PS/8 - OS/8 - OS/12 NEWSLETTER

FEBRUARY-MARCH 1973

Contributions and correspondence should be sent to:

Bob Hassinger, Coordinator
PS/8-OS/8-OS/12 Special Interest Group
c/o DECUS
146 Main Street
Maynard, Massachusetts 01754

NEW FROM DEC

OS/8 Version 2 has been released. The most noticeable change is that OS/8 has been modified so that it will work with BATCH. This involves small coding changes so that BATCH can patch the monitor to intercept terminal input requests etc.

Along with OS/8 V2, BATCH and OS/8 BASIC have been released. BATCH requires 12K to run because it needs the top 1K or so of Field 2. The rest of Field 2 is still available. It gives a fairly nice batch type operation that can spool output that would normally go to the printer, etc., into files when no operator is available.

OS/8 BASIC is an extension of the series of PDP8 BASIC's that have been available. It runs under OS/8, can access OS/8 files, and can handle string type data. This is particularly nice for doing things like inputting file names at run time etc. User written machine language routines and a laboratory interface package are supported. Note the bugs reported below however.

The OS/8 Software Support Manual is finally available (only a year after it was announced). It contains some new information that makes it worth getting.

FALL '72 DECUS SYMPOSIUM

The Fall Symposium was held in Anaheim, California, across the street from Disneyland. When a vote was taken on West Coast locations for future meetings Anaheim got zero votes. That might have been because there were nothing but motels for as far as the eye could see in all directions - nothing else except the convention center building and Disneyland. It might also have been the hour and a half ride from the airport.

The OS/8 meetings were well attended. Denny Pavlock and Richard Larry represented DEC and they discussed the new releases of OS/8 V2, BATCH and BASIC. Richy also talked about OS/8 FORTRAN IV. By inserting code that simulates the FPP12 hardware into RTPS FORTRAN IV he is developing a new version of it that will run on 8K machines with or without the FPP12 option. DEC plans to release the system with three variations of the run-time support. They use:

1) a plain PDP-8 or 12; 2) a PDP-8e with EAE, and 3) a PDP-8 or 12 with the FPP12. While he is at it I understand Richy has been re-coding the run-time support and loader for considerable reductions in the amount of tape rocking or overlaying involved in running them. Richy also hopes to re-work the command decoder interfaces so that they are compatible with the rest of the OS/8 system programs (i.e. /G option will mean load-and-go rather than "do not load-and-go"). I am looking for a release of this in the summer, DEC says late spring or maybe early summer. The earlier plans to make two releases have been changed and now only one release is planned. It will support the full FORTAN IV system including double precision and complex.

Richard N. Stillwell gave a paper on "The IBM 360/50 as a PDP-8/I Peripheral." He has written a software package and worked up an interface so that a PDP-8 can run OS/8 using a 360 as a peripheral and even as the system device if you like. This gives you access to all the peripherals on the 360 such as line printers and card readers, as well as mass storage files. Programs running on the 360 can also access the files if you want to. Mr. Stillwell has written with an offer to share the programs with anyone who is interested.

John Alderman talked about his DSP-8 package of diagnostic programming support routines. It is a nice, fairly general package for any purpose, not just diagnostics. It includes message input and output routines, number input and output routines, push and pop routines, sort and jump type routines, and several more. DSP-8 is available from John for a reproduction fee.

C. G. Roby and D. L. Duffy talked about their PAL12 and SCROLL which are a bilingual assembler for the PDP12 (similar to LAP6-DIAL) and scroll type editor, both for the general OS/8 environments. This PAL12 is different from the one mentioned in previous Newsletters but it is quite similar except that it has a number of additional features. These, and a group of utility programs, are available for a price from the authors at the University of West Virginia.

Dr. Lewis talked about his revised version of SKED that runs OS/8. He has submitted it to DECUS.

NEW PROGRAMS IN DECUS

DECUS 12-81 - TV - PDP12 VR12 scope handler - a further improvement on earlier versions reported on in the past. A model of compact, clever coding. It took a lot of work by very good programmers to write this handler.

" 12-82 - DIAL - LAP6-DIAL to PS/8 source converter.

" 8-538 - INTIOH - Integer data type only IOH routine for FORTRAN

" 8-530 - 8BALIB - Macro library generator for 8BAL (see 8-497). Also included is a set of macros to create a set of routines to do character by character I/O to PS/8 devices that are quite nice.

DECUS 8-549 - Polynomial Least Squares Fit.

" 8-550 - Modified Matrix Inversion - Real Numbers

" 8-552 - Storage Display Device Handler (2 page device handler uses VC8e or VC8I interface to a storage scope. Uses 4 x 6 dot matrix).

