COO-1112-343

# IOS - PDP 11/45 Formatted Input/Output

## Task Stacker and Processor*

John Koschik

Randall Laboratory of Physics
University of Michigan
Ann Arbor, Michigan 48104

## Program Abstract

IOS allows the programmer to perform formatted Input/Output at assembly language level to/from any peripheral device. It runs under DOS versions V8-08 or V9-19, reading and writing DOS compatible files. Additionally, IOS will run, with total transparency, in an environment with memory management enabled.

Minimum Hardware: 16K PDP 11/45, Keyboard Device, DISK(DK,DF,or DC), Line Frequency Clock

Source Language: MACRO-11 (3.3K Decimal Words)



*Work supported by the U.S. Atomic Energy Commission.

## DISCLAIMER

July 8, 1974

# IOS: PDP 11/45 Formatted Input/Output

## Task Stacker and Processor

John Koschik, University of Michigan

## I. Introduction:

IOS is written in MACRO-11 and is 3300 decimal words in length. It runs under DOS versions 08-08 and 09-19, requiring two software modifications:

A. The SYSMAC.SML file must contain the .FSTAK, .FREAD, and .FWRIT Macros, (see section IV).

B. An additional driver, DV.IO, must be inserted in the monitor CIL (see section IV and Appendix A).

The minimum hardware configuration necessary to support IOS is an operator keyboard, line frequency clock, disk, and sufficient memory to run the MACRO Assembler (12K for DOS V08-08, 16K for DOS V09-19).

IOS has two global entry points (IOSET, IOUSET) and is also called by three trap Macros (.FSTAK, .FREAD, .FWRIT). If memory management hardware is implemented by user software (not supported by DOS), IOS must reside in KERNEL virtual space, but the user can issue tasks to IOS from USER or SUPERVISOR virtual space. Switching the mode of CPU operation is internal and transparent (see section VII H).

## II. Purpose:

IOS supplies the MACRO-11 programmer with a FORTRAN-like input and output formatting package. It was designed to function in a real-time environment and therefore queues input/output requests, performing the formatting and peripheral handling only when the processing level interrupt can be acknowledged. It is completely compatible (in fact, dependent on) with versions 8 or 9 of the DOS monitor and only minimally degrades interrupt response (see IIIB,C and Appendix A).

## III. General Description:

IOS can be divided logically into three sections, each of which operates at a different processor level.

### A. Initialization and termination

These phases occur as a result of JSR instructions and therefore operate at the current CPU priority level. It is essential that the processor level is Zero and the CPU mode KERNEL when the statements

```
JSR   R5, IOSET    ; Initialization
```

and

```
JSR   R5, IOUSET   ; Termination
```

are made. IOSET must be called before any tasks are issued and performs the following functions:

1. Initializes IOS variables relevant to stacking tasks.

2. Executes an .INIT, .OPENO, and .SPEC to the IO driver (see Appendix A) in order to patch the monitor clock routine.

3. Initializes the programmed interrupt vector.

4. Initializes the trap vector.

IOUSET is called when no more I/O tasks are to be performed but before associated DOS link blocks have been closed and drivers released. The user does not regain control until all output tasks have been completed. Any pending input tasks are ignored (i.e. aborted). The latter decision was made because an input device may be hanging on a request the user never intends to make. Finally, the clock routine is restored to its normal state to prevent DOS from crashing upon user exit.

B. Task Stacking:

The user causes an Input (Output) task to be placed in the IOS queue by invoking the .FREAD (.FWRIT) MACRO (see IVB). This results in a trap to an internal dispatch table, simultaneously setting the processor level to "PRLVL" (currently = Four, but easily modified by re-assembling). A very simple priority scheme is used. The new task is placed last in the queue if no pending task is using the same link block. Otherwise it is placed after other tasks using that link block which had the same or higher processor levels at the time of the MACRO call. This procedure takes anywhere from 175 to 500 μsec with a typical time being 225 μsec. Before control is returned to the user, his status flag (see IVD) is cleared, and an internal flag is set to instruct the clock routine (which operates at processor level six) to schedule a programmed interrupt. No user interrupt routine operating at a processor level greater than "PRLVL" should use the IOS trap MACROS because of possible stack corruption and an immediate lowering of the processor level when the trap instruction is executed.

C. Task Processing:

Each clock tick (16.7 msec.) an IOS flag is interrogated by the clock routine. If set, a programmed interrupt request at processor level TSKLVL (currently=one) is scheduled. When this interrupt can be acknowledged, IOS checks each of its internal

buffers (currently four) to see if any tasks have been initiated, but not completed, and whose associated driver is free. If these conditions are met, another record for that task is input or output (using DOS .READ or .WRITE), and the appropriate formatting performed.

