THE MISOSYS QUARTERLY

In this issue:

- UNDATE reverses DATECONV by L M Garcia-Barro
- 80x86 assembly language by Phil Oliver
- Converting LDOS filters to LS-DOS
- Previewing output from SCRIPSIT
- Announcing PRO-WAM Release 2
- Learn MORSE with CODE/BAS

Volume I, Issue iv Spring 1987 $10

THE MISOSYS QUARTERLY

Volume I, Issue iv

Spring 1987

Table of Contents

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

The Blurb

Speaking about The Quarterly...

Well folks, all kidding aside, this is the Spring 1987 issue of THE MISOSYS QUARTERLY. I know, many of you are boiling in the hot sun by the pool. Some of you are stocking up on cord wood (see "Letters"). Actually, it should get to you about three months since the last issue. Unfortunately, I haven't been able to compress the time between issues to catch up on the timing of the QUARTERLY. I do hope to shorten the time until the next issue by a few weeks so you're not gethering in the pumkins while you read the Summer issue. Rest assured, regardless of the timing, a "year's" worth of subscriptions will get you four issues.

This Spring 1987 issue marks the completion of Volume I; next issue starts Volume II. Sometime in the near future, we will be mailing out renewal notices to those of you who have subscribed for the first issue (i.e. if your subscription gets expired after this I.iv issue). Please wait for your statement; we are going to process renewals in only a few inputs to our order entry system. That's to avoid wasting Brenda's time. All we really have to get accomplished is to alter your expiration field in the data base and reflect the total dollar value of renewal transactions in the invoice system (monthly revenue figures here are processed electronically).

As I write this Blurb (June 18th), we have no I.i's left, we have 15 I.ii's left, and 105 I.iii's left. Since our last print run was 800, that gives us about 700 subscribers. The actual count in the data base is 717 out of 18066; this means we have over eighteen thousand names in our database - 717 are subscribers of THE MISOSYS QUARTERLY. TMQ I.iii was mailed to 658 subscribers; thus, we have picked up about 59 new readers since our last issue. The rate of renewals will let us know if we're doing a good job with TMQ.

This is another full issue. In fact, again we had to defer some material. I had eight pages of "The Hardware Corner" which must wait until next issue. I also had to forgo "Our Compuserve Forum". Rich's stuff on stat() and rm are likewise postponed. Some of the correspondance which I had selected for QUARTERLY insertion also is postponed. But that's good, it means I have a partial box full of material for starting Volume II. I'll still welcome your input, though. The more contributions I get from readers, the less time I have to spend working up articles, and the more time I can spend on programming.

Now speaking of TMQ, we felt that the content was so valuable, we started paying for address corrections. If you're a First Class Mail subscriber, you may have noticed the phrase, "Forwarding and Address Correction Requested". If you have moved, you'll still get your QUARTERLY but then we have to pay $0.40 to find out that you moved. For Bulk Mail subscribers, the phrase, "Address Correction Requested, Forwarding and Return Postage Guaranteed" will cost us a lot more. To begin with, forwarding is handled according to First Class rates and we still have to pay the address correction fee. Now here is the kicker! Although your change of address filed with the Post Office is supposed to be good for one year, we have been informed that the COA is valid only for THREE MONTHS for magazines! We have gotten a few back already noting "Forwarding time expired". This hits on us for anywhere from $1.18 to over $3.00 (which Brenda refused to pay!). So please, if you move, don't forget your subscriptions and get us a change of address.

The PROGRAMMER'S GUIDE now available

We recently acquired a two-color high quality copier. This gives us the capability of producing in-house small quantities of printed matter. We have decided to use this facility to make available once again THE PROGRAMMER'S GUIDE TO LDOS/TRSDOS VERSION 6. Also, to assist in the "binding" operation, we acquired a GBC "Image-Maker 2000". This is the little machine which binds documents with those plastic finger thingamajigs. We've produced a few GUIDES already and I like the result. The binding allows pages to turn freely, lets the book lie flat when open, and is not much of a burden to install. So if you are still looking for a copy of this valuable book, we got it. The price, of course, has gone up from when we used to get it offset printed. As I stated in TMQ I.i (The Blurb), offset printing of "The Guide" was only economical when the print run was at least 500. At a print run of one per, we will be selling The Guide for $25 plus $3 S&H (US).

Projects at MISOSYS, Inc.

Last issue I reported on the imminence of three products: RATFOR-M4, RATFOR-86, and DSM86. Well the first two made it out. Unfortunately, the free-lancer doing our port of DSM4 over to MS-DOS got tied up with other work forcing us to bring DSM86 in-house. It

will be delayed since there are a few other projects going on.

This issue announces PRO-WAM Release 2. By the time you read this, it will have been in beta test for about two months getting a good shakedown. Also, as soon as I put the wraps on this QUARTERLY, I'll start the massive rewrite to the PRO-WAM manual. I am quite proud of this new release; it has virtually everything that our users have been requesting plus lots more. Of course, due to the state of the 8-bit marketplace, I think this will be the last release of PRO-WAM. That's why we wanted to put so much into it.

With the release of new 8-bit chips (the Hitachi HD64180 and Zilog Z280), we want to explore the feasibility of revamping MRAS to support the extended instructions of these two processors. We may be able to report something next issue.

We also want to spend some time later this year grooming Little Brother (LB and LB86). I have a personal commitment to support a co-existance of LB and PRO-WAM. Currently, LB cannot run with the reduced memory available when PRO-WAM is loaded. That may be due to the file buffering requirements imposed by the Aztec C compiler used to compile the Model 4 version of LB. We hope to explore a redo of the LB source to compile with MC.

From time to time we get program submissions for consideration as potential MISOSYS products. Currently under evaluation is a new interactive game targeted for the Model 4. This MegAdventure package titled "LAIR OF THE DRAGON" has been submitted by David Goben. Preliminary feedback from my evaluators reveals great potential for this package. I'll probably have more to report in TMQ II.i as some decisions will be made during the next few months. If we go ahead with this package, an MS-DOS version will be developed, as well.

I am looking over a fast backup and restore utility for MS-DOS. Even though the market is crowded with such utilities, there's room for more. When you move into a crowded market, the trick is to make your product a little bit better than the rest.

Which brings up the topic of program submissions. Yes, MISOSYS does publish products which have originated from independent authors (like many of our readers are). After all, when MISOSYS started out nine years ago, we were an independent - and we still are, we're just better known due to perseverance and a quest for doing the right thing. We are always open to program submissions. But please, I don't think we need any more Model III software. Even for a Model 4, I have my doubts as to the viability of marketing a new Model 4 program. On the other hand, a program running on a 4 which is easily ported to MS-DOS is another issue. At this point, we have not considered addressing the Apple, Atari, or Amiga markets (nor UNIX, either). So products for those markets won't be viable here. If you have something you think we should see, either call or write first - no need to send your package. Our primary consideration is for products in the MS-DOS arena.

Focus on Public Domain

We continue to get requests from folks (primarily overseas) wanting to obtain files which have been placed into the download section of our Compuserve forum. Since we have had no takers for our plea of assistance in this regard, I am forced to take the bull by the horns. Note that my function as forum manager gives me no right to download files en masse and begin distributing them elsewhere. I also have no intentions to selectively obtain files and then make them available elsewhere. MISOSYS has no time to perform a service of distributing public domain software files. On the other hand, we could probably bend to organize disks of PD software to be made available for a fee similar to what Montezuma Micro is doing for CP/M programs.

Considering these issues, we will start to build a library of public domain software for the TRS-80. The programs must be submitted via floppy disk directly to us by mail from the original author; we will not take submissions via Compuserve. Please use LDOS or LS-DOS formats (40 cylinder, DDEN, 1 or 2 sides). We expect pricing per disk to be similar to DISK NOTES, although we will listen to your input on pricing. Our next issue will have the results of your input.

No Model I LDOS 5.3

Our last issue of TMQ anticipated the decision recently made that there **WILL NOT** be a 5.3 release of LDOS for the Model I. We are extremely disappointed with the sales volume of the Model III LDOS 5.3 Upgrade Kit. Extrapolating potential Model I sales results based on the ratio of actual Model III sales to the entire Model III/4 marketplace precludes any justification for a Model I release of 5.3. However, before the end of the

year we will be making public a set of patches to accomodate Model I LDOS users. These patches will de-activate the Directory file dating and extend the DATE$ dating to accept dates through 1999.

Surplus hardware for sale

Brenda asked me not long ago, "When are you going to get rid of all that extra hardware stored in the back room?" I'm not sure she used the term "hardware", if you get my drift. When MISOSYS picked up the operation LSC used to run out in Milwaukee, we acquired what can be best called an excess of computer equipment (not to mention an excess of office furniture). George Carlin refers to this as "stuff". Brenda's right, it's time to see if anyone has a need for some of this stuff - at a price, of course. Here's a list of what we sorted out so far as extra. Prices shown include ground shipment via UPS. Sorry, but this stuff can only be shipped to the continental US only. Everything is used and sold as is. All equipment was in working order when removed from service.

Qty	Item	Price
4	TEC FB-202 40T floppy	$25
3	Texas Peripherals floppy	$25
1	Shugart Assoc 400 floppy	$25
2	BASF 6106 2/3 ht floppy	$25
1	Percom 20ms floppy, "jaws"	$25
2	MPI B-51 floppy	$25
7	floppy case & power supply	$10
1	MAX-80, 128K keyboard	$200
1	BMC Monitor, BM-12A	$40
1	16K Model I complete, 1/c	$100
1	32K Model I E/I	$50

All floppies noted above are bare drive. We have other stuff around here. Like a Line Printer III, an MX-100, a DP-800, some 5-meg hard drive bubbles, a Lobo WIN5 5meg drive, etc. We also have a ton of 8" floppies - used.

MISOSYS closed in August

Here's an important announcement. No, we won't be closed the month of August. This business doesn't pay THAT well. It just happens to be time for the Soltoffs to enjoy a week off down at the beach. Actually, I should say a week at the shore since I come from Philly and that's what they say there. We will be closed from August 8th through August 16th. That's the second full week in August including surrounding weekends. We expect to be somewhere in Virginia Beach, depending on our guess as to what constitutes an ideal spot for the family. No, we won't be there the whole time, but don't count on us answering our phone. And don't worry if mail orders are delayed - we'll catch up when we get back. Summertime has been a little slow, anyway.

Contents of DISK NOTES 8

Each issue of THE MISOSYS QUARTERLY contains program listings, patch listings, and other references to files we have placed on DISK NOTES. Some people enjoy typing in long listings. Sometimes you may have need for only a short patch. If you want to obtain all of the patches and all of the listings, you may conveniently purchase a copy of DISK NOTES.

There is a cost involved. DISK NOTES is priced at $10 PLUS S&H. The S&H charges are $2 for US, $3 Canada and Mexico, $6 elsewhere. If you purchase the corresponding DISK NOTES with the coupon which accompanies this TMQ issue, you can save $2.50; the cost then being only $7.50 + S&H. Here's what's on Disk NOTES 8:

```
LSIFIX/TXT   - Fixes for LS-DOS 6.3
INSTRUCT/TXT - See APPLICATIONS
LDOSINFO/DAT - ditto
LDOSINFO/TXT - ditto
LIBENTRY/BAS - ditto
LIBENTRY/TBA - ditto
LIBREAD/BAS  - ditto
LIBREAD/TBA  - ditto
UNDATE/CMD   - opposite of DATECONV
PGPAWS5/ASM  - source to PAGEPAWS, v5
UNDATE/SRC   - source to UNDATE/CMD
PGPAWS6/ASM  - source to PAGEPAWS, v6
UNDATE/DOC   - docs for UNDATE
MPATCH/FIX   - Model 4 SS patch
SUPERLOG/JCL - Just what it says
STRIP/BAS    - utility to strip 1 byte
CODE/TXT     - source to code program
SOUND/BAS    - SOUND for PRO-EnhComp
FIXES8/TXT   - The Patch Corner
```

QUARTERLY Coupon

Don't forget the coupon which accompanies this issue. It qualifies you for a $2.50 savings on DISK NOTES 8. We have assemblers on sale (EDAS, PRO-CREATE, ED/ASM-86). We have data base managers on sale (LB, LB86). And here's a great price on The Basic Answer. Remember, you must return the coupon to qualify. Please select only one sale item per coupon, and gosh, don't forget to include appropriate S&H fees. Some folks just never read.

Family Update

Here's where the folks who are not interested in my family can stop reading and proceed to the NEW PRODUCTS section. Judging by the mail I received in response to last issue's Letters to the Editor, everybody wants to hear more about the family. Gee, maybe I'll try to get a family portrait in one of these issues.

Now although it has only been three months since I wrote the last column, Stacey appears to have aged about 3 years. I guess it's just due to her getting out from under our thumbs and starting to play with the other kids in the cul-de-sac. That's a big leap in independence. I did attend her "graduation" from 3-year-old pre-school; however, the class picnic which I was also scheduled to go on was rained out.

Stacey turned four earlier this month. Now somewhere I read that after the first few birthdays - which are usually reserved for immediate family - children are entitled to have at their party as many friends as their age. According to that rule, we should have had four other kids. Unfortunately, it's difficult to narrow down commitments. Since both Brenda's family and my family live far from here, birthday parties for the kids have meant inviting some friends we have known for years. They too have children of Stacey's and Stefanie's age (5 in all). That would put us over the top. But we still had to invite a few of Stacey's classmates. So with an ensemble of about 10 kids and, say, 8 or so adults, Stacey celebrated her fourth. At least I now know why parents start letting their kids stay at birthday parties by themselves when they get to be four. We all had fun, but the pace was non-stop.

Stefanie has made some big advances in her selection of foods. She still won't even taste orange juice - and Brenda growing up in Florida. But Stef has begun accepting meat. I have gotten Stacey turned on to soups. She'll even scoop up the juice left over from the vegetable serving bowl calling that her "soup". Stacey even tried to get Stef to try some "real" soup, but Stef's not quite up to that. Stefanie finished up her "fun for two year olds" class. In the Fall (you know, when I get out the Summer QUARTERLY), Stefanie will start into the same class that Stacey just left. Both of my girls will be in "school" on Tuesday, Wednesday, and Thursday mornings. What will Brenda and I ever do?

Speaking of Brenda, get this! She's going to start driving the pre-school van for the morning and noon trips taking the pre-school kids to and from school. Brenda's a late sleeper - she doesn't like to get up before 8. It's a good thing we work close by. You see, I can talk about her now. Brenda, Stacey, and Stefanie are down in Florida visiting her family. Brenda reports that the kids are coming along fine in the swimming pool. They're even forgoing their water wings. That's a big improvement over last year. We were out a few times at the pool in our community over Memorial day weekend but the water wasn't too warm yet. It's quite a bit warmer in Florida - especially down in Miami.

I would have liked to go to Florida, too; except then you wouldn't have gotten this QUARTERLY for at least another few weeks. Besides, I had other work which had to get done. When you run a small company like we do, it's really hard to take off. The week in August will be the first week I have ever taken off during the summer months since MISOSYS was a full time job. That's been since 1981. In fact, the only week taken off has always been the week between Christmas and New Years. Now when I worked for AT&T, I had three weeks vacation, a week of personal days, and a week of Management days. I never knew what to do with all of that time. It's different now.

The most amazing thing about parenthood which has really struck home to me is the strong perception that I am reacting just like my parents reacted. I'm talking (read as "yelling") like my father talked. I'm thinking like my father thought. Perhaps I am more aware of that because I have the maturity which comes from being 44 years of age and I didn't have children of my own until I was 40. Or perhaps this is just a natural phenomenon.

Doing some vegetable gardening again this year. I'm trying some new "crops"; having planted some peas, green beans, lima beans, cukes, cantelopes, pumkins, and the proverbial tomatoes. Don't misunderstand, it's just a small plot. But it was fun starting everything from seeds. Also started a bunch of zinnias and marigolds which are now in the ground. With luck, I'll be harvesting my pumkins while I get out the next QUARTERLY! <grin> I am really thinking about planting a walnut tree; although they take a good ten years to develop, they are said to be great climbing trees - and Stacey really likes to climb trees. We'll see...

Announcing New Products

RATFOR-86™ and RATFOR-M4™

This product has been a long time coming. But MISOSYS is finally shipping both the MS-DOS compatible RATFOR-86 and the TRS-80 Model 4 compatible RATFOR-M4. Both products were developed by James K. Beard, Ph.D. Currently, James is a Senior Scientist working for the Hughes Aircraft Company. To get you started in understanding a little about the RATFOR language, here is a short introduction to RATFOR and its usage written by James K. Beard.

RATFOR is RATional FORtran. Although sometimes described as a "preprocessor", RATFOR is actually a language unto itself. Conceived in 1973 by Brian Kernighan of Bell Laboratories, RATFOR was originally implemented in collaboration with P. J. Plauger, now of Whitesmiths, Inc., in 1975. The original version, on nine track tape for microcomputers such as the DEC PDP-11, is still available from Addison-Wesley Publishing of Reading, MA. The authoritative book at that time was "Software Tools", by Kernighan and Plauger. The C language and UNIX were under development by Kernighan and Ritchie at Bell Laboratories at the same time. RATFOR became a standard UNIX utility. The West coast UNIX community adopted RATFOR as a systems language in lieu of C for a UNIX-like shell for use on all minicomputers not using UNIX. In addition, the Berkeley UNIX organization has continued to develop and market RATFOR as a major enhancement of UNIX 4.2. A major extension of RATFOR, called EFL (Extended FORTRAN Language), is also included with UNIX 4.2. UNIX 4.2 is marketed commercially by Mt. Xinu (UNIX™, spelled backwards!).

Although alive and well in the UNIX community, RATFOR is largely unknown elsewhere. An article I wrote for Computer Language Magazine for their February, 1986 issue was printed as their Exotic Language of the Month (page 75). Therefore, a brief description of RATFOR is in order.

The RATFOR language is defined in terms of tokens. Tokens are alphanumeric identifiers, numeric constants, and non-alphanumeric characters. Statements are made up of tokens and are delimited by carriage returns or semicolons. The language is free-field, meaning that there is no particular significance to any particular column. Lines are usually limited to 80 columns so that unformatted listings may appear on standard 80 column terminals and printed pages. Nearly all FORTRAN keywords are carried over, including the extensive and powerful standard FORTRAN function and input/output libraries. Also, FORTRAN arithmetic statements are used, including mixed mode arithmetic and the exponentiation operator. In addition, most keywords from the C language are also part of RATFOR; this is not surprising since Kernighan played a key role in both language definitions. The result is that RATFOR is a fully structured language in the tradition of ALGOL, PASCAL, C, and ADA with the power and versatility of FORTRAN.

RATFOR is nearly always implemented as a translator. A translator differs from a compiler in that a compiler translates raw source code to hardware-dependent assembly language source code. Translators are compilers which translate raw source code to source code for another high level language, such as C. Translators are common where portability of the compilers is important and are almost universal in the early stages of development of any compiler. "Compiler-compilers" such as YACC, which use data files which describe the keywords and structure of a language as input data, produce compilers for the new language in C source code. Compiling the result gives a compiler for the new language.

RATFOR translators translate raw RATFOR source code into FORTRAN object code. This has exactly the same advantages as translators to C in contexts where FORTRAN is universal and largely portable. This is true in the microcomputer/mainframe community today. This can be to advantage to the engineering workstation user in that powerful, easy to write structured source code can be executed on any machine which has a FORTRAN compiler. Since uploading and downloading ASCII text and programs is standard in engineering workstations, RATFOR can be translated at the workstation, the FORTRAN object file can be uploaded, and the final compilation and execution can be done on the mainframe. This makes best use of the workstation as a file manager and application or software development facility, and the mainframe is best used in execution of large optimizing compilers and execution of applications programs.

The simplest way to demonstrate the simplicity of RATFOR is to show a few sample programs. The one selected is a simple program used to compute and plot tradeoffs associated with the use of Dolph-Chebychev windows, a weighting

technique widely used in communications to control crosstalk. Examination of the program, reproduced below, shows that a simple relationship between the algebra used and the program is evident from the listing. Also, statement labels are completely absent except for FORMAT labels. Use of "define" and other C language keywords is important to the simplicity of the program. The same program in FORTRAN would be much more complex in interpretation of the source listing.

```
#Program tested on TRS-80 Model 4
define PI 3.1415927 #Mathematical constant
define READLUN 1 #I/O units for console
define PRINTLUN 3
program dcplot #Plot Dolph-Chebychev window
parameters
dimension h(128)
print 1000; 1000 format(" No. of points in
window, output LUN's? ")
read 1005,n,lun,lun1; 1005 format(3i10)
if(lun>4) call open(lun,"plot/dat ",256)
write(lun,5000)
  5000 format(" Windowing Tradeoff"/_
    " Two-Way Sidelobes"/_
    " Lobe Broadening")
if(lun>4) call open(lun1,"plot1/dat ",256)
write(lun1,5010)
  5010 format(" Windowing Tradeoff"/_
    " Two-Way Sidelobes"/_
    " Lobe Crossover, dB")
do isdb=20,100
  {
  sdb=isdb
  call dolchb(n,-sdb,h,sbf,dbd)
  write(lun,1010) sdb,sbf; 1010 format(2g15.6)
  write(lun1,1010) sdb,dbd
  }
end
subroutine dolchb(n,sdb,h,sbf,dbd) #Dolph-
Chebychev window weights
#Inputs:
# n   Number of weights
# sdb Sidelobe level (>0 for weights h(i), <0
for sbf and dbd only)
#Outputs:
# h   Weights (not computed if sdb<0)
# sbf Sidelobe broadening factor (3 dB
width/(1/n))
# dbd Bin crossover with FFT or beam crossover
with delta sine =1/n
#Frequency response is
#  T(n-1)(q*cos(PI*f))/s
#where
#  T(n-1) is Chebychev polynomial of order n-1
#  f is fraction of sample rate
#  q is constant>1 found from
q=cosh(acosh(s)/(n-1))
#  s is sidelobe height as amplitude ratio
dimension h(n),x(256)
asdb=abs(sdb)
q=qfroms(asdb,n) #Compute q
q3db=qfroms(asdb-3.0103,n) #Compute broadening
factor
sbf=2.*n*acos(q3db/q)/PI
dbd=20.*alog10(dcf(n,q,.5/n))-asdb #Compute
beam/bin crossover point
if(sdb<0.) return #Exit now if weights are not
to be computed
#Compute frequency response
nback=n+1
offset=.5*nback
n2=offset
x0=n*dcf(n,q,0.) #Normalization factor
x(1)=1./n #DC term is trivial
do k=2,n2
  {
  f=(k-1.)/n
  x(k)=2.*dcf(n,q,f)/x0 #Factor of 2 here
instead of in series
  }
do j=1,n2 #Use DFT to compute weights from
frequency response
  {
  df=2.*PI*(offset-j)/n
  sum=x(1)
  do k=2,n2
    sum=sum+x(k)*cos((k-1)*df)
  h(j)=sum
  i=nback-j
  h(i)=sum
  }
return
end
function dcf(n,q,fa) #Dolph-Chebychev
frequency response
# n  Number of weights
# f  Normalized frequency (fraction of sample
rate)
#Returns T(n-1)(q*cos(pi*f)), where T(n-
1)(x)=cos((n-1)*acos(x))
ifa=fa+0.5 #Limit f to |f|<=0.5
f=fa-ifa
argl=q*cos(PI*f) #Compute fundamental argument
switch n
  {
  case 1: tn=1.  #T0(x)=1
  case 2: tn=argl#T1(x)=x
  default:       #Compute rest by recursion
    {
    tn2=1.
    tn1=argl
    tx=2.*argl
    do i=3,n
      {
      tn=tx*tn1-tn2
      tn2=tn1
      tn1=tn
      }
    }
  }
```

```
return(tn)
end
function qfroms(db,n) #Computes Dolph-
Chebychev parameter q from sidelobe ratio
# s  Sidelobe ratio (dB=20*alog10(s))
# n  Number of weights
#Equation is q=cosh(acosh(s)/(n-1))
#double precision ddb,s,argl,dval
ddb=db #Convert to double precision
s=10.**(.05*abs(db)) #Compute voltage ratio
from dB
argl=alog(s+sqrt(s**2-1.)) #Compute acosh(s)
dval=exp(argl/(n-1)) #Compute cosh(argl/(n-1))
dval=.5*(dval+1./dval)
qfroms=dval #Convert back to single precision
return
end
#acos(x) function required for Model 4
```

The above example shows how RATFOR extends the power of FORTRAN while at the same time making programming simpler. Another example shows how the use of macros, set up with the "define" statement, can extend the fundamental keyword set in the language. The example selected is INPUT, which performs a function similar to the BASIC language INPUT keyword. A prompt is defined and a variable is read in from the keyboard, and an error in input results in a repeat of the prompt until the keyboard input is successful. Note that local labels are not repeated; this process is similar to that used in better macro assemblers which use non-repeated local symbols in macros. Following the example are samples of its usage.

```
#INPUT macro tested on MS-DOS microcomputer
using Microsoft FORTRAN version 4.0
#Syntax for INPUT macro:
INPUT("<prompt>",<variable>[,<format label>])
define INLAB 1
define(INPUT,[define(INLAB,incr(INLAB))
repeat
  {
  ierr=1
  print *,$1
  read(*,ifelse($3,,*,$3),err=INLAB) $2
  ierr=2
INLAB [continue]
  } until(ierr==2)
])
define READLUN *
define PRINTLUN *

...

#Sample usages:
INPUT("MDL for range computation (0 to
end)?",mdldb) #Floating point variable
INPUT("Input file name?",fname,100) #ASCII
string with FORMAT label
```

```
100 format(15a1)

C Resulting FORTRAN code:
23003 continue
      ierr=1
      write(*,*)'MDL for range computation (0
to end)?'
      read(*,*,err=2)mdldb
      ierr=2
2     continue
23004 if(.not.(ierr.eq.2))goto 23003
23005 continue

C ...

23002 continue
      ierr=1
      write(*,*)'Input file name?'
      read(*,100,err=3)fname
      ierr=2
3     continue
23003 if(.not.(ierr.eq.2))goto 23002
23004 continue
100   format(15a1)
```

Macros can call other macros or even themselves. Macros which call themselves are called recursive macros. These can result in very powerful new keywords which are limited only by the imagination of the user.

MISOSYS publishes two versions of RATFOR. One executes on MS-DOS machines and one executes on the TRS-80 Model 4 under LS-DOS/TRSDOS Version 6. The two versions share a RATFOR manual of about 86 pages including an extensive index. An implementation-specific installation manual concerning hardware and workstation issues is available with each version; the advantages and disadvantages of the FORTRAN compilers available are also discussed candidly in the installation manuals. All features of RATFOR are described in stand-alone reference form in the manual. A shorter narrative of RATFOR features is also provided there.

The MS-DOS version RATFOR translator is provided as an .EXE file so that it can be used with all MS-DOS FORTRAN compilers or in a workstation which does not have its own compiler at all. The Model 4 RATFOR translator is provided as a relocatable object file which needs to be linked with FORLIB/REL. A Job Control Language file is included which performs the linking process using your FORLIB/REL FORTRAN library and L80 linker. The Model 4 version, RATFOR-M4, also includes a customized version of the LED text editor for source code maintenance. Each version comes with a batch or JCL file useful for performing

the RATFOR translation, FORTRAN compile, and subsequent linking to generate an executable file from the RATFOR source.

These RATFOR translators are each currently priced at $99.95 + $5 S&H (US); the price includes installation and reference manuals, disk-based software, and binder. Order M-21-073 for Model 4 or M-86-073 for the MS-DOS version.

PRO-WAM Release 2

Since the release of LS-DOS 6.3, we have had many requests from PRO-WAM (nee PRO-NTO) users asking for revisions to the BRINGUP application to extend its date support beyond 1987. Well here is an announcement that should please every PRO-WAM user; and make many other folks run out to get PRO-WAM Release 2.

In the 80 Micro review of PRO-NTO Release 1 back in November 1985, Hardin Brothers said, "PRO-NTO is one of the most useful products for the Model 4 I've seen." Well we have supercharged this new release to include just about every feature requested by our PRO-WAM users and added other significant enhancements that should make PRO-WAM even more useful to just about every owner of a 128K Model 4.

To those Model 4 owners uneducated about PRO-WAM, here's a little PRO-WAM overview. The "WAM" stands for "window application manager". PRO-WAM is a manager of applications which pop up in windows on your video screen. You can pop up a PRO-WAM application whenever you are running any other "DOS-behaved" program. The PRO-WAM application can capture anything on the video screen by means of a function called "import". Similarly, a function called "export" can be used to pass back to the interrupted program, anything which is in the PWO-WAM application's window.

There are essentially two parts to PRO-WAM: (1) a resident module which supervises and controls the operation of the video windowing and application execution, and (2) a set of application programs some of which may be made to reside in memory. PRO-WAM uses about 2500 bytes of your machine's high memory and one memory bank. That's why it requires a 128K machine.

Application programs provided with PRO-WAM include a calender, a bringup and tickler file, an address/rolodex™ file, a 3x5 card file, a modem autodialer, a 4-function calculator, a disk-based key-stroke-multiplier, a todo list manager, a terminal program, and more.

Let's take a look at some of the features which have been added to the window manager.

PRO-WAM now supports Model 4 reverse video detection and restoral at each level of application execution (there are four levels). For those of you using Visicalc or Multiplan, you can now get into PRO-WAM applications and return to your spreadsheet screen with reverse video intact. Besides that, PRO-WAM applications can make use of reverse video in their screens; we use it in CARDX.

More Model 4 folks are taking advantage of extended memory add-in boards to gain additional memory banks. With that have come requests for better memory bank control in PRO-WAM. So we added a BANK parameter for PRO-WAM installation to force PRO-WAM to use your designated bank number rather than automatically search for a spare. Even if you don't want to make use of that, PRO-WAM's automatic search will now look up to bank 30.

PRO-WAM is installed as a keyboard device filter. Under Release 1, when you wanted to remove PRO-WAM by a "PRO-WAM (OFF)" command, all keyboard filters were reset which required you to re-filter the *KI device with the other filters you had installed. Under release 2, "PRO-WAM (OFF)" now unfilters the *WM device rather than RESET *KI. This leaves your *KI device chain intact (except for the PRO-WAM removal).

PRO-WAM now accesses application modules from a library - not from standalone modules. This keeps the applications from cluttering up your directory. PRO-WAM initially loads from the WAM0/APL library. Thereafter, the UNIVERSAL function request for "Application?" is now satisfied from that library on the drive where the library was found unless overridden by the application module specification (mspec$) entry. Even though a library can hold 32 applications, more than most folks will have use for, UNIVERSAL can still specify the library to access or default to WAM0/APL. Specify "modname" to access WAM0/APL for "modname". Specify "#modname" to access WAM#/APL for "modname". The new drive will be again used in subsequent UNIVERSAL requests until changed by explicit entries in a mspec$ (either UNIVERSAL or programmed execution). You can have up to ten application libraries (designated WAM0/APL through WAM9/APL).

We've expanded the capabilities of the window supervisor call with two significant enhancements: programmed control of export and programmed invocation of an application module. We use the former to automate the PHRASE application and the latter to enable the "bringup hot key" in CAL. Of course both facilities are available to every programmer, as well.

To expand on the concept of "protected fields", we've added the concept of "protected characters" to the behavior of export/import. Any character within the defined rectangular export/import area which has a value greater than 127D (i.e. bit-7 set) will not be output/input; i.e. it will be considered a protected character. Most typically this would be a reverse video character; however if reverse video is not in effect, any graphic or special character greater than 127 will be considered protected and not output/input. This will be extremely useful with our revised CARDX card file application.

Export/import now supports a logical ENTER character which will be translated to a hard carriage return regardless of the import/export state (CR or NOCR). Thus, a text phrase can be export as "multi-line" even though it is a continuous stream of characters without a carriage return. The logical ENTER character is tested before the protected character test; thus, values greater than 127 are acceptable for use as a logical ENTER. The default logical ENTER character is 127d (7FH) which can be entered by <CLEAR><SHIFT><ENTER>. We've added an "ENTER" parameter to the PROWAM installation so you can change it to one of your choice. It's even configurable with DEFAULTS.

We've used the expanded features in the application manager to make considerable improvement in the existing applications and added some powerful new ones. Here's what's been done in the application arena.

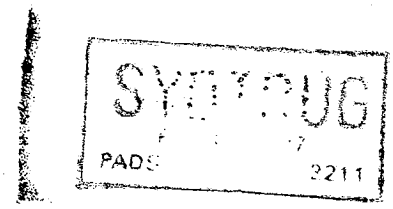
Many users requested that the CALendar application flag days which have a BRINGUP activity entered. Well we added a facility to display an asterisk in the leftmost position of a day cell if there is an active BRINGUP record for that day. CAL scans the entire BRINGUP/DAT file and maintains a table of flags indicative of activity per day for an entire year. The file is only re-scanned when you change years (if it found a BRINGUP file initially). We also added a "bringup hot key" in the CAL command list so that you can easily position to the day showing an asterisk and hit one key to bring up the list of activities you have scheduled for that day. That facility has become indispensible here at MISOSYS.

Speaking of BRINGUP, we altered the structure of the data file so that the year spans 16 years: from 1984 to 1999. The date field storage was also reorganized so that is can be sorted in ascending order. Since sorting by PSORT requires a data file header record, that was added too; BRINGUP data files can now be sorted by date and time using PSORT. This requires a convert utility for your existing files which is included with the release.

Now sorting the bringup data file only makes sense when you have a facility for printing activities. Enter the new BUP/APP application. This wammie, which can be invoked directly from BRINGUP, supports printing of bringup activities by date range using a syntax exactly like that in your DOS for BACKUP and DIR date parameters. We also added '.' as a shorthand for "today". Standard user-input print formatting is supported similar to that facility in ADDRESS and CARD. BUP also incorporates the REMOVE command which used to be in BRINGUP.

I keep an ADDRESS file in the computer on my desk. Since I like to use it to capture address headings when writing letters, I got tired of exporting the address record as it appeared in the window which required editing in my text editor. That's why I designed HEAD. This new application accesses the ADDRESS/DAT file and presents an address record in the format:

Company
First Last
address1
[address2]
City, ST ZIP



If the Company field is blank, then it won't appear in the window. Likewise, if Address2 is blank, it won't take up a line. I even coded HEAD to drop trailing spaces from the fields so, for instance, there's no big gap between First and Last. HEAD provides commands just like ADDRESS: search, next, prev, first, last. An <ENTER> will automatically export the record. The application is useful for extracting address information for use as a heading in a letter. It sure saves me some time and eliminates needless editing.

We've modified ADDRESS/APP to pick up the sort key and key length so that the search command now uses the external key information

specified in the data file header record rather than hard coding the search to the last|first fields. This means you can alter the key location and length fields in the ADDRESS/DAT file to order/search by ZIP, Company, or whatever. This was to satisfy a few requests from folks wanting ADDRESS to sort and search on the Company name field.

I fail to understand why the MS-DOS world clamors for key-stroke-multiplication facilities such as SUPERKEY, POWERKEY, and whatnot, yet the TRS-80 users rarely explore the capabilities of their own KSM facility which has been available in LDOS and TRSDOS 6 for years. Nevertheless, we have added a new application called PHRASE. This wammie accesses a PHRASE/TXT file consisting of any number of phrases. You can scroll through the phrases or enter a phrase mnemonic. An <ENTER> will export the phrase selected. Your phrases can make use of the logical ENTER character of export. This is akin to a more or less infinite KSM. The syntax of a phrase is a 2-character mnemonic followed by 1-78 text characters and terminated by an <ENTER>. The PHRASE/TXT file is composed of any number of phrases. It is terminated by a period, '.', following the last <ENTER>. Use any ASCII text editor to input and edit the PHRASE/TXT file (use TED, for instance).

The phrase facility can work wonders when you want to prepare letters, drafts, etc., using "canned" phrases. You can also use PHRASE to store strings for application print formats. How about DOS command strings? It boggles the mind!

CARDX/APP is a new application which is an enhanced CARD file. It supports up to ten CARDx/DAT files [0-9] rather than the single one under Release 1. It makes use of reverse video to designate protected screen positions which cannot be edited (nor exported). CARDX will actually skip over any screen position which is in reverse video.

CARDXF/APP is used to construct and edit a CARDFORM record for use with CARDX. This wammie allows you to create new data files and add/edit a cardx record identified as "CARDFORM". Reverse video, which is used for protected fields in CARDX/APP, is toggled via <CTRL-R>. The screen indicates the current state of reverse video. CARDXF also allows you to "populate" the CARDX file with any designated quantity of records initialized with the "CARDFORM" form.

It's easy to use the cardxf editor to "paint" your form onto the text area of the window. By using reverse video for every position which you want "protected" from entry while using CARDX, you can create a form to be filled in later. When using CARDX, the cursor will be automatically moved to the unprotected positions. This combination of CARDX/CARDXF can really provide you data base functions.