" 8-554 - ANOVA and DUNCAN - Analysis of Variance

" 8-555 - MULTC - Multiple Correlation Program

" 8-556 - CHISQ - Chi Square Program

" 8-557 - CLUSTR - Cluster Analysis Program

" 8-558 - CORREL - Correlation Program
PCOMP-VARMX - Factor Analysis

" 8-562 - DISCRT - Produces directory listings sorted by file name from a file containing a normal PIP /E type directory listing.

" 8-564 - A STATISTICAL SYSTEM IN PS/8

" 8-570 - BIN4SV - Translates .SV format files to .BN formal files.

" 8-571 - INPUT - OS/8 Version - 8-480 revised for OS/8 FORTRAN/SABR routine to do free format floating point input.

" 8-573 - EDITS - A PS/8 Editor for Non-Storage Scope Display. Has provision for a software character generator or a special hardware character generator interface. Adds view command to PS/8 EDIT. Also G command now does a J type search without outputting text passed over.

" 8-574 - TD8e System Device Handler for 8K PS/8

" 8-576 - LPAL8 - LOCAL PAL8 - Simplifies generation of a special version of PAL8 with a customized permanent symbol table.

PROGRAMS SUBMITTED TO DECUS - Not yet ready for distribution

XDIREC - Outputs directory information for a list of selected files.. Uses the "*" convention to specify a match on all names or on all extensions.

IDXWT & IDXRT - Group of FORTRAN/SABR routines to do unformatted I/O to OS/8 files.

ADFILE - ADTAPE adapted to PS/8 for the PDP12A

OS/8 SKED - SKED running order OS/8 - See DECUS PROCEEDINGS FALL '72

DIBILD - Program to rebuild clobbered directories from a copy of the directory listing.

TEKTRONIX 611 Storage Scope Handler - Uses the interface described by D. C. Uber in the DECUS PROCEEDINGS - SPRING '72.

FUTIL - This is the program that accesses the contents of any OS/8 device in an ODT-like manner for inspection and alteration. It contains many modes and options to allow working in just about any format that suits the data. This has already been a life-saver for me in repairing errors in directories.

ODUMP - Octal Compare and Dump - Program to compare and dump OS/8 files. Masking for compares and searching for dumps are included. Useful for comparing two versions of a .SV file.

FSPEED - Speed Up Patches for 4K FOCAL-69. Changes to a number of the internal routines of FOCAL-69 (could be adapted to FOCAL-8) The patches were developed after an extensive program of timing the execution of FOCAL.

INVENT-8 - A package of FORTRAN/SABR sub-routines for manipulating binary unformatted data under OS/8. It allows the user to open input and output files (i.e. device #4) as well as read and write binary unformatted fixed length records of up to 125 12-bit words per record.

Also included is a generalized sort generator for sorting the records produced. This is the first known sort to become available that will operate under OS/8 on <u>any</u> OS/8 configuration. (The sort is in-place. It does not require any auxiliary storage outside the space occupied by the INVENT format file to be sorted.)

This system can be very useful for doing business type work under OS/8. The author runs an order entry system with it.

WORK BEING DONE

D. Lloyd Rice has written about a revised OS/12 system he is finishing up. It has a scope output routine for the PDP-12 scope. All monitor and command decorder interaction appears on the scope, pushing lines up from the bottom as new lines are entered with about 15 operations at a time on the screen. The scope buffer is saved between operations. There is an optional TTY echo

controlled with control-E and control-K. Also, the current system date always appears in a header line at the top of the screen. All user core is restored as necessary so that the scope system is transparent to the user. He expects to submit the system to DECUS before long.

C. G. Roby at the University of West Virginia got an idea at the DECUS meeting and in a couple of weeks of spare time had implemented a virtual memory OS/8 system that runs on a 4K LINC-8 and uses LINCtape as the swapping device. An 8K PDP-8 is simulated in software. Using LINCtape it is said to be a toy. It takes half an hour to print an index. He thinks that with a DF32 it might be marginally useful however. Given an 8K machine this sort of thing might develop into something that runs much better. Has anyone else looked into virtual memory schemes?

Ben H. Storey sent along a sample of the output from a set of programs that generate Real Estate Appraisal Reports. He would like to hear from anyone who might be interested.

WORK NEEDED (WISH-LIST)

Stan Rabinowitz has suggested the need for a section in the Newsletter to develop ideas for new programs and changes to existing programs. He is particularly looking for things that DEC should do. John Alderman also feels that the same sort of thing is needed to coordinate non-DEC user efforts on generally useful programs, to avoid duplication and to develop and disseminate what is needed and what it should do. That is the intention of this section.

