PS 8
OS 8
OS 12 Sig
DECUS

PDP-8/OS-8 NEWSLETTER

April Number 16 1976

Contributions and correspondence should be sent to:

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

GOOD NEWS FOR THE 8

The past two or three years many PDP-8 users have been expressing the feeling that DEC was ignoring the 8 family in favor of other lines such as the 11. Investigation into the reasons has suggested that one of the reasons for this situation was the fact that marketing of the 8 family has been fragmented among a number of product lines. In each product line the 8 was a secondary concern so it received relatively little support and what support there was tended to be poorly coordinated between the separate product lines sometimes. Recently there have been rumors that DEC was finally recognizing this situation and that something was going to be done to rectify it. Although the formal announcement had not yet been made at Newsletter deadline time, I was able to obtain the following brief announcement because it will not reach you until after the official announcement is made:

"The new PDP-8 Marketing Group was announced at the Spring DECUS Symposium. By concentrating OEM and user resources, Digital management expects to provide better support for PDP-8 customers. As of July 1, the new group, composed of PDP-8 elements of OEM, LDP (Lab), IPG (Industrial) and EDU (Education) product lines, will conduct all OEM and most end user PDP-8 business for DIGITAL. Jim Willis, [illegible] PDP-8 Product Line Manager also performs that function for the new combined product line."

As many of you know, Jim has been very cooperative and helpful with the users and has supported the 8 family very well.

The indications are that this development is accompanied by increased funds for software development. This fact and the transfer of some former 8 software development people has resulted in a substantial re-staffing of 8 software development. We will have a chance at the DECUS meeting to judge better now that effort is going, but early contacts suggest we will see a revitalization and increased level of activity in this area.

SURVEY RESULTS

The responses to the "User Input to DEC Software Planning" survey distributed with the last Newsletter have been coming in steadily. As of the Newsletter deadline 102 responses had been received. The responses range from simple check offs of answers to three page expositions. Copies of the responses are being reviewed by the PDP-8 product line and the PDP-8 software development group.

The meat of the responses is in the many and varied comments. It is not possible to summarize all the comments here but I will list the current count of the question check offs to give some idea of how it is going.

1.	Operating System Development	71
	Language Development	29
2.	"OS/8" Type Single User System Development	30
	"RTS-8" Type Multi-Task System Development	70
3.	RTS-8 FORTRAN IV Development	48
	New BASIC Development	25
	Other:	
	FOCAL	8
	PL-1, PL-M (etc.)	5
	PASCAL	4
	APL	3
	Business Languages	6
	RPG	2
	ALGOL	2
5.	Accept Fewer Supported Configurations	
	Yes	71
	No	26

NEWSLETTER DEADLINE

The final deadline for camera-ready copy for the next Newsletter is June 25, 1976. See Newsletter #15 for deadline details.

"NAME THE S.I.G."

Jim VanZee sent along an idea for a new name for our Special Interest Group (see reque. t in last Newsletter). He suggests that besides the obvious "12 Bit SIG" we might like a group or newsletter name of "BYTE AND A HALF" (Maybe "BYTE 'IM IN HALF" - Jim points out the interesting logo ideas that leads to). Let's have more ideas and ~~~ ~ons. Computer people seem to be very good at this sort of thing.

NEWSLETTER POLICY

At its April meeting the DECUS U.S. Executive Board formally adopted its policy regarding non-commercialization of DECUS publications. Announcements of software and other items for sale are generally limited to technical information as

opposed to obvious promotional material and no prices are allowed except where they are clearly nominal to cover reproduction and handling. Enforcement and interpretation of the policy is basically in the hands of newsletter editors and SIG chairmen. The policy will have little or no impact on our Newsletter as it is essentially the policy I have been following. If you need a copy of the official policy, contact the DECUS office.

EUROPEAN OS/8 SIG

Ernst Lopes Cardozo writes to say that the European group will try to channel its inputs to the Newsletter through Lars Palmer on the new deadline schedule. Ernst supports expanding the scope of the SIG to all 12 bit issues, hardware and software. I expect this step will be formalized at the May Symposium in Atlanta.

Ernst reports that the Dutch LUG-8 has organized a few "tutorial days" in the past. An expert would give a one-day course on TECO, (ROG-)ALGOL, 8BAL, etc. He has been asked to organize some of these user-for-user courses during the day immediately preceding the next DECUS Europe Symposium in September. He inquires about what we do in the U.S. Most of this sort of thing in the U.S. has been informal - mainly individual sessions during the symposium itself. I wonder if there is any sentiment in favor of separate get-togethers isolated from the multi-parallel sessions at the U.S. Symposium? Maybe we should be considering sessions either before or after the symposium or perhaps completely separate meetings - maybe on a regional basis where the costs could be minimized. This sort of thing would tie into a development of 12 Bit Local User Groups in the U.S. and elsewhere. We certainly have enough members (about 1600 worldwide) and there are already many precedents for LUG meetings as well as totally separate U.S. level SIG meetings. What do you think? Will you help put it together the first time? I can help with names of other SIG members in your area but I can't do the main work of organizing and running sessions.

HASSLE STATISTICS PACKAGE

Lars Palmer has written about on-going work on his STAT package (DECUS 8-660) which he expects to submit to the library before long. The package is a statistics package that provides a fairly extensive set of capabilities in a nice combined package that takes advantage of several features of OS/8 FORTRAN IV such as overlays.

FORMS AND OTHER INFORMATION FROM LARS PALMER

Lars Palmer sent along some information on his OS/8 FORTRAN II program "FORMS" which converts a PDP-8 with CRT (particularly a VTØ5) into an intelligent form filling terminal with a "Key to Disk" type function. Lars does not plan to submit this program to DECUS but if anyone is interested they can contact him directly at: AB Hassle, Fack, 431 20 MOLNDAL 1, SWEDEN.

Lars also mentioned some routines he has available separately that might be of interest. KSORT (from DECUS 8-660) for sorting and transforming data, FILSIZ which will determine in an OS/8 FORTRAN IV program what the size of files that have been specified at run time is, LESQ (from DECUS 8-661) which gives

a flexible way to do least squares curve fitting, and MACRO which is a tape with several useful TECO macros for listing directories and files. This last includes a scheme for a BASIC program which is hooked up with the CCL DATE command to create a file with the current date in it for a TECO macro to use in inserting the current date in files as the first line. Also in this package is some material to use with Tom McIntyre's MEDIA management package (i.e., combined sorted directories of all your tapes, etc.).