If buffers remain free, new tasks (if any) are pulled from the queue and initiated (assuming the related driver is free). Note that no more than one record is processed on any given task each clock tick.

## IV. How to Play the Game:

### A. Preliminary

1. The DOS monitor core image library must first be rebuilt to include the (supplied) IO driver. Details of modifying the monitor can be found in the BATCH-11/DOS-11 SYSTEM MANAGER'S GUIDE. Additional information on the IO driver is in Appendix A.

2. The user must link his object modules with IOS.OBJ. Before making any calls to IOS, the user program must contain the statements

```
.MCALL      .FREAD,.FWRIT,.FSTAK
.GLOBL      IOSET, IOUSET
```

The input and output operations are ultimately performed by the DOS EMT's .READ and .WRITE (Normal ASCII mode). Therefore the user must allocate link blocks and file blocks and follow all DOS rules concerning .INIT, .OPEN, .CLOSE, and .RLSE. Obviously format statements are also the user's responsibility. However line buffers and headers are imbedded in IOS and transparent to the programmer.

### B. Creating input (output) requests:

.FREAD and .FWRIT are analogous to FORTRAN

```
READ (UNIT, FORMAT) LIST
```

and

```
WRITE (UNIT, FORMAT) LIST
```

The user controls the device specification through the link block (and alternatively the DOS ASSIGN statement). The form of the .FREAD statement is

```
.FREAD  #LINK, #FMT, FLAG, 1, A, N1, B, N2,. . .
```

where

LINK = address of first word of user link block

FMT = address of first byte of user format

FLAG = user status flag

A = base address of first list vector (scaler)

N1 = length (in 16 bit words) of vector A

B = base address of second list vector

.
.
.

etc.

The code which the MACRO generates is

```
MOV         #FMT,-(SP)
MOV         #LINK,-(SP)
TRAP        ⟨↑05⟩
BR          [next user instruction]
.WORD       FLAG
.WORD       1
.WORD       A
.WORD       N1
.WORD       B
.WORD       N2
.
.
.
etc.
```

Note that a unit word count following a scaler is <u>not</u> assumed by the MACRO. Temporary RADIX control is allowed in the word count but must be enclosed in angle brackets (e.g., ⟨↑D25⟩ for 25 decimal words).

Unfortunately, implicit "DO" loops in the name list are not allowed, but parameters of the Macro expansion can be computed and modified dynamically. For example

```
          .
          .
          .
        MOV       R0,FORMAT          ;replace dummy format address
        MOV       R1,ADRS            ;variable base address
        MOV       R2,CNT             ;word count
       .FWRIT     #LINK,#FMT,FLAG,1,0,0

       FORMAT=.-↑O22
       ADRS=.-↑O4
       CNT=.-↑O2

WAIT:  TST        FLAG
       BEQ        WAIT
          .
          .
          .
```

This technique is particularly useful in a loop. Note, however that it is essential to wait for completion before altering parameters in the Macro call for a subsequent task (see VII J).

The .FWRIT statement is identical to .FREAD except

```
TRAP⟨↑03⟩
```

replaces

```
TRAP⟨↑05⟩
```

in the MACRO expansion. In both cases the user gains control at his next instruction with the stack cleared.

C. .FSTAK MACRO:

In some applications the user might need to know if I/O requests using some particular link block are pending. This is accomplished by inserting the statement

```
.FSTAK    #LINK
```

where LINK is defined as in IVB. This generates the code

```
MOV  #LINK,-(SP)
TRAP ⟨↑00⟩
```

control is returned to the user in 120 μsec (Typical) at his next instruction with the number of pending tasks using LINK at the top of the stack. The user must clear the stack.

D. User Status Flag:

Anytime an .FREAD (.FWRIT) statement contains a variable list, the first element in that list is treated as a status word. Transfer of variables to/from the list begins at the second list word. This flag is optional if no list elements (e.g., all Hollerith) appear. If the first list element is a vector, it is permissible for the flag to be the first word in that vector. It is the user's responsibility to check the value of the flag and make appropriate branches since IOS does not exit due to any user related error (unless intercepted by DOS). The user flag can take on the following values:

| | |
|---|---|
| 0: | task is pending or in progress |
| -1: | task has been successfully completed |
| +1: | error was encountered either during variable conversion (e.g., illegal character) or during the format scan. A nasty message will appear on the IOS error reporting device (default is KB:). The task is immediately aborted, possibly leaving input records for this task unread. |

↑020 task could not be stacked because of an IOS stack overflow (currently room for 20 tasks). This will be accompanied by another nasty message.