C. C. Wilton-Davis is interested in a device handler for an incremental (i.e. Calcomp) plotter. It would accept an ASCII command stream and would interpret it as vector plot commands rather like the "FASTPLOT" and other plotters designed to run off a time sharing system. He presently has a stand-alone program that accepts input from files in the form of XCOORD YCOORD, XCOORD YCOORD, etc., with control characters to initialize and suspend plotting, lift (L) the pen, and quit. All other characters before initializing (!) and between suspension (Q) and the next ! are ignored, except for control-Z which exits to the monitor. The trick is to figure how to get this all in a two page handler so it can be used directly by BASIC, FOCAL, FORTRAN and so on. Anyone have any ideas?

In thinking about ways to extend and improve OS/8 a couple of thought have come to mind. A considerable number of users have expressed a desire to be able to reserve a section of core for protected code like the overflow from a TD8e handler, extensive scope handlers, display buffer space for the VT8e, and many other system enchantments. BATCH would like to have this feature also. If a USR call were implemented that would return the highest core location available for use by a program then a system could be configured that would reserve as much room as needed at the top of core. Cooperating programs would use this call instead of sizing core themselves in their initialization. A new release

of the system could implement cooperation of this sort without too much trouble. Of course if a location in the resident monitor head area can be assigned it would hold this information instead of calling the USR.

A more difficult problem is making provision for interrupt environments such as Foreground/Background, Multi-user or Time Sharing, and so on. Because interrupts always go to location ϕ of field ϕ and because the system loads by full pages only, all of page ϕ of field ϕ must be protected from user programs in the general case. This effectively means that field ϕ is not going to be of much use to user programs when they are loaded in this type of a system. Ordinarily this sort of system would expect to need more than 8K to make room for the Foreground or Multi-user Monitor anyway, so consider the idea of relocating user programs up one (or more) fields. There are several ways to do this. With suitable programming techniques and conventions the sources could be reassembled very easily to run in the new fields. It would be nice if the same version of the program would run in a standard system or in this special one however. We could take advantage of the "time-sharing" instruction trap on the PDP8e (and available at considerable cost on the later PDP8/I's). It has been used to do this sort of thing by John Alderman and others. The trap hardware intercepts the CDF's and CIF's and the monitor translates them to the relocated fields. This slows down the program, however, and it not compatible with all machines that OS/8 runs on. A USR call could be defined that would return a field relocation constant that a cooperating program would use to patch itself during initialization. The programs might even detect the relocation itself by doing an RIF to find out what fields it was loaded into.

If a program cooperates in these ways bits might be set in its job status word so that the special monitors can tell how to handle them.

In this sort of a system the resident monitor or foreground code will usually contain interrupt driven replacements for most of the standard device handlers. If the normal system device handler is replaced with code that has entry points defined for all these devices and which contains a trap or jump to the foreground monitor then the I/O to those devices can be intercepted without the using program ever knowing about it. One particular problem remains. Many programs including the present monitor modules do character by character or line by line I/O to the terminal device without calling any device handlers. Either a teletype driver routine must be resident in the monitor head or such I/O must be done with a USR call if the instruction trap hardware is not available. This is a desirable feature anyway to simplify changing the terminal device when desired.

Is all this understandable? Do you agree that some or all of it is desirable? Needed? Do you have any better ideas? Do you know of any work on these lines? Are you interested in pursuing it if DEC does not? Write and let me know your thoughts.

BUGS

The released version of OS/8 BASIC has a number of bugs. One involves string compares and another one involves checking for control-C. DEC has worked up patches for these and will publish them soon. A third bug involves I-O to non-file structured devices. It has proven to be rather persistent and a complete fix is not yet available.

Stan Rabinowitz reports that EDIT12 (DECUS 12-50) dies in panic mode. Panic mode is when you run out of room on the output device). No fix has been reported.

HINTS AND KINKS

Some people may not have recognized a characteristic of OS/8 LOADER. After the first page of COMMON, space for COMMON is reserved in two page (one block) chunks. This means that, if you have anywhere from one to 128 words of COMMON declared, LOADER will allocate the space from 10200 to 10377 (one page) but, if your COMMON requires anywhere from 129 words up to 384 words the allocation will be from 10200 through 10777 (three pages). For this reason a careless use of COMMON can cost you up to two wasted pages. When you are running FORTRAN on an 8K machine you cannot very easily afford this, so be careful.

A neat idea was circulated at the Fall DECUS meeting. In general, an all zero ASCII character (NULL) is ignored in OS/8. You can take advantage of this when doing I/O. If you put one eight bit character in the right eight bits (4-11) of each location of your buffer and make sure the left four bits (0-3) are zeros, then when the buffer is accessed as OS/8 packed ASCII, every third character (the left 4 bits of two adjacent locations) unpack as a zero and are ignored so the one character per word format that is so convenient to create and use within your program can also be used for I/O without the need for the usual 3-into-2 packing. Of course, if the output goes to a file oriented device it will take more space, but to non-file oriented devices there is no problem and you can output exactly one line at a time if you fill out the buffer with zeros every time. Jim Crapuchettes has used this trick in FUTIL to get a very small, convenient output handler that outputs each line via the standard LPT: handler as it is called for.