Around here we always seem to forget to do this or forget to do that. Sometimes BRINGUP doesn't work when you just want to keep a list of items to be done. Remember that todo pad? That's why I designed TODO/APP, a new application which is used to establish and maintain a simple list of items "to do". Now I wanted to keep Brenda's todo list together with mine so I designed it with a field to indicate "who does it". This wammie generates a TODO/DAT file which is similar to a BRINGUP file. Allows priority of 1-8, a "who does it" value of 0-15 where a value of 0 is considered global (more on that later), and a text field of 29 characters. The "add" date is added to the record. Commands allow you to "add" an item, mark an item as "done" which deletes it, "file" any changes, move to "next" or "previous" display page, and "list" items. A page displays the 8 items in a disk record (rather than working like BRINGUP, the display works like DIALER). The list command allows you to print all records or the current record additionally specifying either all "who does it" or a particular "who does it". Any item tagged as "who = 0" will always appear on a listing. Forget Hi's "job jar"; TODO does it!

PRO-WAM comes with an on-line help facility which has been updated to include all of the features added in release 2. It can keep you from having to resort to the user manual when you forget a particular operation of PRO-WAM. Of course, since every application has a command menu, you will rarely have to resort to the manual once you have familiarized yourself with the operation of the applications you will use.

PRO-WAM also comes with some improved utilities; and some new ones to boot. With all the talk about sorting BRINGUP files by date and time, you may have realized that PSORT needed some work. Well PSORT/CMD has been revised to accept an expanded data definition of up to two key sort fields. Not only that, but the fields can be either character or integer.

Because of the switch to application libraries, we've included WAMLIB, a utility to build and maintain application libraries. WAMLIB has functions to create a library as well as the functions:

add a module
delete a module
replace a module
extract a module
list module directory

WAMLIB is menu driven and easy to use.

Since applications are now invoked from libraries, PRUN has been enhanced to load applications from libraries, too. However, since some folks may have a need to invoke an individual application file, PRUN has been enhanced with a parameter APP (abbreviated A). PRUN will normally load the application from the WAMO/APL library (or other library when the module name is prefixed with the library number). If you want to force PRUN to load a standalone application (i.e. for testing before adding it to a library), specify the APP parameter as in: PRUN application (APP).

Well that's it. PRO-WAM release 2 includes the window application manager, 18 different applications in a library, a library manager, the PSORT utility, the PRUN facility, a help facility, and a DEFAULTS utility to customize your PRO-WAM configuration; all on disk. PRO-WAM Release 2 includes a totally revised reference manual housed in a 5-1/2" by 8-1/2" 3-ring binder. If you don't have PRO-WAM yet, order M-51-025 for $74.95 + $5 S&H (US). If you are already a PRO-WAM (or PRO-NTO) user and want to trade up to release 2, send in your old master diskette and order M-51-125, the complete PRO-WAM Release 2 package for $25 + $5 S&H [note: if you have purchased PRO-WAM release 1 since April 1, 1987, you can trade up to release 2 for $20 + $5 S&H with the return of your old disk]. The trade-in offers are valid only until 03/31/88; so hurry up with your old PRO-WAM tradein to Release 2.

Letters to the Editor

Leap years a la 2000 revisited

Continuing from TMQ I.iii, here's some more on the subject.

(**Fm: Adam Rubin 71320,1052**) Sorry, but you'll have to take that up with Parliament. (Assuming the Statute of Limitations hasn't expired, of course!)

Actually, the Act included provisions for contracts and other "dated" events, which were extended eleven days beyond any old style date agreed upon. For example, George Washington was born on February 11, 1732 (old style), but was not twenty-one years old until February 22, 1753 (new style).

For the benefit of anyone else (if there IS anyone!) who's following this rather disjointed thread, the library finally found a booklet I'd requested. The full title is, "Act and Bull; or, Fixed Anniversaries--a Paper submitted to the Numismatic and Antiquarian Society of Philadelphia, Nov. 4, 1880, by Lewis A. Scott, with an Appendix containing the Bull of Gregory XIII., translated, and the body of the Act of Parliament," and it was published by the Society around 1881 or 1882. The question at hand was the proper date for the bicentennial of William Penn's landing on November 8, 1682 (old style); it contains an interesting discussion as well as the other documents mentioned in the subtitle. I found it very informative, and recommend it to anyone else interested in the subject.

Here's what must be the final remark concerning the next centennial. The following appeared in the Ann Landers column of Friday, May 29, 1987 June 1, 1987 [as printed in THE WASHINGTON POST].

"Dear Ann Landers: Will you please help settle a friendly dispute? The loser has agreed to send a $50 donation to the winner's favorite charity. Question: What is the first day of the 21st century? My friend insists that it is Jan. 1, 2000. I say it is Jan. 1, 2001. Who wins? - Hugh W. in Wilmington, Mass.

Dear Hugh: I called six very bright people and guess what? Three said Jan. 1, the year 2000, and an equal number said Jan. 1, the year 2001. Each person went back to the year Christ was born and from then on it became unbelievably complicated. I am not touching this one. It's one of those tricky questions that can make you crazy."

Well I have been reading the Landers column for quite a few years (who said real men don't read Ann Landers?). After the lengthy discussion which appeared in the last issue of TMQ, I could not remain silent. Taking keyboard in hand, I sent the following letter to Landers.

"Dear Ann Landers, After reading your column for years (I am male, age 44, married with two children), I wanted to offer my insight into Hugh W.'s question on 'the calendar date of the first day of the 21st century'. I am in the computer software business and publish a magazine in support of my product line. Just as you have found with your readers, my readers generally come up with some good advice.

Our subject in question was not which day was the first day of the 21st century but rather was the year 2000 a leap year. The responses also shed some light on the answer to your readers question. In the interest of providing advance discussion on a topic which will certainly rise in significance as we approach 2000, I offer you a copy of the dialogue which was printed in my Winter 1987 issue, for what its worth. I've highlighted the paragraph pertaining to your question.

Glad to be of assistance (but not a contender for your previous job in that 'other' paper)."

Compliments and complaints

(**Fm: Lloyd Davidson**) Hello folks, Have been absent from the forum for some time as I have been busy with other computers, languages and a coupla BBSs. The first thing I did when I logged on then was to download and read (off line) all of the current messages from all categories. A task that is much easier and less expensive since CIS decided to give us a break on 2400 baud. Would like to comment on a few of the messages for no other reason than to get my two cents in.

LS-DOS 6.3 - From an end users point of view, this is a dandy system. Does what it is supposed to and does it well. I have a number of disks configured in many different ways and have not had a hitch from 6.3. One disk I use for word processing has both SuperScripsit (with dictionary) and Powerscript on it. Even threw in TED and have Pronto (PRO-WAM now) resident to bring up a calendar, a card file

and a bringup file. Steve Milliken's excellent SHELL runs fine with a short patch supplied by Steve and I have what has to be a VERY good WP setup. Might add I've had the Model 4 since sometime in 82. It is the old non-gate array model that I've added a beefed up power supply and two teac DS DD drives to.

TANDY aka Radio Shack - True I have not always been treated like a returning prodigal son each time I walk into the store and I have done most of the hardware hacking and repair myself. (Often tuning to maximum smoke) I have met the usual "battery salesman" in many RSCC's but lets face it folks, they make darn good machinery and without them we wouldn't have a thing to grouse about would we?

THE PRICE OF DISKS - Have found that usually cheaper is better. Think more depends on the condition and quality of your drives than on how much you spent on the media. Have had $3.00 a shot disks with pretty jackets fail to format on a PC-clone with a wobbly drive. It is not difficult to maintain and adjust your own floppy drives and you otta learn. Will never be able to get really precise like a mechanic with expensive test racks but you can keep it in the ball park. I have bought nothing but the $.29 specials for some time and have had no trouble. I really think that the Model 4 and LSDOS could format a flat rock if you get the wholes punched right.

INSTALLable Software

(Fm: Byron P. Peebles) Am I imagining things, or are the VAST MAJORITY (read "all commercial") of microcomputer (8- and 16-bit) a little on the defensive side when suggestions are made on improving their products? I get the feeling I'm hearing a whining sound while I read the replies. I know budgets are tight for the small business person. I know theft of software is rampant. I know there must be fifty problems I'm not aware of for the commercial developer. I also know there exists a person called the "end-user/client". This mythical person is NEVER WRONG, is TREATED WITH RESPECT, and has the WORLD'S MOST RESPECTED OPINION. I also know that this is the person that keeps everyone in business.

And once in a blue moon this "end-user/client" has experiences that could provide great insight to the "high tech" developer, were that developer willing to listen.

{Sound of soapbox being shoved back under the desk} The above message relates to the coming death of the 8-bit microcomputer.

(Fm: MISOSYS) Byron, I think when a developer spends a lot of time revamping a package and the first thing folks ask when it gets released is why don't you add this and add that, and when's the next release, it gets sort of frustrating. Very few folks are considerate enough to wait some degree of time before they start asking for the moon again.

On RSCCs

(Fm: jjkd) [Here's] a story relating to one of the better RSCC's I've been to. It happened to be in Ontario, Canada. I went in on a lark, as I was on a business trip and wasn't really equipped to carry anything major back with me. I did walk out pleasantly surprised at the employee's attitudes and capabilities, along with the neatness of the store and stock, and I got a Canadian RSCC catalog. That had to be one of the best RSCC's I've ever been in. I meant to get the name of their DM and write out an "attaboy" letter, but I never did get around to it.

Anyway, to make a long story a bit shorter, I took the catalog back to work with me. A friend of mine was considering buying a 3000, but had been hemming and hawing for some time.

Me: "Hey Lynn: did you buy that Tandy 3000 you were looking at?" Him: "Naw, I haven't really decided yet." Me: With a very sorrowful look on my face, "Gee, did you hear about the new price increases?" Him: "No! Computer prices go down, not up!" Me: "Yeah! Somethin' to do with Japanese chips, or the yen or stomthin' like that, I heard. Look, I just got the new Radio Shack Computer catalog..."

I showed him the Tandy 3000 page in the catalog, with the catalog folded over, of course. There the price was, but in Canadian dollars of course, something like $4000 without the hard disk vs. $2600 in the US. His eyes just sort of glazed over, and the look on his face was priceless. He didn't notice the listing for MS-DOS available in French for $99 or so, thank goodness. I let him off the hook about five minutes later.

Two days later he bought a 3000 through one of the mail order places.

ANSI standards -where to get them

(**Fm:** Adam Rubin) ANSI standards can be purchased from (where else?) Sales Department, American National Standards Institute Inc., 1430 Broadway, New York, NY 10018. (Unless they've moved, that is!) Also, some libraries have reference copies of these; I've found them at both the Chicago Public Library and the Engineering Library here at U-M, for example. Incidentally, I believe the full name is X3.64-1979, where 'X' is the committee, '3' is the subcommittee, '64' is this particular standard, and 1979 is the year of the last revision of this standard (which may be later by now).

X3.64 covers most of the escape sequences, including a few interesting-sounding ones -- Operating System Control, Message Waiting, etc. You'll probably also want X3.41, which covers other control sequences (Shift In and Shift Out, Select Character Set, and so on), and X3.4, which defines printable and control characters, and their functions. (I believe this last is the 'definition' of the ASCII character set.)

If you'll be using these to write a VT-100 emulator for KERMIT, I'd strongly recommend Digital's "VT-100 User Guide", which explains which functions are implemented and what they do. Their "Terminals and Printers Handbook" might be useful, too. (I have a copy of the VT-100 User Guide, if you have any specific questions.)

Requests for CIS files on disk

(**Fm: jjkd**) The original author retains all distribution rights for materials uploaded to CIS. So, if at their behest, an article is routed to somebody else for republication, there is no problem. If the author limits further distribution, or the author is not consulted/does not give permission there is a problem. Wholesale reproduction of a group of files from CIS can be viewed as violating CIS' compilation copyright on them all as a collective work.

Upgrading MISOSYS Products

(**Fm: Bob Connors**) Can you tell me how to and how much is an upgrade to the products you carry? I would like to upgrade my SAID from 1.0 to 1.1 (since I have noted that that is the version currently being patched -- somehow the fact that the version number changed escaped me). Do I send back the master disk or is my registration card I mailed in sufficient to request an upgrade (I guess mailing the disk back makes more sense). I think I have seen this policy stated before, just can't remember where.

One final thing, do you think that it would be possible to include the current version number of your products in your price lists? Would sure help to let us "busy" types know when an upgrade (not patchable) is needed. I just never even realized my SAID was not the current version until I went to patch it! Since I got this version of SAID with my EDAS 4.3, will the EDAS be upgraded also along with MAS, MED, et al.

(**Fm: MISOSYS**) The SAID 1.0 -> 1.1 upgrade costs $5 + $2 S&H. It was covered in TMQ I.ii. Ah, carrying a version number would not work since SAID is no longer a standalone product. And yes, we would reduplicate the entire disk on the SAID update (gosh, you'd get all the patches). You need to return your disk (in a protective mailer).

TMQ kudos

(**Fm: Marc Nowell**) Received TMQ I.iii today, and I'm greatly impressed! It is a HUGE issue, and will give me plenty of reading for the next few weeks. Also, it's nice having my names (first and last!) spelled right in the text where I am quoted. It is appreciated and I'm glad that I could contribute!

(**Fm: MISOSYS**) Jim Gaffney wasn't so lucky. I mispelled his in the TOC. Keyboard must have had a sticky key(!). Now don't think that the next issue is gonna be that big. The weight drives my foreign costs way above subscription costs due to the cost of mailing. That sucker cost $5.49 to mail to Australia(!). Actually, the next issue should stay within 100 pages - it has to.

Also, as I fine tune the "look and feel" of the QUARTERLY, I have decided to include the full names unless specifically requested not to. This includes public messages which have been extracted from our Compuserve forum.

(**Fm: Paul Rehberg**) I enjoyed lying by the pool yesterday (sunny, a hot 91 degrees in Houston) reading my **Winter** edition of TMC. Shall I stock a supply of cord wood for the fireplace in anticipation of the Spring issue?! <grin>

I would gladly pay $59.95 again for a new Pro-Wam that included a business calculator (i.e., one that knows how to perform subtraction like us 9 to 5ers use all day long). Any chance of one in the update?

And what ever happened to 3-ring holes in the TMQ?

(Fm: MISOSYS) We gave up on 3-hole punching after about 50 copies of issue I.i were ruined by the hole punching operation. We do not expect to take another look at hole punching. However, we do expect to continue the perfect binding introduced in issue I.iii - really makes a class appearance.

I assume that a you mean subtraction as in "number - number = result". Have you not used AFPCALC? That is "algebraic". Do you want something else? Kind of hard to squeeze any more functionality in a floating point calculator application as the float routines take up quite a bit of space. If you want an algebraic "INTEGER" calculator, then that can be arranged; however, one is not on the drawing board for release 2.0.

(Fm: Adam Rubin) Yes, the perfect binding (on TMQ) does look much nicer. I also liked the separate sheet for postage and addresses. Any plans to get an ISSN? (Oh, and I liked the content too!)

Incidentally, how should budding authors submit query letters -- message, U.S. Mail, or some other way?

(Fm: MISOSYS) Query letters? If you mean an article, please submit on disk. If you mean a query about submitting an article, then either a note on our Compuserve Forum, PCS-49 (don't use EZP) or a letter or a call.

More on TMQ I.iii

(Fm: Shane Dawalt) I just received my Winter TMQ and was just a bit surprised with the Letters to the Editor. The person who said the material in TMQ was over his head has boggled my mind. There certainly isn't anything that profound. And if it is, it is for hardcore programmers which understand its implications [perhaps on the second or third reading if they are rusty programmers]. If the information doesn't make sense, that particular person is missing background information on the subject and should take a couple steps back.

I agree that user should support companies who are supporting them, but the comment you made, and I'm paraphrasing; user who don't support the people who cater to their machines don't need the support in the first place. If a magazine is talking about BASIC, I'm not going to buy it since I don't use BASIC all that often. It is not that people don't want to buy the product, they must weigh its usefullness. I bought PRO-MC from you because I was tired of pampering BASIC ... and I haven't been sorry for my decision on buying PRO-MC because it is useful. I don't buy what I don't need and I don't expect others to do so, even if it might be a $15.00 subscription to a magazine.

People who are ticked off at you should be ticked off with themselves. I believe you are quite correct to discontinue support if sales figures don't warrant it. [I can hear the nashing of teeth already.] I think the Model I people should be wondering where all THEIR programmers went rather than placing the blame on you.

Finally, portability is a problem with the Model I/III/4. I have a strict rule when I develop software ... if it uses undocumented addresses -- PEEKs, POKEs and whatall -- it should not be sold or distributed by information services or BBSes. Too bloody hard to support ... never know what people are doing on their systems.

Anyway, great TMQ. BTW, are TMQs gettin' larger or do I need new glasses?

(Fm: MISOSYS) I hope I can quote you on that. The last issue (I.iii) was way too big. It got that way because it was about 6 weeks late; thus, it had a greater time interval for input (I did cut out a few things as I noted). I expect the next to compensate so that the average is not above 100 pages. Can't afford the shipping at that weight!

The Source for 6.2

(Fm: Kevin R. Parris) I understand that "The Source" was published by LSI, but I bought my copy from you. I also have read the disclaimer in the front of the book - 'questions will not be answered'. But I am willing to take the risk that your response will simply be "the disclaimer is correct", on the chance that you might answer this question: on page seven of

volume one, in the listing for "Bootstrap Loader", in the section after label "$A1", there are two lines where object code is shown, and comments, but no instruction mnemonics appear. This is at locations 02E4 and 02E6, and also further down that page this happens again, as well as a few other places I have noticed in the book. Why is this? Thank you for your fine work. The new TMQ arrived Thursday - the heavy plastic wrapper is a very good idea!

(Fm: MISOSYS) The source code in question relates to access of Tandy hardware. Proprietary agreements between LSI and Tandy precluded LSI from publishing the "source" to those particular statements.

New @EXMEM SVC, etc

(Fm: Paul Bradshaw) Regarding the newest issue of TMQ, I'm curious of something. Is this new SVC 108 (@EXMEM) official? I see you are including it with the Model 4 interface kit for LDOS... Is it now a "part" of the DOS? Will the next version/upgrade include it? Why wasn't something like this included in the 5.3/6.3 upgrade? Why not in 6.0, for that matter? It is an absolutely WONDERFUL addition, and was sorely missed.

Also, about your new RSHARD hard disk drivers... These are different from the TRSHARD that I got with my Hard Drive, and that was distributed with 6.2, aren't they? If so, how "big" are they? Is there a size savings over the ones I'm using now?

Thanks again for a superb issue of TMQ. I have only one comment/suggestion. In your new section covering assembler on MSDOS, could you "back off" just a little bit and fill in necessary information such as "how many registers there are, what are their names, how do you address memory, what are some instruction names, how do you interface with the DOS", etc? Your first example was nice and fairly easy to follow, but never having seen the structure of the 8088 chip or its brethren, I had to make assumptions regarding the nameing of the registers ("AH and DX", etc). I'm VERY interested in seeing this column continue!

(Fm: MISOSYS) The new SVC-108 is "semi-official". That means that I have received approval from LSI that the @EXMEM can use that SVC number as it is more a system service function. It is not part of the 6.3 release as it was not done by LSI. I would have no comment concerning any future release of 6.x. "No comment" does not imply any between-the-lines assumptions.

I have asked Phil Oliver, author of ED/ASM-86, to address 8086 programming from the standpoint of exactly what you have suggested. I expect to take a different angle. My supposition is that TMQ has a lot of readers already familiar with Z80 programming under LDOS or LS-DOS; thus, my thrust will be to expose these readers to the kinds of things which can be done under MS-DOS which would be very similar to what you are already used to. Thus, it is expected to serve as a bridge. 8086 coding is confusing enough.

From my check just now as I write this, The TRSHD6/DCT driver is 330 bytes long when resident. RSHARD6/DCT is 335 bytes long - 5 bytes longer than the Tandy supplied driver. Mine is longer due to the additional coding overhead necessary to allow partitioning by both cylinder and head and on supporting physical cylinder sizes up to 1024 cylinders.

(Fm: Michael Johnston) Does anyone know if the @EXMEM and BANKER programs have been uploaded to Compuserve? I have the TMQ but I hate to type them all in. I also need the update fixes to the MC libs to include the @EXMEM functions.

(Fm: MISOSYS) The @EXMEM and BANKER program are part of the Copyright which MISOSYS has on its THE MISOSYS QUARTERLY. The only thing we upload onto the board presently are the fixes. File copies of any programs which appear in TMQ can be obtained from DISK NOTES unless the original author of the program is not MISOSYS and chooses to distribute the (his/her) program in another manner. I have not made any decision on uploading other parts of THE MISOSYS QUARTERLY other than the fixes.

On Family news...

(Fm: JACK LOTTEY) I just received my TMQ I.iii as I returned from my vacation. Ref the personal news in or out of "THE BLURB" I find it helpful to get to know a little more about you and your family. I, too, am a Philadelphian, from the Northeast...Frankford High; but left after college for a life scattered between NY and Central America. I am still not much more than an "end-user" but

feel reading the Forum correspondence has given me a few insights into the LDOS and Model III world. Thanks for everything!

(Fm: MISOSYS) We were a little further NE. Actually 0-10 was in Feltonville (C & Wyoming) and 10-20 was in Lexington Park (near Cottman and Blvd). I think most folks are not upset about the family news.

80 Micro <grins>

(Fm: Gary Phillips) Roy, I sent the following letter to 80 Micro today: "Dear Editors;

Your June issue would have left me laughing, except that the inaccuracies were so severe as to be painful.

If Jack Feldman's review of LDOS 5.3 is an example of what we can expect from your present editorial policies and staff, then I suspect users of Z80 equipment might as well quit reading your magazine now. Mr. Feldman makes statements about LDOS that amount to pure libel! I can only wonder whether he actually used the DOS or read the documentation before writing his article.

As a technically competent and experienced Z80 user (I am a professional systems programmer) I can assure Mr. Feldman that LDOS 5.3 in no way "puts your old disks at risk" and that the directory entries under this release are no longer than they were in the past. You can read and write diskettes from previous versions of LDOS (and TRSDOS 6) while running LDOS 5.3 with absolutely no hazard to the data or directories on those diskettes. The only problem is that the directory entries will be maintained in a manner compatible with the previous versions: no timestamp and no dates after December 31, 1987. You cannot inadvertently "destroy your disk if you do not follow instructions" and that is why the manual does not tell you that you can. (Unless of course Mr. Feldman is the sort that uses FORMAT when he means BACKUP, or switches the drive specifiers on a BACKUP command, or does any of the numerous things that can "destroy" disks in any DOS system. No manual can save you from yourself.)

LDOS 5.3 **is** indeed downward compatible with all previous releases of LDOS and of TRSDOS 6. Perhaps what Mr. Feldman really means to say is that previous releases are not **upward-compatible** with 5.3. Under certain limited circumstances involving password protection, LDOS 5.1.x will be unable to access files written on a diskette formatted by version 5.3. This can be prevented by not using password protection on files that must be read in this particular (and unusual) manner.

In reply to his complaint about "the lengthy routine necessary" to set up the RS232 driver or FORMS filter, I urge Mr. Feldman to read the documentation on the SYSGEN command, which is not a new feature. One only has to install such drivers once, if they are made permanent by the SYSGEN command. Thereafter they are automatically set up each time the system is booted.

You are doing your readers a terrible disservice when you print misleading reviews and incomplete information. The advice given in Mercedes Silver's "Feedback Loop" is frequently incomplete or dangerous. Likewise, many of the so-called "hints" that appear in the "Reader Forum" need to be screened by a competent person. I have seen advice given there that is potentially disastrous and could indeed destroy diskettes or even damage hardware. I was astonished to see that you paid someone ten dollars for instructions on using the REMOVE command of TRSDOS 6.2 that are fully explained right in the manual for that system! (Bill Boggess should look at his manual if he thinks he discovered something new when he began deleting more than one file with a single command line.) Jerry Semler's suggested "abbreviation" for controlling passwords from Basic is potentially dangerous: he is overlaying all 8 bits of a system control flag in order to change just one of them. Mike Zarowitz made the correct allowance for this in his original recommendation by using OR or AND to change only the desired bit. Altering the others could produce unexpected or unwanted results.

MSDOS and IBM copycats are dominating the microcomputer market today. I take it you intend to enhance this domination by spreading misinformation about the alternatives. Perhaps it is time you put a warning on your publication, 'Let the reader beware!'"

Recovering REMOVED files

(Fm: Jim Owens) Where can I find info on recovering "removed" files? I just had a major oops this morning on the 4p. Any help will truley be appreciated.

(**Fm: Bob Haynes**) Your files are probably recoverable IF you haven't invoked any writes to the disk in the meantime!

The process involves changing certain data in the directory sectors. There are articles dealing with this subject in the 10/85 and 7/86 issues of 80 Micro magazine which will help you. If you are not a subscriber, any large public library will have them.

The second alternative is to use a MISOSYS product called UNREMOVE which was part of their PRO-ESP package (discontinued) but now available within a larger package called MARK IV ($99). A bit steep just to fix a disk, I'll admit, but the many other useful tools included may make it worth considering.

TMQ Literary Section

(**Fm: Dave Hardy**) In order to further assure the immortality of LDOS, I propose the establishment, on an urgent basis, of a literary and epic poetry section for THE MISOSYS QUARTERLY.

To start the section following is the immortal epic of Don Scarberry from the December 1982 80 U.S.:

"I'm not a DOS lover,
  or a DOS lover's son.
But I'll use LDOS
  'til a better DOS comes."

Now if somebody can just figure out what rhymes with SOLTOFF!

Report from France

(**Fm: Michel Houde**) Please excuse this familiarity, but it is a real pleasure to meet you every quarter in TMQ. It is also a pleasure to deal with you. Your order servcing is FAST! Ten days is the usual time I have to wait when the Customs don't open the packet, a few days more when they do. It's actually faster than getting shirts and trousers for the kids from most mail-order firms here in FRANCE!

Further, I usually have the feeling that I get more than what I paid for. The latest instance is the RSHARD package, which included Model 4 software, although no mention was made of it on the flyer included with LDOS 5.3! By the way, I found 4 bugs in RSHARD6/DCT. While I am on bug reporting, I also found one in MAPPER, from PRO-MACH2, but the cure was easy.

I am pleased to tell you that SET2RAM works perfectly with MODELF/III, which is the French version of MODELA/III (the Germans have a MODELD/III). It is actually a modified version because the original, like your network ROMC, did not allow LDOS 5.1.4 to boot. Of course, I had to remove those mods that made the clock run correctly with 5.1.4.

I had to write KI4F/DVR to accomodate my French keyboard, but that's not a real problem as I've been doing it since Model I time. Of course, I did it for LS-DOS 6.3, THE SOURCE was a good help.

I've done my home work, to enable ALTDISK, NAME, UNREMOVE, all from PRO-ESP, to work with LS-DOS 6.3. I have even improved UNREMOVE to make it display mod time as well as mod date, when more than one REMOVEd files are found.

By now, you may wonder, who's that guy out there in France, doing all those things? Isn't he giving, sharing, selling software? I have never earned a penny, nor a Franc, from computer activity. I am a teacher-researcher in Chemical Engineering at Compiegne University. I use a Model 4P for word processing and data processing (we do a lot of modelization in Chemical Engineering). I prefer using BASIC on my Model 4P than FORTRAN on the VAX!

I am the one and only here in Compiegne, with a TRS-80 Model 4. But I do not feel too lonely, as I am a long time subscriber to 80 Micro, and to TMQ, of course! I know my computer intimately, and thanks to you, I can do what I want with it.

Please allow me to add a few words about my family. My wife is named Sylviane PULVIN; she is also a teacher-researcher at Compiegne University (Biochemistry). Like artists or journalists, female researchers usually keep their name when they get married. I wouldn't mind. She does not really like computing. The Mac is OK for her. We have three kids, one girl and two boys, aged 6, 3-1/2, 1-1/2. They like the Dancing Demon very much.

LDOS Information

80 MICRO review of LDOS 5.3

(Fm: MISOSYS) In response to the terribly inaccurate review of LDOS 5.3 which apeared in a recent issue of 80 MICROCOMPUTING, the following letter was submitted to their editor.

"Mark Reynolds, Review Editor, 80 MICROCOMPUTING

Rarely have I had the necessity of responding to an 80 Microcomputing review of a MISOSYS product. But the June 1987 review of our LDOS 5.3 had so many inaccuracies, that I am compelled to attempt a correction.

The statement, 'Logical Systems Inc., the company that developed LDOS, has once again updated LDOS and added some new features' is far from the truth. LSI did originally develop LDOS; however, LDOS has been a MISOSYS product since March 1986 and the entire 5.3 update was designed and prepared by MISOSYS. The product was announced in a press release printed in 80 Microcomputing in its April 1987 issue. LSI had nothing to do with LDOS 5.3.

Although author Feldman may think that the major change is the time stamping introduced into the directory structure, 5.3 was developed primarily to extend the directory dating to 1999. Time stamping was just an ancillary feature. There would have been no 5.3 release generated without the need to extend directory dating past 1987.

Feldman goes on to state that, 'Unfortunately, the new directory structure poses a potential danger to your LDOS 5.1.4 disks. The problem is that the individual directory entry, being longer, puts your old disks at risk.' That's just not true. The directory is exactly the same size - 32 bytes per directory entry. Besides, there are no dangers posed to 5.1.4 disks. Those disks can be read from or written to by LDOS 5.3 without any corruption whatsoever; to the files themselves or the directory. Of course, after December 31, 1987, the directory date of a 5.1.4 disk file would be incorrect if the file was modified while running 5.3 - but then that's the whole reason for updating. MISOSYS went to considerable lengths to design a date extension environment which was as transparent as possible. You can read/write from/to 5.1.4 disks with 5.3. When a 5.1.4 disk file is updated while under 5.3, the extended date and time fields are not touched on the 5.1.4 directory. It's really quite safe.

The DATECONV utility is only needed when you want to convert a 5.1.4 diskette to the 5.3 extended dating mode. Certainly after December 31, 1987, this issue will be moot if you are using LDOS. You'll need 5.3.

Feldman's final statements concerning 'the lengthy routine to set those parameters' when referring to the serial port and forms kind of left me puzzled. SETCOM uses an identical set of parameters as the SET command which originally installed the serial driver (RS232T). In fact, the serial driver defaults to 71E (that's 7-bit word length, 1 stop bit, parity EVEN, DTR=ON), a common setting for most folks with just a simple "SET *CL RS232T" command from LDOS Ready. That's lengthy? Besides, most users keep a few JCL files around to automate their device setups. MS-DOS users have found that the batch 'language' is extremely useful. It's been that way with LDOS since 1980.

Now even without a hardware clock that is keeping accurate time, relative time is just as important. The significant reason for keeping a time stamp is to enable checking which file written today is the most recent, when a file is written more than once. A week's use of time stamping will turn anyone into a convert. And TRS-80 time-keeping is not that inaccurate.

Perhaps the positive statements in Feldman's review attributed to the 4-star rating. On the other hand, the inaccurate statements concerning risks attributed to the changes in directory structure may cause a few folks to needlessly shy away from this important release of LDOS. And I can't imagine how the update was attributed to Logical Systems. Perhaps the personnel shufflings going on at 80 have resulted in a 'green' staff (no pun intended to Wayne)."

Old Passwords

(Fm: Adam Rubin) The changes in the password structure for 5.3 and 6.3, and the changes in 5.3's passwords, inspired some investigation.

Under 6.2 and earlier, there was a "user" password for the /SYS files. Those files had access of "none", so the user password made no difference anyway, but I recently figured out that it's (was?) LSINC.

For 5.1's passwords, the significance of GSLTD was relatively obvious, but does anyone know what RRW3 stands for? (Was LSI founded by two guys named R. and three named W. ?)

(Fm: LDOS Support jjkd) I hate to admit it, but RRW3 predates me. I'm sure it's a head-slapper. Roy?

(Fm: MISOSYS) "RRW3" came from Roy, Roger, and William. There were three of us. We were the original founders of LSI. Now who's Roger?

What's the version?

(Fm: Theodore Masterton) Ok. I have patched up my 5.3, sent in my cards, and am about ready to convert ALL my disks to the two new DOS's. One problem remains; I have never been careful about noting which DOS (5.1.4 or 6.2.1) I have used to format a disk, assumming they were identical. Now I find out that I can only DATECONV the disks using the same DOS they were formatted with! But I can't remember.

Is there an easy way to tell whether a disk was formatted with 5.1 or 6.2? I do have a track and sector type disk utility but I am not a hacker-type myself. Help please!

(Fm: Adam Rubin) There's at least two places on every 5.x/6.x disk: (1) BOOT/SYS -- Record 2, byte 0. (This is always track 0, sector 2). (2) DIR/SYS -- Record 0, byte CB hex. (This is the first sector of the directory cylinder). In both of these locations, you will find the value 51 hex, 62 hex, or something like that, which is a record of the DOS used to format the disk. "51" hex means 5.1.x, "62" hex means 6.2.x, and so on.

SET2RAM and memDISK driver

(Fm: Gary Phillips) Does the MEMDISK driver in your model 4 LDOS enhancement package work with memory expansions installed? (XLR8er and/or AlphaTech?) I'm planning to order the package anyway, for use on my non-XLR8er machine, but it occurs to me to ask, since I note that the LS-DOS MEMDISK/DCT now crashes the system under XLR8er. Found it out by accident when I invoked a pre-XLR8er JCL file.

(Fm: MISOSYS) To begin with, the MemDISK which is part of the Hardware Interface Kit only permits a 32K or 64K RAM disk using either one or both of the extra banks in a standard Model 4 (banks 2,3). Second, your Model 4 MemDISK should not crash at all if you have either an Alpha Tech board or an XLR8 board installed. MemDISK might crash if you have someone elses RAM drive software installed and you force Memdisk to install. On the other hand, some other RAM disk software should be locking out banks 2 and 3 if they are using it. I suspect some other problem local to your system.

(Fm: Gary Phillips) OK, I'll have to look at the MemDisk thing again. But I thought what happened to me was that if the XLR8er was set to fast speed, and FIXBANK was installed, then invoking SYSTEM (DRIVE=2,DRIVER="MEMDISK") produced a crash. In any case, I need KI4/DVR, so my order will be forthcoming when the tax refund arrives. (No, I'm not trying to use LS-DOS commands under LDOS, just referring to two different issues in the same paragraph.)

When you install a MemDisk driver under LDOS, where does it go in storage? Without a low-memory driver area, I assume it goes in high memory? Then it pages the alternate banks into the low memory area and buffers data back and forth?

(FM: MISOSYS) It goes in low memory. That's discussed in the docs which you get with the package. The package re-uses some of the area of ROMC which, of course, is RAM after the SET2RAM switchover. The cassette routines and the bootup code memory area is reused.

(Fm: Alan Kaplan) Just a couple of points about the interface kit: On page 5 of the documentation, when invoking the memdisk, memdisk should be in quotations, i.e. system (drive=d,driver="memdisk").

Also, when I invoke set2ram and later boot the system via the boot command, I get a "sys error" and then the ldos prompt. Othertimes, I get to the debug screen. After recovering from either of these events, if I boot the system again the computer hangs and I have to push the orange button to recover. I have a model 4d.

(Fm: MISOSYS) Yes, the memdisk installation should have ="memdisk". I thought I had already added that to the README.TXT file.

I'll double check [it was on April 28th]. Also, the BOOT command will be inoperative unless I patch SET2RAM to alter the first 5 or so bytes of the ROM image. Problem is that BOOT does a RST-0; however, in the RAM-image mode, that doesn't bring in the BOOT code since the ROM startup code has been de-activated. It's on my low-priority list to add such a patch. With SET2RAM installed, you have to push the ORANGE button to invoke a re-BOOT.

(**Fm: Gary Phillips**) Just wondering, Roy. The Model 4 hardware interface package arrived and like all your products, works beautifully. Now, since I'm using a 4P, can I install SET2RAM, then dump the ROM image to a file and avoid having to reinstall every time I boot up in model 3 mode? Once SET2RAM is applied to the 4P, what I would like to do is dump the memory image from 0000H through 37FFH to a file, call that file MODELA/III, and have the @BANK routines load into the system along with the ROM image rather than going through all those motions each time I boot. Does SET2RAM have to initialize itself with other parts of the DOS (this is probably a dumb question, how could it not?) or will the functions work if they become a permanent part of the image?

I am planning to use the software on my BBS, which runs on a 4P. I want to use the MEMDISK as a system disk for faster overlay operations. The less complex the bootup procedure can be made, the better, since I'd like the system to be able to reboot automatically after a power outage.

(**Fm: MISOSYS**) The ROM image on the 4P IS a file; that's the MODELA/III file. SET2RAM will work on a 4P; however, since it doesn't know it's a 4P, it still goes through the gyrations of copying the "ROM image" to higher RAM, switching to the RAM model III mode, then copying back the ROM image (which in the case of the 4P, was still in low RAM). However, the memory configuration is maintained in a byte that needs correction. Even if you would try to dump off a "new" ROM image which has the banking and memdisk handlers, you WOULD have to go through the motions because the new @BANK and @EXMEM vectors were used for the BOOT initialization. The reason is that the @BANK and @EXMEM handlers use the memory space and jump vectors which have to do with the ROM boot code. That also happens to be why you cannot (best not) do a software BOOT command when SET2RAM is installed. Suggest you use a startup/jcl to have your system "re-boot" after a power outrage (ooops, outage).