Any other positive value: When a .READ (.WRITE) is completed, the status byte is normally zero. If not, this byte is shifted left one bit and stored in the user flag (see DOS Monitor Programmer's Handbook V08-02,p.3-91). If end of medium was sensed (see VIIF), bit 7 (↑0200) will be on. The task is immediately aborted, possibly leaving records unread.

V. Formats:

Any valid FORTRAN format is acceptable. In addition a format statement currently can have up to three internal levels of nesting with any positive (<32,768) or zero repeat count. If the format is exhausted before the variable list, rescanning begins at the rightmost "(" which does not have a repeat count.

VI. Allowed Conversion Types and Conventions

A conscious effort was made to be FORTRAN compatible except to provide some useful (and easy) enhancements.

A. nIm:

Standard integer (base 10) conversion (16 bit word, 2's complement). On input the word need not be right justified (e.g., b24bbb in an I6 field produces 24, not 24000). The number -32,768 (largest negative integer) can not be input with an Im field. Read in 100000 with an Om field instead.

B. nOm:

Standard Octal conversion; (16 bit word). On input if m ≥ 6, only the rightmost six characters are used, the left-most one of which must be 0, blank, or 1. Blanks are always treated as 0's here.

C. nLm:

Long integer (32 bits) conversion. On output the (2's complement integer in two consecutive memory locations will be formatted as a signed decimal integer. The range of values is $-(2**31)+1 \leq x \leq (2**31)-1$. On input, the L format ignores all blanks in the field just as the I format does, storing the value in two 16 bit words.

D. nEm.$\ell$ or nDm.$\ell$:

Standard single precision (32 bits) or double precision (64 bits) exponent type floating point conversion. Positive and negative numbers in the range (E-38, .999. . .E37) are accepted. Also on input two non-standard conventions were adopted.

1. If no decimal point appears but some mantissa digits do, the decimal point is assumed to be in front of the $\ell$th mantissa digit from the right (e.g., if E15.6, 2643E7 is interpreted as 0.002643E7).

2. If no mantissa digits appear but a valid exponent field does, the mantissa is assumed to be ±1.0 (e.g., ±E6 is interpreted as ±1.0E6).

Note also that, as in integer conversion, the exponent field need not be right justified.

E. nFm.$\ell$:

Standard F type (32 bits) floating point conversion. On input, E and F type are interchangeable since the entire field is scanned for an exponent field. Convention VI,D,1 applies here also. On output, if the number will not fit in the prescribed F field, it is output in an E field (if possible) with the maximum number of mantissa digits allowed by m .

F. nH or '. . . . .':

Standard Hollerith field. Null characters are not allowed. Three consecutive apostrophes in a field delimited by apostrophes are required to get one apostrophe generated in the text. For example

'USER'''S MESSAGE'

will output

USER'S MESSAGE

G. nX:

Standard column skip

H. Tn:

Standard Tab format. On input it sets the pointer to the $n^{th}$ byte of the buffer. On output, it positions the pointer at the $(n+1)^{st}$ byte of the output buffer ($n^{th}$ printing position).

J. nAm:

Standard character conversion, two ASCII characters per word. If, for example, the user wants to input (output) 10 consecutive bytes, A10 and 5A2 are both valid formats, corresponding to

. . .,ARRAY,⟨↑D10⟩

in the MACRO call. Use of the former is discouraged if the word count varies with different references to the same format. This is due to the fact that once the base address is computed, 10 bytes are transferred before checking if the argument list has been exhausted. This form (A10) does have a slight speed advantage since a base address is calculated once instead of five times.

On input, if m is odd, the high byte of the last word will be filled with an ASCII blank.

K. nRm:

RADIX 50 conversion, three characters per word. As in ASCII character conversion, 5R3 is equivalent to R15 for five consecutive words of storage. The same rules and limitations apply. The legal character set is:

A-Z,0-9,BLANK,$, .

If, on input, m/3 does not have a zero remainder, the last word will be padded with one or two RADIX 50 blanks (0's).

L. General input note:

If the record terminates (CR,LF) before all expected variables in that record have been converted:

0 is stored in all I,O,L,E,F, or D variables.

Blank is stored in all A,R, or H variables.

VII. Warnings and Limitations:

A. The TRAP and programmed interrupt low core addresses vector into the IOS program and are therefore not easily available to the user for other purposes.

