RUNCOBOL by all the COBOL users represents a great deal of time!

In case you wonder how I got the password protected RUNCOBOL and RSCOBOL accessed, I'll tell you. I just read the TRSDOS diskette's directory using the LDOS extended debugger and changed the password and attributes to enable the access.

Another use of the packing function is to generate a load module with sequential load records. Non-sequential load record files cannot be properly disassembled by DSMBLR/PRO-DUCE with data screening. Therefore, by processing such a file with CMD-FILE/PRO-CESS, it's ready for disassembly.

Now that I have given you a little lead-in to some uses of this load module maintenance package, I can shift over to supplying two patches. These patches are only for PRO-CESS, not the Model I/III CMD-FILE version.

```
. PROCESS1/FIX - 10/13/83 - Applied 520021
. Corrects cursor UP/DOWN after BREAK
. and MAP printing after printer error.
D00,DD=98
F00,DD=74
D15,E9=F5 CD 74 45 F1 C9
F15,E9=00 00 00 00 00 00
D07,27=D9
F07,27=D1
. End of patch

. PROCESS2/FIX - 10/27/83 - Applied 520021
. This patch corrects the END address
. after an IMAGE LOAD invocation
D02,3C=9E 45
F02,3C=E4 34
D15,EF=06 00 C3 E4 34
F15,EF=00 00 00 00 00
. This patch corrects PACK of X-patches
. where the last byte of a load record
. is being patched.
D09,9C=2F 6F 23
F09,9C=ED 44 6F
. End of patch
```

CON80Z/PRO-CON80Z

Bernd Jung of Dusseldorf, West Germany reports a few problems with the CON80Z program that apparently slipped by our testing. Specifically, Bernd discovered that: 1) "CNC XYZ" was not decoded to "CALL NC,XYZ"; 2) "CPO XYZ" was not decoded to "CALL PO,XYZ"; 3) "ACI n" was decoded to "CP A,n" instead of "ADC A,n"; 4) "RST n" was passed unchanged instead of converting "n" to "p"; and 5) "HLT" was not decoded to "HALT" but was left as "HLT".

Two of the five bugs can be fixed by patches; however, since three of the five cannot, I decided to correct my source and reassemble. Therefore, MISOSYS is now shipping CON80Z version 1.1 and PRO-CON80Z version 1.1. If you

are currently in possession of a 1.0 version vopy of CON80Z and wish to obtain the corrected version 1.1, please return your master diskette in a protective mailer. There will not be a charge for the update to your disk.

CONVCPM/PRO-CURE

A patch is needed to CONVCPM version 1.3. It corrects a problem when using CONVCPM with high-memory drivers. The patch is as follows:

```
PATCH CONVCPM (D01,13=03)
```

The address in question is X'5303' and the old value was an X'01'.

Our new catalog presents information on CONVCPM/PRO-CURE version 2. This release expands the supported CP/M media types. If you currently have the CONVCPM version 1.x and want to upgrade to the CONVCPM version 2.x, please return your CONVCPM master diskette with $20. A completely new package will be returned to you. Note that there is no offer to upgrade from CONVCPM 1.x to PRO-CURE.

DSMBLR/PRO-DUCE

We have had so many positive reports on version III of our disassembler that I am glad to have taken the time to enhance the DSMBLR II package. You may be aware that the DSMBLR is one of the first products ever released by MISOSYS. The original version I was for Model I cassette machines. A version was done for the Exatron stringy floppy. We expanded DSMBLR to version 2.0 and added disk source file output. I believe it was version 2.2 which supported Model III. The third version supports disassembly direct from disk - a function that we received so many requests for. We went beyond just disk input and added a screening mode to provide user definition of data areas. Thus, Version III provides for the specification of literals (messages and other strings), words (DEFWs/DWs for the hackers), and bytes (DBs and DEFBs again for the hackers).

Version III is supported under TRSDOS/LDOS 6.x under the product name PRO-DUCE - since it produces assembly code. The Model I/III compatible product called DSMBLR III was designed to be fully compatible with LDOS 5.0, LDOS 5.1, TRSDOS 2.3 (Model I) and TRSDOS 1.3 (Model III). DSMBLR probably runs under other Model I/III operating systems; however, we only guarantee it under LDOS and TRSDOS (sorry, we don't support anything under TRSDOS 2.7DD).