(**Fm: Gary Phillips**) I am having some unexpected difficulty with SET2RAM and MEMDISK in my environment. I am using a non-gate array 4P. Does it matter which version of the ROM image I load before invoking SET2RAM ? (I have at my disposal the one that came with TRSDOS 6.2, the one that came with LS-DOS 6.3, and one of my own generation which is a true copy of the ROMs in a model 3...)

The reason I ask is that I have experienced consistent "lock-ups" of the system in BASIC while running with the MEMDISK installed as a system disk (I install SET2RAM and MEMDISK, backup all the /SYS overlays onto the MEMDISK, then kill SYS0, SYS5, and SYS9 to make room for BACKUP/CMD as well. Then I do SYSTEM (SYSTEM=2) and a SYSTEM (SWAP=1,DRIVE=2), followed by SYSTEM (DRIVE=0,WP=YES).) DOS commands and utilities work fine in this environment, but BASIC seems to freeze after running for a while: keyboard is dead, the screen unchanging, no response from BREAK, and only a hardware RESET will recover. From the times at which this happens, my guess is that it strikes during garbage collection. I am using the BASIC that is supplied with LDOS 5.3, and have high-memory modules above approx. x'F400', all with appropriate headers. Any guess what might be going on?

Which ROM image did you use for testing on your 4P? (Or was development mainly done on the regular 4?)

(**Fm: MISOSYS**) TMQ I.iii has a two-byte patch to SET2RAM which corrects a problem if you use BASIC's TIME$. The patch is as follows:

```
PATCH SET2RAM (D00,50=00 38:F00,50=FF 37)
```

It's possible that some versions of the MODELA/III file may not operate without that patch. I should think that the "style" of 4P would be irrelevant. The MODELA/III file should likewise be irrelevant once that patch is installed. Could the use of TIME$ be the root of your problem?

(**Fm: Jim Hawes**) I think this is more documentation than code, but BOOT does not work when SET2RAM is in. For some time now I have had several Mod3 programs which either use a complex bank switch deal to get a doubler or copy the ROMs to RAM and go full RAM in the Mod3 mode. I have always found it necessary to hit the reset switch rather than issue the BOOT command. It's hardly worth a lot of worry and I am sure that anyone having

run in to the situation will find the orange button fairly quickly.

(Fm: MISOSYS) The reason why the BOOT command does not re-boot your Model 4 (in III mode) while operating under LDOS 5.3 and the SET2RAM auxiliary utility is due to the following: The BOOT command executes a RST 0 instruction. This, of course, transfers control to address 0. The code in the ROM image at address 0 turns off interrupts and then branches to the bootstrap code through vector 3015H. The technical specifications supplied with our Hardware Interface Kit mention that SET2RAM uses the address range specifically associated with the boot strap code. In fact, address 3015H is reused for the @EXMEM handler.

All is not lost. What needs to be done to "reclaim" the BOOT handler is to change the code at address 0 after SET2RAM is installed. Here's the old and proposed new:

```
-----old------        -------new------
DI                    DI
XOR     A             XOR     A
JP      3015H         OUT     (84H),A
JP      4000H         JP      3015H
                      NOP
```

I may look into altering SET2RAM to add this patch on the fly; however, I don't have it at this time.

RSHARD's ARCHIVE/RESTORE

(Fm: LDOS Support jjkd) Roy: the new MISOSYS hard drive support package, does the HD backup utility for large files preserve the LRL on a restored file? HARDCOPY/BAS does not, and even though this can be fixed via COPY (C=N), 'tis inconvenient.

(Fm: MISOSYS) Yes, it does. Part of the file's header record is the directory info. After the restored file is written to the targeted disk and closed, the directory is updated to match the 1st 20 bytes of the original DIR record. Thus, the LRL is preserved even if the restoral is to an old file of the same name with a different LRL.

Trouble with DEVICE?

(Fm: William Chao) Got a copy of LDOS 5.3 few weeks ago and now I have it running on my Mod.4 with a 15Meg hard-drive. I noticed something peculiar(sp??) whenever I issue a command such as DEVICE or DIR. When given DEVICE, it will show up all the logical drives (I have 8) but it can't finish in 1 screen so it pauses for the second screen, on the second screen, I would get the column header for the DIR command and then device would finish. On DIR, sometimes the column header would appear twice if pauses between screens and logical drives. Any fixes for this?

(Fm: MISOSYS) Double check to see if your SYS6 module is stored in more than two extents. If you backed up the new 5.3 to the old 5.1.4 on your hard drive, it's possible that the SYS6 module went to more than 2. Also check SYS7/SYS as it must also be in no more than 2 extents. That may be your DEVICE problem. It's possible that DIR could do that. I don't treat it as a bug.

RSHARD from JCL

(Fm: Jeffrey Kline) I finally got some time to sit down and work with my system in getting the new driver package installed (RSHARD6/DCT). The problems are as annoying as they come. First off, for some unknown reason, I cannot install the driver via a JCL. I keep getting an "INVALID ENTRY FROM JCL!" and the JCL stops. I tried every angle I could think of to make it function under the JCL configuration file I had created on the disk. The second is not too bad but I was hoping to be able to partition off the drive by head only as I don't really need to be concerned with the amount of directoy slots; I use DISKdisk and PDS. I finally wound up with a moderate configuration of 6 logical drives and partitioned by cylinder.

It would also seem that maybe the problem is only with TRSDOS 6.2 but not so. I also ran into the same problem under LDOS 5.3.0!. I wanted to not only keep a log of the configuration in text form but also use that very same text to re-configure in the event of a missing or damaged CONFIG/SYS file! Why can I not use a JCL to install them?

(Fm: MISOSYS) Because there was a bug in RSHARD. The fix files are in The Patch Corner. Murphy strikes again!

LDOS on Model 4 and cursor speed

(**Fm:** MISOSYS) There appears to be a large contingent of active TRS-80 users in Australia. I believe they number fourth in size based on our customer base (US, CANADA, GREAT BRITAIN, AUSTRALIA, ...). The "Adelaide Micro User News" is one of a few club publications sent to us. The June 1987 issue carried a review of LDOS 5.3 written by Rod Stevenson. One paragraph of Rod's review expressed puzzlement over clock speed and its relationship to cursor speed. Since I am in a prime position to clear up that puzzlement, Rod, this one's for you.

Before Tandy had finalized their implementation of the Model 4 hardware, one of the few requests for a design change submitted by LSI and actually accepted by Tandy was for a faster interrupt rate. The Model III interrupt occurs at a rate of 30 per second. Since a Model 4 has to be able to emulate a Model III exactly, its interrupt rate is also 30 per second when the machine is switched to the 2 Meg clock rate. When the Model 4 is switched to a 4 Meg clock rate, the interrupt rate is doubled to 60 per second. There is nothing magic about that; the rate changes when the clock speed changes.

Now a Model 4 can be operated under DOS 6.x (considered to be Model 4 mode) or under a Model III OS such as LDOS (considered to be then in Model III mode). When operating in the Model III mode, the first 14K of the machine's address space is taken up by ROMs which duplicate the Model III operating mode. Thus, when you boot a DOS such as LDOS, you are using the "Model III ROMs".

The Model III has routines in ROM which are more than just the BASIC interpreter. There are interrupt-based routines which keep the timer and time values as well as blink the cursor. The timer counts down from 30; this is hard coded and unchangeable. The cursor blink rate is also predetermined by a hard coded value. Anyone wanting to look at that code need only disassemble the ROM from 3529H through 3590H. The routine is accessed every time an interrupt occurs - unless something external to the routine traps interrupts and takes a different action. Since the video cursor counter counts down from 7 to 0, the cursor should change state once every 8 interrupts or blink once every 16 interrupts which is approximately every half second or two blinks per second. When the clock speed is changed to 60 per second without any external adjustment to the frequency with which the interrupt timer routine is called, the blink rate doubles because there are now 60 counts per second.

Any program which does not use the ROM blinking cursor routine (such as a word processor or text editor which is performing its own overstrike blinking) will be just as dependent on CPU clock speed since their code which controls blinking is invariably based on the speed with which their code executes; at the 4 Meg clock speed, their code should execute twice as fast (depending on CPU wait states). LDOS 5.1.4 booted up at a 4 Meg clock speed on a Model 4 because LSI had hard patched enabling the 4 Meg clock rate into the startup code. On the other hand, LDOS 5.1.4 had nothing added to it which would "correct" the doubling of the interrupt rate. In fact, the low and high speed slots of the task processor would be serviced at twice the rate expected. That's why the machine's CLOCK became worthless; it gained 8-12 hours every 12 hours.

Okay, what was done to 6.0 to make the interrupts uniform regardless of the system CPU clock speed? A time slice routine was added at the front end of the task processor. This little routine rotated a byte value of 55H (at 4Meg) or 0FFH (at 2Meg) and serviced all but the highest task only when the rotation placed a one bit into the carry flag. That's a very fast and short way to divide by two, reset the counter, and test the result all in one instruction [RRC (HL)]. Using that routine, all SYSTEM (FAST/SLOW) has to do besides toggling the hardware is to change the value of the time slice byte. One more thing taken care of in 6.x that couldn't be done in 5.x is the operation of the @PAUSE routine. In Model III mode, @PAUSE is totally in ROM and unchangeable (unless your on a 4P). Since code executes twice as fast (approximately) when in the 4 Meg clock speed, @PAUSE runs twice as fast and its pausing action is subsequently cut in half. The @PAUSE in 6.x adjusts for FAST/SLOW.

Okay, is the confusion cleared up now? CPU clock speed does indeed effect the blink rate of the cursor unless some action externalto the blink routine adjusts for the CPU clock/interrupt rate.

SYNONYM adapted for 5.3

(**Fm: Dave Lamkins**) This is a patch for one of my favorite utilities. You may use it in TMQ, if you wish. It is a patch for Henry Melton's

SYNONYM processor from LSI Journal V2n5p14. SYNONYM hooks into the overlay loader vector and looks for the @ERROR overlay number, which changed from X'86' in LDOS 5.1.x to X'96' in LDOS 5.3. The patch is easily applied as a command line patch.

```
PATCH SYNONYM (D01,76=96:F01,76=86)
```

Superlog and LDOS 5.3

(**Fm: Luis M. Garcia-Barrio**) SUPERLOG/JCL is a modified INSTALL/JCL for SUPERLOG, to suit the new structure of SYS0/SYS. Feel free to publish it in your magazine if you find it appropriate. [Editor's note: SUPERLOG/JCL will be printed in The Patch Corner].

KSM bug - for years!

(**Fm: Jim Hawes**) I am beginning the job of looking at LDOS 5.3. Most of what I see is valuable. I appreciate that someone is still interested in LDOS and the Z80 in spite of the IBM pressure and the industry (sub)standard mania. I truely hope the financial rewards are there for you. Hey that is pure selfish - if MISOSYS doesn't do well, I know I lose access to some good programs (and maybe some good bugs as well).

This has got to be in the "I can't believe it" category. Do you know or suspect anything about bugs in the KSM/FLT? I had truely meant to write you a note on this about 6 months ago. Please accept my apology - I just got busy with some other stuff and it got forgotten. These go all the way back to 1981. I see that you rewrote some parts of KSM/FLT and rearranged some stuff but the bugs are still there. They require highly improbable circumstances to show up. Both involve replacement of KSM data. Still after all this time someone should have reported it.

Bug one happens only when HIGH$ after installation of KSM is within 10 bytes of a page end. On re-installation, KSM looks for the proper place to put the CR replacement token. The IY register points to an image of OLD and NEWHI values obtained from the KSM module. The value at 4DFC is a pointer to this address group. Now the A register has just picked up the contents at 4DFD to test. If A is NZ you have: LD H,A and LD L,(IY+2). These load addresses refer to LDOS 514. About 24/25ths of the time this is OK but when the above HIGH$ value occurs, value in A and value in (IY+3) differ by one. The result is that HL points one page too high.

If you think that scenario was rare, you will have to agree that bug two is a winner in "most unlikely" contest. If you try to replace a KSM data set with a longer set, the installation checks the lengths and aborts right (not always!). If the new data is only one or two bytes longer, ZAP! Try this - build two /KSM files that differ in length by one byte. Install the short one and then attempt to replace it with the longer one: no error message. Replacement done - then the first use leaves smoke coming out of high RAM.

I went through all of this on the 5.3 version and ran off a bunch of appropriate screen prints and fixes. I am sure you will have no trouble in duplicating the test.

Now I could use a small favor. Could you offer some quick "documentation" on the use of the byte at 4758H. I pretty much get the gist but I would value your reply.

(**Fm: MISOSYS**) Can't believe category! Believe it! If no one writes or calls about a bug, especially an obscure one (rarely encountered), it may never get fixed. Since we acquired LDOS in March of last year, no one has given a bug report on 5.1.4's KSM facility. I can't say whether that was the case with either LSI or LSC, however, the code they sent us (5.1.3 source) and the 5.1.4 master did not provide any indication of a bug report. Can't fix what we don't know about.

I agree perfectly with your analysis of the two KSM bugs which have existed in KSM for more years than I have fingers on one hand. Sure, we revised some of KSM for 5.3. But the revisions were strictly related to the CFLAG checking (and copyright messages). On the other hand, I don't agree with your fix. I don't like X patches when I can introduce a D patch.

Your analysis of the first bug was correct. The DEVTBL stores a pointer to the PRMSAV region which is exactly 10 bytes higher than the "new HIGH$" pointer stored at PRMSAV+2. The initialization code picked up the high-order byte of the DEVTBL pointer and used the low-order byte of PRMSAV in an attempt to "point to" one byte before the KSM module in memory. It then added an offset calculated from the distance between the logical enter character and the module origin. This is using apples and oranges (!) as you pointed out. You

were also correct in demonstrating a correct solution to be the use of both the low and high order bytes from PRMSAV. However, your solution of changing the "LD A,H" with a "LD H,(IY+3)" wouldn't fit into the existing code space. The better solution would be to replace the "LD L,(IY+2)" with a "LD HL,(2*6+DEVTBL$)" and then change the offset to use the distance between the enter char and PRMSAV. This solution actually discloses about 4 bytes which could be NOP'd out. By doing the 16-bit load into HL to begin with instead of the 8-bit load of the high order pointer, the pointer is loaded to begin with! Of course, NOPing is unnecessary because the redundant code is not destructive.

I will use your second patch without alteration. You had the only correct solution (and the reason behind it, I might add - allowance for the NULL word at the top was overlooked). Incidentally, that NULL word is used to store a pointer to the continuation of a KSM string prematurely stopped because of a logical ENTER character. See KSMA/FIX in The Patch Corner.

And now for a word from our sponsor... Seriously, though, you asked about the CFLAG$ (byte at 4758H). An experienced DOS 6.x programmer should have no trouble with this one. However, I should provide some insight for the non-6.x'ers. We'll expand on this in TMQ I.iv, as necessary. Here's a brief intro.

Bit-1: says that @CMNDR is executing. In other words, when a program begins execution, it can test this bit. DIR uses it to force O=N because the sort buffer is not available. CMNDR is used to invoke "library" commands and other transients which run in the library overlay region. LDOS assumes this region to extend from 5200H-5FFFH. Any program which needs memory beyond 5FFFH can test the CMNDR bit to see if continued execution is feasible. Thus, programs ORG'd at 6000H can easily use any DOS library command.

Bit-2: says that SYS1 is requesting line input. That's the only time that the bit is set. MINIDOS's CMD-R uses this to sense if the key request is from "LDOS Ready". Keyboard filters which require system overlays other than SYS1 can use this bit to see if they have to force a reload of SYS1.

Bits-3-7 are adequately explained in THE MISOSYS QUARTERLY, Issue I.ii.

300 rpm, keyboards, and other matters

(Fm: Willi E. B. Wald) My LDOS 513 manual's index refers to a page 4-14A in the MINIDOS section, but there is only 4-14. Was 4-14A in an addendum which I missed?

I am 100% behind you on your need to go the MS-DOS route for your BUSINESS. I have bought my LAST computer from Tandy because of what they are doing to me as a Model I and Model IV customer. What you had to say about 80 Micro was also of great interest to me; they still owe me a T-shirt that was promised me with great fanfare in an editorial last year. Also, by SELECTIVELY quoting from my letter to the editor, Eric Maloney made me sound as if I were advocating software piracy! What I said was that Tandy OUGHT to give away with each computer purchase an assembler, word-processor and a spread-sheet program. If I were Tandy's marketing manager, that is how Tandy COMPUTERS would be sold because the software development cost for those programs had long been recovered and the incremental cost of producing an extra copy of the same program would come out of the advertising budget.

It is only now that I am beginning to learn how to use SAID. I was (am ?) apalled to find that SAIDINS completely ignores the <CTRL>, <F1>, <F2>, <CAPS>, and <F3> keys. For a while I thought of building a clamp for the <LEFT-SHIFT> key to enter text in upper case, but then TMQ3 came to the rescue and I found out about the <SHIFT><Ø> combination. That sure helps, and I have to learn how to use LDOS. I use my Model 4 mostly in the Model 3 mode because 64 chars per line are easier on 58 year old eyes! If your Model 4 hardware interface kit is half as good as I expect it to be, it is the one thing I have been waiting for! You are 100% right in NOT using the 80 x 24 chars in Model 3 mode.

I completely fail to see your point in calling 300.000 RPM a "disaster" in reference to Jack Decker (he no longer publishes Northern Bytes). I can see that such a degree of accuracy is technical overkill but it does NOT constitute a disaster!!! To me it is more of a disaster having to initiate KI/DVR and JKL (of Model I and NEWDOS fame) to be able to use the screen-print function which is already present in ROM.

Now please don't get mad at me, I merely want you to make your software more "USERFRIENDLY"! How I hate that word, but nothing else will do. In the interest of user-friendliness, would you consider issuing zaps or INSTALL/CMD

programs that would allow me maximum flexibility in customizing LDOS? And I do not mean SYSGENed High$ drivers. Such basics as JKL belong in LOW core memory where Tandy put them ditto the printer parameters, etc. Perhaps I have this NIGHTMARE about a hugh collection of essential drivers occupying HIGH RAM where they will be violated by every program which does not constantly check High$!! SMOOTH in low core perhaps?

I do not wish to re-invent the wheel, I am looking for better tires. In that spirit, I am using POWERSCRIPT. While it greatly improves SCRIPSIT, it is not as good as it easily could have been: no ASCII characters below 20 hex in their translation tables, and that is where the Model 4 has the European characters. And again, the extra keys are also ignored. It is enough to make this Tandy orphan weep!

On the one occassion when I used COMM I found its command syntax bizarre; to put it politely. Even if 5.3 Interface does not clear up 4-8 key sequence commands (excluding filenames), I am still game. And if some multiple keystroke commands still insist on <ARROW>-<LEFT-SHIFT>; my keyboard will then sprout a small toggle switch which allows me either 2 left shift or 1 right and 1 left shift keys! As you can se, I am certainly not a software engineer but I am determined to enjoy the use of my TRS-80.

Even as a hardware-hacker of some 45 years standing, I blanch at the thought of doing more brain-surgery on this computer. The expansion to 128K was easily done on the dining room table; but wholesale butchery of internal components and grafting of unmarked (sanded off) chips, with doubtful results, is not MY idea of adventure. I will wait for the arrival of an external 500K RAM drive.

One last remark: I have never been too fond of any DEBUG, they seem deliberately crippled. My monitors of choice are TASMON MICROMIND and MONITOR 5; I still have T-BUG. My next project has to be the study of various keybaord scans. Keep up the good work and don't change TMQ!

(Fm: MISOSYS) The 4-14A must have been a misprint. There is no Appendix which could have been referenced. The correct number should be 4-14.

A Model III computer keyboard does not have the <CTRL>, <Function keys>, nor <CAPS> key; thus, SAID - which is a Model III program - would not expect to have them available and thus does not use them. From your statement about SHIFT-Ø, I can almost guess that you do not have an LDOS User Manual. That kind of information would be in it in the section on the KI/DVR. It would also be in a Radio Shack Model III manual.

I believe the term which I used ala the 300 rpm issue was fiasco. In order to adequately support LDOS keyboard type ahead, the floppy disk driver keeps the CPU interrupts enabled until actual data transfer. The result is that a disk drive locked on to 300 rpm could take a very long period of time to perform the data transfer because 300 rpm is synchronous with the CPU system interrupt clock. When all of a sudden, users who have installed those PLL disk drives become infuriated with LDOS because data transfer is inordinately slow compared to the other DOSs, yes, I call that a fiasco.

You also don't have to initiate the KI/DVR and JKL to be able to use the screen print function! Where ever did you get that idea? You do have to specify the JKL parameter if you CHOOSE to use the enhanced keyboard driver provided by KI/DVR; however, the choice is yours. If you want to make use of <CLEAR> functions such as KSM, a full ASCII keyboard, and type ahead, then yes, you do need to install the LDOS keyboard driver. Since the screen print detection is an inherent part of the keyboard driver and because LDOS's screen print passes graphics values if you have SYSTEM (GRAPHIC) on, the ROM screen print function is worthless. But if you don't install KI/DVR, then you still can use it.

The "customizing" of a DOS is not a trivial manner. No, I don't think that an INSTALL command or commands would be feasible to make it more "friendly". Sorry, but Tandy didn't put the printer parameters in low memory. The code for SMOOTH happens to be within the floppy driver in the resident OS.

HIGH$ is supposed to be checked by EVERY program using memory. Would you drive off in your car (with better tires) without looking at how much gas you had in the tank? When I worked for AT&T, I had the necessity of using a company car occassionally. We were taught to use a safety check list each and every time we took out a car. The few minute inspection covered gas, lights horn, mirrors, seat belts, etc., and the old Lincoln test (that's shoving a penny into the tire tread to make sure Lincoln's head is still visible - does that work with your pennies?). Sure, a company may need a little more double checking when there

are multiple users of a vehicle; however, if everyone made these kinds of checks, you'd probably have less broken down vehicles and less accidents.

KI/DVR was designed for a Model III. The DOWN ARROW key on that keyboard (and on the non-clustered ARROW Model 4 keyboards) is adjacent to the LEFT SHIFT key. The use of LEFT-SHIFT-DOWN-ARROW for a CTRL key stems from the mechanical design of the keyboards in use on the machines (Model Is and IIIs) that LDOS was designed for. To change to the RIGHT-SHIFT-DOWN-ARROW for control, apply the patch: PATCH KI/DVR.DRIVER (D01,EC=4E:F01,EC=46). The KI4/DVR which is part of the Model 4 Hardware Interface Kit makes use of the Model 4's CTRL key rather than a 2-key combination; the Model I and III did not have a CTRL key!

The LCOMM keystroke interface was not designed for menu-based commands. If you care to utilize a "user-friendly" (read: menu driven) communications program, there are about four still on the market at costs ranging from $40 to $200. LCOMM is bundled as part of your DOS. You are not forced to use it; it is there if you choose to learn how to use it. There is no economic justification for totally revamping the LCOMM user interface to be "userfriendly" We are not selling a communications program with the DOS thrown in for the heck of it. A DOS is being sold with a usable communications program thrown in for the heck of it. Actually, it only takes a few moments to learn how to use LCOMM.

Just as you have found that there are many other debuggers around, you should have realized that a DOS can't be all things to all people. Most of the features ancillary to the specific function of a DOS (file management) are really not going to be elaborate. The economic justification is missing, the disk space is missing, the documentation space is missing. Ninety percent or more of the LDOS users don't write in assembler and thus have absolutely no need for DEBUG [99% of Tandy's customers would have absolutely no need for having an assembler bundled with their computers and those same customers would then consume 50% additional customer support trying to figure out what the assembler is for!]. That same percentage of users probably don't have modems and likewise don't use LCOMM. I would venture a guess that 90 percent of the Tandy users run no more than three programs - a word processor, a spreadsheet, and a data base.

Installing 5.3 on hard drives

(Fm: **John P. Lyman**) Recently we purchased LS-DOS 6.3 (from LSI) and LDOS 5.3 from MISOSYS. We appreciate the fact that for the very reasonable price we paid that you can not give individual assistance. However, I have tried to take your advice of 'read the manual' and 'don't bother us,' and after spending several hours in that mode, I feel I must bother you.

In addition to the TRSDOS system which I have successfully converted to LS-DOS 6.3 (their instructions were workable), I have two LDOS Hard Drive systems to upgrade (5 Meg/LDOS 5.1.3) and 15 Meg/LDOS 5.1.4) and a new Hard Drive (to be 15 Meg/LDOS 5.1.4) They are standard Radio Shack systems (HD 0, 1, 2, 3 and Floppy 4, 5). With Radio Shack turning their backs on us (TRS Model III/4 users) there is no one to turn to other than yourselves.

I have had no success trying to convert the new (LDOS 5.1.4) system using your very brief (and inadequate) instructions (page 3, 1/5/87) and I need help. I simply do not have the time to take a crash course in hard drive hardware to learn what your people already know, and I don't dare go near the existing systems with conversion in mind.

One other point! Given the (between the lines) attitude (LSI & MISOSYS) about helping people who don't pay their dues, I want to say that we have purchased the appropriate software with every piece of hardware we use, and we will continue to support those who are willing to support us when we need help.

Enclosed, please find a check for First Class handling of your MISOSYS Quarterly, but please keep in mind, How can we successfully convert LDOS 5.1.4 Hard Drive to 5.3.0 with the following configuration: 15 Megabyte Hard Drive 0, 1, 2, 3, and Floppy 4, 5?

(Fm: MISOSYS) Perhaps you read too much between the lines. You are correct in assuming that MISOSYS cannot afford to provide you with $50 worth of assistance when you bought only a $25 product. We would have gone broke a long time ago if we did that. On the other hand, I believe I should be able to point you in the right direction without you having to take a course in hard drives. What follows is some information which should have been provided to you by Radio Shack with your hard disk package and from information provided in both the LDOS user manual and the upgrade documentation.

First, there is no standard configuration of a hard drive. Even a 4-head 5 Megabyte Tandy hard drive can be arranged in one of about eight different configurations. The most typical configuration of a 5 Meg drive is as four partitions: each partition being one head. If your 5 Meg drive is already configured that way, then the LDOS 5.3 master disk contains a Job Control Language file which will install the new 5.3 release onto your existing drive. The name of the file is INIT54/JCL and assumes the use of the TRSHD3/DCT driver. If you are using some other driver, you will have to make any changes in the JCL responses to the questions asked by your driver. The INIT54/JCL file is mentioned in the README/TXT file.

There is no way I can discern the partitioning of your 15 Meg drive into 4 partitions; how do you divide 6 heads into 4 pieces? Even though your little printout shows the four partitions as 5Meg, 5Meg, 2-1/2Meg, and 2-1/2Meg, must I assume that's heads 1-2, 3-4, 5, and 6? When it comes to re-doing a configuration, one better not make any assumptions; one better know. Thus, I must say that you have to figure that one out yourself. You had to do it for LS-DOS 6.3 if you had to redo the configuration. The procedure for LDOS 5.3 should be quite similar; on the other hand, we provide instructions for doing that whereas Logical Systems had absolutely none.

Now here are some general words that may help you. A Radio Shack hard drive is installed using a driver. The driver may be named TRSHD3/DCT (an early R/S driver supporting only the 5 Meg drive), TRSHD5/DCT (a later Model III-mode driver supporting the 5-15 meg drives), or TRSHD6/DCT (a driver for the Model 4 mode). The TRSHD6 driver is equivalent to the TRSHD5 driver; they both support 5-15 Meg drives and ask the same questions.

Your first job to install a new DOS version into your currently configured system is to make a few backup copies of the new DOS. Since you may need some free space on the system diskette, you may have to delete one or two files. Do this only with one of the backups you will be using to create your boot disk. Next, make sure you know what the current configuration is. That means you need to know what heads (and how many) are currently assigned to each drive partition. You should also know the driver currently being used. If you don't know, how can you tell? The DEVICE command can be used to identify the starting head number of a "rigid" drive [when using the abovementioned drivers]. For the R/S driver, the number which appears after the word "rigid" in the display is the starting head number zero-based (i.e. a #2 indicates head 3). The FREE command also can be used to display the number of heads. Write this information down and keep it. Our MEMDIR utility can be used to display the names of resident high-memory modules - such as the hard disk drivers. But you may know what driver you are using.

A hard drive can be divided up into pieces known as "partitions". A partition is assigned to a logical drive slot (0-7) by means of the SYSTEM command. The exact syntax would be:

```
SYSTEM (DRIVE=d,DISABLE,DRIVER="TRSHD3")
```

[Let's briefly explain these parts. The "SYSTEM" designates the DOS command. The parameters are in parentheses. The "DRIVE" parm designates the logical drive slot to be assigned the HD partition. "DISABLE" tells the DOS to "disconnect" any disk driver currently assigned to the drive slot, if any. That's because the HD driver initialization will not allow you to assign a partition to a drive slot which is already enabled. The "DRIVER" parm specifies the name of the hard disk driver.]

Obviously, the driver name noted above would have to be changed if you are using the TRSHD5/DCT driver. Also, the drive number, 'd', assigned in the above command cannot be "Ø"; it must be in the range 1-7. If you want to set up a partition to be your system drive, drive Ø, you first have to assign it to some other drive slot then swap it later.

What happens next is the [installation part of the] driver begins asking you a lot of questions. The question sequence is identical for the TRSHD5 and TRSHD6 drivers but not the TRSHD3 driver. The TRSHD3 driver will ask you for the drive address. If you have one hard drive connected, that would be drive 1 [additional HDs would be 2, 3, and 4. This is actually determined by a jumper on the hard drive "bubble" itself. Whoever installed your hard drives should have them jumpered as 1, 2, 3, and 4 in that order; although any arrangement works if you tell the driver what is the jumpered number].

You next get asked the quantity of heads and the starting head for the partition assigned to the drive number identified in the SYSTEM command. The response to these questions should now be obvious [from the data you collected with the DEVICE and FREE commands

(or already knew)]. That takes care of the first partition. For each succeeding partition, you repeat the SYSTEM command noted above designating a different drive slot (i.e. 'd').

The TRSHD5/6 drivers will also ask you about the drive step rate (mostly always zero [or the smallest number given in the message prompt], the number of heads (the 5 Megger [usually] has 4, the 15 Megger has 6), and the number of tracks per surface (the 4-head 5-Meg has 153 whereas the 15 has 306).

When you are through assigning each partition to a distinct drive slot, you are almost home. From your descriptions of your current configurations, I suggest you assign the partitions to these drive slots: 4, 1 ,2, 3. Drive 4 is where your current hard disk SYSTEM drive is. You then BACKUP 5.3 to your hard drive SYSTEM partition via a command such as:

BACKUP :0 :4 (S,I)

If you have read the instructions in the upgrade documentation, you will note that I have not told you anything different here; it just takes a little thought as to what your current configuration is.

Here's a few more tips. Note from our documentation that the BASIC files are named BASIC/??? on 5.3 whereas on 5.1 they were LBASIC/???. This means that after the above BACKUP command, the old LBASIC files should be PURGEd. Also note that the upgrade documentation advises you at this point to install any other drivers or filters you normally use (such as the LDOS keyboard driver). That can be done with a:

SET *KI KI (TYPE,JKL)

At this point, you then can switch the system drive to be the hard drive system partition with a command such as (two are shown):

SYSTEM (DRIVE=0,SWAP=4) or
SYSTEM (SYSTEM=4)

which switches drive 0 and drive 4. Since you assigned a hard drive partition to drive 1, you now need to assign drive 5 to be your second floppy. This is done by issuing the command:

SYSTEM (DRIVE=5,DRIVER="MOD3")

The prompt will ask you for the drive I/O address 1-4. Your second drive is #2 so respond with a 2.

Now you get to save the configuration. Your "booting" floppy disk is still in your first floppy drive which is now numbered 4 because of the above SWAP. Issue the command:

SYSTEM (DRIVE=4,SYSGEN)

The configuration will be saved onto your booting floppy.

As my final remark, I will have expected you to write down in sequence, the answer to each question prompted during the installation. That's to make sure you can do it again. Type the commands and answers into a JCL file and you will never have to remember again what you typed; you'll only have to remember the name of the JCL file.

The JCL //EXIT gotcha

(Fm: **C. M. Mendenhall**) I have run into what appears to be a bug in LDOS 5.3 for the Lobo MAX80. Using BASIC, when I start a BASIC program using a JCL file to automatically load and run the program, the computer displays "JOB DONE" and branches out of the BASIC program back to LDOS whenever it receives an INPUT statement. Would you please send me a patch to correct this bug?

(Fm: **MISOSYS**) No patch required - no bug! I suspect that you ended your JCL file with either nothing or //EXIT. The correct way to terminate a JCL file when you want to remain within a program which uses line input (as does BASIC's INPUT) is to use the JCL macro, "//STOP". That terminates the executing JCL and returns control to the program. Omitting both //STOP and //EXIT is interpreted as //EXIT so that the LDOS Ready prompt will appear rather than having a "blind" line input request hanging without a visible prompt. When all else fails, read the manual.

TED Problems?

(Fm: **Arthur R. Wynott**) The text editor (TED) is not functioning properly. I cannot exit TED with <CLEAR SHIFT =>. <CONTROL U> sends the cursor to the beginning of text, not the previous page. Left and right arrows do not work, up arrow gives me a left bracket. Please repair.

(Fm: MISOSYS) Here's some answers to your TED problems. The TED on your LDOS disk worked perfectly for us. The three-key combination <CLEAR> <SHIFT> <=> worked for us to exit TED. The <CTRL> <U> key moved up a page in text (don't forget that a page is 22 lines). Also the LEFT and RIGHT arrow keys worked properly.

You gave me a clue when you said that the UP ARROW key gave you a left bracket. I don't recollect if you related that on the phone, however, it is a strong clue. TED uses the LDOS keyboard driver's Extended Cursor Mode (ECM). ECM can be invoked in one of two ways. There is a bit in the KFLAG$ which, if set, establishes ECM; that's what TED does. The ECM mode can also be toggled by depressing the three key combination, <CLEAR> <SHIFT> <SPACE>. The up arrow would generate a left bracket (5BH) if ECM were not engaged. Also, the arrow keys would not function for cursor movement if ECM were not engaged. Thus, I suspect that you had inadvertantly depressd that key combination and disengaged ECM.

On the other hand, even if ECM were not engaged, CTRL-U and the exit combo would still work. Better double check your procedures.

(Fm: Jane A. Layman) I very much enjoyed receiving the latest issue of THE MISOSYS QUARTERLY with the very welcome patches provided for both LDOS 5.3 and the LDOS 5.3 Model 4 Interface Kit. I have found, however, that even with the patch to TED/CMD, my cavalier editing habits can still cause me to lose the TED buffer contents. I'm just writing now because it took me this long to figure out how to reliably reproduce the phenomenon. It occurs equally reliably with the version of TED that comes with LS-DOS 6.3.

I lost the buffer contents with both the 5.3 and 6.3 DOS versions of TED when I did the following: (1) Load TED and type in (or load in) some text. (2) Delete one character in the text with <CTRL-D>. (3) Press <CTRL-L> to chain in another file from disk. (4) Type in filespec at prompt. (5) When the "Press <ENTER> to confirm" message appears (my understanding is that that message should NOT be appearing in conjunction with the <CTRL-L> command), press <ENTER>. Pressing enter causes the text in the buffer to be deleted. The filespec you typed in on the command line appears in its place followed by a drivespec.

(Fm: MISOSYS) Thanks very much for bringing that TED bug to my attention. You narrowed down the problem very well which enabled me to find the program bug quite easily.

I have developed a patch for both the Model III LDOS version and the Model 4 LS-DOS version. They are both only a one-byte change; the fixes are in The Patch Corner as TED5B/FIX and TED6C/FIX.

I have forwarded the Model 4 LS-DOS patch to Logical Systems Inc for their consideration. Incidentally, this problem should only have prevailed with the LOAD command, not the SAVE command.

Creating a NULL-length file

(Fm: Jack Lottey) I recall under LDOS 5.1.4 I could CREATE a file with Ø records. However, with LDOS 5.3, I always get a "bad parameter" msg when I try to create a file with Ø records. I am usually doing this in one of the sections of Profile III+ HD version, say "DA2" or "/DA3". What is the proper proceedure in an application like this.

(Fm: MISOSYS) You can't use CREATE to generate a NULL-length file.

(Fm: Jack Lottey) In the Profile III+ HD database I needed a file called: database/da3. Without that file the program would not open; the opening screen would note that that file was missing. In LDOS 5.1.4 I would enter CREATE database/da3:d (Records=0) and Voila, I had a file that opened up the whole PROFILE database. With LDOS 5.3, I have discovered that I can create a file if I specify "Records=5" and I suspect any other number but the command "CREATE GIFT1987/DA3:5 (RECORDS=0)" comes back with the response "Bad parameter".

(Fm: MISOSYS) It does that under LS-DOS 6.3 (and TRSDOS 6.2, as well). We made it compatible. CREATE can no longer create a NULL file, which is what you were trying to accomplish. You can just as easily create a NULL length file by the command, "BUILD GIFT1987/DA3:5 <ENTER>" then hitting BREAK as the next key entry.

(Fm: Ray Pelzer) An easy method is the old stand-by: BASIC. [The statement,] OPEN "R",1,"GIFT1987/DA3:5":CLOSE creates the empty file's directory entry with roughly the same number of keystrokes as the CREATE command (including the execution of BASIC).