B. All sections of user programs using .FREAD or .FWRIT (as well as IOS) must remain core resident since the argument addresses which appear in the Macro expansion are not stored by the stacking routine and must therefore occupy the same location when the task is executed.

C. There is no problem in issuing many tasks to the same device, either through the same or different link blocks, without waiting for completion (as long as the parameters of the Macro call are not altered). No scrambling of input or output will occur since IOS will hold a task indefinitely if the file (or device, if non-file structured) is currently engaged in a transfer.

D. IOS contains one link block which is used in reporting error messages. It has the name IOE and should not be duplicated by a user link block. The default device is the keyboard. Reassignment can be made via the DOS ASSIGN statement and if a file structured medium is chosen, the file name will be IOSERR.LST in the user's area. If a file with this name already exists or if DOS detects some other error while performing the .INIT and .OPENO, IOS will execute a DOS .EXIT without sending out the usual fatal error message.

E. All input/output to disk, DEC tape, and magtape is file structured, requiring a fileblock and all associated DOS initialization. For DECtapes there is a restriction of one output file on a unit although multiple input files can be read from the same unit even if one output file is opened. A maximum of one file can be opened on any magtape unit. No similar limitations apply to disk files.

F. Because of a hardware problem, DOS does not set the appropriate bit in the status byte when end of tape is sensed on Mag Tape (See Digital Software News for the PDP11, October 1973). Thus, it is not reflected in the user flag. For our purposes, this isn't much of a problem and will be ignored pending the first disaster.

G. The maximum number of list vectors (scalers) in an .FREAD (.FWRIT) call is 13 (decimal). The maximum allowed word count for any vector is 2047 (decimal).

H. When memory segmentation is enabled, all I/O blocks which communicate with DOS (Link Blocks and Filename Blocks) must reside in KERNEL virtual space. Clearly all explicit monitor requests (e.g., .INIT,.OPEN) must be made from KERNEL mode.

J. Obviously if a task is stacked at a processor level greater than or equal to "TSKLVL", the code must not have a waiting loop or the program will hang indefinitely.

K. A dataset which is involved in transfers through IOS can also be used by the programmer for explicit DOS I/O operations. One should remain aware of potential ambiguity, however, if the .WAIT or .WAITR EMT is used.

## VIII. Simple Modifications:

Changing certain parameters can either optimize IOS for some applications or make it run on a different hardware configuration. No recoding is necessary for the following changes.

A. Although all IOS buffers must have the same length, the total number of buffers and their size may be changed. Currently the numbers are four and 144 (decimal bytes), respectively.

B. The maximum number of stacked tasks is a program variable and presently is 20 (decimal).

C. The processor level at which task stacking (and task initialization) occurs is now four but could be changed to any value greater than that at which task processing operates but less than six (DOS clock routine level).

D. Task processing can be done at any processor level greater than zero, consistent with the above paragraph. It should be noted that large amounts (many milliseconds) of time may be needed at this level.

E. Internal parentheses in FORMAT statements can be nested to any level, each additional one causing only a modest increment in program size and slight degradation in execution speed.

F. IOS can be used on an 11/45 with or without a floating point processor. In the latter case, conditional assemblies reduce the size of the program to approximately 2.7K decimal words and will generate an error message if an attempt is made to perform a D,E, F, or L type conversion.

## IX. Overhead:

If no tasks are in the queue, but an input task has been initiated and is hanging (e.g., waiting for CR,LF from Tektronix 4010 Graphics Terminal), approximately 220 μsec(145 μsec + time required for DOS to process EMT 0) is required each clock tick (about 1%) to discover that nothing is happening.