The disassembler is a powerful product. Although there are still some features that you may want (such as user-defined labels and online entry of screening ranges), I feel it is a bargain at the $40 price. As good as the disassembler is, a bug or two still found its way past our testing. Therefore, a couple of patches follow. The patch syntax shown is for Model I/III LDOS or TRSDOS 6.x (in the case of PRO-DUCE). For TRSDOS 1.3 PATCH, convert the "Dxx,Byy=zz zz zz" syntax to "ADD=aaaa,FIND=bbbb,CHG=cccccc" using the "zz zz zz" for "cccccc", the values shown as WAS for "bbbb", and the address shown as @X'hhhh' for "aaaa". DSMBLR3x refers to Model I/III

[note that DSMBLR31 and DSMBLR32 were applied prior to release] whereas PRODUCEx refers to the PRO-DUCE version:

```
. DSMBLR33/FIX - Applied starting with 530099
. Patch corrects command entry of parameters w/o filespec
D00,D5=00 00; WAS D6 0D @X'5491'
D09,95=C3 3E 67; WAS C2 7B 5E @X'5D2D'
D13,CE=FE 41 DA 30 5D C3 7B 5E; Were zeroes @X'673E'
. End of patch

. PRODUCE1/FIX - Applied starting with 530013
. Patch corrects command entry of parameters w/o filespec
D00,97=00 00
F00,97=D6 0D
D09,3C=C3 97 43
F09,3C=C2 34 3B
D12,1F=FE 41 DA DB 3A C3 34 3B
F12,1F=00 00 00 00 00 00 00 00
. End of patch

. DSMBLR34/FIX - Applied starting with 530310
. Patch corrects swapping disks when output disk is full
D11,60=C3 46 67; WAS F6 01 C9 @X'64DC'
D13,D6=11 BE 73 F6 01 C9; Were zeroes @X'6746'
. END OF PATCH

. PRODUCE2/FIX - Applied starting with 530080
. Patch corrects swapping disks when output disk is full
D0F,CD=C3 9F 43
F0F,CD=F6 01 C9
D12,27=11 34 4F F6 01 C9
F12,27=00 00 00 00 00 00
. END OF PATCH
```

The following patch is derived from an X-patch that was placed on the LDOS Compuserve bulletin board. The name of the contributor had scrolled off the display before I was able to SCREEN PRINT the text so I cannot give credit where credit is due. I can suggest that if you have developed any patch to a MISOSYS product, why not send it to us so that we may evaluate it. Perhaps there is a better way of accomplishing the desired result. Perhaps we can at least give the other users a chance at implementing your change. In any event, I rewrote the patch to shorten it a little and converted it to a DIRECT patch. It now is official and uses a small portion of the patch area I reserved in the disassembler product.

```
. DSMBLR35/FIX - 10/06/83 - APPLIED 530409
. CORRECTS BUG WHEN A LITERAL SCREENING RANGE
. IS SPECIFIED THAT INCLUDES A LABEL EQUATE
. OF NEGATIVE DISPLACEMENT GREATER THAN 9
.
D09,33=CD 4C 67
. WAS  CD 47 5C @X'5CCF'
D13,DC=FE 3A 38 08 F5 3E 31 12 13 F1 D6 0A C3 47 5C
. WAS  00 00 00 00 00 00 00 00 00 00 00 00 00 00 @X'674C'
. END OF PATCH
```



```
. PRODUCE3/FIX - 10/06/83 - APPLIED 530080
. CORRECTS BUG WHEN A LITERAL SCREENING RANGE
. IS SPECIFIED THAT INCLUDES A LABEL EQUATE
. OF NEGATIVE DISPLACEMENT GREATER THAN 9
.
D08,E3=CD A5 43
F08,E3=CD F7 39
D12,2D=FE 3A 38 08 F5 3E 31 12 13 F1 D6 0A C3 F7 39
F12,2D=00 00 00 00 00 00 00 00 00 00 00 00 00 00
. END OF PATCH

. DSMBLR36/FIX - 11/21/83 - Applied 530486
. Corrects possible crash on excess interstitial labels
D09,38=CD; Was BF @X'5CD4'
. End of patch

. PRODUCE4/FIX - 11/21/83 - Applied 530100
. Corrects possible crash on excess interstitial labels
D08,E8=CD
F08,E8=BF
. End of patch
```

Now that the buggy patches have been identified, let me turn to some improvements. Bernd Jung is one of my devoted users in West Germany. I sometimes feel that I know the guy personally having communicated so frequently via letters. In any event, Bernd does a lot of work with the 8085 chip. This chip is an improved 8080. The disassembler can handle 8085 code to produce Z-80 mnemonics [not that useless of a result as you may imagine]. The only problem is that the 8085 uses the op codes 20H and 30H for the RIM and SIM instructions [Read Interrupt Mask and Set Interrupt Mask]. These op codes are used in the Z-80 as two-byte instructions. Thus, DSMBLR would cough a little on RIM and SIM instructions.