LS-DOS Information

Fix for Tape100

(Fm: Paul Jaeger) I don't use tape100 very often, but occasionally I want to move a bunch of files from my model 4 to a mod100 cassette under batch control. When I switched to 6.3, tape100 stopped working. I noticed that the sound of the data was much higher in pitch coming from the 4 compared to the data from the 100. I can't figure out why 6.3 would screw it up, but here is the patch I arrived at (hours of experimentation) to adjust the cassette write timing in tape100:

```
.tape1001/fix
.Patch to adjust write timing for 6.3
.adjusts the wait timing between bit
. set/reset to cassette port
D06,0D=1A 16 18 03 11 32 2E
F06,0D=17 12 18 03 11 2F 2B
.Patch to remove "Cassette ready - ENTER"
D00,8D=00 00 00
F00,8D=CD 3B 2A
.Eop
```

The ratio of the two changed words seems to be important, so keep that in mind if you want to experiment.

(Fm: LDOS Support jjkd) I'm not surprised that you needed to alter the timing to make TAPE100 work. What is surprising is that it worked before and you needed to change it because of the LS-DOS 6.3 upgrade. TAPE100 has problems because it is sensitive to CPU speed. It must be modified depending if you are running on a M4 with zero, one or two wait states on M1.

(Fm: Shane Dawalt) Why couldn't a simple, but small, program be written into TAPE100 to check the wait states of the hardware TAPE100 is being run under? I downloaded a program from one of the DLs which checked the wait states and reported its findings. I'm sure a simple ML program could be written to do the same. After this program determined the # of wait states in the system, it could insert the appropriate timing values into the counter storage locations. (Hum, now I know why I couldn't get TAPE100 to work.)

(Fm: LDOS Support jjkd) That certainly is possible. A more elegant method in machine language would be to test how much work you can do in a fixed number of timer ticks (which will always occur at a fixed rate regardless of CPU wait states). This info can then be used to choose timer values that result in proper operation.

This problem has existed in TAPE100 since day one of TRSDOS 6.0.0, and Tandy never caught it or complained to LSI. We had about two direct inquiries in about two years, and thus never really considered it a big enough problem to fix.

Transferring data from a Mod100 is done far better via RS232, though the tape method could be useful for sending info from a Mod100 to a Mod4 via the mail or otherwise when you don't have both machines together at the same time.

BASIC & @SOUND

(Fm: Paul Jaeger) I have always felt that the sound command in basic and the @sound service was fairly useless because it was soooooo looooooong. Here is a patch for LS-DOS 6.3 to shorten the tones:

```
.sys001/fix to shorten the sound in 6.3.0
.patch sys0/sys.lsidos using sys001/fix
D10,83=00 00 00 00 00 00
F10,83=CB 24 CB 25 CB 21
.eop
```

The code is the same in 6.0, 6.1 and 6.2, although not in the same place, so you can apply the patch with a file editor just by locating the code CB 24 CB 25 CB 21. Don't forget that the patch will not become active until you re-boot since sys0 is only loaded at boot time.

I have been using this patch since the days of 6.0, so I think it is bugless.

Special Characters

(Fm: Paul Bradshaw) Using CHR$(28) will reset many things (inverse video, for example). To test whether the special characters or space compression codes are in effect do a print chr$(28);chr$(194);: Then test the POS(0) function of BASIC. If POS(0) returns a 1 then

the special characters are in effect, otherwise the space compression codes are in effect.

(Fm: **Pete Betz**) Ureka, you've got it! I blush to say that I never thought to look in cousin Roy's tome. Thanks to all. (Paul's method is pretty foxy too.)

But... not to be an ingrate, but I'm still only halfway there. After much experimenting I don't see any change in $DO when changing between SPECIAL characters and ALTERNATE characters, nor is this status included in the above mentioned documentation. Life will not be complete until I can distinguish between these modes too. Any Ideas?

(**Fm: Paul Bradshaw**) I'm not sure about this, so I'll look it up later and correct this if I'm wrong, or there's a better way to do it, but port x'EC' contains the bit that controls the SPECIAL/ALTERNATE character set switch. This can be retreived via an INP(255) as BIT 3 (1=alternate,0=non-alternate). This port image in maintained in one of the @FLAG$ bytes (MFLAG$ I believe). It is also bit three of that flag.

(Fm: **Pete Betz**) Looks like you've done it again! The INP(255) approach is giving me the desired information. I think you had the significance of the bit reversed -- I'm getting bit 3 SET for special characters and RESET for alternate characters. As one would expect, the setting holds even when you drop out into space compression. Very nice.

What I don't get now (probably a dumb question) is: why am I getting this information from an INP(255)? Both you and the Tech Manual refer to port x'EC', which translates to 236, not 255. In fact, I tried INP(236) first and got a big nothing. Why 255?

(Fm: MISOSYS) Probably because an input from port X'EC' resets the interrupt latch. You really don't want to extraneously reset that. The Tech Manual on the Model 4 notes that an "IN from ports FC-FF are a mirror of port EC". Of course, does anyone realize that the Model 4P ties port FF and port 90H together for WRITE? That's why a 4P will use the sound port for programs strobing the cassette port for sound output to a speaker/amplifier? We found that one when doing GOBBLER.

Disk driver function codes

(**Fm: Adam Rubin**) Serious interlude here! I have two questions about the "proper" method of interfacing with LS-DOS:

(1) How can a disk driver accept requests for additional functions? (I.e. other than the documented functions 0-15.) I've considered either using an unused register to pass the information, or using function values 16-255, but is there a better "official" way?

(2) The module headers of the system device drivers have useful data that's not obtainable anywhere else. I know that they shouldn't be accessed as absolute addresses, but is it "acceptable" to use @GTMOD and find the modules' data that way?

Obviously, there's no problem with either question under the "Radio Shack School of Programming", but I'd rather use the correct method if there is one.

(Fm: **LSI - Virgil**) Why not use the four unsupported functions (@DCINIT, @DCRES, @RDHDR, and @RDTRK), that were "provided in case utility software needs these requests for communications with custom drivers"? Quote is from The Programmers Guide to LDOS/TRSDOS Version 6 by Roy Soltoff. Do you need more that four additional functions?

As far as I know, offhand, the other four bits of B are not used for anything, but without having Bill here to ask, I don't have a practical way to verify this. He told me something about that once but I have forgotten. It seems like there was some reason for not using the top four bits, but I can't imagine what it would be now.

@GTMOD looks like it was designed for just that kind of stuff. Wouldn't it have been nice if Basic's VARPTR returned two pointers like that? Of course, the very nature of a function precluded that.

(Fm: LDOS Support jjkd) (1) Virgil's suggestion is a good one, though I would probably try to use subfunctions of a single function, perhaps @DCINIT, rather than grabbing all remaining functions.

(2) Access to module header data via @GTMOD is certainly kosher. The only problem is finding the documented structure of the header. Most

are documented in Roy's book, and can also be found in THE SOURCE.

(Fm: Les Mikesell) Are you planning to write more than one program that would use the new disk functions? If not, why not just imbed the functions in the program instead of using an SVC linkage? What kind of new functions do you have in mind? Obviously, nothing in the DOS is going to use them.

(Fm: Adam Rubin) Thanks for the information! Let's see...if I use function values above 15, that means there's no SVC for them, and so I'll have to duplicate the routine to find the DCT, swap in bank 0, and so on. It looks like it would be easier to implement new functions as @DCINIT, @DCRES, etc.

On the other hand, if new functions are implemented as subfunctions of (for example) @DCINIT, then any software which expects @DCINIT to initialize the disk controller may end up writing to the disk (my new function), with predictably unpredictable results. Doesn't sound too good either.

In this particular case, I'm tinkering with a floppy disk driver, which suggests two possibilities. One: extend the existing @RDSEC and @WRSEC to perform new (related) functions depending on bits 6 and 7 of the track and sector values. Two: use @HDFMT for my new "write" functions, and abort if it's actually a HD. Any thoughts on these?

Oh, and can I count on the NMI routine staying essentially the same in future versions (i.e. POP return address, RET), or should I stick my own routine into 66H-68H and restore it each time?

(Fm: MISOSYS) My suggestion would be to utilize the @RDHDR SVC. You could use HL as a pointer to a data region to pick up any set of values you need; useful since you are deciding your own protocol. @RDHDR has the advantage that no software could expect the DOS to suppport that function (since it doesn't). Any utility package actually needing a RDHDR function needs to write their own disk driver to support it.

As an aside, I had hoped to add an SVC to perform a disk function call using the value contained in the B register; but it was determined to be fluff overhead since most folks didn't need that facility. The only modules in the DOS which set up their own disk driver interface are BACKUP and FORMAT. They could have used the un-implimented SVC; but why use up another four bytes; memory is always at a premium. The existing disk driver interface architecture has served us well for a few years now.

IBM PC Diskettes

(Fm: Kevin R. Parris) I have what I think is a rather simple application: I want to read a sector from an IBM-PC single-sided diskette (180K, PC-DOS 2.1), using the factory original diskette drives in my TRS-80 Model 4 Portable, and a program running under 6.x TRSDOS (btw, jjkd, there is one possible solution to your concern about "trailing periods" in the text at the end of a sentence - put the version number in front of the name). I have THE SOURCE and THE PROGRAMMER'S GUIDE, and the MISOSYS PRO-MRAS assembler system (all purchased from MISOSYS), and a First Class Postage Subscription to TMQ; now, the GOTCHA.

I also have a great deal of ignorance as to the workings of Z80 processors and companion diskette drive controllers. I understand the IBM 370 processors and MVS Operating System well enough but this is relatively new to me. I have tried the (to me, at least) obvious things: @RDSEC says data record not found; @RDSSC turns on the drive light and never comes back, only way out is the RESET switch; @RDTRK, true to the documentation, does not read a track.

I do not want to invest in a commercial program right now- I do not need PC Disk directory handling, or multiple TRS80 Operating System format support, or BASIC token conversion. I just want to read a selected PC Disk sector so I can put the data from it in a TRSDOS file in the other drive. I have not studied it very closely yet (in The Source), but I do not see why @RDSEC locks up; I do know that the PC uses 9/512 sectors while TRSDOS uses 18/256, and suspect that has something to do with the problem. A way to implement support for @RDTRK would be OK- I could take the sectors apart (I think) without too much trouble. BTW, (NOT a Complaint, just curious) what were the reasons for not including @RDTRK support in 6.x TRSDOS? And if it were included, would it help me with this project? Any assistance will be greatly appreciated, you may be sure! I confess I have made little contribution here, and have asked several questions (most of them answered very well), but with so much help from so many

people, one day I will learn enough to do something really useful! Thanks, Everybody, in advance.

(Fm: Adam Rubin) As you may know, each sector on the floppy disk is preceded by a header, which gives (among other things) the sector number and the length of the sector (128, 256, 512, or 1024 bytes). First, let's look at how @WRSEC works, and why it will work on an MS-DOS diskette.

Once the floppy disk controller (FDC) has found the requested sector, it also knows that sector's length. It then asks the CPU for the data, a byte at at time, until it's been given all the data for that sector (based on the length value in the sector header). Once the FDC has been given all the data, it sends out a signal to tell the computer that it's done. In the M4, this signal is a non-maskable interrupt (NMI). Anyway, this (i.e. @WRSEC) works properly for any valid sector length.

What @RDSEC does in 6.x is assume that every sector is 256 bytes long, so the CPU asks the FDC for data 256 times, and then assumes the read is done. However, if the sector is longer than 256 bytes, the FDC realizes that the CPU hasn't asked for all of the sector's data and (correctly) signals a 'lost data' error. 6.x is designed to retry a lost data error indefinitely, and of course each time is another lost data error. Thus, it keeps trying, until you press RESET.

I don't know why @RDSEC is implemented this way. I once asked jjkd, and he said it was needed because of the interrupt structure, but I don't see why that would matter for reading and not for writing.

If @RDSEC were implemented in the same manner as @WRSEC, adding 'read sector header' and 'read track' would have been fairly straightforward. However (as I learned from a discussion here), @RDTRK isn't very useful for reading sector data. (Nothing to do with TRS-80s; it's part of the FDC's design.) The FDC interprets several valid data values as sync bytes (is that the right name?), assumes that it's at the beginning of a sector, and the rest of that sector's data is garbled. (Remember that data is stored on disk as a stream of bits, not bytes.)

(Fm: Les Mikesell) The Trsdos 6 floppy @rdsec is hard coded to stop reading after 256 bytes, so if you want to read 512 byte msdos disks, you will have to write a substitute. RDTRK is not generally considered reliable in DDEN (and is not needed by the OS), so it is not implimented.

The infinite retry on lost data errors is to allow interrupts to be kept on between finding the sector header and the start of the data. This makes it fairly likely that the read will be interrupted by the heartbeat clock, and since the disk revolution speed is an even multiple of the interrupt rate, if it does it once it will take a while to recover. I suspect the 256 byte limit was imposed by someone thinking about the chance of accidentally reading a 1024 byte sector into the DOS 256 byte buffer. However, the lockup from the retries is not much better. The write code lets the FDC interrupt out when the command completes because it must also handle @wrtrk.

(Fm: Gary Phillips) You need Machine Language Disk I/O and Other Mysteries by Michael Haguer, published by IJG, if you want to learn to program disk controller operations at the assembly language level. It IS possible to read IBM PC diskettes in a model 4 or even a model 3. Commercial software to do this is available from HyperSoft and/or PowerSoft at a very reasonable price ($50. see 80-Micro ads).

IBM diskette formats use 512-byte sectors, while TRSDOS/LS-DOS/LDOS uses 256-byte sectors, so not all the disk i/o primitives will work directly. I once read PC-DOS diskettes on a model 3 using only my bare hands and DEBUG, but a small zap had to be made to the DOS so it would read the full 512 bytes. Now I use SuperCross and I'm glad I do!

@RDTRK is not very effective with the Western Digital controllers and standard floppy disk formats. Although there is a hardware command code that tells the controller to read an entire track, due to the way synchronization is handled at the hardware level the only guaranteed correct data you will retrieve is the sector address marks. The rest is usually "phase-shifted" and appears as garbage. Such a function is useful for determining the structure of an unknown alien diskette, but useless for actually reading the data located thereon. I would have expected @RDSEC to work, if you have coded your program correctly.

(Fm: Adam Rubin) In this case, the two topics (adding disk driver functions, reading 512-byte sectors) are related, since the program

I'm working on (a looong-term project!) will copy files to/from MS-DOS disks. To read 512-byte sectors, I followed the method LS-DOS uses for writing sectors (i.e. loop until NMI), and haven't noticed any problems. That's why I wonder about the DOS's method. (I admit reading 1024-byte sectors into a 256-byte buffer isn't a good idea!)

Once I had @RDSEC working well enough, the only additions needed for @RDHDR and @RDTRK were the appropriate FDC commands in place of "read sector". In the 'format MS-DOS disk' section, I'm using @RDHDR to determine if the disk is already formatted, and thought that the multiple-sector read and write commands would be a quick way to fill all the sectors with 0F6H and then verify them. I can't write 0F6H using @WRTRK, and figured that the sector layout of SS MS-DOS disks (every track is 1,2,3,...9 -- is that 'no track skew, no interleave'?) would otherwise require 18 revolutions to read and verify all the sectors on a track. I realize the multiple-sector commands lose bytes, but that doesn't matter in this case, does it?

Anyway, I don't have anything else in mind that would use @RDHDR or @RDTRK, but figured it would be easier to implement the whole thing as a disk driver in case I ever wanted to use them in the future. What exists now is the MSfile program installs and removes my driver each time the program is run, so my program can use @RDSEC regardless of which drive is being accessed. (In fact, @WRSEC and some other requests are passed on to $FD.)

Including the driver directly in the program (which I've sort of done, I suppose) might be the best way, and I'll certainly have to think about it. Do you, or anyone else, have any comments on anything here?

(Fm: LDOS Support jjkd) @RDTRK would not be what you want. It does not work very well in the WD FDC controllers, especially in double density. Nothing in the OS needs it, so it would have been excess baggage that wouldn't work anyway.

You basically need to code a replacement for @RDSEC as noted in the other replies. Note that MS-DOS sectors are numbered from one to nine, where LDOS/LS-DOS sectors are numbered from zero to seventeen. Trying to read sector zero on an MS-DOS disk won't get you very far at all.

There is no interleave on many MS-DOS disks because real IBM DOS on a real IBM PC (or close enough) can keep up with 1:1 interleave on a multiple sector I/O request.

One trick you can use to eliminate the problem is to write and read verify the sectors in the following sequence 1, 3, 5, 7, 9, 2, 4, 6, 8 instead of 1, 2, 3, 4, 5, 6, 7, 8, 9. You'd be doing the interleave in software instead of by the format, as in some releases of CP/M.

(Fm: Les Mikesell) The best approach for multi-sector i/o would be to make up your own interleave table for the order of sector access (jumping around in the buffer as necessary). This would give a speed-up during file access as well as the format and verify. You might kick around the idea of making a resident driver that would let TRSDOS read/write directly to MSDOS formatted disks. This could be done by faking the i/o to the TRSDOS dir cylinder while actually writing the MSDOS directory. I don't know if it could be done in a reasonable amount of memory, though.

(Fm: Adam Rubin) I never thought of that before...but I'll do it! Let's see...four revolutions per track, times forty tracks, is 32 seconds, which sounds fast enough to me.

An interleave table would definitely be an improvement, and I'll use it in the format routine.

One of the first things I considered was that many files need some sort of conversion to be usable when transferred, such as CR to CR/LF. I decided to pass the file through a filter as it's moved, so (CONV="TEXT") on the command line would load TEXT/CNV and call it for each byte read. With this byte-at-a-time approach, I don't know how much faster multiple-sector I/O would be for file access. Also, I'm not sure how useful direct access to "unconverted" MS-DOS files would be. What do you think?

(Fm: Les Mikesell) If you are at all concerned with speed, use full sector I/O on the TRSDOS disk and read at least 1K (a track would be better) at at time. Then do any necessary conversions in memory before writing out to the MSDOS disk using the interleave table. It would probably also be worthwhile to buffer as much as possible in memory (even multiple files), and allow swapping disks in the same drive. Remember that most TRS80's only have two floppies. As for the file conversions, we

use the LDOS version of Wordstar and often need binary conversions between these disks and MSDOS since the file format is identical. Most other files need the CR/LF conversions, though.

(Fm: Adam Rubin) Well, I moved a few files around, and timed them, and you are absolutely right. I'm already using a 1-sector buffer for both the source and destination, so expanding both to an entire track should be fairly simple. Swapping disks in a single drive would be useful too, but something like COPY's (X) parameter (with appropriate overlays SYSRESed) would serve the same purpose and probably be easier to implement.

It turns out there's something else that's taking up almost half the time of an MS-DOS to TRSDOS transfer -- allocation. "Programmer's Guide" has an example of that, and I suppose the value in the MS-DOS directory would be a good guess for its length, so that doesn't sound too hard either.

So...that oughta keep me busy for a little bit, at least. Thanks again...I'll be back with my next few dozen questions on this soon!

(Fm: LDOS Support jjkd) Transferring files in a "raw" or "binary" mode as an option is indeed useful. Any movement of an .?Q?, .LBR or now .ARC file, for example.

What is the format of the CNV file? Is it actually executable code? Perhaps a translation table definition file might be easier to change, even though not as flexible. If the translation table approach is used, an input format like MAXLATE's to allow multiple byte output for an input byte would ease tasks like CR/CR+LF.

(Fm: Les Mikesell) Yes, it is also worthwhile to write the last sector first on the TRSDOS side so all the disk space will be allocated at once instead of accessing the directory for each gran. Writing dummy data will also work if the real data for that sector is not available yet.

(Fm: Adam Rubin) I have a related question or three. (Don't I always? <grin>) I discovered that the DOS floppy disk driver uses the "directory cylinder" value to determine when to start write precompensation, so my MS-DOS/TRSDOS file moving project uses 20 (14H) for the MS-DOS disk's "directory cylinder". (It uses the DOS @WRSEC, and its @RDSEC updates the "current cylinder" value.) Since some other DCT values are altered, the program sticks 0 in the "directory cylinder" value when it's done, to force logging the next time the drive's accessed.

Questions: Is cylinder 20 a good place to start write precompensation on an MS-DOS disk? Will what I am doing cause any problems? Is it all necessary? Is there a better way?

FORMS questions

(Fm: Dick Houston) In using the FORMS command with say a BASIC program, it works OK for the first page, but merely starts out in the middle of a page or wherever it left off on the last run when I run it again. Does anybody know how to reset the line counter?

(Fm: Ray Pelzer) I may have missed a discussion of this once before but maybe someone can tell me if the following was an intentional design or a missed bug:

When the FORMS filter is installed and you attempt to set the MARGIN, you will get a "Parameter Error" if the CHARS setting is OFF. I was messing with a file I wanted to print at 12 pitch and loaded FORMS to shift the printout over 10 characters rather than messing with physically shifting my forms in the printer. Got the error message upon issuing "FORMS (M=10)", but "FORMS (C=132,M=10)" worked just fine. The REAL kicker is that AFTER loading up the margin, I could go back and do a "FORMS (C=OFF)" and the margin stayed loaded properly!

Am I losing my mind, or is it already long gone?

(Fm: MISOSYS) That's how its supposed to work. You must have a chars-per-line value before setting a left margin.

(Fm: Alan H. Pesetsky) I'm using the forms command to set the number of lines printed per page. If I interrupt the printout and then reset my paper to the top of the page manually or by the printer Form Feed button how do I tell Forms/flt to reset to line 1 for the next print out? (I'm not using basic - printing out the PROWAM CARD/DAT file.) Hope I just didn't miss that bit of information in the manual!

(**Fm: Paul Bradshaw**) Sending a CHR$(6) will reset FORMS's internal page counter. To do this easily (since you mentioned that you have PRO-WAM) use the program PRCTRL.APP that you can find in DL2 here in LDOS. Or use PRCTRL.CMD. Also, if you get The Misosys Quarterly, there was a program in the first issue (and duplicated in the second issue) that allowed you to send characters to your printer by specifing their ASCII codes.

(**Fm:** MISOSYS) You don't use the FORMS command to reset the line counter. The command LPRINT CHR$(6); will reset the system line counter to top of page.

(**Fm: LDOS Support jjkd**) If you use KSMPlus, you can advance the paper to the next perf by using <clear><shift><T> instead of manually or at the printer. That will keep the printer and FORMS in sync.

If you can get out to DOS, or to someplace you can issue a FORMS command, re-issuing a FORMS command with the same number of lines should also re-set the counter. Might be easier than getting to where you can do a CHR$(6);

(**Fm: Les Mikesell**) Printing a CHR$(6) will reset the FORMS line counter, but if you are able to do that, why not just send a form-feed in the first place instead of moving the paper manually?

(**Fm: Alan H. Pesetsky**) Using PRCTRL/CMD I tried sending @6 to the printer to reset the internal FORMS page counter -- didn't seem to work. I have been using PRCTRL/CMD (never got around to getting the /app version) to send a form feed and then would set the paper to the top of the page manually. Was wondering how to do it the other way round.

I assume that LPRINT CHR$(6); only works from basic. Was looking for a solution from DOS.

KSMPLUS is a good idea, also re-issuing my forms set-up sound even more convenient as its in a /jcl file. This problem came up when I was printing out my Pronto card/dat file in various ways. I usually have KSMPLUS disabled then because my current KSM file using the function keys which conflict with PRONTO. Thanks to all for your suggestions

When I think of it that's what I do. But sometimes I interrupt the printing -- move the paper up to tear something off - then reset the paper, etc. Just looking for a way to make sure that the computer knows that it's the top of the page now!

(@*2Fm: Gary Phillips@*0) From DOS level you can always try: COPY *KI *PR [followed by] <CTRL-F><SHIFT-CTRL-@>. That should achieve the same results as the LPRINT CHR$(6). Clumsy but it works.

(**Fm: Paul Bradshaw**) I have no problem sending a CHR$(6) to FORMS using PRCTRL. If you wish to reset the printer to top of form, use the PRCTRL keyword "SETTOF" -- this will reset the printers counter. A chr$(6) will set the FORMS internal counter, and all will be in sync.

(**Fm:** LDOS **Support jjkd**) Your problem is that you are manually advancing the paper to tear it off. Don't. Your BASIC program can advance the paper via the command, **LPRINT CHR$(12);**, and then you tear it off. This will keep all five (the printer, the paper, the FORMS filter, your program and you) happy and well synchronized. If you insist on manually advancing the paper, you can put a **LPRINT CHR$(6);** in your program wherever you want the FORMS line counter reset to the top of the page. Then however, it is the responsibility of the operator to check and confirm the paper position whenever this is used.

As a final gotcha, if you are using a printer with hardware formfeed capability (FORMS with FFHARD), this is not sufficient. You need to send a **LPRINT CHR$(12);** to the printer, it will advance to where it thinks the top is, and then prompt the user to manually advance the paper to the proper position to force synchronization.

The MARGIN parameter requires the CHARACTERS per line parameter in an attempt for the filter to know when to apply the margin to lines that wrap the right margin. This, taken to its logical conclusion, would require that setting C to zero also set M to zero. This must have been overlooked. If you do for some bizarre reason not want wrapped lines to also be indented to the margin, we can look upon this as a fortutious occurence.

Accessing the RS232 port from BASIC

(Fm: LDOS Support jjkd) Welcome to the LDOS/TRSDOS 6 Forum! This here is the most experienced group of folks you'll ever find when it comes to LDOS, TRSDOS 6 and the Z80 family of Tandy computers.

As I recall, the question was RS232 output and perhaps input in BASIC. For example, from TRSDOS 6, before entering BASIC, do a

```
SET *CL COM/DVR
```

This installs the OS support for the serial port. Now, from inside BASIC, you can do this:

```
10 SYSTEM"SETCOM
(W=7,P=Y,EVEN,DTR,BREAK=0)"
```

or whatever other parameters you need to talk to the device in question. You may also change these on the fly as necessary simply by repeating this line as needed with the appropriate changes. To actually go ahead and produce output, you can do this:

```
20 OPEN"O",1,"*CL"
30 PRINT #1, "This is a test"
```

Remember that BASIC will provide a CR after this output, but no linefeed. If you want to provide output to both the display and the serial port, one at a time, you can do this:

```
100 GOSUB 1100 ' Initalize for output
110 DEV = 1:A$="This will go to the
RS232":gosub 1000
120 DEV = 2:A$="This will go to the
display":gosub 1000

1000 PRINT # DEV, A$:return
1100 OPEN"O",1,"*CL":OPEN"O",2,"*DO"
```

A minor modification of this will allow you to do either output with no CR, output with both CR and LF, or output to both devices. When you've digested this, we'll go on to serial input.

The problem with input is that the byte I/O routines in BASIC are hosed up. There are two choices:

(1) Link the keyboard to the RS232. You can then scan for RS232 input via INKEY$. The problem with this is that you can't distinguish between keyboard input and RS232 input, but is usually adequate for bulletin board programs and the like.

(2) Use at least a little assembler for input. With the tools available in the new 6.3.0 release, you can use the SVC interface USR call without really writing any assembler at all.

LS-DOS 6.3 questions

(Tim Clute) (1). Why wasn't QFB/cmd added instead of the new DISKCOPY/CMD? (2). What is the reason for the distribution disk only having 2 floppies enabled? Love 6.3 hope that some of the suggestions make it into 6.3.1 or 6.4 <grin><hint>

(Fm: MISOSYS) Only 2 floppies are enabled on the 6.3 distribution disk because only 2 were enabled on 6.2. Most Model 4 folks have only two floppy drives. If drives 2 and 3 were enabled, then there would be some delay during global search for the system to ascertain that you had no drive 2 nor 3. Thus, it makes sense to disable 2&3 on release. That speeds up global search for most users. It's quite easy to ENABLE 2 and/or 3 if you need to using the SYSTEM command.

I believe that LSI chose to implement a "DISKCOPY" instead of the more recognizable "QFB" for a few reasons. One, they licensed a "QFB" to Tandy in the Utilities Disk sold by Tandy under license from LSI (that may be the prime reason). Second, "diskcopy", at least in name, is the same name as a similar utility under MS-DOS. I personally would have opted for them to call it QFB since LSI had already bundled a "QFB" with LDOS 5.1.4. They had already established the function association with a name. It should have remained "QFB".

(Fm: LDOS Support jjkd) The functions of QFB and DISKCOPY are virtually identical. The reason for the name and appearance change are most likely that LSI licensed a (poorly marketed by RS too) version to Tandy, and DISKCOPY is what the equivalent tool is called in MS-DOS. I know, I know, who cares about MS-DOS, but anything that reduces the sheer mass of command differences when there is no major overriding need for a difference ~~is a plus~~ in my book.

DEBUG Help

(Fm: Ken Kane) I can't find that thread on GOOD/BAD docs...but the Model 4 manual gets a LOW rating for its docs and examples for

DEBUG. The 'E' command is poorly explained. Hunt and peck, (about an hour of work but oh well...) shows that the address must be followed by a comma. No mention thereof, and no example. Quality control, anyone?

I am running through the SVC examples at the back of the Tech Ref Manual. Example pgm B has a series of prompts and @KEYINs and they keep grabbing control of my single stepping. My next 'I' entry gets handled as if it were a response rather than a DEBUG command. Is there a ready way to regain control?

Another way to look at the problem...in DEBUG, what is the easiest way to simulate keyboard entry at a prompt and continue single stepping?

(Fm: MISOSYS) The best way to satisfy KEYIN response is to use the 'C' debug command. When the KEYIN is called, a "C" will be left on the screen. If you are looking at the screen, that "C" will remind you that you are keying in. If it's a @KEY call, then you have to slowly use the 'C' command, otherwise you will get the result you mentioned. Don't forget that 'C' works exactly like 'I' for everything but CALLs. What you really need is to have PRO-DD&T's Debug Disassembler resident so that the screen display also includes the current instruction. Of course, PRO-DD&T is no longer available as a separate item but is available only bundled in the MARK IV Collection.

(Fm: LSI - Virgil) I'd aggree that the 'e' command of the extended debugger is not explained as well as it could be. Just using the following for the example would have helped.

Eaddress,byte

BTW, <spacebar> works as well as <comma> for the delimiter between the address and the new contents.

Cursor strangeness

(Fm: LSI - Virgil) There was a bug in the previous versions of BASIC that has been fixed. Trouble is that it was such nice bug and a lot of people miss it. When execution of a program in basic is ended, and you are returned back to the BASIC Ready prompt, BASIC is supposed to tidy a few details up, just in case the program left a mess. One of those details is to turn the cursor on. So when you do a PRINT CHR$(15); from the BASIC Ready prompt, it looks like nothing happens. Indeed the net result is that you are back where you started. But put exactly the same statement inside a program, followed by a statement that should present a cursor, INPUT A$ for example, and the cursor is gone! Then comes the end of the program and the BASIC Ready prompt, complete with cursor.

The only case that I can think of that is still a problem is if the break key is hit, interrupting the program for some reason, and the program is resumed with the CONT command. CONT does not and should not turn the cursor off, even if it was off before the BREAK. Of course, the programs that have to survive a BREAK could easily turn the cursor off more often, just in case.

One-character device names

(Fm: Shane Dawalt) I was SETting a filter into memory under LS-DOS 6.3, but I messed up on the devspec. Instead of typing "*SS" I entered "*S". The DOS didn't complain so I didn't investigate it. I filtered the *KI using *S and this too worked fine. However, when I executed the DEVICE command, the device map was screwed up. This is what it showed: (Note: the DOS was freshly booted, no other programs had been executed prior to this SETting/FILTERing.)

```
*KI <#  [*S*DO <=> X'0B88'
*PR  => X'0E01'
*SI <#  *KI
*SO <=> *DO
*JL  => NIL
*S
```

The filter module set to *S worked perfectly. It filtered the *KI device as it should have. The device table just doesn't show it correctly (notice *S has no route/link information behind it). Also notice the *S*DO. Looks as if the *KI data line terminated without a x'0D' before listing *DO. I was able to duplicate this problem by routing a device (*PR) to another 1 character device name (*Z). (*Z had been previously routed to a file, the filename wasn't displayed in the DEVICE listing.) Again, the route worked fine.

According to the operations manual, the devspec must be two characters. Why doesn't LS-DOS fuss when given a one character devspec?

(Fm: LDOS Support jjkd) If the docs say two characters required, and DEVICE barfs on a single character devicespec, then it does indeed make sense that SET shouldn't allow anything other than two characters. In my book, I'd call this a bug, even though it probably has always worked this way. Do you have a copy of 6.2 handy to try?

(Fm: Shane Dawalt) I don't have a copy of 6.2.1 but I do have a copy of 6.2.0 (isn't that strange). I attempted the same device setup on the 6.2.0 system and the same results were found.

I wrote a program which did some fetching of device specs from the DCB. I never considered the fact that only 1 character could be in a devspec (since I didn't think that could occur). The program stuffs the two characters inside of a message string before displaying the message with the @DSPLY SVC. I noticed that when the program comes to a 1 character devspec, the message up to and including the first character of the devspec is displayed, but subsequent characters of the line are not. The cursor is positioned at the "end" of the line as if an ETX had been encountered, but actually a '00'h is encountered. But that raises another question: doesn't @DSPLY terminate the displaying of a line when either a CR or ETX is found? This implies any other control character is displayed without termination of the print line.

(Fm: LDOS Support jjkd) As I suspected, this is not a problem created by 6.3.0, as it has been there all along, most likely since 6.0.0 without anybody finding it. I imagine that Virgil will put it on the list for future attention, but I certainly wouldn't consider this something bad enough to force a new release.

As far as @DSPLY goes, yes, the documented termination characters are ETX and CR, and anything else below X'20' should be output for display. I can see taking a zero as a terminator also, it is a useful construct. It was most likely just never documented.

Even though you can think of a zero as both X'00' and as ^@, I don't consider it as a "real" control character. Far too many things treat a binary zero as a special case. For example, it does have a special meaning to the *DO device that I'm not sure is documented anywhere. Who all knows what it is?

(Fm: MISOSYS) Taking a zero as a terminator is against the use of a zero to imply that the next character has a value of 0-31 and should be displayed instead of being treated as a control character. Of course, the bug with that would be that the @DSPLY handler won't trap and pass the next character if it turns out to be a ETX or CR.

(Fm: LDOS Support jjkd) Allowing a X'00' to terminate an @DSPLY string output would not preclude the ability to use a X'00' with @DSP or @PUT, of course. And, of course, it should either be fixed or documented, depending on whether it is considered "broke" or "mis-documented", right Virgil?

(Fm: MISOSYS) I doubt it immensely to see LSI change @DSPLY so that a NULL will terminate the string. One reason is that historically, only two values have terminated the string: CR and ETX. Although passing a NULL actually does something on the Model 4 implementation, I believe that that operation is machine dependent (i.e. a Model 4 "feature" and not part of the DOS per se). Another reason is that any change such as that would require additional memory for its implementation and thus would be out of the question. Personally, I don't think that @DSPLY should ignore the next character in the string after a NULL. I also don't think anyone is taking advantage of the lower 31 characters (32 on a Model 4P since a NULL-NULL will print a character as there is a character in the 00 position of a 4P's character generator) for display purposes through @dsply. Maybe through @DSP, though. So who is supposed to document it, Tandy or LSI?

(Fm: Bryan Headley) Tandy did document it, in the Tech Manual. Over in the appendices which deal with the character set, it addresses the business about character Ø. Machine dependency. (Although I still laugh at the circumstance that brought the question to the head)

(Fm: LDOS Support jjkd) In my opinion, the only problem is that SET accepts a single character devicespec. If you fix that, the fact that DEVICE hoses up on a single character spec may or may not be considered necessary to fix, because you'll plug the only documented hole that could result in a single character devicespec. Of course, a program that mucks with the device table directly

Could still insert one, but then it's their fault.

I'm not suggesting a change, that could certainly barf up something that's already out there. My suggestion is that if indeed the @DSPLY routine has changed to terminate on a zero, as somebody mentioned they observed it to, that it should either be fixed or the documentation changed to reflect the new behavior. (yes, I did actually put a u in there, but saw it and deleted it before sending <<grin>>)

Assuming that the doc is changed because this is an "official" alteration from the spec, LSI should do the documenting.

(Fm: MISOSYS) To my knowledge, @DSPLY has not been changed to terminate on NULL. It's not in THE SOURCE, and LSI stated that low core was not touched.

(Fm: MISOSYS) Folks, I just passed along to Bill today (29-May-87) a little patch which would correct the DEVICE command of 6.3 (if "correct" is the proper word). What it does is to transfer only the first character of the device name if the second character is NULL. Here it is:

```
D1F,D4=ED A0 7E B7 C8
F1F,D4=7E 12 13 2C 7E
```

The patch, of course, is to SYS6/SYS.LSIDOS. That's really the only place that should have any problem with one-character device names - if you really think that it is a problem. Also, @DSPLY has not been changed, it does not terminate on a NULL. I believe what happened to the "strange display" from the nulled 2nd character was that the 1-char device name was the last thing on the line. The following CR was then picked up by the @DSP driver to designate as displaying rather than as control.

Hard Drive Problems

(Fm: LDOS Support jjkd) Let me mention that if you currently have the physical floppy drive zero in as logical drive zero, there are only two methods of getting the floppy addressed elsewhere.

