THE MISOSYS QUARTERLY

In this issue:

- BANKER – RAM bank control
- BINHEX revisited
- Exposè on LDOS 5.3
- Fractals in FORTRAN
- Model 4P BOOT ROM exposed
- Split REL libraries with SPLITLIB

Volume I, Issue ii

Fall 1986

THE MISOSYS QUARTERLY

Volume I, Issue ii **Fall 1986**

10-2-87

Table of Contents

PO Box 239, Sterling, VA, 22170-0239
703-450-4181

Greetings astute individuals, and welcome to this, the second issue of THE MISOSYS QUARTERLY. I call you astute, because you recognize the value of a QUARTERLY subscription. Not only that, you share the pleasure with less than 600 other folks. What did we do, scare the subscribers off with our $25 subscription fee?

Before we get gloomy, take some time to look at the cover of this issue. That graphic staring at you in the middle of the page is our new logo. It is our concept of a tetrahedron; the four vertices representing the four categories of our products: operating systems, languages, applications, and utilities. You'll start seeing a lot more of this symbol. I hope you like it.

While I am on the subject of symbols, we have on occassion been asked the derivation of our corporate name - MISOSYS. It's simple. It is derived from MIcrocomputer SOftware SYStems. Back in the early days when we started out, we wanted to use the initials; however, the other old timers amongst us will recollect a company called "Michael Shrayer Software" which used the initials of "MSS". Remember him?

Turning to the gloomy side for a bit, I know we get discouraged by all of the theft going on in the software industry. It just so depressing after you spend all of the time developing and supporting a product that you hear of scoundrels giving away copies to any and all comers.

In mid-October, I posted a message on our Compuserve SIG that posed three possible reasons why our product sales are poor in light of the marketplace size. I suggested that (1) our products are being stolen, (2) we are not marketing them effectively, or (3) our products are just not usable. I ended with my hunch that the first reason is the correct one. Here's one response (the only response) to my message.

JB responded with, "I bet that your option (1) is correct. In this city there's one guy handing round free copies of the entire pro- line. I was disgusted, but he just grinned at me and passed on to the next guy."

I agreed with JB. Every PRO-NTO user I have spoken to has told me that PRO-NTO is the best software investment they have ever made. 80 MICRO told me my advertising was poor and they could do much better. So they took over the job. Ever since we went to a full-page ad and dropped back to a 1/2 page, 80 MICRO's graphics shop has been doing our ads. That's going to change as we now are working with a local outfit for ad preparation. You can see their work in the LDOS 5.3 ad appearing in this issue. Thus our marketing efforts were all they could be and the product was right. You have another idea?

My grumblings about software theft were based on my perspective that since the release of PRO-NTO (PRO-WAM) in April of 1985 - that's a total of 19 months of sales, we have sold less than 1200 copies. That's an average of 63 copies per month. Even those figures distort the recent picture. PRO-NTO sales peaked late last year. The recent performance since March of this year shows an average of 40 per month. Even worse yet is the average of 30 per month since June. Now 40 per month represents about $2400 of gross sales. Out of the $2400, Karl gets $360 in royalties, 80 MICRO gets $1800 in advertising costs which leaves $240 to pay for raw materials (disks & documentation), manufacturing (cardboard, shrink wrap, duplication, warehousing), sales (invoicing, telephone, office supplies, office machines, clerical staff), support (answering questions, responding to letters), as well as providing funding for new product development (new PRO-NTO/PRO-WAM applications, other Model 4 prodcuts), and general office overhead (taxes, licenses, fees, light, heat, management, etc) - not to even mention the development cost of the pieces not authored by Karl. Unfortunately, very little of the latter gets paid for out of that $240. Now you tell me, how do I continue to justify development of Model 4 (or III) software?

With that out of the way, let me state the policy for the use of names here in TMQ. Unless otherwise requested, or made necessary by the content, we will only use

initials when referencing any person; except, of course, my own name, Roy Soltoff. With this policy, we should offend no one by having their identity revealed unnecessarily. The BLURB, constituting our "letters to the editor" column, will still carry names unless we are specifically directed to do otherwise.

Turning to a brighter issue, JG asked, "If MAXIMUL is still alive, please send me the info. Thanks in advance. For those folks not in the know, MAXIMUL is the user group for MAX-80 users. The big question these days is, "where's Lobo?". Someone on Compuserve supplied the current address of the MAXIMUL group. Here it is

Maximul III:
Attn: H. Glen Guyer
1111 South Ditmar
Oceanside CA 92054
(619) 722-5002

Glen is also selling the Adaptec HD controller which can be hooked up to the MAX for use with the WIN drivers (M80HD) which we wrote for Lobo. You MAX-80 owners also ought to be pleased that we did a 5.3 release of LDOS for the MAX-80. Our thanks to Les Mikesell for getting together some of the SYS0 source code which he had archived (Les did the original 5.1.3 version when he was employed at Logical Systems).

Paul Bradshaw reports, "Roy (and MQ readers), After attempting to type in your POKEPR program from the current MQ [summer 1986 issue, page 45], I've discovered some trouble. It seems that the BUILD command with the HEX parameter doesn't accept spaces between the codes. Furthermore, the BUILD command doesn't provide any Load File Format information when it writes the file to disk, so your build file doesn't work. The fix for this is to eliminate all spaces (or press ENTER between each number) and add the following code to the beginning: "01 11 00 26" and add "02 02 00 26" to the end (eliminating spaces and the quotes, obviously). I also noted that changing the name to POKEDO and the 5th byte of your listing to a "02" [from a 06] created another, more useful, program. POKEDO will allow you to gain control over those troublesome character set switches, and inverse video! It's the little things like this that make the MISOSYS Quarterly worth it! (BTW, I've renamed the two files to be PRT/CMD and DSP/CMD to correspond to their SVC counterparts)."

Must have been one of those days, Paul. Thanks for keeping me on my toes. Here's the complete listing with spaces for readability only:

```
01 11 00 26 3E 60 EF 3E 06 EF 7E 23
FE 0D 20 F4 ED 62 C9 02 02 00 26
```

HB asked us, "Any chance of an update to The Source (or at least insert pages) to reflect the changes in 6.3?" I was forced to respond with, "Absolutely not. THE SOURCE did not sell well enough to justify any continuation of its accuracy as the DOS migrates upward." Looking at some of the facts, LSI sold less than 200 copies of the 3-volume set. Since March, we have sold about 200 sets at the special $99.95 price. Less than 500 copies in over two years does not justify anything else but selling out of the initial stock which is taking up warehouse space.

Ed Weber wrote us to say, I want to compliment you on the QUARTERLY; it was good to have it back. A suggestion: try to entice some of the "old time" but now-unheard-from TRS-80 oriented authors to submit articles. For example, I haven't seen anything in print from Tim Daneliuk since the LSI QUARTERLY ceased publication."

Well Ed, I'll put out the pleas; however, I suspect that those folks have long since left the TRS-80 community. I did happen to see a product review written by Tim recently which appeared in one of the MS-DOS publications. Note that there are plenty of "new" experts in this business. Many of the current users are able to take up the slack. The experience level on our Compuserve SIG, for instance, has not abated but actually grown over the years. The names have changed; but you still find well qualified, capable, and adept people

discussing the 8-bitters today as well.

Ken Arntsen, one of our United Kingdom readers wrote, "May I congratulate you on the first issue of THE MISOSYS QUARTERLY. It is packed with interest and will fill an important need for the TRS-80 community. Since the number of publications geared towards the TRS-80 user are getting more limited by the day, I shall look forward to the next issue with great hopes.

Your idea of printing some of the dialogue carried out through the Compuserve SIG is a great one but have you considered offering a selection of the useful public domain applications on DISK NOTES for those of us not able to dial into Compuserve?

I have one query about LITTLE BROTHER which I otherwise cannot fault (after translating and rewriting the manual!). Since the system and application files seem to be SYSGENed or on memDISK, is there not an easy way to backup files from 1 to 2 after a session?"

Ken, The first responsibility we have is to ensure that we continue supporting our customers; thus, one goal is to stay in business. MISOSYS is not a big company. We are a few people dedicated to this business and working our tails off. With such a small staff, there is no room for handling "freebie" services such as providing public domain software for the cost of handling the service. Also, I personally have a distaste for folks setting themselves up in business SELLING public domain software at a profit. If anybody should make money off of a software product, it first should be the author. With this in mind, it would not be in our best interests to provide such a service.

On the other hand, if some of the folks using our board got together to make available the PD programs at the cost of handling **where it was determined to be legal for such an undertaking**, we would not have any objection. Bear in mind, though, the author of a PD piece should be the one to decide how his work should be distributed. Not someone arbitrarily downloading it. I would not suggest the sysop undertake it because Joe must be burdened already doing everything that he does to keep our board running smoothly. Perhaps the users of our board may want to discuss the issue.

We will continue to make available in DISK NOTES, the programs and listings which appear as articles in the QUARTERLY.

As far as the LB problem, I suggest that you create a BOOT disk which sets up the extra RAM as a full memDISK, and turns it into a SYSTEM drive. Then you can easily use the two floppies for by-file backups.

Bob Zinn wrote us with the following, "I am including a diskette with my augmented version of cc (allowing -e [output to screen instead of file for testing cc] -g [go after linking WITH parameter passing to the go step] and other options I found handy. I also eliminated upper/lower case and added +,- checking. There is still plenty of room for further mods. I wrote and integrated with cc, comment/ccc to add #asm comments to the asm source code." Bob also provided a few other MC goodies like "dump(addr,len)", "printat(cursorloc,...)" just like printf(...), and of course, a list of bugs. Since this issue is full, his input will be delayed until the next issue; however, I am making his version of cc/ccc available on DISK NOTES 6.

Cliff Richards reports from Sydney Australia, "QUARTERLY is GREAT but pages 44 and 57 are blank. Is it possible for a fix to PRO-NTO to enable it to reside in other memory banks with an Alpha Supermem board? I have been holding off on Mister ED bacause PRO-NTO cannot be removed when memDISK is running. It is not practical to use Compuserve from Australia. Page 87 of the QUARTERLY, RAMDRV.LQR, would be most valuable to us here. Do you have any ideas? Congratulations on the QUARTERLY, and your refreshing stand to continue with the Z80. No IBM for me, I love my Model 4 and it does everything I need."

Cliff, sorry about the pages. It happened to too many folks and our printer heard

about that! For PRO-NTO, take a look at the BANKER utility in this issue. If the above discussion concerning our SIG's PD software made available on disk reaches fruition, your solution lay there.

Sam Wells writes, "First, congratulations on the impressive first edition of THE MISOSYS QUARTERLY. It certainly promises to be quite an informative and enjoyable publication.

I consider myself fairly well read computerwise (I subscribe to 80 MICRO, Northern Bytes, The Kepner Letter/Ramparts, and various other publications). I find it hard to keep up with the various software names you employ. For example, when scanning the generous software offered to subscribers of the QUARTERLY, I found myself unsure of which programs ran on what computer. This was still the case after I reviewed your ads in the latest 80 MICROs, your latest catalog in my possession (84-1), and other materials I have. Apparently I am not receiving the latest mailouts containing the descriptions of your products.

You provide a great service to us end users out here in the great beyond. Your prices are quite reasonable, your software very utilitarian. The effort put forth on the Compuserve SIG goes well beyond any other support system I've come across and insures that I'll be using your products as long as I have a Model III or 4. I suspect that will be for a few more years at least, and I suspect by the time I join the rest of the world in its mediocre rush to MS-DOS, you will be there with the same quality software and support that I've grown accustomed to over the years."

Sam, we don't automatically send our new catalog to everyone in our database. We do enclose a new catalog when we ship every order. Since about March of 1986, we have used the reader response requests from our ads for entry into our database with an indicator to flag for a catalog. If the name was already in and flagged as having been sent the latest catalog, they would not get another. Because it takes a minimum of 200 pieces for BULK mailing, catalog mailings are done only about once a month. Included in that mailing would be any requests for a catalog arriving by mail or telephone. At a cost of about $1 per mailing (much more if out of the States), we just can't afford to mail the catalog to our entire database of 17,000. We did a flyer back in March sent to 16,000 at a cost of about $5000 in printing, bundling, and mailing charges. That's kind of tough to do. We may be considering dropping our ads in 80 MICRO for a month to pay for another flyer early next year. On the other hand, to circumvent the problem you had in deciphering our specials noted in the last issue, I will give brief descriptions in The BLURB for this issue's specials.

Harry G. Maurer writes, "My overall impressions of the QUARTERLY was that it is well worth the money and I was pleased with its content, format and general professional appearance. The information presented by you in the QUARTERLY has been very useful to me. As far as I know your advice and tidbits of information is not available from any other source. Sadly to say I feel that LOGICAL SYSTEMS missed the boat by cancelling the LSI JOURNAL. It became obvious to me that as each issue was released, they devoted less and less in-house technical support to the publication. Little items such as your short comments about the family that you have in the NOTES and the QUARTERLY gives me the feeling that there is a real live human being at the end of the chain. As a personal request, please continue to print in the QUARTERLY, those messages which you deem would be of interest to your subscribers."

Harry, I share your views; however, I have come to understand why LSI took the position they did with their JOURNAL. When I decided to initiate THE MISOSYS QUARTERLY, my decision was based on the assumption that the tens of thousands of potential customers in the LSI data base and the thousands of customers in my database would provide a few thousand subscribers. Let's talk economics. I really expected 2000 subscribers initially. That subscription base would take in a minimum $50,000 in revenue immediately. With that financial resource,

my plan was to hire someone to do everything associated with putting the QUARTERLY together. It also was going to fund the acquisition of a desktop publishing system - including a LASER printer for the job. Plus, the interest on the invested subscription base would help offset the revenue loss of the product specials. My estimate was that each QUARTERLY would take about one person-month of preparation effort. That person would then be available other times for different work (writing user, manuals, for instance). Now with an actual initial subscription base of less than one-fourth the expected, none of the plan can be carried out. In addition, each issue is so much more expensive to print as the economies of scale associated with offset printing don't come into play at such low levels. It still takes up about a month of my time - time which I can ill afford to devote - but I am committed to the job. So THE MISOSYS QUARTERLY is to be continued; but we may investigate ways to make it profitable.

Gary W. Shanafelt writes concerning the imminent release of LDOS 5.3 and LS-DOS 5.3, "I am not interested in a new release of the two systems with a lot of new features. I would like to see only three things: (1) extended date capability after 1987 (I don't care about recording the time on my files); (2) as complete a common command structure as possible between LDOS and TRSDOS 6 (so that both would share commands like CAT and BACKUP would work the same way on both -- now VIS is the default on one but not the other, etc.); and (3) correction of any bugs in the current release."

Gary went on to note a number of very valid reasons too lengthy to continue here. Needless to say, Gary will be pleased with what we have done in LDOS (other than it does keep the time stamp and we just didn't have any more room in the library table for CAT). Interested readers must read all of the details on the LDOS 5.3 release appearing in the NEW PRODUCTS section.

Ron Ungashick said, "Roy, I received THE MISOSYS QUARTERLY yesterday. I am very impressed! I really liked the LDOS Quarterly, but I believe this is even better. Anyone who used to receive the LDOS Quarterly should subscribe to THE MISOSYS QUARTERLY. It is well worth the price. Thanks for a good publication. It has been too long since one has been available."

Thanks for the input. I hope to get my printer to do a better job on issue I.ii targeted for mid-November.

Bill Evans reported, "I just got THE MISOSYS QUARTERLY and personally feel that Volume 1 Issue 1 is more valuable than a year's subscription to 80 MICRO (to much MesSDOS stuff in 80 MICRO). I strongly urge those who did not subscribe to do so now. Roy, there was a mistake in my copy. When the pages were bound together a couple of sheets were inadvertently left out. I am missing pages 43-46 and 55-58. Please send me these pages A.S.A.P."

Bob Haynes said, "Hi Roy. Like Bill Evans, I also just received 'THE MISOSYS QUARTERLY'. Very, very nice! Really like the idea of condensing the tips and techniques found in the SIG, and it looks like 'Roy's Technical Corner' has been resurrected in the Programmer's Corner section. I also like the double column format and smaller type (than the LDOS Quarterly). Really packs in the information. Another nice touch is the cello envelope; my postman won't dare scrunch up THIS magazine. It's well organized information in a nicely bound package. Congratulations, thanks, and be encouraged! If Tandy users won't stand behind THIS kind of SUPPORT, they don't deserve any! Oh one small problem, same thing happened to my copy as Bill's, pages 45-46, 55-56 are missing. Please send dupes? (kinda like goin' to heaven, nd steppin' on a thumbtack on the way.) Thanks again!"

We'll get you a replacement as well. Starting this issue, every one mailed will be enclosed in a plastic called, "envelowrap™".

Jack Lottey chimed in with, "Volume I.i of THE MISOSYS QUARTERLY is great! I have one problem: pages 45-46 and their mate, 55-56 are missing. I ended up with two of 41-42 and mates, 59-60. I am also interested in the MISOSYS NOTES that apparently preceded the Quarterly. How can I get them?"

Jack, We will get you a replacement for those missing pages. Seems like my printer must have had a problem. About a half dozen folks so far were missing those specific pages. Old NOTES are $3 apiece; there are 4 issues. S&H is $0.50 an issue. We also have old LSI QUARTERLIES/JOURNALS. The entire 6-issue Volume II is $9.95 plus $3.50 S&H.

Joe Kyle-DiPietropaola (ever wonder how to spell his name?) reported, "Oops! In the current MISOSYS Catalog, it appears that Filter Disk #1 and Filter Disk #2 were omitted from the price lists. In parity with the other products, I imagine that they are $29.95, is this correct?"

No they're not! I already know that Filter Disk I and II were omitted from the price sheet. Each is $19.95 + $2S&H. The price for Mister ED is also wrong. It's $59.95. We will be doing a one sheet flyer as an insert addendum.

Jeff Brenton reports, "Thank you for publishing my article on the development of SETBAUD, Roy! There is one problem, however. Apparently the updated version never got to you, so the program (as published) doesn't correctly sense 6.x machines. If you'd like, I can submit the corrected article to you directly, or just a "letter to the editor" about the differences. Turns out that the real test was simpler than the original! There are only 3 lines that need replacing:

```
590 LD A,(002AH) ;Test to see if this is
600 CP 40H       ;TRSDOS/LDOS 6.x system
610 JR C,L6      ;SVC table below 4000H
```

the rest is OK, except for the text explanation."

A letter to the editor is fine. I believe that I used the article which you recently sent to me; not the one dug out of the archived floppy boxes of LSI.

Dave Krebs reports, "I just received the MISOSYS QUARTERLY ... a yoemen's job! Looks GREAT! Even on coated stock, classy. In lookin' through the patches, most of which were already on my software, I noticed SAID 1.1. My version, delivered on PRO-CREATE, was 1.0. Is there an upgrade I should be asking for?"

Actually there was to be mention of a $5 (+S&H) upgrade for SAID. I don't believe it got mentioned. V1.1 adds a few features (like line # reporting and GOTO line#). Send back your master with the fee for an update. It's noted in this issue.

Theodore Masterton gave us these consoling words, "You may be weary of hearing this but I want to thank you and yours for staying with the Z80's in the Tandy line. No doubt the end is in sight, but for me, LDOS/TRSDOS and my 4 are all I would ever need. I have used CP/M and MS-DOS 1 and 2 professionally and am continually amazed at the power of LDOS/TRSDOS compared to these "Industry Standards". If I move to MS-DOS, it will be because there is no software left to buy at fair prices, NOT because my system is "obsolete.

I would encourage you to consider making 5.3 as alike TRSDOS 6.x as is possible. They are so much alike now that the few differences in the keyboard and commands are a bit of a slowdown. I get flying and find myself trying to do LDOS stuff in TRSDOS and vice-versa. My best and very selfish wishes for your venture in staying with the I, III, and 4/4p/4d. A long life."

Theodore, Our goal in 5.3 is to make every library and utility command exactly compatible with its 6.3 counterpart. Of course, SET/FILTERing techniques will not change. Nor will the BASIC approach MBASIC. It may even turn out that LDOS 5.3 will have a few things above what gets into 6.3. Now then, may I change LDOS's "KILL" command to "REMOVE"? That would make them more similar!

Here's an item from Jim Gaffney. "In going through the FIXES/TXT file in the MQ/NOTES, I find that I have a fix for a PSORT version which is newer that the one that I have on my PRO-WAM Master disk. I assume, therefore, that I need an update. Suggestion: would it be possible to publish in the MQ a listing of the latest README/TXT files for all the MISOSYS products? Or better yet, maintain it on one of the DL'S in the SIG?"

Jim, I don't think the majority of readers would like the QUARTERLY filled with that kind of information. The latest PSORT includes an enhancement. Thus, it is not a freebie. Since I intend to release another version of PRO-WAM early next year (to extend BRINGUP to 1999), etc.), I suggest you wait until then unless it's critical for you. A normal fee is $5. I expect to ask Joe to put the last FIXES file and this issue's FIXES file in the section 2 DL. I also expect to have a few of the articles up on the board under a MISOSYS copyright!

From ole eagle eye Bob Zinn comes this tidbit, "Ref: bugs/errors/cockpit problems page 14, MISOSYS QUARTERLY - summer 1986, col 1: Looks like lines starting with I and III may be reversed. On my mod I, the second patch seems the right one to use. Page 99, PDSC/FIX - format of patch is bad: no indication is given that (I think) APPEND is to receive the patch. Although the patch "installed", the pds(build) (and perhaps others) then gave a "Load file format error". So, I copied out APPEND and tried PDSC from DISKNOTES 5, - it then gives parameter error message. On FIXES/TXT - from DISKNOTES 5, I wrote a little program to unarch it. But had to edit it first because the use of EOP to indicate end of patch was not consistent. And the next line did not always begin with a legal file name."

Bob, You're right on about the Model I and III patches being reversed. The Mod I should have been (D10,88=4A). The PDSC format was correct. The ':' is discussed on page 88 under General Information. The PATCH actually should not have appeared in that issue nor the corresponding DISK NOTES since it appeared in NOTES FROM MISOSYS, Issue IV where it was discussed. The fix is for PDS(APPEND) which must be copied out, patched, then re-appended after the old is killed and purged. We also will be consistant with the "syntax" of the patches this time on DISK NOTES 6. Every patch will start with a ". filespec" and end with a ". Eop".

The SPECIALS this issue include the following for Model I/III users: **EDAS** - our Z80 macro assembler which is recommended for novice to advanced experience level. It operates as a memory resident line editor and assembler. EDAS also includes a full screen text editor, separate disk assembler, and a cross reference utility. **MSP-02** is a collection of utilities and filters [CRLF video filter to properly handle the CR-LF combo; CTLG filter to "ring" the sound port; IOMON disk I/O error filter, NAME to rename disk; RD40 to read a 40-track disk in an 80-track drive; XONXOFF filter for the serial port; UNKILL to bring 'em back alive; a DOEDIT video filter to provide editing; a Disk EDitor; and NODAM, a filter to read a non-LDOS Model I SDEN disk on a Model III]. **DSMBLR** is a two pass labeling Z80 disassembler with a screening text mode for user specification of data regions; The **LDOS 5.1.4 Manual** should be obvious. If you have an older manual (more than about two years old), you may want to consider replacing it. If you just have an LDOS disk and want a manual, here is a good price. We have about 300 of these. When they're gone, they're gone!; **HartFORTH** is a full-blown 1979 STANDARD compiler for the FORTH language. Want to play a little with FORTH? At this price, you can't do better.

For the Model 4 folks, **PRO-CREATE** is the macro assembler equivalent to EDAS mentioned above, **PRO-DUCE** is equivalent to DSMBLR, **PRO-HartFORTH** is the FORTH equivalent, and **THE SOURCE** is the complete commented assembly source listing for TRSDOS 6.2.0 excluding BASIC and HELP and hard disk support programs. Make note the the S&H fee is $5 for all customers: US, Canada, and foreign. That's because THE SOURCE is shipped surface to all locations

due to its seven pound weight. The LS-UTILITY disk contains [addition/subtraction in binary, decimal, hex with the CALC filter; KSMPLUS - a superset of the KSM to allow for key redefinition and more; a MAXLATE translation filter allowing one to one or one to many translation; a PRCODES filter to effect slashed zeroes, boldface, overstrike, and underlining on certain printers, READ40 to read a 40-track disk in an 80-track drive, a filter to TRAP a character in a device call, and TYPEIN to perform automated JCL-like features with the single key input service call.

A reminder to our readers, any order placed for a special **MUST** be accompanied by the SPECIAL ORDER BLANK which is included with your issue. That's because you are entitled to only one product at the special price. Also make note that DISK NOTES are priced at $10 plus S&H. DISK NOTES 6 contains all of the programs, listings, and patches which are in this issue. When ordered with the SPECIAL ORDER BLANK, you can get a copy of the DISK NOTES related to the issue for $7.50 plus S&H - a $2.50 savings.

Folks, we get lots of letters. We get more letters than we can answer. Some of you won't get a response. Here's how we deal with this problem. I take a quick scan of the correspondance. Adequately described bug reports get high priority. THE MISOSYS QUARTERLY input goes in the TMQ box for the next issue. Letters asking a question which can be answered quickly by noting a response on the sender's letter and mailing it back in the sender's supplied Self Addressed Stamped Envelope (SASE) get answered. Five page letters with a hundred questions will almost NEVER get answered. The courtesy of supplying a SASE will earn your letter a higher spot on the list. Letters which do NOT relate to a MISOSYS product will usually get canned. Letters with questions general to a wider audience may see some exposure in this column so more can benefit from the reply. At any one time, there may be 25-50 letters in the big box on my desk. Unanswered letters over three months old usually get canned. Folks, this policy may upset some of you because you feel each letter is quite important to you (of course, for why else would it be sent); however, there's only one of me to respond to all of you. Rest assured, I will read or scan through EVERY letter received - even the ones I don't respond to. They make an impression on me and the way I run this company even though they may not get a reply.

Note on holiday schedule: One of the few times I take a holiday and close up the business is the festive time surrounding Christmas/Chanuka and New Year's. I think that there needs to be at least one time throughout the year that families who live apart need to get together. On Brenda's side of the house, we have her parents in Miami, her brother in Gainesville FL, and her sister near Atlanta. Her mom and dad have a cabin in the mountains near Edneyville NC (about 30 miles from Ashville); That has been a collection point a few times in the past for family get togethers. This season, that's where we'll be.

Of course Christmas and New Year's come at a very awkward day of the week for vacations this year. I am sure that a lot of you have already recognized that. Here's our plan. **MISOSYS will be closed from December 23rd until January 2nd.** We are going to fill any orders received up to the 23rd. Thus, the 23rd will be the last shipping day in 1986. As soon as UPS has picked up, we're off to NC. We will be back probably the Tuesday after Christmas but the phones won't get answered until the Monday after New Year's day. The "holiday" season is usually kind of slow, anyway. LDOS 5.3 upgrade kits will probably start shipping on January 6th. Have a good and safe holiday.

Stacey and Stefanie are growing faster than weeds. This is a very interesting and rapid period of development (age 2-4). I know some of you more experienced parents will be telling me every "age" seemed rapid - especially those teen years.

Stefanie started in "fun for two year olds" recently which around here is a one-hour "class" two days a week. I happened to take her for the first class since that day was also the day Stacey's class went to the pumkin patch. Both Brenda and I wanted to share both of those experiences; however, since Stacey's school is 2-1/2 hours and Stef's is 1 hour, Brenda got Stacey and I got Stefanie. Good thing there's two of us.

It's getting more difficult to get (and keep) Stacey in bed at a reasonable hour. On the other hand, she is a really good eater. When I was her age (not to mention 5-7 years older), I don't think I would touch a shrimp. From chicken to fish to beef to pasta, common vegetables like peas, fordhooks, carrots, corn, green beans (not cauliflower, squash, zuchini), to potatoes, rice, noodles, and Cheerios™, feeding her is not a chore. She even likes broccoli (calls them little trees). Stef, on the other hand, doesn't "like" chicken, doesn't like beef, doesn't like fish, doesn't like orange juice, ... Thank heavens for Cheerios, Rice Crispies, Corn Flakes, noodles, and Macaroni and Cheese. Spaghetti is also a favorite. And, of course, both girls like park eggs (that's my mom's invention from many years ago; sort of an omelette pushed up while it's cooking to get ridges which look like hills and valleys of a park; add a few cheeze twist park benches and a square cheeze pond sprinkled with a little parsley grass). Well, it's all fun. Since the office is at home, I get the opportunity to "enjoy" all three meals with the family. Sometimes, that's too much.

Best of luck to anyone else raising a family out there. Don't forget to take some time and smell the flowers. Growing up is great. Don't lose that time with your children. See you next issue. And see you next year.

The Gobbling Box - a pursuit game

I know what you're saying. MISOSYS publishing a **game** program? What's the world coming to? Well we did it. Tony Cosentino came to us one day to show us some of his programming efforts. Little did I know they were for the Model I! The game was cute. The holidays were approaching. But a Model I? It was then that Tony told me that the game used nothing from the ROM, nothing from the DOS! My eyes lit up. I said, hey! I bet we could get this one game to work on a Model III and a Model 4.

Since The Gobbling Box was totally self-contained, all that Tony needed was a way to find out what machine the program was invoked on and adapt the machine environment to the game. You recollect the last issue of TMQ had an article by Jeffrey Brenton on machine sensing. We used that technique, and some of my techniques to get at the BOOT ROOM of the 4P (which, by the way, appear in this issue), and came up with a single program which works on a Model I, Model III, Model 4, Model 4P, or MAX80. The program actually converts a Model 4 or 4P into the Model III mode during the execution of the game and restores it intact as a 4 at the completion of the game - or whenever you get tired of playing.

Let's speak a little about the game. The GOBBLING BOX game generates a variety of special sound effects and music which complement the action on the screen. The action sounds are ported simultaneously through both the cassette port and the sound port of the Model 4/4p. The four arrow keys are normally used to control the movements of the GOBBLER in this game; although the game also supports the use of an Alpha Products joystick for "gobbler" positioning.

The object of the game is to have your GOBBLER eat up as many dots as possible, while trying to avoid the ZONKERS. The ZONKERS won't stop chasing your GOBBLER until one of them eats it or until the GOBBLER eats all the dots on the GameBox. As a reward for cleaning up the GameBox, the GOBBLER gets a new Box to play around in, with about 200 more dots to eat. To help you play longer, you'll get a BONUS GOBBLER. The GOBBLER can Tame the ZONKERS for a short while by eating one of the ENERGIZERS on the board. Then it's the GOBBLERS turn to chase, catch and eat the ZONKERS.

The game has two skill levels; the pace is fast; the sound is great; the action is continuous. For only $14.95 plus $2 S&H ($3 Canada, $5 foreign), you can't beat this bargain of a game. Its shipping now.

UNREL - REL to ASM translator

Here's one of those rare utilities designed for the programmer. I asked Rich Deglin, author of MC, about the feasibility of such a tool since he already had a REL decoder and a Z80 dissassembly module (the DD of DD&T). It wasn't too much later when he came back to me with UNREL. UNREL will decode a relocatable object module which has been assembled by either Microsoft's M80 or MISOSYS' MRAS assemblers. The output is an assembler source file compatible with MRAS and one which should also be equally usable with M80 (probably change the output file extension to MAC!).

Now you can take a binary REL file which looks like this when displayed in hexadecimal:

```
8091D15391D4942045053455481941
49154D155205504F494E5481131253
916054C494E4542819312539150 91A
0553455458598194D155161654A064
44F424F5846819113D1D4941220642
4F584C494E819113D3125391605584
4454C5481565111531520647455450
5859815 1D15516166054E4547484C8
1521311911160344445481159251 55
E06475250434C3180D0D314E500001
```

into a form more usable by your MRAS or M80 assembler; an ASCII file such as the following:

```
;GENGRP/ASM:1
    NAME    ('GENGRP')
    EXTRN   BAKCLR,CLIPP,DCOMPR,DOWNC
    EXTRN   FCERR,FETCHC,FORCLR,GRPACX
    EXTRN   GRPACY,GXPOS,GYPOS,ICOMP
    EXTRN   LEFTC,LINSTL,MAPXYC,MAXDEL
```

```
    EXTRN   MAXUPD,MINDEL,MINUPD,NSETCX
    EXTRN   PGRPCX,PGRPCY,READC,RIGHTC
    EXTRN   SCALXY,SCNCRD,SEEGRP,SETATR
    EXTRN   SETC,STOREC,UPC,VIEWMP
    EXTRN   VXLEFT,VXRGHT,VYLEFT,VYRGHT
    EXTRN   XCHGAC,XCHGX,XCHGY
    PUBLIC  BOXLIN,CLS,DDT,DOBOXF
    PUBLIC  DOGRPH,DOLINE,GETPXY,GETXY
    PUBLIC  GRPCL1,HLFDE,LINE,LINEB
    PUBLIC  LINEBF,NEGHL,POINT,PRESET
    PUBLIC  PSET,SETXY,SETXYR,VIEW
    PUBLIC  XDELT,YDELT
    CSEG
SETXYR:
    PUSH    DE
    PUSH    HL
    LD      HL,(GRPACX)
    LD      (PGRPCX),HL
    EX      (SP),HL
    LD      E,(HL)
    INC     HL
    LD      D,(HL)
    POP     HL
    ADD     HL,DE
    LD      (GRPACX),HL
    LD      (GXPOS),HL
    LD      HL,(GRPACY)
    LD      (PGRPCY),HL
    EX      (SP),HL
    LD      E,(HL)
    INC     HL
    LD      D,(HL)
    POP     HL
    ADD     HL,DE
    LD      (GRPACY),HL
    LD      (GYPOS),HL
    RET
SETXY:
```

This miraculous transformation is made possible by the UNREL utility. Of course, there are limitations.

First, UNREL makes the assumptions that anything in a code segment is code, and anything in a data segment is data. Those of you already having experience with object code disassemblers know that decoders can sometimes get "fooled" by data being interpreted as code. With object module REL files, this problem still exists. However, if good programming practices, such as segregation of code and data, have been followed by the original programmer of the REL module, your decoding job is simpler.

Second, UNREL supports only the following special link items: 0 - entry symbol, 1 - select common block, 2 - program name, 3 - request library search, 5 - define common size, 6 - chain external, 7 - define entry point, 9 - extern+offset, 10 - define data size, 11 - set location counter, 13 - define program size, 14 - end program, 15 - end file. The undocumented special link items (4 and 8) as well as chain address, SLI-12, are not supported. The later is used in Microsoft's one-pass compilers. I believe that SLI-4 and SLI-8 are used in a more recent version of M80 for handling 8-bit externs with greater support of arithmetic expressions resolvable at link time; however, Microsoft wasn't too potent in letting the world know of the specific details associated with that link item. In any event, we're not supporting it.

If you have the MISOSYS librarian for M-80 type REL files, MLIB, you can easily pull apart relocatable libraries and translate the associated modules into ASM source. The example above, incidentally, is from the graphics library, GRPLIB/REL, which is distributed with Tandy's hi res graphics board. This should make a few folks happy.

The UNREL package also includes a utility, DECODREL, for displaying the bit stream of a REL file. This can be used to more fully understand the actual bit stream. If you have MRAS, take a look at page 7-6 of the manual and you will see a sample output of DECODREL. A copy of SPLITLIB is included with the package in case you have a library file too big to handle by MLIB. Note that MLIB is not included with this package - it is available either as part of our MRAS assembler packages [MRAS M-20-083; PRO-MRAS M-21-083] or as a separate product [MLIB M-30-061; PRO-MLIB M-31-061].

There's still a few loose ends to do on this package. At the time this article is being written (mid-November), the documentation is left to do. Also, I have asked Rich to enhance DECODREL to also output the relative starting byte and bit for each piece of decoded information. That kind of data could be useful in certain instances. Thus, the package will be released in January. For a retail price of $39.95 + $2 S&H ($3 Canada), UNREL

should be the perfect professional assembler's tool for your bag of tricks. Here's also another surprise. The TRS-80 version of UNREL will include programs for both the Model 4 (DOS 6.x) and Model I/III LDOS 5.x! A CP/M version will be available shortly thereafter. [UNREL-T80 M-30-054, UNREL-CP/M M-32-054].

LDOS™ 5.3.0 - Upgrade Kit

Here is what all the talk has been about. This LDOS release and the corresponding Logical Systems release of LS-DOS™ 6.3 (for the native mode model 4) is where it's at. Why a new release of these DOS products? The primary reason for such a release is due to system dating. The current vintage supports a system date up to and including December 31, 1987. That's about one year away. A lot of you folks have gotten very nervous about that. LSI and MISOSYS explored a number of alternatives for extending date support beyond 1987. The implemented solution was considered to be the best possible under the circumstances. Here's what we did.

From a user standpoint, the password field described as the ACCESS password in LDOS 5.1.4 and the USER password in TRSDOS™ 6.2.1 was dropped. The password associated with that field will now be interpreted as if it were a blank password. The remaining field, known as UPDATE under LDOS 5.x and OWNER under TRSDOS 6.x, will in both cases now be termed the OWNER password. The protection level granted to the file's access will be based on the PROTection level, as before. Knowledge of the OWNER password will provide full access.

Dropping the one password field frees 16 bits of directory record data. From a technical standpoint, here's what has been done to the directory.

-byte-	--bits--	--------contents--------
DIR+18	bits 3-7	the hour
	bits 0-2	most significant bits of the minute
DIR+19	bits 5-7	least significant bits of the minute
	bits 0-4	year offset from 1980

The existing year field, DIR+2 bits 3-7, will still contain the year offset from 1980; however, it will be ANDed with 7 to incorporate, at most, the three low-order bits. Thus, after 1987, the date will be continued on to 1999 for the DOS; however, other DOS products will still see dates between 1980 and 1987. The dating facility provides the greatest degree of forward and backward compatibility.

The disk type byte (GAT+0CDH) now assigns bit 3 as the date type bit. If this bit is set, the disk is assumed to use the new time/date fields. The new versus old dating style is logged by the @CKDRV DOS service call. Knowledge of the disk type is maintained in a YFLAG$ byte - with bits 0-7 assigned to indicate the status of disk drives 0-7. Under LS-DOS 6.3, the YFLAG$ is accessible via the @FLAGS service call. Under LDOS 5.3, the YFLAG$ is fixed at 475DH.