To help Bernd out of this limitation, I worked up a little patch to DSMBLR III to support the 8085 [actually I had to work it up twice since the first patch created a bug in the disassembly of two other instructions: "LD (nnnn),HL" and "LD (nnnn),A". If I get sufficient requests, I'll try to work up such a patch for PRO-DUCE [or maybe yet someone who has both DSMBLR and PRO-DUCE could work up the patch]. The change is shown in assembly source code form. It is left to the reader as an exercise to convert it to a patch. Don't forget, the Model 100 uses an 8085 compatible chip!

```
;D8085
MOVOP   EQU     5A3BH
SHFT3   EQU     572DH
        ORG     6BB8H
        DB      '1'       ;Change 20H & 30H
        ORG     56E9H
        CALL    PATCH+1   ;Patch single-byte OPs
        ORG     5A9DH
PATCH   RET               ;Mask B2LBL
        CP      20H       ;Is it RIM?
        LD      HL,SIM$   ;Init to SIM string
        JR      Z,DORIM   ;Go if RIM
```

```
        CP      30H     ;Test for SIM and do it
        JR      NZ,MOVOP ;Else do the original code
        DB      0C2H    ;Ignore next inst
DORIM   LD      L,RIM$.AND.0FFH
        POP     AF      ;Clean the stack
        JP      SHFT3
RIM$    DB      'RIM'
SIM$    DB      'SIM'
        END
```

Some people are never happy with the designer's choice of expression. That's why we always want to customize canned software. Bernd was unhappy with our choice of "M" as the label prefix [we actually got complaints about the letter "Z" used in DSMBLR II so I changed to an "M" for MISOSYS]. To satisfy those that want their own character, you can patch DSMBLR III at X'5AE5' with the character of your choice - make it a character acceptable to your assembler. The corresponding PRO-DUCE location is X'3895'. The value is currently an X'4D' [note: giving addresses helps promote the sale of LSI's FED utility which runs circles around all other utilities for zapping load modules].

James A. Sladek, of Norfolk, VA, points out a minor typo in the documentation on page 19. Step 4 of the DEBUG procedure used with Model I TRSDOS should specify the new value as "3C" not "7C" as noted. In fact, the "7C" is what the old value is! While I'm at it, an easier way from BASIC is to just "POKE &H46B0,60". One of these days I might write an article on the entire procedure to convert TRSDOS 2.3 to the Model III data address mark convention.

James must be one of my early DSMBLR users as he mentioned the "infamous TRS-80 printer DCB bug of initialization to 67 lines per page". This pertains to Model I only. To remind the Model I users, the *PR device control block lines-per-page field was initialized by TRSDOS to 67 although the printer driver's counter started from zero. Thus, paper tended to migrate up one line per page (11" paper has 66 lines per page). My old DSMBLR documentation mentioned this and suggested that you poke a 66 into X'4028'. As an aside, Tandy "corrected" the problem on the Model III by keeping the 67 but reindexing the starting value to one. Enough of that! James supplied a patch to Model I TRSDOS SYS0 to have SYS0 correct the value to 66. The values shown are for DISK TRACK, SECTOR and BYTE:

```
T01,S05,BF2=3E 42 32 28 40; WAS 2C 20 00 3E 03
T01,S05,BFD=00; WAS F5
T01,S06,B12=00 00 00; WAS F1 3D 20
T01,S06,B16=00 00 00; WAS 3A EC 37
```

As a final note from James, he wanted more space for the title on disassembler listings. He there worked up a patch to "sacrifice" my title and lengthen the user title to 51 characters. James suggested that I implement his change in a future release; unfortunately, vanity precludes me from doing so. I will share his patch with you. This patch is for DSMBLR III only. If someone wants to work up the equivalent patch for PRO-DUCE, I will publish it in a future NOTES.


```
5FC1 from 0C to 33
5FC3 from 9E to 75
5FCC from 0C to 33
625B from 9C to 75
6874 from 4D to 03
6875-68AA from TEXT to 20
```

Let me share a few responses to letters received concerning the disassembler version III

Ervan Darnell qustioned me about the "MFFFF" label which always seemed to be generated even though the program being disassembled neither referenced X'FFFF' nor had a -1 (X'FFFF') reference as an operand to a 16-bit instruction. The "MFFFF" label is always generated as that value is used to prime the table which collects address references. Its presence in no way affects the integrity of the output nor is it detrimental in any way, short of using one line of a printout.