First off, if you have the system files installed on some logical partition of the [hard] drive, you can use SYSTEM (SYSTEM=n), where n is the number of the logical drive to be used as the system drive. Typically you'll want to set up the hard drive "bent" slightly so that it comes out straight after doing this command.

As an example, [here's a drive setup] before:

Logical	Physical
0	Floppy zero
1	HD partition two
2	HD partition three
3	HD partition four
4	HD partition five
5	HD partition six
6	HD partition one

with system files on the HD partition one. After a SYSTEM (SYSTEM=6)

Logical	Physical
0	HD partition one
1	HD partition two
2	HD partition three
3	HD partition four
4	HD partition five
5	HD partition six
6	floppy zero

To bring in logical drive seven as your second floppy, you can then execute a

```
SYSTEM (DRIVE=7,DRIVER="FLOPPY",DISABLE)
```

and answer the (1-4) prompt with "2". Finally, a

```
SYSGEN (D=6)
```

will get you a boot disk that will produce this configuration, given a bootable system disk in the first floppy drive.

Keyboard Filters

(Fm: Bob Haynes) I feel very foolish bringing this up, as I bet I'm overlooking something stupid! Nevertheless, here goes:

I'm experimenting with a small piece of code which will eventually be used to filter *KI under 6.2/6.3. Init code/memory header are properly established. I'm trying to intercept keyboard chars under certain conditions and pass them to the *CL device as well as back through the *KI chain. Here's the code:

```
SVC     MACRO   #NUM
        LD      A,#NUM
        RST     40
        ENDM
```

```
; preserve DCB link if present
START   PUSH    IX
; get DCB pointer (from header)
RX      LD      IX,(MODDCB)
; call the *KI driver
        SVC     @CHNIO
; restore link
        POP     IX
; exit if no char
        RET     NZ
; (abbrev'd for benefit of this mssg)
        PUSH    HL,DE,BC,AF,AF
; get *CL DCB into DE
        LD      DE,'LC'
        SVC     @GTDCB
        EX      DE,HL
; get char into C; send to *CL
        POP     AF
        LD      C,A
        SVC     @PUT
; restore regs (again abbrev'd)
        POP     AF,BC,DE,HL
        RET
```

Of course there's no error trapping yet, and the @GTDCB isn't needed for every loop thru the chain, but as I say, this IS skeleton. It works for all chars except <ENTER> and <BREAK>. When either of those are pressed, it crashes. I suspect the KFLAG$ scanner, but I'm not familiar with what is involved at the Task level. Thought KFLAG$ scanner would be transparent here, since all it does is update the KFLAG$ bits, which I don't care about, since there's no need to sense and act on BREAK/ENTER other than to pass the char's value via @PUT. Any suggestions?

(Fm: MISOSYS) Since @GTDCB does disk I/O (may need a system overlay), it would be better to isolate that into the installation code to extract the address of the *CL DCB, then stuff the address into your resident filter code. That just makes good programming sense.

(Fm: Bob Haynes) My concern is when the *KI filter remains relatively static and resident in configuration, but the *CL chain is more dynamic (filtering, etc). If filtering or redirection was applied to *CL AFTER my filter was applied to *KI, I believe the *KI filter would access the wrong *CL vector with your suggestion.

What I've done is set up the code to trap certain command characters, and if appropriate, perform a @GTDCB, then pull SYS1 back in via your technique outlined on pg 181 of your Programmer's Guide. 'Tis almost finished, just need to adjust a few error traps and set up a relocation routine for the hard addresses. Then I THINK I'm done! ('cept for the debugging, of course...). Would appreciate any further comments you might have.

(Fm: MISOSYS) Absolutely wrong. I said use @GTDCB to get the address of the *CL device control block. Once *CL is installed into the device table, it's DCB address never gets changed. You used the @GTDCB in your filter code after it was installed. That's wrong. Use the @GTDCB SVC in your installation code and plug the address returned by it into your resident filter code as if that address was just returned by @GTDCB.

(Fm: LDOS Support jjkd) You've got it, and I think that you have the fix too, no? Your code can check the currently resident system overlay. Any device I/O that causes a resident SYS1 to be overwritten by anything else is responsible for reloading SYS1 before releasing control. That basically means any file I/O, open or close, along with @GTMOD, @GTDCB, @CHKDSK, @RAMDIR and so on.

Note that this applies to resident drivers and filters, anything resident hooked into keyboard byte I/O. And finally, even though not related to this problem, I'd like to mention that RTC tasks can't do any disk I/O (but can hook through @KITSK).

(Fm: Les Mikesell) If you want to output to the unfiltered com/dvr, you could use @gtmod to find $CL. It takes a little more set-up to call the driver directly instead of using @put, but it would certainly be more efficient.

(Fm: Bob Haynes) And now here I stand, my hat in my hand, Eating some humble pie... Your system well planned, the DOS support grand, How silly I feel, said I!

Re your comments on needing @GTDCB withing a filter module, you are, of course, correct (as usual)! I was confusing the vector's VALUE with its ADDRESS. The @GTDCB call is now dutifully installed within the init code where it belongs. Thanks, Roy, appreciate your patience...

(Fm: MISOSYS) That's what we're here for.

The LSI LS-DOS Column

Hello again from LSI. Again in this column I would like to point out that any material contained herein is the opinion of LSI and NOT that of MISOSYS. Any comments regarding the content of this column should be directed to LSI at PO Box 55235, Grand Junction Co. 81505 (303) 243-7070.

This time, I want to stress the importance of sending in your registration cards. Response has been fair, but note that we will have to be very strict in applying the rules in the future; UNREGISTERED USERS WILL RECEIVE NO SUPPORT. Tandy will probably have mailed out update notices to registered TRSDOS 6.2 owners by now and all of those users, that don't read 80-Micro or TMQ, or check into the LDOS forum on Compuserve, will finally be finding out about LS-DOS 6.3. We expect to be rather busy here at LSI, filling those orders. The demands on the customer service department will probably go up accordingly. To assure that the Customer Service departments time is spent on the right people, we will have to verify that we have received your registration card. All registration cards are entered into our database computer on the day received and are available in just a few seconds since they are indexed by the Customer Service ID#. If you have that number ready when you call the customer service department, we can deal with your problem quickly, with no wasted time.

Some people are having difficulty finding their Customer Service ID#. Actually, it is quite simple, but the instructions just don't get the message across to everyone. Just type in ID <ENTER> from the LS-DOS Ready prompt. I am amazed at the number of people that send in the registration card with the serial number instead of the Customer Service ID#. They are very different and serve different purposes. The Customer Service ID# is a must on ALL communications with the LSI customer service department. Now, with 20/20 hindsight, I wish we had used "CS" instead of "ID" for the command to get the Customer Service ID#. That may have prevented some of the confusion.

While I am covering some of the basics, let me answer one more common question. Why is the TRSDOS62 or "DISK NAME" that appears after the drive designator in a directory listing, of so many disks, not changed in the LS-DOS 6.3 update process? That field is, of course, a user maintained and created field. It can be changed to whatever you want with the ATTRIB command and should not be tampered with by an update utility.

An interesting tidbit about Basic has been brought up by Hardin Brothers (author for 80 Micro). It turns out that there is a restriction on the name of the array that you use to pass the parameters to the SVC when using the USR11 access feature in BASIC on 6.3. The name of the array can not be more that two characters long. There isn't much need for a longer name for this purpose anyway, so it is no big deal (unless you have spent all day trying to make a longer name work). It is all a consequence of the way VARPTR works. Remember that the syntax is USR11(VARPTR(ARRAY(0))). Well, VARPTR returns the address of the beginning of the data not the beginning of the header and the length of the header depends on the length of the name and the number of dimensions, and the dimensions and the data type. The problem is that the fields in the header that give the information needed to calculate how long the header is are at the OTHER end of the header. USR11 does (as it should) check that the type of the array is integer, but to know where the type information is, it has to assume that the array name is less than three characters long and that the number of dimensions is one. It is just too cumbersome to do the checking any other way. The other choice would be to do no checking at all. I do have a patch that disables the checking for those of you that think a longer array name is worth the effort. Write to the Customer Service department if you want it.

There have been several requests for LSI to support 720K 3.5" disks on the Model 4. Because of several hardware differences, it would require that a completely new driver be written to support these drives. LSI does not have any plans to write a driver for 3.5" drives. It is hard to see much chance to break even, let alone make a profit, on that project. But if anyone out there tackles the project, we wish you luck. The physical configuration for 3.5" drives doesn't seem to be "standardized" yet, a driver that might work well for one manufacturer's drive may not work at all on another drive. Just going to 720K probably doesn't justify the expense of new drives, but if they start getting 4 or 5 meg on each disk, reliably and inexpensively, then they will have something.

HYPERSOFT'S announced new Model 4 emulator, PC-Four, for PC-compatible machines is intriguing. At the time of writing this, I haven't seen or heard anything about it except

the ad in July 80-MICRO. LSI considered doing something like it, but never thought a software only approach would work very well. The 8088 just isn't faster than a Z80 such that it could emulate a Z80 and take care of the MS-DOS environment without a significant loss of speed. There seems to be some demand for something like this, and LSI is not going to fulfill it, so we are glad to see that HYPERSOFT has moved into this area. I will look into this product and discuss it at length in my next column.

Where do we go from here? LSI has had frequent requests to make a good DOS for PC hardware. The trouble with that is the limitations of the PC hardware, which is already approaching its own obsolescence and a need to be MSDOS compatible which would defeat the whole purpose. The PC came out BEFORE the Model 4, remember? The trick is to write a really good DOS for the machine that hasn't been built yet. Although a better DOS could easily be done for the IBM world, (for One or Two Million dollars), we don't feel that it would be economically feasible. So for now, we wait. Now where did I put that crystal ball? ...

I am working on a major new project; it is for the IBM world of users though, and it is only for users who invest in mutual funds. The design is almost complete now and I have started the coding. It should be ready in a year or so. If any of you are involved with mutual funds and would be interested in a system that completely handles and monitors your holdings drop me a letter or give me a call I'll tell you all about it.

We knew it would happen! There are some patches out there on some BBS that claim to be able to fix the date handling in TRSDOS 6.2. A word of caution: If it could be fixed adequately with patches, we would have done it that way. The patches that we have seen just alter the base date from 1980 and deal with only about half of the other things that need to be done to keep all of the operating system working correctly. And the range of dates remains only 8 years. For example, to get 1988, you lose 1980. Please don't get involved with this mess if you don't have to.

There was an error in the patches in our last column. Don't panic! I think everyone just fixed it on the fly without giving it a second thought. What we did was to give the same name to two of the patch files. Both patch #002 and #009 were given DCOPY1/FIX and #009 should have been DCOPY2/FIX, as it was referenced in MAKE63I/JCL. Sorry about the oversight. If you didn't notice it at the time, you should go back and verify that patches #002 and #009 are both installed.

There are four new patches. All are very minor but a new level has been assigned to keep things orderly. The fifth patch is just to change the level designator in the boot banner. Remember that different level letters are assigned according to whether you put in the patches yourself or LSI does it by recutting your master. All of these patches are very short and could easily be done from the command line, but they are presented here in JCL format for the benefit of Disk Notes and this same file will be put into Data Library 0 (and/or DL6) of the LDOS forum on CompuServe.

Make that five new patches plus the level change. Roy just called with a bug that he just found (and he fixed). Under some very unusual circumstances, DIR refuses to read in an extended directory record because it thinks it already has it when it doesn't.

To install the patches included here you must first create each PATCH file exactly as shown as an ASCII file. Then save it with the patch name specified in the patch. Each physical line shown must be terminated with a <ENTER>. Each individual patch has been separated by a line of = signs. The /JCL file that follows will properly install all these patches. After all files have been created do a: DO MAKE63K (D=d,S=s) <ENTER> where the "s" in the example is replaced by the Source (where the /FIX files are) and "d" is replaced by the Destination (where a COPY of your Master 6.3 disk is).

LSI patches to LS-DOS 6.3

```
==============================================================================
.*** MAKE63K/JCL ***
.-JCL- Procedure to create a LS-DOS 6.3.0 Level - "K" from a level "I" or "J"
. NOTE: there are two version of BOOT3/FIX. Use the correct one for the
. level that you have.
. NOTE: Destination to be patched and Source of patch file must be given on
. JCL command line, ie - DO MAKE63K/JCL (D=d,S=d) <ENTER>
. Replace the "d" in example with correct drive numbers
```

```
PATCH SYS0/SYS.LSIDOS:#D# using SYS01/FIX:#S#
PATCH TED/CMD.UTILITY:#D# using TED3/FIX:#S#
PATCH SYS6/SYS.LSIDOS:#D# using DEVICE1/FIX:#S#
PATCH DISKCOPY/CMD.UTILITY:#D# using DCOPY3/FIX:#S#
PATCH BOOT/SYS.LSIDOS:#D# using BOOT3/FIX:#S#
PATCH SYS6/SYS.LSIDOS:#D# using DIR1/FIX:#S#
.
.  ******************************************************
.  **** Your LS-DOS is now a version 6.3.0 / Level - K ****
.  ******************************************************
.END
```

Here are the current patches that are applied by the above JCL.

```
==============================================================================
. ** SYS01/FIX - 02/20/87 - For Level J - SYS0/SYS.LSIDOS - - - - PATCH #011
. This patch will cause the day of week byte at DATE$+4 to be set
. to 0 during the load of SYS0, correcting the state of the LEAP BIT.
. This patch to be applied to SYS0/SYS.LSIDOS - File dated before 07/01/87
X'0036'=00
==============================================================================
. ** TED3/FIX - 05/22/87 - For Level J - TED/CMD.UTILITY - - - - PATCH #012
. This patch corrects a problem that occurred when another file is loaded
. immediately after deleting one character with <CTL> D.
. This patch to be applied to TED/CMD.UTILITY - File dated before 07/01/87
D04,B1=AC;F04,B1=8A
==============================================================================
. ** DEVICE1/FIX - 05/29/87 - For Level J - SYS6/SYS.LSIDOS - - PATCH #013
. This patch enables DEVICE to function correctly when fed a one character
. device name, which is forbidden in the manual but not actually prevented
. by the operating system.
. This patch to be applied to SYS6/SYS.LSIDOS - File dated before 07/01/87
D1F,D4=ED A0 7E B7 C8;F1F,D4=7E 12 13 2C 7E
==============================================================================
.** DCOPY3/FIX - 06/05/87 For Level J - DISKCOPY/CMD.UTILITY - PATCH #014
. This patch will correct DISKCOPY to properly deal with software
. write protected source drives.
. This patch to be applied to DISKCOPY/CMD.UTILITY - File dated before 07/01/87
D03,A8=78;F03,A8=F8
==============================================================================
.** BOOT3/FIX - 06/09/87 - For Level I - BOOT/SYS.LSIDOS - - - - PATCH #015
. This patch will cause the display LEVEL of 6.3.0 to change from "I" to "K"
. This patch to be applied to BOOT/SYS.LSIDOS - File dated 12/01/86.
D02,1E=4B;F02,1E=49
==============================================================================
.** BOOT3/FIX - 06/09/87 - For Level J - BOOT/SYS.LSIDOS - - - - PATCH #015
. This patch will cause the display LEVEL of 6.3.0 to change from "J" to "K"
. This patch to be applied to BOOT/SYS.LSIDOS - File dated 12/01/86.
D02,1E=4B;F02,1E=4A
==============================================================================
.** DIR1/FIX - 07/01/87 - For Level J - SYS6/SYS.LSIDOS - - - - PATCH #016
. This patch will correct the handling of a very rare exception by DIR.
. This patch to be applied to SYS6/SYS.LSIDOS - file dated 02/10/87.
D09,C9=FF;F09,C9=00
==============================================================================
```

MS-DOS Information

Tandy 2000 - general

(Fm: Steve Jensen) I have recently acquired a Tandy 2000 with 256K. My experience is with IBM PC's and my software is all in that format. My questions -- Can I run any of it on the 2000? How can I expand the RAM without paying a fortune? Are there third-party hard drives that are not similarly prohibitively expensive? I appreciate any information you can give me about the 2000.

(**Fm: LDOS Support jjkd**) Well-behaved IBM applications will run fine on the T2K. This means screen output via DOS or BIOS, not directly to the hardware. Any other "fancy" program that uses anything other than BIOS and DOS resources is going to be a problem. WordPerfect works fine, for example, but the IBM version of MS-Word wouldn't have a chance.

Extra memory from Tandy is not bad at all, around $170 for the board. You can save 20% by going mail order. That comes with 128K of chips, you can add two IBM PC 64K sets to bring you to 512K. The board can be modified to use 256K chips, giving you 768K. A commercial board for 896K total is available, but expensive at around $500. The same vendor also has a 20 meg HD kit for a bit less than Tandy's 10 meg unit.

(**Fm: Steve Jensen**) Thank you for the quick response. It is good to know that IBM software will run on the 2000, but is there not a read problem the disk drives - I understand they format differently. Also, who is the commercial vendor you mentioned? Is it possible to modify the board to accept 256k chips as a non-tech or must this be done by a computer technician? Thanks again for the information.

(Fm: **LDOS Support**) No, be very careful! The opposite is true, the majority of IBM programs will not run, due to the use of direct video access and/or hardware access. Virtually anything that "pops up", for example, is immediately ruled out. WordPerfect is one of the lucky exceptions.

The T2K will read and write either 360K or 720K floppies, though some care needs to be taken when writing 360K disks.

The vendor I'm thinking of is Envision Designs, you can find references to them over in the TRS-80 Pro Forum, go PCS21, in either DL 9 or DL 10, I forget which. They also used to advertise in 80micro, I haven't looked recently.

Most of the boards I've heard modified by non-techs have been problems, it isn't something I'd recommend to somebody who isn't at least reasonably familiar with digital logic, anti-static techniques and soldering on multi-layer boards.

(**Fm: MISOSYS**)There is a newsletter called Tandy 2000 Orphans which may be of assistance. The address is 387 Main St., Westport CT 06880. "Envision Design" mentioned by Joe is at 1539 West Pearl St., Pasco WA 99301 [509-547-1139].

Clear to end of frame

(Fm: **Nate Salsbury**) Does anyone know a command in GW BASIC that will **partially** clear the screen from the present cursor position to the EOS (lower RH corner.)

In M4 BASIC, the command "PRINT CHR$(31);" would clear from where the cursor was at that moment to the EOS. In GW BASIC I find PRINT CHR$(12) which clears the WHOLE screen - same as CLS. Not what I want.

(Fm: **LDOS Support jjkd**) I don't think there is one. A kludge would be:

```
1000 print string$(80-POS(0),32);
```

but that type of thing can give you troubles with wrapping the right margin and scrolling the screen unless you always stop one short, which can in turn cause problems with not clearing the last column. I think that you'd get an error back from the string$ function if the cursor happened to be in the last column.

If you were not working in BASIC, I'd say use ANSI.SYS, but BASIC ignores the ANSI.SYS driver.

(**Fm: JIM OWENS**) Joe, What do you mean? I only have a vague understanding of what the ansi driver does. In fact I know only two things about it. (1) Some programs need it; and (2) it slows down execution if it is not needed.

So Basic does not use it any way? Well darn it Joe don't you see what you've done? You have opened up a whole **new** set of questions for me to ask! (grin)

(Fm: LDOS Support jjkd) The ANSI driver is an attempt to provide a "Standard" interface for programs to use. This is the equivalent of using the SVCs and the *DO device in the Model 4 environment. Unfortunately, Microsoft is their own worst enemy, and they generally refuse to follow their own guidelines when producing their products. The result? Virtually all MS-DOS BASICs ignore ANSI.SYS and go directly to the display hardware.

(Fm: Les Mikesell) Did you try using ANSI.SYS, then opening CON: and printing the control sequences to it? (Just a thought, I haven't tried that either..). If it does work, BASIC will probably then forget the current cursor location.

(Fm: Nate Salsbury) I solved MY problem about clearing to the end of screen with a ML program which I can merge into any GWBASIC program. I just put the details into DL 0 with the name KLEER.ASM. Perchance, it might be of use to you with slight modification. I found that printing a string of blanks was fast enough for my purposes (to just clear one line).

LDOS/MSDOS Utility

(Fm: Ed Boudrie) Is it possible to transfer files from LDOS 5.1.4 into a MS-DOS system? I keep a large mailing list on Profile III+ HD (over 4000 names) and would somehow like to install that data into a MSDOS database. Any suggestions?

(Fm: LDOS Support jjkd) Yes Ed, as a matter of fact there are numerous methods. The problem breaks down into two stages: (1) Physically transferring the data from one machine to the other, and (2) Altering the data format so that it is useable by software availalble on the new machine.

The first is easily handled in one of two ways. The data may be moved to a disk readable by the new machine, or it may be transferred over an RS232 link. The disk method is limited by the fact that (a) special software will need to be purchased, and (b) you can only transfer a diskfull at a time.

If your database is on a hard disk, you will have to break it into chunks, move each chunk over and then re-assemble the chunks on the hard disk at the other end.

The RS232 method can generally be accomplished using public domain software at both ends, so no special software need be purchased. Also, the entire file can be transferred from hard disk to hard disk, no segmentation required. The disadvantage is that you have to have both machines in one place at one time. Modeming probably isn't practical, and you can't trade in the old machine on the new machine.

Massaging the data to fit into the new program is another matter entirely. You need to decide what database you are going to use on the destination machine first, then we can examine the options in more detail.

Editor kudos

(Fm: Marc Nowell to H. Brothers) You speak of a new "configurable" editor. I'm using BRIEF now and am blown away by it! I set mine to a **WEIRD** mix of commands from the Norton Editor, WordStar, SideKick, and about 30% homebrew! What do you use?

(Fm: H. Brothers) I'm using BRIEF for programming, and something called EDIX for word processing on the MS-DOS machine (LeScript on the Model 4). BRIEF is great!, isn't it?

PC DOS vs MS-DOS

(Fm: Adam Rubin) Probable dumb question here, but that's never stopped me yet: What does it mean when a program works under PC DOS, but won't run properly under MS-DOS? Did someone use something undocumented somewhere? Have official MS-DOS guidelines been ignored? How frequently does something like this show up in commercial programs? (The program in question is "Webster's New World Writer", and was tried under both DOSes on an XT clone.) Many thanks for any enlightening comments!

(Fm: LDOS Support jjkd) Several possibilities:

1) The program relies on undocumented internals of the OS. This will usually tie a program to a particular release of DOS, or at least those that it "knows" about, in addition to eliminating MS-DOS compatibility. I wouldn't expect a program like that to work under, say, the just released DOS 3.3, for example. There is generally no excuse for this. In many cases it is caused by the copy protection scheme, if any. I wouldn't keep such a program, as I wouldn't trust it to keep on working for any length of time. It would be returned immediately.

2) What you have is a barfy, botched-up version of MS-DOS. When MicroSoft licenses a version of MS-DOS to a vendor, the vendor is responsible for final assembly of the OS, along with mastering and duplication. The vendor is also usually responsible for writing or at least assembling a number of the utilities that come with. Some just never can get it right. Who's version of MS-DOS is this?

3) Possible BIOS/Hardware incompatibilities. Because part of DOS directly modifies the low memory regions normally maintained, containing and controlled by the BIOS interrupts, a not fully 100% compatible machine BIOS could become a problem. An example of this is popping up with the use of some 3.5 inch drive options on some clone PCs. They will only format to 360K with many releases of MS-DOS, but will usually work fine with real IBM DOS (except for BASIC, of course). Often a BIOS change will fix the problem.

(**Fm: jeff brenton**) Well, technically, since "PC-DOS" is a trademark of Digital Research Corp., and refers to their concurent CP/M-86 (which now supports most MS-DOS stuff from what I hear), I would be surprised if ANYTHING that requires "PC-DOS" would work under generic MS-DOS! [grin]

but, seriously, MS-DOS is bigger than IBM-DOS due to trade secret problems, plus IBM essentially re-writes MS-DOS anyway. there ARE differences, but anything that follows the rules for MS-DOS programming [you know the ones; written by Microsoft?] should run on either.

Howsomeever, some people decide that "by the rules" isn't good enough, and do implementation-specific kludges that make even switching from version 3.1 to 3.2 of IBM DOS risky....

(**Fm: Adam Rubin**) Thanks for your reply! I'd guess it's "3) BIOS/Hardware incompatibilities". If it'll help clarify anything, here's some specifics on everything. (It's actually my father's system; I'm just visiting.)

This is an Epson Equity II (PC XT clone), with a V30 @ 4.77/7.16 MHz, 640K RAM, and 2 360K floppy drives. MS-DOS 3.10 was included with it. The program in question is "Webster's New World Writer", v1.01, which appears to have been compiled with Lattice C v2.1. It's not copy-protected.

The one problem I've noticed when running Webster's under the Epson MS-DOS 3.11 occurs when the program checks to see if a disk has been inserted, e.g. right after "Insert Spelling Checker in A: and press ENTER." The result is an apparently endless sequence of "Drive not ready -- Abort, Retry, Ignore" messages. When it doesn't check for a disk, e.g. Save Document, there's no problem.

I tried the program under PC DOS 2.00, 2.10, and 3.10, and that problem doesn't occur. In fact, my father called the customer support number, and was told that the program is supported under PC DOS only, not MS-DOS. (That sound as weird as I think it does?)

Does all that explain anything? My knowledge of MS-DOS is admittedly pretty near zero, at least compared to the level of knowledge in this forum.

Well, the manual says, "Webster's runs with PC DOS, versions 2.0 and above," though "PC DOS" isn't mentioned in the trademark acknowledgements. Yep, I've run into the rules for MS-DOS programming; I saw the 1000+ page book Microsoft finally published last summer. Anyway, thanks for your comments! A few of the gory technical details are in my previous message to jjkd, if you're interested (and masochistic?).

HELP with EXE files

(**Fm: Nate Salsbury**) I am in serious need of help. I'm doing things in MS-DOS (using MASM and LINK) that I don't understand. Consequently, when the results are not as expected, I have no idea of how/where to fix things. I'm not trying to shirk my responsibility to "read the manual..." I have read it MANY times plus several "explanatory" books and I am still completely in the dark.

1. In MISOSYS' TMQ, Roy says:" No MSDOS file has absolute loading instructions. A '.COM' file loads relative to the next paragraph. An '.EXE' file has a header record listing a chain of file locations needing segment fixup."

This reads to me like a description of what I know as a "relocatable" program in the Z80 world, i.e. it will work/run/play ANYPLACE in memory.

My exquisitely BADLY written DOS manual says this:"... the output file from MS-LINK (run or .EXE) is not bound to specific memory addresses; therefore it can be loaded and executed at ANY CONVENIENT ADDRESS (my emphasis) by the operating system."

I therefore have the idea that an .EXE file can (and will) be loaded into ANY place in memory by DOS (HOW does it decide WHERE to load?) and that DOS "fixes" the necessary addresses so it will play 'where it is'. Is that the correct view?

2. If I use MASM and LINK on my XT CLone to produce an .EXE file (a simple file with only DOS commands and proper interrupts - NO direct hardware access) shouldn't it be executable on a Tandy 1000 HD "as is"? If not, WHY not?

3. I wrote a very short 8088 ASM program which was eventually to be merged into a BASIC program. Following procedures outlined in "Assembly Language Primer for the IBM XT & PC", I used MASM (no problem) followed by the use of LINK with its "HIGH" switch, e.g. LINK file/HIGH. The accompanying text: "Until now, LINK created .EXE files AT THE LOWEST AVAILABLE MEMORY ADDRESS (ed. ?????). This will now be occupied by BASIC."

"Fortunately,... /HIGH option will cause an .EXE file to be installed in High memory, directly below the transient part of DOS." My VERY poorly written DOS manual confirms this: "Use of the /HIGH switch causes MS-LINK to place the run file as high as possible in memory. Otherwise MS-LINK places the run file as low as possible". Now how does THAT fit in with the idea that an .EXE file is NOT tied to any part of memory? What does that mean: ".... installed in LOW memory". These are all real, English words but they DON'T MEAN ANYTHING to me because they seem SO contradictory. In one place I'm told that .EXE files can load ANYPLACE in memory.... here I'm told that that LINK is selecting a SPECIAL part of memory (with the /HIGH switch.)

4. So, now I have my .EXE file composed with the /HIGH switch of LINK. Using DEBUG, I load it into memory and find that the DOS has assigned it to Code Segment (CS register) 9F94H. I am supposed to use THAT address to merge this file into a BASIC program with THAT address being set by using a BASIC statement to DEF SEG = &H9F94. I do all that is outlined and it works just fine.

HOWEVER, when I move the program and .BIN file that I created on my 640K XT over to a 256K Tandy 1000 HD, the thing bombs. After some experimentation, I discover that when I load my .EXE file (created on my XT) using DEBUG on the Tandy 1000, it NOW loads at CS = 3BE0H (!?????) so I have to REWRITE my BASIC program (and recreate the .BIN file needed) using THIS address. This means that my BASIC program is NOT PORTABLE TO OTHER GW BASIC MACHINES because the EXE file used with it loads at 9F94H on MY machine, 3BE0H on ONE Tandy machine and WHO knows where on any other MS-DOS computer. Is this progress?

Is there ANY way to use MASM and LINK to create an .EXE file that will ALWAYS load into some predetermined address? Even .COM files with an "ORG 100H" will still be actually loaded into various ACTUAL addresses when called in by DOS. Or, asking the question differently, is there a way to 'fool' DOS into thinking that MY machine has only 256K of RAM so that what I do HERE would work on my customer's 256K machine also?

I'm probably asking for too much information via CIS. I would be happy to just get back some recommended reading that would explain the REALITY of what is going on, the structure of the files being created, why they act the way they do (this stuff MUST make sense to Lotus!!) and how to take real CONTROL of the processes that create and use these files. The manual that came with my XT (prepared by Compaq) is absolutely devoid of ANY information along these lines. e.g. There is a program called EXE2BIN which "Changes an .EXE file to a .COM file." Well, that is TRUE but, to me, no help. WHY does one do that, HOW do you tell if you CAN do it to any specific EXE file, WHAT changes are made to the file itself? These are just a couple of completely unmentioned, unexplained items. I would like to be able to read something that described the entire system and the rationale behind it.

Thanks for reading this far. If you can shed some light in these murky quarters, I cannot think of adequate words to thank you further.

(Fm: Jim Beard) Welcome to the wonderful world of MS-DOS. The 8086 uses a "segmented architecture" which means that it is basically a 64K chip with schizophrenia. The .COM files are CP/M-86 compatible files which load at 100H in the lowest available segment. '.EXE' files are relocatable, and I haven't been able to get full documentation on them myself yet. Roy tells me that there is an old article in PC-WORLD on '.REL' and '.EXE' files, but I haven't looked it up. Apparently, headers on .EXE files can give a preference for location, but I am very hazy on that. Good luck. [Editor's note: the article referred to was in PC TECH JOURNAL, October 1985, Vol 3, no. 10, and it referred to the structure of '.OBJ' files. The Microsoft standard is a subset of the Intel standard which is documented in, "8086 Relocatable Object Module Formats, An Intel Technical Specification"].

(Fm: H. Brothers) Nate, Answers to your first two questions: (1) Yes. DOS decides where to load the next program by using the first free paragraph given up by the parent process (normally Command.Com). That is, a program will load at the lowest free address in memory, in the normal run of events. (2) Yes. As long as you aren't trying to use DOS 3.x calls on a DOS 2.x system, your program should be directly portable.

(Fm: LDOS Support jjkd) 1) Part one - True. Part two - Not True.

Yes, a .COM program will end up running anywhere in memory. This is not relocatable code, however. The segmentation structure of the Intel processors uses the CS (code segment) to make the program "think" it knows where it is executing from beforehand. Only the segment registers ever need to be changed.

An .EXE file is more like dynamically relocating a driver or filter under LDOS. A table of things that need to be fixed or relocated is maintained, and then once you know where the code is going to go, you can re-align the stuff in memory.

The quote you mention is not strictly true, the segments contained in the program must be aligned to some degree. For executable code, that is typically a paragraph (every sixteen bytes) boundary. That's because the segment registers can't be adjusted any closer that that.

2) For a standalone machine language program that uses only interrupts, no interface to BASIC, true.

3) The /HIGH option places a flag into the header code so that the system .EXE file loader will shove it up to the top when loading. Your DOS tech ref manual (if IBM's) documents the structure of the header for an .EXE file, not that you really need to know it for this.

4) Sure. The more memory you have, the higher the top of high memory is, right? Remember the 32K Model 3 vs. the 48K Model 3?

Forcing your code to load at a specified address is even more foolhardy than doing so in the LDOS/TRSDOS 6 Z80 world.

The correct way to do this is documented in Appendix B of the IBM BASIC Manual for version 3. Do you have access to this, or do you want more details?

You really need "real" documentation. Get the BASIC manual mentioned in my previous reply. Then get every book written by Peter Norton. There are better references (Peter has a few minor errors, and tends to use decimal when he should use hex) but few better tutorials.

You can graduate from there to "Advanced MS-DOS" by Ray Duncan and "MS-DOS Developer's guide" by Angelmeyer (sp?) and somebody else. Both are great references and sources of code ideas (but little or nothing on BASIC).

I find it fascinating that Intel and the IBM 8086/8088 line are based entirely upon an architecture that most sophisticated programmers and hardware hackers would have considered a "clever kludge" at best. Of course, the 80286 and up do have linear address models, unfortunately it is far too late for that to do much good.

(Fm: Jim Beard) What do you mean, clever! It's just a kludge. The extended addresses of the 80286/80386 are in new instructions; portability with the 8088/8086/80186 machines demands that these instructions be avoided. Microsoft FORTRAN 77 V. 4.0 and probably other languages have compiler options for use of 186/286/386 instructions, but I doubt that these options have much effect except to prevent portability. I'll find out soon enough & say, as I run RATFOR/FORTRAN 77 V 4.0 on an AT at work. With the same 8087 programs, it is a dead heat with my Tandy 1000A with 8087,

though. I would have to compare programs on the same machine with and without the 286 instructions.

(Fm: **Nate Salsbury**) One of my problems seems to be with the word "relocatable". Both .COM files and .EXE files can be loaded into ANY segment. To me, that is the meaning of 'relocatable'. The fact that .COM files always start at 100H of the segment they find themselves in... that is a minor 'distinction". I would still call both type "relocatable".

Joe - if I write a little .COM ML program to print the famous "hello" message, I have to put that 'hello' into a DB place (with a label) and then, someplace in my code I have to "POINT" at that label, e.g. when the program is actually running, it has to be told AN ABSOLUTE ADDRESS in memory where it can find 'hello' in order to reveal it on the monitor.

Since .COM programs can load at ANY segment, this means that my 'hello' text could be (almost) ANYPLACE in memory (the 'almost' is allowing for the constraint on 16-byte boundaries for segments). Since the program is finding an absolute reference SOMEPLACE INSIDE ITSELF, my education from Z80 says that this is what is called a RELOCATABLE program. The fact that is must start at 100H in its present segment is trivial because that 'absolute' start address can be changed in 16 byte increments by merely changing the CS setting!

So, to me, BOTH .EXE and .COM files are 'relocatable'.

(Fm: **jeff brenton to LDOS Support**) You should see P.J.Plauger's editorial comments on the 80x86 family in the April 1987 Computer Language. He LIKES segmentation, but feels that Intel's IMPLEMENTATION of segmentation on the '86 family is botched at best, and incredibly stupid at worst...

[well, to be honest, he didn't say he liked segmentation, but he did point out the advantages of a WELL DESIGNED segmented architecture.]

(Fm: **Jim Kyle**) First and foremost, keep in mind that MS redefines the language to mean what THEY want it to mean, just like Humpty Dumpty did! With that settled, we can discuss their meaning for "relocatable". In the '86 world of segmentation, an address consists of two parts: the segment, and the offset. In any program, the base value for segment 0 can actually be at any of the possible segments in the memory space (that is, every address in which the low four bits are 0), and in this sense you are right that both kinds of program files are relocatable.

However, in the COM file all of the OFFSET values within the program are fixed and cannot be changed, and what MS means by "relocatable" is "changeable offset values". In the EXE structure, the OFFSET values can be (and are) manipulated by the loader.

In more familiar terms, it's the difference between a /CMD file that always loads at 3000h or 5200h, and a /DVR file that must relocate all internal references. The BIG difference between the Z80 and the '86 worlds comes from the segmentation, but so far as programs are concerned this is largely invisible (being handled by the loader) unless you have to be concerned with multiple program segments...

(Fm: **Nate Salsbury to H. Brothers**) I have successfully merged my EXE file into my BASIC program. The major problem, pointed out by you and others, was that /HIGH on a 640K machine is different from /HIGH at 256K! (Seems to me that I **should** have known THAT <grin>. Perhaps it is the BIG NUMBERS that I am not used to yet!)

I once reported to you that I was having a problem with "ML in a string". Turned out to be a clerical error on my part - worked just fine when I did it the right way.

Thanks for participating in "The (electronic) education of Nate (Hyman Kaplan) Salsbury". BTW - with all your Sysopping/writing/product reviewing are you ALSO/STILL teaching?

(Fm: **Nate Salsbury to LDOS Support**) Joe - Sorry to break the thread but I fell quite sure that your earlier reply would have scrolled off by now. First is a semantic problem re: "relocatable COM files".