Note that the above changes introduce time stamping to the DOS. This time stamp indicates when a file was created or last written to by hour and minute. That information, coupled with the modification date, should prove quite useful for file maintenance. It is also suggested that those users not already doing so, should activate the TIME prompt on BOOT as applicable to ensure that the system time is "realistic".

Since MISOSYS acquired the rights to LDOS with the acquisition of the retail operation of Logical Systems, it was our decision to make the LDOS release more similar in command syntax to the corresponding DOS 6 product. This was due to a couple of good reasons. The strongest argument was the many Model 4 users operating occasionally in the Model III mode. The next strongest argument was that the DOS 6 release was an enhancement of LDOS 5. Thus, many of you expected a lot of change in this upcoming release. I don't think that we will disappoint too many folks. There are, of course, those who will never be satisfied. Here's what you'll find in LDOS 5.3

The LDOS 5.3 release is a major enhancement from the 5.1.x release of LDOS. Although there have been improvements to most library commands,

utilities, and BASIC, the upgrade kit does not provide documentation as replacement pages to your user manual. You will be provided documentation covering the changes since the 5.1.4 release. A completely new user manual may be made available at a future time.

The master disk contains an imbedded **serial number** which is displayed (between the disk name and disk date) when booting the disk. This serial number will be maintained on every BACKUP copy of your LDOS disk. It is this serial number which should be registered with MISOSYS. **Absolutely no support for LDOS 5.3 will be provided by MISOSYS without that number.**

Technical Changes for the Programer

A new DOS service call, **@CMNDR**, has been provided. This function allows for the execution of programs correctly written to maintain the stack, execute entirely within the region 5200H-5FFFH, and provide proper terminating conditions. The vector is at 429CH. The proper terminating condition is placed in register pair HL. A Ø indicates successful completion. Any other value is considered to be an error exit. If JCL is in execution (as determined from SFLAG$), then an abort exit should be directed to @ABORT rather than via a RET with HL <> Ø.

A new flag, **CFLAG$**, has been provided. This flag is at address 4758H. Bits have been assigned as follows:

Bit-1: If set, @CMNDR is executing. This flag is reset by @EXIT, @ABORT, @CMD, and @CMNDI.

Bit-2: If set, it indicates that the command interpreter in SYS1 is requesting the line input from the keyboard. This condition is important for keyboard filters that may change the resident system overlay.

Bit-3: If set, then the system is requesting execution from either the "SET", "FILTER" or "SYSTEM (DRIVER=" commands. This bit should be tested by drivers or filters upon installation to ensure that they are being installed by the proper system command rather than just by RUN or execution.

Bit-4: If set, then the @CMNDR service call will execute only system LIBrary commands. Bear in mind that "RUN" will be invokable which could then be used to explicitly override the limitation.

Bit-6: If set, then @ERROR will not display any error message.

Bit-7: If set, then @ERROR will pass the error message to the buffer pointed to by register pair DE instead of displaying it. This provides a facility to capture the DOS error diagnostics in string form. The diagnostic is forced to the abbreviated mode.

For those Model 4 folks using LDOS along with variously provided hardware memory interface software, LDOS 5.3 is providing an OFLAG$ for maintenance of a memory management port image. The only function performed by the DOS is to zero this byte at BOOT time. It's address is 40ADH.

One last change in the DOS was instituted to permit the classification of both a data disk (providing up to 254 free file slots) and a system disk (providing up to 240 free file slots). For large drive capacity users, this frees up 14 additional directory slots on data drives. The directory maintains a bit to so indicate the state of the disk. GAT+0CDH, bit-7 will be set to indicate a data disk, reset to indicate a system disk. A DATA disk is turned into a SYSTEM disk via the BACKUP utility.

LIBrary command changes

In **ATTRIB**, the UPDATE password field has been redesignated as the OWNER password field - this is a name change only. The access password field has been removed from all files, so the ACC parameter is no longer valid on 5.3 disks. The DOS now assumes a blank access password for all files. If a protection level has not been assigned to a file, full access will automatically be granted regardless of any owner password. The OWNER password will still be required for full access on password protected files that have a protection level other than FULL. To have

a file that allows no access whatsoever without the use of the OWNER password, change the protection level to "NO"

The **AUTO** command has had several improvements. You can now install, display, or invoke an AUTO command on any drive - not just the SYSTEM drive. The **CLOCK** command has been moved to the TIME library command. The **CLS** command has been added to the library. It clears the video screen.

In **COPY**, when copying from a 5.1.x or earlier version disk to a 5.3 disk, the old access password, if any, will be removed and the 5.3 style date/time information will be established, the time being set to 00:00:00.

A new parameter, fill, has been added to **CREATE**. This parameter allows you to specify a particular character to propogate throughout the created file. For **DATE**, the acceptable range of dates is from January 1, 1980 through December 31, 1999. The **DEBUG** command has been improved to allow for the removal of the extended debugger from resident memory provided the memory-resident module was the most recent module installed into memory.

The **DEVICE** command has had both its display and its operation improved. A new parameter, N, has been added and the display has been paged. The **DIR**ectory command has been considerably enhanced. The display now includes the time stamp and byte EOF position for each file. Also, certain characteristics of the disk itself are displayed (such as density, number of heads, number of cylinders, number of free file slots, number of total file slots, free space, total space).

The **DO** command has been improved. First, parameter strings of any ASCII character are acceptable by enclosing them within double quotes. Second, JCL compilation will search for the first available drive which is not write protected for writing the SYSTEM/JCL file.

The **FORMS** command has been added to allow you to display or alter the operating parameters of the PR/FLT forms filter once the filter has been installed.

The display format of the **FREE** command has been improved. It will now display the identical disk configuration information as displayed by the header of the DIR command.

The **KILL** command has been improved to allow you to specify more than one device specification on the command line. The **LIB** command has been improved to always produce a correct display regardless of the compressed spaces vs special characters state of the video driver.

The **LIST** command has been considerably revised. The hexadecimal display mode now produces a combined hexadecimal/ASCII display of a complete 256-byte record. The older format is still available by turning off the "compressed" display mode via the "C=OFF" parameter. Both the hexadecimal and ASCII display modes will produce paged displays, stopping when the display screen fills. The "N" abbreviation for the "NUM" parm has been eliminated. The "N" parameter now allows an override to the paged display to produce a non-stop display. This operation is similar to the "N" parm of the old (and new) DIR command. Finally, the "TAB" parameter has been enhanced to accept a tab column number between 1 and 32 with a default of every 8 columns.

MEMORY had its display improved to report on the modules resident in protected memory. This requires that all installed modules adhere to the LDOS design specifications of memory resident modules (this spec has been a part of the LDOS system since January 1, 1982).

ROUTE has been improved in two ways. First, a new parameter, REWIND, has been added which allows you to start the routed file at its beginning. Second, the command has been improved by allowing it to reuse the previously installed but currently unused high-memory support module of a previous ROUTE of the same device.

The **SETCOM** command has been added to the library. It allows you to change and/or display the parameters of the RS232 device driver after the driver has been installed into memory. Coupled with the ability to execute library commands from within

LCOMM, this should be a biggie!

SYSTEM - The SYSTEM command, used to select DOS features, has been improved by the inclusion of additional parameters and enhancement of some existing ones. The **ALIVE** module can now be removed from memory. The **DRIVE** parameter now pertains to the **SYSGEN** parameter so that the configuration may be targeted to a specific drive. The Model III (**FAST|SLOW**) parameter has been adapted to utilize the Model 4 hardware clock speedup while still maintaining an accurate time clock. A **SMOOTH** parameter has been added. It alters the floppy disk driver so that the system interrupts are disabled earlier than what would otherwise occur. This has the effect of providing faster I/O with disk drives precisely aligned to 300 rpm where extra sector retries would be necessary. Note that when SMOOTH is turned ON, you will not be able to type ahead during disk I/O nor will you be able to effectively use dump-to-disk ON with LCOMM even at 300 baud. A **SWAP** parameter has been added that, with DRIVE, allows you to switch the logical drive assignments of any two drives even with a JCL command line. The command is functional while JCL is in execution even if one of the referenced drives holds the executing JCL file. If one of the designated drives is the SYSTEM drive and JCL is in execution, the other designated drive must contain a SYSTEM. The **SYSRES** parameter has been improved to have the system overlay use only the amount of high memory normally used by the overlay when it is resident in the system overlay region. The **TRACE** command has been removed as a separate LIBrary command and is now a parameter of SYSTEM. The **TYPE** parameter was altered to just inhibit the type-ahead operation rather than removing the type-ahead task.

The **TIME** command has been enhanced to provide for turning on or off the video screen clock display. **TOF** has been added to the library. It will emit a form feed character (12d) to the *PR device. If the printer is currently unavailable, the command does nothing.

Utility command changes

BACKUP has been changed to allow it to construct a SYSTEM disk from a formatted DATA disk. The VIS parameter has been dropped. The INV parameter has been altered to designate that files invisible to the directory are to be included as well as visible files. The SYS parameter has been altered to designate that system files are to be included in addition to visible files. SYS also reconfigures a DATA disk into a SYSTEM disk by allocating directory entry codes for /SYS files in the Hash Index Table.

FORMAT has been modified to generate a DATA disk after formatting. DATA disks reserve only two file slots out of the total number of directory slots available. SYSTEM disks, configured by the BACKUP utility, reserve 14 additional directory slots for /SYS files. This facility provides 14 additional file slots for DATA drives over that previously available under earlier releases of LDOS.

The **LCOMM** communications program has been enhanced via the addition of parameter control over the codes used for XON and XOFF. Also, LCOMM now makes use of the new @CMNDR command-and-return vector of LDOS. This feature provides you the ability to access LDOS library commands while running LCOMM (good for SETCOM access!).

PATCH has been enhanced to support the required finding of bytes matching a particular "F-format" patch line prior to installing a "D-format" patch. A new parameter, OPTION, allows you to force or inhibit the required matching. Coincidentally, a new parameter, REMOVE, has been added to un-install a "D-format" patch that was installed with the appropriate required matching of "F-format" patch lines.

The Text EDitor (**TED**) is a full screen "quick" text editor which has been added to LDOS. It is a full-screen ASCII text editor with many typical word-processing type features (four-directional cursor movement; two-directional scrolling; text insertion and overstrike; string search and replace; block copy, delete, and move; directional delete; large text buffer;

etc); however, TED was not designed to be a full featured word processor. TED was designed for you to be able to rapidly enter a full-screen text editing environment while accomplishes many of your text file editing tasks. Those of you familiar with our Mister ED application pac for PRO-WAM™ will recognize that TED/CMD is the TED/APP application ported to the Model III as a command file program. It provides the facility of editting ASCII text files such as JCL, KSM, and FIX files; it is very useful for those LDOS users not already owning a text editor or word processor.

The **KI/DVR** keyboard driver has been improved to allow the key combination, <CLEAR><SHIFT><Ø>, to generate a code of 16Ød - the same as <CLEAR><SPACE>. Note that it will not toggle CAPS LOCK; only <SHIFT><Ø> will toggle CAPS LOCK. Another enhancement to LDOS will restrict KI/DVR from being inadvertantly loaded by issuing it as a command; it requires installation via the SET library command.

The **RS232x/DVR** serial driver has been enhanced to provide for the selection of a received logical BREAK character rather than arbitrarily using a code of Ø1d. The driver parameters may also be altered after the driver is installed by using the SETCOM library command.

The **MINIDOS/FLT** keyboard filter has been improved to allow the "repeat last DOS command" to properly function regardless of the filter's position in the keyboard device chain.

The **PR/FLT** printer filter program has been enhanced to allow its operating parameters to be changed after the filter has been installed into memory.

A **HELP** facility has been added to provide on-line help information for both DOS commands and disk BASIC statements/functions. There are two HELP files: DOS and BASIC. Together they provide 117 screens of help.

The **DATECONV** utility program has been added to convert DOS data disks earlier than this 5.3 release to the extended date and time stamp usage. SYSTEM disks must first have the DOS files moved to them via BACKUP then processed by DATECONV.

BASIC enhancements

The disk BASIC interpreter has been improved by several additions and the name has been changed to "BASIC/CMD". First, two more edit functions have been added. These are the ability to copy a BASIC program line to another line number and the ability to move a BASIC program line to another line number. No automatic renumbering of imbedded line number targets is done so you will still have to adjust the targets of any GOTOs, GOSUBs, etc. These two edit additions have been added to the BASIC/OV3 overlay file so that overlay must be present on-line for you to be able to use those two commands.

INPUT@ provides a much-requested facility of controlling keyboard input to a string variable with prompting and screen field highlighting. Two forms of INPUT@ are supported. One provides the minimal improvement of allowing the input prompt to appear starting at a designated video position - similar to a PRINT@ statement. The second form is more powerful. It allows you to specify an input field width, an input field fill character, whether the input should be alphanumeric or just numeric, and whether the input should automatically terminate when the "field width" number of characters have been entered rather than requiring a hard <ENTER>. The two forms of INPUT@ are:

```
INPUT@pos[,"message"];var[,var]
INPUT@pos[,"message"],fw,"$|#[*][f]";var$
```

Finally, the **CMD"V"** BASIC extension can be used to dump a list of active variables and their values and user defined functions while a program is running (or after it was interrupted or ended).

All in all, you will find that this upgrade kit provides an unbeatable value of performance for the price. The LDOS 5.3.0 upgrade kit will be shipping in January 1987 for the TRS-80 Model III/4/4P and MAX-80. Specify your machine. Your old disks do **NOT** have to be returned. The price? A low $24.95 + $2 S&H ($3 Canada, $6 foreign).

Note: A Model I compatible release of LDOS 5.3 will not be available unless there will be a minimum of 1000 purchasors; thus, if you are interested in such a release, please provide us with a firm order request and get your name on the list. No orders will be returned nor will notification be made. A Model I release will require a double density equipped machine; some features of LDOS 5.3 may require lower case hardware support.

DED86™ - a disk/file/memory editor

Rich Deglin has been working on this one for our MS-DOS product line. DED86 has been groomed with features that should make it a winner of a product - and a most versatile and powerful tool for you. The fundamental purpose of a tool such as DED86 is to allow you a sector-oriented disk editing environment where your changes are not recorded until you "save" them. DED86 essentially brings together many of the features common to our TRS-80 editing product line. It has its roots in Karl Hessinger's DED. It has many of the operational characteristics found in our File EDitor, FED. It also has the memory editing feature of our PRO-WAM application, MED. We also added some features unique to the MS-DOS operating environment such as sub-directory access. After looking at a number of product reviews of file recovery utilities, I also had Rich incorporate a powerful subfunction called KEEP. This function allows you to examine a disk or file and flag any sector or cluster to be kept in a list. At any time, the sectors making up that list can be written to a new disk file. How's that for making recovery easy? The memory editor facility even allows you to dump a memory region to a disk file - useful for text recovery after a program crash.

Let me bring up a little of the DED86 highlights. First of all, invoking DED86 automatically logs in the current directory or the one specified on the command line. This brings up an editing display similar to the following [note that the contents of this sector is not the actual contents found on the drive identified by the status]:

```
  0123456789ABCDEF   Byte    00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
=================================================================================
| ..&!.........&$.. | <000> | 00 1E 26 21 00 10 01 0F 03 CD BB 04 26 24 01 01 |
| ....&........!p.: | <010> | 01 CD BB 04 26 14 04 0C CD BB 04 FD 21 70 14 3A |
| ...o.Ad.I. ....E  | <020> | D8 07 CB 6F 01 41 64 11 49 1D 20 05 CB A8 11 45 |
| ..w ....o.'. ..$  | <030> | 11 CB 77 20 0C CB B0 CB 6F 11 27 0F 20 03 11 24 |
| ..~.....w..q..r.  | <040> | 09 FD 7E 03 E6 03 B0 FD 77 03 FD 71 04 FD 72 07 |
| .s.:...w._..&#..  | <050> | FD 73 08 3A 02 04 FD 77 09 5F 06 00 26 23 CD BF |
| ..K.#x. ....w.>"  | <060> | 04 ED 4B CC 23 78 E6 20 FD B6 04 FD 77 04 3E 22 |
| ..w.%.....:.#..2  | <070> | 81 FD 77 06 25 06 04 CD BF 04 3A 00 23 E6 10 32 |
| .$!....>..2..~2.  | <080> | 0F 24 21 00 13 06 10 3E 10 90 32 E0 07 7E 32 E1 |
| .#..>.!..U..>...  | <090> | 07 23 10 F3 3E 04 21 00 15 55 5D F5 3E 09 F1 01 |
| .....>.....>....  | <0A0> | 00 04 CD D6 04 3E 06 CD D6 04 0B 3E 02 CD CE 04 |
| ...>......$....H  | <0B0> | 11 00 0C 3E 01 CD CE 04 C3 00 24 1E 00 18 01 48 |
| .W...!........... | <0C0> | AF 57 FD E5 FD 21 D8 07 CD 03 00 FD E1 C9 D5 E1 |
| .2..6 ..2........ | <0D0> | 13 32 DC 07 36 20 D5 C5 32 DC 07 ED B0 C1 D1 C9 |
| ................. | <0E0> | 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |
| ................. | <0F0> | 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |
=================================================================================
Drive: C: Cyl: 0    Head: 1  Sector: 0    Logical sector: 17     Cluster: 0
Byte: X'000' = X'49' = 01001001 = 73   File sector: 0      File byte: 0
File: /
Cwd:  /
Command:          Ignore case: OFF  Entry/display mode: DEC  Write protect: OFF
```

The display is busy with information; however, the command entry highlights the location of each entry prompt in reverse video directing your eyes about the status

display. The sector display shows you, at any one time, 256-bytes of a 512-byte sector - either the first half or the second half. You see the ASCII values of the sector bytes displayed on the left and the hexadecimal values displayed on the right. Status concerning the cursor positioning, as well as its entry, can be displayed/entered in either decimal or hexadecimal. The "File:" field always shows you what file is assigned to the sector currently being displayed.

DED86 provides a host of commands allowing you to move easily about the sector, the disk, the file, or the directory hierarchy. The help screens displayed later on will shed a little light on this flexibility. For now, here's a tip on easily moving to different files. When DED86 logs a directory, your display is first positioned to the disk location of that directory. Anytime the cursor is positioned over any byte of the directory record, a simple <!> [exclamation point] will automatically begin editing the corresponding file, if a file, or will log the specified directory if its a directory entry. You, of course, can switch to the file edit mode by the <E> command.

The "Keep sectors" facility can be extremely useful when you want to recover data from a damaged disk. The damage may have been one of an inadvertant ERASE (major oops) or a disk crash (major woe is me). You can keep a list of recovery sectors which will be written to a file (on some other drive) at your direction. When your data is visibly understood by examination, you can easily scroll through the disk drive and "keep" any sector you determine to be part of your file. Keep gives you this subcommand selection:

<A>dd sector/cluster to the list
<R>emove sector/cluster from the list
<L>ist the sector numbers being kept
<S>ave the kept sectors to disk
<C>lear the kept list
<N>ext sector in list
<P>revious sector in list
<H>ome> to first sector in list
<E>nd to last sector in list

At any time DED86 is awaiting a command entry, a simple <RETURN> will bring up the first help screen. This provides a display similar to the following:

```
        ========================================================
        | DED - MS-DOS Disk EDitor - for IBM PC - Version 1.0d |
        ========================================================

 ==============================      ==============================
|       Disk positioning       |    |       File positioning       |
|------------------------------|    |------------------------------|
| <+><=> Next sector           |    | <HOME>     To file start     |
| <->    Previous sector       |    | <END>      To file end       |
| <N>    Next cylinder         |    | <PAGE-UP>  Previous sector   |
| <P>    Previous cylinder     |    | <PAGE-DN>  Next sector       |
| <R>    Select sector/cluster |    | <J>        Select sector/byte|
|==============================|    |==============================|
|       File operations        |    |       Byte positioning       |
|------------------------------|    |------------------------------|
| <E> Examine file             |    | <UP>         Up one row      |
| <C> Change working directory |    | <DOWN>       Down one row    |
| <T> Touch file date/time     |    | <LEFT>       To previous byte|
|==============================|    | <RIGHT>      To next byte    |
|       String Search          |    | <CTL-HOME>   To byte 0       |
|------------------------------|    | <CTL-LEFT>   To start of row |
| <F> Find string: disk mode   |    | <CTL-RIGHT>  To end of row   |
| <Q> Find string: file mode   |    | <@>          Directly to byte|
| <G> Go to next occurrence    |    |                              |
 ==============================      ==============================
```

Some of the data in this screen may be different in the released version of DED86

as this QUARTERLY information was prepared prior to the addition of the memory editing capability. The "Touch file date/time" command may need a little explanation. It allows you to change the date/time field maintained by MS-DOS in the directory of a file corresponding to the time the file was last written to.

Other than that, the help screen should prove sufficiently explanatory for your insight to the commands presented.

A second help screen is available by a <RETURN> from the first screen. It looks something like this.

```
=========================================================
| DED - MS-DOS Disk EDitor - for IBM PC - Version 1.0d  |
=========================================================

==================================    ==================================
|           Miscellaneous          |  |          Sector Editing          |
|----------------------------------|  |----------------------------------|
| <BRK><ESC> Cancel command        |  | <H>      Hexadecimal edit        |
| <D>        Select new drive/file |  | <A>      ASCII edit              |
| <K>        Keep sectors submenu  |  | <U>      Undo last edit          |
| <V>        Verify sectors        |  | <INSERT> Insert byte             |
| <L>        Print sector          |  | <DELETE> Delete byte             |
| <O>        Output formfeed       |  | <Z>      Zero to sector end      |
| <!>        Edit file under cursor|  | <S>      Save edited sector      |
| <B>        Bring up DOS subshell |  ==================================
| <X>        Exit DED              |  ==================================
|                                  |  |          Mode Selection          |
|                                  |  |----------------------------------|
|                                  |  | <W> Toggle write protect         |
|                                  |  | <M> Toggle entry/display mode    |
|                                  |  | <I> Toggle ignore case in search |
==================================    ==================================
```

Note the availability of the "Bring up DOS subshell" command. This allows you to temporarily enter a subshell of DOS and return to DED86. What can you do from this? Well just about anything! Go into BASICA and run some programs. Do some backups. Invoke the Little Brother Data Manager. Even invoke DED86 again!

As noted before, you also have memory editing capabilities - all with the protection of not effecting changes until you say SAVE. And if WRITE PROTECT is toggled ON, you have another safeguard.

DED86 requires MS-DOS version 2.x or 3.x and a PC-compatible (whatever that means these days). DED86 ships in January of 1987. The suggested retial price of this product is $59.95 + $2S&H ($3 Canada, $6 foreign). However, until April 30th, 1987, its introductory price saves you ten bucks. At $49.95, DED86 needs to be in your MS-DOS software library. Order DED86 [M-86-020] now!

ED/ASM-86™ - MS-DOS™ assembler

Phil Oliver (Oliver Computing Company) has been working on this one since 1984. Earlier versions of ED/ASM-86 have been in use for almost two years now. MISOSYS is now going to publish the newest version with a user manual completely re-written by Phil in tutorial form. ED/ASM-86 provides an assembly language environment much different from that of the Microsoft MASM/MLINK environment. Old TRS-80 EDTASM programmers who have converted to MS-DOS will love this exciting environment.

The ED/ASM-86 program is a full macro assembly language development system for 8086/8087/8088 based computers. It is an integrated package comprising two editors, assembler, linker, and debugger for the PC and compatibles as well as generic MS-DOS based machines. It includes support for the 8087, 80186, and 80286 instruction sets for assembly and disassembly as well as the 8088/8086 base including conversion of decimal numbers to 80-bit temporary

real 8087 floating point format.

The guiding philosophy in the development of the package was to create a powerful, easy to use, and reasonably fast assembly language development system, with most of the Intel™ and Microsoft™ assembly language "standards" available if needed, but not required if a programmer does not want to waste the unnecessary time and effort needed to conform to some of those standards. Given these premises, the nature of the program naturally follows.

The line **EDITOR** is a full featured line editor with intra-line editing, block move and copy, partial save and load, and more. It optimizes for assembly language by efficiently tokenizing the instructions, saving memory and disk space and speeding assembly. The editor is quite similar to the editor of our EnhComp BASIC compiler - also a product originally authored by Oliver. If you have a Color Graphics Adaptor (CGA) installed, ED/ASM-86 will automatically detect it via the ROM BIOS current video state call and then enable some expanded features which are otherwise not available. First of all, ED/ASM will use direct access to video memory for all of its screen I/O, which is much faster than putting up with the usual slow ROM BIOS video call. Second, the editor and debugger now do screen I/O in two separate memory pages, allowing for rapid switching back and forth as well as preserving the contents of one screen while working in another.

If your machine is equipped with a color graphics adaptor, a second full screen **EDITOR** is available. This editor allows typical screen editing functions. You can even switch between the screen editor and the line editor with a keystroke. Also in CGA mode, an /SCR command sets an option which with the ED/ASM-86 screen editor and separate debugger windows available, switches to color graphics screen Ø whenever a "GO" command is issued. On "INT 3" standard breakpoint, the screen will be switched back to the debugger screen (screen 1). This allows a reasonable amount of transparent debugging with screen intensive applications such as a word processor or a menu screen. The entire 16K CGA buffer is saved and restored, along with the running program's video mode. The /NSCR command turns the "/SCR" option mode off. When executing program code with the "GO" command, there will be no screen switching by ED/ASM-86. When testing a routine which does not write to the screen, this may be the preferable mode although functionally there is no difference either way between "/SCR" or "/NSCR".

The **ASSEMBLER** is a powerful macro assembler with structured assembly language support, direct .COM or .EXE write capabilities, and direct assembly into memory capability. It is largely MASM compatible, but more logical.

Since ED/ASM-86 was designed to be a powerful production assembler for potentially large assembly language programs, there are three methods of including disk file data in a single assembly with a single output file. This greatly expands the limits of the size of an assembled machine language program by transferring the burden of storing source code from RAM to disk files; thus the primary system constraint is the size of your disk space. The requirement for RAM is reduced to that of data tables needed during assembly, such as symbol storage.

ED/ASM-86 has four output options, mutually exclusive. Three of them are outputs to disk files. You can specify output as either of the "standard" .COM or .EXE type files, directly, or as an ED/ASM-86 specific "link" file. The fourth option is assembly to memory. With this option, the assembly output is loaded into the next segment directly following the in-memory source code. This last option is a very powerful feature of ED/ASM-86. Once assembled into memory, you can invoke the built-in debugger. You can go from the debugger to the assembled program at any time via the debugger 'GO' command, and return to the debugger via the "INT 3" instruction. Naturally, you can switch back and forth from the program editor to the debugger at will.

The assembler fully supports all 8087 instruction data types: 16 bit integer, 32 bit integer, 32 bit floating point (SHORT REAL), 64 bit floating point (LONG REAL),

10 byte (80 bit) BCD, and 80 bit temporary real.

With ED/ASM-86, your "program" can consist of just about any logically related collection of in-memory source code, INCLUDEs, LINKs, and INJECTs, all with a single program output - which may itself be an in-memory assembly, or a .COM, .EXE, or .LNK file! The power of this ability is clear. It goes beyond the conventional assembly concepts to a much freer, and therefore more powerful system, bypassing the conventional arbitrary constraints of a linear link of modules. Naturally, conventional linear module linking is easily possible with ED/ASM-86 simply by having a list of LINK pseudo-ops - you just aren't forced to do it, and you have more direct control over the physical placement of the modules.

The **DEBUGGER** provides a wide range of debugging facilities, including symbolic disassembly of arbitrary code, disassembly of previously assembled in-memory program, 8087 register display in scientific notation, 8087/80186/80286 disassembly, direct assembly to memory, use of complex expressions with previously defined symbols and/or register values, and much more.

Besides the "standard" debugging commands, you get: (1) The ability to directly assemble at any location in memory, one statement at a time; (2) A symbolic disassembler which optionally generates labels and can write the disassembled code to disk as a source file for ED/ASM-86; (3) The ability to modify the disk loading parameters such as sector size; (4) The ability to save all of the registers in a special area and recall them as desired; (5) The ability to calculate an integer arithmetical expression, including symbols used in the last assembly; the ability to change, independently, the default input and output radixes to any of base 2, 8, 10, or 16; (6) And the ability to input or output either a byte or a word to any I/O port.

The **LINKER** is an all in one special ED/ASM-86 instruction (LINK or MLINK) that takes an ED/ASM-86 link file, which is a highly compressed version of virtually any source file, which may include complex expressions involving multiple external symbols, and treats it as though it were the original source file, allowing easy absolute control over the placement of all your source code, with fast assembly. ED/ASM-86 allows ANY logical combination of in-memory source code, INCLUDEs, and LINKs, and even a special pseudo-op (INJECT) which allows verbatim insertion of a file as data into the output stream; and can produce a single .COM, .EXE, .LNK, or in-memory program from these combinations.

The best thing of all is that **ED/ASM-86** is all in one program that takes up about 60K of memory (excluding source text and assembly data tables). For users with non-IBM compatible machines, the additional "EMS.COM" program provided is a strictly MS-DOS version of ED/ASM-86. It contains all of the essential features of ED/ASM-86 but does not include the fast screen output that the IBM (with graphics card) version has in it. This version requires the use of the ANSI.SYS device driver.

We expect to release this updated version of ED/ASM-86 in January 1987. Until April 30th, 1987, the introductory price is $89.95 plus $5 S&H ($6 Canada). If you want to get into xxx86 assembly language on your MS-DOS machine, this is the assembler for you. Order ED/ASM-86 [M-86-030].

Across the C

Concerning separate compilation of modules with the MC compiler: Everything you need to know about separate compilation is stated in the MC manual on pages 5-7 and 5-8. MC/JCL is set up to LINK a program with main(). You can still use MC/JCL to compile any function by using the (C) parameter for compile only. The Tech section on 5-7/5-8 shows you the exact MRAS and M80 command lines to assemble any function which does not have main() as well as provides examples of linking a main() function with 3 other separately compiled and assembled functions. Don't forget, the other thing you can do is to bundle your separately compiled and assembled functions into a library for automatic linking. The docs on USERLIB explain this. Finally, the cc/ccc program listed in the last issue of THE MISOSYS QUARTERLY can be used to automatically generate the Job Control Language file necessary to compile, assemble, and link more than one module. Bob Zinn has also provided us with an updated version of cc which has been mentioned in THE BLURB and is on the DISK NOTES 6.

(JJS to RS) Roy, if a C function that is being compiled separately from the main function has references to stdin and stdout, like with fputs() and fgetc(), is it necessary to #include stdio/h?

(RS to JJS) No. The file pointers, stdin, stdout, and stderr are not defined in stdio/h. The FILE DESCRIPTORS are. Thus if you are using the stream functions, you don't need stdio/h for them. You may, of course, be needing some of the other constants defined in stdio/h.

(JJS to RS) Thanks for clearing that up. I suppose the other constants you're referring to are things like "TRUE" and "EOF", right ?

(RS to JJS) Yes, and things like the file descriptors, STDIN, STDOUT, STDERR.

Concerning the O_KBECHO flag value of the option() function: "1" is the correct value for the 2nd argument (not in quotes). The reason is that TRUE is 0xffff whereas option() is looking for a Ø, 1, or 2. For those folks always wanting to have KBECHO set to ON, an easy way is to change the option bit in the MC/ASM file. That's where it's defined. Shouldn't take too much figuring to find out what bit it is. Let me save you the time. O_KBECHO is defined as 4 in stdio/h. That stands for bit 4. The options are stored in a word labeled as $OPTION in the MC/ASM and M80/H files. Bit 4 set would be a value of 16. Therefore, change the DW argument to 16 and the KBECHO option is permanently defaulted to ON and will stay ON unless overridden by an option(O_KBECHO,Ø).

The Great Communicator

Concerning NULL Modems used to hardwire two computers via the RS232 port: (JJKD to AHP) There are all different possible configurations for a null modem. Which is the "Right" one? Why, the one that works, of course. But seriously folks, the particular style of null modem necessary depends on what and how much line level handshaking the two pieces of equipment expect each other to honor. The configuration that I've used to hook a Model 100 to a Model 1 and/or a Model 4 is:

```
1 --------- 1
2 --------- 3
3 --------- 2
4 --------- 5
5 --------- 4
7 --------- 7

6 -
  |
8 ---------- 20

20 ---------- 6
              |
              - 8
```

Don't forget to turn DTR and RTS on at the Model 4 end of things.

The DOS Connection

DEW was trying to transfer LDOS from a single density Model I disk to a double density Model I disk on a single-drive

system. Of course, there is no facility for doing this. However, in trying, DEW came up with a "bug" in CMDFILE. The documentation provided with the 5.1.4 update stated that CMDFILE was enhanced to be able to load the entire library file of SYS6/SYS or SYS7/SYS by entering an X'FF' in response to the ISAM number request. True, CMDFILE would then load the entire library; however, it would abort after reading and issue an error message, "Not a command file". Here's my solution which I directed to DEW.

This may be too late for your use since by now you probably have acquired a second drive. On the other hand, I can address the problem you were having with CMDFILE. First, even if CMDFILE would have been successful in loading SYS6 and SYS7, your efforts would still have been fruitless as CMDFILE cannot be used to copy system files into the directory slots they require for the same reason that COPY cannot be used. It takes a very special piece of code to move SYSTEM files. SYSx/SYS files (as well as BOOT/SYS and DIR/SYS) use specific hash index table positions. BACKUP is the only facility for moving these files onto a newly formatted disk.

On the other hand, I can still come up with a solution to CMDFILE's inability to deal with the two library files. True, the documentation does state that they are loadable. Actually, the entire file does load; but since the SYS6 and SYS7 files are missing the "02 02" transfer record used by CMDFILE to ascertain the end of the "load module", it aborts. The easy solution is to add the transfer record to the end of the file. Here is a patch to accomplish this.

```
PATCH SYS6/SYS.SYSTEM (D33,1C=02 02 00 00)
PATCH SYS7/SYS.SYSTEM (D26,4B=02 02 00 00)
```

With these patches installed, CMDFILE will load the entire library file and allow you to copy it to a previously made system disk.

Using FED to locate the end of file on Model III LDOS SYS6/SYS and SYS7/SYS files, anyone can easily come up with those comparable patches.

BG wanted a method for patching a module which is part of a CONFIG/SYS file without having to rebuild the configuration. On the one hand, that's a good reason to create a JCL file to develop your configuration. With JCL, you can always easily recreate your current configuration. Here's another method which I related in my response to BG.

You would have to use FED to examine adjacent bytes in the file identified by the patch then look for that sequence of bytes in the CONFIG/SYS file. That's another use for FED. Other than that, there is no way.

The right route and other matters: (DS) I know I've probably missed this somewhere, but isn't it possible to route the screen display (*DO) to the screen AND a file? I've needed this a couple of times and can't seem to get it to work on my Model 4 with good 'ole TRSDOS 6. Another question. I'm getting the Alpha MegaMem board. How easy is it to get PRO-NTO to work with that? I'd like to get it if I know I've finally got the memory.

(RS to DS) Use the ALPHA1.FIX patches found in DL3 for @BANK. That way, PRO-NTO will search up to the first 7 banks for a free one. The way to get *DO output to both the screen and a disk file is to use a link. First ROUTE a dummy device to a disk file then link *DO to that dummy device.

```
ROUTE *ZZ VIDFILE/TXT
LINK *DO *ZZ
```

after that, you got it.

Dialog on passwords: (KW) Roy, What do you mean about eliminating the USER or ACCESS password field in order to accomodate dates up to 1999? Do you mean passwords will be eliminated or just that there will only be one password in the future? Although I personally don't use different passwords as a general rule, I feel like I would if others at my office were using the computer.

Surely there are several users out there who use one password for access to a file and a different password to make any changes to the file. Although it would take some of the protection out, why not just "hash" the password down to one byte instead of two? That way you could still keep both passwords and free up one or two bytes in the FPDE if you are needing more room.

Or better yet, since DIR+30 only has two valid values (FF or FE), why not use four or five of the remaining bits for the year of modification. That way you could handle dates for 16 or 32 years!

(RS to KW) Sorry, but DIR+30 uses more than just those two values. You have to understand the manner of calculating disk sectors from directory information and I can't get involved with that lengthy discussion here. Dropping the USER password does not drop password protection. You will still have the OWNER password and the USER PROTection level. Most users do not use both password fields. When a file is protected, it usually has a non-blank OWNER password and a blank USER password with a PROT level of READ, or WRITE. Thus, no password entry gives READ or WRITE access, in that example. This won't change in x.3.

There's always a RETURN when you don't want it department: (JB to RS) Why in the world does SYS1 output a CR on a CMDNI?

(RS to JB) It happens that the code is common to both @EXIT and @CMNDI. Actually, a little patch could be applied to bypass that piece for @CMNDI; change the value at X'1E40' from 71 to 7B. This will cause it to bypass the code which checks for a '.' comment to suppress the CR. I personally can't think of any reason for it to output the CR at that point for @CMNDI.

The other side of life: (GP to all) Is this a bug or have I missed something...doing BACKUP on a two-drive system running LDOS 5.1.4, I used the following command:

BACKUP :1 :0 (X)

The catch was that the source disk (drive 1) was double-sided, while the system disk (drive 0) was single-sided. The destination disk, which I inserted when prompted, was formatted double-sided. The backup ran normally, verified, and confirmed completion, then prompted for reinsertion of the system disk into drive 0. No error messages, just a return to LDOS Ready. BUT the backup copy created had only been copied on one side of the disk. Some files had directory entries but no contents other than blank sectors, while others had no directory entry at all. I tried again and got the same result. No error messages, but the backup is no good. I just lost a bunch of data by trusting the utility when it was lying to me. What have I missed?

(RS) I believe that you fooled BACKUP. Perhaps I can introduce some complaining or resolving code into 5.3. That problem was, I believe, corrected in the TRSDOS 6.2 release. It should be corrected in LDOS 5.3.