Ervan also was one of the first to report difficulty in screening a program direct from disk when the program contained load records that were not in sequential order [actually, Robert Newton first identified such a quirk]. The problem, that of not "screening load blocks which load into lower memory than the previous block", is a limitation due to the design of the disassembler. I would venture to guess that 95% of all programs are in sequential load order. The DSMBLR properly handles those 95%. It does this by first sorting your array of screening data then searching the "array" at each logical PC address. If DSMBLR wanted to handle the 5% of programs that are out of sequence, it would have to search the table at each logical PC. This was felt to be inefficient and would occupy considerable time. DSMBLR was programmed to handle the 95% of the cases rapidly! I suspect that the DSMBLR documentation will be tightened a little to reflect on this limitation. As part of this "problem", you are restricted to not have a screening range that is lower than the first logical PC address.

Another problem also stems from a limitation that can be easily rectified by paying careful attention to your screening data ranges. If a file has a sequential but non-contiguous load record, the disassembler by itself properly handles the disassembly and generates another ORG statement. However, if you have specified a screening data range that traverses the boundary between such a load record and its previous record, the disassembler cannot handle it. We may revise the decoding algorithm to handle such a case, but for now, tighten up your screening data. For instance, a program has a message extending from X'7777' through X'7787', a 16-byte gap (perhaps a DEFS 16), then another message from X'7797' through X'779F'. Do not enter one range as "$7777-779F" but specify the two ranges, "$7777-7787,$7797-779F".

As long as I am talking about screening data, let me talk a little on why I came up with the scheme that I did. This discussion is for the programming users. Every one of you knows that you love to tinker with programs that you have purchased. Its difficult, however, to understand every program you want to tinker with. Wouldn't it be great if those that tinkered and wanted to offer their information to others, had a convenient means of doing so. With assemblers as powerful as EDAS/PRO-CREATE, all you need is a good set of source code to make whatever customization you wanted to make. If you have spent the time to disassemble a machine language program and want to

share the results of your efforts, all that is needed is for you to publish your screening data file - which can be placed into the public domain. If you wanted to make it commercially available, you could publish it as a product. Imagine what you could do by creating a set of screening files for FORTRAN, or SCRIPSIT, or PROFILE, or ... With your screening files, the MISOSYS disassembler, and a purchased copy of the product referenced by the files, most individuals could easily alter the canned software product to suit their own desires. The screening data file is a legal means to develop and promote the generation of source code files. Enough said?

EDAS VERSION 4.1/PRO-CREATE

The EDAS/PRO-CREATE column this issue will first start with a correction to a patch published in the last issue. The line in EDAS410/FIX that read D1B,7E=28 08... should have read D1B,7E=28 04... The corrected patch is shown below.

```
. EDAS410/FIX - 01/04/83 - Roy Soltoff
. This FIX corrects the symbol table addition on the
. statement: LABEL <TAB> ;COMMENT
. Patched starting with 820428
.
D12,55=FE 3B C2 63 72 4D C3 28 72
.; WAS 00 00 00 00 00 00 00 00 00
D1B,7E=28 04 C3 54 69 00 FE 3B 28 DE
.; WAS 20 04 FE 3B 28 DA FE 3B 28 C5
. end of patch
```

The next three patches are for Model I/III EDAS. They should be applied to your working copy of EDAS based on the serial number of your master.

```
. EDAS415/FIX - 05/18/83 - Applied starting 820663
. This fix corrects the bug where the -IM assembly option
. may cause an erroneous "File not open" error.
.
. PATCH EDAS/CMD USING EDAS415
.
D12,7A=B6 21 01 5A B6 C9
. was  00 00 00 00 00 00
D1A,5F=CD 79 69
. was  21 26 38
. End of fix

. EDAS416/FIX - 05/18/83 - Applied starting 820663
. This fix corrects the bug introduced by EDAS414/FIX
. when MACROs are used and -WS option is specified.
.
. PATCH EDAS/CMD USING EDAS416
.
D12,80=2A B4 56 22 CE 57 C9
. was  00 00 00 00 00 00 00
D19,DA=CD 7F 69
. was  2A B4 56
```



```
. End of fix

. EDAS417/FIX - 10/18/83 - Applied starting 820996
. This fix corrects the bug that sometimes occurred
. when you entered EDAS with a wrong parameter.
.
. PATCH EDAS/CMD USING EDAS417
D02,FD=67 44; WAS 2C 5B
D06,87=0D; WAS A1
. End of patch
```