NOTE ABOUT TIME SHARING FROM BILL HAYGOOD

Bill has written to say he has completed work on his multi-user time shared operating system which runs OS/8 as a job on multiple terminals. He has modified ODT and some of the device handlers for efficient time sharing under OS/8. The present system is a specialized one which runs on a U.S. Postal Service dual PDP-8e configuration. Bill says that performance is dependent upon what OS/8 program you are using. EDIT is slowed somewhat but TECO runs well. The system includes "down line bootstrapping" of the slave machine from the host. At present he is running OS/8 at two terminals and special CAI programs on the other five. Bill says that the system will soon get an additional 96 K of memory and a special interface to give a 128K PDP-8e. He plans to develop the software for this configuration so OS/8 can still run as a job.

If anyone is interested in seeing Bill's system in operation they can contact him. He says he cannot handle much correspondence but if anyone is going to be in Salt Lake City he would be glad to show off the system. Bill feels he has solved some very sticky problems and most of DEC's OS/8 programs will run unmodified, even BUILD works correctly. If you want more information, Bill's address and phone are: Computer Services Co., 3704 Ridgecrest Drive, Salt Lake City, Utah 84118, his phone is: (801) 966-1414.

SCIENTIFIC SUBROUTINE PACKAGE UPDATE

Lars Palmer has contributed some work to improve the S.S.P. The DECUS library now has the original OS/8 submission with complete comments on five DECtapes. It now has, in addition, a compressed version. Lars removed all comments making it possible to squeeze the package on two tapes. I rearranged the tapes so that all routines that require an FPP with double precision option (i.e., DOUBLE PRECISION is used) on one tape and all the rest of the package on another. Thus you can have the entire non-double precision library available on a single tape (about 270 files!). This also allows you to order just one or two tapes if you have sufficient documentation. If you can't wait for the S.S.P. to appear in the next library catalog then contact the library, but please don't bug them unless you really need it. They have been hard pressed lately with personnel shortages.

FLAP

My note about FLAP in the last Newsletter drew several responses. Hans Goebel says that as of last October a version of FLAP that runs under OS/8 and the FFP Support Package were still being distributed for the FPP-12. He has found the Support Package quite useful and has adapted the random number generator and an interrupt mode driver for a second terminal to RALF. Lars Palmer writes that the version of FLAP he received still requires the FPP hardware. As some

of you know, FLAP and RALF are alternate conditional assemblies of the same source. RALF was modified to work without the FPP hardware back when RTPS FORTRAN was re-written to run without the FPP and was re-designated OS/8 FORTRAN IV. It would appear that the version of FLAP still being distributed came from a version of the sources from about four years ago. I have corrected some errors that have crept into the sources since then so that I can generate an updated version of FLAP which takes advantage of a number of improvements put in for RALF and, of course, the new version does not need the FPP to assemble programs.

Professor Zimmerman indicates that he has used FLAP but that the version he received had a limited symbol capacity (100 or 200 symbols). I have not determined if the new version is any better but I think it may be. Full use of extended memory is not made by the FORTRAN IV compiler and RALF, however, so FLAP probably does not make good use of extra memory either.

Brian Converse says that he and Larry Alber have both been interested in FLAP but that they both have experienced a series of problems trying to get the distributed versions to work successfully. Brian would like any information available on timings for the ATX and XTA instructions on the FPP-12 and the FPP-8A as he is very concerned with working on 12 bit data as fast as possible and the information supplied by DEC is inadequate. Brian points out that at the very low price for a complete machine, the PDP-8A/800 with an FPP-8A should swell the ranks of FLAP users.

Lars Palmer sent along information on the FPP Support Package. It is a sort of library in the form of a source file full of conditional assemblies that you assemble with your program. It contains the code to service the FPP, you can get buffered I/O support, I/O formatting and conversion, and a substantial collection of mathematical functions. Only what you ask for is included in your program. It looks like it has good potential in spite of the fact that it is almost unknown and DEC has given it almost no promotion. Brian Converse points out one question, however. The FORTRAN library appears to have been largely derived from this support package. It appears that as corrections to the code in the FORTRAN library have been made that no retrofit to the FPP Support Package has been attempted. An outdated version seems to still be distributed. This would not be difficult to correct, however.

Because the FPP Support Package is in source form it seems that integrating the FPP simulators from FORTRAN IV would not be hard. This will give a floating point package that is more powerful than the standard DEC software and which is upward compatible from the plain basic PDP-8 through 8e EAE to a full scale floating point hardware configuration. This package could even be the basis of a higher level language compiler (RALF and its loader would be nice if it did not presume the FORTRAN IV Run Time System). If anyone is interested enough in this idea to put in some work helping refine what I have already done on it let me know.

LINC Mode Under RTS-8

Dr. Goetz Romahn has written to say he has modified RTS-8 to handle LINC mode interrupts. If you are interested you can contact him at: Heinrich-Hertz Institut, Einsteinufer 37, 1000 Berlin 10, Germany.

CCL MODIFICATION TO MAKE DEFAULT DEVICE DSK:

The currently distributed versions of CCL force the default device to be the permanent device SYS: (i.e., internal device #1). If you would prefer DSK: to be the default device consistent with the conventions in the rest of OS/8 it is easy to change if you can settle for permanent device #2 (this will be DSK: if you use BUILD - otherwise use RESORC with /E to check). Find the symobl NMOVE. A few locations later (depending on the version you have) you will find where a 1 is forced into the accumulator if no device has been specified. Just change that instruction to CLA CLL IAC RAL to force a 2 instead. Note that the USR is already in core during this routine so a slightly more complex patch could actually determine the device number for the device currently assigned the name "DSK" instead. This would allow you to use ASSIGN to temporarily re-assign the default device whenever you want.

DIRECT /A

I recently incorporated some material into DIRECT that was circulated at the Fall DECUS Symposium. In particular I inserted the code to implement a /A option which causes the directory listing to be alphabetized (it overrides /E if both are given) while retaining all the other features of the V3C release. Unfortunately I am not sure who to give credit to for the code I used as all identification seems to have been separated from it. I hope I can convince DEC to use this enhanced version of DIRECT in future OS/8 releases but in the meantime if anyone wants to try it out they should contact me.

FORTRAN II USING THE FPP-12

Hans W. Goebel is looking into trying to use the FPP-12 (or presumably the FPP-8A) with OS/8 FORTRAN II. If you would like to contact him, his address is: Purdue University, Properties Research Lab, School of Mechanical Engineering, 2595 Yeager Road, West Lafayette, Indiana 47906.