Acknowledging that COM files must start at an offset of 100H in "this" code segment, it IS true that CS can be located "anywhere" in memory (subject to the 16-byte 'steps'). Doesn't this mean, in practice, that I can have my COM program loaded into and run almost ANY place in memory (at 16-byte 'start' intervals) by suitably diddling CS? If so, to

ME that is "relocatable code"... it will run (almost) anywhere in REAL memory without CHANGING THE CODE.

As I read statements from you and others, I **think** you are saying that EXE files are "more" relocatable because THEY can play at ANY address in a segment e.g. they are not tied to that 100H offset. BUT, are they not "fixed on the fly" by DOS as they are being loaded to ADJUST addresses for where the EXE file will be in ACTUAL memory - more or less the same scheme used to relocate a filter just under HIGH$ in you-know-what. Right? If so, then they are not "relocatable" by MY definition ("Put it anywhere in memory and it will run").

Question: SUPPOSE I manage to load an EXE file high in the current CS space and the EXE file extends beyond the end of this CS segment. NOW what? Does it work? Does my computer BEEP? Does my mono screen display colors? Or, will DOS refuse to load it this way?

Joe - I **BLUSH** at not realizing that a program fixed to load near 640K would differ from one fixed for 256k! I **knew** that in my Z80 32K Model III vs the 64K Model 4! Must have been blinded by the BIG NUMBERS (I can STILL recall my original Model I complete with 4K and a cassette!)

I have a lot of good reference books - both P. Norton, AT&T's BASIC manual, getting DOS manuals from Bob Haynes and have "Adv. MS DOS" on order. STAND BACK... MORE questions are SURE to follow! Thanks for the current info!

(Fm: LDOS Support jjkd) It's all a matter of point of view, I guess. Both .COM and .EXE files won't work without some form of management by the DOS. For .COM, it is as simple as setting all the segment registers to 0010:0000 less than the code start address. For .EXE files, the loader has its work cut out for it. All internal segment references must be adjusted to account for the offset, and the segment registers set to (probably) various different values.

What makes both of these not "relocatable code" in my book, is that you can't move, let's say, half the code in a segment relative to the first half at run-time, and have the end result work. Of course, to have this work even on a machine with "truly relocatable" code, like the 68000, each piece must be self contained and not reference the other. In an .EXE file, they could then also be put in different code segments.

As for loading, no, it doesn't work that way. If you load an .EXE file, it doesn't load into "this" code segment, it always loads into its "own" code segment. Those could, of course, overlap.

(Fm: Les Mikesell) Remember that in .COM files, the segment sizes cannot exceed 64K - that is the limit of the relocation that can be done by setting the segment pointers at run time.

(Fm: LDOS Support to Jim Beard) I call it a clever kludge because it works at all in the general case for non-trivial programs. RE the 80286/386, I was referring to protected mode operation in regards to the linear addressing mode. Linear addressing is not available in real mode with or without the new instructions. Of course, the 386 memory management also allows you to operate in virtual '86 mode to simulate the kludge.

(Fm: LDOS Support to Nate Salsbury) Entirely a matter of semantics. To me, a program is relocatable if the code is entirely based on relative jumps, or if locations in memory are calculated at run-time (not compile or assemble time, of course offsets must be calculated then). "Real" 6809 and 680x0 relocatable code works this way, due to the linear address space and the fact that almost every instruction has a relative form. In both .COM and .EXE files, there are absolute offsets all over the place.

The .COM file works only because the segment registers are set up to "magic" values that represent the location the program segment (only one allowed) was loaded at.

In .EXE files, since more than one segment is allowed, any offsets requiring reference to an absolute location are "hot patched" on the fly when the program is loaded.

Note that according to this definition, the Z80 code used in LDOS filters and drivers is not "relocatable code", but instead is "code that is relocated". A fine difference indeed...

(Fm: Nate Salsbury) I think the distinction (re: LDOS filters) between "code that is relocated" vs "relocatable code" is NOT a 'fine difference'. In fact, that is exactly

the point I'm trying to make (and understand)!!

Is it true that a COM file will be placed into absolute memory by the loader as a function of what has gone on before? In other words, at different times, a COM file might find itself with different Code Segment (CS) i.e. be located in some DIFFERENT PLACE in ABSOLUTE memory. But, (here is the important thing to me), NOTHING IN THE PROGRAM has been changed, corrected, modified, etc. - every byte of working code is identical to the "previous time" it was used - EVEN IF AT A DIFFERENT ABSOLUTE ADDRESS.

If that concept is "-1", then I would say that a COM program is RELOCATABLE. On 'tother hand, an EXE file would be "code that is relocated".

Do I have the right concept? I hope so; I've got to get on with "bigger" things.

(Fm: LDOS Support jjkd) Yes and no. <<Grin>> Under that definition of "relocatable", then yes, .COM are and the .EXE are not. Only the CS register need change. Assuming that's settled, let's look at this from a slightly different point of view.

The Z80 provides two basic kinds of jump instructions, absolute and relative. Absolute jumps go to a particular spot in the 64K addressable by the Z80. The relative jumps, on the other hand, go to a particular spot relative to the current instruction pointer. Thus, no matter what the "absolute" IP may be, code written only using JR may be moved around without concern, as long as we can get to the entry point and back from the exit point. Presto Changeo, we now have relocatable code.

On the 8088, (non-long) jumps are always absolute within the current code segment, but are treated relative to the value of CS. Thus, if we move an entire code segment, no problem, everything is hunky dorey. Since the CS register has been changed, all the offsets within the segment are still OK. One can consider this the same as the way Z80 absolute jumps work, the only difference being that the 64K doesn't have to start at zero in the physical memory map, that's what the CS segment register is for.

Now, let's move just part of the code in that code segment, just the second half (at run-time, not at assemble-time). Since the assembler calculated all the displacements from the start of the code segment, and the distance from the start has changed, all the code in that second half will stop working.

On the 68000 (and 6809), relative jumps can be coded relative to the current IP, not some other landmark. So, as long as we keep track of the entry points and where we came from, properly generated code can be moved willy-nilly without fixing anything on the part of the DOS, the loader or anything else. In my book, only that is truly relocatable code.

(Fm: H. Brothers) Joe, Now I'm confused about your definition of "relocatable" vis a vis the Intel architecture.

The byte format for a conditional jump and a SHORT jump is "[opcode] [displacement]" and the operation is "ip <- ip + displacement".In both cases, the displacement is a signed 8-bit quantity, just like a Z-80 JR, and is sign-extended. Isn't that relocatable by your definition? For a NEAR jump, the byte format is "[E9] [dispL] [dispH]" and again the operation is "ip <- ip + disp". As far as I can see, there's nothing in any of the jump instructions that makes them tied to the CS. Only indirect jumps and jumps out of the current segment are non-relocatable.

The same is true for calls. The byte format for a NEAR call is "[E8] [dispL] [dispH]", and again the effect is "ip <- ip + disp". Indirect calls and calls out of the segment are non-relocatable, but calls within a segment (and that stay within the segment after relocation) should be able to move without problem, I think.

The instruction which the 80x86 family is missing (and I think the 68000 has it, but it's been a while since I looked closely at the 68K instruction set) is a relative indirect jump or call, which would have the effect of "ip <- ip + REG16" or "ip <- ip + MEM16". But if a programmer stays away from things like jump tables and other indirect addressing on an Intel chip, a .COM program or any portion of it should be relocatable within the current segment, no?

(Fm: Jim Kyle to H. Brothers) Hardin, You realize, of course, that there's a column crying to be written here. To me, what Joe is calling "relocatable" is "position independent code" which will operate the same way no matter where in memory it's located, without any changes at all being made to it. My definition for "relocatable" is based on the

derivation of the word: "capABLE of being RELOCATed", and in the general sense applies to any and all code since you can always add some delta to a hard-coded immediate address (the first thing I ever saw Gary Kildall's byline on was a method for doing just that to 8080 code, in an early DDJ). In practice, though, I follow the G-E mainframe convention which gave me my first exposure to the idea, and use it to refer to code which is modified at LOAD time to account for its memory location.

Interesting things come from this: when considered as a single entity, a COM program is position independent, because the loader only sets the segment registers and does no modification at all to the actual image. An EXE file, on the other hand, is relocatable IF (and, I think, only if) it makes any references to the SEG operator of MASM (i.e., requires segment fix-ups). Yet the nature of the relocation done by the loader is not very different from the non-relocation it does for COM files.

And, of course, as you point out, all 8086/88 conditional jumps are position independent, like the Z80 JR or the 68xxx series.

(Fm: LDOS Support to H. Brothers) You are entirely right. Now, I only have to figure out what I was thinking of.

(Fm: H. Brothers to Jim Kyle) Jim, I agree that this is fodder for a future column -- was thinking that to myself when Nate first brought the subject up. There's a bunch of experimenting I have to do first, to see exactly what happens with an EXE file (don't think I've written many .COM files at all, lately, BTW) when LINK is done with it and as the loader gets ahold of it. Of course, the outcome from a user's point of view is that almost all programs are relocatable, since the user doesn't have to worry about addresses nor whatever happens to be resident in memory (conflicts in fights over hotkeys, etc., is an entirely different question).

Hmm -- I wonder if an analog of @WHERE would have some use? Especially in programs that use the absolute indexed jumps and calls?

Applications for the User

UNDATE - by Luis M. Garcia-Barro

I wrote UNDATE in answer to the query from Kenyon F. Karl, and to your guidelines (THE MISOSYS QUARTERLY), page 48, Winter 1987), in reference to the incompatibility created by the new time/date stamp of LDOS 5.3. Since I am not a CIS subscriber, I have not a way of sending it to him.

UNDATE/CMD
For LDOS only
(c)1987 Luis M. Garcia-Barrio
Free distribution. Not for sale

Syntax: UNDATE :drive

The new time/date stamp used by LDOS 5.3 and LS-DOS 6.3 has created a minor incompatibility with previous releases of those DOSs. In the new format, bytes 18, 19 of each directory entry are used by the date/time stamp. Under the old format those bytes were used by the ACCESS password. (This incompatibility was mentioned in THE MISOSYS QUARTERLY, Winter 1987).

As a result, a disk formatted (or DATECONVerted) with LDOS 5.3 OR LS-DOS 6.3, will appear to have all its files with ACCESS password protection when used under LDOS 5.1 or TRSDOS 6.2. There are two ways to solve the problem: (1) Format all disks with LDOS 5.1 or TRSDOS 6.2. The new versions of those DOSs will not attempt to stamp the files with the date/time. (2) UNDATE the disks, by resetting bit 3 of GAT+CDH and giving files an ACCESS password of 8 spaces. LDOS 5.3/LS-DOS 6.3 will access them as if formatted with LDOS 5.1/TRSDOS 6.2, and these two will access them as their own. In short, UNDATE reverses the process of DATECONV.

The problem I can foresee is a file trying to access another, expecting it to have an ACCESS password. In those rare cases, use ATTRIB to change the password. For bug reports, suggestions etc., leave a note to the sysop at 8/N/1 #4, (215) 848-5728.

```
                 00001 ;-------------------------------------------------------
                 00002 ; UNDATE/CMD by Luis M. Garcia-Barrio, 8/N/1 #4 (215) 848-5728
                 00003 ;            For free distribution. Not for sale.
                 00004 ;               Based in LDOS 5.3 DATECONV/CMD
                 00005 ;   Allows conversion of LDOS 5.3/LS-DOS 6.3 disks with the
                 00006 ;     new date/time stamp to the LDOS 5.1 standard format.
                 00007 ;   The ACCESS password installed is 8 spaces (no password).
                 00008 ;-------------------------------------------------------
4030             00009 @ABORT  EQU     4030H
4209             00010 @CKDRV  EQU     4209H
4451             00011 @DIV    EQU     4451H
4467             00012 @DSPLY  EQU     4467H
4409             00013 @ERROR  EQU     4409H
002B             00014 @KBD    EQU     002BH
428A             00015 @LOGOT  EQU     428AH
0060             00016 @PAUSE  EQU     0060H
478F             00017 GETDCT  EQU     478FH
4777             00018 RDSECT  EQU     4777H
4772             00019 VERSEC  EQU     4772H
4768             00020 WRPROT  EQU     4768H
429F             00021 KFLAG$  EQU     429FH
442B             00022 SFLAG$  EQU     442BH
5200             00023         ORG     5200H
5200  ED73C052   00024 START   LD      (SPSAVE),SP      ;save for the end
5204  E5         00025         PUSH    HL               ;save parameters
5205  CDE752     00026         CALL    KEY
5208  214253     00027         LD      HL,TITLE         ;say who
520B  CD6744     00028         CALL    @DSPLY
520E  E1         00029         POP     HL               ;parameters in HL
520F  7E         00030 PRAMS   LD      A,(HL)
5210  FE20       00031         CP      20H              ;bypass the spaces
5212  23         00032         INC     HL
5213  28FA       00033         JR      Z,PRAMS
5215  FE3A       00034         CP      3AH              ;look for ":"
```

```
5217  C23E53  00035          JP      NZ,BADDRV
521A  7E      00036          LD      A,(HL)          ;check for drive 0-7
521B  D630    00037          SUB     30H
521D  FE08    00038          CP      8H
521F  D23E53  00039          JP      NC,BADDRV
5222  320D53  00040          LD      (DRIVEW),A      ;store drive for writes
5225  321F53  00041          LD      (DRIVER),A      ;and for reads
5228  4F      00042          LD      C,A             ;check the drive in C
5229  CD0942  00043          CALL    @CKDRV
522C  C23E53  00044          JP      NZ,BADDRV       ;if drive is not available
522F  DA2E53  00045          JP      C,WPDRIV        ;if disk is write protected
5232  CD8F47  00046          CALL    GETDCT          ;else get DCT address in IY
5235  FD7E09  00047          LD      A,(IY+9H)       ;DCT+9 is the DIR cylinder
5238  325252  00048          LD      (DIRCYL),A      ;store the DIR cylinder number
523B  57      00049          LD      D,A             ;and prepare to read it
523C  1E00    00050          LD      E,0H
523E  CD1B53  00051          CALL    READS           ;read GAT
5241  3ACD55  00052          LD      A,(GATCD)       ;type-of-disk byte
5244  CB5F    00053          BIT     3,A             ;bit 3 set for new date/time?
5246  2009    00054          JR      NZ,NEWTYPE
5248  214554  00055          LD      HL,MSGOLD       ;tell that is it old type
524B  CD6744  00056          CALL    @DSPLY
524E  C33853  00057          JP      EXITOK          ;and do not bother any more
5251  1600    00058 NEWTYPE  LD      D,0H
5252          00059 DIRCYL   EQU     $-1H
5253  1E03    00060          LD      E,3H            ;read sector 3 of DIR cylinder
5255  CD1B53  00061          CALL    READS
5258  3A1455  00062          LD      A,(DIRERN)      ;ERN of DIR/SYS
525B  D602    00063          SUB     2H              ;find the number of sectors
525D  FE21    00064          CP      21H             ;   to be read
525F  3802    00065          JR      C,FILES
5261  3E20    00066          LD      A,20H
5263  3C      00067 FILES    INC     A
5264  329852  00068          LD      (DIRSIZE),A
5267  1E02    00069          LD      E,2H            ;read sector 2 of DIR cyl
5269  CD1B53  00070 GET      CALL    READS           ;read a sector
526C  E5      00071 AGAIN    PUSH    HL
526D  CB66    00072          BIT     4,(HL)          ;is the record in use?
526F  CA8652  00073          JP      Z,NEXT
5272  DDE5    00074          PUSH    IX
5274  E1      00075          POP     HL
5275  3E96    00076          LD      A,96H           ;first byte of PASSWORD
5277  DD7712  00077          LD      (IX+12H),A      ;in place
527A  3E42    00078          LD      A,42H           ;second byte of PASSWORD
527C  DD7713  00079          LD      (IX+13H),A      ;in place
527F  3A7054  00080          LD      A,(CHANGD)      ;number of files updated
5282  3C      00081          INC     A
5283  327054  00082          LD      (CHANGD),A      ;in place
5286  CDE752  00083 NEXT     CALL    KEY
5289  C23253  00084          JP      NZ,ABORT
528C  E1      00085          POP     HL              ;restore pointer
528D  7D      00086          LD      A,L             ;advance to next entry
528E  C620    00087          ADD     A,20H
5290  6F      00088          LD      L,A
5291  C26C52  00089          JP      NZ,AGAIN        ;last entry in sector?
5294  CD0953  00090          CALL    WRTSYS          ;if so, write the new sector
5297  3E00    00091          LD      A,0H
5298          00092 DIRSIZE  EQU     $-1H
5299  1C      00093          INC     E
529A  BB      00094          CP      E
529B  D26952  00095          JP      NC,GET          ;done with all DIR sectors?
```

```
529E  1E00     00096          LD      E,0H            ;if so, read GAT sector
52A0  CD1B53   00097          CALL    READS
52A3  3ACD55   00098          LD      A,(GATCD)
52A6  E6F7     00099          AND     0F7H            ;reset bit 3 of GAT+CDH
52A8  32CD55   00100          LD      (GATCD),A       ;in place
52AB  CD0953   00101          CALL    WRTSYS          ;write new GAT
52AE  2A7054   00102          LD      HL,(CHANGD)     ;number of changes to
52B1  11EA53   00103          LD      DE,MSG1         ;prepare exit message
52B4  D5       00104          PUSH    DE
52B5  CDCE52   00105          CALL    MSG1NO          ;put number in message
52B8  E1       00106          POP     HL
52B9  CD8A42   00107          CALL    @LOGOT
52BC  210000   00108          LD      HL,0000H
52BF  310000   00109 EXIT     LD      SP,0000H        ;restore SP and prepare to end
52C0           00110 SPSAVE   EQU     $-2H
52C2  7C       00111          LD      A,H
52C3  B5       00112          OR      L
52C4  C8       00113          RET     Z
52C5  3A2B44   00114          LD      A,(SFLAG$)
52C8  CB6F     00115          BIT     5,A
52CA  C8       00116          RET     Z
52CB  C33040   00117          JP      @ABORT
52CE  0605     00118 MSG1NO   LD      B,5H            ;put # of files in message
52D0  3E20     00119          LD      A,20H           ;first 5 spaces in place
52D2  12       00120 MSG1N1   LD      (DE),A
52D3  13       00121          INC     DE
52D4  10FC     00122          DJNZ    MSG1N1
52D6  D5       00123          PUSH    DE              ;do it backwards
52D7  1B       00124          DEC     DE
52D8  3E0A     00125 MSG1N2   LD      A,0AH           ;HEX to DEC
52DA  CD5144   00126          CALL    @DIV
52DD  C630     00127          ADD     A,30H           ;to ASCII
52DF  12       00128          LD      (DE),A          ;in place
52E0  1B       00129          DEC     DE
52E1  7C       00130          LD      A,H
52E2  B5       00131          OR      L
52E3  20F3     00132          JR      NZ,MSG1N2       ;till done with the five
52E5  D1       00133          POP     DE
52E6  C9       00134          RET
52E7  E5       00135 KEY      PUSH    HL              ;kb check for <BREAK>
52E8  219F42   00136          LD      HL,KFLAG$
52EB  CB46     00137          BIT     0,(HL)
52ED  2818     00138          JR      Z,KEY3
52EF  F5       00139          PUSH    AF
52F0  C5       00140          PUSH    BC
52F1  D5       00141          PUSH    DE
52F2  CB86     00142 KEY1     RES     0,(HL)
52F4  01000B   00143          LD      BC,0B00H
52F7  CD6000   00144          CALL    @PAUSE
52FA  CB46     00145          BIT     0,(HL)
52FC  20F4     00146          JR      NZ,KEY1
52FE  CD2B00   00147 KEY2     CALL    @KBD
5301  B7       00148          OR      A
5302  20FA     00149          JR      NZ,KEY2
5304  D1       00150          POP     DE
5305  C1       00151          POP     BC
5306  F1       00152          POP     AF
5307  E1       00153 KEY3     POP     HL
5308  C9       00154          RET
5309  210055   00155 WRTSYS   LD      HL,BUFSEC       ;write system sector
530C  0E00     00156          LD      C,0H
```

```
530D            00157 DRIVEW  EQU    $-1H            ;drive number for writes
530E  CD6847    00158         CALL   WRPROT
5311  2014      00159         JR     NZ,ERROR
5313  CD7247    00160         CALL   VERSEC
5316  FE06      00161         CP     6H
5318  200D      00162         JR     NZ,ERROR
531A  C9        00163         RET
531B  210055    00164 READS   LD     HL,BUFSEC       ;read disk sector
531E  0E00      00165         LD     C,0H
531F            00166 DRIVER  EQU    $-1H            ;drive number for reads
5320  CD7747    00167         CALL   RDSECT
5323  C8        00168         RET    Z
5324  FE06      00169         CP     6H
5326  C8        00170         RET    Z
5327  F6C0      00171 ERROR   OR     0C0H
5329  CD0944    00172         CALL   @ERROR
532C  180A      00173         JR     EXITOK
532E  212E54    00174 WPDRIV  LD     HL,MSGWP
5331  DD        00175         DEFB   0DDH            ;changes next to LD IX,xxxx
5332  211554    00176 ABORT   LD     HL,MSGABR
5335  CD8A42    00177         CALL   @LOGOT
5338  21FFFF    00178 EXITOK  LD     HL,0FFFFH
533B  C3BF52    00179         JP     EXIT
533E  3E20      00180 BADDRV  LD     A,20H
5340  18E5      00181         JR     ERROR
5342  0A        00182 TITLE   DEFB   0AH,0AH
      0A
5344  55        00183         DEFM   'UNDATE/CMD v1.0 - '
5356  4C        00184         DEFM   'LDOS 5.3 to LDOS 5.1 Date Conversion.'
537B  0A        00185         DEFB   0AH
537C  28        00186         DEFM   '(c)1987 Luis M. Garcia-Barrio,'
539A  20        00187         DEFM   ' 8/N/1 #4 (215) 848-5728.'
53B3  0A        00188         DEFB   0AH,0AH
53B5  54        00189         DEFM   'The ACCESS passwords for all files'
53D7  20        00190         DEFM   ' will be removed.'
53E8  0A        00191         DEFB   0AH,0DH
53EA  20        00192 MSG1    DEFM   '      file(s) updated, conversion complete'
5414  0D        00193         DEFB   0DH
5415  44        00194 MSGABR  DEFM   'Date conversion aborted!'
542D  0D        00195         DEFB   0DH
542E  57        00196 MSGWP   DEFM   'Write protected drive!'
5444  0D        00197         DEFB   0DH
5445  54        00198 MSGOLD  DEFM   'This disk does not use the time/date stamp'
546F  0D        00199         DEFB   0DH
5470  0000      00200 CHANGD  DEFW   00H
5500            00201 BUFSEC  EQU    5500H
55CD            00202 GATCD   EQU    BUFSEC+0CDH
5514            00203 DIRERN  EQU    BUFSEC+14H
5200            00204         END    START
```

STRIP/BAS - by James Beard, Ph.D.

I had occasion to search for a way to strip off the last byte of a file - for purposes that I don't need to go into here. My first attack was to try the appending a NULL file to the target file with the OS's APPEND command using the STRIP parameter; but APPEND aborts when the source file is NULL.

My next attack was to write the STRIP/BAS program in BASIC which follows. This can be used to remove the last character of a file by copying all but the last character to another file.

```
10 DEFINT I-N: INPUT "Input filespec? ",INFILE$: OPEN"R",1,INFILE$
20 INPUT "Output filespec? ",OUTFILE$: OPEN"O",2,OUTFILE$
30 LC=VARPTR(#1): GET 1: IPTR=0: OD$=""
40   GOSUB 80: ITYPE=ASC(CH$): GOSUB 80: NLEN=ASC(CH$): PRINT ITYPE,NLEN
50   IF NLEN<3 THEN IF ITYPE<>2 THEN NLEN=NLEN+256 'Allow lengths to 258 bytes
60   FOR I=1 TO NLEN: GOSUB 80: NEXT I 'Feed through block
70   IF ITYPE=2 THEN PRINT# 2,OD$;: CLOSE: END ELSE GOTO 40
80 IF IPTR=255 THEN 100
90 CH$=CHR$(PEEK(LC+IPTR)): IPTR=IPTR+1: OD$=OD$+CH$: RETURN
100 CH$=CHR$(PEEK(LC+IPTR)): PRINT# 2,OD$;CH$;: OD$="": GET 1: IPTR=0: RETURN
```

MORSE Code from a Model 4

(Fm: MISOSYS) Here's one I found on one of the hard drives which we acquired when we acquired the retail operation of Logical Systems last year. Judging from some other files, I was able to discern that this Morse code program was derived from one written for the IBM PC. Apparently, LSI reworked it for use on a Model 4. I note that LINE 100 pokes 18H into low memory. This makes a slight alteration to the @SOUND routine. The address is consistent across 6.2 and 6.3. On the other hand, the original value of 28H is not restored when you terminate the program (the menu doesn't really give you an option to STOP). Anyway, for what it's worth to you aspiring Morse code folks, here's CODE/BAS.

```
100 POKE &H3AF,&H18:CLS
110 CLEAR 5000:DEFINT A-Z:DIM TBL$(48)
120 FOR L= 1 TO 42:READ TBL$(L):NEXT
130 RATE=2:TONE=5:V=-1:RANDOM
140 DELAY=100*RATE:IF DELAY=0 THEN DELAY=50
150 DOT=RATE
160 DASH=RATE*4:IF DASH=0 THEN DASH=2
170 GOTO 530
180 A$=INKEY$:IF A$="" THEN 180
190 IF A$="@" THEN 530
200 GOSUB 210:GOTO 180
210 IF A$>"@" THEN A$=CHR$(ASC(A$)AND &H5F):PRINT A$;"  "; ELSE 230
220 CODE$=TBL$(ASC(A$)-64):GOTO 290
230 PRINT A$;"  ";
240 IF A$<"," OR A$>";" THEN 260
250 CODE$=TBL$(ASC(A$)-43+26):GOTO 290
260 IF A$="(" OR A$=")" THEN CODE$="-.--.-":GOTO 290
270 IF A$="?" THEN CODE$="..--..":GOTO 290
280 PRINT:FOR L=1 TO DELAY*20:NEXT:RETURN
290 A$=CODE$:PRINT A$
300 FOR L=1 TO LEN(A$)
310     IF MID$(A$,L,1)="." THEN SOUND TONE,DOT ELSE SOUND TONE,DASH
320     FOR LL= 1 TO DELAY*2:NEXT
330 NEXT: FOR L= 1 TO DELAY*9:NEXT
340 RETURN
350 PRINT:PRINT"Enter Filespec: ";:LINE INPUT F$
360 OPEN "R",1,F$,1
370 FIELD #1, 1 AS C$
380 FOR F= 1 TO LOF(1):GET 1,F:A$=C$:GOSUB 210
390 A$=INKEY$:IF A$="@" THEN CLOSE:GOTO 530
400 NEXT F:CLOSE:GOTO 530
410 DATA .-,-...,-.-.,-..,.,..-.,--.,....,..,.---,-.-,.-..,--,-.,---,.--.,--.-
420 DATA .-.,...,-,..-,...-,.--,-..-,-.--,--..
430 DATA --..--,-....-,.-.-.-,-..-.,-----,.----,..---,...--,....-,.....
440 DATA -....,--...,---..,----.,---...,-.-.-.
450 A$=INKEY$:IF A$="@" THEN 530
460 A=RND(79)+43:A$=CHR$(A):GOSUB 210:GOTO 450
470 PRINT:PRINT "5 = 4 WPM, 4 = 5 WPM, 3 = 7 WPM, 2 = 10 WPM, 1 = 16 WPM, 0 = 26 WPM"
```

```
480 PRINT:PRINT"Set Rate (0-7): ";:INPUT A
490 IF A<0 OR A>7 THEN 480
500 RATE=A:GOTO 140
510 PRINT:PRINT"Set TONE (0-7): ";:INPUT TONE
520 IF TONE<0 OR TONE>7 THEN 510
530 CLS:PRINT:PRINT TAB(13)"M O R S E   C O D E   L E A R N I N G   S Y S T E M"
540 PRINT TAB(13)"Version 1.0.9 - Feb. 1985 - by Logical Systems Inc."
550 PRINT TAB(21)"This program is Public Domain"
560 PRINT:PRINT:PRINT:PRINT TAB(28)"M A I N   M E N U"
570 PRINT:PRINT:PRINT,"(1) .... Set TONE"
580 PRINT,"(2) .... Set RATE"
590 PRINT,"(3) .... Send Ascii FILE"
600 PRINT,"(4) .... Send from KEYBOARD"
610 PRINT,"(5) .... Send RANDOM sequences"
620 PRINT,"(6) .... Send QUIZ"
630 PRINT:PRINT,"The <@> key will return you to this menu"
640 PRINT:PRINT,"Your selection please: ";:LINE INPUT MM$
650 MM=VAL(MM$):IF MM<1 OR MM>6 THEN 530
660 CLS:ON MM GOTO 510,470,350,180,450,670
670 CLS:PRINT"You will be sent a character, you must respond by typing that character."
680 PRINT"<SPACE> will RESEND the character (counts as 1 wrong)
690 PRINT"<*> will PASS the character (counts as 1 wrong)"
700 PRINT"<F1> will TOGGLE VIDEO display of dit/dah"
710 PRINT"<@> will TERMINATE quiz and display score"
720 PRINT
730 PA=0:CA=0:RT=0:Q=0:WA=0:GOTO 850
740 B$=INKEY$:IF B$="" THEN 740
750 IF B$=CHR$(129) THEN V=V*-1:GOTO 740
760 IF B$>"@" THEN B$=CHR$(ASC(B$) AND &H5F)
770 IF B$=Q$ THEN 820
780 IF B$=" " THEN GOSUB 300:RT=RT+1:GOTO 740
790 IF B$="@" THEN 930
800 IF B$="*" THEN WA=WA+1:PA=PA+1: GOTO 830
810 WA=WA+1:GOTO 740
820 CA=CA+1
830 PRINT "(";Q$;") ";
840 FOR L=1 TO 2000:NEXT L
850 IF Q/8=INT(Q/8) THEN PRINT
860 Q=Q+1
870 GQ=RND(47)+43:Q$=CHR$(GQ)
880 IF Q$>"@" AND Q$=<"Z" THEN CODE$=TBL$(GQ-64):GOTO 910
890 IF Q$>"," AND Q$<";" THEN CODE$=TBL$(GQ-43+26):GOTO 910
900 GOTO 870
910 IF V<0 THEN PRINT " ";CODE$;
920 A$=CODE$:GOSUB 300:GOTO 740
930 Q=Q-1:PRINT:PRINT:PRINT"Characters sent:";Q
940 PRINT"Retries sent   :";RT
950 PRINT"Correct answers:";CA
960 PRINT"Wrong answers  :";WA
970 PRINT"Passes         :";PA
980 PRINT"Percent correct with retries   :";(CA/Q)*100;"%"
990 PRINT"Percent correct without retries:";(CA/(Q+RT+WA))*100;"%"
1000 PRINT:PRINT,"Press <@> to return to menu ";
1010 A$=INKEY$:IF A$="@" THEN 530 ELSE 1010
```

Data Indexing using BASIC by Paul Wade, VICTORIA, AUSTRALIA

The (not quite so recently arrived) Misosys Quarterly was again up to the professional standard that I have come to expect of yourself, please keep up the good work. Accompaning this letter is a program (written in BASIC using The Basic Answer, TBA). Files are as follows: (1) LIBENTRY/TBA, TBA source of the cross reference data entry programs; (2) LIBREAD/TBA, TBA source of the cross reference data retrival program; (3) LIBENTRY/BAS, compiled and packed basic program; (4) LIBREAD/BAS, compiled and packed basic program; (5) LDOSINFO/DAT, title file names for cross reference programs to access; (6) LDOSINFO/TXT, data file of cross referencing to the title file; (7)INSTRUCT/TXT, instructions on using the cross reference programs in ASCII but will run as a basic program.

This program was originally written by myself for a friend who is a social worker at a nearby hospital. He required some method of being able to compile data on successful methods of treatment. The example in his case was that the title was the name of the patient and the data for the txt file was the treatments provided (e.g. drugs, ect, accomodation help, welfare payments etc.), with all the data entered by specifing treatments recived he could find the patient. As this is an ongoing method updating had to be quick and easy.

The example that I have included with the disk contains the references to all your previous 'NOTES', 'the LDOS publications' and now 'TMQ'. I realise that BASIC is not the best language for such a program but looking around I could not see any other program which would allow such a file structure, since the size of the search is only limited by the disk space (versafile is about the next closest program that I know about, perhaps nobody has ever done it before?). The biggest problem I have with it is the search time; perhaps 'EnhComp' would speed it up? (for cooks state the ingredients and the program will sort out the possible recipies - imagine Gargleblasters!). You have my permission to include it in contributions (public domain but rights reserved, if your laws are as I understand them). [Because of the extent of the contribution, it will appear only as part of DISK NOTES 8 -Ed]

SCRIPSIT - previewing output

Here's a little tip for those of you still using Model 4 SCRIPSIT. We still use it here at MISOSYS for some word processing jobs; specifically letters and small documentation booklets. We have always wanted to conserve paper; thus, we have always looked for better ways to proofread rough drafts of text for form, content, and outline.

Back in July of 1984, we put together a series of patches for Tandy's SCRIPSIT program. These patches were called MPATCH and they were published in NOTES FROM MISOSYS, Issue III (the /FIX files are also on DISK NOTES III). The primary enhancement to SCRIPSIT added by these patches was the 'Q' command. This command allows you to invoke any DOS library command (or other program running totally in the library region of the DOS) from within SCRIPSIT. For instance, one can easily invoke a DIR command, or a REMOVE, or ROUTE for that matter.

The 'Q' command makes it very easy for us to preview the formatted text output of SCRIPSIT - without wasting any paper. This operation makes use of three DOS commands: ROUTE, RESET, and LIST. The idea is to redirect the printer device to a disk file with a command of the form:

```
ROUTE *PR PRINT/TXT:2
```

Since the DOS needs some high memory to set up a buffer and file control block space for such a ROUTE command, I issue this command before entering SCRIPSIT. When I am in SCRIPSIT ready to print, since the *PR device is redirected to a disk file, SCRIPSIT generates a listing file instead of actually generating the output to the printer - it doesn't know that the DOS is sending the "printer" output to a disk file, nor does it care. Using the 'Q' command once the "printing" has stopped, I can close the listing file and restore the *PR device by simply invoking a,

```
RESET *PR
```

DOS command; again from within SCRIPSIT using the 'Q' command added by the MPATCH. Now I am ready to look at the file listing. For this, I use the DOS command (invoked via the 'Q' command),

```
LIST PRINT
```

I have the convenience of the PAUSE facility (SHIFT @) to slow down the listing while I read it. This gives me the advantage of making revisions to the source text without having to waste paper, ribbons, and typewheels. And printing to a disk file is quick - especially if its a hard disk or a RAM disk.

When I am ready to generate another "listing", I can easily invoke the ROUTE command from within SCRIPSIT. That's because the DOS will re-use the high memory previously allocated during the first invocation of ROUTE *PR PRINT/TXT". Of course, I have to add the "(REWIND)" parameter to that ROUTE command to ensure that subsequent "printings" are not appended to PRINT/TXT but rather written anew.

Now another DOS facility further eases the burden of issuing these 'Q' commands; that facility is Key Stroke Multiplication (KSM). If you haven't used KSM, you better start looking at it; it can save you a great deal of repetitive typing. Here are the KSM strings I use which are pertinent to this modified version of SCRIPSIT.

```
q list print:2;;
q reset *pr;;
q route *pr print/txt:2 (r;;
Sincerely,;;;;;Roy Soltoff;
This is in response to your letter of
```

The semicolons are important. KSM allows you to specify a logical <ENTER> character which defaults to the semicolon. In the 'q' commands noted above, you will find them terminated by two semicolons; one designates an <ENTER> to complete the DOS command while the second designates an <ENTER> to satisfy the 'Q' command's prompt after completing the requested command. Finally, I normally assign the LIST string to the 'L' key, the RESET string to the 'Q' key, and the ROUTE command to the 'R' key, although you can make those assignments to suit yourselves. If you use Model 4 SCRIPSIT without the MPATCH, you are losing out on some very handy enhancements. DISK NOTES 3 are still available for $10 plus S&H ($2 in the US).

The Programmer's Corner

Using I/O redirection with PRO-MC for debugging purposes - by Michael R. Johnston - CIS #:73240,1777 - 05/06/87

Users of MISOSYS' MC or PRO-MC compilers may or may not know of the special I/O redirection inherent in the pre-processor (MCP) and compiler (MC) that are part of the package. Using these features it becomes possible to save a great deal of time and some money. In this article I will attempt to give some helpful instructions on using this feature.

Some special notes need to be made before we proceed. First, the standard output file (stdout) is normally the CRT. This file is used to display all copyright notices and informative messages such as 'Including: STDIO.H'.

Second, the standard error file (stderr) is used by the pre-processor and compiler to output ERROR messages. Normally , stderr is sent along with stdout to the CRT. It is only when one begins to play with the redirection feature of these two programs that the two begin to go their separate ways.

