the system call attempted to reference the user's core image.

7.5 The System Job

There are various tasks ITS would like to perform, most of which include input-output, which would be inconvenient to implement directly due to the procedure originated orientation of almost all system rountines. Because of this a procedure was synthesized that is actually a part of the monitor and always runs in executive mode. Known as the system job, it is scheduled [Sec 5.4] with lower priority than other procedures. 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 7.5.1]. The following are some of the tasks the system job performs:

(1) Does .CORE's [Sec 5.3.1] 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 [Sec 7.8.4], .IOTLSR [Sec 8.8.3], or .OPEN [Sec 3.2} on device SYS [Sec 4.2] for output 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, login, or logout occurs.

(4) Initially determines the 24 hour time and calendar date [Sec 7.3.2, 7.3.3].

(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 explained in section 7.5.2 below.

7.5.1 The .SUPSET UUO

The system job is partially controlled by ITS through a word in system memory. Bits in the left half of this word represent requests for some transient action. In the right half only the bottom bit (1.1) is currently used and controls the system checking feature. If this bit is on, the system job makes a copy of the constant parts of ITS and procedes 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.

The .SUPSET system call (.OPER 53) enables the user to control the system checking bit by XOR'ing the contents of the right half of the specified accumulator with the system job control word and returning the resulting value in the specified accumulator.

7.5.2 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 7.8.4]). After this is first noticed, all procedures for which it is enabled will receive a system going down interrupt [Sec 5.2] and all free teletypes have an appropriate message typed on them. At this time a five minute timer is started and further ^Z's on idle teletypes ignored [Sec 2.1]. At the end of the five minute period or when the number of console controlled trees in which successful .LOGIN's [Sec 7.1] have been executed goes to zero, all trees are forceably logged out and a message typed on all teletypes. Then a program is loaded from device SYS and started such that if the address switches on the PDP-6 console are set to zero and the amplifier and loudspeakers attached to the low order bits of the memory indicator turned on, a rendition of the United States national anthem will be heard.

7.6 Examining ITS

7.6.1 The .GETSYS UUO

This sytem call (.OPER 23) 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 3.4]. The location following the specified accumulator contains, in left justified sixbit, the area requested as follows:

Sixbit	Area
TRANS	Translation table [Sec 3.3].
GETS	List of things which can be requested with .GETSYS.
DEVS	List of available symbloic input-output devices [App B].
MEMORY	Memory allocation tables [Sec 5.3].
UTAPE	DEC tape routine variables [Sec 4.1].
DDISK	Disk routine variables [Sec 4.2].
CLINK	Core link variables [Sec 4.3].
CALLS	List of symbols in squoze for system calls [App A] and .USET [Sec 6.2.3] left halves and their values.
DSYMS	Symbol table for ITS.
IMPX	Area input multiplexor cycled into [Sec 4.9].
OMPX	Area output multiplexor cycled from [Sec 4.10].
USERS	User variables for all procedures.
USER	User variables for particular user specified by UNAME and JNAME in locations two and three greater than the specified accumulator [App C]. 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.

7.6.2 The .GETLOC UUO

This system call (.OPER 25) simply transfers the contents of the absolute location pointer 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 users memory bound he will receive an illegal instruction interrupt [Sec 5.2].

7.6.3 The .RSYSI UUO

This sytem call (.OPER 52) 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.

7.7 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 procedures by .USET'ing [Sec 6.2.3] 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 users status, including this mode, will be restored by a normal .DISMISS [Sec 5.2.2].

7.8 Miscellaneous

7.8.1 The .SLEEP UUO

This system call (.OPER 3) allows a procedure to stop running, while remaining interruptable, by either specifying a length of time to remain quescent 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 7.3.1]) 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.

7.8.2 The .GENSYM UUO

This system call (.OPER 32) 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.