MICROPROCESSOR CROSS ASSEMBLERS

I recently received a package of information from William Bonham of Sierra Digital Systems. It documents his "X8" line of microprocessor cross-assemblers that run on the 8 family under OS/8. The line presently covers the M.O.S. Technology 6502, Motorola 6800, and the Intel 8080. Bill plans to support the Signetics 2650 and the Fairchild F8 shortly. He is also looking into the possibilities for the 4040, SC/MP, PACE, 1800, PPS-8 and CP1600.

The X8 series are written in PDP-8 assembly language rather than FORTRAN so they run much faster on the 8, require less DEC software investment (i.e., no FORTRAN IV), and they all run in any 8K OS/8 system. I hope those of you who use this software will contribute comparisons on the features and performance as they seem quite nice, but I do not have enough experience with microprocessor software to make a good judgment.

The prices seem reasonable in comparison to the competition. Anyone who already has an OS/8 system or who is planning to do extensive microprocessor software development should find the X8 series very cost effective compared to the typical time sharing costs I have heard of. The Bonhams are at 1440 Westfield Ave., Reno, Nevada 89509.

PATCH FOR FORTRAN IV

Lars Palmer has forwarded a patch done by A. Windram. It is for PASS3.SV V3.03 to correct some bugs and implement a /M option which causes only lines with errors to be listed. Because it may need to be modified for the new release of FORTRAN I will not publish it in this Newsletter, but if anyone needs a copy write to me.

NOTE FROM MIKE LOWCOCK

Mike Lowcock and Mike Humberston have written to say they have three systems developing RTS-8. They would like to hear from others who have solved or have ideas of how to solve the restrictions of the OS/8 filing system, and support of the FPP (under FORTRAN IV). They have developed a card reader drive for RTS-8. Until they decide to submit it to DECUS contact them at: Department of Economics, Alfred Marshall Building, University of Bristol, 40 Berkeley Square, Bristol BS8 1HY, England.

NOTE FROM STUART DOLE

Stuart Dole has written about OS/8 BASIC. He comments on the problem of needing a symbol table to work with user functions. DEC usually fails to update published symbol tables when they release new versions of software. I think this is the case with BASIC and it is also true of updates to information published for OS/8 FORTRAN IV. He has written BASIC functions for reading and writing random access files in 12 bit format. He also has included some useful TECO macros and patches that I will try to include. His address is: c/o Anesthesiology Research, 1386 HSE, U.S. Medical Center, San Francisco, CA 94143.

MATERIAL FROM STEVE LIGETT

Steve (formerly with DEC, now at Dartmouth) has sent along a number of interesting items. Among them are an article on OS/8 Bootstraps, BATCH log on the LA-36, a BASIC Margin Function, and several patches for DIRECT. I will include all of them that will reproduce. For any that don't reproduce, plus some trial code to use the type ahead buffer in RTS-8 to bring up OS/8 with commands in the buffer contact me.

REVIEW OF EDUCOMP'S TIMESHARED OPERATING SYSTEM

EDUCOMP's ETOS makes available a 32K virtual OS/8 system to each of many simultaneous on-line users. It will handle interactive, batch and real-time tasks. Peripherals currently supported by the ETOS monitor are the extended arithmetic element, line printer, card readers, Dectape, high speed paper tape reader, high speed paper tape punch and a variety of terminals.

For devices ETOS does not support, EDUCOMP has devised a unique plan for implementation. Device handlers may be run as detached user jobs. Users who desire to utilize the device send the job a message concerning the respective user task. This method has the advantage of separating the monitor and the device handlers. Users can debug their handlers external to ETOS. Therefore, ETOS can be more easily and satisfactorily supported, and modularity is maintained. Documentation is provided for the writing of device handlers and hooking these jobs into the ETOS interrupt skip chain. Because the detached task is hooked into ETOS in this manner, the unusual peripheral can be supported as efficiently as a standard peripheral and timesharing need not be noticeably affected.

The real-time support may also be used in conjunction with the device support. The user has the facility under ETOS to lock his job into core for a specified period of time. During this period, he may or may not decide to service interrupts. The user can actually program most real-time applications so that other timesharing users are unaware of the real-time processing. This capability will allow analog to digital conversion, digital to analog conversion, and simulation of events taking place in a critical period of time.

When ETOS is running, most processes are in user mode. This mode of operation causes the hardware to trap all Input/Output instructions, which include instructions that load and read the IF and DF registers. When the user executes a field instruction, the monitor maps the virtual address (the field specified in the change field instruction) into a physical address space. If the field which the user desires to access is not resident, the monitor must swap into memory an image of this field stored on the disk. This mapping procedure is transparent to the user. He programs as if he had a real 32K word machine to himself.

To speed up the mapping process, EDUCOMP designed a timeshare control module. This module plugs into the omnibus like any other PDP8 module. The difficulty that this option overcomes occurs in the case of the Change Instruction Field (CIF) instruction. When this instruction is executed, it does not take effect until the next JMP or JMS instruction, thus making cross-field subroutine calls possible. As a result, when it gets the CIF instruction, the timesharing monitor may only note the fact that a change is pending. As no monitor trap is caused by the JMP or

JMS, either the monitor would have to emulate all instructions from the CIF to the next JMP or JMS, or a hardware unit must trap JMP or JMS when it is enabled. EDUCOMP chose the latter course because of its greater efficiency. It is important to note that a computer with this additional module operates normally when the user processes stand-alone.

In addition to multi-user OS/8 capabilities, the ETOS user has the facility to enter the System Command and Login/Logout Executive (SCALE). This provides each user with absolute control facilities over his virtual machine. In essence, the user has a virtual front panel. All virtual registers can be displayed and modified. Virtual memory locations may also be examined or changed. This provides a powerful debugging tool for Assembly language programs. If the user's program halts or he destroys his copy of OS/8, no other users are affected, and the virtual machine with virtual OS/8 may be restarted from the terminal with a BOOT command.

File security and selective information-sharing have also been the subject of a great amount of development effort. A password and an account number must be specified to allow access to system resources. If the password or account number does not match a table of valid accounts set up by the system manager, entry to the system is denied.

Each account number has a set of attributes assigned to it by the manager. A file quota may be imposed to limit the amount of storage available to an account. Users can thus be inhibited from inadvertently or intentionally filling up the public disk structure. A user's files are, in addition, stored with a protection code which allows or disallows read/write privileges on those files of users under other accounts. Multiple users can, if the file protection code permits, read the same file simultaneously. An account may allow a user to mount his own private pack to extend his available storage area. Such private packs may be either totally private (single-user access only) or semi-private (accommodating more than one account).