Turning to page 1-8 of the MC manual one will notice on the bottom of the page, a general example of how stderr might be redirected:

```
MC TESTLIB #*PR
```

This example has all error messages that the compiler outputs , going to the printer. The astute observer will note that although this example has used a printer as the target for the output, this does not necessarily mean that it could not be directed elsewhere. In fact, using the wonderful facilities of DEVICE INDEPENDENCE that were so thoughtfully included in TRSDOS, one can redirect this error message output to ANY device or FILE.

Since there are no other examples of this remarkable feature given in the manual, it is left to the user to uncover and utilize them.

A practical example is one where stderr is redirected to a file for reference when debugging. If you are using SAID (the text editor supplied with the MRAS package) you could use this error listing to aid in your debugging process. In fact, the error messages will tell you which line that the problem occurred. It is then a simple matter to use the GO command provided in the META-MENU , to locate the line for you.

The ideal setup is one where you load the C source file in one bank and have the error file in another. All one has to do then is to switch between banks when referencing error messages.

By redirecting stderr to a file , you also avoid another problem: **** THE PAPER BLOB ****. Yes, that disgusting animal that proliferates beside your desk when debugging will no longer have a fighting chance. And there is yet another advantage to this method too: Your wife won't throw your printer out the window because it is keeping her awake.

It may also be possible to write a filter that will read the error file and make a note of what line numbers are in error. The filter could then feed SAID with the GO commands and LINE numbers to position the cursor to the proper place in the file for correction. Of course this would require someone to actually WRITE a filter for this specific purpose. Which is something that most people are probably not willing to do. However in the long run it would prove to be worth it in terms of the amount of time saved looking for the line numbers in the listing and then issuing the proper GO command. Just a pipe dream of course but possibly worth looking into if your serious about pro-gramming in C on the mod 4.

Writing FILTERS for LS-DOS 6.x by Roy Soltoff

Because of the size of TMQ I.iii, a few articles (programs) had to be deferred. One was a discussion of filters for DOS 6.x. I'll pick that up in this issue; Rich's stat() function and rm utility will have to still be deferred.

Device filters have been an inherent part of the LDOS 5.x and LS-DOS 6.x environment since the early days. Device filters are kind of neat; they provide an opportunity to alter the behavior of a device. For instance, both LDOS and LS-DOS come supplied with a key stroke multiplication filter (KSM/FLT) and a printer filter (PR/FLT or FORMS/FLT), both of which alter the utility of the *KI and *PR devices, respectively.

Under the LDOS 5.x environment, Logical Systems (and subsequently MISOSYS) have made many other filters available in the FILTER1 and FILTER2 program products (now bundled into The Mark III Collection). At one time, MISOSYS offered a proportional spacing filter for the

DW-II. There also were filters as part of the MSP-02 package. For LS-DOS 6.x users, the LS-UTILITY disk (still available) has provided a KSMPLUS filter. The PRO-ESP package which we used to sell also included filters (it's bundled now into The Mark IV Collection). I would hazard a guess that our Compuserve forum also houses some filters. THE PROGRAMMER'S GUIDE TO TRSDOS 6 (once again available from MISOSYS for $25 + $3S&H) even lists three complete filters for DOS 6 in its Appendix.

On the other hand, more filters were available for LDOS 5.x than were made available for LS-DOS 6.x. Why is that? Well for one thing, the architecture of a filter was changed to accomodate the altered architecture of the device handler in LS-DOS 6.x. I changed the structure of the device handler in 6.0 for many important reasons. Here's a few: LSI received many complaints over the inability to RESET devices and conveniently re-install a filter. Numerous kludges were implemented to various releases of 5.x's filters to provide a limited form of re-installation. The LDOS device handler also did not permit filtering a ROUTEd device. Linking also took up high memory with a link module for each LINK command. It was also necessary for a 5.x filter to pick up vector values from the filtered device's DCB and poke in its own. Also, when filtering the *KI device, the filter installation code had to account for the effect of JCL operation on the *KI DCB's vector. All in all, it was very messy. Now the ground rules for the device handler were restricted by the ROM resident device I/O routines; specifically, @GET, @PUT, and @CTL handlers were in ROM - unchangeable.

The development of DOS 6.0 permitted me to clean up that mess and design a reasonable device handler. I think that got accomplished. Anyone who has read my discussion of device handling which appeared in the book mentioned above should be able to bear that out. Since most of the filters I mentioned above were written by LSI, perhaps there just wasn't sufficient time to adapt all of them to 6.x. On the other hand, they may have felt that not enough people purchased the 5.x versions which resulted in no justification to spend any further time on them.

What follows then is a look at one of the filters from the filter pack, PAGEPAWS. This module is designed to filter the *PR device and pause awaiting a keystroke when a form feed is received. That allows you to use single sheets to feed a printer when the program is expecting continuous forms. Let's take a look at the LDOS 5.x version of this filter. But first, some information is needed. Let's look at a command such as:

```
FILTER *XY PAGEPAWS
```

The DOS's FILTER library command will search the device table for the "XY" device, then load the PAGEPAWS module - assuming *XY was found. When control is passed to the filter program, register DE contains the address of the DCB for the device designated. Register HL points to the next character on the command line so the filter module could check for parameters/options/etc. Now the LDOS 5.x filter...


```
                 00001 ;PAGEPAWS/FLT   Version 5.1
4030             00002 @ABORT  EQU     4030H
4467             00003 @DSPLY  EQU     4467H
402D             00004 @EXIT   EQU     402DH
002B             00005 @KBD    EQU     002BH
5200             00006         ORG     5200H
5200  D5         00007         PUSH    DE
5201  1A         00008         LD      A,(DE)          ;type byte
5202  F5         00009         PUSH    AF
5203  217552     00010         LD      HL,SIGNON
5206  CD6744     00011         CALL    @DSPLY
                 00012 ;*=*=*
                 00013 ;       Mod I or III determination
                 00014 ;*=*=*
5209  3A2501     00015         LD      A,(125H)        ;Mod I or III?
520C  FE49       00016         CP      49H             ;III=49h
520E  2808       00017         JR      Z,M3
5210  214940     00018         LD      HL,4049H        ;Mod I HIGH$
5213  117B44     00019         LD      DE,447BH        ;@logot
5216  1806       00020         JR      LDHIGH
5218  211144     00021 M3      LD      HL,4411H        ;Mod III HIGH$
```

```
521B  118A42    00022          LD      DE,428AH          ;@logot
521E  224452    00023 LDHIGH   LD      (MHI+1),HL        ;install HIGH$
5221  225052    00024          LD      (SHI+1),HL
5224  ED537052  00025          LD      (ERROR+1),DE      ;install @logot
5228  F1        00026          POP     AF                ;device type byte
5229  CB5F      00027          BIT     3,A               ;device set NIL?
522B  2035      00028          JR      NZ,DEVNIL
522D  CB67      00029          BIT     4,A               ;Routed?
522F  2036      00030          JR      NZ,DEVROUT
5231  CB4F      00031          BIT     1,A               ;Output device?
5233  2837      00032          JR      Z,NOTOUT
                00033 ;*=*=*
                00034 ;        ok to filter device
                00035 ;*=*=*
5235  DDE1      00036          POP     IX                ;get dcb
5237  DD6E01    00037          LD      L,(IX+1)
523A  DD6602    00038          LD      H,(IX+2)          ;get old address
523D  222553    00039          LD      (OUTP2+1),HL      ;  ''
5240  222953    00040          LD      (OUTP1+1),HL
                00041 ;*=*=*
                00042 ;        install new HIGH$ and move filter code
                00043 ;*=*=*
5243  2A0000    00044 MHI      LD      HL,(0000H)        ;get current HIGH$
5246  221053    00045          LD      (OLDHI),HL        ;put in filter header
5249  012900    00046          LD      BC,LENGTH         ;len of filter
524C  AF        00047          XOR     A
524D  ED42      00048          SBC     HL,BC             ;figure new HIGH$
524F  220000    00049 SHI      LD      (0000H),HL        ;store HIGH$
5252  23        00050          INC     HL
5253  DD7501    00051          LD      (IX+1),L
5256  DD7402    00052          LD      (IX+2),H          ;install addr into dcb
5259  EB        00053          EX      DE,HL             ;DE to new address
525A  210E53    00054          LD      HL,PAWSTRT        ;HL to filter code
525D  EDB0      00055          LDIR                      ;relocate filter
525F  C32D40    00056          JP      @EXIT             ;all done
5262  21D652    00057 DEVNIL   LD      HL,NILMSG
5265  1808      00058          JR      ERROR
5267  21E852    00059 DEVROUT  LD      HL,ROUTMSG
526A  1803      00060          JR      ERROR
526C  21F952    00061 NOTOUT   LD      HL,OUTMSG
526F  CD0000    00062 ERROR    CALL    0000H             ;call @LOGOT
5272  C33040    00063          JP      @ABORT
                00064 ;
5275  1F        00065 SIGNON   DB      31,'PAGEPAWS - Line Printer Pause Filter -
Version 5.1',0AH
52A9  43        00066          DB      'Copyright (c) 1981 by Logical
Systems',2CH,' Inc.',0AH,0DH
52D6  44        00067 NILMSG   DB      'Device not active',0DH
52E8  44        00068 ROUTMSG  DB      'Device is routed',0DH
52F9  4E        00069 OUTMSG   DB      'Not an output device',0DH
                00070 ;
530E  180B      00071 PAWSTRT  JR      START
5310  0000      00072 OLDHI    DW      $-$               ;HIGH$ before filtering
5312  08        00073          DB      START-FNAME
5313  50        00074 FNAME    DB      'PAGEPAWS'
      41 47 45 50 41 57 53
                00075 ;
531B  3807      00076 START    JR      C,OUTP2
531D  F5        00077          PUSH    AF
531E  79        00078          LD      A,C
531F  FE0C      00079          CP      0CH               ;tof char
```

```
5321  2804     00080          JR      Z,OUTCF        ;go if so
5323  F1       00081          POP     AF             ;else restore
5324  C30000   00082 OUTP2    JP      $-$            ;do regular driver
5327  AF       00083 OUTCF    XOR     A              ;be sure CF reset
5328  CD0000   00084 OUTP1    CALL    $-$            ;send TOF char
532B  D5       00085 KBD1     PUSH    DE             ;save during key input
532C  CD2B00   00086          CALL    @KBD
532F  D1       00087          POP     DE
5330  FE0D     00088          CP      0DH            ;see if CR
5332  20F7     00089          JR      NZ,KBD1        ;get key if not
5334  F1       00090          POP     AF             ;else ready to return
5335  AF       00091          XOR     A
5336  C9       00092          RET
               00093 ;
0029           00094 LENGTH   EQU $-PAWSTRT          ;length of filter code
5200           00095          END     5200H
```

Let's take a look at some of this code in more detail. The first part of the program from line 1 through line 35 should be pretty obvious. The three tests at lines 26-32 are integrity checks of the device being filtered. Note that I am not going to discuss the ramifications of filtering in general, that you can get from THE PROGRAMMER'S GUIDE. The code from line 36 through 40:

```
POP     IX
LD      L,(IX+1)
LD      H,(IX+2)
LD      (OUTP2+1),HL
LD      (OUTP1+1),HL
```

pops the address of the DCB, originally passed in register DE by the FILTER command, into register IY. Then the current driver address (which could be a filter entry point) is picked up from the DCB and put into register HL. This vector is then poked into the pagepaws filter code so that the filter can chain to the current device routine.

The code from line 41 through 56 relocates the pagepaws filter code to high memory. Note that the actual filter code itself is in lines 71 through 92. The rest of the program consists of messages and error handling routines. The front end of the filter

```
PAWSTRT JR      START
OLDHI   DW      $-$
        DB      START-FNAME
FNAME   DB      'PAGEPAWS'
```

is the standard module header as documented for the LDOS environment. It consists of an absolute branch around the data area, followed by a word of storage for the previous value of HIGH$ to be inserted by the filter installation code , followed by the one byte module name length, followed by the module name. This structure is required for all LDOS 5.x filters. The guts of the filter is the following:

```
START   JR      C,OUTP2
        PUSH    AF
        LD      A,C
        CP      0CH
        JR      Z,OUTCF
        POP     AF
OUTP2   JP      $-$
OUTCF   XOR     A
OUTP1   CALL    $-$
KBD1    PUSH    DE
        CALL    @KBD
        POP     DE
        CP      0DH
        JR      NZ,KBD1
        POP     AF
        XOR     A
        RET
```

PAGEPAWS is an output filter. Since input is flagged when the carry flag is set, the code starts out by bypassing the form feed test based on the CF. To be totally correct about it, the filter should also test explicitly for the control call which is flagged by a NC, NZ flag state. Note that the jump and call vectors denoted by "$-$" were previously stuffed with the original driver vectors as obtained from the DCB of the filtered device.

The filter code is straightforward; The flag state is saved "PUSH AF" to avoid altering the flag conditions in case the filter should operate transparently during this character output. The character under test is obtained from the C register. If it is not a form feed (0CH), the original flag state is restored and the filter jumps to the original DCB driver address. This action was then totally transparent to any downstream modules.

If a form feed is detected, the flag state is forced to the state proper for output by the "XOR A" instruction (Z, NC); of course that was the state after the "CP 0CH" - but then, I didn't write PAGEPAWS. At OUTP1, the form feed character is passed downstream by a CALL instruction; this means the filter regains control after the formfeed is sent to the device (assumed printer).

The next short piece polls the keyboard taking care to save register DE as that is the only register whose contents are corrupted by an @KBD call. Once an <ENTER> is pressed, the filter issues a POP to clean up the stack. It then clears the flag register and returns to what called the module. That's it; a short introduction to filtering. Now let's take a look at the same filter rewritten for DOS 6.x

```
                 ;PGPAWS/FLT - Version 6.x
000A             LF       EQU     10
000D             CR       EQU     13
0014             @CHNIO   EQU     20
0015             @ABORT   EQU     21
000A             @DSPLY   EQU     10
000C             @LOGOT   EQU     12
0016             @EXIT    EQU     22
0008             @KBD     EQU     8
0064             @HIGH$   EQU     100
0065             @FLAGS$  EQU     101
3000                      ORG     3000H
3000  D5         BEGIN    PUSH    DE                ;Move DCB pointer to IX
3001  DDE1                POP     IX
3003  216F30               LD      HL,SIGNON
3006  3E0A                 LD      A,@DSPLY
3008  EF                   RST     40
3009  3E65                 LD      A,@FLAGS$         ;Get flags pointer
300B  EF                   RST     40
300C  FDCB025E             BIT     3,(IY+'C'-'A')    ;System request?
3010  CA6330               JP      Z,VIASET
                 ;*=*=*
                 ;         install new HIGH$ and move filter code
                 ;*=*=*
3013  210000               LD      HL,0              ;get current HIGH$
3016  45                   LD      B,L               ;Set B=0
3017  3E64                 LD      A,@HIGH$
3019  EF                   RST     40
301A  204B                 JR      NZ,NOMEM          ;Error if no memory
301C  220C31               LD      (OLDHI),HL        ;put in filter header
                 ;*=*=*
                 ;         Relocate internal references in driver
                 ;*=*=*
301F  FD213B31             LD      IY,RELTAB         ;Point to relocation tbl
3023  113A31               LD      DE,MODEND
3026  B7                   OR      A                 ;Clear carry flag
3027  ED52                 SBC     HL,DE
3029  44                   LD      B,H               ;Move to BC
302A  4D                   LD      C,L
302B  FD6E00     RLOOP     LD      L,(IY)            ;Get address to change
302E  FD6601               LD      H,(IY+1)
3031  7C                   LD      A,H
3032  B5                   OR      L
3033  280F                 JR      Z,RXEND
3035  5E                   LD      E,(HL)            ;P/U address
3036  23                   INC     HL
3037  56                   LD      D,(HL)
3038  EB                   EX      DE,HL             ;Offset it
3039  09                   ADD     HL,BC
303A  EB                   EX      DE,HL
```

```
303B  72        00051          LD      (HL),D          ;And put back
303C  2B        00052          DEC     HL
303D  73        00053          LD      (HL),E
303E  FD23      00054          INC     IY
3040  FD23      00055          INC     IY
3042  18E7      00056          JR      RLOOP           ;Loop till done
                00057 ;*=*=*
                00058 ;        Move filter
                00059 ;*=*=*
3044  ED5B0C31  00060 RXEND    LD      DE,(OLDHI)      ;Destination address
3048  213A31    00061          LD      HL,MODEND       ;Last byte of module
304B  013100    00062          LD      BC,LENGTH       ;length of filter
304E  EDB8      00063          LDDR
3050  EB        00064          EX      DE,HL           ;Move new HIGH$ to HL
3051  3E64      00065          LD      A,@HIGH$        ;Set new HIGH$ into the system
3053  EF        00066          RST     40              ;B=0 from above
3054  23        00067          INC     HL              ;Bump to filter entry
3055  DD360046  00068          LD      (IX+0),40H.OR.6 ;Stuff TYPE byte
3059  DD7501    00069          LD      (IX+1),L
305C  DD7402    00070          LD      (IX+2),H        ;install addr into dcb
305F  210000    00071          LD      HL,0            ;Successful...
3062  C9        00072          RET
                00073 ;*=*=*
                00074 ;        Error message handling
                00075 ;*=*=*
3063  21F430    00076 VIASET   LD      HL,VIASET$
3066  DD        00077          DB      0DDH
3067  21D630    00078 NOMEM    LD      HL,NOMEM$
306A  3E0C      00079          LD      A,@LOGOT
306C  C31500    00080          JP      @ABORT
                00081 ;
306F  0A        00082 SIGNON   DB      LF,'PAGEPAWS - Line Printer Pause Filter -
                                       Version 6.x',10
30A3  43        00083          DB      'Copyright 1987 MISOSYS, Inc., All rights
                                       reserved',LF,CR
30D6  48        00084 NOMEM$   DB      'High memory is not available!',CR
30F4  4D        00085 VIASET$  DB      'Must install via SET!',CR
                00086 ;*=*=*
                00087 ;        The actual filter
                00088 ;*=*=*
310A  180D      00089 PAWSTRT  JR      START
310C  0000      00090 OLDHI    DW      $-$             ;HIGH$ before filtering
310E  06        00091          DB      6,'PGPAWS'      ;Length of name and name
      50 47 50 41 57 53
3115  0000      00092 MODDCB   DW      $-$             ;Loaded with DCB pointer
3117  0000      00093          DW      0
                00094 ;
3119  2005      00095 START    JR      NZ,CHAIN        ;Go if not output
311B  79        00096          LD      A,C
311C  FE0C      00097          CP      0CH             ;tof char
311E  280C      00098          JR      Z,OUTCF         ;go if so
                00099 ;*=*=*
                00100 ;        Pass to next filter/driver
                00101 ;*=*=*
3120  DDE5      00102 CHAIN    PUSH    IX              ;Save current pointer
3122  DD2A1531  00103          LD      IX,(MODDCB)     ;P/u this module's DCB
3124            00104 RX01     EQU     $-2
3126  3E14      00105          LD      A,@CHNIO        ;Chain to the next
3128  EF        00106          RST     40
3129  DDE1      00107          POP     IX
312B  C9        00108          RET
```

```
312C  CD2031   00109 OUTCF   CALL    CHAIN          ;send TOF char
312D           00110 RX02    EQU     $-2
312F  F5       00111         PUSH    AF             ;Save RET code
3130  D5       00112         PUSH    DE             ;save during key input
3131  3E08     00113 KBD1    LD      A,@KBD
3133  EF       00114         RST     40
3134  FE0D     00115         CP      0DH            ;see if CR
3136  20F9     00116         JR      NZ,KBD1        ;get key if not
3138  D1       00117         POP     DE
3139  F1       00118         POP     AF
313A  C9       00119 MODEND  RET
               00120 ;
0031           00121 LENGTH  EQU     $-PAWSTRT      ;length of filter code
313B  2431     00122 RELTAB  DW      RX01,RX02,0
      2D31 0000
3000           00123         END     BEGIN
```

Before we dissect this filter, let's remember one thing about DOS 6.x. A filter is installed into memory by the SET command. This command passes the address of a spare DCB which has been assigned to the filter. The FILTER command also inserts into the DCB, the device name you have designated and passes the address in register DE of the DCB assigned to your filter. The action which the installation code must take is to relocate the resident portion of the filter to high (or low) memory then setup the first three bytes of the DCB.

The code in the DOS 6.x filter from lines 1 through 16 are pretty obvious. This next piece pertains to 6.x.

```
LD      A,@FLAGS$
RST     40
BIT     3,(IY+'C'-'A')
JP      Z,VIASET
```

One of the problems a user could confront under LDOS 5.1 was that nothing restricted a filter from being invoked as if it were a command program. Some folks, for instance, tried to install the LDOS keyboard driver simply by typing "KI/DVR". This, of course, wouldn't work; but the user would get no indication of failure as the driver really had no way of knowing that the entry conditions to the driver (register DE) were incorrect without some fancy checking. This problem was eliminated in DOS 6.x by a system flag which was turned on only by the SET (and SYSTEM (DRIVER=)) library commands. Thus the test being performed here is to insure that this filter was invoked via a command such as

```
SET *XY PAGEPAWS
```

It is worth mentioning here that this same flag convention was added to LDOS 5.3.

Lines 22-30 are similar to the 5.x filter counterpart; the code obtains the current value of HIGH$ and pokes it into the resident filter's module header. Lines 31-56 are new for this 6.x version filter; however, the routine is one discussed a long time ago in the LSI Journal. The routine corrects all fixed address references in the resident portion of the filter to reflect the correct value after the filter is relocated to high (or low) memory). Thus, the routine won't be dissected here. On the other hand, the code,

```
LD      (IX+0),40H.OR.6
LD      (IX+1),L
LD      (IX+2),H
LD      HL,0
RET
```

is new for DOS 6.x and is essential for DOS 6.x filter installation. Remember that the job of the filter installation section is to place the filter code into memory and set up the first three bytes of the assigned DCB. At this point, register IX contains the address of the DCB assigned. Since the filter is expecting to be associated with a device which can support output and control calls (@PUT, @CTL), the DCB TYPE bit must have bits 1 and 2 turned on. Also a DCB indicates a filter assignment by having bit 6 of the DCB TYPE byte turned on. Here the first instruction does all of that. The next two place the entry address of the resident filter into the DCB driver vector storage area. After that, it's simply necessary to clear register HL which tells the DOS that the program successfully executed. The RET transfers control back to DOS.

```
PAWSTRT JR      START
OLDHI   DW      $-$
        DB      6,'PGPAWS'
```

```
MODDCB  DW      $-$
        DW      0
```

This section of code begins the resident portion of the filter. The first three instructions are the same as under LDOS 5.x; they represent the module header. However, under DOS 6.x, the module header has been expanded. The two additional word fields are used by the DOS for linking between DCB's. The MODDCB field is loaded by the filter installation code with the DCB assigned to the filter by the DOS. Actually, the second word field is not yet used by the DOS but remains reserved for future use.

```
START   JR      NZ,CHAIN
        LD      A,C
        CP      0CH
        JR      Z,OUTCF
CHAIN   PUSH    IX
        LD      IX,(MODDCB)
RX01    EQU     $-2
        LD      A,@CHNIO
        RST     40
        POP     IX
        RET
```

The first four instructions here are similar to that for LDOS 5.x. Here I corrected the entry state test. Since @PUT (represented by Z,NC)is the only condition I want to examine, I test for this simply by branching directly to CHAIN on an NZ flag condition. This single jump is necessary and sufficient since @CTL and @GET calls each produce and NZ entry state. The "CHAIN" routine is unique to 6.x. Under the LDOS 5.x convention, we chain downstream to the filtered device by CALLing or JumPing to the address vector which was originally in the DCB of the device being filtered. Here, we may take up more code space, but we gain a consistent interface. It is necessary to preserve the state of register IX. Also, the DOS @CHNIO service call sets up the entry flag state for the next filter/driver based on the value in register B (which we have not disturbed). That's why I don't need to PUSH AF/POP AF here. The key point is that when we want to pass control to the device being filtered, we use the protocol of this "chain" routine. The "RX01" label is used by the relocation routine mentioned above. Since the CALL CHAIN instruction generates an absolute address reference, it must be corrected to the new reference needed when the routine is relocated in memory.

```
        PUSH    AF
        PUSH    DE
KBD1    LD      A,@KBD
        RST     40
        CP      0DH
        JR      NZ,KBD1
        POP     DE
        POP     AF
MODEND  RET
```

Finally, we get to the code which polls the keyboard awaiting an <ENTER> before returning to the caller. Again we preserve register DE as that register is corrupted by the @KBD service call. Under DOS 6.x, all devices utilize return codes. The convention is that a "Z" flag state indicates a successful operation; "NZ" indicates an unsuccessful operation. This filter preserves the state of the return code coming back from the @CHNIO call which passed the form feed character down the device chain. This return code state, rather than the result of the keyboard call and compare for 0DH (which would always return Z), is returned back to the caller. Thus, if the printer became unavailable during the form feed, the error status would be seen.

That's it. I don't think that this filter conversion was excessive; albeit, it was a simple filter. However, more complex filters are just built up from simple concepts. So all of you folks with copies of FILTER DISK 1 and/or FILTER DISK 2 (both of which were supplied with the full source code) can do ports to DOS 6.x - or develop some of your own filters. Send your work in for a future QUARTERLY!

8086 Assembly Language for Z80 programmers by Philip Oliver

I was recently asked by Roy Soltoff to present a short introduction to IBM PC assembly language, somewhat with the intent of targeting those of you who are already familiar with the venerable Z80 of the original TRS-80s. Being the author of both ED/ASM-86, for the IBM PC, and ENHCOMP, for the TRS-80 Model 1/3/4, I'm fairly familiar with both processors. The intent, of course, is to sspecifically asdress 80x86 code from an ED/ASM-86 frame of reference! So I've agreed, and here it is...

As a short prefix, I'll note that, from the viewpoint of the programmer, the 8088 and the 8086 are essentially identical, save for timing. The difference is that the 8088 has an 8 bit external data bus, the 8086 a 16 bit bus. Furthermore, the 80286, used in the AT, and even the 32 bit 80386, are currently operated in modes which essentially make them

fast 8086's. This will change over the next year or two (now 1987). For the time being, however, there are no widely accepted operating systems which utilize the full power of the new instruction sets. Therefore, all remarks applying to the 8086 apply to the 80286 and 80386 operating in the 8086 mode.

There are three major divisions of comparison between the TRS-80 line and the IBM PC. First of all, there's the difference between the instruction set of the Z80 and the 8086. This is essentially separate from the individual hardware the processors happen to be designed with. The hardware, such as video output and disk drives, makes up the second category. Third, and last, is a comparison between the operating systems of the TRS-80 and the IBM PC. I'm going to start at the beginning, with the processor instruction sets, and then lead on to the "higher level" things such as hardware and operating systems.

I believe an instructive way to compare two processors is to look at the following major categories: the number of registers a processor has, the flexibility of its addressing modes, and the amount of memory directly addressable. Generally, the more registers a processor possesses, and the bigger (number of bits) they are, the faster it is. Similarly, the addressing modes available will determine how hard it is (how many instructions it takes) to accomplish a particular data movement. And the amount of memory addressable by the processor determines the possible size of programs (and data) which may effectively be used with it.

I. A Comparison of Instruction Sets

First, a look at the Z80. The Z80 has the following registers: A, B, C, D, E, H, L, IX, IY, and SP [Editor note: two additional registers, I and R are special purpose registers which are not usually dealt with by the applications programmer]. The single letter registers are 8 bits wide; IX, IY, and SP are 16 bits. Additionally, certain combinations -- BC, DE, and HL -- of 8 bit registers may be treated as one 16 bit register. The two letters of the 16 bit register name corresponds to the simple concatenation of the two 8 bit registers involved; HL, for example, is made up of registers "H" and "L".

The addressing modes of the Z80 are not symmetrical; that is, certain operations may use only certain registers, and not others. One example, common to most processors, is the exclusive use of SP for stack oriented operations, such as PUSH, POP, and CALL.

Also, most arithmetic operations are oriented for use with the "accumulator", or "A" register, or the HL register pair. You cannot, for example, directly add "5" to the "C" register.

Memory transfers are also oriented primarily for use with HL, IX, and IY. The latter two registers, IX and IY, provide the most versatile memory addressing modes of the Z80. Any 8 bit value or register can be loaded or stored to memory, with an optional offset from the base register value supplied.

For the most part, the Z80 can be described as an "8 bit" processor. Although certain 16 bit operations are supported, such as addition of a 16 bit register pair to HL, or the loading and storing of 16 bit registers to/from memory, most instructions of the Z80 deal with 8 bits at a time.

Furthermore, there is an upper limit of 64K of memory directly addressable to the Z80. This is due to its 16 bit address bus. This imposes a restrictive limit on the sophistication and size of the programs which may readily be run on a Z80 based computer.

Now to look at the registers of the 8086 and compare them with the Z80. First of all, the 8086 has the following 8 bit registers: AL, AH, BL, BH, CL, CH, DL, DH [Editor's note: notice the similarity in quantity and correspondance of 8-bit registers between the Z80 and 8086]. Additionally, there are the 16 bit registers: BP, DI, SI, SP, CS, DS, ES, SS.

As with the Z80, certain pairs of 8 bit registers may be treated by some instructions as a single 16 bit register. Thus, AL and AH combine to form AX; and similarly with the other 8 bit registers, to form BX, CX, and DX. There is no convenient way, however, to "break apart" the other registers, such as SI, into upper and lower 8 bit pairs.

As with the Z80, addressing modes on the 8086 are not symmetrical. However, they are generally more flexible than the Z80. For example, there is less bias in using the accumulator, which is the AX register in the 8086, over the other registers. For example:

```
SUB     CL,5
```

is a valid instruction, which subtracts 5 from the CL register. Note, at this point, that the

8086 follows the same convention as the Z80 is terms of the position of source and destination operands. The familar Z80 arithmetic instructions which specify only one operand, implicitly specifying the accumulator as the other operand, are not supported in the 8086. For example, the Z80 instruction:

```
ADD     7
```

would be analogous to the 8086 instruction:

```
ADD     AL,7
```

where you must explicitly specify "AL", since "BL", "CL", etc. can be specified as well. Note that the use of AL and AX is generally more efficient than using other registers; one byte rather than two is needed for the opcode.

The 8086 does support a wider range of addressing modes than the Z80. The "BP", "BX", "DI", and "SI" registers may be combined (with restrictions) to form a memory address for most instructions that access memory. For example:

```
MOV     AL,[BX+DI]
```

would load the "AL" 8 bit register, or the lower half of "AX", from the byte located at the address formed by BX+DI. I won't give an exhaustive list of all the valid combinations here; but remember that they are limited. For example, [SI+DI] is NOT a valid addressing mode. In general, "SI" and "DI" are treated as "index" registers, and "BX" and "BP" are treated as "base" registers. A "base" plus a "base" is not allowed, nor is an index plus an index. Nor are combinations of more than 2 registers allowed. You may specify either an 8 or 16 bit signed offset, akin to the Z80 (IX/IY+disp8), in any of the addressing modes.

Note that the general 8086 opcode name for data movement is "MOV", as opposed to the Z80's "LD".

The 8086 is probably best thought of as a 16 bit processor. Most instructions can deal with 16 bits at a time, and the external data path of the 8086 is 16 bits wide.

It would appear, on the surface, that the 8086 is a 16 bit processor in terms of memory addressing as well. After all, the maximum register size is 16 bits long, right?

Here we get into the most complicated and vexing aspect of the 8086 family. This is known as "segmentation". Note the registers listed above that haven't been mentioned yet: CS, DS, ES, and SS.

The 8086 address bus is not 16 bits, but is actually 20 bits wide. This means that it can address 2^20, or about a megabyte, of memory. This is 16 times the memory capacity of the Z80 and this means that more sophisticated programs (or, at least, bigger ones!) can be run on the 8086. The natural question at this point, is how this somewhat odd number, not an even power of 2, is supported by the addressing modes of the 8086. The answer is relatively simple. The 8086 actually forms all physical memory addresses by the following formula:

```
(16 * segment register) + offset
```

"Segment register" means any of CS,DS,ES, or SS. Offset is what is conventionally thought of as the memory address, and is 16 bits wide. In hexadecimal, this can more easily be visualized as:

```
  S S S S 0
+   0 0 0 0
  ---------
  A A A A A  (20 bit physical address)
```

The convention for specifying a full address is: **SEGMENT:OFFSET.** That is, the segment value, followed by a colon, followed by the offset.

The 8086 decides which segment registers to use based on the context of the instruction. When fetching instructions, for example, the CS, or "code segment" (clever acronym, eh?), is used, in conjunction with the internal, conventional, PC, which represents the offset. Thus "CS:PC" would denote the full 20 bit address of the currently executing instruction.

Data accesses usually use the DS, or "data segment", register, implicitly. Thus the example given earlier:

```
MOV     AL,[BX+DI]
```

actually means load "AL" from the memory location specified by "DS:[BX+DI]", or 16 times the DS register value, plus BX, plus DI.

The SS register combines with SP to form the stack address. Note that addressing modes involving "BP" use SS, not DS, as the segment. This is due to the fact that BP is designed to access local procedure variables allocated on the stack. ES is a "spare".

It is generally possible, for MOV instructions, to "override" the default segment value. In this case, the "SEG segreg" instruction is used; or the segment name can be directly given, as with:

```
MOV     DS:[BP+DI+12H],SI
```

Rather than using the stack segment (by default), the DS, or data segment, would be used instead. Note that segment overrides only last for the following instruction.

One of the amusing (?) implications of segmentation is the fact that there are 64K ways of specifying exactly the same physical memory location, using 64K different combinations of segments and offsets (arithmetic overflow allowing).

Another implication is that is it difficult, time consuming, and annoying to attempt to deal with more than 64k of memory at a time with the 8086, due to segmentation. This is because there are no arithmetic instructions which operate directly on the segment registers, and no way to add a "20 bit" address directly to a segment register/offset register combination. The main reason for this mickey-mouse scheme is that Intel designers basically didn't want to have to deal with the issue of extending an 8 bit processor, the 8080, and so just slapped in a few extra registers with minimal instructions for dealing with them.

ED/ASM-86 manages segments in the following manner. If all the program code and data can live in the confines of one segment, or 64k, then life is easy. ED/ASM-86 will handle the overhead for you, transparently. Otherwise, you will have to define multiple segments in the following manner:

```
segname  SEGMENT PARA 'class'
         (your code or data
          definitions go here)
segname  ENDS
```

You will define one segment per segment you need. Usually, for multiply segmented programs, there are at least three segments: A code segment, a data segment, and a stack segment. MSDOS defines two executable file types: .COM and .EXE. The former are limited to 64k, though not necessarily one segment (ED/ASM-86 can actually generate multiply segmented programs and output a .COM file, unlike MASM.) The latter, the .EXE file, is designed for larger programs, and can support the 640k limit imposed by MSDOS.

Segmentation affects all aspects of data moves and program branching statements. For example, there are two basic types of jumps and subroutine calls in the 8086; inter-segment and intra-segment.

As the names imply, inter-segment jumps and calls jump between segments. They are so-called "far" references. These require 32 bits of information, the segment and offset of the target routine. Since these are long and slow, programmers try to avoid them.

Far more common is the intra-segment, or near, jump or call. These require from 8 to 16 bits of displacement information. It is interesting to note that inter-segment references specify an absolute address, whereas the intra-segment branches or calls are PC relative. This makes it impossible to directly search for instructions jumping or calling a specific address.

The Z80 has one useful feature lacking in the 8086: 16 bit conditional jumps. For example:

```
JP      NZ,target
```

branches to "target", a 16 bit address, on the non-zero condition. Amazingly, there is no conditional 16 bit jump in the 8086; only conditional 8 bit jumps or branches. Furthermore, the 8086 lacks a conditional CALL. To synthesize these extremely useful instructions, I developed a set of macros, included with ED/ASM-86, to simulate them using the 8 bit conditional branches.

The Z80 supports block move and block byte search with the instructions LDIR/LDDR and CPIR/CPDR. The 8086 has similar instructions; REP MOVSB and REP CMPSB. In the Z80, there are separate instructions to perform incremental block moves (LDIR) and compares (CPIR), and decremental (LDDR and CPIR); in the 8086, a processor flag (the direction flag) indicates the direction of the move/compare. The "CLD" instruction clears the flag and specifies a decrement; "STD" sets the flag and specifies increment.

To set up a block move with the Z80, you load the HL register with the address of the source block, DE with the destination block, and BC with the move count. Then, LDIR or LDDR performs the move:

```
LD      HL,SOURCE
LD      DE,DEST
LD      BC,COUNT
```

LDIR (or LDDR)

The 8086 operation is very similar; the SI register is loaded with the source offset, DI with the destination offset. The CX register contains the move count. The segment register used for the source is DS; the destination segment is contained in ES. In other words, source = DS:SI, destination = ES:DI. Thus:

```
(appropriate setup
of DS and ES values)
...
MOV     SI,OFFSET SOURCE
MOV     DI,OFFSET DEST
MOV     BX,COUNT
CLD     (or STD for decrement)
REP     MOVSB
```

Actually, "REP" and "MOVSB" are two separate 8086 instructions. "MOVSB" specifies a single byte transfer between DS:SI and ES:DI, incrementing or decrementing SI and DI depending on the transfer flag value. The "REP" prefix indicates that the operation is to be repeated "CX" times.