7.8.3 The .IOTLSR UUO

Procedures run by ITS in IOT-user mode may execute hardware input-output instructions which would otherwise by ineffective and trap to the monitor. This system call (.OPER 51) set the IOT-user status of the procedure executing it from the sign bit of the specified accumulator. If this bit is a one and the procedure is not currently in IOT-user mode it will enter it and a message will be output by the system job [Sec 7.5] unless the procedure is in a disowned tree in which case it will receive an illegal instruction interrupt [Sec 5.2].

7.8.4 The .SETLOC UUO

This system call (.OPER 26) 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 7.5] unless the procedure executing it is in a disowned tree in which case the procedure will receive an illegal instruction interrupt [Sec 5.2].

References

1. Digital Equipment Corporation
Programmed-Data-Processor-6 Handbook (F-65)
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. Thomas F. Knight
Modifications to the PDP-6 Teletype Logic (AI-105)
August 1966, MAC AI group memo.

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.

Appendix
A. System Calls in Numeric Order
(correct as of ITS version 350)

Name	Number	Section
.IOT	UUO 040	3.4
.OPEN	UUO 041	3.2
.OPER	UUO 042	2.2.1
.LISTEN	.OPER 2	4.5.2
.SLEEP	.OPER 3	7.8.1
.SETMSK	.OPER 4	5.2.1
.SETM2	.OPER 5	5.2.1
.LOGIN	.OPER 6	7.1
.CLOSE	.OPER 7	3.7.1
.UCLOSE	.OPER 10	6.2.1
.ATTY	.OPER 11	4.5.1
.DTTY	.OPER 12	4.5.1
.IOPUSH	.OPER 13	3.7.2
.IOPOP	.OPER 14	3.7.2
.DCLOSE	.OPER 15	4.11.3
.DSTOP	.OPER 16	4.11.4
.RDTIME	.OPER 17	7.3.1
.RDSW	.OPER 20	7.2.1
.UDISMT	.OPER 22	4.1.1
.GETSYS	.OPER 23	7.6.1
.GETLOC	.OPER 25	7.6.2
.SETLOC	.OPER 26	7.8.4
.DISOWN	.OPER 27	6.2.2
.RD760	.OPER 30	7.2.2
.WR760	.OPER 31	7.2.2
.GENSYM	.OPER 32	7.8.2
.LOGOUT	.OPER 33	7.1
.WSNAME	.OPER 35	3.8.4
.UPISET	.OPER 36	5.2.3
.RESET	.OPER 37	3.7.3
.WMAR	.OPER 41	7.4.2
.ASSIGN	.OPER 43	4.1.2
.DESIGN	.OPER 44	4.1.2
.RTIME	.OPER 45	7.3.2
.RDATE	.OPER 46	7.3.3
.RD710	.OPER 47	7.2.3
.EOFC	.OPER 50	3.4.1
.IOTLSR	.OPER 51	7.8.3
.RSYSI	.OPER 52	7.6.3
.SUPSET	.OPER 53	7.5.1
.ORGO	.OPER 54	7.2.4
.ORGI	.OPER 55	7.2.4
.UBLAT	.OPER 56	4.1.3
.IOPDL	.OPER 57	3.7.2
.ITYIC	.OPER 60	3.8.3
.MASTER	.OPER 61	App D
.DIAL	.OPER 63	App D

.DIALW	.OPER 64	App D
.HANGUP	.OPER 65	App D
.CALL	UUO 043	2.2.1
.DISMIS	.CALL 1,	5.2.2
.TRANS	.CALL 2,	3.3.1
.TRANAD	.CALL 3,	3.3.2
.VALUE	.CALL 4,	6.3.1
.UTRAN	.CALL 5,	6.1.1
.CORE	.CALL 6,	5.3.1
.TRANDL	.CALL 7,	3.3.3
.DSTART	.CALL 10,	4.11.1
.FDELE	.CALL 11,	3.5
.DSTRTL	.CALL 12,	4.11.1
.SUSET	.CALL 13,	5.5
.LTPEN	.CALL 14,	4.11.2
.USET	UUO 044	6.2.3
.BREAK	UUO 045	6.3.2
.STATUS	UUO 046	3.6
.ACCESS	UUO 047	3.3.5