In addition to the capabilities available to all users, there exists a variety of features which the system manager may utilize to effectively control the system processing. The manager can assign different priorities and different quanta for user tasks. The maximum amount of core for a user may also be restricted. Job accounting may be used in conjunction with system features such as line printer spooling to increase system throughput.

ETOS was developed by the EDUCOMP Corporation, 196 Trumbull Street, Hartford, Connecticut 06103.

Operation of FRTS and BASIC with a non-TD8E 2-page System Handler

We are running OS/8 with a single floppy disk (Sykes) as the system device. To increase disk capacity (and incidentally to improve speed) we are packing four 12-bit words into six 8-bit bytes, and this extra complexity necessitates the use of a 2-page system handler.

Such a system runs into complications with FRTS and BASIC, since both these programs shift various areas of core around, and are programmed to take special care of the second page of the system handler only if the TD8E handler is recognized.

I have identified the locations involved in the recognition and special action, and it seems worthwhile to make this information available to other users who may run into similar problems.

1. FRTS

The TD8E handler is recognized by examination of the content of the field 0 address stored at 12757. At present 12757 contains the address 7642, and the TD8E handler contains 6223 (CIF CDF 20) at this address. If the content of this address is 62x3 (bits 6-8 are masked out), the addition of 1575 contained in 12755 gives zero, and this initiates the special action.

A subroutine at 17526 changes three field 2 instructions in page 1 of the handler to field M (max), and the second page of the handler is shifted to the top of core. The non-zero content of 12703 following this action ensures that a subsequent transfer of 3000 words is made into 4600-7577 of field M rather than into 5000-7777 .

On completion of the run, the second page of the handler is returned to field 2 and the field 0 instructions are rewritten by the same subroutine as before.

In the special case of a 12K system, where M=2, the only action necessary is to place some arbitrary non-zero instruction (e.g. HLT, 7402) in 12703. This ensures that the second page of the handler will not be overwritten. In the more general case, however, the address at 12757 and the two's complement of its (masked) content at 12755 must be modified to correspond with the new handler, and the instruction-changing subroutine must be altered. Initial entry to the subroutine occurs with DF=0 and 62M2 in the AC, following a TAD (CDF MAX) at 12675 and an IAC at 12676. The reverse change occurs at 17416 where the content of 17576 (currently 6222) is loaded before calling the subroutine.

A listing of the subroutine is given below. In general the modifications required will be trivial. In our case it is more convenient to enter with a CDF instruction in the AC, hence the IAC at 12676 is changed to a NOP, and the 6222 at 17576 is changed to 6221. The three addresses at 17561-3 have to be altered to those appropriate to the new handler.

2. BASIC

The approach here is somewhat similar, except that BLOAD.SV tests in three places whether TD8E is the system device by examining the first word in the device control word table. Bits 0-2 and 9-11 are masked out, and 7570 (2's complement of 0210, since the TD8E device code is 21) is added. Locations 567, 2561 and 7374 of BLOAD.SV must be changed to contain the 2's complement of ONNO, where NN is the new system device code.

If TD8E is found to be the system device, BLOAD then tests in addition that the content of 7642 is 6223. Locations 566, 2560 and 7373 contain the address 7642, and 565, 2557 and 7372 contain 1555, the 2's complement of 6223.

The field 0 instruction changes are done by one or other of routines which start at 500 and 7256 and, as before, deposit 62x3 in 7642, and 62x2 in 7721 and 7727. Restoration is done by the same routines but an alternative restoration routine, which appears to be used in some circumstances, exists in the BASIC.FF overlay. This starts at 14561, and simply places a 6223 from 14573 in the address contained in 14574 (7642), and a 6222 from 14575 in the addresses contained in 14576 and 7 (7620 and 7655).

With appropriate patches in these locations BASIC runs, and returns to OS/8, as expected, but I cannot guarantee to have found all the problem areas!

```
/FRTSM                                          PAL8-V9H  04/07/76  PAGE 1

                    /FRTSM
                    /
                    /FRTS SUBROUTINE WHICH MODIFIES PAGE 1 INSTRUCTIONS
                    /IN THE 2-PAGE TD8E SYSTEM HANDLER
                    /
                    /
            0001    FIELD 1
            7526    *7526
17526       0000    MOD,        0                   /ENTER WITH 62X2 IN AC, DF=0
                                                    /X IS MAX AT START, 2 AT END
17527       3763                DCA I AD1           /62X2 IN 7721
17530       1763                TAD I AD1
17531       3762                DCA I AD2           /AND IN 7727
17532       1763                TAD I AD1
17533       7001                IAC
17534       3761                DCA I AD3           /62X3 IN 7642
17535       5726                JMP I MOD
            7561    *7561
17561       7642    AD3,        7642
17562       7727    AD2,        7727
17563       7721    AD1,        7721
                    $
```

From JOHN COWAN **OS/8 NEWSLETTER** SUBMISSION

I am currently engaged in work on a SNOBOL semicompiler for OS/8. The language will be Bell Labs (informal) subset SNOBOL IV, with modifications for the PDP-8 and OS/8. Full OS/8 interface including device independence and core image generation is provided. The package is rather unusual in that it is only one core image as loaded by the Monitor; sections loaded into field 1 are then saved on the scratch blocks and loaded as overlays; in fact, all of the interpreter except the parts shared with the compiler (a goodly section) is technically an overlay. The compiler is unusual in compiling into absolute core locations a la IBM 360 WATFOR and producing no compiled-binary output files. Flowcharting and documentation are complete--only the coding and testing need to be done (alas!). Any information or assistance would be greatly appreciated from anyone who is fed apropos as I am) with PDP-8 high-level string handling.

Apropos of nothing much, has anyone given thought to some sort of CCL command for calling non-Command Decoder programs? I propose .DO filnam arg when filnam is a .SV on SYS: to be chained to, and arg is a string of characters to be placed in the Decoder's output buffer, one character to a word. Currently only TECO and CCL itself use this sort of call; however, if it were generally available, user programs might be written or modified to use it. In the latter category I think particularly of U/W FOCAL and PS/8 LISB, neither of which may now be called by BATCH even for noninteractive jobs due to their total terminal orientation. Comments?