(JJKD to GP) Whenever the disk in drive zero is swapped with a disk of differing density or number of sides, it must be logged. Now, you as a user can't execute the LOG command from inside of BACKUP, that part's obvious. What's not obvious is that BACKUP won't log the disk for you (unfortunately). This was at least partly fixed in TRSDOS 6.2, and hopefully might be completely fixed in TRSDOS 6.3 and LDOS 5.3. The solution? Use a double sided system disk in drive zero before starting the BACKUP, or use QFB instead.

(GP to LM) I didn't try the :0 to :1 sequence. After finding out what had happened, I started switching to TRSDOS 6.2 for the backups. It did not display the same problem, but apparently detects the switch between single and double sided diskettes automatically. I would have thought LDOS could do the same, or at least would have told me it was going to try a "reconstruct" to put the double-sided data onto a single-sided format. The output diskette from LDOS was still double-sided in format, but only contained track images from side 0 of the original!

(GP to RS) Yes, it seems that is what happened. What I don't understand is, if BACKUP thinks the destination disk swapped into drive 0 is a single-sided disk because the system disk swapped out was single sided, why doesn't it do a "reconstruct" instead of just duplicating one side of the source disk and leaving the destination formatted as a double-sided diskette? It seems as though it is successfully recognizing that the destination is double-sided, but still somewhere along the way disables the write to the second side. Presumably this is akin to the problem in earlier versions of TRSDOS 6 when you tried a similar stunt; only TRSDOS got the backup right and crashed when you reinserted the system disk because it then expected the system disk to be double-sided. I note that the TRSDOS bug has been corrected in 6.2 or 6.2.1, at least it doesn't happen to me any more.

(RS to GP) These are all good questions. Good answers escape me at the present. I have the scenario logged in on the tack board which I refer to while working on 5.3.

(GP to JJKD) I'd forgotten about QFB - used to use it, but (this is a BBS system running on a model 3) when the old single-sided drives started to go flaky a few months back they wouldn't format reliably any more. Consequently, I couldn't be reformatting every time I did a backup. (Backup disks were formatted by my 4P). Then I realized how much faster it was to do BACKUP instead of QFB when the disks weren't very full, so I abandoned QFB entirely. Yes, I figured using a double-sided system disk would have worked, but part of the problem was that I was in the process of building one! I botched it, and I admit that I should have been suspicious about doing a swap like that, but it works in TRSDOS 6.2 so I just didn't think about it until too late. And it does seem as though LDOS should realize there is something fishy - either do a reconstruct and change the destination disk to single-sided like the system disk or else complain about the situation. After all, the X parameter declares your intention to swap disks on it, so it would seem that an automatic LOG is in order.

(JJKD to GP) QFB is always faster than BACKUP+FORMAT, just make sure that you specify "don't copy unallocated granules". QFB is not necessarily faster than BACKUP alone if the destination disk need not be formatted. Yes, the disk should be logged, isn't in 5.1 and should be in 5.3. Remember that all versions of 5.1 are at least a year older than 6.2, and I don't think that any significant changes were made to 5.1 BACKUP since 1983 or so.

(LM to GP) Actually, the problem is only partially in Backup. The disk sector numbers are duplicated on both sides of the disk, and when a disk is not logged in as being double sided, a read or write of a sector on the back side will access (successfully) the same sector on the front side. Some eight inch disks use a side-verify in the sector header, but this is not available on the five inch drives. Backup actually goes through the motions of writing both sides, but because the drive is not logged correctly (fixed in 6.2) only one side is actually accessed.

(GP to LM) My disk formatting data shows that the 5" drives still have a space in the sector header to indicate the side. Also, on duplicating the "problem" with the LDOS BACKUP, I find that it most certainly is NOT reading both sides or writing both sides, even if they ended up on the same side. The proof lies in the fact that when it backs up a double sided disk to another double-sided disk, it can only move 5 tracks at a time (or rather 5 cylinders) but when it backs up single-sided, it moves 9 tracks at a time. Under the circumstances in question, BACKUP reads cylinders 0 through 9 from the INPUT double-sided disk BEFORE it ever writes to the output disk!

(RS to All) This problem is fixed in LDOS 5.3!

Forms, anyone? (BB to all) I have only recently begun trying to make use of FORMS/FLT (with a DW II). It works fine -- for one file. The next time around, the line counter picks up where it left off. The only solutions I've found for this are

to reboot (which is a nuisance, since I have a long, tedious startup routine) or to run the following /JCL:

```
RESET *PR
RESET *FF
SET *FF TO FORMS/FLT FILTER
*PR *FF FORMS
```

Is there an easier way to get that counter back to zero?

(HB to BB) How about just typing TOF at the DOS prompt to move to the top-of-form?

(JJKD to BB) First off, the last thing that your program should send to the line printer is a TOF to advance the paper. If you are manually advancing the paper, that is what is throwing off the sync. A formfeed can be sent by using the DOS command "TOF". KSMPlus, on the LS-Utility disk (available from MISOSYS), will do the same with a <CLEAR><SHIFT><T> from the keyboard. Finally, if you really want to zero the line counter, a decimal 6 (in BASIC that's CHR$(6);) will zero the line counter.

Number 5, Number 5,...: (HO) Could anyone tell me the significance of a "Attempted to read system record" error I got while running a friend's computer under LDOS from remote? This happened while I was copying a file from one hard drive partition to another. Would this indicate a need for a reformat of the hard disk?

(RS to HO) Sounds like that hard drive partition got the unused portion of the directory cylinder de-allocated and some other file got written to it. If you can find out what file got written there (use MAPPER from MACH2), then move it elsewhere and then stuff an X'FF' into the GAT for the directory cylinder.

(HO to RS) Roy, Thank you very much for your prompt and helpful reply. Your response does raise another question, though: What is MACH2 (I'm a relative newcomer to TRS-80)? From what you tell me, the middle (directory) cylinder must not contain anything but the directory, even though the directory only uses 32 of 255 sectors on that cylinder. I'll let you all know how I make out with this.

(RS to HO) MACH2 is a product of ours which allows you to easily (via menu) control where files are to be placed on a disk. The reason that the directory allocates the entire cylinder even though it only needs 34 (note the 34) sectors max, is because of the simulation of the directory data address mark convention on hard drives. Of course, even on floppies with greater than 34 sectors per cylinder, the entire directory cylinder is allocated for the directory. On your hard drive allocated with 256 sectors per cylinder, there is a little wasted space. On the other hand, DESCRIBE, another of our utilities, can make use of that "wasted" space by utilizing it for extensions to the directory. It adds a 63-character descriptor for each file.

4P Mystery revealed on Compuserve!: (Adam Rubin 71320,1052 to All) OK, folks, here it is! Have you ever wondered why your 4P's "boot from hard disk" option never seemed to work? The solution is here! HDBOOT.DOC, now available in DL0 ("General/New Uploads"), explains how your 4P can boot directly from a RS 5-meg without a floppy. The file includes instructions, system requirements, and limitations. Because of the low-level tinkering involved, this project is recommended for experienced users ONLY!

Speaking of booting...: (VD to all) I work with 17 Model IIIs in a middle school classroom, and just replaced the C-ROMs (vintage spring of '82) with replacement ROMS from RS. The new ROMs are easier to use with our Network III HD system. They work fine, except that the keyboard driver appears to go haywire when I boot one of the computers independent of the network under LDOS 5.1.3. The keys produce consistently inappropriate characters, e.g., 1=6, 2=#, 0=TAB, etc. When I replace the new ROMs with the old ones LDOS works fine. Does anyone have a suggestion as to what might be the problem with the new ROMs?

(JJKD to VD) The network III bootable "C" ROM alters the structure of the keyboard

driver because of the built-in "universal" or "international" keyboard support. If you SYSGEN the KI/DVR on a "normal" machine, that disk should be OK on the new "C" machines.

A time to grow...: (BP to RS) I think it's about time LDOS/TRSDOS admitted TIME is significant. Perhaps a later version could maintain separate creation and access date/time (in addition to mod).

(RS to BP) Now I really don't think the directory would permit that. After all, we couldn't dump the OWNER password also!

Oh why do we put up with misbehaved application software? (KB to all) I am attempting to put TRSDOS system overlays into high memory. I have tried this with my Model 4 and a Megadisk hard drive, my model 4 with floppies and a model 4P with floppies. I am using the "SYSTEM (SYSRES=n)" command and have "SYSGEN" the configuration, use an AUTO command and both methods yield poor results. Most programs I use after putting the overlays in high memory crash at some point. PROFILE 4 PLUS, in particular gives me problems, although I have had problems with Radio Shack's Accounting Programs writer as well. I'm not a "computer whiz" but I can usually follow the TRSDOS manual fairly well, and I can't seem to locate what I'm doing wrong. Can anybody help? Thanks a lot.

(JB to KB) You cannot use much high memory (which putting the overlays in ram qualifies as "using a LOT") with Profile, 'cuz it doesn't bother to CHECK high memory before overwriting it. It is not a "well behaved program".

(PJ to KB) Kevin, Welcome to the PROBLEMS WITH RADIO SHACK club. I too encountered problems with high memory and overlays, and I too am using a Megadisk from Software Support. The problems are not with the hard drive, nor are they with TRSDOS, which is amazingly sophisticated considering the limitations of 64/128 K. Every time I have had a problem with system crashes, I have traced it to R/S software that I have been using. So, first, the driver for the Megadisk resides in high memory. Apparently there is not enough room in low memory to hide it there, so it takes up a fair chunk when you boot up. You can see this by typing MEMORY immediately after boot. HIGH does not = FFFF, but some lower value. In my case I have a mess of drivers and filters up there, so I wind up with HIGH$ = F2xx something. So, any program that you load must honor that HIGH$ pointer, and not use the memory above that. Otherwise you get what we have gotten - CRASH! Neither Profile 4+ nor Deskmate honor the HIGH$ pointer.

I "auto" do it: (KV to all) Can anyone tell me how to use the "AUTO" command with a hard drive. I have tried: AUTO (file name) but the desired program does not load when the HD is booted up.

(RS to KV) Under TRSDOS 6, the AUTO command storage is in the System Information sector of BOOT/SYS. This is read off of the BOOT drive. Under LDOS 5.1.x, the AUTO command storage is in the GAT sector of DIR/SYS. But the AUTO command is read off of the BOOT drives DIR/SYS file before the CONFIG/SYS file is loaded. Thus, in both cases, the AUTO command must be installed on the drive which is used for BOOTing - the floppy in that case! Under TRSDOS 6.x, just issue the AUTO install command as:

AUTO :b your-command-line

where ":b" references the disk drive which currently contains the booting floppy. Under LDOS 5.1, you need to switch back to the boot floppy as the SYSTEM drive then issue the AUTO command. This, of course, will change in LDOS 5.3 to match the capability of TRSDOS 6.

Double, double, toil and trouble: (JB to RS) MicroSoft FORTRAN only uses single-precision, even when you explicitly declare everything as DOUBLE PRECISION. not very honest of them, is it?

The case of ACCURACY versus PRECISION: (RS to JB) I don't think that the term

"accurate" can be applied to floating point math routines. The term that should be applied is "precision". The problem is that most folks tend to not understand how imprecise floating pont can be for certain work. They just assume that because a computer does the calculations, it has to be correct. However, fp math is correct only to a certain quantity of digits - that's precision. What you will find when you evaluate the fp routines on one implementation versus another is not so much how good the routines are but to how many digits of precision are they keeping. Some math packs use a 3-byte fractional part and 1-byte exponent (MS), others use 4&1. The double precision folks use either 7&1 or 6-7/8's & 1-1/8 (9-bit exponent). Definitely, double precision math evaluation is going to show up as more precise than single if you are only going to look at 10 digits. Also, some language implementations give dp functions - some do not. Don't forget, some sp functions have only 5-digits of precision. Errors can really compound when you cascade inverse functions. The real point is recognizing the imprecision of your calculations and understanding the validity of the result.

(JB to RS) This was pointed out in the original DDJ columns on the "Savage Benchmark". It was also pointed out that SOME compilers' FP routines generated "imprecisions" of opposite sign, which would therefore cancel out, making the overall benchmark look a lot better than the actual precision available. With the Savage benchmark, everything cancels out (in theory). How close the particular compiler/interpreter under test comes to that ideal result is of interest, but won't necessarly blow a language implementation's usefulness in the "real world". After all, if accuracy were important in computer work, how could MicroSoft have survived all these years?

(RS to JB) When I was involved in engineering, your normal instrument was a slide rule. These "primitive" devices usually had 3-4 digits of precision. That was OK for most work. When you needed a few more digits, you went to a CRC handbook (that's a Chemical Rubber Company handbook, not Cyclical Redundancy Check). Thus, even in "Microsoft's world", 6-7 digits is good enough for most work. When you need more, you use doubles and write your own double precision functions. When more digits of precision are needed, you then resort to extended precision techniques. M/S has survived because we really don't need too many digits. The one's with the problems are the accountants. When you deal with billions and need to keep track of cents, the inaccuracies of fp cause them consternation. Techniques are available, however, to combat even that problem. Unfortunately, most folks tend to misunderstand the errors of floating point calculations. Perhaps that's because they're never disclosed to the "layman".

(HT to RS) Roy, I think I mentioned having three 12 inch diameter ROUND slide rules. Two were log-log-trig and the log scale spiraled around and around into the center. Much fun deciding where the correct answer was <grin>. The other was a special rule for Thermo heat transfer calcs. Beat the heck out of Mollier Diagrams, or pounding the tables in Keenan & Keys. We ancient slip stick jockeys knew what floating point was and writing our formulas in Scientific Notation with exponents to keep track of the elusive decimal point or how many zeros to append to the three place answers <grin>. When the early (and expensive) scientific calculators came out we had contests to see who was faster. Those who knew how to use an Abacus were the fastest <grin>. I held those HP and reverse polish calculators in distain. Then in 1976 I had a Model I, and learned Forth. RPN now seems second nature <grin>.

The LDOS SIG on Compuserve - PCS49

Fm: Alan H. Pesetsky 75675,1535

It seems they have eliminated the possibility of changing your line length for received messages in the Forum. Or am I missing something obvious in the OP menu?

Fm: jeff brenton 76703,1065

There is a new command at the Function:

prompt to set your screen width **FOR THE CURRENT SESSION ONLY.** It is

SET WID xx

where xx is the actual width: 64, 80, etc. One side effect of the new software is that EVERYONE should make a trip to DEFAULT (GO CIS-9) and review your terminal settings. These settings now follow you "EVERYWHERE" (certain minor exceptions), so they better be set the way you like them. Used to be that DEFAULT settings didn't matter, but you had to customize your terminal definition every time you entered a new forum.

Fm: Jim Kyle 76703,762

Your DEFALT settings now rule what goes on here as well as elsewhere. You can GO DEFALT and set up your line length. If you want different characteristics in different sessions, that is still possible but now it's a little more complicated. You gotta define alternate parameter lists from the OK prompt in the Personal File Area, and call for the one you want when you log in! If enough folk are interested i'll try to put together a file on how this is done and send it to Joe for DL0.

Fm: Nate Salsbury 72167,1750

I found "how to Get the Most out of Compuserve" by Charles Bowen and David Peyton, published by Bantam Books to be and EXCELLENT tutor for the many 'mysteries' of CIS. My copy (1 yr ago) cost $14.95. Also, Charles Bowen runs a Forum here to answer specific CIS-related problems with (almost) instant feedback (e.g. usually next day). GO OCC.

Fm: Adam Rubin 71320,1052

I have another few thousand questions. Here's some about the Forum software: 1) How can UA be used if Stop after messages = To me? 2) There doesn't seem to be much documentation on SET. Do SETtings apply to just the forum I'm in, or all over CIS? If I change any, is the change permanent? 3) In a message's subject heading, what does a "#" before the subject mean? (E.g. 68729 is "Sb: #MISOSYS Catalog.") 4) Even with the new software, my High Msg pointer *still* gets set incorrectly occasionally. Is there still supposed to be a problem with this? Thanks for all your time!

Fm: LDOS Support 76703,437

1) To use UA to reply to a message but change subject, and you are not the recipient of the message, you must OP;SM A;S 2) SET is systemwide, and they are not permanent. Some SET options can be made permanent, but generally this must be done via DEFALT (Go CIS-9) or in your PERsonal file area. 3) The # means that there is at least one unread reply in the thread. 4) The High message pointer is not set if you read via RI and some other selective read options, or are dumped from the system during a session. Doing a RR;0 before exiting will ensure that your high message pointer is set to the last message on the board. Note that it is possible for a message to be left between the time you finish reading and the time you are going to exit (especially if you are in the DLs in between), and you will miss it/them.

Fm: jeff brenton 76703,1065

1) UA can only be used at the ("UA RE T)" prompt. So, you must be set up to stop at the message in question to use it. If you set Stop on messages to you only, then you cannot UA to any others. 2) SET only applies to the current forum. If you want something to work system-wide, you must set it in DEFAULT, page CIS-9. SET is also for the current session only. 3) The #-sign means that there is at least one reply to the message. 4) The only Read commands that affect the high message pointer are RN, RT(n), RF and RR. RI and RM do **not** touch the HMC. If you only read the messages that are marked for you, then your HMC is not changed. If you can document a case where one of the commands that should update it doesn't, let us all know, because we can ask the forumware programmers about it. Ok, where are the other 996 questions?

Fm: Paul Jaeger 76010,2324

If you own both a Model 4 and a Model 100, you will want to download two new files in the LDOS sig, DL-3. X10024.cmd and X42100.cmd are assembly language programs written for the model 4 (TRSDOS 6.2) that facilitate the movement of ASCII files from one computer to the other. It is no longer necessary to SETCOM or SET *CL from the DOS prompt, as the two programs take care of all those boring details. The transfer is at 4800 baud, and file opening and closing is automatic for both computers. Documentation is in "X100.doc". And they are for free! Enjoy.

Fm: Michael E. Oliver 71056,1527

Some provision should be made to permit users to access library entries by filename. I was dismayed that after reviewing the Model 4/4p library directory, having noted programs that looked interesting, I could not browse them by requesting a specific filename (other than a direct download). Thanks!

Fm: LDOS Support 76703,437

Let's say that you saw a program that looked interesting called MOOSE.DAT. You can browse just this file by doing a BRO MOOSE.DAT and then <R>eading it, <D>ownloading it or whatever. Of course, it won't do you much good to <R>ead programs. In the situation where you only remember part of the filename, you can use CompuServe wildcarding, ala BRO MO????.???, BRO MO????.*, BRO *.DAT, BRO *.DA?, but not BRO MO*.DAT. In other words, you can use "*" to match any full name or extension, but for part of a name or extension you must use a "?" for each wildcard character position.

Fm: LDOS Support 76703,437

To estimate how long it takes to download any particular program, do this: (1) Use BRO to read the description of the file. (2) On the first line, along with the filename will be one number, optionally another number in parentheses, a date and another number. (3) If there is only one number before the date, take that number. If there is a second number in parentheses, take 95% of that (the second) number instead. (4) The result is the size of the file to be downloaded. (5) Divide by the effective character rate for your baud rate. A reasonable estimate is 25 for 300 baud and 85 for 1200 baud. (6) The result is the number of seconds the download will take. Of course, you can divide by 60 to get minutes. (7) Add 5% if you are using XMODEM instead of a CIS protocol. (8) Add at least 10% if you are logging on during CIS peak traffic periods.

Fm: LDOS Support 76703,437

In general, messages should be left so that they can help others. The exceptions are (P)rivate messages which should be deleted after reading, as nobody else can see them anyway, and one liner "thanks" messages. The space for messages here is finite, so following these rules should help the useful messages stick around longer.

Here's the new section topics set up by the sysop, JJKD (Joe)

0 General/NewUplds
1 Languages
2 MISOSYS Products
3 M1/3 Programming
4 Mod4 Programming
5 LDOS 5 Support
6 TRSDOS 6 Support
7 Applications
8 Hardware Hacking
9 Open Forum
10 MS-DOS Interest

Fm: Roy Brayton 73007,2575

I am a new subscriber to CompuServe and a new member of MISOSYS SIG. The other day I left you a message that I had intended to leave as "private". Later I was browsing through the public messages, and there was my private message, "big as life". It's fortunate that the privacy of the message was not very important. I believe I used the "SPU" parameter that the Help File

specified. Where did I go wrong?

Fm: jeff brenton 76703,1065

Did the message have a (P) in the header? YOU and the recipient of the message will always be able to see the message, even if it is private. However, anyone whose UserID is not in the header will not be able to see it. Private messages are NOT kept in a separate section of the board; just tagged to not let others read them. Also, only use SU or SPU if you are writting a message the contains information that MUST NOT be reformatted; otherwise, the Forumware cannot adjust the line lengths to suit the readers' terminals. 80-character lines look like hell when printed on 64-character screens! [use S or SP instead of SU or SPU]

Fm: Dan Vestal 71330,2730

The command,

```
DO MNETA/JCL
```

made a file on disk #1 MNETA/CMD. How does this program interface with this communication disk? Do I follow the same steps like the old way using the <CLEAR-,> <SHIFT-CLEAR-:> method, or does this program work when I select the CIS DL protocol A. Please start a step #1 for me and give as much detail as possible.

Fm: LDOS Support 76703,437

Ok, to use MNETA/CMD: (1) Enter the following commands:

```
SET *CL COM/DVR
SETCOM (W=8,P=N,BREAK=0)
MNETA *CL
```

for 1200 baud, add the BAUD=1200 parameter to the above SETCOM command as in

```
COM (BAUD=1200,W=8,P=N,BREAK=0
```

(2) You are now in the terminal program, and ready to go. Dial the phone and connect to CompuServe via the normal control-C dialog. Go to the desired forum, ala GO LDOS or GO PCS49. (3) Enter the desired DL area, let's say the TRSDOS 6 Utilities area, which is DL6. (4) Pick a file to download. As an example, let's use XMODEM.CMD. (5) Bring up the file via this command at the DL 6 prompt:

```
BRO XMODEM.CMD
```

6) Download the file. At the (R D M) prompt, specify D. (7) You will get a display of protocol options. Pick the CIS "A" option. If you do not get a choice, then your terminal type is not set right. You need to GO CIS-9 and set your terminal type to "OTHER". (8) You will be prompted for a filename for your computer. Use

```
XMODEM/CMD
```

Note the use of the slash instead of a period, 'cause the filespec is for your computer. We already specified the CIS filename (which is the one with the period) in step 5. (9) You will now receive the file. When the download is complete, a message to that effect will be displayed. That's it! You may now proceed to the next file to be downloaded.

The BANKER by Roy Soltoff

With the release of several memory expansion options available for the Model 4 (TRS-80), some MISOSYS customers have requested an **easy** method of placing specific external memory resident applications (like PRO-WAM) into specific memory banks. Here's the reason for that request. The original design of the Model 4 DOS architecture supported a maximum of 8 banks. The use of a single 8-bit byte for the BAR$ and BUR$ storage regions was fundamental to this limitation. My patches to TRSDOS 6.2 extended this banking scheme to support 32 banks for use with the Alpha Technology memory board. Associated with the patches was a relocation of the BAR$ and BUR$ to other memory regions using a 4-byte field for each.

The primary use of external memory by applications provided with the Model 4 DOS was a RAM emulation of a disk drive (memDISK) and a spool memory buffer for the library SPOOLer. Although the DOS architecturally could support 8 banks, the implementation of that DOS on the Model 4 limited it to 3 banks because the Model 4 hardware supported only 128K. Thus, those two applications were written to specifically limit themselves to banks totally within the 128K.

More advanced products like PRO-WAM use a single memory bank but scan for an available bank out up to bank number 8. Unfortunately, if PRO-WAM is installed before memDISK, it will automatically find and use bank 1. This limits memDISK to a single bank. Now if memDISK were to be installed first, it could use banks 1 and 2. This would allow PRO-WAM to select bank 3 (assuming your machine was supporting more than 128K). Of course, this would require you to install memDISK first - which you may not care to do. Another procedure would be to somehow reserve banks 1 and 2, install PRO-WAM, then release banks 1 and 2. BANKER allows you do do just that.

Banker will allow you to reserve or release a single bank or a range of consecutively numbered banks. BANKER will also display a map providing the status of all memory banks. This map looks like the following.

```
32K Banks avail = 01/05, In use = <-++++>
```

The utility is invoked by the command:

BANKER (Reserve=val,Free=val)

As can be observed, both parameters can be abbreviated to a single character. The "val" parameter can be either a single number or a number range entered as a string. For instance, the string "1-2" references banks 1 through 2 inclusive. You may also enter a single number as a string, as in "12". Thus, a typical command to reserve the first two external memory banks would be:

BANKER (R="1-2")

Both parameters may be entered with one command invocation. Reserve is processed first. Therefore, if your selections overlap, the released banks will be that specified by the FREE parameter. Note also that the status map will be displayed first with the existing status and will be subsequently displayed after each parameter is serviced.

BANKER does no error checking on your bank numbers other than ensuring that the starting bank number does not exceed the ending bank number in a string parameter entry. BANKER relies on the DOS to perform error diagnostics on the bank number passed in the @BANK service call (SVC).

The assembly language program which follows makes good use of the @PARAM service call and the extended parameter entry mode of V6. There are a few other techniques introduced which should prove educational to the assembly language programmer. This code can be assembled by either PRO-CREATE OR PRO-MRAS. Other assemblers may require some adjustment of the source (for instance, the SVC macro, the use of ".SHL." for a shift operator). A hexadecimal version of BANKER follows the source code. For those without an assembler, enter the hexadecimal version and use BINHEX to convert the hex file into an excutable binary file.

```
                00001 ;BANKER/ASM - 10/29/86
                00002 ;*=*=*
                00003 ;       BANKER - RAM bank utility
                00004 ;       Copyright (c) 1986 MISOSYS, Inc.
                00005 ;       All rights reserved
                00006 ;       Licensed for personal use only
                00007 ;*=*=*
000A            00008 LF      EQU     10
000D            00009 CR      EQU     13
000A            00010 @DSPLY  EQU     10
000C            00011 @LOGOT  EQU     12
0011            00012 @PARAM  EQU     17
001A            00013 @ERROR  EQU     26
0060            00014 @DECHEX EQU     96
0063            00015 @HEX16  EQU     99
0066            00016 @BANK   EQU     102
                00017 ;
                00018 SVC     MACRO   #NUM
                00019         LD      A,#NUM
                00020         RST     40
                00021         ENDM
                00022 ;
2700            00023         ORG     2700H
                00024 ;
2700 ED733727   00025 BEGIN   LD      (SPSAV),SP      ;Save stack
2704 310027     00026         LD      SP,BEGIN        ;Set stack
2707 E5         00027         PUSH    HL              ;Save INBUF$
2708 21D727     00028         LD      HL,HELLO$       ;Greet the people
270B            00029         SVC     @DSPLY
270B 3E0A       00030+        LD      A,@DSPLY
270D EF         00031+        RST     40
270E 115C28     00032         LD      DE,PRMTBL$
2711 E1         00033         POP     HL              ;Get INBUF$
2712            00034         SVC     @PARAM          ;Scan for parms
2712 3E11       00035+        LD      A,@PARAM
2714 EF         00036+        RST     40
2715 2023       00037         JR      NZ,PRMERR
                00038 ;
2717 CD7B27     00039         CALL    DSPBNKS         ;Display current config
                00040 ;*=*=*
                00041 ;       Routine to check for reserve-a-bank(s)
                00042 ;*=*=*
271A 210000     00043 RPARM   LD      HL,$-$          ;P/u any parm
271D 3A6D28     00044         LD      A,(RRESP)       ;P/u the response flag
2720 B7         00045         OR      A
2721 C44827     00046         CALL    NZ,RESFRE       ;Go make the reservation
                00047 ;*=*=*
                00048 ;       Routine to check for free-a-bank(s)
                00049 ;*=*=*
2724 210000     00050 FPARM   LD      HL,$-$          ;P/u any parm
2727 3E01       00051         LD      A,1             ;Change RESFRE to release;
2729 325227     00052         LD      (RESFRE+1),A
272C 3A6228     00053         LD      A,(FRESP)       ;P/u the response flag
272F B7         00054         OR      A
2730 C44827     00055         CALL    NZ,RESFRE       ;Do release, as required
                00056 ;*=*=*
                00057 ;       Exit procedures
```

```
                    00058 ;*=*=*
2733  210000        00059 EXIT     LD       HL,0
2736  310000        00060 ERREXIT  LD       SP,$-$
2737                00061 SPSAV    EQU      $-2
2739  C9            00062          RET
273A  3E2C          00063 PRMERR   LD       A,44            ;Parm error abort
273C  6F            00064 IOERR    LD       L,A
273D  2600          00065          LD       H,0
273F  E5            00066          PUSH     HL
2740  F6C0          00067          OR       0C0H            ;Abbrev & return
2742                00068          SVC      @ERROR
2742  3E1A          00069+         LD       A,@ERROR
2744  EF            00070+         RST      40
2745  E1            00071          POP      HL
2746  18EE          00072          JR       ERREXIT
                    00073 ;*=*=*
                    00074 ;        Routine to reserve or release bank(s)
                    00075 ;*=*=*
2748  CB6F          00076 RESFRE   BIT      5,A             ;String entry?
274A  2016          00077          JR       NZ,RSTRING
274C  CB7F          00078          BIT      7,A             ;Numeric entry?
274E  28EA          00079          JR       Z,PRMERR
2750  4D            00080          LD       C,L
2751  59            00081 RESFRE1  LD       E,C             ;Set for terminate
2752  0603          00082 RESFRE2  LD       B,3             ;Set to reserve a bank
2754  C5            00083          PUSH     BC
2755                00084          SVC      @BANK
2755  3E66          00085+         LD       A,@BANK
2757  EF            00086+         RST      40
2758  C1            00087          POP      BC
2759  20E1          00088          JR       NZ,IOERR
275B  79            00089          LD       A,C             ;Finished?
275C  BB            00090          CP       E               ;We are if start matches end
275D  281C          00091          JR       Z,DSPBNKS
275F  0C            00092          INC      C
2760  18F0          00093          JR       RESFRE2
2762                00094 RSTRING  SVC      @DECHEX         ;Decode start bank
2762  3E60          00095+         LD       A,@DECHEX
2764  EF            00096+         RST      40
2765  7E            00097          LD       A,(HL)          ;If terminator is a '"',
2766  FE22          00098          CP       '"'
2768  28E7          00099          JR       Z,RESFRE1       ;  then only one to reserve
276A  FE2D          00100          CP       '-'             ;  else check for xx-yy
276C  20CC          00101          JR       NZ,PRMERR
276E  23            00102          INC      HL              ;Bump past the dash
276F  C5            00103          PUSH     BC              ;Save start
2770                00104          SVC      @DECHEX         ;Decode end bank
2770  3E60          00105+         LD       A,@DECHEX
2772  EF            00106+         RST      40
2773  59            00107          LD       E,C             ;Save end bank in reg_E
2774  C1            00108          POP      BC              ;Get start
2775  7B            00109          LD       A,E             ;Validity check for s>e
2776  B9            00110          CP       C
2777  38C1          00111          JR       C,PRMERR
2779  18D7          00112          JR       RESFRE2         ;Go do it
                    00113 ;*=*=*
                    00114 ;        Routine to display bank usage
```

```
                00115 ;*=*=*
277B  012B1F    00116 DSPBNKS LD      BC,31.SHL.8+'+' ;Init the display string
277E  213B28    00117         LD      HL,INUSE$
2781  71        00118 LP1     LD      (HL),C
2782  23        00119         INC     HL
2783  10FC      00120         DJNZ    LP1
2785  0E00      00121         LD      C,0             ;Start checking from bank 1
2787  0C        00122 BANKLP  INC     C               ;Reg_B starts at 0
2788  C5        00123         PUSH    BC
2789            00124         SVC     @BANK
2789  3E66      00125+        LD      A,@BANK
278B  EF        00126+        RST     40
278C  C1        00127         POP     BC
278D  28F8      00128         JR      Z,BANKLP
278F  C5        00129         PUSH    BC
2790  48        00130         LD      C,B             ;Restore bank 0
2791            00131         SVC     @BANK
2791  3E66      00132+        LD      A,@BANK
2793  EF        00133+        RST     40
2794  C1        00134         POP     BC
2795  59        00135         LD      E,C             ;Save highest avail
2796  79        00136         LD      A,C
2797  CDCC27    00137         CALL    CVTDEC
279A  ED432D28  00138         LD      (INTENS$),BC
279E  213A28    00139         LD      HL,INUSE$-1
27A1  01FF00    00140         LD      BC,255          ;Find which ones are in use
27A4  23        00141 BANKLP2 INC     HL
27A5  0C        00142         INC     C
27A6  79        00143         LD      A,C
27A7  BB        00144         CP      E
27A8  280E      00145         JR      Z,BANKLP3
27AA  C5        00146         PUSH    BC
27AB  0602      00147         LD      B,2
27AD            00148         SVC     @BANK
27AD  3E66      00149+        LD      A,@BANK
27AF  EF        00150+        RST     40
27B0  C1        00151         POP     BC
27B1  20F1      00152         JR      NZ,BANKLP2
27B3  04        00153         INC     B               ;Count number available
27B4  362D      00154         LD      (HL),'-'
27B6  18EC      00155         JR      BANKLP2
27B8  363E      00156 BANKLP3 LD      (HL),'>'
27BA  23        00157         INC     HL
27BB  360D      00158         LD      (HL),CR
27BD  78        00159         LD      A,B
27BE  CDCC27    00160         CALL    CVTDEC
27C1  ED432A28  00161         LD      (UNUSED$),BC
27C5  211828    00162         LD      HL,BANKS$
27C8            00163         SVC     @DSPLY
27C8  3E0A      00164+        LD      A,@DSPLY
27CA  EF        00165+        RST     40
27CB  C9        00166         RET
                00167 ;
27CC  0E2F      00168 CVTDEC  LD      C,2FH           ;Init counter
27CE  0C        00169 BANKLP1 INC     C
27CF  D60A      00170         SUB     10
27D1  30FB      00171         JR      NC,BANKLP1
```

```
27D3  C63A      00172          ADD     A,3AH           ;Get units
27D5  47        00173          LD      B,A
27D6  C9        00174          RET
                00175 ;
27D7  0A        00176 HELLO$   DB      LF,'BANKER 1.0 - Copyright 1986 '
      42 41 4E 4B 45 52 20 31
      2E 30 20 2D 20 43 6F 70
      79 72 69 67 68 74 20 31
      39 38 36 20
27F4  4D        00177          DB      'MISOSYS, Inc., All rights reserved',LF,CR
      49 53 4F 53 59 53 2C 20
      49 6E 63 2E 2C 20 41 6C
      6C 20 72 69 67 68 74 73
      20 72 65 73 65 72 76 65
      64 0A 0D
2818  33        00178 BANKS$   DB      '32K Banks avail = '
      32 4B 20 42 61 6E 6B 73
      20 61 76 61 69 6C 20 3D
      20
282A  64        00179 UNUSED$  DB      'dd/'
      64 2F
282D  64        00180 INTENS$  DB      'dd, In use = <'
      64 2C 20 49 6E 20 75 73
      65 20 3D 20 3C
283B  2B        00181 INUSE$   DB      '++++++++++++++++++++++++++++++>',CR
      2B 2B 2B 2B 2B 2B 2B 2B
      2B 2B 2B 2B 2B 2B 2B 2B
      2B 2B 2B 2B 2B 2B 2B 2B
      2B 2B 2B 2B 2B 2B 3E 0D
                00182 ;
285C  80        00183 PRMTBL$  DB      80H             ;Indicate V6 table
285D  B4        00184          DB      0B4H            ;Accept: numeric, string, abbrev
285E  46        00185          DB      'FREE'
      52 45 45
2862  00        00186 FRESP    DB      0
2863  2527      00187          DW      FPARM+1
2865  B7        00188          DB      0B7H            ;Accept: numeric, string, abbrev
2866  52        00189          DB      'RESERVE'
      45 53 45 52 56 45
286D  00        00190 RRESP    DB      0
286E  1B27      00191          DW      RPARM+1
2870  00        00192          NOP
2700            00193          END     BEGIN
```

```
01020027ED733727310027E521D7273E0AEF115C28E13E11EF2023CD7B27
2100003A6D28B7C448272100003E013252273A6228B7C448272100003100
00C93E2C6F2600E5F6C03E1AEFE118EECB6F2016CB7F28EA4D590603C53E
66EFC120E179BB281C0C18F03E60EF7EFE2228E7FE2D20CC23C53E60EF59
C17BB938C118D7012B1F213E28712310FC0E000CC53E66EFC128F8C5483E
66EFC15979CDCC27ED432D28213A2801FF00230C79BB280EC506023E66EF
C120F104362D18EC363E23360D78CDCC27ED432A28211828 3E0AEFC90E2F
0CD60A30FBC63A47C90A42414E4B45522031 2E30202D20436F7079726967
68742031393836204D49534F5359532C20496E63017300282E2C20416C6C
2072696768747320726573657276656 40A0D33324B2042616E6B73206176
61696C203D2064642F64642C20496E2075736520 3D203C2B2B2B2B2B2B2B
2B2B2B2B2B2B2B2B2B2B2B2B2B2B2B2B2B2B2B2B2B2B2B2B2B2B3E0D80B44652
4545002527B752453453525645001B27000202002 7
```

BIN to HEX and back by Roy Soltoff

Many times there has been mention of a BINHEX program. I use the term a little loosely since the function of BINHEX has been implemented within quite a few separate programs. It's almost like XMODEM. The term BINHEX may be more known as a concept rather than a hard and fast specific program implementation. To the non-initiated, BINHEX is used to convert a binary file (i.e. any file which uses an 8-bit character code) to a file containing pairs of hexadecimal digits - one pair for each binary byte. Such a converted file, being made up completely using ASCII characters, uses a 7-bit code and can thusly be transmitted over a communications line limited to a 7-bit word length. A secondary utility of a hex file is that it can be treated as plain text. This makes it easy for editing with any plain vanilla (or sophisticated) editor.