There are three patches for PRO-DUCE Version 4.1. They are as follows:

```
. PCREAT01/FIX - 05/18/83 - Applied starting 820013
. This fix corrects the bug where the -IM assembly option
. causes low memory to be altered.
.
. PATCH EDAS/CMD USING PCREAT01
.
D12,6B=B6 21 26 38 B6 C9
F12,6B=00 00 00 00 00 00
D1A,55=CD FC 47
F1A,55=21 26 38
. End of fix

. PCREAT02/FIX - 05/18/83 - Applied starting 820013
. This fix corrects the bug introduced
. when MACROs are used and -WS option is specified.
.
. PATCH EDAS/CMD USING PCREAT02
.
D12,71=2A B4 34 22 CE 35 C9
. was  00 00 00 00 00 00 00
D19,D0=CD 02 48
. was  2A B4 34
. End of fix

. PCREAT03/FIX - 10/18/83 - Applied starting 820996
. This fix corrects the bug that sometimes occurred
. when you entered EDAS with a wrong parameter.
.
. PATCH EDAS/CMD USING EDAS417
D02,9C=3E 0A EF
F02,96=CD 5F 39
D06,8A=0D
F06,8A=A1
. End of patch
```

Now that I am discussing PRO-CREATE, a minor drawback in its operation was discovered when testing the PRO-LC package. If an error occurred during the assembly process, the assembler aborts to DOS ready. This left any output files open. In order to correct this problem, it was necessary to alter the source code and reassemble the assembler instead of patching. Thus, starting with serial number 820072, MISOSYS is shipping PRO-CREATE version 4.2a. If you have PRO-CREATE version 4.1 and want the 4.2a version, you can return

your master disk with $5 to cover the update handling costs. Be aware that the PRO-LC package contains only one 40-track double density diskette. That diskette includes the entire compiler and assembler files. Therefore, if you have tendered an order for PRO-LC and have already received version 4.1 of the assembler, you will NOT have to return your PRO-CREATE diskette as you will automatically get version 4.2 with the PRO-LC. However, if you are not ordering PRO-LC, you may want to get the 4.2 release to keep current. Let me stress that this does NOT pertain to the Model I/III EDAS.

Let me now get on to some discussions concerning the use of EDAS. Robert Newton asks, "A subject that I would like to see explained in your lucid style in the next NOTES is the use of LORG. I recently had occasion to use it, but just never got the hang of it. For example, how do you revert back to where you really are?" Bob raises a good point.

The EDAS manual states that "A load-origin assembler directive, 'LORG', is provided to cause the load addresses of the object file to be based on the LORG operand while the execution code address references will still be based on the 'ORG' operand." Microsoft's M-80 assembler uses .PHASE and .DEPHASE to switch the "load origin" on and off, whereas EDAS provides only the one pseudo-OP.

Before I state exactly what needs to be entered to answer Bob's question, let me address the concept of LORG. Being able to generate a load module that loads at an address different from where it executes is most important when you are writing what is called ROMable code. Many TRS-80's are used as Z-80 (or even 8080) development systems. Such a system is used to generate software which will run on some other machine. The typical use is the generation of software to be placed in a Programmable Read Only Memory (PROM). There are a few companies that sell "PROM burners". Take a look at any TRS-80 publication and you will see ads for such equipment.

The PROM burner gets its name from the method in which a PROM is programmed. The typical PROM direct from the factory essentially contains memory cells that are all 1's. That is, each bit is turned on by a diode connection. In order to transfer a memory image to the PROM, wherever the image has a zero bit, the corresponding PROM bit must be changed from a one to a zero. This is done by electrically destroying the diode - the process sometimes called "burning" [Erasable PROMs, or EPROMS, are programmed differently]. Now the function of the PROM burner is to read the memory image and burn the appropriate bits in the PROM. The memory image usually must be scanned hundreds of times in the process. Where the PROM burner connects to the development machine's bus, it addresses the RAM. On a Model I, about the lowest address you could load a program is X'5200'. What do you do when the program that is to be placed into the PROM needs to execute at X'1000'. You have the requirement to get the program into memory at an address different from where it executes.