PDP8 SOFTWARE PLANNING

Submitted by E. Lopes Cardoso

I would like to make a few remarks on possible software developments for the PDP8.
To start with the operating systems, I see the following deficiencies in OS/8 (in order of importance):

1. single-user only
2. cannot support large file structured devices
3. does not allow multiple open output files in a single directory
4. restricted to 15 devices (device-units!)
5. filestructure does not allow realtime file acces by independent processes

One could imagine a project to extend the RTS/8 background to a time-sharing system. Unfortunately, RTS/8 is not well suited for such operations, for instance because the scheduling strategy does not allow re-entrant tasks. Anyhow, such a project would be wastage as there are at least two OS/8-timesharing packages available from aother sources (MULTI8 and ETOS).

It should be possible to rebuild the file-managing part of OS/8 to aleviate problem 2-5. Such a system would have "channels" much like RT/11, each channel being associated with a file on a certain device. In that case a file can be up to 4096 blocks long, and, provided the directory structure allows it, very large devices could be supported.
Most of the CUSPS would have to be changed very little, only PIP, FOTP and DIRECT are really sensitive. Such developments are largely dependent on future hardware policies.
One might think of large disk drives (RP03 etc.) and a "memory management" option, doing things like field relocation, untrapped execution of CDF in User Mode, etc. A proper kind of hardware protection unit for the PPP is another one.

In the languages area I would like two things:

1. a relocatable-macro assambler (MACREL)
2. a really good implemented high level language.

Currently we have BASIC (well implemented, but an awfull language to program), FORTRAN II (spoiled by SABR) and FORTRAN IV (I cannot be enthousiastic about integers that are managed as if they were reals). The new language should preferably have the possibility to specify in the source whether a certain integer should be 12 or 24 bits. One might think of PASCAL or a subset of ALGOL 68. But perhaps a working MACREL will open the door to an improvement of FORTRAN II.

Concluding I would vote for the MACREL project. I think it is the one that affects most of the users and could have a far reaching effect on other software projects as well.

STANDARD IOT-CODES FOR FOREGROUND/BACKGROUND SYSTEMS

Submitted by E. Lopes Cardozo

The following is another proposal for the standardisation of IOT-codes used in foreground/background systems. It is largely based on the MULTI-8 system. The general idea is to differentiate between three types of IOT's:

1. IOT's that can be and mostly are truly emulated. Among these the CDF, terminal IOT's, etc.

2. IOT's that cannot be emulated, but may be expected to occur in some programs.

3. IOT's that should not occur in normal PDP8-programs and get a special meaning assigned by the monitor system.

To ensure the utmost compatibility with e.g. regular OS/8 user programs I propose to keep group 3 as small as possible.
Many installations have non-standard interfaces to all sorts of application-dependent devices. Some of these can be serviced by a fairly simple IOT-emulator that just executes the trapped IOT. Such applications should not get into conflict with some funny monitor IOT.
On the other hand group 3 should be large and flexible enough to implement any local specialties. In our MULTI-8 system a number of important functions are activated via special IOT's:

- an analog sample task that samples a given number of channels with given frequency and writes the data to an user-defined diskfile.

- a task that supports a so called 200 User Terminal (remote batch) communications line with our university's CYBER 73.

- various simple things like digital I/O etc.

The solution to these apparently conflicting needs is to designate one IOT (in MULTI8 called the GIANT IOT) and encode all functions in the users AC. This opens the door to 4096 unique monitor functions. For some reason our GIAT IOT has code 6770, but that is not part of my proposal here.

I have objections to monitor functions line KSR (read Keyboard String) and SAS (Send A String). I think the multi-user systems should be kept as much compatible with regular OS/8 as is possible. I do not think extensions like these do increase system performance noticable.

For reference I include a copy of the MULTI8 IOT list. As you can see the number of monitor functions is very modest. Almost everything (like device-sharing) is fully automatic and unnoticable.

IOT-list for MULTI8 background

6000 Call block driver emulator. used by the fakehandler to pass parameters from a handler call to the forground.

```
TAD (0CDU       /D=DEVICE TYPE, U=UNIT
6000
JMP .+4         /JUMP OVER PARAMETERS
FUNCTION        /JUST LIKE OS/8 HANDLER CALL
BUFFER          /
BLOCK           /
RETURN          /AC=0 OR 4000 (=ERROR)
```

6001 IOH; Invalid instruction for MULTI8

6002 IOF; A no-op for MULTI8

6003-6005 Error

6006 SGT; If EAE is present, Skip on Greater-Than flag

6007-6200 Error

601X Reader IOT's

602X Puncher IOT's

603X Keyboard IOT'S

604X TELEPRINTER IOT's. 6041 will always skip.

62N1 CDF N; Change data field to field N if field N is available; Otherwise no-op.

62N2 CIF N; Change instruction field to N id field N is availble; Otherwise no-op

62N3 CDF CIF N

6254 SM8; Skip-on-MULTI8

6264 Look-into-forground; Delivers a word from the foreground memory into the users AC. This IOT should be followed by a CDF to the real field that must be looked into. The address within that field is specified in the AC.

666X Lineprinter IOT's. 6661 is changed to SKP.

From V.J. BLACKMORE (CHRISTIE HOSPITAL - ENGLAND)

OS/8 Modified to use a KV8 Display

OS/8 has been designed so that all commands input to the keyboard monitor or command decoder are echoed at the console terminal using KL8-E type instructions. This paper describes how a non-standard device that requires a software character generator, such as a KV8 visual display unit may be used as an alternative output device.

The character generator used for the KV8 requires three pages of core and, in order to make this routine 'invisible' to the user, it is held as a disk resident overlay and swopped into core as required. A shorter routine to perform this swopping permanently resides in the top page of field 2 (or any higher field designated by the user) and is considered as part of the core-resident OS/8 read in when the bootstrap is run. The monitor, command decoder, ODT and some system programs have been patched to jump to this routine whenever they have a character to be displayed and the character generator is read into core if required; the contents of core overlayed by the character generator are saved on system scratch blocks. In a similar way the standard device handler TTY has been modified to use the KV8 but with characters half the size of those used for monitor commands.

The speed of the editing program, EDIT, has been considerably improved by these modifications and additional facilities such as the display of line numbers, high speed reader append and insert commands and the ability to define macro commands have also been added.

The majority of users of the system use Fortran II which uses its own input/output routines and not the system device handlers. Additional routines have therefore been added to the library such that alphanumerics of any size can be output on the KV8 and on an incremental plotter, using standard Fortran I/O statements. Additional subroutine calls enable the user to draw graphs on these devices.