X. Sample Program Flow:

```
        .MCALL  .WORLD,.CALL    ; U.of M. System MACROS
        .PARAM
        .GLOBL  IOSET,IOUSET
USER:   .CALL   IOSET           ;initialize IOS
        .INIT   #LNKIN          ;initialize link blocks
        .INIT   #LNKOUT
        MOV     #FILIN,R0
        .OPENI  #LNKIN,R0       ;open files
        MOV     #FILOUT,R0
        .OPENO  #LNKOUT,R0
          :
        .FREAD  #LNKIN,#FMTIN,FLG1,1,A,<↑D14>
3$:     MOV     FLG1,R0
        BEQ     3$              ;wait for completion
        BGT     XIT             ;exit if abnormal
          :
        .FWRIT  #LNKOUT,#FMTOUT,DUMMY,<↑D15>
6$:     MOV     DUMMY,R0
        BEQ     6$
        BGT     XIT
          :
        .CALL   IOUSET          ;disable clock interrupt
        .CLOSE  #LNKIN          ;now close files and
        .CLOSE  #LNKOUT
        .RLSE   #LNKIN          ;release drivers
        .RLSE   #LNKOUT
XIT:    .EXIT
FLG1:   .BLKW   1
DUMMY:  .BLKW   1               ;note user flag first element in
                                 vector
A:      .BLKW   ↑D14
FMTIN:  .ASCII  #(2I5/D20.10,2(/2D10,10X,A4))#
```

```
FMTOUT: .ASCII  #('USER VALUES'/10X,2I6/D15.8,2(/207,5X,2A2))#
        .EVEN
        .WORD   XIT             ;link block error exit
LNKIN:  .WORD   0
        .RAD50  /INP/           ;logical name
        .BYTE   1,0
        .RAD50  /CR/            ;physical device (card reader)
        .WORD   XIT
LNKOUT: .WORD   0
        .RAD50  /OTP/
        .BYTE   1,0
        .RAD50  /DK/
        .WORD   XIT             ;file block error exit
        .WORD   0
FILIN:  .RAD50  /DAT/           ;file name not meaningful, but
        .RAD50  /IN/            ;innocuous, unless input device
        .RAD50  /DAT/           ;assigned to file structured medium
        .BYTE   20,20
        .BYTE   233,0
        .WORD   XIT
        .WORD   0
FILOUT: .RAD50  /DAT/           ;filename DATOUT.TST
        .RAD50  /OUT/
        .RAD50  /TST/
        .BYTE   10,20           ;in area [20,10]
        .BYTE   233,0           ;protection code 233
        .END    USER
```

APPENDIX A: IO DRIVER

I. Since IOS necessarily perturbs the DOS clock routine, it was essential to find a way to restore that routine in the event of a user program crash so that the monitor would not collapse at the next clock tick. The solution was to add a driver to the monitor which would perform these patching operations. IO is a driver only in a formal sense. No input or output to any peripheral device occurs. At the initialization stage (IOSET) of IOS, an internal link block associated with the IO driver is .INIT'ed and a dummy file is .OPEN'ed for output. Then a special function request is made, causing the driver to be entered. The executed code saves the DOS clock routine jump address both in the driver and in the special function block. The address of IOS clock routine is inserted in the monitor, allowing the following sequence of operations each clock tick:

A. Enter the DOS clock routine

B. Jump to the IOS clock routine which interrogates an internal flag.

C. Exit with a JMP @ off the address stored in the special function block.

D. Continue with normal DOS clock routine processing.

This modification increases the overhead of the clock routine by approximately 10 μsec per tick.

II. The termination stage (IOUSET) of IOS .CLOSE's the file associated with the IO driver, again causing the driver to be entered and executing the code to restore DOS. Thus, it is permissible for a user program to sequentially call IOSET and IOUSET any number of times. Additionally, if the program is aborted at the keyboard by a ↑C,KI sequence (e.g., after a DOS fatal error message). DOS will automatically close all opened files, thereby resetting its own clock routine before removing the user program from core. Clearly if a user related error writes all over the IOS program, the monitor will crash and have to be rebooted.

III. When altering the monitor CIL, the file SYSTEM.MAC must be changed to indicate the presence of a non-standard "device" (see Appendix E of the SYSTEM MANAGER'S GUIDE). Arbitrarily, the interrupt address of the I O driver was chosen as 104 (octal) and the external page address (necessary only for version 9) was taken to be that of the line clock (777546). Therefore, when modifying the GDVUO Macro, the indefinite repeat block statement should read

```
.IRP     P,<<IO,104>,. . .>
```

for a version 8 monitor or

```
.IRP     P,<<IO,104,777546>,. . .>
```

for a version 9 monitor

IV. The IO driver is written so that only one link block can be associated with it at a time. If the user does not need IOS, but would like to transfer control into his own core area each clock tick, he could easily do so by following the scheme outlined above. As another alternative, the IO driver can be employed as a dummy output device, throwing away all transferred information.

V. The IO driver contains two assembly parameters which concern memory management and have direct application only to the operating system modified by the University of Michigan. If memory management is not enabled or if no additional monitor patches are required the statement

```
MEMSEG = 1
```

should be removed from the IO driver, leaving the quantity undefined. If it is desired to patch (and, subsequently, unpatch) the monitor to allow handling EMT calls from USER or SUPERVISOR virtual space, the statement

```
V08.08=1
```

should be left in if running under DOS V08-08 or removed if running under a version 9 monitor.