There are various ways of accomplishing the address offset function. You could use TRSDOS 1.3's RELO command. You could use CMDFILE/PRO-CESS. You could also directly generate the file to load at X'5200' but execute at X'1000' by assembling it with EDAS using "ORG 1000H" followed by "LORG 5200H".

Let's look at what EDAS does with this set of statements. EDAS maintains an offset counter whose contents are always added to the load address when a load record is generated. If EDAS encounters an LORG, it evaluates the operand and subtracts the current program counter value from the LORG operand. This result becomes the LORG offset counter value. Thus, with the above ORG/LORG statements, the offset counter would contain a value of X'4200'.

Say that we have a situation where the "offset" is to be in effect for only a portion of the assembled program. How can we SWITCH OFF the offset? In order to switch the offset OFF, we have to return it to a zero value. Since we know its value is actually the difference between the LORG operand and the current program counter, it becomes easy to switch the LORG off by specifying a new ORG and LORG with the operand values identical. I originally told Bob that he can do this by specifying "LORG $". That statement will reset the LORG offset to zero since the "$" specifies the current program counter. However, since there may have been code that was generated but not yet output as a load block, we want to force it to be output as a load record before resetting the LORG offset. This can only occur if either the load record is at maximum size (256 bytes) or an ORG with a non-sequential address is encontered. Therefore, the proper way to turn off the LORG is to specify both an ORG and LORG with identical addresses but non-sequential to the last address assembled.

There are other reasons for using LORG. As an example, let's discuss the LDOS BACKUP program. This program contains three modules of code. The first is the module which handles all of your parameters, checks the source and destination disks, etc. The second module performs the mirror-image backup function. The third module performs the reconstruct or by-class backup function. In order to maximize buffer space for the actual backup performed, it would be ideal to relocate the mirror-image or reconstruct module to the lowest available memory. Also, a portion of the first module is not needed after the options are parsed and evaluated. Thus, we have a classic case of a large program divided into three modules, two of which must be loaded at an address different from where they will execute.

Prior to EDAS 4.1, Each module was assembled separately. The second and third modules referenced labels in the first so an EQUATE file was generated with XREF so the second and third modules could be properly assembled. After the second and third modules were assembled, they were offset using CMDFILE, and rewritten. The entire process was clumsy although easily accomplished by Job Control Language. LORG was added to EDAS specifically to handle BACKUP (and FORMAT) as a single assembly. Thus, the two modules that needed offset loading from their execution were generated as part of the single assembly process.

So much for LORG. The next discussion stems from a request forwarded by Ray D. Greet, of Lockleys Australia. Ray wanted to be able to use the *SEARCH directive of EDAS to load macros from a macro library just like *SEARCH is used to obtain assembler source subroutines need to resolve your program's references. Let me share my response to Ray.

The *SEARCH directive was implemented to permit the inclusion of library modules during the assembly process. This function corresponds to the library search process during a linkage of separately assembled relocatable modules.

The Microsoft LINK80 (L-80) performs a sequential search of a REL library file which may contain numerous modules. LINK80 performs a SINGLE sequential search. Thus, if the tenth module needed a routine located in the third module, the third module would not be linked unless referenced by some other module linked prior to reaching the tenth module. That deficiency is due to the single search of the library.

When I implemented the *SEARCH directive, I felt that it was necessary to resolve all references (i.e. externals) regardless of order within the library. To accomplish this, EDAS performs multiple searches of the library's directory until it makes a complete pass through the directory without bringing in a member. EDAS brings in a module when it finds the member name matches a symbol in the symbol table which is currently referenced but undefined. So that EDAS does not lock up in a condition whereby a module is brought in which does not define the label that caused it to be brought in, EDAS will provide a member definition error if the module does not define the label which caused its inclusion.