Similar subroutines written in RALF code may be used with Fortran IV but because of the limited size of the system there are incompatabilities between the Fortran run-time system and the modified OS/8 which means that the KV8 device handlers cannot be used at run-time. It is hoped that this problem will be overcome in the near future. —— it has to a certain degree !

The log program has two main functions:

(a) To log users on and off the machine. The user uses the program through CCL calls LOG ON and LOG OFF and the program maintains a file containing details of the user's initials, department, date and time. LOG changes the resident monitor coding such that the next jump to 76ØØ after LOG OFF will automatically recall the log program for the next user.

(b) To allow each user access to his own disk partition.
In order to avoid directory overflow problems on our RK8E we have split the disk into a number of files (known as .SY files) and each file may be used as a separate device with its own directory. The log program asks the user which file is required, does a look-up on the file and sets up parameters in a special 2-page handler, MULT. MULT is not a true device handler in that it must use another handler to actually perform the transfer. Usually this is the resident SYS or RKBØ but it can be a non-resident handler in which case the required handler is held in the empty second page. MULT has up to eight entry points LSKØ-7 and therefore up to eight files may be 'logged on' to at any one time. In our system DSKØ, the default entry to MULT, is assigned as DSK, and each user once he has logged on is usually only aware of SYS, holding F4 etc., and his own programs on DSK.

Block Ø of every .SY file may be used, if required, to hold a password to ensure only authorized access to that file, a list of authorized users against which the users initials are checked and the option to set MULT, for that DSK, to read only.

The log program also contains a number of 'maintenance' routines. For example, creating .SY files and examining the log data. These routines allow the user to readily set up and check the system.

SPR #8-1893 BUG IN OS/8 V3C RELEASE OF BLOAD

PROBLEM: WHENEVER BASIC OR BCOMP COMPILES A PROGRAM WITH ERRORS ON A MACHINE THAT HAS THE SOFTWARE CORE SIZE SET, LOCATION 07600 IS CHANGED TO AN INCORRECT VALUE WHICH LATER CRASHES THE SYSTEM.

RESOLUTION: THE PROBLEM IS CAUSED BY AN ERROR IN BLOAD AT THE POINT WERE IT RESTORES 07600 TO THE NORMAL VALUE. WHEN THE SOFTWARE CORE SIZE WORD IS NON-ZERO AN INCORRECT VALUE IS RESTORED. THE FOLLOWING PATCH CORRECTS THE PROBLEM.

```
.GET SYS: BLOAD
.ODT
2315/ 1371  5363
2363/ XXXX  7200
2364/ XXXX  1371
2365/ XXXX  5316
3030/ 0140  0240
^C
.SAVE SYS: BLOAD
```

THIS PATCH UPGRADES BLOAD FROM V4A TO V4B.

TECO PATCHES FROM STUART DOLE

```
1331/4515  5214          (KILLS ECHO OF ^T COMMAND)

1332/1035  1037          (SET VALUE OF EOF FLAG; 0 IF EOF)

3764/XXXX  3024;         (CHAR TYPEOUT - KILL NUMBER FLAG)
           1027;         (GET ARGUMENT)
           4515;         (TYPE IT)
           5506          (POPJ - RETURN)

5002/0011  1332          (^B COMMAND - VALUE OF EOF FLAG -
                          4095 IF FILE OPEN, 0 IF EOF SEEN)

5126/0011  3764          (V COMMAND - COULD BE ANY OTHER CHAR
                          WITH 0011 VALUE. TYPE ARG AS ASCII ON
                          CONSOLE)
```

THESE ALLOW THINGS LIKE: <PB;>$$ - EMPTY INPUT FILE, BRANCH OUT ON EOF. THE ECHO ON THE ^T IS A BOTHER WHEN WRITING INTERACTIVE MACROS..IE: A MACRO TO GO THROUGH AN UNCOMMENTED SOURCE FILE AND ALLOW EASY INSERTION OF COMMENTS:

```
!A!
L 2R 0T
!B!
^T VA
QA-13 "E L QA$'
QA-127 "E R 0AV D QB$'
QA V
QA I $
QB$
```

STUART NOTES HE HAS A BETTER EXAMPLE THAT DOES LINE EDITING LIKE

FOCAL.
HE ALSO NOTES TROUBLE WITH THE BCOMP /K SWITCH SEVERAL OTHERS HAVE ALSO. I AM WAITING TO SEE IF THE RECENTLY PUBLISHED PATCHES FIX THE TROUBLE AND IF IT HAS BEEN FIXED IN THE NEW RELEASE BEFORE PRINTING ANY MORE ON THE SUBJECT.

PATCHES FOR DIRECT (OS/8 V3,V3B,V3C) FROM STEVE LIGETT

1. ALLOWS =N OPTION IN RANGE OF 0 TO 13

```
12171/7770  7765
```

2. SQUISHES DIRECTORY TOGETHER ON A LISTING

```
13160/0005  0003
13750/0005  0003
```

3. REMOVES LENGTHS OF EMPTIES FROM /E/F

```
13712/6201 1767
13713 /1426  0374
13714 /6211  7640
13715 /7041  5325
13716 /4754  6201
13717 /2753  1426
13720 /5325  6211
13721 /4752  7041
13722 /1751  4754
13723 /3753  1347
13724 /5641  4763
13725 /1350  2753
13726 /4763  5333
13727 /1767  4752
13730 /0374  1751
13731 /7640  3753
13732 /5641  5641
13733 /1347  1350
13734 /4763  4763
```

4. CHANGE DEFAULT NUMBER OF COLUMNS =N TO =3

```
12305/7040  1747
12306 /1747  7450
12307 /7500  7125
12310 /7040  7041
```

AN OS/8 BASIC MARGIN FUNCTION FROM STEVE LIGETT

PROBLEM: USERS OFTEN WANT TO CREATE FILES OR TERMINAL OUTPUT WITH A RECORD LENGTH GREATER THAN 72 CHARACTERS.

DISPOSITION: SUPPLIED HERE IS A MARGIN FUNCTION (MAR) THAT CAN BE INSTALLED IN THE BASIC USER FUNCTION OVERLAY "BASIC.UF". A CALL TO THIS FUNCTION ALLOWS A USER TO USE DIFFRENT RECORD LENGTHS IN DIFFRENT PROGRAMS, AND TO CHANGE WITHIN A PROGRAM.