The typical TRS-80 user was first presented with a simplistic BINHEX program by its appearance in the old LDOS JOURNAL [Volume I, Number 2] dated October 1, 1981. That copy of BINHEX was written by Tim Mann for use with LBASIC. For the nostalgia buffs, that program is reprinted here. It is actually usable under TRSDOS 6 with a little editing. If recollection serves me correctly, the next version of "BINHEX" added a checksum preceded with an asterisk when a binary file was converted to a hex file. The two-hex-digit checksum value was then detected by BINHEX when that hex file was converted to a binary file. If the recalculated checksum did not match that supplied with the hex file, BINHEX gave you a warning. This version was in an executable form compiled by a pseudo-compiler known as BASIC/S [sold by PowerSOFT].

Other versions of BINHEX have appeared. MISOSYS includes a program called HEX with the LS-Host/Term package. This does the 2-way conversion plus it converts to/from Intel format. Another version I'll print here is a derivative of Tim Mann's BINHEX usable with EnhComp or PRO-EnhComp. A change I made was to drop the "0" from the OPEN "RO" statement since EnhComp doesn't support that kind of OPEN and add an ALLOCATE statement. I also altered the GET statement since EnhComp requires an index variable [that's the reason for the FOR-NEXT loop]. The "PRINT#2," statements also were not syntactily correct for EnhComp - they were altered.

Maybe next issue we'll provide some changes to install checksum generation and detection. For now, here's the two versions of BINHEX.

```
10 REM -- Hex to binary/Binary to hex file converter
20 REM -- Tim Mann
30 CLS:PRINT:PRINT"Hex to binary/Binary to hex"
35 PRINT"    file converter":PRINT
40 CLEAR 5000
50 GOSUB 58000
100 PRINT "Type 1 to convert a binary file to hex"
110 PRINT "     2 to convert a hex file to binary"
120 PRINT:INPUT D
130 PRINT
140 ON D GOTO 400,200
150 GOTO 100
200 LINE INPUT "Hex file name: ";HF$
210 LINE INPUT "Binary file name: ";BF$
220 OPEN"I",1,HF$
230 OPEN"O",2,BF$
240 IF EOF(1) THEN 320
250 LINE INPUT#1,D$
255 IF D$="" OR D$="OK" THEN 240
260 FOR I=1 TO LEN(D$) STEP 2
270    PRINT#2,CHR$(FND2(MID$(D$,I,2)));
300 NEXT I
310 GOTO 240
320 CLOSE
330 PRINT:PRINT"Done":PRINT
340 GOTO 100
400 LINE INPUT "Binary file name: ";BF$
410 LINE INPUT "Hex file name: ";HF$
420 OPEN"RO",1,BF$,1
430 OPEN"O",2,HF$
440 FIELD 1,1 AS F$
450 FOR I=1 TO 30
455    IF EOF(1) THEN 505
460    GET 1
470    PRINT#2,FNH2$(ASC(F$));
480 NEXT I
490 PRINT#2,
500 GOTO 450
505 PRINT#2,
510 CLOSE
520 PRINT:PRINT"Done":PRINT
```

```
530 GOTO 100
58000 DEFFN
H1$(X)=MID$("0123456789ABCDEF",(X AND
15)+1,1)
58010 DEFFN H2$(X)=FNH1$(X/16)+FNH1$(X)
58040 DEFFN
D1(X$)=INSTR("123456789ABCDEF",LEFT$(X$,1))
58050 DEFFN
D2(X$)=FND1(RIGHT$(X$,1))+16*FND1(RIGHT$(X$
,2))
58070 RETURN
60000 END

REM Here is the EnhComp Version
10 through 35 same as above
40 CLEAR 5000:ALLOCATE 2
50 through 410 same as above
420 OPEN"R",1,BF$,1
430 OPEN"O",2,HF$
440 FIELD 1,1 AS F$
445 J%=1
450 FOR I=1 TO 30
455     IF EOF(1) THEN 505
460     GET 1,J%:INC J%
470     PRINT#2,FNH2$(ASC(F$));
480 NEXT I
490 PRINT#2,""
500 GOTO 450
505 PRINT#2,""
510 through 60000 same as above
```

SPLITLIB/CCC by Rich Deglin

Ever since MC was released for the Model I and III computers, a few folks have puzzled over the inability to load the LIBC/REL library file into MLIB (the REL module librarian provided with MRAS). That's understandable since LIBC is approximately 32K of modules and MLIB provides about 29K of free buffer space - some of which would be needed for table space.

Many moons ago, Rich worked up a C-language program to display the decoded bit stream of a Microsoft REL object module. If memory serves me correctly, the reason for that project was to aid in developing the librarian, MLIB. The decoding program was called DECODREL. Since that program had to interpret the object module bitstream of a REL file, I suggested that it wouldn't take too much effort to revise DECODREL to split a large library into smaller pieces. The smaller modules could then be loaded into MLIB for maintenance purposes. The DOS' APPEND library command can be used to concatenate two or more modules together (using the STRIP parameter) so that the separated files can be recombined.

What follows, then, is Rich's effort along these lines. SPLITLIB is written in C and can be compiled with MC. Actually, it is more easily compiled with PRO-MC. You MC'rs will have to split (no pun intended) the source modules of SPLITLIB into distinct files and separately compile them. Perhaps you can make use of the "cc" program provided in the last issue of THE MISOSYS QUARTERLY.

Now the real astute observer will note that SPLITLIB could be very easily reworked to become EXPLODE-LIB. What I mean is that a little re-write could produce a utility to take a REL library file apart and create a separate file for each module - most likely using the module name as the file name. Case 14, end program, in the special link item class, indicates when the end of a module has been detected. By using this indicator as the decision for terminating the output file and restarting another would turn SPLITLIB into EXPLODE. Add a little more code to use the module name to generate the output file and you have it. Of course that would require you to either buffer the REL bit stream in memory until the module name was detected, or rename a fixed name output file to the correct name. The former would certainly be more sophisticated than the latter.

By the way, note that SPLITLIB is copyrighted by Riclin Computer Products. You may use SPLITLIB for your own personal use; however. We will accept a derivative work for the EXPLODE facility based on Rich's SPLITLIB in consideration for the next issue of THE MISOSYS QUARTERLY. Here now is Rich's SPLITLIB.

```
/*
 * SPLITLIB/CCC - split a /REL library into manageable parts
 * Version 1.0 - 09/08/86
 * Copyright 1986 Riclin Computer Products. All rights reserved.
 */
#include <stdio.h>
#option MAXFILES    2
#option FIXBUFS     ON
#option INLIB       ON
#define FSPECLEN    14
#define ME          "SPLITLIB"
#define RELEOF      0x9E

char inspec[FSPECLEN+1] = "", outspec[FSPECLEN+1], symbol[9], *drive = "";
int byte, bitno = 7, currlength;
FILE *infp, *outfp = NULL;

main(argc, argv)
    int argc;
    char *argv[];
{
    static int maxlength;
    static char *p;

    printf("%s - Split /REL Library - Version 1.0 - %s\nCopyright 1986 Riclin Computer
Products. All rights reserved.\n\n",
        ME, __DATE__);
    if (argc < 3)
        usage();
    for (p = argv[2]; isdigit(*p++); )
        ;
    maxlength = atoi(argv[2]);
    if (--p - argv[2] > 5 || *p || maxlength < 10000)
        usage();
    if (argc == 4)
        drive = argv[3];
    if (drive[0] != ':' || drive[1] < '0' || drive[1] > '7')
        usage();
    if (! (infp = fopen(addext(strncpy(inspec, argv[1], FSPECLEN),
        "REL"), "r")))
            ioerror(inspec);
    printf("Reading input file %s\n", inspec);
    openout();  /* open first output file */
    getbyte();  /* prime the pump */
    for ( ; ; ) /* loop forever */
    {
        if (! getbit())     /* absolute item */
            skipbits(8);
        else        /* relocatable item */
            switch (getnum(2))
            {
                case 1: /* program relative */
                case 2: /* data relative */
                case 3: /* common relative */
                    skipbits(16);
                    break;
                case 0:     /* special link item */
```

```
                    switch (getnum(8))  /* compute ctl field */
                    {
                        case 0:     /* entry symbol */
                        case 1:     /* select common block */
                        case 3:     /* request lib search */
                        case 4:     /* reserved */
                            b_field();
                            break;
                        case 2:     /* program name */
                            b_field();
                            printf("Module %s\n", symbol);
                            break;
                        case 5:     /* define common size */
                        case 7:     /* define entry point */
                        case 8:     /* reserved */
                        case 6:     /* chain external */
                            a_and_b();
                            break;
                        case 9:     /* external + offset */
                        case 10:    /* define data area size */
                        case 11:    /* set counter */
                        case 12:    /* chain address */
                        case 13:    /* define program size */
                            a_field();
                            break;
                        case 14:    /* end program */
                            a_field();
                            if (currlength >= maxlength)
                                openout();  /* close current, open next */
                            if (bitno != 7) /* force to byte boundary */
                            {
                                bitno = 7;
                                getbyte();
                            }
                            break;
                        case 15:    /* end file */
                            fclose(outfp);  /* 0x9E already written */
                            exit(0);
                    }   /* end inner switch */
                }   /* end outer switch */
            }   /* end for */
        }

        a_and_b()       /* decode A- and B-fields */
        {
            a_field();
            b_field();
        }

        a_field()   /* decode A-field */
        {
            skipbits(18);
        }

        b_field()   /* decode B-field */
        {
            static int len, n;
```

```
    for (len = getnum(4), n = 0; n < len; ++n)  /* get symbol */
        symbol[n] = getnum(128) & 0x7F;
    symbol[n] = '\0';
}

skipbits(n) /* skip n bits */
    int n;
{
    while (n--)
        getbit();
}

getnum(power)   /* get a number <= 2*power-1 */
    int power;
{
    static int n;

    for (n = 0; power > 0; power >>= 1)
        n += getbit() * power;
    return n;
}

getbit()        /* retrieve next bit from file */
{
    static int bit;

    bit = bitest(bitno--, byte);
    if (bitno < 0)
    {
        bitno = 7;  /* bitno runs modulo 8 */
        getbyte();  /* next byte */
    }
    return bit;
}

bitest(bitno, value)    /* test bit version 2 */
    int bitno, value;
{
#asm
$BITOE   EQU    43H             ;BIT 0,E instruction
         $GA    BC,DE           ;bit # in C, value in E
         LD     A,C
         ADD    A,A             ;shift bit # left 3x
         ADD    A,A
         ADD    A,A
         OR     $BITOE          ;mask bit # into instr
         LD     ($BITINST),A    ;modify bit test code
         BIT    0,E             ;test bit in E
$BITINST EQU    $-1
         LD     HL,0            ;assume not set
         RET    Z               ;rtn FALSE
         INC    HL              ;rtn TRUE
#endasm
}

getbyte()   /* read next byte from input file, write out */
```

```
{
    if ((byte = getc(infp)) == EOF)
        if (ferror(infp))
            ioerror(inspec);
    putbyte(byte);
    ++currlength;
}

putbyte(c)
    int c;
{
    if (putc(c, outfp) != c)
        ioerror(outspec);
}

usage()
{
    abend("usage: %s infile[/REL] maxlength [:d]\n", ME, NULL);
}

ioerror(f)
    char *f;
{
    extern int errno;

    abend("%s: %s\n", f, sys_errlist(errno));
}

abend(format, a1, a2)
    char *format, *a1, *a2;
{
    fprintf(stderr, "%s: ", ME);
    fprintf(stderr, format, a1, a2);
    exit(1);
}

openout()
{
    static char outext[] = "Rnn";
    static int currout = 0;

    if (outfp)
    {
        putbyte(RELEOF);
        fclose(outfp);
    }
    outext[1] = ++currout / 10 + '0';
    outext[2] = currout % 10 + '0';
    strcpy(outspec, drive);
    if (! (outfp = fopen(genspec(inspec, outspec, outext), "w")))
        ioerror(outspec);
    printf("\nWriting output file %s\n", outspec);
    currlength = 0;
}
```

Enhance FORTRAN by Harry G Clayton, Jr

Here are the patches and enhancements to the Model 4 FORTRAN package that I promised you. Sorry it took me so long to get them to you. Some of the things I fixed are: a lack of an INKEY$ type function, a random number generator seeder (their random number generator always repeated the same sequence), and a BREAK key scan function (they had no way to abort execution. Nothing like losing control of your system and having to reboot, sort of brings back pre-LDOS memories). I also fixed the FORLIB module called "MODEL-", since some routines trying to do a DOS exit would hang the system.

I have included quite a few files, so I will describe them one at a time in alphabetical order. Most of the files are self explanatory but these notes provide additional useful information. The files are:

BREAK/MAC: This file is the M80 source code for a new module to be added to FORLIB/REL using MLIB. This adds the following important subroutines that were missing from the 3.44 release.

CALL BREAK(logical_variable) - Checks system keyboard flags, pauses for shift_@. Returns logical_variable true if break flagged, else logical_varible is false. I like to be able to Abort programs.

CALL RANDOM(integer_variable) - Uses system clock data fields to make a number that can be used as a new seed for the random number generator so that a different starting point in the sequence can be obtained each time a program is run. The file RNDMIZ/FOR is a subroutine which uses RANDOM to seed the random number generator.

CALL INKEY(logical_variable) - Scans keyboard, returns char in logical_variable if key was pressed, else returns null char.

DOKFLG/MAC: This file contains the M80 source code for the routines SCANFL and MSKFL which handle the system keyboard flags. This file is 'INCLUDE'ed in programs needing KFLAG$ support. I do it this way to make it easier to make an LDOS 5 version of progs I write.

FLL/JCL: This is my FORTRAN Compile and Link Load command file. DO FLL for help.

FRACTL/FOR: FORTRAN program that uses all of the enhancements. This program uses fractals to generate mountains. Needs Hi-Res graphics hardware. Contains examples of the subroutines added in BREAK module of FORLIB. Also, if this is compiled with a FORLIB having the original MODEL- module, the bug in the $EX and $EXIT entry points will be demonstrated (will lock up sys unless DEBUG is active).

LSVC/MAC: LDOS 6 equates. Is an INCLUDE file in all my M80 source files, so you need this to assemble BREAK, etc.

MACROS/MAC: My standard M80 MACROs included in most M80 programs. Needed to assemble BREAK, etc.

MODEL/MAC: Modified M80 source code for the MODEL- module in FORLIB. Fixes the location of the $EXIT label so that system will not 'hang' when routines use $EXIT. MLIB is used to replace the original MODEL- module in FORLIB with the /REL file produced by assembling this revised version.

RANTEST/FOR: FORTRAN program to test random number generator and the seed maker. The program outputs the results of two sequences, one that is seeded with -10.0, and one that is seeded with number from RANDOM. The odd number columns are the -10 sequence (always the same), and the even numbered columns are the RANDOM seeded sequence. Uses RANDOM to make new seed via the subroutine RNDMIZ.

RNDMIZ/FOR: FORTRAN subroutine currently 'included' in programs to provide a new starting point in the random number sequence. Uses RANDOM to make a new seed. This really belongs in FORLIB, but I hav'nt got around to putting it there.

Notice that most of these changes to the FORTRAN package would not have been

possible without DSMBLR and MLIB. Again many Thanks.

I will be experimenting with MC as soon as possible. MC is very impressive, although I've only touched the surface so far. Congratulations on another excellent product. It looks like it will become my standard language so I will be making an MC to GRPLIB interface at some point in the near future. I will send a copy of it to you when I have it working.

```
;*****************************************************************
;*  BREAK  Ver. 1.0              21  Aug  1985  *
;*     Routine for FORLIB to look for pause or <break>  *
;*     <C> 1985        Harry G Clayton Jr  /  symuli  *
;*****************************************************************
TITLE  BREAK  Ver 1.0
SUBTTL <C> 1985   Harry G Clayton Jr  /  symuli
        PAGE 60
        GLOBAL  BREAK, RANDOM, INKEY
        INCLUDE LSVC/MAC
        INCLUDE MACROS/MAC
        CSEG
BREAK:  PUSH    HL
        LD      HL,INIFLG
        LD      A,(HL)
        INC     A
        JR      Z,FIRSTT
        CALL    SCANFL
        POP     HL
        JR      Z,NOBREA
        LD      (HL),0FFH
NOBREA: RET
FIRSTT: LD     (HL),0
        CALL    MSKKFL
        POP     HL
        LD      (HL),0
        RET
        INCLUDE DOKFLG/MAC      ; get MSKKFL and SCANFL
        DSEG
INIFLG: DEFB 0FFH
;*****************************************************************
;*  RANDOM  Ver. 1.0             23  Oct  1985  *
;*     Routine for FORLIB to make new seed for the random *
;*        number generator.        *
;*     Use:  CALL RANDOM(INTVAR)    *
;*     and RANDOM returns seed in INTVAR a Integer Variable *
;*     <C> 1985        Harry G Clayton Jr  /  symuli  *
;*****************************************************************
        CSEG
RANDOM: PUSH    HL              ; save variable ptr
        LD      HL,BUFFER       ; pt to buffer
        SYSCALL @TIME
        EX      DE,HL           ; ptr to sys time data
        LD      A,(HL)          ; get secs
        DEC     HL              ; get 1/30th secs
        ADD     A,(HL)          ; secs + 1/30th secs
        INC     HL
        INC     HL
        ADD     A,(HL)          ; add minutes
        LD      E,A             ;  to sum
```

```
        LD      D,0
        INC     HL
        LD      A,(HL)          ; get hours
        LD      L,D
        LD      H,D
        LD      B,8
MULOOP: RRCA                    ; multiply sum by hours
        JR      NC,NOADD        ; test bit
        ADD     HL,DE
NOADD:  SLA     E
        RL      D
        DJNZ    MULOOP
        EX      DE,HL           ; put result in DE
        POP     HL              ; get result pointr
        LD      (HL),E          ; store result
        INC     HL
        LD      (HL),D
        RET
        DSEG
BUFFER: DEFS    8
        DEFB    ETX
;*****************************************************************
;*  INKEY  Ver. 1.0             23  Oct  1985  *
;*     Routine for FORLIB to scan the keyboard *
;*     Use:  CALL INKEY(LOGVAR)    *
;*     and INKEY returns char in LOGVAR a Logical Variable *
;*             or returns null char if no key pressed.  *
;*     <C> 1985        Harry G Clayton Jr  /  symuli  *
;*****************************************************************
        CSEG
INKEY:  PUSH    HL              ; save variable ptr
        SYSCALL @KBD
        POP     HL
        JR      NZ,NOCHAR
CHAROT: LD    (HL),A          ; store result
        RET
NOCHAR: OR      A               ; see if zero
        JR      Z,CHAROT        ; go if not error
        RES     7,A
        LD      C,A
        SYSCALL @ERROR          ; abort to DOS
        END

;*****************************************************************
;*   DOKFLG/MAC  <C> Harry G Clayton Jr / symuli  23 Oct 1984   *
;*      Routines to handle system and keyboard flags to  *
;*      find  <break>, <enter>, and <shift>@ (pause)  *
;*****************************************************************
; initialize Keyboard flags
INITKF: PUSH    AF              ; save REGS
        CALL    MSKKFL          ; go mask KFLAG$
        POP     AF              ; restore REGS
        RET
; scan Keyboard flags ----------------------------------------
SCANFL: SYSCALL @FLAGS
        LD      A,(IY+18)       ; sys flags, SFLAG$
SFIX0   EQU     $-2
```

```
        AND     020H            ; test sys flags to see
        XOR     020H            ;   if JCL in progress
        RET     Z               ; quit if JCL process
        LD      A,(IY+10)       ; key flags, KFLAG$
KFIX0   EQU     $-2
        BIT     0,A             ; <break>
        RET     NZ
        BIT     1,A             ; <pause>
        RET     Z
PLOOP:  LD      A,(IY+10)
KFIX1   EQU     $-2
        BIT     0,A
        RET     NZ
        SYSCALL @KBD
        OR      A
        JR      Z,PLOOP
        CP      ''
        JR      Z,PLOOP
        CALL    MSKKFL
        XOR     A
        RET
; mask B0, B1, B2 of keyboard flags -------------------------
MSKKFL: SYSCALL @FLAGS
        LD      A,(IY+10)       ; mask KFLAG$
KFIX2   EQU     $-2
        AND     0F8H            ; resets B0 <break>,
        LD      (IY+10),A       ;   B1 <pause>, & B2 <enter>
KFIX3   EQU     $-2             ;   key flags
        RET

;*****************************************************************
;*   LSVC6/MAC  <C> Harry G Clayton Jr / symuli   21 Jan 1985 *
;*   abbreviated SYSTEM Vectors for LDOS Version 6.2  *
;*****************************************************************
@ERROR  EQU     01AH    ; post and display error msg
@FLAGS  EQU     065H    ; get sys flag table base into IY
@KBD    EQU     08H     ; *KI INKEY Routine, scan keybrd
                        ;   return char or 0 if none; USES: AF,DE
@TIME   EQU     013H    ; get current time
SVC     EQU     028H
; end of file

;*****************************************************************
;*   MACROS/MAC  <C> Harry G Clayton Jr / symuli  23 Oct 1984 *
;*      MACROS          for LDOS 6.2     21 Jan 1985 *
;*****************************************************************
SYSCALL MACRO   SVCNAME
        LD      A,SVCNAME
        RST     SVC
        ENDM
SAVREG  MACRO   ALL             ; Saves GP registers
        IRP     X,<AF,BC,DE,HL>
        PUSH    X
        ENDM
        IFNB    <ALL>           ; Save All
        IRP     Y,<IX,IY>       ; Registers
        PUSH    Y
```

```
        ENDM
        ENDIF
        ENDM
RESREG  MACRO   ALL             ; Restores Registers
        IFNB    <ALL>           ; restore all registers
        IRP     X,<IY,IX>
        POP     X
        ENDM
        ENDIF
        IRP     Y,<HL,DE,BC,AF>
        POP     Y
        ENDM
        ENDM

TITLE   MODEL-
SUBTTL  Modified by  Harry G Clayton Jr / symuli   18 Aug 1985
;  This version changed to prevent lack of return to DOS
;       when used with GRPLIB
        PAGE 60
        GLOBAL  $CLSFL, $EC, $EX, $EXIT, $INIT, $IOFLG, $IOINI
        GLOBAL  $LINBF, $LNPTR, $PA, $ST, $TTYIN, $TTYOT
        EXTERNAL $DTBF, $FATAL, $IOERR
        DSEG
$LNPTR: DEFW    $LINBF
$LINBF: DEFB    0
        DEFS    80
$EC:    DEFB    0
$IOFLG: DEFB    0
BRADDR: DEFW    0
$CLSFL: DEFW    0
        CSEG
$INIT:  XOR     A
        LD      ($EC),A
        LD      ($IOFLG),A
        LD      H,B
        LD      L,C
        LD      (BRADDR),HL
        LD      HL,0
        LD      B,00H
        LD      A,100
        RST     28H
        DEC     HL
        LD      SP,HL
        LD      HL,(BRADDR)
        JP      (HL)
$IOINI: LD      A,($IOFLG)
        OR      A
        RET     NZ
        INC     A
        LD      ($IOFLG),A
        LD      HL,$EXIT
        LD      ($CLSFL),HL
        LD      HL,$LINBF
        LD      ($LNPTR),HL
        LD      (HL),0DH
        RET
$EX:    LD      HL,$EXIT        ; $EXIT used to be here which
```

```
        PUSH    HL              ; caused system to hang
        LD      HL,($CLSFL)     ; when exiting from some
        JP      (HL)            ; routines, particularly
                                ; GRPLIB routines
$EXIT:  LD      HL,0
        LD      A,16H
        RST     28H
$TTYOT: PUSH    BC
        PUSH    DE
        PUSH    HL
        PUSH    AF
        LD      C,A
        LD      A,02H
        RST     28H
        JP      NZ,$IOERR
        POP     AF
        POP     HL
        POP     DE
        POP     BC
        RET
$TTYIN: PUSH    HL
        LD      HL,($LNPTR)
        LD      A,(HL)
        CP      0DH
        CALL    Z,M30BE
        OR      A
        CALL    Z,M30BE
        INC     HL
        LD      A,(HL)
        LD      ($LNPTR),HL
        POP     HL
        RET
M30BE:  PUSH    BC
        PUSH    DE
M30C0:  LD      B,80
        LD      C,00H
        LD      HL,$LINBF
        LD      A,09H
        RST     28H
        LD      A,B
        OR      A
        JP      Z,M30C0
        LD      HL,$LINBF-1
        POP     DE
        POP     BC
        RET
$PA:    LD      A,08H
        RST     28H
        JP      Z,$PA
        LD      HL,M3135
        CALL    M3108
        CALL    M3101
        CALL    M311E
        LD      A,01H
        RST     28H
        AND     '_'
        CP      'T'
```

```
        RET     NZ
        JP      $FATAL
$ST:    LD      HL,M312F
        CALL    M3108
        CALL    M3101
        CALL    M311E
        JP      $FATAL
M3101:  POP     HL
        EX      (SP),HL
        CALL    M3113
        EX      (SP),HL
        JP      (HL)
M3108:  LD      DE,$DTBF
        LD      A,0DH
        LD      (DE),A
        INC     DE
        LD      A,0AH
        LD      (DE),A
        INC     DE
M3113:  LD      B,06H
M3115:  LD      A,(HL)
        LD      (DE),A
        INC     HL
        INC     DE
        DEC     B
        JP      NZ,M3115
        RET
M311E:  LD      HL,$DTBF
        LD      B,0DH
M3123:  PUSH    BC
        LD      A,(HL)
        CALL    $TTYOT
        POP     BC
        DEC     B
        INC     HL
        JP      NZ,M3123
        RET
M312F:  DEFB    'STOP  '
M3135:  DEFB    'PAUSE '
        END

CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
        PROGRAM RANTEST
        LOGICAL BRK,BREAK,PF
        LOGICAL A, F, H, L, MSG(40)
        REAL    R, DI, SR, VX, VY, DT, X(32), Y(32), DU
        REAL    RR, EK, GK, DX, DY, OX, OY, ET
        INTEGER KS, CX, CY, HF, XMAX, YMAX, TTY
        COMMON  X, Y, CX, CY, XMAX, YMAX, SR, HF, ET, PF
        DO 100 K= 10, 100, 10
        X(1)= RAN(-10.0)
        DO 10 I= 1, 27
        X(I)= RAN(10.0)
10      CONTINUE
```

```
        CALL ABORT
        KK= -K
        CALL RNDMIZ
        DO 20 I= 1, 27
        Y(I)= RAN(1.0)
20      CONTINUE
        CALL ABORT
30      FORMAT(6(G13.6))
        DO 40 I=1, 27, 3
        WRITE(5,30) X(I), Y(I), X(I+1), Y(I+1), X(I+2), Y(I+2)
40      CONTINUE
41      FORMAT(I6/)
        WRITE(5,41) KK
42      CALL INKEY(PF)
        IF (PF.EQ.(.FALSE.)) GO TO 42
100     CONTINUE
        STOP EXIT
        END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
        SUBROUTINE REM
        LOGICAL BRK,BREAK,PF
        LOGICAL A, F, H, L, MSG(40)
        REAL    R, DI, SR, VX, VY, DT, X(32), Y(32), DU
        REAL    RR, EK, GK, DX, DY, OX, OY, ET
        INTEGER KS, CX, CY, HF, XMAX, YMAX, TTY
        COMMON  X, Y, CX, CY, XMAX, YMAX, SR, HF, ET, PF
        CALL ABORT
        TTY= 5
318     WRITE(TTY,320)
320     FORMAT(/' Clear Screen (Y or N) ?  ')
        READ(TTY,322,ERR=318) J
322     FORMAT(A2)
        WRITE(TTY,332) J
332     FORMAT(A1)
        CALL ABORT
308     WRITE(TTY,310)
310     FORMAT(/' Scale factor ==>  ')
        READ(TTY,312,ERR=308),KS
312     FORMAT(I3)
        WRITE(TTY,312) KS
        CALL ABORT
200     WRITE(TTY,300)
300     FORMAT('1'//' Initial distance (planet radii) ==>  ')
        READ(TTY,302,ERR=200),DI
302     FORMAT(F7.3)
        WRITE(TTY,302) DI
9999    STOP Exit
        END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C       FORTRAN SUBROUTINES             23  October  1985
C               some useful routines
C                       Harry G Clayton Jr  /  symuli
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
        SUBROUTINE ABORT
        LOGICAL BRK,BREAK
```

```
        CALL BREAK(BRK)
        IF (BRK) GOTO 801
        RETURN
801     STOP    **BRK
        END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
        SUBROUTINE WAIT
        DO 2 J=0, 10, 1
        DO 1 I=0, 30000, 1
1       CONTINUE
3       CALL ABORT
2       CONTINUE
        RETURN
        END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
        INCLUDE RNDMIZ/FOR

CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C       RNDMIZ SUBROUTINE               23  October  1985
C               Seeds random number generator
C                       Harry G Clayton Jr  /  symuli
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
        SUBROUTINE RNDMIZ
        INTEGER SEED
        REAL    SD, DU
        CALL RANDOM(SEED)
        SD= FLOAT(SEED)
        DU= RAN(-SD)
        RETURN
        END

C       VARS/FOR
        REAL    FM, XP, YP, XS, YS, ZS, RC, RH, RS, VT, VC, VS
        REAL    XX, XO, XT, YY, YT, ZZ, Z2, ZT
        REAL    W3, X2, X3, Y2, Y3, Z3
        INTEGER D(70,40)
        INTEGER AX, AY, BX, BY, DD, D1, D2, DS, EX, EY
        INTEGER IB, L, LE
        INTEGER MJ, MX, MY, OC, PR, SK, XE, YE
        LOGICAL F1, MSG(11)
        COMMON  FM, XP, YP, XS, YS, ZS, RC, RH, RS, VT, VC, VS
        COMMON  XX, XO, XT, YY, YT, ZZ, Z2, ZT
        COMMON  W3, X2, X3, Y2, Y3, Z3
        COMMON  AX, AY, BX, BY, DD, D1, D2, DS, EX, EY
        COMMON  IB, L, LE
        COMMON  MJ, MX, MY, OC, PR, SK, XE, YE
        COMMON  F1, MSG
        COMMON  D
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C               Three Dimensional Fractals
C
C       Based on BASIC program appearing in "Creative Computing"
C       July 1985 p.78 '3-D Fractals' by Michiel van de Panne.
C
C       Converted to FORTRAN-80 by  Harry G Clayton Jr
C                       24 Oct 1985
```

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C       Requires the Model 3 or Model 4 Hi-Res Graphics board.
C       The following libraries must be searched during link
C       loading:
C                       FORLIB  (happens automatically)
C                       GRPLIB  (Hi-Res Graphics Lib)
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C       Modules 'included' at compile time:
C                       RNDMIZ/FOR
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
        PROGRAM FRACTL
        INCLUDE VARS/FOR
        CALL GRPINI(0)
        CALL RNDMIZ
        WRITE(5,30)
30      FORMAT(' Number of Levels (1 to 6)  ==> ')
        READ(5,31) LE
        IF (LE.LT.1) LE= 1
31      FORMAT(I2)
        WRITE(5,35)
35      FORMAT(' Make Ocean (1=yes, 0=no)   ==> ')
        READ(5,31) OC

C   init array
        DO 38 I= 1, 64
        DO 36 J= 1, 32
        D(I,J)= 0
36      CONTINUE
38      CONTINUE
        DS= 2
C       DO 40 N=1,LE
C       DS= DS + 2**(N-1)
C 40    CONTINUE
        DS= DS**LE + 1
        MX= DS-1
        MJ= MX-1
        FM= FLOAT(MX)
        MY= MX/2
        PI= 3.1415926
        RH= PI/6.
        VT= -PI/5.
        RC= COS(RH)
        RS= SIN(RH)
        VC= COS(VT)
        VS= SIN(VT)
        DO 120 N=1,LE
        L= 10000/INT(1.8**FLOAT(N))
        WRITE(5,70) N
70      FORMAT(' Working on Level',I3)
        CALL ABORT
        IB= MX/(2**N)
        SK= IB*2
C       WRITE(5,71)
C 71    FORMAT(/' XXXXXXXXX'/)
        CALL XHGHT
C                       assign heights along x in array
C       WRITE(5,72)
```

```
C 72    FORMAT(/' YYYYYYYYY'/)
        CALL YHGHT
C                       assign heights along y in array
C       WRITE(5,73)
C 73    FORMAT(/' DDDDDDDDD'/)
        CALL DHGHT
C                       assign heights along diag. in array
120     CONTINUE
C               DISPLAY Here
640     CALL SETUP
C                       set up plotting device or screen
        CALL SETXY(0,0)
        CALL LOCATE(0)
        ENCODE(MSG,333) LE
333     FORMAT('Levels =',I3)
        CALL GPRINT(11,MSG)
C    scaling factors
650     XS=.03
        YS=.03
        ZS=.03
C    Draw in y direction
660     DO 688 AX=0,MX
C   set first pix in line flag
        XO=-999.
        DO 686 AY=0,AX
670     CALL GETDAT
        ZZ=FLOAT(DD)
        YY= 10000.*FLOAT(AY)/FM
        XX= (10000.*FLOAT(AX)/FM) - YY/2.
C       WRITE(5,672) AX, AY, XX, YY, ZZ
C 672   FORMAT(' Plotting in y dir.   AX=',I4,
C     1  ' AY=',I4,'   x, y, z = ', 3(F10.2))
680     CALL PLOT
686     CONTINUE
        CALL ABORT
688     CONTINUE
690     DO 718 AY=0,MX
        XO=-999.
        DO 716 AX=AY,MX
700     CALL GETDAT
        ZZ=FLOAT(DD)
        YY= 10000.*FLOAT(AY)/FM
        XX= (10000.*FLOAT(AX)/FM) - YY/2.
C       WRITE(5,702) AX, AY, XX, YY, ZZ
C 702   FORMAT(' Plotting in x dir.   AX=',I4,
C     1  ' AY=',I4,'   x, y, z = ',3(F10.2))
710     CALL PLOT
716     CONTINUE
        CALL ABORT
718     CONTINUE
720     DO 748 EX=0,MX
        XO=-999.
        KK= MX-EX
        DO 746 EY=0, KK
730     AX=EX+EY
        AY=EY
        CALL GETDAT
```

```
        ZZ=FLOAT(DD)
        YY= 10000.*FLOAT(AY)/FM
740     XX= (10000.*FLOAT(AX)/FM) - YY/2.
C       WRITE(5,742) AX, AY, XX, YY, ZZ
C 742   FORMAT(' Plotting in diag. dir.   AX=',I4,
C     1  ' AY=',I4,'    x, y, z = ',3(F10.2))
        CALL PLOT
746     CONTINUE
        CALL ABORT
748     CONTINUE
        ENCODE(MSG,749)
749     FORMAT('  Done')
        CALL SETXY(0,15)
        CALL GPRINT(6,MSG)
750     CALL WAIT
C                       done Plotting, go to end loop
        CALL SCREEN(1)
        WRITE(5,752)
752     FORMAT(/)
        END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C       Heights along x
C
        SUBROUTINE XHGHT
        INCLUDE VARS/FOR
        DO 200 YE= 0, MJ, SK
        KK= IB+YE
        DO 190 XE= KK, MX, SK
        AX=XE-IB
        AY=YE
        CALL GETDAT
        D1=DD
        AX=XE+IB
        CALL GETDAT
        D2=DD
        DD= (D1+D2)/2 + INT((RAN(1.0) - 0.5)*FLOAT(L/2))
        AX=XE
        AY=YE
C       WRITE(5,100) AX, AY, DD
C 100   FORMAT(' XXXXX  ',3(I10))
        CALL PUTDAT
190     CONTINUE
        CALL ABORT
200     CONTINUE
        RETURN
        END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C       Heights along y
C
        SUBROUTINE YHGHT
        INCLUDE VARS/FOR
        DO 270 N= 0, MJ, SK
        XE= MX - N
        DO 260 YE= IB, XE, SK
        AX=XE
```

```
        AY=YE+IB
        CALL GETDAT
        D1=DD
        AY=YE-IB
        CALL GETDAT
        D2=DD
        DD= (D1+D2)/2 + INT((RAN(1.0) - 0.5)*FLOAT(L/2))
        AX=XE
        AY=YE
C       WRITE(5,100) AX, AY, DD
C 100   FORMAT(' YYYYY  ',3(I10))
        CALL PUTDAT
260     CONTINUE
        CALL ABORT
270     CONTINUE
        RETURN
        END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C       Heights along diagonal
C
        SUBROUTINE DHGHT
        INCLUDE VARS/FOR
        DO 350 XE= 0, MJ, SK
        JJ= MX-XE
        DO 340 YE= IB, JJ, SK
        AX=XE+YE-IB
        AY=YE-IB
        CALL GETDAT
        D1=DD
        AX=XE+YE+IB
        AY=YE+IB
        CALL GETDAT
        D2=DD
        AX=XE+YE
        AY=YE
        DD=(D1+D2)/2 + INT((RAN(1.0) - 0.5)*FLOAT(L/2))
C       WRITE(5,100) AX, AY, DD
C 100   FORMAT(' DDDDD  ',3(I10))
        CALL PUTDAT
340     CONTINUE
        CALL ABORT
350     CONTINUE
        RETURN
        END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C       Return Data from array
C
        SUBROUTINE GETDAT
        INCLUDE VARS/FOR
        IF (AY.GT.MY) GO TO 390
        BY=AY + 1
        BX=AX + 1
        GO TO 400
390     BY=MX+2-AY
        BX=MX+1-AX
```

```
400     DD=D(BX,BY)
        RETURN
        END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C       Put Data in array
C
        SUBROUTINE PUTDAT
        INCLUDE VARS/FOR
        IF (AY.GT.MY) GO TO 440
        BY=AY + 1
        BX=AX + 1
        GO TO 450
440     BY=MX+2-AY
        BX=MX+1-AX
450     D(BX,BY)= DD
        RETURN
        END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C       put in sea level here
C
        SUBROUTINE SEALEV
        INCLUDE VARS/FOR
        IF (OC.EQ.0) GO TO 625
        IF (XO+999.) 500, 480, 500
480     IF (ZZ.GT.0.) GO TO 490
C          turn on ocean
        F1= .TRUE.
        Z2=ZZ
        ZZ=0.
        GO TO 620
C          turn on land
490     F1= .FALSE.
        GO TO 610
500     IF ((Z2.GT.0.).AND.(ZZ.GT.0.)) GO TO 610
510     IF ((Z2.LT.0.).AND.(ZZ.LT.0.)) GO TO 615
520     W3=ZZ/(ZZ-Z2)
        X3=(X2-XX)*W3+XX
        Y3=(Y2-YY)*W3+YY
        Z3=0.
530     ZT=ZZ
        YT=YY
        XT=XX
540    IF (ZZ.GT.0.) GO TO 590
C               going into water
560     ZZ=Z3
        YY=Y3
        XX=X3
        CALL PPLOT
570     F1= .TRUE.
        ZZ=0.
        YY=YT
        XX=XT
        Z2=ZT
        GO TO 620
C               coming up out of water
```