In the case of MACROs, you will not have a fixed label within the MACRO. By its very nature, they must have different labels on each expansion or else multiple symbol definition errors will occur. Therefore, the "self-defining symbol" problem would prevail. Another restriction with MACROs is that they need to be defined prior to their reference (how else would the assembler know how many addresses to advance the program counter?). All is not lost. EDAS can be fooled into loading a MACRO from a library. Consider that the name of the MACRO can be referenced BUT NOT VIA A MACRO CALL REQUEST. For instance, if you have a MACRO named "ADDHLA" stored in a PaDS library with member name ADDHLA, use the following statements at the beginning of your assembler program:

```
DUMMY   DEFL  ADDHLA!MACRO2!MACRO3
*SEARCH MACLIB
```

Note that you can have more than one MACRO name referenced by ORing the names. You could OR, AND, ADD, etc. The above DEFL statement can be correlated with an EXTERNAL statement on other assemblers. Yes, you will define the symbol, DUMMY, but so what? If you do not want to "waste" another label, use "@@1", "@@2", "@@3", or "@@4" which are always defined by the assembler (note: don't use the @@n labels if you are using command-line arguments P1, P2, P3, and P4).

The above will work since EDAS will not flag as an error, a reference to a symbol of the same name as a MACRO. It will flag as an error a definition of a symbol the same name as a MACRO. Good luck in your "search".

Another issue involves the *GET directive and actually also pertains to *SEARCH. One of my users had experienced difficulty when nesting *GETs. EDAS can handle up to five levels of nesting (this is version 4.1). However, when EDAS tried to read the second level, it gave this user a "Bad parameters" error. The error pertains to a bad file format. Read on to find out why this error appeared and what you can do about it.

The standard header of assembler source files is a X'D3' followed by a 6-character name. Since this convention was established by Microsoft in the original version of EDTASM and carried forward by Apparat in their disk

version of Radio Shacks EDTASM, I can only guess at its significance. For cassette files, it's useful to be able to identify the type of file. The X'D3' serves that purpose. Is it a coincidence that X'D3' is the letter "S" with the high order bit set ("S" for source). Anyway, that's the header record. Microsoft also established the assembler statement format as a 5-character line number with each high order bit set followed by a space (X'20') followed by the source statement, and terminated by a carriage return. M-80 and FORTRAN actually use a TAB in lieu of the space as used in EDTASM.

Our position is that headers and line numbers take up space and waste file loading time. EDAS 3.5 permitted you to load a file either with/without a header and with/without line numbers. This was done by specifying auxilliary characters with the "L" command. The EDAS 3.5 *GET directive required a headered and numbered file. EDAS 4.1 automatically detects the presence of a header and line numbers. The header is ascertained by examining the first byte of the file. If it's a X'D3', the file is assumed to have a header. Once the header determination is made, the first character of the first statement is examined. If it has the high order bit set, the file is assumed numbered.

When *GET is used, the determination is performed for the first file. If a *GET is nested, the determination has already been made. Thus the secondary file must have the same arrangement as the higher level file in terms of header and line numbers. This means that if the higher level file has no header nor line numbers, the nested file cannot have line numbers nor can it have a header. It is possible to redesign the assembler so that the nesting routine would save the current header/number flags and restore them at the nest exit; however, it does not appear to be too necessary. Just remember to keep all files used with *GET with the same header/number configuration. Also, if a *GET file has a *SEARCH directive, all of the library routines should be the same construction as the *GET file.

Michael D. Caubarreaux, of Houston, TX, has a problem with the conflict of pagination when using the LDOS printer filter (FORMS filter for TRSDOS 6.x). Since EDAS and XREF both do their own line counting, If you have a filter in your printer device that also does line counting and forms control, the two programs are fighting each other. Let me share my response to Michael.

Let me attempt to resolve the queries concerning paging conflicts. To begin with, both EDAS and XREF incorporate the capabilities of pagination and titling the listings that each produces (assembler listing for EDAS and cross-reference listing for XREF). In order to accomplish this listing pagination, it is necessary to keep an internal line counter. Any external filtering that does its own pagination is self defeating for the purposes of titling. Printer filters are nice to be able to supply customization to meet certain formats (such as long/short paper, wrap-around, indent, paper eject, etc.). However, when you have an application that generates a title on each page of output, either the application must take care of its own line counting, or there must exist STANDARDIZED feedback from the OS to the application.

The feedback from the system to the application is insufficiently standard, in my opinion, to deal with its use by an application (such as EDAS

or XREF). Therefore, it is necessary to reach some compromise. If you demand to be able to control your paging external to our applications, then you will have to give up the titling generated by EDAS/XREF. Patches to do this will follow this text. Alternatively, you can recognize what parameters you have set for page-length and lines printed-per-page within the printer filtering process and tell EDAS/XREF what those are. XREF provides command-line options to adjust its line-width (LEN), page-length (PAGE), and lines-per-page (LINES). EDAS provides a command (1) to set PAGE and LINES for its internal use. Why not use them?