STEP 1. INSTALL THE FOLLOWING PATCH TO BRTS.

```
.GET SYS: BRTS

.ODT

2436/1313  1713
2513/7670  3252
2533/1360  1573
02534 /3014  1361
02535 /1573  7510
02536 /1414  5347
02537 /7510  7450
02540 /5347  5513
02541 /7650  2007
02542 /5513  5334
02543 /2007  7200
02544 /5335  7000

2566/7774  7770
3252/7670  7600

^C
.SAVE SYS. BRTS
```

THIS SETS BRTS UP TO USE 9 COLUMNS, 128 CHARACTERS PER RECORD (LOCATIONS 2566 AND 3252).

STEP 2. CREATE THE FOLLOWING FUNCTION. THIS CAN BE ORIGINED TO 3400, IF IT IS THE ONLY USER FUNCTION NEEDED, OR IT CAN BE ADDED TO OTHER FUNCTIONS THE USER HAS WRITTEN.

```
/MARGIN FUNCTION FOR OS/8 BASIC

ORG=3400        /OR ANYWHERE WITHIN THE USER FUNCTION AREA
        *ORG
MAR,    0               /MARGIN UDEF
        JMS I   INTL    /FIX ARGUMENT
        CIA             /NEGATE
        DCA I   LWIDTH  /AND STORE
        TAD     D14     /DIVIDE BY 14 FACTOR
        JMS I   MPY     /MULTIPLY
        STA             /LOAD A -1
        TAD     TEMP6   /HIGH ORDER PRODUCT=# OF COLUMNS
        DCA I   MCOMMA  /STORE AS COMMA COUNT
        JMP I   MAR     /RETURN
LWIDTH, 3252            /ADDRESS OF -LINE WIDTH
MCOMMA, 2566            /ADDRESS OF # OF COLUMNS(-1)
D14,    445             /  4096/14

TEMP6=50                /TEMP AREA USED IN MPY
MPY=121                 /POINTER TO MPY
INTL=114                /POINTER TO FIX RTN
        $
```

STEP 3. RENAME THE LAB 8/E FUNCTIONS TO SOMETHING OTHER THAN "BASIC.UF"; ASSEMBLE THE FUNCTION; AND CREATE THE USER OVERLAY

```
.RENAME BASIC.LF<BASIC.UF
.PAL MARGIN<MARGIN
.LOAD MARGIN
.SAVE SYS: BASIC.UF 3400-4577
```

STEP 4. INSERT THE ENTRY POINT INTO BRTS (OS/8 HANDBOOK, PAGE 6-125).

```
.GET SYS: BRTS
.ODT

1560/0240  3400

^C
.SAVE SYS: BRTS
```

STEP 5. TEST THE FUNCTION. THE FOLLOWING PROGRAM DEMONSTRATES THE USE AND ACTION OF THE "MAR" FUNCTION.

```
.R BASIC
NEW OR OLD -- NEW MARTST
READY
100 UDEF MAR(D)
110 PRINT "ENTER TERMINAL WIDTH";
120 INPUT W
130 FOR M = 14 TO W STEP 14
140     D = MAR(M)
150     C = M / 14
160     FOR N = 1 TO C
170         PRINT N,
180         NEXT N
190     PRINT 0;"<--RETURN"
200     NEXT M
210 FOR M = 4 TO W
200     D = MAR(M)
230     PRINT M;
240     NEXT M
250 PRINT
260 GOTO 110
999 END

RUN

MARTST  BA    4A    10-MAR-76

ENTER TERMINAL WIDTH?84
 1
 0 <--RETURN
 1              2
 0 <--RETURN
 1              2              3
 0 <--RETURN
 1              2              3              4
 0 <--RETURN
 1              2              3              4              5
 0 <--RETURN
 1              2              3              4              5              6
 0 <--RETURN
 4
 5  6
 7  8  9
 10  11  12
```

```
 13  14  15  16
 17  18  19  20  21
 22  23  24  25  26  27  28
 29  30  31  32  33  34  35  36  37
 38  39  40  41  42  43  44  45  46  47  48  49
 50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65
 66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84
ENTER TERMINAL WIDTH?^C
READY
BYE
```

NOTE ON OS/8 SYSTEM BOOTSTRAP FROM STEVE LIGETT

ADEQUATE EXPLANATION AND DOCUMENTATION EXIST IN THE OS/8 HANDBOOK (DEC-S8-OSHBA-A-D), AND THE OS/8 SOFTWARE SUPPORT MANUAL (DEC-S8-OSSMB-A-D), AND THE KL8E SOURCES TO ENABLE ONE TO WRITE AN OS/8 NON-SYSTEM HANDLER. IN ATTEMPTING TO WRITE A SYSTEM HANDLER, HOWEVER, WE FIND A CRUCIAL PIECE OF INFORMATION IS MISSING: WHAT DOES A BOOTSTRAP LOOK LIKE, AND WHAT DOES IT DO?

THERE ARE TWO BOOTSTRAPS FOR EACH DEVICE: THE PRIMARY BOOTSTRAP THAT IS TOGGLED IN, AND THE SECONDARY BOOTSTRAP THAT IS CODED WITH THE SYSTEM HANDLER. THE PRIMARY BOOTSTRAP IS WRITTEN TO BE AS EASY TO ENTER AS POSSIBLE AND MOST OF THE WORK IS GENERALLY LEFT TO THE SECONDARY BOOTSTRAP. HERE ARE EXPLANATIONS OF THE BOOTSTRAPS FOR A ONE PAGE OR A TWO PAGE SYSTEM HANDLER.

FOR A ONE PAGE HANDLER, BLOCK 0 OF THE SYSTEM CONTAINS:

BLOCK 0 (ADDRESSES REFER TO RESIDENT MONITOR LOCATIONS)

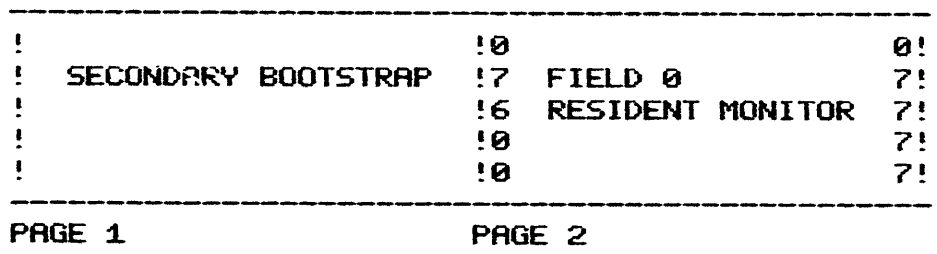
```
!1              1!1             1!0                     0!
!7              7!7             7!7                     7!
!6  SECONDARY   6!6  FIELD 1    7!6  FIELD 0            7!
!0  BOOTSTRAP   4!4  RESIDENT   7!0  RESIDENT MONITOR   7!
!0              6!7  MONITOR    7!0                     7!
```