```
590     ZZ=Z3
        YY=Y3
        XX=X3
        CALL PPLOT
600     F1= .FALSE.
        ZZ=ZT
        YY=YT
        XX=XT
610     Z2=ZZ
        GO TO 620
615     Z2=ZZ
        ZZ=0.
620     X2=XX
        Y2=YY
625     RETURN
        END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C               Rotate
C
        SUBROUTINE ROTATE
        INCLUDE VARS/FOR
        OX=XX
        XX= XX*RC - YY*RS
        YY= OX*RS + YY*RC
C       WRITE(5,742) AX, AY, XX, YY, ZZ
C 742   FORMAT(' ROTATE    AX=',I4,' AY=',I4,
C     1 ' x, y, z = ',3(F10.2))
        RETURN
        END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C               Tilt down
C
        SUBROUTINE TILT
        INCLUDE VARS/FOR
        OX= XX
        XX= VC*XX - VS*ZZ
        ZZ= VS*OX + VC*ZZ
C       WRITE(5,742) AX, AY, XX, YY, ZZ
C 742   FORMAT(' TILT      AX=',I4,' AY=',I4,
C     1 ' x, y, z = ',3(F10.2))
        RETURN
        END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C       PLOT to (xp,yp)
C
        SUBROUTINE PLOT
 940    CALL    SEALEV
        CALL    PPLOT
        RETURN
        END
        SUBROUTINE PPLOT
        INCLUDE VARS/FOR
950     XX=XX*XS
        YY=YY*YS
```

```
        ZZ=ZZ*ZS
960     CALL ROTATE
970     CALL TILT
990     XP= YY
C        + CX
        YP= ZZ
1030    XP= XP*1.91
        YP= 80.0 - YP
        IF (XO+999.) 1032, 1035, 1032
1032    IF (.NOT.F1) GO TO 1040
1035    CALL SETXY(INT(XP),INT(YP))
1040    CALL SETXY(INT(XP),INT(YP))
C       WRITE(5,1042) XP, YP
C 1042  FORMAT(' PLOT      XP=', F10.2,' YP=', F10.2)
        CALL LINE(1,-1)
        XO=XP
1050    RETURN
        END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C               set up screen
C
        SUBROUTINE SETUP
 1110   CALL SCREEN(0)
        CALL CLS
C       CALL SCREEN(1)
        RETURN
        END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
        SUBROUTINE WAIT
        DO 20 J=1,16,1
        DO 15 I=0,20000,1
15      CONTINUE
        CALL ABORT
20      CONTINUE
        RETURN
        END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
        SUBROUTINE ABORT
        LOGICAL BRK,BREAK
        CALL BREAK(BRK)
        IF (BRK) GO TO 801
        RETURN
801     CALL SCREEN(1)
        STOP **BRK
        END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
        INCLUDE RNDMIZ/FOR
```

Product Highlights: ADE

Kerry Wilson asked us, "What are the differences between ADE and diskDISK? I already have diskDISK and am familiar with it. ADE sounds like it is the same product." Dan Dible asked us, "What are the differences between LS-diskDISK and PRO-ADE? Are there any advantages of one over the other? How does their syntax differ?"

These are good questions. Ever since MISOSYS acquired the retail operation of Logical Systems, we wound up with two very functionally similar products: ADE and diskDISK. ADE is operationally similar. ADE takes up a little less memory space in both the Model 4 and Model I/III environments. Here's the specific sizes of memory usage for each utility. Model I/III (ADE 195 + 32 per slot, diskDISK 216 + 42 per slot). Model 4 (PRO-ADE 196 + 32 per slot, LS-diskDISK 216 + 42 per slot).

Depending on the size of disk you want to emulate and the granule size of your host drive, diskDISK may be able to create a larger pseudo drive. For the technical minded, the maximum sized floppy you can emulate depends on the granule size of your host drive. ADE uses a maximum of 4 directory extents for the emulation file; thus, 4 extents times 32 granules per extent times your granule size (usually varies between 2K and 32K) gives you the maximum sized floppy supported. On the other hand, a diskDISK can be created using up to 6 directory extents.

In my opinion, diskDISK is a little easier to install and remove (or change). Here's how you install the ADE drive emulator by responding to some very basic questions:

```
How many "slots" for this driver <1-8> ? 4
Slots already in use <.-.-.-.> Enter slot
number ? 1
What is the drive emulation filespec? >
floppy1:7
Enter the drive size <5,8> ? 5
Single or Double density <S,D> ? d
Enter the number of sides <1,2> ? 1
Enter the number of cylinders <35-96> ? 40
```

ADE then invokes FORMAT to create the directory information. Each "floppy" is created using the ADE/DCT driver and its brief series of questions. Once the driver is installed, a utility program called ADE/CMD is available to manage your "floppy" linkage. Existing "floppies" may be linked to the slot table or the existing table linkage may be altered to connect an emulated floppy to the DOS drive table. For example, invoking ADE/CMD displays the slot linkage:

```
Slot        Drive          Filespec
====        =====          ========
 1            4            FLOPPY1/ADE:7
 2                         not linked
 3                         ADE1/ADE:7
 4                         not linked

<M>ove, <L>ink, <A>ssign, <U>nlink,
<D>isable or <E>xit ? 1,4,ade2:7
```

The input response of "1,4,ade2:7" links slot 4 to the ADE2:7 file. Command entries and operands may be strung together as illustrated or entered individually via prompts. Commands may also be entered from DOS Ready.

DiskDISK creates the emulation drive by a separate utility called DDFORM/CMD. This "formatter" prompts you for the disk type, the number of sides, the density, and the number of cylinders via the following sequence of queries.

```
ddform testdisk:2

DDFORM - DiskDISK Creation Utility

Enter Disk Type {1,2,5,8 - default=5} :
Enter Sides     {Default=1 side} :
Enter Density   {Default=double} :
Enter Cylinders {Note : 1 Cyl= 4.50K} :
DiskDISK File Created
```

Once a diskDISK pseudo disk is generated, you enable it not by the SYSTEM (DRIVE=d,DRIVER=) command but by issuing a command of the form,

DD :d filespec[/DSK]

where filespec is a diskDISK pseudo disk created with the DDFORM utility.

DiskDISK becomes a little easier to use when you want to change from one diskDISK pseudo disk to another. You just invoke

the two commands,

```
DD :d (DISABLE)
DD :d filespec2
```

which disables the one currently assigned to drive "d" and enables the new one.

The comparison between LS-diskDISK and PRO-ADE is the same. ADE is a little smaller overall compared to diskDISK. On the other hand, diskDISK is a little easier to use when changing pseudo disks. You also may find that ADE prohibits you to create an 80 cylinder 2-sided double density pseudo disk when diskDISK allows it due to the 4 vs 6 extents limitation. Unfortunately, we have confused the issue by pricing both the same. That saves me from deciding which one to drop (assuming that we have to drop one of them). The marketplace will hopefully decide that for us; we may just keep both available.

Product Highlights: BSORT

RH claimed to be having a problem using BSORT to sort a 2-dimensional array. We investigated the problem he reported with BSORT and 2-dimensional arrays. Here's our test program using the data example shown on page 10 of the manual.

```
10 DATA dale,brown,25,boston,03021,ma,rep
20 DATA dan,jones,34,butte,78654,mt,rep
30 DATA don,smith,19,balt,23376,md,client
40 DATA DICK,GREEN,53,PHIL,19769,PA,ADV
50 DATA dock,peters,42,pitt,16511,pa,stock
60 DIM A$(7,5)
70 FOR J=1 TO 5:FOR I=1 TO 7:READ A$(I,J):
   PRINT A$(I,J);" ";:NEXT I:PRINT:NEXT J
80 SYSTEM"run bsort 5,a$(2,1)"
90 FOR J=1 TO 5:FOR I=1 TO 7:
   PRINT A$(I,J);" ";:NEXT I:PRINT:NEXT J
```

We experienced no problem with this program sorting as specified. Thus, if anyone appears to be having a problem, please supply us with the program and any data needed to display the problem. Also include a copy of your version of BSORT.

Here's a note from Paul Bradshaw 72177,2032 on Compuserve. LS-FED II is fantastic (as expected) [editor note: I couldn't help but leave this in even though it has nothing to do with BSORT]. "I have a patch in DLØ that patches your program BSORT/CMD to work with BASICG. I would like you to make that patch available to your MQ readers. It is part of the HGRFIX.LBR file here in DLØ. You have my full permission to make it available as you see fit." So readers, make note that if you use BASICG and are using BSORT, Paul has the above patch uploaded to our bulletin board. Due to a recent reorganization of the download libraries, it's probably now in DL6.

Product Highlights: EnhComp

JS had some input on EnhComp. Here's the Q&A dialog.

1. (Q) ON GOTO line numbers do not adjust properly when renumbering a program. (A) ON exp GOTO/GOSUB renumbering. The bug was traced to a deficiency in the routine which tokenizes BASIC lines. Seems that only the first line number of an address list was converted to a compressed line number. The 'N' command only adjusts compressed line numbers during a renumbering operation; thus, only the first line number of the address list was corrected. I have worked up a patch to revamp the tokenizer so that it compresses all the line numbers in an address list. After applying this patch (CED62/FIX in your case), you will have to re-edit any line which contained such an address list so that the list will get tokenized. Just edit the line then <ENTER> without making any changes will suffice.

2. (Q) REF/CMD does nothing. (A) REF generates the reference report as documented for me. Could you be so kind as to expand on your statement that it "does nothing"?

3. (Q) REM (') shows up as 'Error' if used at the end of a line of programming. (A) A "REM" is a statement which must be the first thing on a logical statement (i.e. at the beginning of a line or immediately following a colon). Interpretive BASIC allows you to eliminate the ':' statement separator; EnhComp does not.

4. (Q) I Need an editing command to find

the last line of a program. (A) Page 3-1, paragraph 2, notes that the letter 'B' when used as a line number target denotes the last line of the text buffer. Thus, 'PB' lists the last line and 'EDB' edits the last line.

5. (Q) Any way to get around the 'CALL' command to use subroutines? (A) Put your external CMD file subroutine into a Z80-MODE block then GOSUB it. See the example on page 6-1. There is no need to have external memory modules accessed via CALL since the routines can be assembled directly into your BASIC program.

6. (Q) Suggest the repeat key for Fstring/Sstring be mentioned in the "QUARTERLY". It took me a while to figure it out. (A) Yes. A NULL string will "Find" or "Search" the next occurrence of the previous find/search string. This was omitted from the manual. A note has been added to the README/TXT file.

7. (Q) Good program!!! (A) Thanks, we think so.

CAA has provided us with a number of bug reports concerning the earlier releases of EnhComp [we think we have a pretty stable product in release 2.5 -ed]. In one of his reports, we were not able to disclose any problem. Of course, the exact program he was having difficulty with was not supplied to us. That can usually make the difference between finding a bug and not finding a bug.

We tried the following program suspecting that maybe the LOC() function was amiss:

```
ALLOCATE 2
OPEN "r",1,"testloc/dat:2",75
FIELD 1,73 AS B$,2 AS AB$
FOR I=1 TO 500
   LSET AB$=MKI$(LOC(1)):PUT 1,I
NEXT I
FOR I=1 TO 500
   GET 1,I:PRINT CVI(AB$);
NEXT
CLOSE
```

This program produced correct numbers. Thus, I would have liked to try out his exact program and data to duplicate the problem and get at its cause. However, we find it impractical to type in all of the data and and all of the program from a paper listing. If you have the suspician of a bug, either give us a call to discuss it first or submit it on disk if it requires us to type in programs. If you have a two or three line program which can demonstrate the bug, then that can be done via a paper copy. Your disk, of course, will get returned at the conclusion of our evaluation if we have found a problem with our product or where you have provided postage for the return of your disk.

By the way, we have discovered that although you should be able to field an "R" file as "FIELD buf,256 AS A$", the runtime support traps 256 as an error. We will correct that in the next release. Save you a stamp in case you get to that bug before the next release is issued.

WDC advised us on Compuserve, "Roy, I have been playing with your Pro-EnhComp and have run into three problem areas. I have found how to circumvent two of them but can't seem to get the third to work. Since others may have the same problems, I will relate all three.

1. (Q) REM or <'> used in the same line as active statements will return a syntax error unless there is a <:> after the last active statement. (A) This is covered in the above material.

2. (Q) In the following example, each of the lines is supposed to line up vertically (in normal BASIC) but they don't when compiled. To make it work, the second line must be changed to 20 PRINT" "TAB(18);"Line ";I;" ";:..etc.

```
10 FOR I = 1 TO 5
20   PRINT TAB(18)"Line ";I;"
";:LINEINPUT A$
30   IF A$ = "" THEN GOTO XXX
40 NEXT I
```

(A) I duplicated the results WDC incurred. In checking into the code, I found no problem with the algorithm for the TAB(xx) routine. In the calculation to check if the tab position is greater than the current column position, the code deals

properly with any current column position. The code essentially complements the current column position stored in a variable named "@@SLNPOS" then adds the TAB value. If the result requires a carry (as it would if the tab value is more positive than the current value is negative), the result incremented by one becomes the number of spaces to output. For instance, if the current value is 2, and the tab value is 10, 2~ is 253. Add 10 to 253 and you get 7 with a carry (8-bit arithmetic). Increment 7 by one to 8 and 8 spaces are needed to tab over to column 10. Yet I still knew that my results duplicated WDC's - something of a puzzle. The only other problem could be with the value stored in @@SLNPOS. Unfortunately, this is a difficult thing to DEBUG since the cursor value changes with DEBUG's video display, also. The best thing to do was to put it to bed for awhile and rethink the problem. After much soul searching, it became obvious. The above program fragment followed the PRINT statement with a LINEINPUT. That uses the ROM or DOS supplied @KEYIN routine. The @KEYIN routine does its own display through the DOS video driver. The <ENTER> which terminates the lineinput does relocate the cursor to column zero; however, since this is done within the DOS' video routine and it's the EnhComp character output routine which maintains @@SLNPOS, the column position kept in BASIC was not updated to column zero. The first time through the loop, the routine tabbed to column 18. The second time through the loop, @@SLNPOS would hold the column position of the cursor after the last character was PRINTed. Thus, the second time through the loop, TAB(18) was less than the value stored in @@SLNPOS. The third time through the loop, although the actual physical cursor location was at column 0, @@SLNPOS was storing an eight. The TAB(18) would then generate only 10 spaces. You can calculate out the other starting cursor locations for subsequent lines. If the cursor position maintained by @@SLNPOS was correct, as it would be after the PRINT " " command executed, the TAB(xx) would achieve the desired result. The solution will be to add a little code to the LINEINPUT subroutine in the SUPPORT/DAT library to update @@SLNPOS after a call to the @KEYIN routine. Until that release, TAB(xx) cannot be the first thing in a PRINT statement or immediately following an INPUT.

3. (Q) I have been unable to make the USING function work at all. I get no errors during compile but when the program is run at the end of the compile phase, the program exits immediately to TRSDOS Ready. I am initiating the compiler with the S/CMD program. Sample lines of code used follow.

```
PRINT USING "#";QT
PRINT USING "###";45.486
```

I would appreciate your comments on the first two problems and hopefully, the solution to the third. I am satisfied with the program as a whole. The graphics commands add a bit of spice to some old programs. (A) Don, I believe that one or two PRO-EnhComps may have gotten out with an incomplete patch that caused USING not to work depending on what routines were loaded from the SUPPORT/DAT library. If you try the following patch:

PATCH SUPPORT/DAT (D02,B4=8A:F02,B4=7A)

and it is accepted, that will cure the problem. If you get a find error, then the patch is already installed. That problem crept in when we patched the library module to extend the USING string support to 79 characters from 63 characters. I would also avoid jumping out of FOR-NEXT loops as your program fragment illustrated.

BH gave me some feedback on PRO-EnhComp. Yes, we accept all criticism and sometimes print negative comments as well. Such types of comments generate valuable insight into user needs and desires - although user desires sometimes exceed the economic viability of providing a solution. At least the following dialog may clear up some confusion concerning ON ERROR GOTO, RESUME, and line numbers in general. Here's BH's thoughts.

Roy - Got some experience down, and have some stuff to report to you on EhnComp. CED doesn't understand LBASIC ".", ",", abbreviated commands, doesn't properly

handle basic line numbers when you are in line insert mode, doesn't save files in ASCII as an option (that I can see), has these stupid line numbers you can't see, doesn't handle mixed-mode programs where some lines have line numbers, and others don't,...argh! Also, it's handling of lower case is weird, in case the user uses labels (a label in quotes in lower case is in lower case; the GOTO label is in upper case, causing diagnostics during compile - why EnhComp wants upper case tokens is beyond me anyway).

S has some strangeness, too. If you fork off to compile a program, the compiler hangs up after the "first pass" prompt. Has to do with HIGH$, I noticed, after purging IOMON out of memory. But hanging up isn't cool; I suggest a diagnostic saying "not enough memory". Also, I see nothing is being done with the error diagnostics issued by the compiler. That should have been either fed to CED as automatic line numbers to go to for editing, or routed to a file for printing, or something!

No error code numbers. So, are EnhComp's error codes based on Model 3 BASIC, Model 4 BASIC, itself?? Need a list of these codes returned by ERR.

RESUME is a pip! RESUME lineno? I don't know the line number to return to! Since my program incremented by 10s, I tried a RESUME ERL+10. No go. Besides, if without line numbers, what does ERL have in it? In interpreted BASIC, ON ERROR GOTO is a GOSUB, with RESUME a RETURN. C'Mon! Help on this! As it stands, RESUME and ON ERROR GOTO are nearly worthless! More comments as my experience with this thing gets better. So far, RESUME has me stumped.

Some of BH's comments are well deserved. Others stem from misreading the user manual and design limitations. Here's my response. The README/TXT file on the EnhComp disk shows the syntax for saving a file in ASCII from CED. It is:

W:#filespec

Mixing a program with both numbered and unnumbered lines should work. now. Renum had problems with resequencing the lines wrong but that was fixed. If BC is invoked from DOS Ready instead of from the S supervisor, all diagnostics can be directed to the printer via the PRT option. The "S" facility is designed to be quick in going through the entire compilation-test-edit iteration; thus S provides no facility for directing the compiler's options. Page A-9 and A-10 list the error numbers. Sure, you could crash if you loaded the compiler when the compiler loaded above the current HIGH$. This is true for any large program. The DOS environment has no feasible way to overcome this. It's true that we could add a check in the supervisor program, S, to see if the current HIGH$ would allow sufficient space for loading BC/CMD; however, there is no way to deal with this if BC is invoked from DOS Ready. I'll put the check into S on the next release. RESUME is stated to behave exactly like GOTO. It is NOT like interpretive BASIC's RESUME. This is stated in the manual. What do you mean you don't know the line number of the line you wish to branch to on an ON ERROR GOTO? If you write the program, the line number is the BASIC line number assigned to the line when you code it (either by the CED "/" command or the ASCII editor of your choice). If you didn't number the line you wish to branch to, then it has NO number. Use a label instead. I didn't realize that interpretive BASIC'S ON ERROR GOTO is a GOSUB. It's RESUME does allow some leeway, but it is certainly not a GOSUB! The current implementation of EnhComp does not permit you to "resume" back into a subroutine if the error trap came from a command/function within a sub. It has to do with a stack cleanup operation. I expect to look into that at some future time to see if I can support RESUME NEXT, etc. The current ON ERROR GOTO is certainly not worthless.

Here's more from BH. Roy - It is so. Otherwise, how would the resume know where to resume to? Think about it. Error handling routines handle multiple lines where errors may occur. Let's say I have one for data entry. Any one of the (say,) six lines may have caused the error. How do I know where to resume to, unless I write separate error handlers FOR EACH LINE? Or have a variable that keeps track

of which line number I'm executing? This implementation of ON ERROR GOTO is tough to work with.

A little more from me should help clarify the use of EnhComp's error trapping statement. Why not use the ERL funtion which returns the line number of the line where the error ocurred. If you are going to have an active ON ERROR GOTO, then you will have to use line numbers! Also, you cannot use the "assumed" default line numbers which are only valid for diagnostic reporting but cannot be used as labeled targets. If you want to reference a line number within an EnhComp program, the line must be either explicitly numbered or labeled. As long as the error did not occur in a GOSUB'd routine, you can RESUME to wherever you want to by line number or label. For instance, RESUME 1968 or RESUME "OVERTHERE". EnhComp currently treats RESUME exactly like it treats GOTO. That may change in a future release.

Roy - Perhaps. Your "assumed default line number" has me confused, but perhaps that is what I am doing. Say I used SAID to whip up a program with an ON ERROR GOTO "ERROR":..."ERROR": Print "Erl = "; ERL... Interesting. If the line number HAS a number, that is the ERL. If the line number doesn't, then the ERL is the physical line number... So, what if I have a line 10 (label), which is the first line, and line 10 (physical), with an ERROR 5 on it. Wonder how my error handler would know WHICH 10 we're referring to? I'm not nitpicking; I'm going somewhere with this... (It involves that address of line number command in EhnComp, which at the moment, it looks like ERL should be rigged to return.

Look, ERL should return a line number since that is what it logically references in the programmer's mind. Now since EnhComp supports the use of labeled lines (and a label resolves to an address) and does not require line numbers, perhaps I need to add another function, say ERA, to return the address of the line which contained the error. But so what? How are you going to use that? The only thing in EnhComp which currently lets you get at the address of a line is the (#line) operator in the Z80-MODE (this is, of course, not yet documented, either). Under your case, if an error occurred on an unnumbered line, ERL would report the auto-generated line number. You can't use that as a GOTO target because the targets of GOTO, GOSUB must be actual line numbers resolved at compile time.

Moving on to another EnhComp topic, HB asked, "Roy, I'm more than a little confused by the EnhComp use restrictions. I understand the difference between application and utility programs in the more extreme cases, but it seems to me that the two categories overlap in a large grey area. For example, would a program that created a /JCL or TYPEIN file be a utility? What if the resulting file were meant to run a specific application program, also written in EnhComp? Would a program that generated reports from a MultiPlan or Visicalc spreadsheet be an application or utility? How about one that simplified the creation of a spreadsheet? I'm not sure how you or Philip Oliver plan to interpret the use restrictions, but it seems to me that it is opening a real can of worms. Where, for example, does one draw the line between a generalized database program like Little Brother or Profiles (I assume those are both "applications") and a specialized programming language like dBase? Any comments?

Yes, I was able to comment on this. I believe the distinction is whether or not the compiled program is to be used to generate something else with the original library module(s) used as module(s) for the downstream process. For example, if you used EnhComp to write another compiler, you would not be permitted to re-use the EnhComp library code as a library in the generated compiler product. That's where I draw the line. It's the same situation as using MC.

MH had a problem with a bulletin board program he was working on. Here's the text and program fragment which was causing the difficulty. Yes the program fails when compiled directly. BC/CMD gives error code 133 undefined label or symbol. For some reason the compiler isn't recognizing the

RETURN statement in line 6040. If this statement is replaced with a GOTO 60, the program will RUN. If you don't have the original listing here it is.

```
10 CLEAR 5000:DEFSTR C:DIM C(1):
   ALLOCATE 1:SW=64
50 OPEN "I",1,"MAGC1/TXT"
60 IF EOF(1) THEN CLOSE:PRINT:
   PRINT "THE END":END
70 LINEINPUT#1,C:GOSUB 6000
80 GOTO 60
6000 ' TABLE OF CONTENTS WORDWRAP ROUTINE
6010 IF LEN(C)<=SW THEN PRINT
C:C="":RETURN
6020 FOR Y=SW TO 1 STEP -1:IF
MID$(C,Y,1)=" "
     THEN C(1)=" "+MID$(C,Y+1):PRINT
     LEFT$(C,Y-1) ELSE NEXT Y
6030 IF LEN(C(1))>SW THEN C=C(1):
     GOTO 6020 ELSE PRINT C(1)
6040 RETURN
```

I admit the program isn't elegant (RETURNing to a GOTO) but the second RETURN statement should be recognized.

Here's the solution. Somehow it just seems that the 6000 subroutine has a FOR NEXT loop which is not too cool for EnhComp. First, it appears that if the 1st char is a blank, the FOR loop does not terminate by exhausting the index. This has a tendency to leave a FOR index counter on the stack. I suggest you look at rewriting that loop so it terminates "properly". M/S interpretive BASIC is pretty lax at requiring the proper termination of FOR-NEXT loops. For example, a properly terminated loop is:

```
FOR I = 1 TO 100:
   IF I = 10 THEN I = 100:
   GOTO 50 ELSE WHATEVER:
50 NEXT I
```

MH took my suggestion and responded, OK Roy, You were right on. The FOR NEXT loop was the problem. I'll just have to remember to terminate properly. EnhComp otherwise has been very easy to use. The Supervisor command (S/CMD) makes it very quick to write, compile, and test a program or routine.

Finally, my last note to MH: I thought that it might have been that. EnhComp is very persnickety about proper completion of FOR-NEXT loops - meaning don't jump out of the scope of the loop.

MH had another problem with EnhComp. He passed this message along to us via Compuserve. Is it possible that the copy of EnhComp I received doesn't have the -NX directive operational. The README/TXT file that came with the package states -NX was operational with version 2.4. I have tried unsuccessfully to use this -NX directive. The command tried was BC filespec$,,,-NX-WD. This format caused the generation of undefined label or symbol during pass #2. Just for fun I used *-NX and the program compiled without errors but the BREAK key is still operational. Right now I'm ignoring this option when compiling and everything else seems fine. It would be nice to know why the -NX directive generates this error message.

Well when I got around to tsting this report, I at first didn't come up with any problem in compiling. I was just about ready to ask for a copy of his compiler so I could see if it was damaged. I then tried another little four liner and discovered that there was indeed a problem - but only when keyboard input was requested via the INPUT or LINEINPUT routines. This is another case where the exact program source one is having trouble with is necessary for analysis.

Here's also some more insight into the information which EnhComp displays during compilation. Any having any occurrence where "Undefined label or symbol" errors are output, should be able to provide a little more feedback concerning the error. Here's the BC output of my test four liner. By the way, I captured this output by routing the *PR device to a disk file then compiled with the -PRT switch.

```
Pass #1
Appending SUPPORT subs
(0) (1) (2) (3) (4) (10) (11) (13) (14)
(16) (17) (19) (20) (21) (22) (24) (27)
(28) (29) (30) (33) (34) (40) (43) (44)
(45) (46) (47) (71) (88) (89) (95) (97)
(144) (145) (158) (173) (178) (179) (193)
(227) (229)
```

```
Pass #2
1 FOR X = 1 TO 100
2 PRINT X,
3 NEXT
4 LINEINPUT A$
Appending SUPPORT subs
(0) (1) (2) (3) (4) (10) (11) (13) (14)
(16) (17) (19) (20) (21) (22) (24) (27)
(28) (29) (30) (33) (34) (40) (43)
@DOBREAK --> Error, Code 133 : Undefined
label or symbol (44) @DOBREAK --> Error,
Code 133 : Undefined label or symbol (45)
(46) (47) (71) (88) (89) (95) (97) (144)
(145) (158) (173) (178) (179) (193) (227)
(229)
-- Compilation complete
Loading area (HEX): 5200 - 782E /  Length
(DECIMAL):   9775
Data Table lengths:
5,2185,0,2,4,2,2,0,1,0,1,0,0,50,1631,1056,0
,0
(Total used: 6280 bytes out of 14271
available bytes)

Total Compilation Errors: 2
```

In this information, the numbers within parentheses indicate the subroutines being loaded from the SUPPORT/DAT library. Pages 5-3 through 5-11 describe what each SSR does. Note that the Error 133 occurs after "(43)". This means that the symbol or label was referenced in SSR 43 but never defined. It was also referenced again in SSR 44. The label or symbol name was @DOBREAK. With this information, I was able to look at my source code for the library at subroutine 43 and 44. A little soul searching revealed that the @DOBREAK address was the target of a jump if the INPUT routine detected a BREAK and the break key was active. A little checking revealed that @DOBREAK is in SSR 191. That subroutine certainly does perform the scanning of the BREAK key between BASIC statements; however, it is also needed to service a BREAK sensed after a keyboard input. The -NX compiler option keeps SSR 191 from being loaded.

The resolution is a little patch to BC to request SSR 191 when either INPUT or LINEINPUT is used. This you will find in the PATCH Corner. Anyone want to know what the table lengths indicate?

Product Highlights: HELP

PG on Compuserve reported, "I have found that the TRSDOS 6.2 HELP program, on exiting, gets confused (frequently) as to whether the screen was last in inverse video or not. I'm guessing, but it seems if there were ANY inverse video characters on the screen before HELP was invoked, it returns you to DOS with the inverse on. I'm sure the power users who design and maintain TRSDOS 6 don't ever need HELP, but I sure do".

To PG: It's not a question of confusion. A HOME video code will reset inverse video. TRSODS 6.x provides no way to recover the video state. Thus, device filters which may change the state of inverse video because they @DSP a HOME code, cannot restore the video state which existed prior to their filtering. Now don't get me wrong, I'm not saying it is impossible to detect the video state, but it requires knowledge of a few undocumented addresses of locations within the video driver which are not necessarily constant across every release of DOS; thus, there is no documented or standard way to extract the information.

Product Highlights: LED

BH had some questions he addressed over on our Compuserve SIG. I think others may find the dialog useful.

I use LS-LED to edit CIS message downloads for storage. Sometimes those files contain null bytes (00H) due to another program I use to split up oversized ASCII files. The docs say LS-LED docs should load the file, while ignoring the nulls (which would be a useful feature), but my copy hangs. I use "LED filename (x=x'0001')" and delete out the 01H bytes, but that's kludgy. I'd like to get it working right. What's the story?

Les Mikesell came upon the solution to BH's problem. Re: LS-LED appearing to hang after reading a file containing 00 bytes. You just have to be patient. LED will read the entire file into memory, then go through and delete the nulls by moving all the text as each one is encountered. With a large file this might take a couple of

minutes. You might try using the (t) parameter to see if that helps. Specifying tab expansion will cause the file to be read a byte at a time and the nulls should be discarded before loading into the buffer.

Thanks for the tabs option suggestion, it helps. Had no idea LS-LED did byte-by-byte null purge AFTER the file load. I figured that happened during file I/O, something like the old LDOS TRAP/FLT program. What I really should do is fix my utility to stop at true EOF and not end of sector.

Don't suppose you'd know what byte to patch to make that tabs parm default to 3 spaces instead of 8? I want to start working in C (have Roy's MC) but hate to give up LED for SAID. It's like giving up my favorite pair of slippers. (Yeah, I know, I won't forget the ^Z at the end).)

Product Highlights: MC

PHL had a question about updating an older version of LC to the "latest" release, 1.2b. LC was our original C compiler. We are no longer offering updates of older LC versions as LC is now a discontinued product. It has been replaced by MC, a full C compiler. An announcement of the LC to MC tradein offer was mailed to all registered MISOSYS customers in our November 1985 flyer. We will continue to accept tradeins of LCs for MCs for the remainder of this year, 1986. Beginning January 1, 1987, we will not accept any tradein of LC towards the purchase of MC. The offer extended a credit of $89.90 towards the purchase of an MC/MRAS combination or a PRO-MC/PRO-MRAS combination.

LR was having a problem using MC's scanf() function to input a single character. He said, "%1f%c%1f doesn't work as described in K & R! I tried using %1f %c %1f, and it doesn't function properly either. These work in ECO C too. I believe they worked in LC even. Here's my test program:

```
/*  scantest    */
#include stdio.h
#include math.h
main()
{  double val1, val2;
   char oper;
   int x;
   option(0_KBECHO,TRUE);
   printf("Value1 operator value2\n");
   x =
scanf("%1f%c%1f",&val1,&oper,&val2);
   printf("\nscan = %d  val1 = %1f  op =
%x  val2 = %1f\n",x,val1,oper,val2);
   switch (oper)
   {
      case '+':
         printf("%.2f\n",val1 + val2);
         break;
      case '-':
         printf("%.2f\n",val1 - val2);
         break;
      case '*':
      case 'x':
      case 'X':
         printf("%.2f\n",val1 * val2);
         break;
      case '/':
         if(val2 == 0)
            printf("Division by zero.\n");
         else
            printf("%.2f\n",val1 / val2);
         break;
      default:
         printf("Unknown operator.\n");
         break;
   }
}
```

K&R is certainly terse when discussing scanf(), although the function is covered on pages 147 through 150. The **UNIX System V Interface Definition** manual expands a bit on the use of %c. I responded to LR with the following: I am enclosing two pages from the UNIX System V Definition manual. They are associated with scanf(). Note the highlighted portions. Specifically, on page 229, it states "trailing white space is left unread unless matched in the control string". On page 227, the statement concerning %c translation is, "... The normal skip over white space is suppressed in this case; to read the next non-space character, use %1s."

Now then, our documentation states that we adhere to the System V spec. I tried your example but used %1s in lieu of %c; mine worked as advertised. Ecosoft's compiler cannot adhere to the System V definition for scanf() if it did not suppress the

skip over white space for the %c translation. They may not even support scanset. On the other hand, my implementation of %e and %f requires white space for a separator. However, the %ls as suggested in the System V manual is the best solution; it should work for everyone.

LR also complained about MC's use of integers to store bit fields in a structure. He wanted to break apart a 3-byte subfield of a 32-byte record (actually the first three bytes of a BRINGUP/DAT record from our PRO-WAM application). This is covered in K&R on page 196. K&R state that field members are packed into machine integers. I believe that the Z80 chip is classified as having a 16-bit machine integer. Someone with this problem may be able to overcome the difficulty by using a union. Have the first structure declare the 24 bits with the remaining 28 characters declared in char array. Then union it with another structure which has the first three bytes declared as char array with the remaining 29 another char array. Access the bit fields with the former structure and the activity description with the latter.

BAQ asked about a method for a C program to access the graphics library provided with the Radio Shack hires graphics board. Here's my response.

"I wish I could help you with your access of the graphics library provided for FORTRAN. Let me give you a little insight.

The interface to subroutines or functions is different for C and FORTRAN. Microsoft documents their interface as being in registers. They talk about the first arg in HL, the second in DE, and 3 or more in a memory list pointed to by BC. Since these registers are only 16-bit wide, I would guess that FORTRAN is actually passing pointers to the variable storage - it could be a 32-bit floating point number, you see. C, on the other hand, passes its arguments on the stack. This is documented in our manual on page 5-26.

If you want to access the graphics libraries, you will first need to know the exact subroutine linkage as far as arguments are concerned. Then you will have to write some form of argument interface conversion routine to connect what the C language would deal with vs what the FORTRAN subroutine expects.

That's the direction you need. Good luck. If anyone else poses a solution to me, I'll probably bring up the solution in a future issue of THE MISOSYS QUARTERLY. Note the possibility of using our recent announced utility, UNREL, to uncover the exact nature of the GRLIB routine linkage.

FB presented another MC problem to us. "I found an interesting problem the other day. I've been writing an ECI. But whenever the program sent a blank command line, the system would crash. After much twiddling and nashing of teeth, I found that if I turned off the ARGS and REDIRECT options, the program worked OK. I don't need these options, so it is just as well that I turned them off. Any ideas? Here's my test program using TRSDOS 6.2.1, PRO-MC v1.5a."