Stan Rabinowitz sent along the following useful item:

PASSING SWITCHES TO OS/8 PROGRAMS WITHOUT INVOKING THE COMMAND RECODER

At the last DECUS meeting, it was pointed out to me that comments could be inserted after commands to the PS/8 keyboard monitor by preceding them with either a slash or semicolon. For example:

```
.ODT /INVOKE DEBUGGER

.GET SYS FOO ; BRING FILE INTO CORE
```

```
.; BE RIGHT BACK - TERMINAL IN USE

.R PAL8/RUN THE ASSEMBLER
```

This works because the PS/8 keyboard monitor terminates its scan whenever encountering / or ; or certain other special characters.

Although this has limited utility (inserting comments into a BATCH stream is one possible use), it gave me a good idea which I believe others may wish to use.

Often a user program may have several options or alternatives during its execution. The program frequently must type out messages and ask the user to reply with the option desired. If the program happens to require the command decoder, then the user may type in the options via slashes to the command decoder. For example:

```
*FCO/A ← GLICHS/B/C
```

The command decoder then sets corresponding bits which the user may interrogate. This is very convenient. However, if the program does not require file or device specifications, then it is wasteful to call the command decoder (and do swapping) just to allow the user to type in options.

I propose an alternative for small programs which have room to spare and require options to be typed in by the user.

The program should be careful not to load over the PS/8 input line buffer (from the keyboard monitor). When it starts up, it should then scan this buffer (which contains the RUN command which invoked this program) and look for options specified following a slash.

For example, I have a program called BOOT which is used to boot-strap from OS/8 to any device (not necessarily OS/8). It loads the appropriate boot-strap into core and branches to it. Normally, this program would begin execution and then ask you what device you want to boot-strap onto and you would reply with the appropriate mnemonic, e.g.:

DT	(DECtape, TCØ8)
TD	(TD8E DECtape)
LT	(LINCtape)
RK	(RK8)
RF	(RFØ9)

etc.

Using the method described above, no dialogue is necessary. The user merely types in the device along with the run command, e.g.,

.R BOOT/DT

.RUN DTA3: BOOT.SV/RK

This artifice is much more convenient than other methods. Also, a greater amount of versatility is permitted than with the command decoder since OS/8 will ignore anything after the slash.

The only drawback is that you have to work a little harder (but not much) to get at the options specified because you have to do your own scanning. But the algorithm is simple - you merely scan down the OS/8 input line buffer (which begins at location 1ØØØ) until you find a slash. Then you scan off any information you want. The end of the line buffer is signalled by a word containing Ø. The characters in the input line are conveniently placed in this line buffer in consecutive order, one ASCII character per word.

You must be careful that your program does not load over this area of core (locations 1ØØØ through 1377 inclusive) if you are going to use this trick.

If the option scanned off is incorrect, you can give an error message and either abort or ask him to retype the information (which you can put into the OS/8 line buffer so that you can reuse your scanner).

Or you can use the presence of options to mean execute the program differently. For example, I have a program called PRO which is used to protect various blocks (sectors) on an RK8.

It is normally called by

.R PRO/O

it doesn't ask any questions but instead protects all those sectors used by OS/8 whereas

.R PRO/C

protects those sectors used by COS.

One additional comment: Although DEC is no longer supporting PS/8, it would be nice if programs using this trick could be run under either PS/8 or OS/8. The problem is that PS/8 and OS/8 have their keyboard monitor line buffers at different locations in core. This difficulty is easily programmed around if you can spare both buffer locations (i.e. your program mustn't load into locations 1ØØØ-1777). The code then looks like:

```
XR=1Ø          /index register which will point to
               /just before appropriate line buffer
```

```
            TAD I (1ØØØ         /get contents of OS/8 line buffer start
            TAD   (777
            SNA CLA             /is it 7ØØ1?
            TAD   (6ØØ          /yes, we were loaded by PS/8
            TAD   (1ØØØ-1       /no, we were loaded by OS/8
            DCA   XR            /store location before start of line buffer
LOOP,       TAD I XR            /get next char
            SNA
            JMP END             /end of line buffer
            TAD   (-"/
            SZA CLA             /found slash?
            JMP LOOP            /no, keep looking
            :                   /yes

END,        :
```

This code uses the following facts:

(a) The OS/8 line buffer starts at location 1ØØØ

(b) The PS/8 line buffer starts at location 16ØØ

(c) PS/8 contains a 7ØØ1 at location 1ØØØ