PAGE 1 PAGE 2

THE PRIMARY BOOTSTRAP READS BLOCK 0, WHICH INCLUDES THE SECONDARY BOOTSTRAP. THE SECONDARY BOOTSTRAP MUST FIT WITHIN THE COMMAND DECODER TABLE AREA AND MUST BE 47 (OCTAL) LOCATIONS OR LESS. THE BOOTSTRAP MUST ALSO BE AT LEAST 21 (OCTAL) LOCATIONS. BLOCK 0 ALSO INCLUDES THE COMPLETE RESIDENT OS/8 MONITOR AS SHOWN ABOVE. THE SECONDARY BOOTSTRAP MOVES THE FIRST PAGE TO THE TOP OF FIELD 1 AND MOVES THE SECOND PAGE TO THE TOP OF FIELD 0. THEN THE BOOTSTRAP BRANCHES TO LOCATION 07605, THE DESTRUCTIVE ENTRY TO OS/8.

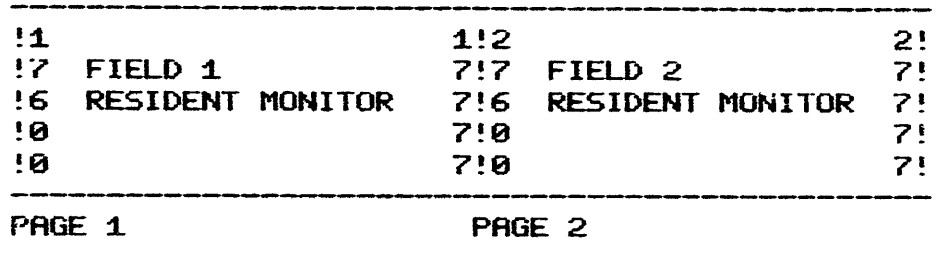
FOR A TWO PAGE HANDLER, BLOCK 0 CONTAINS:

BLOCK 0



IN ADDITION, THE CONTENTS OF BLOCK 66 (OCTAL) ARE DEFINED:

BLOCK 66



THE PRIMARY BOOTSTRAP READS BLOCK 0, INCLUDING THE SECONDARY BOOTSTRAP. THE SECONDARY BOOTSTRAP OCCUPIES THE ENTIRE FIRST PAGE OF BLOCK 0. THE SECOND PAGE CONTAINS THE FIELD 0 RESIDENT MONITOR. THE SECONDARY BOOTSTRAP READS BLOCK 66 (OCTAL). THIS CONTAINS RESIDENT PORTIONS OF THE MONITOR FOR FIELDS 1 AND 2. THE BOOTSTRAP THEN MOVES ALL THREE PAGES OF THE RESIDENT MONITOR TO THE TOP OF THE PROPER FIELDS AND BRANCHES TO LOCATION 07605.

NOTE ON BATCH LOG ON LA-36 FROM STEVE LIGETT

PROBLEM: OS/8 BATCH NORMALLY OUTPUTS A LOG TO THE LINEPRINTER USING AN INTERNAL LINEPRINTER HANDLER. INITIALIZATION CODE DETERMINES WHETHER OR NOT A LINEPRINTER EXISTS. THIS CODE WILL NOT RECOGNIZE A SLOW (TERMINAL) DEVICE AS A LINEPRINTER. USERS WITH "CLASSIC" TYPE CONFIGURATIONS OFTEN HAVE AN LA-36 FOR A LINEPRINTER AND MAY WANT TO USE IT FOR THE BATCH LOG.

DISPOSITION: THE FOLLOWING PATCH CAUSES BATCH TO ASSUME THAT A LINEPRINTER EXISTS. THE /T SWITCH MAY BE USED TO DIRECT THE LOG TO THE TERMINAL.

NOTE: IF THE LINEPRINTER IS ATTACHED VIA A SERIAL INTERFACE, BATCH WILL BE UNABLE TO DETERMINE IT IS OFFLINE.

```
.GET SYS: BATCH
.ODT
310/5320  7000
^C
.SAVE SYS: BATCH
```

TO USE A DEVICE OTHER THAN 66 AS THE LINEPRINTER, INSTALL THE FOLLOWING PATCH.

```
.GET SYS: BATCH
.ODT
```

```
6526/6666  6XX6
6531/6661  6XX1
^C
.SAVE SYS: BATCH
```

INSERT THE DEVICE CODE OF THE DESIRED DEVICE IN PLACE OF XX ABOVE (E.G., XX=04 WILL OUTPUT THE LOG ON THE CONSOLE).

NOTE ABOUT BAT: FROM LARS PALMER

AS YOU PROBABLY KNOW THERE ARE TWO "SPECIAL" HANDLERS IN THE OS/8 SYSTEM, THE "BAT" AND "NULL" HANDLERS. THE "NULL" HANDLER WAS DRAWN ATTENTION TO IN THE DIGITAL SOFTWARE NEWS RECENTLY BUT THE "BAT" HANDLER HAS BEEN DISCUSSED VERY LITTLE. THE OS/8 DOCUMENTATION SAYS THAT IT CAN ONLY BE USED IN THE NON FILE STRUCTURED BATCH INPUT DEVICE BUT THIS IS NOT TRUE. ANY PROGRAM THAT DOES NOT TOUCH THE BATCH FIELD MAY CALL THE "BAT" HANDLER WITHOUT RESTRICTIONS. FORTRAN IV AND BASIC ARE DIFFICULT AS THEY CREATE HAVOC IN THE BATCH FIELD. IF YOU CAN AFFORD TO RESERVE SPACE IN CORE TO LEAVE THE BATCH MONITOR UNDISTURBED WHEN USING FORTRAN IV THE FOLLOWING SMALL PATCH TO THE RUN TIME SYSTEM PRESERVES BATCH (NOTE THAT WITH THE LOGIC IN FRTS THE SAME CONSERVATION OF CORE WILL BE DONE AS FOR AN IN-CORE TD8E SYSTEM). CHANGE LOCATION 12744 TO 2000 (WAS 4600) AND CHANGE LOCATION 12745 TO 3000 (WAS 200).
(NOTE: I HAVE TESTED THIS WITH THE V3C RELEASE OF FORTRAN IV (V4) AND IT SEEMS TO WORK OK - RH)