```
#option   ARGS   0
#option   REDIRECT   0
char   cmdline[4]={' ',' ','\n','\0'};
main()
{
   cmdi(cmdline);
}
```

Well I tried that program both ways, deleting the two #option staements and also leaving them in. Anyone else having such a problem?

Here's some input from SD. He said, "In the latter part of May 86 I bought PRO-SAID & PRO-MC. After using PRO-SAID I discovered a bug, but it turned out to be a bad disk on my part. While I was talking to you about SAID, you said something about shipping a new [version] of PRO-MC. (This suddenly hit me last night ... I know; it took long enough.) I thought I read somewhere that the PRO-MC I bought was not totally completed, i.e., some functions were still in the designing phase. Is this true? If this is true, can I obtain the "new" PRO-MC? What would the cost be to get it? Also, I have looked all

over the manual (or so I think I have) and cannot see anywhere where it talks about using a structure type to make variables be like bit flags. I have been using "C Primer Plus" by Sams and they talk about fields. They say to use a structure definition to label fields and identify their widths in bits and NOT bytes. Thus one variable could be defined as being one bit within an unsigned integer. These would be nifty to use for software flags, but I cannot find anything in the manual which talks about using bit fields with PRO-MC.

My response to SD may be needed for someone else. It goes, "If you were shipped MC 1.4, then your package should have included a letter requesting you to return the master disks for a free update. You should have also received a prepaid disk mailer. The final release (1.5) has some info in the README file on bit fields. There is also an example program, DCT/CCC, included on the MCLIB disk which "documents" their use. The demo /CCC file displays the status of each Drive Control Table. Nothing was in the manual on bit fields because it was not certain if they were going to be implemented. K&R is very terse in the bit field department. Other books may be more enlightening. Our example may prove the most useful for you.

JB also reported a gotcha Here's his dilemma. "I've got a program that I've written with PRO-MC, that I would like some things to go to the printer, regardless of I/O redirection. Is there a way to use putchar, or ANYTHING(?) to direct output to the printer? The only way that I've thought of, is to have all of the text for the screen sent to the stderr channel, and all of the "normal" output sent to the *PR device instead of the screen. It's a kludge. Is there a easier way? That's problem one. Here's problem 2; In my program, I want to be able to en/dis/able the printer spooler. I issue the command system("spool (bank=2,disk=0)"); to enable it. Then, when I try and disable it, I'm given a message "Can't find SPOOL", or something of that nature. I've tried disabling it from TRSDOS ready, but still can't. I've enabled it from TRSDOS ready, and my program wasn't able to disable it either. Any ideas?"

Part of my answer wasn't too useful. Here goes. You cannot use the system() function to install a driver into high memory. That must be done external to the C environment. That's because HIGH$ is altered during the function's execution and changed afterwards. Then again, maybe no part of the spooler is put in high memory in your case if enough low memory space is available. The problem with finding a high memory module via the system() function is caused by the change of HIGH$ during the execution of the system() function. The C program and data is moved to butt up to the current HIGH$ and then HIGH$ is lowered to protect the "saved" C program. Unfortunately, we neglected to insert a memory module "header" below that. Thus, the @GTMOD SVC cannot find anything above the C program. We'll have to redo the system() module for that.

The easiest way to direct standard output to the printer if you need no video output would be to use freopen() with arguments: freopen("*pr","w",stdout);. That will essentially perform internal redirection regardless of what preceded it on the command line. The freopen() function is pretty powerful for such things. Take a look at the simple unarc.ccc program in the manual. Freopen() has some good uses. If you still need video output, then you can either open some other stream to "*do" or freopen() stderr (in case it was redirected! You could, of course, disable redirection by use of the #option REDIRECT OFF

Here's a query from Hardin Brothers. Roy, I've been running some benchmark programs with MC, and I came across one that looks like it should compile, but won't. According to the MC docs, there can be up to 255 levels of indirection. However, the compiler chokes on the following declaration:

```
struct cptrl {
    char ****************ptrl;
    };
```

Now, admittedly this is an unlikely declaration for a "real" program. But

unless I missed something in the docs (a real possibility), I don't see why it won't compile. Any comments or suggestions?

Yes, a later change in the compiler's inards in order to accomodate structures/unions and multi-dim arrays restricted a variable to at most 16-levels - if memory serves me. Had to do with the number of bits available in one of the tables. The docs never got changed (since the docs were printed back last year and the change occurred this year). As you note, I don't think that this limitation is significant.

SD came upon another quirk. This time in connection with escape constants. Roy, This is rather late, but, thanks for the prompt upgrade to PRO-MC 1.5a. Now for a question on PRO-MC; does PRO-MC want to see delimiters behind hex escape control sequences? I have found that PRO-MC doesn't like to see text directly behind a control sequence. I don't know if it does the same in octal or not, I'm not very good in the octal base. Anyhow, a good example of line interpreted wrong by PRO-MC is as follows:

```
printf("\n\n\x10Aborting ...\n\n");
```

When this is compiled, PRO-MC should pick up the hex byte as x'10', instead, PRO-MC picks up the hex WORD of x'10A'. When the program is executed, the message "borting ..." will be displayed. Note, no errors result from this problem. I thought there might be a delimiter, perhaps another "\", so I looked in the manual, but I couldn't find anything on a delimiter. Is this a problem that has already been corrected or am I mis-reading a crutial piece of information in the manaul? BTW, I never reported it before but this problem appeared in the "older" version of PRO-MC too. Thanks for your help.

Here's what I found. The hexadecimal escape should pick up only two characters after the "\x". It appears that MC is not stopping the hex escape until a non-hex digit is encountered. The logical extension from octal escape in K&R would be to accept at most two hex digits. Rich tells me that the escape conversion is done in MCP. I'm going to take a peek at that to see what is exactly the case. After that peek, I found the ESCAPE function in MCP needing a touch up on hex escape; it was okay on octal limiting the octal escape sequence to at most 3 octal digits. The hex coding did not limit it to 2 hex characters but terminated on the first non-hex digit encountered. There's a patch in The PATCH Corner which corrects for this (MCP51 and MCP61).

SD asks, "Roy, Is there *ANY* way to make MC give the address of a function? I want to use the task processor on the Model 4. I need to load the task slot with the address of the task code (a C function). I was looking long and hard at the longjmp() and setjmp() functions, but it doesn't seem these are made for the problem I'm trying to overcome."

I believe that using the name of a function represents the address. On the other hand, what you may be really interested in is "pointer to function". This is discussed in K&R on pages 114-115. Incidentally, NOTES FROM MISOSYS, Issue IV had an article on pages 4-52 through 4-55 which supplied a routine that interfaced to the DOS interrupt task routine. Due to the similarity between LC's and MC's argument passing, that routine should be easily adaptable to MC.

DN had another problem posed to us. "Roy, I am having a small problem in MC with the suppression character "*". In the example given under the scanf() function, it shows the suppression character being used successfully on the conversion character "d". My problem is using * with the conversion character "c". Shouldn't it work the same? Why won't the following work?"

```
#include <stdio.h>
#include <math.h>
main()
{  char string[81]; float fl;
   int n_items, aint, isnt;
   n_items = scanf("%*c%3d%2d%6s%f",
   &aint,&isnt,string,&fl);
   printf("Count = %d |%d %d %s %e\n",
```

```
   n_items,aint,isnt,string,(double)f1);
}
```

The %c is a very funny animal. Offhand, I would have to look at the code to see whether or not it would "work" and what the effect would be. One more for the hook. (time out for looking) The decision to either skip or not skip over whitespace based on the %c or %[translation is made before input of the "field" is either accepted or suppressed based on the %* specifier. My hunch is that the "*" suppressor is just not going to function with %c. Many folks actually get confused over the difference between "%c" and "%1s". There is a difference! Check the System V specs. It may be possible to use "%*1s" instead of "%*c".

DD let us know on Compuserve, that the WILDCARD option was not working for him. He was using MC with Microsoft's M80 assembler. He reports, "Has anyone had problems using the wildcard option with PRO-MC and M80? I am using the sample program given in the docs:

```
#option WILDCARD
#include <stdio.h>
#include m80
int i;
main(argc,argv)
int argc; char *argv[];
{
  printf("argc:%d, argv:%#04x\n\n",
     argc,argv);
  for (i=0; i<argc; ++i, ++argv)
     printf("argv[%4d]: %#04x is '%s'\n",
        i,*argv,*argv);
}
```

My JCL to compile is, "DO MC80 (n=wild,k,d=3)". Results with the compiled program are:

```
wild wi*/cmd w???/cmd

argc:3, argv:0x483c
argv[   0]: 0x481d is 'wild'
argv[   1]: 0x4824 is 'wi*/cmd'
argv[   2]: 0x482d is 'w???/cmd'
```

Any help would be appreciated.

Well after verifying and investigating DD's report, I found that the bug was a result of a missing instruction in the M80/H header file. This file is equivalent to MC/ASM for the MRAS folks. The missing instruction was

```
      LD     HL,@_WILDCARD
```

The instruction is shown below added into the correct position within the M80/H file.

```
      $SR    HL,IX,BC
      IF     @_WILDCARD
      LD     HL,@_WILDCARD
      JP     $WILDCARD##
      ENDIF
      ENDIF
      IF     @_FIXBUFS
$$FREE:
```

VK brought to our attention a problem with typedef. He wrote, "I believe I have encountered two bugs in MC for the TRS-80 Model III. One set of problem code is as follows:

```
typedef int *TREE; TREE *root;
```

At this point I receive a "declaration too complex" error message from pass one of the compiler. According to the error message description in the appendix, 16 levels of "pointer to", "array of", or "function returning" are allowed. A second problem involves a definition in the stdio.h header file. Consider the following code:

```
#define FILE char* FILE *fp1, *fp2;
char **fp1,*fp2;
```

The first line of code is from MC's stdio.h file. The second line of code is a simple declaration. The third line of code is the result of passing the first two lines through the preprocessor. The problem is that the value of fp1 is a pointer to a character pointer while the value of fp2 is a character pointer. The intent of K&R seems to be that the types here be identical. FILE should be defined as "typedef char *FILE" but then trying to use a FILE in a declaration would result

in the same error message that TREE received above!

My initial work-around to the first example above involved code similar to that presented in the second example. In my application I wish to use TREE as a dummy type similar to FILE, a type whose internal structure is unknown. When I got the error message mentioned above, I decided to see how MC handled the FILE type. I don't think that it does so properly: using the preprocessor seems to generate problems for declarations. One could not expect correct results from expressions involving pointer arithmetic when the pointers involve objects of different sizes (in the example, a pointer in one case and a character in the other).

I hope this information will enable you to resolve this problem. I have had to carefully rewrite my declarations to ensure that types are identical where they should be. I would prefer to be able to directly port code from other systems."

We think so, too. Our initial testing with typedef did not disclose the error which VK had exposed. Nor did we ever make use of a FILE pointer which involved pointer arithmetic; thus masking the error with the #define. I passed this problem on to Rich (author of MC). It took awhile, what with busy schedules and the difficulty in patching the compiler. Rich did find the bug and provided me with patches for all MC passes for both the MC and PRO-MC releases. These are found in The PATCH Corner as MCTDxx series. Also, after applying the patches to your MC, please change the '#define FILE char*' in the stdio/h header file to 'typedef char *FILE;'. Note specifically that this statement NEEDS a semicolon.

Product Highlights: MRAS

TW had a number of questions and comments concerning our MRAS package. Here's the dialog which we felt useful to share.

0. (C) First a comment. The only other assembler I've ever used before was the old Radio Shack assembler on the Model I. MRAS is a far cry from that old assembler. After using MRAS, I would never dream of using that old assembler, even though it sits on the next desk, ready to go.

1. (Q) I have only one page, 5-8, which covers the installation of SAID. I feel that the next two pages must have been left out of my manual, or were they left out of all the manuals? (A) There is only one page (5-8) which covers the use of SAIDINS. Since the installation is dialogue driven, we didn't think more material was necessary. If you have a particular question concerning the use of SAIDINS, please address it to us.

2. (Q) One 'bug' nearly drove me up a tree. Normal TRSDOS filespecs support 8 characters, while MRAS only supports 7 in the NAME pseudo-op. This was a tough one to track down. (A) The NAME pseudo-OP is documented on page 2-31 as, "the first seven non-blank characters". Seven is not our doing, this is Microsoft's format for all global symbols. You see, there is a 3-bit field for the length specifier. Microsoft doesn't encode it as excess 1; thus module names as well as globals are limited to seven characters maximum.

3. (Q) The documentation leads me to believe that SAID automatically terminates each file with a 1AH byte. My copy doesn't. On the other hand, I have found SAID to be a very useful editor. I have used it often for direct modification of disk files. Lately I have been using it to write HELP files. A very versatile editor. (A) SAID will automatically terminate the text file with a 1AH if the CCC or ASM parameters are specified (indicating language text). If you enter SAID without a language specifier, not only is the 1A not added to the output file, but neither are tab stops set to every 4 (or 8) columns. Furthermore, not specifying the ASM parameter for assembler files will result in the output file not being upper-case converted as required by the assembler.

4. (Q) I can't get FIXUP to work properly. (A) I cannot duplicate the problem you are having with sample/asm. I fixed source files both larger and smaller than your sample. I tried one with a blank line as the ending line (that's what yours was)

and still no problem. I used a copy of FIXUP which came off the distribution disk. Thus, I have no explanation to offer you. If this happens on all of your files, perhaps you ought to send me a copy of your MRAS disk along with some of the sample files you are trying to fix.

5. (Q) One thing that MRAS needs is a card listing all the switches for MRAS and MLINK. SAID needs one or two minor improvements. One is to replace the <CLEAR> key with the <CTRL> key. [the other related to a method for indicating column 59 so as to avoid entering assembler text which would wrap on listings]. (A) I agree that a quick reference card for MRAS would be nice. We may do such a QRC when we are sure that the package would not be enhanced further. Don't forget that SAIDINS allows you to alter every command function in SAID to be invoked on the key of your choice. SAID thus does NOT limit you to use the CLEAR key; that's only the default layout. If you want to use CTRL, you'll find that the selection of keys is too limited as the CTRL key is only valid with A-Z keys whereas CLEAR is usable with all but the function keys. A possible solution to the line bend around on printing would be for character counting during assembler output. We have always considered that in the past; however, we have not felt it sufficiently important for inclusion. Perhaps in the future...

LR was another individual reporting a problem with phase errors using MRAS. Here's a little discussion of importance to all MRAS users.

Here's some feedback concerning the reports which you had forwarded to me. Let's start with the MRAS problem. As discussed on the phone, I suspected a phase error during the assembly. The culprit was the statement:

```
LD      BC,ERRTBL-LCSDVS
```

In this statement, the first pass finds ERRTBL undefined so it is considered to be ABSOLUTE; LCSDVS has been defined and is code relative. Subtraction is invalid for absolute-relative; thus, the assembler flagged it as a relocation reference error. These types of errors are treated as fatal errors; the instruction did not get assembled. On pass 2, both symbols were found to be code relative which is valid for subtraction. Thus, no error was generated. The difference between passes was the three-byte instruction not assembled on pass 1. Thus, the phase errors.

Although the QUARTERLY (Summer 1986 issue) addresses this topic on page 21 and covers the side effects if the error was treated as a warning, it did not present a simple patch which could accomplish that. I have not fully explored the ramifications of converting the error to a warning; on the other hand, if the error persisted by pass 2, the assembly would still be flagged as having an error(s). Thus, here is a patch to convert the error to a warning.

```
PRO-MRAS: PATCH MRAS (DØB,37=Ø6:FØB,37=Ø1)
MRAS:     PATCH MRAS (DØB,79=ØF)
```

LP reported a problem with MRAS. He said, "I have two files which are quite similar, yet one produces an error and the other doesn't." Of course, LP neglected to send me the two files! That makes it tough to debug. After I responded to his report with a request for the files, LP shared them with me. This provided me with a concrete example which would fail 100% of the time. That kind of test is very useful in tracking down the reason for a problem. Here's what I found.

What is usually the case with bugs is that if they are repeatable, the fix is easy. Of course, when we get the file which demonstrates the bug, it usually can be isolated. Such was the case with your reported bug using MRAS after getting your disk.

The bug was traced to the NAME statement. The operand should have been restricted to at most 7 characters since that is what is supported (or shall I say required) by Microsoft's link format. Unfortunately, MRAS was not performing a check in the routine where it was passing the name field to storage for use as the module header name. The argument in your

ALPHANUM/ASM file was 8 characters long. That 8th character clobbered a byte necessary for use with the file include nesting.

I worked up a patch and applied MRS58/FIX to the MRAS copy on your disk. I also applied MRS57/FIX to fix a small problem with page eject when assembling with -LP while the file generated assembly errors.

CS had some problems using the FIXUP program provided with MRAS and PRO-MRAS. After looking over the documentation, I believe that it was a little confusing. Here's a little text expansion which may clear up the syntax for invoking FIXUP.

The manual states the syntax as:

"FIXUP filespec {(*/strip/header/number}"

The proper syntax for FIXUP is one of the following:

```
FIXUP filespec (*
    to rewrite the in-memory file,
FIXUP filespec (STRIP
    to strip numbers and headers,
FIXUP filespec (NUMBER
    to add line numbers,
FIXUP filespec (HEADER
    to add headers and numbers.
```

You choose one of the parameters; the slashes are only used in the manual to show that you have a choice of any one listed. That should "fix up" your problem with FIXUP.

GP posed some observations on our Compuserve SIG which resulted in quite a bit of dialog. The subject is a good one. Here goes:

(GP) I have been using and enjoying MRAS ever since receving the upgrade in my mailbox. However, I have one suggestion and one complaint.

When you write the assembly listing to a file, is it not with the intention of printing it later? That would be my assumption, and yet the file created by the +L= option is not formatted with headings and page numbers and form feeds. Since I can get the desired result by ROUTE *PR filespec instead, this isn't a complaint but rather a bemused suggestion.

Now the complaint - maybe a bug though I can't make it happen consistently yet. Sometimes, on assembling a file that has already run through once without errors, the MRAS goes crazy and reports nearly every symbol in the source file as being multiply defined! So far this has only happened when using the above ROUTE *PR approach or the +L= option, so I have only detected the problem later when I went to print the file. Consequently, I can't yet reconstruct the exact sequence of events that produced the error. However, when I reassemble the file yet another time, it always comes up clean; at least the source file doesn't seem to be the cause.

(RS) I cannot explain the behavior with the undefined symbols on various assemblys. On the other question, the use of a route to disk for the listing may be better used to avoid wasting paper. I use that facility to get a listing I can later look through with FED. Helps in debugging. In that way, I don't have to continuously use up trees when I need many iterative listings. Eliminating paging in the listing file keeps it a tad shorter. MRAS also happens to bypass the code which keeps track of printed lines when it is being routed to a file. If you ROUTE at the DOS level, MRAS' line check routine is not bypassed; thus you get paging. You can thusly have it either way.

(GP) Agreed, I could run the thing through a separate editing step manually, but that seems like a waste of time when the assembler already provides suitable functions. I don't mind using pseudo-ops in my source code, but when I do, I want them to affect the output, not just be ignored. The MISOSYS header also serves a purpose. It gives me the date and time of the assembly for later reference (suppose I have two listings, and need to know which is more recent) and also the version of the assembler that was used, in case something flaky shows up when attempting to re-assemble the code later. These things are often relevant in other environments, too. The IBM MVS assembler F

and assembler H (two versions of the latter) are NOT completely compatible. When reassembling someone else's old code, you can pull a zillion assembler errors by using the wrong version of the assembler!

(GP to LM) The VAX runs VMS 4.4, the IBM runs MVS/XA. Yes, there are OS facilities to print files, but I like to control the paging and headers and subheaders in my listings myself. Utility print programs just insert a header and page number every 50 lines or so. After all, what are the TITLE, SUBTTL, PAGE, and SPACE pseudo-ops for if not to control a listing format to make it orderly and easy to read? Even the IBM assembler has them.

(JB to GP) Why not just run it through a word processor, and put in the EXACT headers and footers YOU want, along with selecting the number of lines per page? after all, you might not need to see "MISOSYS....." at the top of each page, and could put a more meaningful title on it. You could also select your page breaks without having a lot of *PAGE pseudo-ops

As for routing the listing to a file, I usually do that so that I can save the paper, and use an editor to look through it. Finding errors is easy when you have an editor doing all the searching for you. It lets you QUICKLY look for all occurances of an offending lable, or find where it is defined. Printouts are essentially useless for this, comparatively. By putting in headers for printing, you just make the file that much bigger!

(GP to JB) While your approach is certainly logical enough, it isn't my way of working. I like a hard-copy listing for documentation and for making revisions, so I can flip back and forth between various parts of the program while I'm working. Headings and page numbers are essential to a readable listing in my opinion. Certainly I don't print the thing until I'm ready to do so, that's why I route it to a file - obvious errors can be corrected and the assembly re-run. Sometimes I even do some testing before deciding to print the listing or go back and make more revisions - as you say, it saves paper. But I invariably print the stuff eventually, and prefer not to run the assembly yet another time just to generate the listing to the printer. Also, if I have a "formatted" listing on a diskette, I can carry it to work and print it on a fast laser printer attached to our VAX or IBM systems rather than wait twenty minutes or more for my old Epson to spit it out. In that case, the listing file with no headings and page numbers is relatively useless to me.

(GP to RS) Agreed, once I figured out what I had to do, it was nice to be able to generate the listing either with or without the headings. Maybe a brief note pointing this out could be added to the manual next time around? I'll try to get more specific info for you about the multiply defined symbol problem. It has happened to me several times, and always with source that assembled fine when I just ran it through again.

(RS) What we may do in the next release of MRAS is provide an alternate switch to direct a listing to a file as it would appear if it was Line Printed.

HB asked a more technical question. One which others may like to know. He asked, "Roy, Can you give me a little technical info about how EDAS and MRAS handle macros? What I'm mostly concerned about is how much space each macro requires in the symbol table or wherever it is stored in memory. Is it a pointer to the macro definition in the source code or is the entire definition itself pulled into a memory table? Another way of asking the same question: if I have a large number of macros, perhaps from a *GET file, and some of them aren't used in a program, will they still take up a significant amount of space in memory during assembly?"

(RS) When a macro definition is read, it is stored in memory. It has to be permanently in memory since it needs to be accessible. It's stored in a compressed form with any macro arguments stored as a single byte. Comments are stripped unless you specify you want them (via the ';;'). A pointer to the macro storage is added to the symbol table. Thus, macros read but unused do take up space. There's also a little overhead in the header record which

is used to retain pointers during an expansion. Help any? A big disertation can be developed but this space (the Compuserve SIG) is impractical for that. Now then, if this topic is of interest to TMQ readers, I can produce an article on how we implemented macros in EDAS and MRAS.

(HB) Thanks, Roy. That gives me the information that I needed. But I thought that comments starting with a ';;' were stripped and comments starting with ';' were retained, yes? Then an intelligent way to define a large macro library for someone who doesn't have PaDS would be to wrap definitions in a IFREF / ENDIF envelope and use REF at the beginning of a program before *GETting the library, wouldn't it?

(RS) Now you're making me read my documentation. Page 2-46 says that comments with ';;' are not stored but are for ';'. You are right. Can't use REF unless you again follow the technique which I outlined in a previous NOTES FROM MISOSYS (Issue II) - that of forcing a dummy reference to the MACRO name. That's because macros must be defined before they are used.

KK had a question concerning the size of a program generated via the MRAS-MLINK sequence. He asked, "I do have a question about (NOT PROBLEM WITH) MRAS, MLINK and associated files. I wrote a short assembly program, actually rewrote my file list program to operate directly from DOS, with the idea of also being able to use the paged list part with LDOS (appropriate mods, of course). When the assembly and link was complete I called the program up to try it and wondered how such a short program could take so long to load. Turned out it was over 10K! After reassembling with the -GC switch the utility was back to its <2K size. I hadden't used any of the library routines, just calls to my SVC library (via *GET). Don't know that it's that much of a problem as small stuff like that can be easily done without the linker, but it might be interesting. Do you still have a labeling disassembler available?"

(RS) Did you by any chance have large DS regions in a code segment? If you go through the REL facility of MRAS with DS regions in a CSEG, then that will generate a ØØ byte for every space. For example, a DS 256*10 which would generate zero space by EDAS, would generate 2560 zeroes via the REL facility. If that same DS was in a data segment (i.e. DSEG), then it would again generate no object file space after the link unless you specified the Z option to be ON. A data space is only restricted from generating object space if it is in a data segment or common.

JB gave me the following report. "Hi there! I've got version 1 of MRAS, and am having a little bit of a problem with it. I'm using the OPTION pseudo-op, as described on page 2-37. All of the switches work fine (NE,NC,WE) with the exception of GC. When given that option in the /ASM file, all of the addresses are prefixed with ', meaning that it's code relative. When I use -GC from the command line however, it works just fine and leaves a blank space where the ' was (Normal operation, as far as I know.) Is there a problem with handling GC in the file itself?

(RS) The -GC switch must be entered in from the command line as it is too late for a useful detection via the OPTION statement. The default of CSEG is set before the source file is read. Once the file is accessed, that segment default can only be altered by a xSEG statement. It's not a bug in the code, just an oversight in the docs. It should have been noted.

BZ was the second MRAS user to report a problem with the -V switch. He gives us all kinds of reports and keeps us on our toes. I'm proud to have him as a customer. Here's his report. "I am attempting to compile DIFF/CCC (DL1) on my model I. 1) I had to split it into two pieces in order to compile (expected, split done right after end of main.) 2) When I tried to link, symbol table over flow (not unexpected). 3) When I used -V option in linking, the load module produced (without error message) does not work. Looking at the load module with FED-II, it appears OK up to AF47, but is all zeros above that

through the B575 high end of the top block. 4) Above problem experienced with high FCB8 FCB9 BLINK FD06 PRFLT11 FE25 ZDUBL1 (slightly modified rsdubl-with added delay when changing drive select to allow a modified drive with motor on upon select to work). An original copy of MLINK from distribution disk was also tested with same results."

(RS) You need a patch to MLINK. We recently discovered that the -V option never worked on the Model I/III release. Seems that output was still being placed into the C register (aka DOS 6) instead of the A register. Kevin Scott, who is working on a spreadsheet program for us, called us about that one. Here's the patch: **PATCH MLINK (DØ9,13=7D).**

JG reported that the -NH switch didn't work as advertised in MRAS. I checked it out and found a bug. Here's my answer. True, when MRAS detects the -GC switch, it forces -NH. This is, of course, after -NH is set/not set by the command line. That means that you can't specify -NH from the command line. As an aside, it would have worked had you specified -NH via an OPTION pseudoOP. In any event, MRS69/FIX is a patch which fixes it up. I assume you need the M4 version.

Product Highlights: PaDS

BPP had this comment concerning our Partitioned Data Set utility. "When I used to create Pads libraries, I got tired of listening to the drives, and waiting for the load. Somehow the /SYS libraries are lightyears faster. Maybe I just put oversized modules in mine, though; memory fades - out and in. That was probably it, since I put EDAS and everything else in them. Can Pads extract from /SYS now?

(RS) No, The /SYS ISAM files are a little different. Perhaps its time to really discuss this issue in the next QUARTERLY. The /SYS libraries do not have the member name directory in them. That piece is an internal part of SYS1. They also don't have the front end loader which was my invention. On the other hand, you are correct in that you probably put larger files into your working PaDS files. I have a user in West Germany (FRG) who has 200K-300K PaDS files. He complains about the slowness of the PURGE facility. Little did I know when I used a bubble-up algorithm in PURGE that folks would be creating such huge PaDS files. Such is life.

HB asks, "Roy, Is there a patch for Pro-Pads that would let it accept member names that are legal EDAS/MRAS labels? Its refusal to accept member names with underscores, "@" signs, etc. reduces its usability to store source code routines and macros.

(RS) At one time I could have rattled it off to you since we had such a patch for generating the LC library. Of course, that was only to the APPEND module. That meant that you could not use any other module (like COPY, LIST, etc) on those "funny" names. No one ever worked up patches to all the PaDS members.

BZ had a question about patching a PaDS member. "Can you tell me how to find a member in a PDS and patch it (I suspect some offset calculation is needed). right now I will just copy out, delete, compress, and append. But that is sure not the easy way. BG also wanted to directly patch a PaDS member.

(RS) There is no way to directly patch the member of a PaDS file short of writing a PATCH program specifically for that. We don't intend to. PaDS never sold well enough to justify such a facility. Unless you can generate a D-patch, the only way to X-patch a PaDS member is to copy it out, patch it, delete and purge the old, then append the new. That's because the PATCH utility provided with the DOS won't go past the transfer address record of the Front End Loader.

Here's a final note concerning PaDS for this issue. Don't use a password with the PDS(BUILD) command. BUILD internally generates its own ATTRIB command to change the protection level and password of the PaDS file. It doesn't scan the input line to see if you added a password field. An

error abort would then occur when PDS(B) tries to invoke its "invisible" ATTRIB command without a password in the file specification when you protected the file!

Product Highlights: PRO-WAM/PRO-NTO

Note to our new readers. The PRO-NTO product was recently renamed to PRO-WAM so as to avoid an infringement on Chemical Bank's Pronto Electronic Banking System's registered trademark; thus, any reference here to PRO-NTO is a reference to PRO-WAM - the new name for PRO-NTO.

Here's a question from DS, passed to us on Compuserve. "Another question about PRO-WAM. Is it something that would tie up memory in a 128K machine? For instance, even with the 1-Meg board made by Alpha, I need the extra 64K for big Visicalc/Multiplan spreadsheets, etc. Does it operate in memory or is it disk based? I'd like to buy it, but I want to make sure I'll have the memory to operate with comfortably. Does that make any sense?"

(RS) PRO-WAM uses one 32-K RAM bank plus about 2700 bytes of high memory. If you are using the Alpha board AND MY PATCHES TO TRSDOS (available in DL3 as alphal.fix or in this issue), then PRO-NTO will scan up to the first 7 banks for a free one. You could therefor have the lst two in use by memDISK or kept available for m/p & still have PRO-WAM installed. PRO-WAM can store 4 applications in the bank. It also uses that space for swap storage of the overlay area and video buffer (for 4-deep). It also needs the bank for export/import buffer area.

(JB to DS) PRO-NTO uses 32K of your extra memory. If you have installed the patches Roy developed to allow TRSDOS 6.2.x to control the Alpha board through the @BANK SVC, PRO-NTO can use any of the first 7 available, as can SAID. You would, however, have to come up with a way to make the first two look occupied until PRO-NTO is loaded, however.

(RS) JB's response pinpointed the reason why we wrote the BANKER utility. See that item in our language section.

BG wanted to use PRO-WAM with a memDISK and OverDrive all at the same time. Here's what I told him. You couldn't run OverDrive and memDISK and still run PRO-WAM. OD uses a memory bank; memDISK needs one as well. PRO-WAM also needs one. Your solution is to expand the memory of your Model 4. Check out the Alpha Tech memory board or the H.I.Tech XLR8 board. When the DOS is patched to enable @BANK support of additional memory, PRO-WAM, at least, will check for it.

KK asked us, "Roy, Is there a way to invoke PRO-NTO from a JCL? I CAN run PRO-NTO using a pre-made command line through KSMPLUS3. I used KSMPLUS3 to define a <CLR>-key combo to invoke the CARD/APP of PRO-NTO. (I set PRO-NTO to be invoked with <CTRL>W for Windows). The line I use within KSMPLUS3 is <CTRL><W> <F2>. This works fine.

I can't get the same line to work from within a JCL. This is true whether I write the JCL file with ALLWRITE or BUILD it with the HEX parameter. The JCL file looks OK when I examine it with PMOD6 zapper.

When I DO the JCL this is what happens: The special character for 17H, (a horizontal hollow rectangle), is not seen. The following character 82H for the F2 press is a high verticle solid rectangle. This prints to the screen. Then the job aborts. It isn't special for <CTRL><W> because I tried installing PRO-NTO with default <CTL><P>, same result.

I know that control characters can be used from within a JCL. Can they be used to run PRO-NTO?

(RS) The way to invoke a PRO-NTO/PRO-WAM application is by using the utility provided with PRO-NTO for doing it via JCL. That's what PRUN/CMD is for. Just put the line:

PRUN appname

into your JCL file when you want an application invoked. That, of course gets

it off of disk rather than via the memory installed version; however, I don't think that is a detriment.

Now then, I don't agree that you can blanketly state that JCL accepts control codes. The inference is that any control code is acceptable. That just is not so. Job Control Language uses the @KEYIN routine since it fetches file input only on line input requests. This means that the restrictions of the @KEYIN handler are at issue here. You can put anything in a JCL file, but to see its effect, look at what @KEYIN can accept. @KEYIN is displayed on pages 14-16 OF THE SOURCE. From this listing, one can observe that @KEYIN accepts the following:

0DH - terminate entry
08H - backspace one position
18H - backspace to BOL
09H - TAB to next tab stop
0AH - insert a linefeed
1FH - clear screen, reset to BOL

That's the entire list of control codes acceptable. Thus, you cannot have a JCL line entry using control codes to invoke any module filtering the keyboard because @KEYIN ignores control codes except those used for its own keyboard handling. Note that @KEYIN does not ignore extended ASCII codes - those generated in combination with the <CLEAR> key. Thus, it would be possible to use <CLEAR-P>, for instance, to invoke PRO-WAM. That would be acceptable to @KEYIN.

Note also that even with a charcter acceptable to @KEYIN, using KSM or KSMPLUS to generate the character string would require KSM or KSMPLUS to be installed before PRO-WAM is installed. That's because the "keystrokes" generated by KSM need to be "seen" by PRO-WAM. If KSM is installed after PRO-NTO, any keystrokes generated by KSM strings are passed back to the program calling @KEY before modules installed "closer" to the keyboard driver can detect them. In a device chain, input requests are passed along the chain until the driver gets the request; however, if a filter traps a code coming back from the driver and passes back a string of characters, no filter between that filter and the driver gets to detect the character string.

Here's a little item from Ken Kroninger 76317,2576. "Received the first copy of the quarterly, and it's definitely a good publication. Your list of Pronto (or whatever it's called now) applications even prompted me to update my LIST/APP to correct a bug. In case anyone's interested, the new version is now on-line.

I do have one question about the Mr. Ed package, specifically the VED/APP. Other than the obvious limit of the number of characters on the screen, is there any limit to the number of characters that can be exported to DOS? (not an application program, MAXDOS Ready). I at first thought that the size of the type-ahead buffer would probably be the limit, but I'm sure that buffer isn't as large as the number of characters I've thrown at it at times. Works beautifully!

(RS) Exporting is accomplished by hooking into the keyboard DCB and passing back a character as if it came from the keyboard; thus, there is no limit as to how many characters can be exported, in general. In export applicable to PRO-WAM, the limit is of course the entire screen. That's because the screen is copied into a 2K space in the alternate bank used by PRO-WAM. Then, *KI requests get satisfied by pulling characters out of the bank until the end marker is reached.

ERR asks, "Have you considered making Pro-Wam available to us 64k people with harddisk?"

(RS) PRO-WAM needs the extra memory for swap space. Virtual memory (on hard disk) would be too slow. PRO-WAM intensively uses the 32K bank it takes. Besides, RAM is very cheap these days.

From MC, "Roy, As an avid user of PRONTO (sorry, I mean PROWAM), I am gathering quite a library of $/APP files. Since I am also a devout user of PDS, as I have most of all the Utilities & programs in a PDS of some sort, even if a program calls upon

another, I usually patch the program for proper access. For example, in IFC, IFCLIST is called upon at times to list a file. I have thus patched IFC to read A(ILST) (where A is the PDS library file & IFCLIST was shorten to ILST so as not to exceed the 8 character name) as opposed to the IFCLIST. This works like a charm. But since PRONTO is so "complex" in a way, is there a way that something similar could be done with all those $/APP files and have them read from a PDS file ?? This would save Directory Entries and also some disk space. My 15Meg is getting quite tight. Thank you and keep up the good work, you are doing tremendous.

(RS) Yes, PRO-WAM is complex (note the required hyphen). You may want to explore PRUN which would be a lot easier to convert to loading a /APP from a PDS. The resident PRO-WAM /APP loader, being resident in HIGH memory, may be a lot more difficult to patch. Once you get PRUN adapted, you could then revise the resident module, if necessary, as PRUN uses the same loading routine as is in memory. Help? p.s. after you get PRUN patched, why not submit the patch for TMQ?

Here's a complaint from GM. "I saw the MISOSYS ad (in the August issue of 80-MICRO, page 95) about PRO-NTO!, now called PRO-WAM. According to this ad, PRO-NTO! is available at Radio-Shack via express order (#90-0353). I ordered mine a long time ago, but my Radio-Shack's dealer told me he could'nt get me one because PRO-NTO! is not available yet. Who's right? Should I trust the ad, proceed with a mail order and bypass my dealer? Or will it really be soon available at my Radio-Shack store?

(RS - not to be confused with Radio Shack) PRO-NTO/PRO-WAM has been available from R/S since about the fall of 1985! It seems that Radio Shack is not too concerned about express order of TRS-80 Model 4 products. They are supposed to have 25 units of PRO-NTO on consignment at their Ft Worth warehouse. Since MISOSYS has no pending order from Tandy on this product, I can only assume that they choose not to handle their part of the bargain. If you want the product within about 2-4 days of ordering it, give us a call at 800-MISOSYS (1p-5p M-F) and order it.

Finally, Paul Bradshaw provides us with another surprise. Here's his message from Compuserve.

#: 71090 S4/Mod4 Programming, 29-Oct-86 18:37:41; Sb: #new PRCTRL version; Fm: Paul Bradshaw 72177,2032; To: all

I have just uploaded a NEW version of PRCTRL (give the SYSOP time to put it up). It is a PRO-NTO/PRO-WAM application, and can be customized to work with **ANY** printer. It requires a separate data file (PRCTRL/DAT for non-Tandy DMP printer owners, or PRDMP.DAT for Tandy DMP series owners). Documentation is provided (and is highly recommended) in PRAPP.DOC. New features include user definable keywords (up to 8 chars in length, with any ASCII character valid), user definable code sequences (up to 7 codes per keyword, substitution fields, repetitions, and more). A total of 32 possible keywords/codes are now supported. A "point 'n click" method of sending codes (without typing the keyword) has been added as well, and there is wildcard searching through the keyword list, and the ability to "Send" strings.

(RS) Paul's PRCTL/APP is available in the download portion of our Compuserve SIG, PCS 49.

Product Highlights: SAID

LP reported that under certain conditions, SAID loses track of the current line number. SAID does have a problem in registering a correct line number when a block of lines are either added or deleted. SAID doesn't bother to resynchronize the line count after such operations. A simple GOTO top corrects the problem.

DM reported a few bugs in SAID. One was some problem he was having with error recovery after invalid file specifications. The other was a strange, very rare, problem of altering a small region of the text buffer after a DOS command was invoked. Here's my solutions

for DM's problems.

Would you believe that I finally got around to running down the DOS Command problem in SAID which you reported back in March? Well, I did. Here's the results of all your bug reports.

The three problems associated with error recovery after invalid or illegal drivespec designations were apparently corrected as they do not appear in version 1.1 of SAID but do in 1.0.

The DOS Command bug was tracked down to a situation where the invoked command was a program which set up a high-memory stack pointer (BASIC does this). Then, if the stack pointer would be in a memory position which was in the text region of SAID after SAID relocated itself down from high memory when it regained control, the stack pointer would be in the text region until it was changed by SAID. Unfortunately, by the time SAID got around to resetting its program stack, the damage was already done. I worked up a patch to version 1.1 which corrects this problem. [this was MSD64/FIX and MSD54/FIX included in the previous TMQ].

BG had a question about SAID. He wanted to know if the Model III version of SAID would use ony extra keybaord key not normally found on a Model III - like if it was a Model 4. Model III SAID uses a Model III keyboard driver - either its own or that of the DOS. If the keyboard driver supported it, you could use any key that was supported by the driver. SAID's internal driver recognizes only a Model III standard keyboard.

JG reported a strange problem to us which was hard to duplicate. Seems that if he tried to do a bank swap with PRO-SAID and all extra memory banks were already in use, SAID would crash. We finally came upon a sequence of command steps which duplicated the problem for us. It was then an easy manner to debug the problem. That patch is MSD65/FIX in The PATCH Corner.

Now for those who asked about the cost of Version 1.1, SAID 1.1 will be an upgrade. Please return your master disk containing SAID. It could be a SAID disk, an EDAS or PRO-CREATE disk, or an MRAS or PRO-MRAS disk. Make sure the disk is in a protective mailer. You will get back version 1.1. Also included will be a copy of the documentation which covers the added line number control. The cost will be $5 plus S&H charges [$2 for US, $3 for Canada, and $5 for other foreign].

Product Highlights: TYPEIN

The TYPEIN utility is part of the LS-UTILITY disk (Model 4 product) and the UTILITY DISK I (Model I/III product).

Here's some important dialog which was recently passed on to our message base on Compuserve.

(DK) Joe, I got the LS-Utility pkg as you suggested a couple of weeks ago and proceeded to utilize TYPEIN to do my PROFILE 4+ extended selection indexes. Everything looked A-OK and the thing ran fine 'till I looked at my sort! My FIRST sort field lacked a field number (it had a length), so the index was sorted on the first instead of last name. All other responses were in the RIGHT place. It so happens that the first field prompt in the extended menu is just after PROFILE searches ALL the drives for whatever files it needs. It's as though while out searching the drives TYPEIN lost the character. By the way, I tried to turn off TYPE and SMOOTH. I EVEN disabled my floppy drives to shorten the search time. None of these efforts worked. I finally answered that prompt with <SPACE><4><ENTER>, which worked. I "lost" the "space" that I didn't want anyway. It technically shouldn't have worked and wouldn't have manually. Any ideas? Like I said, I DID fix it and it DOES work. I just don't like to do things when I don't understand WHY they work, right OR wrong. Thanks in advance.

(LM to DK) Typein is not affected by timing. If you appear to lose a character, it means that the running program read it and ignored it. Programs often call the keyboard driver until nothing is returned to clear the buffer before each prompt is

printed (not my programs, though). Typein actually returns a "no character" status on every other call to avoid most of these problems. Either profile is doing this trick twice or a non-numeric entry was needed to terminate the previous entry.

(JJKD to DK) 'Tis most likely that Profile 4 is eating the keystroke. This is not an unusual occurance, and if Profile 4 is coded much like previous products from the small computer co., I'm not surprised. You have found the correct fix, and the fix should work consistently. As for why this happens, my guess is that there is a "vestigal" keystroke request to a question that is never asked. Perhaps a response to a prompt to mount a disk containing "missing" files that is never printed.

Product Highlights: ZSHELL

PB asks, "This may be a silly question (speaking of ROUTES), but is there anyway to direct the keyboard to take its INPUT from another source? (device or file)? I realize that the file aspect is taken care of with JCL and TYPIN, but is it possible with the ROUTE commands? (automatic RESET on EOF?). If not now, is it at all possible? (6.3 or later?). Routine and/or linking the *KI device has always confused on a Model 4 with TRSDOS 6.2.1, Pro-Zshell ver. 2.2 installed with pipe files on drive 0. No other drivers are installed. After displaying the first line of the file, the system displays a "redirection error" message, and then a repetitive display of the dos error message:

```
** Error code = 13, Returns to X'FB84'
** Data record not found during write
Open FCB, Drive = 0, DEC = 2C
Last SVC = 102, Returned to X'1A19'
```

The keyboard is totally un-responsive, but drive 0 is constantly being accessed. Re-booting and removing Zshell from the system removes the problem.

(JK to JB) Could it be that the "<" and ">" in his comment message are NOT being ignored by Zshell? If it scans that line, it will try to redirect input from "break", etc... me a bit."

(RS) You can ROUTE the *KI device from a file if the file then contains all the keystrokes neccessary to pick up from there. If you haven't already done so, you may want to explore our PRO-ZSHELL facility. That package allows complete control when you invoke a command line. You can have it switch back to keyboard on file EOF, automatically register a keyboard BREAK on EOF, or @ABORT on EOF. You can also redirect either *DO or *PR depending on a toggle character. ZSHELL also supports piping through tempy holding files and multiple commands on a line via the ";" command separator. It also accepts up to a 255-character physical line so that you don't run out of space with looooooong redirection, piping, and multiple commands entered at once.

PJ posed this conflict he was having with ZSHELL and JCL. The dialog of messages may prove useful to you.

(PJ) The following JCL:

```
.<break> to abort, <enter> to
continue
//alert (7,7)
```

(RS to PJ) ZSHELL permits redirection of a command line while JCL is in progress. There is no problem with that. What ZSHELL could do is also ignore commands which start with a period as it does for commands which start with a double-quote.

(LM to BP) Starting a line with a period only makes it a comment to *DOS*, not everything that might happen to look at the line. As I recall, even JCL evaluates the line (string substitution etc.). If Zshell is supposed to emulate unix, this is correct behaviour, as the line:

```
: >file
```

will create an empty file under unix (":" is a similar type of comment character to the unix sh).

(BP to LM) I disagree. Under UNIX (System V, anyway) the equivilant to the dot must be the pound-sign. The colon indicates a

null statement. Therefore # <Hello> is ignored as a comment. As is # cat * > /dev/tty

(LM to BP) No, the unix sh does not evaluate anything to the right of the pound-sign at all (i.e. no variable substitution as well as no redirection). Since JCL performs substitutions on lines that start with a period, it is more similar to the ":" under sh (i.e. a command which does nothing, but the side effects of evaluating the arguments to the right still occur). Thus there is no direct equivalent of the unix # for DOS, (but I think " would work for Zshell). Actually, earlier versions of the unix sh didn't have the # either, just the ":" type of commenting.

(PJ to RS) Roy, I guess I just assumed that Zshell would not operate during a JCL, but since it does, I will just work around this quirk. It seems there are quirks in every system I use, including the stuff I write. No matter how hard we try, little things like this slip in. Zshell happens to be one of my favorite utilities, and comes up during every boot. And I don't use half of its capabilities. Thanks for a neat program!

(PJ to BP) If you change your '<' and '>' to '[' and ']' then all muttering will be ignored. At least by Zshell. [<(chortle)>].

(PJ to LM) True, true. The program should work under JCL. But since it is possible to see if a JCL is active (bit 5, Sflag$), then I would expect Zshell to treat as a comment anything that DOS sees as a comment. I guess it all depends on how *pure* you want to be. Since I am running TRSDOS and not UNIX, I really don't care about UNIX rules.

(RS to PJ) I think that if you patch PRO-ZSHELL with:

PATCH ZSHELL (D08,52=2E:F08,52=22)

it will chage its "don't parse" handling from ignore on '"' to ignore on '.'. Thus, a TRSDOS 6 "comment" command will get ignored. I haven't checked it out but I just searched for the FE 22 code and it appeared only only once. Since it also went to @CMNDI directly when the comparison was TRUE, I really belive that is the spot. With that patch, then, the '.' will inhibit checking instead of the '"' currently used by ZSHELL. How's that?

Here's some bad news for BZ and others in the same boat. He asks, "When I try to use DOCONFIG with ZSHELL installed, it hangs up when I try to install the configuration later. Any patches known for this problem? Using a Model I with LDOS 5.1.4, 48K, 3 DSDD floppies."

(RS) ZSHELL is hooked into the system in a manner which cannot be switched out by DOCONFIG. There are no patches. There is also no "clean" way to provide the capability of ZSHELL as it is essentially an extension of DOS which has no interface demarcation point. One possible solution could have occurred, if in the past, the DOS supprorted a standard way of doing the reverse of @ICNFG, say @UCNFG. Then, a call to such a handler would have "unhooked" all such configured modules if they adhered to the design specs of the "undesigned" @UCNFG. ZSHELL can remove itself. Indeed, a ZSHELL (OFF) command does just that. It's ironic that the MS-DOS world has the same problem with terminate and stay resident modules. There is no easy way to remove them. There is also no way to redo a configuration without reBOOTing.

Double-sided obsession

(RW to JB) I would like to install, physically inside my model 4P, an 80-track, double-sided drive (in place of drive :1 only for now). I understand how to logically implement the installation with TRSDOS 6.2.0 and 6.2.1. Having read of your experience in your recent message to PG, your experience would be most appreciated! Please advise me: (1) the make and model of drive that you recommend. (2) the best source for this drive. (3) any major problems installing drive mechanically. 4) any major problems installing drive electrically, outside of the missing "side select" pin in the R/S cable (how about power reqd?) 5) any major problems maintaining the new drive. Thanks very much!

(JB to RW) Ok, I'll try almost anything once, so here goes! I use Teac 55F's. There are modifications that must be made to use this drive. First of all, the mounting holes in the drive cage need to be re-drilled, since the drive is longer between the front mounting hole and the front bezel. About 3/16" back from their original positions does it. Another mod required is that one of the door latches will interfer with the side drive cut-out in the front panel. This requires a bit of surgury, which took some time with an X-acto knife. A power tool, like a Dremel, will make quick work of the offending plastic (of the front panel, NOT the door latch).

Power requirements are actually less than with the original Tandon drives. there is a problem that the power connector is not long enough, as supplied. The fix is to look into the power supply compartment (already open if you've gotten far enough to work on the drives), and clip the first wire tie. This will allow you enough slack to pull the power lead further into the drive compartment.

Flipping the drive cable over will fix the side select problem, but you must make sure that both drives (not just the one you install if you only put one in) have exactly one drive select jumper. On the Tandons, this means pulling the 14-pin jumper block, and bending pins. DS0 is the second jumper from the "notch" end of the jumper block, so you would bend the pins for the 3rd, 4th and 5th jumpers out. The Teac is easier, since it is marked. you need jumpers on the following pins to work with TRSDOS/LDOS: HM, IU, SM, one of the DSx positions (DS1 for drive :1), and PM (found near the power connector).

After that, you can follow the instructions in DSDSK.HLP, found in DL3.

(RS to RW) I have recently installed Tandon TM50-2 drives into one of my 4P's. I have also put Teacs into Brenda's 4P. Believe me, the Tandons pop right in with no drilling or power cable pulling. They appear to be an exact replacement, mounting wise, for the original single-sided drives. Priority One Electronics was selling them for $69 in quantities of 10 or more. Sounds great for a group buy!

HD Controllers

(jjkd to JB) Well, usually neither SASI nor SCSI have anything to do with the TRS-80 line. The TRS-80 controllers are interfaced to the computer via a series of decoded I/O ports, and connect to the drive via a ST506 drive interface. Both the SCSI and SASI interfaces require host adapter hardware and driver software that is not commonly available for the TRS-80.

The drives used (for SCSI) are also somewhat less available and more expensive than ST506-style drives, though that should change soon.

For a TRS-80 hard drive, you are best off either getting a Radio Shack HD controller board, or a surplus WD1000/WD1001 type of board. RS's software will work with the former, and PowerSoft can supply software for the latter.

(jjkd to JB) The generic WD controller cards have a port based interface. You must supply "card select", which indicates that the current I/O operation is within the address range of the HDC. You'll also need to handle re-mapping of pin assignments as far as IN*, OUT*, A0-A7, D0-D7, etc.

The device you'll end up with will plug between the Model 4 and the HDC, and is called a "Host Adapter".

As far as Diplomat goes, distributors like them have branches all over the country. Check the local phone book.

Keybounce on the 4P

(DH to All) A friend of mine with two 4P's says that he has been having keybounce problems on both machines even in the slow mode. No problems in the model 3 mode unless he runs a WP that takes over the keyboard directly rather than runs thru KB/DVR. I hadn't heard of keybounce problems on the 4P before so wasn't much help. Anyone have any suggestions?

(BH to DH) I've got a pair of them myself, one an earlier B&W model, and the other a green screen gate array. Both exhibit a certain amount of keybounce, but not enough to get upset about.

I checked out the schematics for both versions and found the 8x8 keyboard matrix uses a 1.5K resistor pack for pull-ups (part 'RP2'). Am inclined to think 1K or even 820 ohm would have been a better choice, given the length of the keyboard cable. Check out your local RSCC Service Center, I understand they have a Tech Bulletin which covers this.

I physically checked the resistor pack on my systems, both are teeny little parts requiring surgeon's hands to replace, but a good hardware tech would be able to handle it. I think I could also, but really don't want to risk damaging a CPU board for such a trivial problem for me.

On the H.I.Tech XLR8 board

(PB to RS) That's great news about the new expansion board! I have a few questions about it that you may be able to answer. Is the memory expandable beyond 256K? Is the 8Mhz speed compatable with all software (that you can tell)? What's this about a "HD64180" Z80 equivilent chip? I know you'd rather wait until you can evaluate the product in full, but could you give an advance thumbnail evaluation? I was certainly supprised to find a company offering something NEW for the model 4, and would like very much to purchase the board! Thanks for any enlightenment you can provide.

(RS to PB) The XLR8 board includes an Hitachi CPU chip (HD64180) which supports a superset of the Z80 instruction set. For a lot of details concerning that chip, see the article by Steve Ciarcia in one of the recent (within the past year) issues of BYTE magazine where he provides plans for his "lunchbox" computer. The XLR8 uses the same bus as Ciarcia's for external access; thus, you can expand it further using any of the expansion modules discussed in BYTE. The 64180 itself directly supports 512K of RAM although the XLR8 implementation provides only 256K on the board. The board is small so it can fit into the modem slot of a 4P. Therefore, there is no room for an additional 8 memory chips. It's a really good board.

Radio Shack National Parts

(SS to All) Does anyone know of a source for a Technical Reference manual for the TRS-80 Model III? I have no idea where to find one. I think I asked about this once before, and was given an answer, but then was told by my local Radio Shack dealer that she could get me one from Fort Worth. Seems she was wrong, but I forgot to save the message, and so here I am again.

I understand that there is a Sams Photo-fact manual on the Model III, but that it might be out of print. Where can I get in touch with Sams, and how much should I expect to have to pay for such a book?

(jjkd to SS) The service manual and hardware tech reference manual can be ordered from Radio Shack National Parts at (817) 870-5662. They take VISA/MC/AMEX and ship promptly if in stock. I think it's number MS-260-1063 (insert the stock number from the sticker on the case of the computer). Should be less than $20, but they will tell you the exact amount (less shipping) when you place the order.

I've seen the SAMS book on the PC, if their Model 3 manual is of the same caliber you are better off with Tandy's. Similar price point, any full service book store should be able to order it for you if you really want it (B. Dalton's, Krochs & Brentanos, etc.).

Bored of boards?

(SS to All) How can I tell if there is a serial port on my TRS-80 Model III? It looks as though there might be one, as there is a small board with edge connectors at the bottom mounted right behind the disk drive rack, but since I have absolutely no documentation at all on the computer I am at a loss to determine if that is in fact the serial card. If it does not have such a device, to where should I go to get one?

(jjkd to SS) In the Model 3, there are a possibility of up to four cards in the computer. These are:

1) CPU
2) FDC
3) Serial port
4) Graphics board

The first everybody has. The second comes along with at least one disk drive, and is the card closest to the disk drives. A 34 conductor ribbon cable runs from the drives to the FDC board. The serial board is next to the FDC board, and its DB25 connector will show up on the bottom of the computer next to the fifty pin bus connector. The last, the graphics board, is mounted in a pregnant protrusion of a modification to the rear CPU metal shield.

Any or all of the above can be ordered from Radio Shack. 26-1148 at $79 should be orderable by any true Radio Shack Computer Center and will get you a serial port if you don't already have one.

The 4P BOOT ROM by Roy Soltoff

I'm sure that a lot of you Model 4P owners already know that your 4P is a strange animal when it comes to booting up a DOS disk. That's because the 4P does not have the BASIC interpreter in a ROM like the normal Model 4. The 4P actually contains a little 1K ROM which is used to read the BOOT sector off of a number of devices. That ROM can also read in a Model III BASIC ROM image file (called MODELA/III) to emulate a Model III machine.

The behind the scenes operations of this boot ROM take place to boot your DOS disk then the ROM mysteriously disappears. For those of you having an interest in the inards of your machine, here's a little program to capture the contents of that ROM and copy it to a safe place - a place whereby you can preserve a more readable copy. The program is short enough. I won't even switch to single column to print it.

```
        ORG     2600H
GETBOOT DI
        LD      A,101
        RST     40
        LD      A,(IY+'o'-'a')
        EX      AF,AF'
        XOR     A
        OUT     (84H),A
        LD      A,1
        OUT     (9CH),A
        LD      HL,0
        LD      DE,8000H
        LD      BC,1000H
        LDIR
        EX      AF,AF'
        OUT     (84H),A
        XOR     A
        OUT     (9CH),A
        EI
        RET
        END     GETBOOT
```

What does this short program do? First it turns off the interrupts since we don't want any of those buggers getting in the way of our machine gyrations. Next it obtains and preserves the contents of the system's OFLAG$. This flag keeps the image of the memory management port, port 84H. The out to port 84H with a zero value selects the standard Model III memory image mode - at least it does in a Model 4. The kicker is that OUT of a 1 to port 9CH. This switches in the BOOT ROM.

The ROM is addressed starting at address 0. Thus, the next piece of code copies 1K of memory starting from address 0000H to

start at address 8000H. This would obviously extend to 8FFFH. After that, the routine performs the reverse of what it did on entry. The BOOT ROM is disengaged and the memory management port is restored to its original value. A final EI brings back the interrupts.

What you now have is a copy of the BOOT ROM offset to address 8000H. You can either disassemble this directly using PRO-DUCE or DUMP the image to a disk file and look at it later. Once you dump it to a disk file, you may want to use CMDFILE or PRO-CESS to offset the load module back to address 0000H for direct disk disassembly. What follows is a screening data text file for use with PRO-DUCE in disassembling the BOOT ROM - at least the ROM version which is in my machine. I will avoid providing my commented disassembly of this ROM as it is considered a work copyrighted by Tandy Corp. Besides, it's good exercise for those learning assembly language. However, as some of you Compuserve subscribers have noticed lately, a few individuals have already been hard at work looking at the "booting" portion of the 4P ROM so as to write a booting driver to be used in directly booting their 4P off of their hard drive.

```
$1-4,$63-65,$6c-73,$7b-7e,$80-8d
$9c-b0,#273-4b6,$4b7-4bb,$4bc-4c4
$4c5-4cf,$4d0-4d8,$4d9-4e8,$4e9,4ea
$4eb-4f2,$4f3-4f8,$4f9-500,$501-50b
$50c-51e,$51f-52f,$530-537,$538-542
$543-556,$557-55d,$55e-571,$572-57c
$57d-581,$582-58c,$58d-58f,$590-59f
$5a0-5a8,$5a9-5ba,$5bb-5c6,$5c7-5d5
$5d6-5e1,$5e2-5e5,$5e6-5f0,$5f1-5fe
$5ff-612,$613-618,$619-625,$626-642
$643-652,$653-65a,$65b-66c,$66d-674
$675-67b,$67c-691,$692-697,$698-6a0
$6a1-6cd,$6ce-6d6,$6d7-6dc,$6dd-6e2
$6e3-6ef,$6f0-6f9,$6fa-6ff,$700-70c
$70d-71c,$71d-725,$726-735,$736-74c
$74d-755,$754-75a,$75b-777,$778-780
$781-786,$787-791,$792-7b4,$7b5-7ba
$7bb-7c2,$7c3-7d0,$7d1-7d9,$7da-7e1
$7e2-7ea,$7eb-7f6,$7f7-807,$808-80b
$80c-81a,$81b-829,$82a-83d,$cc4-cd4
cd5-cf1,$ed2-ee2,$ee3-f0b,$f0c-f31
$f32-f55,$f56-f69,$f6a-f6d,$f6e-f91
$f92-f97,$feb-fff.
```

Update on Alpha Tech memory board patches

With the upcoming release of Logical System's LS-DOS 6.3 for the TRS-80 Model 4, it was necessary to slightly revise the patches released by MISOSYS. Out of the three patches provided, BOOTAT, SYSØAT, and SYS1AT, only the SYSØAT patch required a change. In the interests of continuity, this issue will print all three of the previous patches and the revised patch for SYSØ in The PATCH Corner.

In the past, MISOSYS has made these patches available to users by having them send us a disk in a mailer with return postage or $1 and a return address label. We have had too many requests which did not adhere to those guidelines (we get a disk but no return address label, we get a disk but no return postage, we get no disk but are asked to provide one, etc.). Such requests cost us money. Since we're not in business to lose money, we have decided that the patches are now available from us only on DISK NOTES 6 (the corresponding DISK NOTES for this TMQ issue). DISK NOTES are $10 + S&H ($7.50 to subscribers when ordered with the QUARTERLY COUPON). The following information describes the installation of the patches.

MISOSYS has developed three patches to TRSDOS 6.2 so as to enable the DOS to directly address the Alpha Technology memory board. Once the patches are applied to a system diskette, that diskette must be used on a Model 4/4p with the Alpha Tech board installed. The DOS will scan for the installation of 256K, 512K, 768K, or 1024K of memory.

The @BANK supervisor call (SVC-102) will now support a bank number in the range of <0-30>; the SVC will behave exactly as it does under a standard 64K or 128K Model 4/4p with the exception of the highest bank number supported being bank 30. Note that this represents 31 banks of 32K RAM per bank (the other 32K bank is addressed from address 0000H and is not part of the bank switching).

It is recommended that ALL individuals or companies making use of the Alpha Tech board and are writing software to utilize same will NOT write specialized handlers

for the bank switching. PLEASE rely on the DOS's @BANK handler for support! The patched system should correctly handle banking of all 31 banks even with interrupts enabled. The patches effect all components of the DOS which relate to the @BANK handler. Thus, if you code a program exactly as you would for an unpatched TRSDOS 6.2, you will NOT have to write specialized programs for each arrangement. There are three patches which make up the 6.2 modifications. These are:

```
BOOTAT/FIX - to patch BOOT/SYS
SYS0AT/FIX - to patch SYS0/SYS (6.2)
SYS0AT63/FIX - to patch SYS0/SYS (6.3)
SYS1AT/FIX - to patch SYS1/SYS
```

Apply all three patches to a working system diskette. Reboot while you inhibit any configuration which may be on the disk. You should be up and running. The proper patch command syntax for each patch is shown in the associated FIX file (i.e. it tells you the /SYS password and designates the use of the (O=N) parameter for the PATCH command. Since the patch to SYS1/SYS will cause your system to lock because the changes to BOOT/SYS and SYS0/SYS won't be installed into RAM until you reBOOT, I recommend that you SYSRES the SYS1/SYS module before starting the patching. The following sequence of commands will result in a procedure that will not lock up your system:

```
SYSTEM (SYSRES=1)
PATCH BOOT/SYS.LSIDOS BOOTAT (O=N)
PATCH SYSØ/SYS.LSIDOS SYSØAT (O=N)
PATCH SYS1/SYS.LSIDOS SYS1AT (O=N)
BOOT -----> hold down the <CLEAR>
key to inhibit any configuration
after you enter any DATE and TIME
```

Note that these patches are being supplied by MISOSYS on an as-is basis. Absolutely no support will be guaranteed by MISOSYS. These patches apply specifically to TRSDOS 6.2.0. and LS-DOS 6.3.0. They may not pertain to any other release of TRSDOS 6.

Notes on the H.I.Tech XLR8

We have had an opportunity to try out a new board which plugs into the Model 4/4P/4D. This board changes the machine over to a much faster and more powerful CPU chip which just happens to be totally compatible with the current Z80. The XLR8 board plugs into the existing Z80 socket once the old Z80 is carefully removed. I say carefully because you may want to, or need to, remove the XLR8 board sometime and re-insert that Z80 chip to restore your machine to its original state.

The XLR8 board measures approximately 4-1/2" by 5". The board includes an Hitachi HD64180 CPU chip which runs at about 6.144 MHz, 256K of RAM, and additional support circuitry. The package includes documentation. Since I received an early board, my documentation was only preliminary; however, it included about 22 pages of information on the board, its installation into your computer, the software provided with the board, and a little information for programmers. Unfortunately, we have been so ingrossed in getting together the 5.3 release of LDOS, this issue of THE MISOSYS QUARTERLY, and riding honcho on a handful of MS-DOS products, that we have not had time to do a thorough workout of the XLR8.

With that in mind, let me just tell you a little about this product. Speed is one thing. You should be able to see improvements of from 20% to 100% in increased program speed. Why the big variation? For one thing, it depends on your computer's vintage. The 64180 is a very fast chip. Part of its increased throughput is due to more efficient micro-code. A large percentage of its increased throughput is due to its ability to reduce the amount of time taken up by memory refresh. The chip has programmable timing control over wait states, memory refresh, and I/O channels. Depending on your vintage of Model 4, various parts of your machine may not be able to operate at the increased CPU speed; thus, software filters provided with the package may be needed to slow the CPU down during certain functions (such as keyboard scanning, device I/O). Therefore, depending on a program's utilization of system resources,

various periods of CPU slowdown may be utilized.

The folks at H.I.Tech have done their homework when it comes to interfacing the board to TRSDOS 6.2. They have supplied all the software modifications to (1) dynamically adjust the speed based on DOS requests, (2) alter the @BANK handler to support the additional 8 banks of 32K (the 256K of RAM), and (3) provide a new RAMDISK which utilizes all of the extra memory above 64K. With their FIXBANK software installed, the @BANK handler allows bank requests to designate banks 0 through 10 instead of just 0 through 2. Folks already familiar with my @BANK handler for the Alpha Tech board already know the value of addressing extra memory through @BANK. The only problem I can say I had with the XLR8 board and the supplied software is that it takes up more of my limited low memory space than I would like to give up.

I have the board installed into a somewhat recent green-screen 4P. Thus I can run this machine without the supplied FIXALL filter installed. It is this filter which reduces the CPU speed dynamically according to DOS service requests. One extreme caution for now is that if the XLR8 board is operated without the FIXALL filter, then single density disk transfer should not be attempted - at least not until I can more fully investigate the problem I experienced and come up with a patch. There is a tendency to lose data. When operating WITH the FIXALL filter supplied, which slows down the CPU during disk I/O requests, I didn't have any problem with SDEN data transfer. Timing problems such as this are usually correctable in ways other than requiring a CPU slowdown. I really feel that the problem is correctable in the disk driver by a simple patch. I expect to be looking more into reducing the low-memory requirements needed by the FIXALL and FIXBANK modules by strategic patches to the DOS just as soon as time permits.

Any hardware or software tinkerer who is going to get interested in this board would be best advised to dig up all of the issues of BYTE magazine where Steve Ciarcia's articles on his lunchbox computer appear. Arlen Nipper, the designer of the XLR8 board, hung a bus extension onto the XLR8 board which is identical to that used by Ciarcia. Thus, any of the additional boards usable with the lunchbox, is also usable with the XLR8. For instance, the November 1986 issue of BYTE begins to discuss the GT180 - a high resolution color graphics board with specifications comparable to the IBM PGA and Enhanced Graphics Adaptor (EGA), the McIntosh, Amiga, and the Atari 520ST. The GT180 display resolution is 640 by 480 with 16 of 4096 colors. With the XLR8 and the GT180, guess what you could do with your old Model 4?

An astute and highly interested individual would also comb through the DOS looking for ways to make better use of the 64180. You can probably bank on LS-DOS 6.3 as a stable release for the machine for many years to come; thus, you would not have to be worried about DOS changes. The 64180 has a hardware 8x8 multiply which could be used in all sorts of places. Prime consideration would be the video driver. Next would be replacement of the @MUL8 SVC routine with one using the 64180.

The 64180 chip has two full-duplex serial channels. There is a DIP header on the XLR8 board for a connecting RS232 cable hookup; however, such a cable is not provided with the package as it would then not pass FCC regulations for emission of a class B device (according to what I understand). I believe that H.I.Tech is making such a cable available as a separate item - I don't think the FCC can stop you from selling a cable! The 64180 also has a programmable timer (PRT) which operates under software control. It should be possible to utilize this PRT with appropriate software to operate as a hardware clock - albeit not one which automatically maintains the date/time when powered down but one which could probably keep accurate time once set after bootup. These are just some of the possibilities of this device.

As I previously stated, I have been burdened with other work which has kept me from doing much with the XLR8; however, next year is another story. I hope to be spending some of my time on the XLR8.

The PATCH Corner: General Information

The following information should be read before you type into a file, any of the patches noted in THE MISOSYS QUARTERLY.

It is unfortunate that our printer prints the letter "O" and the number "0" almost identically. Unless we utilize a filter to "slash" the number zero, the two are difficult to distinguish. However, when it comes to patches, all is not lost. In an LDOS 5 or TRSDOS 6 direct patch, the letter "oh" is not used in the patch code (it may appear in comments which are lines beginning with a dot). The direct patch format of TRSDOS 6 which we use in our patches is:

```
Drr,bb=xx xx xx xx xx xx ...
Frr,bb=xx xx xx xx xx xx ...
```

The patch is usually a pair of lines. The first line begins with the capital letter, "dee". This is immediately followed by the "rr" field (which stands for record). The "rr" field is always two hexadecimal digits. Actually, it can be a 4-hexadecimal digit number if the file to be patched has more than 256 sectors. Hex digits use nothing but the numbers zero through nine and the first six letters of the alphabet: A,B,C,D,E,F, or a,b,c,d,e,f. The record number is immediately followed by the "bb" field (which stands for byte). The byte field is also two hexadecimal digits - just like the record field. This is immediately followed by an equal sign, "=". The equal sign is immediately followed by the first patch byte (the "xx" shown above). The patch byte is again two hexadecimal digits. Where more than one patch byte is included on a line, it is separated from its predecessor by a single SPACE. The line is terminated with an ENTER.

TRSDOS 6 patch format uses a "find" line record. This is used to verify that the file being patched is actually the file you want patched. All of the bytes noted in the "F" line or lines must be matched in the file before any of the "D" patches will be utilized. The second line of the pair begins with the letter "F" which stands for FIND. The next six positions are identical to the preceding "D" line. Following the equal sign on the FIND line are pairs of hexadecimal digits which should align themselves with the preceding line.

So far, the letter "oh" is not used. The only place outside of a comment line where you could find the letter "oh" used is if instead of showing the patch bytes as a series of hexadecimal pairs, it was depicted as a string. A string could be used if one was patching a string of displayable ASCII characters. For instance, the patch:

```
D03,14="This is a new string"
F03,14="extra space for what"
```

would replace the string, "extra space for what", with the string, "This is a new string". Strings are shown within double quotes. That's the only place where a letter "G" through "Z" could be used.

Also, even though TRSDOS supports the colon notation to put more than one patch line on the command line (e.g. "PATCH TEST (D01,27=56:F01,27=65)"), it does not support the colon separator when used in a FIX file. The "D" and corresponding "F" records must be on physically sepapate lines. In order to conserve space in THE MISOSYS QUARTERLY, we may logically print more than one FIX line on a printed line; HOWEVER, ALWAYS USE A HARD <ENTER> FOR THE COLON WHEN TYPING IN A FIX FILE.

```
. BC51/FIX - 09/03/86 - OPTIONAL Patch to EnhComp BC/CMD:. Apply via: PATCH BC BC51
. Patch changes the raise-to-a-power character to UP ARROW: . from the "^" character.
D3E,02=5B:. WAS =5E
. Eop ----------
. BC52/FIX - Patch to EnhComp BC/CMD - 11/05/86
. Patch corrects BREAK handler with -NX option specified:. Apply via: PATCH BC BC52
D29,D2=0F 5B:. was =EE 70
D05,27=CD EE 70 0E BF C3 EE 70:. was =00 00 00 00 00 00 00 00
. Eop ----------
```

```
. BC61/FIX - 09/03/86 - Patch to PRO-EnhComp BC/CMD
. Corrects closing of SUPPORT/DAT file:. Apply via: PATCH BC BC61
D1F,78=00:F1F,78=C8
. Eop ----------
. BC62/FIX - Patch to PRO-EnhComp BC/CMD - 11/05/86
. Patch corrects BREAK handler with -NX option specified:. Apply via: PATCH BC BC62
D29,E5=09 25:F29,E5=06 45
D05,21=CD 06 45 0E BF C3 06 45:F05,21=00 00 00 00 00 00 00 00
. Eop ----------
. CED51/FIX - Patch to EnhComp's editor
. Corrects tokenization of ON exp address lists so that
. renumbering will adjust the entire list as required.
. After applying the patch, it will be necessary to edit
. any line containing an address list to re-tokenize.:. Apply patch via: PATCH CED CED51
D14,17=7E B7 CA 27 79 FE 2C C2 2D 78 DD 77 00 23 DD 23 C3 B9 78
. WAS =00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
D1B,59=B2 6A:. WAS =2D 78
. Eop ----------
. CED61/FIX - 09/03/86 - Patch to PRO-EnhComp CED/CMD
. Corrects "p" to display 23 lines instead of 15 lines:. Apply via: PATCH CED CED61
D1C,A3=17:F1C,A3=0F
. Eop ----------
. CED62/FIX - Patch to PRO-EnhComp's editor
. Corrects tokenization of ON exp address lists so that renumbering will adjust
. the entire list as required. After applying the patch, it will be necessary
. to edit any line containing an address list to re-tokenize.
. Apply patch via: PATCH CED CED62
D14,2C=7E B7 CA 3C 4D FE 2C C2 42 4C DD 77 00 23 DD 23 C3 CB 4C
F14,2C=00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
D1B,6E=C7 3E:F1B,6E=42 4C
. Eop ----------
. DSORT61/FIX - 09/02/86 - Patch to 10/23/85 Ver DSM4's DSORT module
. Patch corrects index file generation when virtual memory is used
.   and the number of records is a multiple of 128:. Apply via: PATCH DSORT DSORT61
D0C,4E=00 00:F0C,4E=28 03
. Eop ----------
. MC51/FIX - correct pointer to array - RND - 08/22/86:. Apply via: PATCH MC1 MC51
D42,80=7E 21 61 8E 01 04 00 ED B1 21 01 00 C8 2B C9 00 00 00 00
. was =6E 26 00 E5 C1 21 61 8E CD EE E4 21 01 00 C9 21 00 00 C9
D42,D8=3E 04 C1 CD F5 93 C5 C8:. was =F1 E1 E5 F5 11 05 00 19
D42,E0=18 15 20 05 DD 36 05 03 C9 C5 CD 8C 94 C1 7C B5
. was =6E 26 00 11 04 00 CD A0 E3 7C B5 20 09 F1 E1 E5
D42,F0=C8 C5 CD 54 AE F1 C9 3E 0B 18 02 3E 0C D1 E1 E5
. was =F5 E5 CD EF 93 F1 C9 F1 E1 E5 F5 11 05 00 19 6E
D43,00=D5 11 05 00 19 BE 18 05 01 02 00 94 00 3E 08 18
. was =26 00 11 0B 00 CD A0 E3 01 02 00 94 C9 F1 E1 E5
D43,10=F0 21 01 00 C8 2B C9 00 00 00 00 00 00 00 00
. was =F5 11 05 00 19 6E 26 00 11 08 00 CD A0 E3 C9
D3D,55=02 03 0C 05 00 00 00 00 00 00 00 00 00 00 00 00
. was =87 93 03 00 05 00 83 93 03 00 83 93 02 00 83 93
D33,02=F3 93 C1 CD DA 93 18 05 00 00 00 00 00:.was=8C 94 F1 7C B5 28 06 DD E5 CD 54 AE F1
D35,B1=F3 93 C1 CD DA 93 18 05 00 00 00 00 00:.was=8C 94 F1 7C B5 28 06 DD E5 CD 54 AE F1
D5E,28=0C:. was =03:D03,11=62:. was =61
. Eop ----------
```

```
. MC52/FIX - correct pointer to array - RND - 08/22/86
. Apply via: PATCH MC2 MC52
D41,C9=7E 21 45 8D 01 04 00 ED B1 21 01 00 C8 2B C9 00 00
. was =4E 06 00 21 45 8D CD 36 E6 21 01 00 C9 21 00 00 C9
D42,23=3E 04 C1 CD 40 93 C5 C8 18 15 20 05 DD:.was=F1 E1 E5 F5 11 05 00 19 6E 26 00 11 04
D42,30=36 05 03 C9 C5 CD A3 93 C1 7C B5 C8 C5 CD 8C AE
. was =00 CD E8 E4 7C B5 20 09 F1 E1 E5 F5 E5 CD 3A 93
D42,40=F1 C9 3E 0B 18 02 3E 0C D1 E1 E5 D5 11 05 00 19
. was =F1 C9 F1 E1 E5 F5 11 05 00 19 6E 26 00 11 0B 00
D42,50=BE 18 05 00 3E 08 18 F0 21 01 00 C8 2B C9 00 00 00 00 00 00 00 00
. was =CD E8 E4 C9 F1 E1 E5 F5 11 05 00 19 6E 26 00 11 08 00 CD E8 E4 C9
D3C,35=02 03 0C 05 00 00 00 00 00 00 00 00 00 00 00 00
. was =D2 92 03 00 05 00 CE 92 03 00 CE 92 02 00 CE 92
D33,2A=DD E5 CD 3E 93 C1 CD 25 93 18 05 00 00 00 00 00
. was =DD E5 CD A3 93 F1 7C B5 28 06 DD E5 CD 8C AE F1
D34,DF=DD E5 CD 3E 93 C1 CD 25 93 18 05 00 00 00 00 00
. was =DD E5 CD A3 93 F1 7C B5 28 06 DD E5 CD 8C AE F1
D5E,5C=0C:. was =03:D03,06=62:. was =61
. Eop ----------
. MC60/FIX - correct pointer to array - RND - 08/20/86
. Apply via: PATCH MC MC60
D50,A2=7E 21 13 6E 01 04 00 ED B1 21 01 00 C8 2B C9 00 00 00 00
F50,A2=6E 26 00 E5 C1 21 13 6E CD 23 DC 21 01 00 C9 21 00 00 C9
D50,FA=3E 04 C1 CD DF 75:F50,FA=F1 E1 E5 F5 11 05
D51,00=C5 C8 18 15 20 05 DD 36 05 03 C9 C5 CD 76 76 C1
F51,00=00 19 6E 26 00 11 04 00 CD D5 DA 7C B5 20 09 F1
D51,10=7C B5 C8 C5 CD 1C 95 F1 C9 3E 0B 18 02 3E 0C D1
F51,10=E1 E5 F5 E5 CD D9 75 F1 C9 F1 E1 E5 F5 11 05 00
D51,20=E1 E5 D5 11 05 00 19 BE 18 05 00 3E 08 18 F0 21
F51,20=19 6E 26 00 11 0B 00 CD D5 DA C9 F1 E1 E5 F5 11
D51,30=01 00 C8 2B C9 00 00 00 00 00 00 00 00
F51,30=05 00 19 6E 26 00 11 08 00 CD D5 DA C9
D49,37=02 03 0C 05 00 00 00 00 00 00 00 00 00 00 00 00
F49,37=71 75 03 00 05 00 6D 75 03 00 6D 75 02 00 6D 75
D3D,5A=DD E5 CD DD 75 C1 CD C4 75 18 05 00 00 00 00 00
F3D,5A=DD E5 CD 76 76 F1 7C B5 28 06 DD E5 CD 1C 95 F1
D40,32=DD E5 CD DD 75 C1 CD C4 75 18 05 00 00 00 00 00
F40,32=DD E5 CD 76 76 F1 7C B5 28 06 DD E5 CD 1C 95 F1
D71,3C=0C:F71,3C=03:D03,3A=62:F03,3A=61
. Eop ----------
. MC60C/FIX - correct #asm and +c combination - 08/30/86
. Apply via: PATCH MC MC60C
D88,56=B7 28 16 2A A0 AC 7C B5 20 0F 2A 5D A8
F88,56=6F 26 00 7C B5 28 06 2A A0 AC CD D9 DB
. Eop ----------
. MC61/FIX - correct pointer to array - RND - 08/20/86
. Apply via: PATCH MC1 MC61
D42,6F=7E 21 50 62 01 04 00 ED B1 21 01 00 C8 2B C9 00 00 00 00
F42,6F=6E 26 00 E5 C1 21 50 62 CD 9E B8 21 01 00 C9 21 00 00 C9
D42,C7=3E 04 C1 CD E4 67 C5 C8 18:F42,C7=F1 E1 E5 F5 11 05 00 19 6E
D42,D0=15 20 05 DD 36 05 03 C9 C5 CD 7B 68 C1 7C B5 C8
F42,D0=26 00 11 04 00 CD 50 B7 7C B5 20 09 F1 E1 E5 F5
D42,E0=C5 CD 43 82 F1 C9 3E 0B 18 02 3E 0C D1 E1 E5 D5
F42,E0=E5 CD DE 67 F1 C9 F1 E1 E5 F5 11 05 00 19 6E 26
D42,F0=11 05 00 19 BE 18 05 00 3E 08 18 F0 21 01 00 C8
F42,F0=00 11 0B 00 CD 50 B7 C9 F1 E1 E5 F5 11 05 00 19
D43,00=2B C9 00 00 00 00 00 00 01 02 00 68 00 00
```