Appendix
B. Available Symbolic Devices in Alphabetic Order
(correct as of ITS version 350)

Device	Section	Other references
CLO	4.3	[Sec 3.2.1, 3.8.4]
CLU	4.3	[Sec 3.2.1, 3.8.4]
COD	4.14	[Sec 3.4]
COM	4.2	[Sec 3.2.1]
DDn	4.2	[Sec 3.2.1, 3.4.2, 3.8.4]
DIS	4.11	[Ref 1]
IDS	4.11	
IMX	4.9	[Sec 3.1, 3.4; Ref 5.6]
LPT	4.6	[Sec 3.4]
NUL	4.4	
NVD	4.13	[Sec 3.1, 3.4]
OMX	4.10	[Ref 5, 6]
PLT	4.7	[Sec 3.4; Ref 9]
PTP	4.8	[Sec 3.4; Ref 1]
PTR	4.8	[Sec 3.4; Ref 1]
SYS	4.2	[Sec 3.2.1, 3.4.2]
Tnm	4.5	[Sec 2.1, 3.4; Ref 1, 7]
TTY	4.5	[Sec 2.1, 3.4; Ref 1, 7]
USR	4.12	[Sec 3.8.5, 5.3] (most subsections of section 6)
UTn	4.1	[Sec 3.2.1, 3.4.2; Ref 4]
VID	4.13	[Sec 3.1, 3.4]

Appendix
C. System Variables per User
(correct as of ITS version 350)

```
IOCHNM:     BLOCK 20 ;input-output channel words
SIOCHN:     BLOCK LUIOP ;input-output channel push down list
SIOCP:      SIOCHN-1 ;pointer into input-output push down list
UPC:        0        ;stores program counter when not running
ACOS:       BLOCK 15 ;swap out accumulators
AC15S:      0        ;commonly referenced swap out accumulator
AC16S:      0        ; "
AC17S:      0        ;
UUO:
SUEXIT:     JRST 2,@UUOH ;user UUO exit instruction
SUUOH:      0        ;contents of @41 (absolute)
SAC17P:     MOVEM U, ;accumulator 17 in user's shadow memory
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
UPR:        0        ;protection and relocation registers
APRC:       APRCHN   ;cono to processor
                     ;4.9 implies user in .UCLOSE
UQUAN:      0        ;quantum time
USTP:       0        ;stop word ;zero implies runnable
PIRQC:      0        ;pending interrupts
SUPPRO:     0        ;pointer and interrupt bit to superior procedure
PICLR:      0        ;interrupt level status flag
MSKST:      0        ;interrupt enable mask
FLSINS:     0        ;blocking condition instruction
MEMTOP:     0        ;six less than first illegal location
RMEMT:      0        ;first illegal location
SV40:       0        ;last interrupting UUO
UNAME:      0        ;user name
JNAME:      0        ;job name
TTYTBL:     0        ;console assignment for this procedure
VALUE:      0        ;contents of effective address of last .VALUE
UDISOW:     -1       ;zero implies top procedure of a disowned tree
USYSNM:     0        ;current system name
USYSN1:     0        ;system name used inside disk routines
IOCHST:     BLOCK 20 ;input-output channel status words
UTRNTM:     0        ;total run time
UTRPTR:     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
RUPR:       0        ;protection and relocation registers actually used
UFINAL:     0        ;nonzero implies finalization required to interrupt
IFPIR:      0        ;second word of interupt requests
```

```
 MSKST2:        0       ;second word interrupt enable mask
USRPDL:        -LUPDL,,UPDL-1 ;push down list pointer for system calls
UPDL:          BLOCK LUPDL-2 ;push down list area for system calls
EPDL:          0       ;temporarily saves accumulator B for ACORE
EPDL2:         0       ;temporarily saves accumulator T for FLSINS
```

Appendix
D. 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. The following persons may also be helpful:

Donald E. Eastlake III

Richard D. Greenblatt

Thomas F. Knight

John T. Holloway

Stewart B. Nelson