Herewith are patches to disable paging checks:

Patch to EDAS/CMD Version 4.1
D04,87=C9; was C0

Patch to XREF/CMD Version 4.1a
D06,62=C9; was C0

I hope that these patches are sufficient to resolve what you consider to be conflicts with EDAS/XREF and printer filtering.

Note: PRO-CREATE users may utilize the following patches:

Patch to PRO-CREATE 4.1
D04,44=C9
F04,44=C0

Patch to PRO-CREATE 4.2
D04,8E=C9
F04,8E=C0

Patch to XREF supplied with PRO-CREATE
D06,38=C9
F06,38=C0

Finally, D. F. Roberts of Cirencester, England reports two problems. I won't bore you with the first one since it was corrected with EDAS48/FIX. His second problem was as follows: "The second is one which strange to say is similar to one which also existed on the Microsoft Editor Assembler Plus. Namely if one symbol is equated to a label using the EQU directive, a 'Multiple Definition Error' is incorrectly given. This only happens if the value of the label is non-zero!!. If the DEFL directive is used the error is not generated so this can be used to avoid the problem, however if there is a simple cure it would be nice to have it corrected."

D.F went on to illustrate sample code which produced the error:

```
ABC   EQU   DEF
DEF   EQU   1
      END
```

I believe that D.F has totally misunderstood the differences between EQU and DEFL because what he/she is purporting is ABSOLUTELY NO ERROR in EDAS!!! It is not a bug in EDTASM+, either. Since it is possible that someone else may also misunderstand the difference, let me emphasize. The EDAS manual states

that, "The 'EQU' pseudo-OP is the generally accepted way to define constant values for use in your program". I should emphasize, CONSTANT. The manual says of DEFL that, "This declaration is similar to the 'EQU' declaration except that the label value is permitted to change during the course of the assembly without producing phase errors".

Let's also remember that the assembler also operates in at least two passes in order to generate the object code. The first pass through the source, the assembler is building a table of all labels referenced or defined in the code. Any time that a label is referenced before it is defined, the assumed value is zero. Thus in D.F.'s sample, the assembler's first pass assigns a value of zero to ABC since "DEF" is not yet defined with a value. The second pass finds that "DEF" has a value of one and thus "ABC" is equated to one. However, since ABC was defined via an EQU pseudo-OP which must assign a CONSTANT value, the symbol's value has been redefined in error. That is what a "multiply defined symbol" error means! The fact that the "bug" didn't materialize if DEF was equated to zero is only due to the assembler's assumption of a zero value for any undefined symbol. If the sample code is revised to read:

```
DEF   EQU   1
ABC   EQU   DEF
```

the desired result will be achieved. The use of DEFL permits the redifinition of symbols with new values because that is the function of DEFL. The EQU is an "equate" of a symbol to a constant value. It is wrong to interchange the two since that will most likely result in a program bug. DEFL is usually used when you want to change the value of a symbol. I recommend that those who are unclear on the EQU/DEFL distinction re-read the manual covering the two pseudo-OPs.

GRASP

In the last issue of NOTES, I delivered a pitch that asked for you GRASP users to submit any character fonts that you have developed. MISOSYS would then build up a font disk and make it available to all GRASP users. I am happy to report that the first disk of character fonts, GRASPF1, is available. Via the courtesy of both R. W. Odlin of Sedro-Woolley, WA and Karl Hessinger of College Park, MD, GRASPF1 contains ten different character fonts. Karl has supplied CLASSIC, BIGCHAR, COMPUTER, and COMPRESS. R. W. has developed some nifty character sets complete with KSM files and JCL installation procedures. R.W.'s include Anglo-Saxon, Anglo-Saxon BOLD, Irish, Irish BOLD, Coptic, and Egyptological.

To whet your appetite, I will cite some of the text of R.W.'s letters. "The Anglo-Saxon set is designed to provide vowels with macron with CLEAR-{vowel}; the two "th" letters with CLEAR-d and CLEAR-t, on grounds of visual resemblance; the ligature "ae" with "j". In all cases, Capitalized forms may be ascertained by pressing CLEAR-{letter}, backspacing, and typing SHIFT plus the key indicated.

The great abundance of special forms of 'H' needed by Egyptologists has made it impossible for the KSM to cover all needed letters in that character