```
F43,00=6E 26 00 11 08 00 CD 50 01 02 00 68 B7 C9
D3D,44=02 03 0C 05 00 00 00 00 00 00 00 00 00 00 00
F3D,44=76 67 03 00 05 00 72 67 03 00 72 67 02 00 72 67
D32,EE=DD E5 CD E2 67 C1 CD C9 67 18 05 00 00 00 00 00
F32,EE=DD E5 CD 7B 68 F1 7C B5 28 06 DD E5 CD 43 82 F1
D35,9D=DD E5 CD E2 67 C1 CD C9 67 18 05 00 00 00 00 00
F35,9D=DD E5 CD 7B 68 F1 7C B5 28 06 DD E5 CD 43 82 F1
D5E,17=0C:F5E,17=03:D02,FE=62:F02,FE=61
. Eop ----------
. MC62/FIX - correct pointer to array - RND - 08/21/86
. Apply via: PATCH MC2 MC62
D41,E4=7E 21 60 61 01 04 00 ED B1 21 01 00 C8 2B C9 00 00
F41,E4=4E 06 00 21 60 61 CD F7 C1 21 01 00 C9 21 00 00 C9
D42,3E=3E 04 C1 CD 5B 67 C5 C8 18 15 20 05 DD 36 05 03 C9 C5
F42,3E=F1 E1 E5 F5 11 05 00 19 6E 26 00 11 04 00 CD A9 C0 7C
D42,50=CD BE 67 C1 7C B5 C8 C5 CD A7 82 F1 C9 3E 0B 18
F42,50=B5 20 09 F1 E1 E5 F5 E5 CD 55 67 F1 C9 F1 E1 E5
D42,60=02 3E 0C D1 E1 E5 D5 11 05 00 19 BE 18 05 00 3E
F42,60=F5 11 05 00 19 6E 26 00 11 0B 00 CD A9 C0 C9 F1
D42,70=08 18 F0 21 01 00 C8 2B C9 00 00 00 00 00 00 00
F42,70=E1 E5 F5 11 05 00 19 6E 26 00 11 08 00 CD A9 C0 C9
D3C,50=02 03 0C 05 00 00 00 00 00 00 00 00 00 00 00 00
F3C,50=ED 66 03 00 05 00 E9 66 03 00 E9 66 02 00 E9 66
D33,45=DD E5 CD 59 67 C1 CD 40 67 18 05 00 00 00 00 00
F33,45=DD E5 CD BE 67 F1 7C B5 28 06 DD E5 CD A7 82 F1
D34,FA=DD E5 CD 59 67 C1:F34,FA=DD E5 CD BE 67 F1
D35,00=CD 40 67 18 05 00 00 00 00 00:F35,00=7C B5 28 06 DD E5 CD A7 82 F1
D5E,7B=0C:F5E,7B=03:D03,27=62:F03,27=61
. Eop ----------
. MC62C/FIX - correct #asm and +c combination - 08/30/86
. Apply via: PATCH MC2 MC62C
D73,3E=B7 28 16 2A D9 97 7C B5 20 0F 2A F2 93
F73,3E=6F 26 00 7C B5 28 06 2A D9 97 CD AD C1
. Eop ----------
. MCP51/FIX - Patch to MC - 10/23/86
. Patch corrects hex escape constants to accept at most 2 hex digits.
. Apply via: PATCH MCP MCP51
D3E,C4=3A AB 88 FE 78 20 4B 21 00 00 22 AB 88 22 AD 88
. was =2A AB 88 11 78 00 CD 7C C1 7C B5 28 45 21 00 00
D3E,D4=2A 4F 88 23 22 4F 88 3A AB 88 3C 32 AB 88 FE 03
. was =22 AD 88 2A 4F 88 23 22 4F 88 2A 4F 88 6E 26 00
D3E,E4=30 2C 2A 4F 88 6E 26 00 E5 CD 94 B6 F1 7C B5 28
. was =E5 CD 94 B6 F1 7C B5 28 25 21 04 00 ED 5B AD 88
D3E,F4=1D 2A AD 88 29 29 E5 2A 4F 88 23 22 4F 88 2B
. was =CD 61 C2 E5 2A 4F 88 23 22 4F 88 11 FF FF 19
D3F,11=C9:. was =CC
. Eop ----------
. MCP51/FIX - Patch to MC - 10/23/86 - Corrected 10/28/86
. Patch corrects hex escape constants to accept at most 2 hex digits.
. Apply via: PATCH MCP MCP51
D3E,C4=3A AB 88 FE 78 20 4B 21 00 00 22 AB 88 22 AD 88
. was =2A AB 88 11 78 00 CD 7C C1 7C B5 28 45 21 00 00
D3E,D4=2A 4F 88 23 22 4F 88 3A AB 88 3C 32 AB 88 FE 03
. was =22 AD 88 2A 4F 88 23 22 4F 88 2A 4F 88 6E 26 00
D3E,E4=30 2C 2A 4F 88 6E 26 00 E5 CD 94 B6 F1 7C B5 CD
. was =E5 CD 94 B6 F1 7C B5 28 25 21 04 00 ED 5B AD 88
D3E,F4=10 91 C8 29 29 29 29 E5 2A 4F 88 5E 16 00 23 22 4F 88 D5
```

```
. was =CD 61 C2 E5 2A 4F 88 23 22 4F 88 11 FF FF 19 6E 26 00 E5
D3F,11=C9:. was =CC
. Eop ----------
. MCP61/FIX - Patch to PRO-MC - 10/23/86
. Patch corrects hex escape constants to accept at most 2 hex digits.
. Apply via: PATCH MCP MCP61
D3E,DD=3A C8 5C FE 78 20 4B 21 00 00 22 C8 5C 22 CA 5C
F3E,DD=2A C8 5C 11 78 00 CD 49 95 7C B5 28 45 21 00 00
D3E,ED=2A 6C 5C 23 22 6C 5C 3A C8 5C 3C 32 C8 5C FE 03
F3E,ED=22 CA 5C 2A 6C 5C 23 22 6C 5C 2A 6C 5C 6E 26 00
D3E,FD=30 2C 2A 6C 5C 6E 26 00 E5 CD 67 8A F1 7C B5 28
F3E,FD=E5 CD 67 8A F1 7C B5 28 25 21 04 00 ED 5B CA 5C
D3F,0D=1D 2A CA 5C 29 29 E5 2A 6C 5C 23 22 6C 5C 2B
F3F,0D=CD 2E 96 E5 2A 6C 5C 23 22 6C 5C 11 FF FF 19
D3F,2A=C9:F3F,2A=CC
. Eop ----------
. MCP61/FIX - Patch to PRO-MC - 10/23/86 - Corrected 10/28/86
. Patch corrects hex escape constants to accept at most 2 hex digits.
. Apply via: PATCH MCP MCP61
D3E,DD=3A C8 5C FE 78 20 4B 21 00 00 22 C8 5C 22 CA 5C
F3E,DD=2A C8 5C 11 78 00 CD 49 95 7C B5 28 45 21 00 00
D3E,ED=2A 6C 5C 23 22 6C 5C 3A C8 5C 3C 32 C8 5C FE 03
F3E,ED=22 CA 5C 2A 6C 5C 23 22 6C 5C 2A 6C 5C 6E 26 00
D3E,FD=30 2C 2A 6C 5C 6E 26 00 E5 CD 67 8A F1 7C B5 CD
F3E,FD=E5 CD 67 8A F1 7C B5 28 25 21 04 00 ED 5B CA 5C
D3F,0D=2D 65 C8 29 29 29 29 E5 2A 6C 5C 5E 16 00 23 22 6C 5C D5
F3F,0D=CD 2E 96 E5 2A 6C 5C 23 22 6C 5C 11 FF FF 19 6E 26 00 E5
D3F,2A=C9:F3F,2A=CC
. Eop ----------
. MCTD51/FIX - 11/08/86
. Fixes TYPEDEF problem in MC:. Apply via: PATCH MC1 MCTD51
D14,C9=21 0E 00 39 E5 21 18:. WAS =21 0E 00 39 E5 21 06
D14,D0=00 39 CD AC E4 CD 8B E3 CD CC 6E 21 06 00 39 CD
. WAS =00 39 E5 21 1A 00 39 CD AC E4 CD 8B E3 E1 CD 93
D14,E0=93 E3 C5 D5 CD 47 6E F1 F1 D1 CD B1 E4 21 0C 00
. WAS =E3 C5 D5 CD 47 6E F1 F1 D1 CD B1 E4 21 0C 00 39
D14,F0=39 36 00 23 36 00 00:. WAS =11 00 00 EB CD B1 E4
D1D,33=F1 D1 C1 C5 D5 F5 CD CC 6E 2A 26 66 C9:.Was=F1 D1 C1 C5 D5 F5 CD 5E E2 7C B5 28 39
D1D,40=21 00 00 22 26 66 CD 5E E2 7C B5 C8 7B E6 03 32
. WAS =F1 D1 C1 C5 D5 F5 C5 D5 01 00 00 11 03 00 CD BB
D1D,50=26 66 C0 C5 D5 01 00 00 11 02 00 CD 18 E2 18 EC
. WAS =E1 EB 22 26 66 CD A4 E4 7C B5 28 19 21 02 00 39
D1D,60=00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
. WAS =E5 CD 8B E3 C5 D5 01 00 00 11 02 00 CD 18 E2 E1
D1D,70=00 00 00 00:. WAS =CD 93 E3 18:D1D,78=00 00 00 00:. WAS =CB 2A 26 66
. Eop ----------
. MCTD52/FIX - 11/08/86
. Fixes TYPEDEF problem in MC:. Apply via: PATCH MC2 MCTD52
D12,1F=21:. WAS =21
D12,20=0E 00 39 E5 21 18 00 39 CD F4 E5 CD D3 E4 CD 45
. WAS =0E 00 39 E5 21 06 00 39 E5 21 1A 00 39 CD F4 E5
D12,30=6A 21 06 00 39 CD DB E4 C5 D5 CD C0 69 F1 F1 D1
. WAS =CD D3 E4 E1 CD DB E4 C5 D5 CD C0 69 F1 F1 D1 CD
D12,40=CD F9 E5 21 0C 00 39 36:. WAS =F9 E5 21 0C 00 39 11 00
D12,4C=00 23 36 00 00:. WAS =00 EB CD F9 E5:D18,9C=F1 D1 C1 C5:. WAS =F1 D1 C1 C5
D18,A0=D5 F5 CD 45 6A 2A A1 63 C9 21 00 00 22 A1 63 CD
. WAS =D5 F5 CD A6 E3 7C B5 28 39 F1 D1 C1 C5 D5 F5 C5
```

```
D18,B0=A6 E3 7C B5 C8 7B E6 03 32 A1 63 C0 C5 D5 01 00
. WAS =D5 01 00 00 11 03 00 CD 03 E3 EB 22 A1 63 CD EC
D18,C0=00 11 02 00 CD 60 E3 18 EC:. WAS =E5 7C B5 28 19 21 02 00 39
D18,C9=00 00 00 00 00 00 00:. WAS =E5 CD D3 E4 C5 D5 01
D18,D0=00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
. WAS =00 00 11 02 00 CD 60 E3 E1 CD DB E4 18 CB 2A A1
D18,E0=00 00 00 00 00 00:. WAS =63 C9 21 00 00 C9
. Eop ----------
. MCTD60/FIX - 11/08/86
. Fixes TYPEDEF problem in PRO-MC:. Apply via: PATCH MC MCTD60
D15,ED=21 0E 00:F15,ED=21 0E 00
D15,F0=39 E5 21 18 00 39 CD E1 DB CD C0 DA CD EC 43 21
F15,F0=39 E5 21 06 00 39 E5 21 1A 00 39 CD E1 DB CD C0
D16,00=06 00 39 CD C8 DA C5 D5 CD 67 43 F1 F1 D1 CD E6
F16,00=DA E1 CD C8 DA C5 D5 CD 67 43 F1 F1 D1 CD E6 DB
D16,10=DB 21 0C 00 39 36 00 23 36 00 00:F16,10=21 0C 00 39 11 00 00 EB CD E6 DB
D1E,57=F1 D1 C1 C5 D5 F5 CD EC 43:F1E,57=F1 D1 C1 C5 D5 F5 CD 93 D9
D1E,60=2A 46 3B C9 21 00 00 22 46 3B CD 93 D9 7C B5 C8
F1E,60=7C B5 28 39 F1 D1 C1 C5 D5 F5 C5 D5 01 00 00 11
D1E,70=7B E6 03 32 46 3B C0 C5:F1E,70=03 00 CD F0 D8 EB 22 46
D1E,7C=D5 01 00 00:F1E,7C=3B CD D9 DB
D1E,80=11 02 00 CD 4D D9 18 EC:F1E,80=7C B5 28 19 21 02 00 39
D1E,88=00 00 00 00 00 00 00 00:F1E,88=E5 CD C0 DA C5 D5 01 00
D1E,90=00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
F1E,90=00 11 02 00 CD 4D D9 E1 CD C8 DA 18 CB 2A 46 3B
D1E,A0=00 00 00 00 00:F1E,A0=C9 21 00 00 C9
. Eop ----------
. MCTD61/FIX - 11/08/86
. Fixes TYPEDEF problem in PRO-MC:. Apply via: PATCH MC1 MCTD61
D14,B8=21 0E 00 39 E5 21 18 00:F14,B8=21 0E 00 39 E5 21 06 00
D14,C0=39 CD 5C B8 CD 3B B7 CD BB 42 21 06 00 39 CD 43
F14,C0=39 E5 21 1A 00 39 CD 5C B8 CD 3B B7 E1 CD 43 B7
D14,D0=B7 C5 D5 CD 36 42 F1 F1 D1 CD 61 B8 21 0C 00 39
F14,D0=C5 D5 CD 36 42 F1 F1 D1 CD 61 B8 21 0C 00 39 11
D14,E0=36 00 23 36 00 00:F14,E0=00 00 EB CD 61 B8
D1D,22=F1 D1 C1 C5 D5 F5 CD BB 42 2A 15 3A C9 21
F1D,22=F1 D1 C1 C5 D5 F5 CD 0E B6 7C B5 28 39 F1
D1D,30=00 00 22 15 3A CD 0E B6 7C B5 C8 7B E6 03 32 15
F1D,30=D1 C1 C5 D5 F5 C5 D5 01 00 00 11 03 00 CD 6B B5
D1D,40=3A C0 C5 D5 01 00 00 11 02 00 CD C8 B5 18 EC
F1D,40=EB 22 15 3A CD 54 B8 7C B5 28 19 21 02 00 39:D1D,4F=00:F1D,4F=E5
D1D,50=00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
F1D,50=CD 3B B7 C5 D5 01 00 00 11 02 00 CD C8 B5 E1 CD
D1D,60=00 00 00 00 00 00 00 00 00 00 00 00:F1D,60=43 B7 18 CB 2A 15 3A C9 21 00 00 C9
. Eop ----------
. MCTD62/FIX - 11/08/86:. Fixes TYPEDEF problem in PRO-MC:. Apply via: PATCH MC2 MCTD62
D12,3A=21 0E 00 39 E5 21:F12,3A=21 0E 00 39 E5 21
D12,40=18 00 39 CD B5 C1 CD 94:F12,40=06 00 39 E5 21 1A 00 39
D12,4C=C0 CD 60 3E:F12,4C=CD B5 C1 CD
D12,50=21 06 00 39 CD 9C C0 C5 D5 CD DB 3D F1 F1 D1 CD
F12,50=94 C0 E1 CD 9C C0 C5 D5 CD DB 3D F1 F1 D1 CD BA
D12,60=BA C1 21 0C 00 39 36 00 23 36 00 00:F12,60=C1 21 0C 00 39 11 00 00 EB CD BA C1
D18,B7=F1 D1 C1 C5 D5 F5 CD 60 3E:F18,B7=F1 D1 C1 C5 D5 F5 CD 67 BF
D18,C0=2A BC 37 C9 21 00 00 22 BC 37 CD 67 BF 7C B5 C8
F18,C0=7C B5 28 39 F1 D1 C1 C5 D5 F5 C5 D5 01 00 00 11
D18,D0=7B E6 03 32 BC 37 C0 C5 D5 01 00 00 11 02 00 CD
F18,D0=03 00 CD C4 BE EB 22 BC 37 CD AD C1 7C B5 28 19
```

```
D18,E0=21 BF 18 EC:F18,E0=21 02 00 39
D18,E4=00 00 00 00 00 00 00 00 00 00 00 00:F18,E4=E5 CD 94 C0 C5 D5 01 00 00 11 02 00
D18,F0=00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
F18,F0=CD 21 BF E1 CD 9C C0 18 CB 2A BC 37 C9 21 00 00
D19,00=00:F19,00=C9
. Eop ----------
. MLK510/FIX - Patch to MRAS - 10/20/86
. Patch corrects one problem with COMMONs, with -V switch, and PDCL
. switches where value is > 9FFFH and value is in 1/c.:. Apply via: PATCH MLINK MLK510
D03,DD=00 00 00:. was =C2 8D 66:D06,86=2A 6A:. was =18 56:D06,38=2C 6A:. was =13 69
D08,5F=CD 39 6A:. was=6F 87 85:D09,13=7D:. was=4D:D0F,CA=DA 47 6A 00:. was =30 02 D6 07
D11,E7=00 00 FE 03 C2 13 69 2A 2A 6A 16 00 C3 16 69 FE
. was =00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
D11,F7=03 20 06 2A 2A 6A 22 18 56 6F 87 85 C9 CD 88 5A
. was =00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
D12,07=38 04 D6 20 D6 07 C3 19 68:. was =00 00 00 00 00 00 00 00 00
. Eop ----------
. MLK59/FIX - 09/02/86 - Patch to MRAS's MLINK
. Corrects -p, -d, & -c switches:. Apply via: PATCH MLINK MLK59
D00,F8=04:. was =0C:D03,EB=CD 1C 6A 21 0C 56 85 6F:. was =21 10 56 85 6F 8C 95 67
D06,86=18:. was =1A:D06,9C=14 6A:. was =04 56:D07,82=47 07 07:. was =07 07 47
D07,9C=CD 1E 6A 21 0C 56 85 6F:. was =21 10 56 85 6F 8C 95 67
D08,5F=6F 87 85 87 21 04 56 85 6F 00:. was =87 87 21 0C 56 85 6F 8C 95 67
D08,7A=04:. was =0C:. was =CD 24 6A:. was =2B ED 42:. was =14 6A:. was =04 56
D10,C0=CD 1E 6A 21 04:. was =87 3C 87 21 0C:D10,D8=CD 1E 6A 21 04:. was =87 3C 87 21 0C
D11,D9=0F 0F 6F 87 85 3C 87 C9 ED 42 3F C8 3F C9
. was =00 00 00 00 00 00 00 00 00 00 00 00 00 00: D13,51="b":. was ="a"
. Eop ----------
. MLK68/FIX - 09/02/86 - Patch to PRO-MRAS's MLINK
. Corrects -p, -d, & -c switches:. Apply via: PATCH MLINK MLK68
D00,E4=B4:F00,E4=BC:D03,EE=CD 58 3D 21 BC 29 85 6F:F03,EE=21 C0 29 85 6F 8C 95 67
D06,96=C8:F06,96=CA:D06,A8=50 3D:F06,A8=B4 29:D07,92=47 07 07:F07,92=07 07 47
D07,A8=CD 5A 3D 21 BC 29 85 6F:F07,A8=21 C0 29 85 6F 8C 95 67
D08,6B=6F 87 85 87 21 B4 29 85 6F 00:F08,6B=87 87 21 BC 29 85 6F 8C 95 67
D08,8A=B4:F08,8A=BC:D0A,83=CD 60 01 02 00 36 3D:F0A,83=2B ED 01 02 00 36 42
D0E,DA=50 3D:F0E,DA=B4 29:D10,CC=CD 5A 3D 21 B4:F10,CC=87 3C 87 21 BC
D10,E4=CD 5A 3D 21 B4:F10,E4=87 3C 87 21 BC
D11,FD=0F 0F 6F 87 85 3C 87 C9 ED 42 3F C8 3F C9
F11,FD=00 00 00 00 00 00 00 00 00 00 00 00 00 00
D13,80="b":F13,80="a"
. Eop ----------
. MLK69/FIX - Patch to PRO-MRAS - 10/20/86
. Patch corrects one problem with COMMONs and PDCL switches where value
. is > 9FFFH and value is in 1/c. Apply via: PATCH MLINK MLK69
D03,E0=00 00 00:F03,E0=C2 B1 39:D06,96=66 3D:F06,96=C8 29:D06,44=68 3D:F06,44=37 3C
D08,6B=CD 75 3D:F08,6B=6F 87 85:D0F,D6=DA 83 3D 00:F0F,D6=30 02 D6 07
D12,0B=00 00 FE 03 C2 37 3C 2A 66 3D 16 00 C3 3A 3C FE
F12,0B=00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
D12,1B=03 20 06 2A 66 3D 22 C8 29 6F 87 85 C9 CD 8C 2D
F12,1B=00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
D12,2B=38 04 D6 20 D6 07 C3 3D 3B:F12,2B=00 00 00 00 00 00 00 00 00
. Eop ----------
. MRS610/FIX - Patch to MRAS - 10/20/86
. Patch corrects a number of problems with COMMONs:. Apply via: PATCH MRAS MRS510
D02,46=75 8F:F02,46=01 8F:D1E,9C=CD 5D 8F:F1E,9C=22 2B 58:D1E,A7=66 8F:F1E,A7=11 5C
D1E,EE=53 8F:F1E,EE=DC 78:D20,67=CD 1C 8F:F20,67=E5 E6 03:D20,97=48 8F:F20,97=11 5C
X'8F1C'=E3 E5 47 E6 03 FE 03 C0 F5 C5 78 E6 1C 0F 0F 47
```

```
X'8F2C'=3A D4 67 B8 28 13 78 32 D4 67 87 21 D5 67 85 6F
X'8F3C'=8C 95 67 CD C6 6F CD 43 79 C1 F1 C9 E1 C1 C5 E5
X'8F4C'=78 CD 1F 8F C3 11 5C CD DC 78 3A D4 67 32 67 8F
X'8F5C'=C9 22 2B 58 3E FF 32 67 8F C9 3E FF 3C 28 07 3D
X'8F6C'=07 07 47 CD 24 8F C3 11 5C
. Eop ----------
. MRS511/FIX - Patch to MRAS - 10/20/86
. Patch changes relocation reference error to warning:. Apply via: PATCH MRAS MRS511
D0B,79=0F:F0B,79=0A
. Eop ----------
. MRS57/FIX - Patch to MRAS - 08/13/86
. Patch corrects unnecessary page eject if -LP & assembly errors
. Apply via: PATCH MRAS MRS57
D10,CA=00 7C B5 28 04 AF:. was =28 07 7C B5 28 03
. Eop ----------
. MRS58/FIX - Patch to MRAS - 08/13/86
. Patch corrects NAME arg where arg > 7 chars long:. Apply via: PATCH MRAS MRS58
D20,DF=CD B7 8A:. was =21 71 58
D31,15=4E 6F 74 20 61 20 50 44 53 A1:. WAS =53 65 61 72 63 68 20 66 69 6C
D31,1F=21 71 58 3E 07 B9 D0 4F C9 00 00 00 00 00 00 00
. WAS =65 20 69 73 20 6E 6F 74 20 61 20 50 44 53 21 0D
. Eop ----------
. MRS59/FIX - Patch to MRAS - 10/16/86
. Patch corrects -NH switch entered on command line:. Apply via: PATCH MRAS MRS59
D0E,56=62:F0E,56=63
D0E,A3=15 FE 2D 20 DE CD 66 61 21 78 67 06 01 CD 9A 62
F0E,A3=16 FE 2D C2 95 5F CD 66 61 21 78 67 06 01 CD 9A
D0E,B3=DA 95 5F 70 18 E5 3A 9F 67 B7 28 03:F0E,B3=62 DA 95 5F 70 18 E5 3A B3 67 EE 01
D11,DA=32 47 58 3A AF 67 3D CA CD 6A 21 0A 6C E5 3A B3
F11,DA=AF 32 47 58 3A AF 67 3D CA CD 6A 3A B3 67 B7 CC
D11,EA=67 B7 CA 2C 87 3A A3 67 B7 CA:F11,EA=2C 87 3A A3 67 21 9F 67 B6 CC
. Eop ----------
. MRS610/FIX - Patch to PRO-MRAS - 10/20/86
. Patch corrects a number of problems with COMMONs:. Apply via: PATCH MRAS MRS610
D02,2A=60 62:F02,2A=EC 61:D1E,5F=CD 48 62:F1E,5F=22 DB 2B:D1E,6A=51 62:F1E,6A=32 2F
D1E,B1=3E 62:F1E,B1=D8 4B:D20,2A=CD 07 62:F20,2A=E5 E6 03:D20,5A=33 62:F20,5A=32 2F
X'6207'=E3 E5 47 E6 03 FE 03 C0 F5 C5 78 E6 1C 0F 0F 47
X'6217'=3A CB 3A B8 28 13 78 32 CB 3A 87 21 CC 3A 85 6F
X'6227'=8C 95 67 CD C2 42 CD 3F 4C C1 F1 C9 E1 C1 C5 E5
X'6237'=78 CD 0A 62 C3 32 2F CD D8 4B 3A CB 3A 32 52 62
X'6247'=C9 22 DB 2B 3E FF 32 52 62 C9 3E FF 3C 28 07 3D
X'6257'=07 07 47 CD 0F 62 C3 32 2F
. Eop ----------
. MRS611/FIX - Patch to PRO-MRAS - 10/20/86
. Patch changes relocation reference error to warning:. Apply via: PATCH MRAS MRS611
D0B,37=06:F0B,37=01
. Eop ----------
. MRS67/FIX - Patch to PRO-MRAS - 08/13/86
. Patch corrects unnecessary page eject if -LP & assembly errors
. Apply via: PATCH MRAS MRS67
D10,8D=00 7C B5 28 04 AF:F10,8D=28 07 7C B5 28 03
. Eop ----------
. MRS68/FIX - Patch to PRO-MRAS - 08/13/86
. Patch corrects NAME arg where arg > 7 chars long
. Apply via: PATCH MRAS MRS68
D20,A2=CD A2 5D:F20,A2=21 21 2C
D30,C7=4E 6F 74 20 61 20 50 44 53 A1:F30,C7=53 65 61 72 63 68 20 66 69 6C
```

```
D30,D1=21 21 2C 3E 07 B9 D0 4F C9 00 00 00 00 00 00 00
F30,D1=65 20 69 73 20 6E 6F 74 20 61 20 50 44 53 21 0D
. Eop ----------
. MRS69/FIX - Patch to PRO-MRAS - 10/16/86
. Patch corrects -NH switch entered on command line:. Apply via: PATCH MRAS MRS69
D0E,14=62:F0E,14=63
D0E,61=15 FE 2D 20 DE CD 5D 34 21 6F 3A 06 01 CD 91 35
F0E,61=16 FE 2D C2 A9 32 CD 5D 34 21 6F 3A 06 01 CD 91
D0E,71=DA A9 32 70 18 E6 3A 96 3A B7 28 03:F0E,71=35 DA A9 32 70 18 E5 3A AA 3A EE 01
D11,9D=32 F7 2B 3A A6 3A 3D CA C4 3D 21 06 3F E5 3A AA
F11,9D=AF 32 F7 2B 3A A6 3A 3D CA C4 3D 3A AA 3A B7 CC
D11,AD=3A B7 CA 10 5A 3A 9A 3A B7 CA:F11,AD=10 5A 3A 9A 3A 21 96 3A B6 CC
. Eop ----------
. MSD65/FIX - Patch to PRO-SAID V1.1 - 10/24/86
. Patch corrects crash when META-EXTERNAL is selected and
. no external memory banks are available.:. Apply via: PATCH SAID MSD65
D14,E6=47 0E 31 DD 21 9C 4D E1 C8:F14,E6=C8 47 0E 31 DD 21 9C 4D E1
. Eop ----------
. S61/FIX - 09/03/86 - Patch to PRO-EnhComp S/CMD
. Corrects closing of CED, BC, and TEMP files:. Apply via: PATCH S S61
D00,7B=8F 27:F00,7B=F5 26:D01,9B=CD F5 26 C3 52 27:F01,9B=00 00 00 00 00 00
. Eop ----------
. SDAT51/FIX - 09/03/86 - Patch to EnhComp SUPPORT/DAT
. Apply via: PATCH SUPPORT/DAT SDAT51:. Correct the ^ operator [raise x to power y]
D88,84=04 28 27 30 3A:. WAS =08 28 3C 28 25
. NOTE: The following patch is optional to change the
. scientific notation specifier in USING from ^^^^ to
. 4 UP ARROWS "]]]]". Remove the "." to install
.D44,C5=5B 5B 5B 5B:.  WAS =5E 5E 5E 5E
. Eop ----------
. SDAT52/FIX - Patch to Model I/III EnhComp
. Apply via: PATCH SUPPORT/DAT SDAT52
. Corrects a bug in SDAT51/FIX and corrects LOF().
D49,2E=C2:. WAS =D2:D88,2A=4D 0D 80 65 00:. WAS =04 28 27 30 3A
D88,84=04 28 27 30 3A:. WAS =08 28 3C 28 35
. Eop ----------
. SDAT61/FIX - 09/03/86 - Patch to PRO-EnhComp SUPPORT/DAT
. Apply via: PATCH SUPPORT/DAT SDAT61
. Correct USING string to support a length up to 79 chars
D02,B4=8A:F02,B4=7A:D72,FE=50:F72,FE=40:D73,2A=50:F73,2A=40
. Correct the ^ operator [raise x to power y]
D88,2A=04 28 27 30 3A:F88,2A=08 28 3C 28 25:D02,B4=8A:F02,B4=7A
. Eop ----------
. BOOTAT/FIX - 07/18/85
. Copyright 1985 Roy Soltoff, All rights reserved
. Patch to BOOT/SYS (TRSDOS 6.2/6.3) for Alpha Tech board
. Apply via: PATCH BOOT/SYS.LSIDOS BOOTAT (O=N)
. Patch low memory pointer at 206H:D00,06=19 10
. Patch byte I/O handler at 65EH:D04,5E=88
. Patch byte I/O handler at 67DH:D04,7D=88
. Patch KCK@ routine at 7F6H:D05,F6=2F
. Patch KCK@ routine at 7F9H:D05,F9=81:D05,FB=57
. Patch ENADIS_DO_RAM and @BANK routines at 817H
D06,17=CD 4A 08 21 03 0C 39 30 09 E1 ED 73 74 08 31 40
D06,27=03 E5 21 55 08 E3 E5 21 7F 08 23 C5 0E 43 ED 40
D06,37=23 70 47 3A 78 00 30 02 E6 7F 77 E6 FC F6 82 ED
D06,47=41 18 1C F3 22 7C 08 F5 E1 22 77 08 AF C9 CD 4A
```

```
D06,57=08 2A 2F 08 7E CB 7F CB FF 2B C5 0E 43 46 2B 22
D06,67=2F 08 32 78 00 D3 84 ED 41 C1 20 03 31 00 00 21
D06,77=00 00 E5 F1 21 00 00 FB C9 00 00 00 00 00 00 00
D06,87=00 E6 7F FE 1F 30 15 05 FA CB 08 0E 86 28 1D 0E
D06,97=46 05 28 18 05 28 0E 05 3A 02 02 C8 C3 ED 0D E5
D06,A7=21 14 04 18 0B CD B3 08 C0 78 0E C6 E5 21 10 04
D06,B7=CD 10 10 78 E6 07 07 07 07 B1 32 C8 08 AF 3E 08
D06,C7=CB 46 E1 C9 E5 21 05 80 39 E1 38 D0 C5 0E 46 CD
D06,D7=A6 08 78 C1 20 C6 CD F4 0F E6 7F 32 02 02 A9 B0
D06,E7=4F CB 79 06 00 C8 E3 BF C9
. Patch video driver at 0C8FH:D0A,8F=2F
. Change pointer in FDC driver to account for patch at 0E3FH:D0C,3F=18 10
. Add code at end of FDC driver for @BANK and TASKER at 0FF4H
D0D,F4=B7 28 01 3C D3 43 3A 02 02 47 79 C9 21 77 00 CB
D0E,04=F6 21 02 02 C9 32 02 02 21 77 00 C9 47 0F 0F 0F
D0E,14=E6 03 85 6F C9
. Eop ----------
. SYS0AT/FIX - 09/10/85
. Copyright 1985 Roy Soltoff, All rights reserved
. Patch to SYS0/SYS (TRSDOS 6.2) for Alpha Tech board
. Apply via: PATCH SYS0/SYS.LSIDOS SYS0AT (O=N)
. Patch the SVC table at 1CCH:D08,6E=88
. Patch the disk I/O handler at 19FDH:D09,5B=88
. Patch the TASKER at 1C06H:D0B,8C=CD 00 10 7E F5 DB 43 F5 AF 77 D3 43
. Patch the TASKER at 1C3DH:D0B,C3=D3 43 F1 CD 09 10 CB B6
. Patch the TASKER at 1C91H:D0C,17=F1 E3
. Patch system initialization routines at 1E3FH
D0C,CD=D3 84 26 FF 56 01 43 1F 71 C5 48 0D 06 03 3E 66
D0C,DD=EF C1 ED 41 5E 34 00 00 00 7E BB 00 00 00 73 28
D0C,ED=09 C5 48 0D 06 01 3E 66 EF C1 10 DD CD 47 21
. Patch system initialization routines at 2147H
D0F,E1=ED 41 72 6C 22 0E 04 22 1C 00 21 10 04 CB 86 11
D0F,F1=14 04 0E 04 ED B0 3A 6F 00 C9
. Eop ----------
. SYS0AT63/FIX - 10/28/86
. Copyright 1985,86 Roy Soltoff, All rights reserved
. Patch to SYS0/SYS (TRSDOS 6.3) for Alpha Tech board
. Apply via: PATCH SYS0/SYS.LSIDOS SYS0AT (O=N)
. Patch the SVC table at 1CCH:D08,6E=88
. Patch the disk I/O handler at 19FDH:D09,5B=88
. Patch the TASKER at 1C06H:D0B,8C=CD 00 10 7E F5 DB 43 F5 AF 77 D3 43
. Patch the TASKER at 1C3DH:D0B,C3=D3 43 F1 CD 09 10 CB B6
. Patch the TASKER at 1C91H:D0C,17=F1 E3
. Patch system initialization routines at 1E48H
D0C,D6=D3 84 26 FF 56 01 43 1F 71 C5 48 0D 06 03 3E 66
D0C,E6=EF C1 ED 41 5E 34 7E BB 73 28 09 C5 48 0D 06 01
D0C,F6=3E 66 EF C1 10 E3 ED 41 72 CD 94 21
. Patch system initialization routines at 2194H
D10,2E=6C 22 0E 04 22 1C 00 21 10 04 CB 86 11 14 04 0E:D10,3E=04 ED B0 C9
. Eop ----------
. SYS1AT/FIX - 07/17/85
. Copyright 1985 Roy Soltoff, All rights reserved
. Patch to SYS1/SYS (TRSDOS 6.2/6.3) for Alpha Tech board
. Apply via: PATCH SYS1/SYS.LSIDOS SYS1AT
. Patch cleanup routine at 1E77H and 1E7AH:D00,9C=7F:D00,9F=2F
. Patch restoral of bank_0 at 1EC0H:D00,E5=88
. Eop ----------
```

MISOSYS, Inc.
P. O. Box 239
Sterling, VA 22170-0239

Contents: Printed Matter