You can also move 16 bits, or a word, of data, at a time with the MOVSW instruction. This ability to move either 8 or 16 bits is common to most of the MOV instructions.

Note the use of the "OFFSET" operator. This means that the "offset" value (what in Z80 terms would be the PC value of the symbol) is used, rather than the contents of the memory location specified by the symbol. The MASM convention is to assume the reference is indirect; the ED/ASM-86 convention assumes the reference is direct, that is, OFFSET is implicitly specified.

This brings up the issue of "typed symbols" or "variables". It is possible to define, say, a byte variable in the following way:

```
XYZ     DB      0
```

or a word:

```
WVAR    DW      0
```

Obviously "DB" denotes a define byte operation, "DW" a define word. Unlike Z80 symbols, when a symbol is defined immediately prior to a "DB" or "DW", it is typed to that definition. "XYZ" in the above example then is typed as a byte variable, and "WVAR" is typed as a word. Essentially, this information can be used, if the symbol has been previously defined on pass 1, to specify the type of instruction generated. So:

```
MOV     AX,WVAR
```

would, if XYZ had been defined as above, generate the instruction to move the contents of WVAR into AX. Otherwise, according to ED/ASM-86 convention, it would move the offset of WVAR into AX. Note that:

```
MOV     AX,OFFSET WVAR
```

would unconditionally generate a direct load of the offset of WVAR into AX, regardless of the type of WVAR, with ED/ASM-86. Similarly:

```
MOV     AX,[WVAR]
```

would unconditionally generate an indirect load of the contents of WVAR into AX, regardless of the type of WVAR, again using ED/ASM-86 conventions, which are that bracket characters always specify an indirect reference. Note, however, that if WVAR has been typed, then it must match the size of the register. For example:

```
MOV     AL,[WVAR]
```

would generate an error, because WVAR was typed as a word, and AL is a byte sized register. This could actually be overridden, if required, by:

```
MOV     AL,[OFFSET WVAR]
```

The Z80 contains I/O ports which are separate from the main memory. So does the 8086, which has 65536 addressable ports versus the Z80's 256. The general form of the 8086 port output instruction is:

```
OUT     DX,AL
```

or

```
OUT     DX,AX
```

which uses the contents of the DX register to specify the port. For port addresses less than 256, a direct form may be used:

```
OUT     portnum,AL  (or AX)
```

And of course, inputs from a port are allowed as well. The syntax is the reverse of OUT's.

II. Comparison of Hardware & Peripherals

Both the IBM PC and the TRS-80 have certain

hardware standards, such as types of video adaptors, etc.

There is far more add on hardware available for the IBM PC than was ever available for the TRS-80. Aside from the popularity of the IBM PC, the reason for this is the expandibility of the PC, with its internal slots. Therefore, it would be impossible to fully cover the possible options. However, there are certain more or less standard hardware configurations.

The TRS-80 has supported only one type of video adaptor, primarily because all TRS-80s have had them built into the standard hardware; although there have been some "high resolution" add ons available.

The PC has at least 3 "standard" types of video output, all addressed in a standard way. These are the monochrome adaptor; the Color Graphics Adaptor (or CGA); and the Enhanced Graphics Adaptor (or EGA).

The monochrome and CGA adaptors are controlled by the MC6845 video controller chip [Editor's note: The MC6845 is the same type of video controller chip found in your TRS-80 Model III and 4]. Various display parameters are controllable by on-chip registers, which are addressed at standard I/O port locations.

Similarly, the display memory resides at standard locations. The start of monochrome display memory is 0B000:0, and CGA memory is at 0B800:0. The standard setup is an 80 by 25 line screen, with two bytes specifying one displayed character. One byte is the ASCII code of the character to be displayed, the other is a so-called attribute byte. Depending on the mode enabled, this can specify whether the character is to be blinking, dim, etc. The adaptors conventionally have 16k on board; it is possible to have 4 separate "screens" in memory at once in alphanumeric mode.

The CGA can also support graphics. The conventionally supported modes are 320x200 in either B&W or 4 colors; and 640x200, B&W.

The EGA is much more complicated, although it can emulate CGA mode. In one enhanced mode, 640x350 resolution in 16 colors is possible. ;sk There are standard I/O port addresses for accessing serial and parallel ports; the standard serial I/O chip is the INS8250. The PC also has DMA [Direct Memory Addressing] controller chips and timers, as well as a built in speaker with limited capability. Details are beyond the scope of this article.

As far as disk drives go, the conventional 5.25" floppy disk is a 40 track, double sided, double density drive with a 360K capacity. Hard disk sizes vary dramatically; but it's now possible to get 20 meg drives with a controller for around $400. Ten meg drives are a thing of the past. You can get off the shelf hard drives up to 120 megabytes and beyond.

III. Comparison of Operating Systems

The PC, as well as the TRS-80, has a more or less standard boot rom, or "BIOS" as it is called, which contains various useful low-level routines for doing I/O. For example, although it possible to directly control the video hardware yourself, and there are definite speed gains in doing so, the short and simple way to print things out is to use a BIOS call to do it for you.

Essentially, a BIOS routine is called with a useful, short 8086 instruction called "INT". The INT instruction is analogous to the Z80's RST instruction, although the 8086 supports 256 possible INTs compared to Z80's meager 8. INT is basically a short, position independent form of the inter-segment far CALL.

The lower numbered INTs are reserved for BIOS call usage. For example, INT 10H (note that 'H' after a number means hexadecimal) supports the video adaptors. For example, to do a simple "teletype-like" character print on the screen:

```
MOV     AH,14
MOV     AL,char
MOV     BL,foreground color
MOV     BH,display page #
INT     10H
```

And there are various other routines which set up serial communications, send a byte to a printer, do disk I/O, and so forth. All of these are completely standardized in terms of the INT # involved and the registers used for parameters; with the possible exception of certain limited extensions for non-standard hardware. The IBM BIOS technical reference manual describes all of them in detail, as do many other reference sources.

The TRS-80 has (had) a long line of varying disk operating systems. One of the first was "NEWDOS"; and of course LDOS.

Somewhat strangely, there has never been a real competitor to MSDOS. It is the standard operating system for the IBM PC. Essentially, MSDOS has its roots in a cheap CP/M clone

hacked up at some time, and purchased by Microsoft and cleaned up a bit. As with the BIOS, MSDOS contains its own set of system calls which are accessible to application programs. Most of these calls are accessed via the INT 21H instruction; the AH register is loaded with a function code, along with other pertinent registers, as entry parameters. Various utilities such as file access are supported by MSDOS.

IV. Conclusion

The future of the IBM PC world is hazy at this point (as is the rest of the future!) With the introduction of their new machines, IBM has essentially said "to hell with existing hardware and eventually existing software". This move seems to be motivated primarily by a desire to squelch competition from the clone makers.

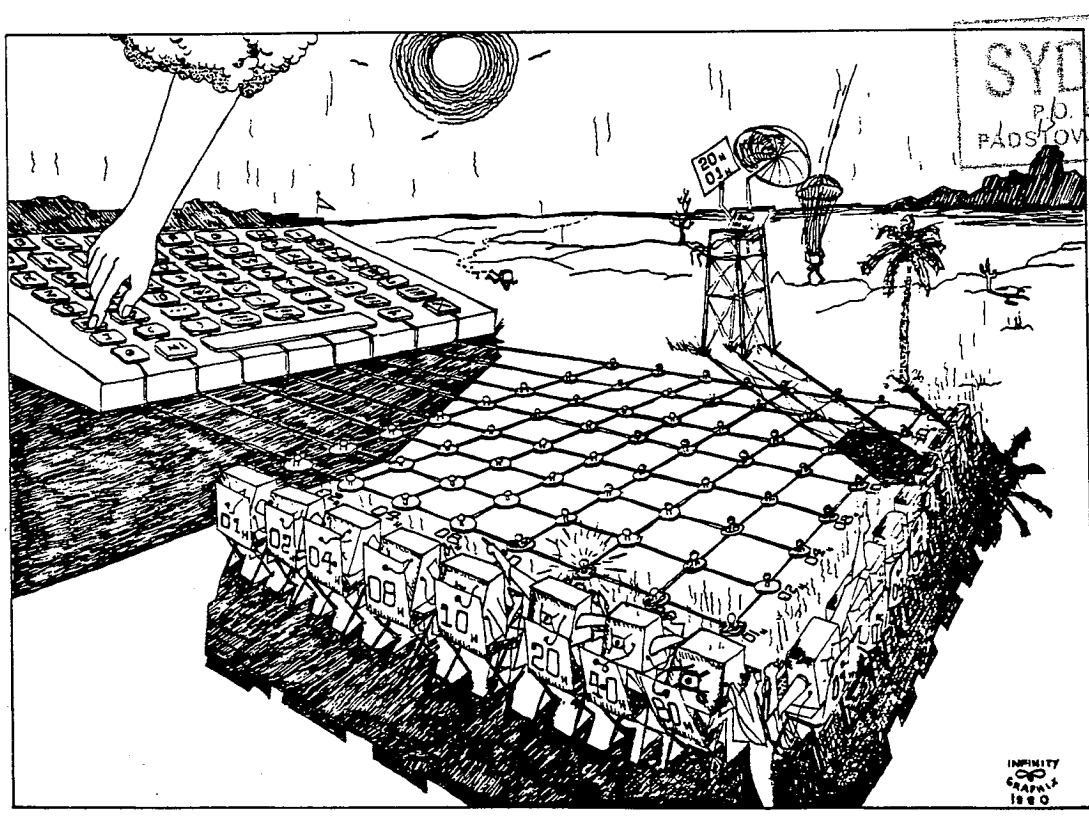
This is leading to divergent pathways. On the one path will be the millions of IBM PC clones which will almost surely continue to be manufactured like popcorn, with the attendent low prices. On the other will be IBM, aiming at the corporate markets who want 3270 terminal capabilities and the Blue label. What this holds in store as far as the operating system goes is anyone's guess.

The 386 based machines will become more popular, and cheaper. But here, there are no set standards, either hardware or software. Microsoft has indicated that a 386 operating system is at least two years away -- probably more considering their usual schedules. What will the market be? There are indications that UNIX will become a popular OS for the 386. I do intend to start doing development with the 386. If so, then ED/ASM-86 will, as it has for the past 2-3 years, continually evolve to a new and better lifeform.

In any case, regardless of the way the hardware and operating systems work out, 8086 assembly language will be a common denominator of everything from IBM PC clones to 386 powerhouses. At least until a truly nice processor like the 68020 is widely accepted (grin).

My next article, if/when there is to be one (depending on the readers' attitude towards this one), will concentrate specifically on using ED/ASM-86 to get acquainted with 8086 assembly language. I intend to use this article as a base for the next, presuming a little bit of familiarity next time.

[Editor's note: You may address these 80x86 and ED/ASM-86 isses with Phil on our Compuserve forum, PCS-49. Philo's PPN is 72437,2057.]



Product Tidbits

ALTRES (part of PRO-ESP)

(**Fm: Paul Bradshaw**) After a little digging, I have found the patch areas that allow the ALTRES/CMD program (from your PRO-ESP package) to be moved around from Bank 2. This specific patch modifies ALTRES to use bank 4, but it may be modified to select any bank available. This makes it possible to move ALTRES out of the way for those who have the Alpha Tech or XLR8er boards (leaving banks 1 and 2 free for applications that MUST use one of those banks). See ALTBANK/FIX by Paul M. Bradshaw in The Patch Corner.

Assemblers - PRO-CREATE

(**Fm: Clay S. Scott**) While reading throught THE PROGRAMMER'S GUIDE TO TRSDOS6 and other source listings, I have noticed the frequent use of the expression ($-$) as operand in source statements as "LD HL,$-$", "DW $-$", etc. Since it appears that this expression will always evaluate to zero, I wonder why it is used in lieu of the constant value zero. I own PRO-CREATE Ver. IV and would appreciate an explanation for this practice.

(**Fm: MISOSYS**) It has become common practice to use the "$-$" nomenclature when coding a word space which is to be subsequently altered during runtime. Thus a quickie look at the source code gives you the knowledge of the dynamics of the code; if you use the technique consistently.

DED86 - MS-DOS based disk editor

(**Fm: Jim Gaffney**) I received DED86 today and I'm impressed! It's all that I anticipated it might be and more. I particularly like the documentation and having the foldout pages to illustrate the displays while you read the docs sans a computer is a masterstroke.

I assume from the docs that you would have no problem with the DEMODED.EXE being uploaded to the bulletin board circuit? (Not the DED.EXE file.)

DSM - Disk Sort and Merge

Earl Jamison gave us a brief report that DSM4 aborts a Job Control Language file at the conclusion of processing. That's the Model 4 version. Bill Norman gave us a similar report concerning DSM51 (Model III version). Bill's report was that if you invoke a DSM51 MAP run from LBASIC (5.1.4) via a CMD"DO...", when it gets back to LBASIC, the program which invoked the CMD"DO aborts. Because of two similar reports, I thought it best to go over the documentation and evaluate the two reports.

In my DSM4 test, I sorted a 1200 record file of 128 byte records. The JCL file was:

```
. this is a test of dsm4 via jcl
dsm4 usermap jcl
s
n
userclub/idx:2
a
2
. this is the jcl continued
```

I experienced absolutely no problems with this procedure. If anyone has difficulty using DSM4 from JCL, you must send me complete details. I will need a copy of your data file. I will also need your MAP file, your JCL file, and specific details concerning all of the responses you made to the DSM4 module which generated that MAP file. Please provide all of the files on a floppy disk.

I also investigated the report of DSM51 aborting the running BASIC program which invoked a sort job via Job Control Language and BASIC's CMD"DO commandstring". In my test, I sorted a 1200 record file of 128 byte records. My test BASIC program was:

```
10 PRINT "Here before invoking DSM"
20 CMD"DO = TESTDSM"
30 PRINT "Here after sorting"
40 END
```

The JCL file was:

```
. this is a test of dsm51 via jcl
dsm51 userclub/map:4 (jcl)
userclub/idx:4
a
4
. this is the jcl continued
```

I experienced absolutely no problems with this procedure. If anyone has difficulty using DSM51 with JCL from BASIC, you must send me complete details. I will need a copy of your data file, a copy of your BASIC test program which demonstrates your DO invocation and abort after return (please slim down your program to the bare essentials neccessary to reconstruct the error). I will also need your

MAP file, your JCL file, and specific details concerning all of the responses you made to the DSM51 module which generated that MAP file. Please provide all of the files on a floppy disk.

ED/ASM-86™ 80x86 assembler

(Fm: MISOSYS) I have made arrangements for Phil Oliver, the author of our ED/ASM-86 assembler product, to check into our Compuserve forum more frequently than he has in the past. So if you have any questions concerning ED/ASM-86 or 80x86 assembly language, please leave a message for our "expert". Phil's PPN is 72437,2057.

(Fm: Bob Haynes) Will be getting into MSDOS assembly in the near future and am interested in your ED/ASM-86 and DED86 packages. Forgive my ignorance of 16 bitters, but here's a couple of questions: Will a DSMBLR-86 also become available? Do these packages run on any MSDOS system? What sort of compatiblity exists in the 8086/8088/80x86 instruction sets, and how do your packages fit in as a broad range development package? I'd really like to have the same kind of professional software under that environment as on the Z80.

(Fm: MISOSYS) DSMBLR-86 will most likely not be done. ED/ASM-86 includes a memory disassembler which generates labels. Considering that the debug portion can load an EXE file, what more do you need?. All of the ...86 packages run on an MS or PC DOS machine. Some (like FED86) require a "true" compatible. DED86 includes a version for the T2K. Compatibility in the 8086... instruction sets? Do you mean compatible across them or compatible with Z80? DED86 is for everybody. Just like everybody using a TRS-80 needs a good "disk zapper". Obviously, ED/ASM-86 is only for a programmer. It's architecture is perfectly suited for those folks wanting to learn 80x86 assembly language since it combines the editor, assembler, linker, debugger, and disassembler into one memory resident file.

ED/ASM-86 questions #1

(Fm: Nate Salsbury) Roy - this is sent to you from the bottom of my learning curve! To gain some experience, I tried reproducing Hardin's tutorial "Listing 3" in the June, '87 issue of '80 Micro', page 44.

After a LOT of cut and try, I ended with a program that would assemble into memory without error. My stickiest problem (e.g. it took the longest to "cure" - by brute force, NOT reason!) was when the 'wrt_ln' procedure was assembled. It produced "Symbol value difference between passes" for wrt_ln, and wrt_l as well as mov_curs and wait_key.

This problem "went away" by changing the second line of code in wrt_ln from

```
mov bl,attrib
```

to

```
mov   bl,[attrib]
```

(As an old Z80 man, that was what I was inclined to do but your manual says on p. 47: "If a symbol has been defined... then an indirect reference will be assembled whenever the symbol is referenced." This confusing (to me) system is what MASM does as shown in Hardin's listing.)

In the third-from-last line of code in the same module, I had to revise it to

```
mov   [attrib],bl
```

and the next line had to be changed to

```
inc byte ptr crs_row
```

before I could get a "good" assembly. Oh, also in the set_curs procedure I had to use

```
mov    dh,[curs_row]
```

Are these truly differences in design between MASM and ED/ASM-86? From the sentence in your manual which I quoted earlier, I had expected ED/ASM-86 to work the SAME way as MASM.

(Fm: MISOSYS) Nate, in investigating your report, I found that your quote from the ED/ASM-86 manual was missing a VERY IMPORTANT WORD. You should have quoted, "If a symbol has been previously defined immediately before the DB, DW, ..., then an indirect reference will be assembled unless overridden." You left out the word "previously". Now in Hardin's listing 3, the symbols CRS_ROW and ATTRIB are defined in a DATA segment which is placed after the CODE segment in the source. Since that is not "previously", I must assume that the assembler did not treat the non-bracketed references as indirect. Just for the heck of it, I moved that DATA segment block to precede the CODE

segment and re-assembled. Didn't get any errors. I'm not sure what MASM does in handling references to symbols not yet defined. If MASM doesn't generate an error, then I guess it defaults to always assuming an indirect refference unless OFFSET symbol is specified. That may be arbitrary to an assember's implementation. My suggestion is to always use the brackets to indicate indirection to the observer of the source code. In that way, you know what is the case.

EnhComp BASIC Compiler

(Fm: Clyde W. Preble) Received PRO-EnhComp in good condition and have converted several of my BASIC files. However, when I first tried to convert the "Russian Roulette" game, I got error conditions in lines 280 and 305 [280 PRINT"CLICK":SOUND 0,0 and 305 SOUND 0,2].

When I deleted 'SOUND 0,0' in line 280 and 'SOUND 0,2' in line 305 the program converted correctly and works all okay except the SOUND, of course, is absent.

When I tried to transfer EnhComp to TRSDOS 1.3 I got errors which showed that the files were not converted and were listed as protected files which as far as I can determine are not protected. The disk I formatted for the conversion formatted correctly and the four EnhComp files transferred correctly before I tried to convert the disk to TRSDOS 1.3, and the TRSDOS 1.3 disk had ample granules available.

(Fm: MISOSYS) Here's some answers for your EnhComp questions. EnhComp does not support any "SOUND" statement. It's not listed in the manual. On the other hand, a little COMMAND addition in Z80-MODE should allow you to adapt that Russian Roulette program very easily. Here's a little code which adds a SOUND statement and demonstrates its use:

```
DEFINT A-Z
FOR D=0 TO 4
FOR T = 0 TO 7
%SOUND(T,D)
NEXT T,D
STOP
COMMAND SOUND(TONE,DUR)
Z80-MODE
LD A,(&(TONE)):AND 7:LD B,A
LD A,7:SUB B:LD B,A
LD A,(&(DUR)):AND 31
RLCA:RLCA:RLCA:OR B:LD B,A
LD A,104:RST 40:RET
HIGH-MODE
ENDCOM
```

You would need to include the code verbatim from "COMMAND SOUND(TONE,DUR)" through to "ENDCOM". A statement such as the fourth line will beep the requested sound. Although this syntax 'SOUND(tone,duration)' is a little different from 'SOUND tone,duration' in Microsoft's BASIC, it's a straightforward conversion.

For our other readers, this is another example of the ease in which an assembly language routine can be integrated into your BASIC programs without USR, CALL, or packed strings.

None of the EnhComp files are protected. The Model III EnhComp comes supplied on a stripped down LDOS 5.1.4 disk. To transfer the files over to TRSDOS 1.3 (if you really want to), FORMAT a 35-track Single Density disk using LDOS. Then copy the BC, CED, S, and REF "/CMD" files and the SUPPORT/DAT file over to that 35-track disk. You should have no problem in using TRSDOS 1.3's CONVERT utility.

(Fm: Bob Connors) Was just using EnhComp with LDOS 5.3.0M and came across an interesting fact. It appears that when using the FORMS command with PR/FLT installed, if I specify only one parameter on the command line, FORMS resets all the other parameters to their DEFAULT (and no, DEFAUT is not specified on the command line). Is this the way it should work? Could it be that the default for DEFAULT in FORMS is set to ON for the MAX-80?

Also, why isn't EnhComp set up to use the PR/FLT. I have PR/FLT set up as follows (parameters): P=66,L=57,M=5,C=75,T,F). While EnhComp will use the margin setting (and I assume number of characters on a line), it does not abide by the lines per page or the hard top of form code. Instead, it uses (apparently) its own internal printer routine. Line feeds for TOF are most annoying! Is there anyway to get EnhComp to use the system configuration at least as far as the printer routine is concerned?

(Fm: MISOSYS) That's the way it works. I didn't realize that TRSDOS 1.3 has a PR/FLT. That's what it would take to have EnhComp use it. Although I would certainly like to see LDOS as the standard DOS for the Model III, it just ain't so. Thus, EnhComp was not written to use the LDOS PR/FLT formatting. That's the real world. I don't think there is a way to

get EnhComp to use the LDOS PR parameters short of writing a library function in Z80-MODE to accomplish that. Perhaps this message will tickle me to post the internal-to-EnhComp storage names which pertain to the print formating (I assume you are referring to compiled stuff).

Here goes for the internal variables:

```
@@PAGELEN     - page length
@@LINESPAGE   - printed lines per page
@@LMARGIN     - left hand margin
@@RMARGIN     - right hand margin
@LINESP       - internal line counter
```

Now the left/right margins can be set with BASIC commands LMARGIN and RMARGIN; LINESPAGE will set the printed lines per page; and PAGELEN changes the first value. This means that you really don't need to twiddle with any of those variables at the assembly language level unless you want to poke the number of lines already printed.

(Fm: Bob Connors) What I was referring to was the use of the "H" command of CED. I have the PR/FLT parameters set and CED does not recognize them at all. Instead of getting the hard formfeed after 57 lines, I get that, then I also get the formfeed (aka line feeds) from CED when it apparently prints 60(?) lines. This tends to throw off my listings pretty much. However, if it cannot be fixed (since it ain't broke), then I guess I will have to live with it! Wish there was someway to change the number of printed lines per page though, similar to the way EDAS does it, as well as the margin (although CED does honor the PR/FLT margin setting).

(Fm: MISOSYS) CED does not "honor" the PR/FLT margin setting; since it uses @PRT, the PR/FLT still is doing its thing. I made reference to TRSDOS 1.3 since EnhComp (that same version) works with Tandy's DOS, as well. EnhComp does not attempt to detect what DOS it is running on so it could check to see if PR/FLT is in use and then use those parms. In fact, if it did do that, it would then be incompatible with 5.3! And then I would not be changing it, either. My best suggestion would be to give you zapping instructions to the SUPPORT/DAT library to totally deactivate EnhComp's print formatting; but I really doubt that that is what you want.

(Fm: Bob Connors) Roy, I have been working on the following code for 4 days. Reason: it will not compile. This is a stripped down version of the code (all REM statements removed) so that I can fit it into this message. When trying to compile using the "RUN" command from CED (invoked by "S"), I get an error message, "Source file not in directory." When using the command line "BC RECEIVE/BAS,,,-WE-NO" from LDOS, "Pass # 1" displays two times and the system dies. I have checked the syntax over and over and cannot find any errors. Do I have a defective compiler? I have applied all patches to the EnhComp package through TMQ I.iii. By the way, with the REM statements included, the system just dies no matter how the compiler is invoked. My EnhComp serial number is 00027 and the errors occur under both LODS 5.3.0M and LDOS5.1.4M. Thanks for your help. Code follows. [trimmed down to the bare essentials -ed]

```
IF CHAR=&H8 THEN
  IF COUNT>0 THEN
    DEC COUNT:POKE POSITION+COUNT,32
    GOSUB "OUTBYTE"
    GOTO "GETKEY"
  ELSE
    GOTO "GETKEY"
  ENDIF
ENDIF
```


(Fm: MISOSYS) There were 5 lines in the program which were causing EnhComp to crash. Each was of the form, "IF expression THEN". The syntax of "IF" is either "IF exp" or "IF exp THEN action". Of course, EnhComp should not crash. Patches BC53/FIX and BC63/FIX to BC/CMD which change compilation from a crash to a displayed "Syntax error" message are in The Patch Corner. After applying the patch, you will "correctly" get a syntax error.

(Fm: Bob Connors) I assume (here I go again, assuming) that the culprit is the first "THEN" of each IF statement shown? So instead of "IF exp THEN IF exp..." I should have "IF exp IF exp THEN... ELSE ....", is that correct?

(Fm: MISOSYS) That's sort of correct. If you look at page 4-50 of the manual, you will see that the IF statement has two forms. You were trying to use a combination of both of them. "IF cond progcode ELSE progcode ENDIF" is one of the forms. "IF cond THEN action ELSE action" is another. You can't have "IF cond THEN IF ..." since the second IF starts a new statement. That doesn't match either of the acceptable forms. The compiler didn't check

for the absence of an "action" once it detected the "THEN". Use of the "THEN" required an action because the compiler didn't trap the error. It would be wrong to just forget the "syntax error" (I could have done it that way in the patch); it would be wrong since the omission of an action after the THEN violated the two forms of the IF. Got it?

(Fm: Richard C. Baker) I recently purchased your PRO-EnhComp BASIC compiler. So far I have found it totally misrepresented. I ordered it on the basis of a glowing review in Micro 80 magazine.

The biggest criticism I have is your manual. It explains very little. I am not an advanced BASIC programmer but I can get practically nothing reading your manual. The examples are uselessly simplistic. These should show all major possible variations of function uses, as the TRS BASIC manual does.

What I would most like to understand is: just what is the procedure for compiling existing programs written in TRS Model 4 BASIC. I have tried several of my programs which run very well with the interpreter but will not compile by EnhComp without a multitude of errors. And I can not discern what the errors are. Most of them are listed as 138 and 139. But try as I might I can not figure out what is wrong or how to fix them to work.

Also, I would like examples showing the use of the "USING" function in a working program. I have tried every way I can but this function will just not work. Even your examples won't work. Nor will any but the very simplest versions of "PRINT USING". But what about printing variables? My programs need to print several variables of all different formats on the same line. This has to be in a loop processing lots of different numerical values and strings.

(Fm: MISOSYS) Nothing in our advertising or in our catalog states that PRO-EnhComp will compile a program written to run under Microsoft's BASIC Interpreter used on the Model 4. There is no commonality amongst BASICs; each implementation may include more or less than some other manufacturer's interpreter or compiler.

Your third paragraph suggested that EnhComp should compile your existing BASIC programs. Quite the contrary, the review in 80 MICRO (which you reference to be your reason for purchasing EnhComp) states, "While EnhComp bears a strong relationship to Basic, the word from MISOSYS is that there is no attempt to make a Basic-compatible compiler. Besides, differences among Basic interpreters for the Model I/III/4 would make the chances of compatibility practically nil." The term "Basic" as used by the reviewer represented the Microsoft BASIC interpreter as distributed with TRSDOS 6 under license to Tandy Corp.

Your criticism of the EnhComp reference manual may well be a personal one. There are literally hundreds of books covering the BASIC language. Every interpreter or compiler does not need a detailed tutorial on the language since each BASIC verb and function should operate similarly. You will find that true with EnhComp. Besides, you already have a tutorial in your existing BASIC Interpreter reference; although I would question the term "tutorial" applied to that manual.

I don't understand your problem with the USING function. It should work exactly like interpretive BASIC in "PRINT USING str$;var,..." as well as a more powerful string function as in "var$ = USING str$;var,...". Thus, the formatted result can be manipulated further in the assigned string. This you cannot do with Microsoft's Basic! Besides, the examples do work just like they are documented (as modified by the note in README/TXT). For instance, the first example of 'USING "###.##";1.5555,2.6666,3.9999' could have been written as 'USING "###.##";var1,var2,var3', or 'USING "###.## #####.# #.#";var1,var2,var3', etc. I don't understand your confusion. Perhaps our stating that USING is a function that can assign the result to a string may have led you to believe that you couldn't use it normally. For instance, consider the analagous pair of statements, 'PRINT TIME$' versus 'T2$=TIME$:PRINT T2$'. See the similarity?

Compile time errors usually result from either invalid syntax or use of verbs not supported by EnhComp (like COMMON, CHAIN, EQV, IMP, ERASE, CALL, WHILE, WEND, WAIT, DEFUSR, NAME, or SOUND). You can't open files without first specifying the number of buffers with ALLOCATE. It does support a lot of other things that Microsoft's BASIC doesn't (like a built-in array sort, graphics, random files greater than 256 LRL, strings greater than 255 characters, etc.). You may be fighting it instead of trying to work with it. I would suspect that most programs would need some degree of modification.

Finally, if you still run up against a brick wall, send me a copy of a program on DISK which you cannot get to work.

(Fm: James J. Chouinard) I am experiencing problems with the output from a program compiled with your basic compiler. The output is not consistent. Variable "K" is supposed to be .02 and is okay when output without using (PRINT USING), is used. Any help you can provide would be greatly appreciated.

(Fm: MISOSYS) We have been able to reproduce the problem and also narrowed its scope to PRINT USING a certain fractional power of ten and multiples of it. For instance, PRINT USING "##.##";0.01 prints as "1.00"; .02 as ".20"; .08 as ".80". Because of the complexity of the floating point to ASCII conversion program under an edit mask (i.e. USING), I referred this problem back to the author of EnhComp, Phil Oliver. Phil has isolated the cause of the USING problem which you reported concerning our EnhComp compiler. A one-byte patch to the SUPPORT/DAT file should correct the error. This is SDAT53/FIX and SDAT62/FIX in The Patch Corner

(Fm: MISOSYS) In TMQ I.iii, Mike Harrow asked about the Data Table length values issued during an EnhComp compile. Here's what the tables are for in order from 0-17: variable names, label names, label addresses, variable types, variable addresses, data forward references, line numbers and resolved addresses, IF forward references, COMMAND names, COMMAND addresses, FUNCTION names, FUNCTION addresses, FUNCTION types, ERROR list, reserved, reserved, reserved, COMMAND/FUNCTION local variable pointer list. For what its worth, there it is.

MC - C Compiler changes for x.3?

(Fm: Donald P. Vincent Jr.) Are there any changes required to PRO-MC to support LS-DOS 6.3?

(Fm: MISOSYS) We expect to have an upgrade to MC later this year (when I get unburied). We expect to tweak the time functions to make use of the time field and extended date for stat() & fstat(). Stat() is a newie. When we have something to announce, we'll announce it.

MC and GREP speed

(Fm: mark harris) By the way, grep compiles and works under PRO-MC. Assuming that GREP4/CMD was compiled under the AZTEC C, PRO-MC optimized CMD file is about 3K smaller but seems to run a tad slower. Speed is only an issue on certain expressions. Ex. ".*foo" takes a lot longer to process than "foo.*"

(Fm: David Huelsmann) One explanation that might account for the speed issue is that AZTEC C uses a 1 K buffer for file I/O.

(Fm: mark harris) Could be, but target search file was on memdisk. Therefore disk I/O speed shouldn't figure in too much. I should point out that I didn't run exhaustive tests and the difference in speed was slight. (For example, something like AZTEC's 10 seconds to PRO-MC 11 seconds.)

(Fm: David Huelsmann) Even on memdisk, you would see some impact and the slight impact you described would be consistant, I believe.

(Fm: Les Mikesell) Yes, I also compiled grep with PRO-MC to test it. (The odd looking asm code at the end that watches for the break key press appeared to be the only way to interface to both compilers). You can reduce the size a bit more by using PRO-MC's wild-card expander instead of the coded one, but then you give up the ability to use * followed by an alpha character in the filenames. In the (very) limited testing I did with the two versions the AZTEC version ran nearly twice as fast. I expected this on floppies since the AZTEC compiler uses buffered access (reads in 1K chunks), but I was surprised that the difference remained consistent even on a memdisk.

The other point in favor of AZTEC is that I can link to a mod 3 version of their library (modified to also work with mod 1 LDOS) while running in the Mod 4 mode.

(Fm: MISOSYS) You could do the same thing with PRO-MC if you have the MC model III (also Model I compatible) version package also. There is nothing specific about each implementation except the supplied libraries and where the executables run. Although I had intended to make the Model III libraries available for some fee, there was never

interest sufficient to even justify grooming a disk and documentation to explain how to do it. The few folks needing Model III libraries have purchased an MC package to go along with their PRO-MC package - it's really not that expensive.

(Fm: Mark Harris) Did you optimize the PRO-MC code and then compare it? I didn't notice a factor of 2 difference in speed, but I didn't research very much either. In any case, PRO-MC seems fast enough.

(Fm: Les Mikesell) Yes, I did optimize the PRO-MC code. To see the difference, you must make sure the search time is significant compared to the time taken to load the programs. Try searching all the files on a couple of disks.

MC and Separate compilation

(Fm: Nate Salsbury) I have been attempting to 'take apart' LU/CCC and compile parts of it separately to use in my LRUN project. I, like (undoubtedly many others), am having problems in getting the proper SYNTAX of all the various things the compiler looks for.

My present problem stems from what is probably an incorrect concept of the "extern" principle. Here is a snippet of code, mostly taken directly from LU/CCC. The things that I have added are noted with a "<==" which, of course, are NOT in my REAL source code!!

```
extern char lbrname[] ;      <==
extern char nulls[] ;        <==
extern int num_slots ;       <==
#define MAXFILES  256        <==
find_memb(fname)
  char *fname ;
{ int slot ;
  if (get_directory() == -1) error
("Fatal error - program terminated\n");
      slot = find(fname) ;
  if (slot == -1) {
      printf("%s is not in the library!\n",
fname);
         return (-1) ;
     }
  return (slot) ;
}
get_directory()
{ unsigned size ;
  int i,dircrc ;
  unsigned cksum=0 ;
  extern struct _ludir ludir ;    <==
  extern struct _ludir ldir[] ;  <==
  lseek (lbr,0L,0);
  if (read(lbr,&ldir[0],sizeof(ludir))
!= sizeof(ludir)) /* First record */
       return(-1) ;
num_slots=(ldir[0].len*128)/sizeof(ludir);
```

The pre-processor (MCP) doesn't have any trouble with this but when MC starts up (I am only trying to get the /REL module to put into a /LBR) it promptly stops with the following error message:

```
.. get_directory() + 9: error 79: Unknown
struct/union
if(read(lbr,&ldir[0],sizeof(ludir))!=sizeof(lud
ir))
```

and, if I let it continue, it barfs on '&ldir[0]' next. I tried this, first, with the 'extern struct etc.' declarations at the very top of the file, along with the other 'extern' declarations. Same result - in fact, that's why I moved them into the body of get_directory().

The comment in the MC manual for Error 79 is: The structure/union object being declared or referenced has not had a template defined for it. [I had expected that the 'template' would be the external one at the beginning of LU/CCC which is going to be part of my main().]

I did notice one strange thing when I examined my /TOK file with LSFEDII. In the phrase "extern struct _ludir ludir" the 'extern' and 'struct' keywords were replaced by tokens, the '_ludir' and 'ludir' were ASCII characters as expected, but the SPACE between _ludir and ludir was replaced by a token (?) B0. I also looked at the source file and that space is REALLY a space - it's 20H just like it should be.

There is (undoubtedly) something I'm doing wrong vis-a-vis the 'extern' concept. Can you make me any smarter on this arcane subject? I really need to be made smarter on THIS point!!

(Fm: H. Brothers) "extern" means "the SPACE for this object is allocated in another module, and will be accessible at link time."

"struct" does not necessarily allocate any space. Each module which refers to the structure must have a copy of the template so it knows how to get to the structure members. Only one module should have the actual allocation of space. A handy way to do all

this is to put your global structure declarations in a project.h file and simply include them in all modules for the program. Just don't use them to allocate space in the header or things will really be weird!

(Fm: Les Mikesell) There are a couple of rules you have to observe when compiling modules to be linked together. The variables that are global to more than one module must be defined in only one file and declared "extern" in all the others. Preprocessor #defines, structure declarations and typedefs must be repeated in each file. This is typically done by creating a "header" file with the necessary common information and #including it in each dependent source file. Note the difference between declaring a structure and defining one:

```
struct junk {
      char line[100];
      struct junk *nextjunk;
     };
```

is a declaration and does not allocate any memory for storage. However, sizeof(junk) will be known to to compiler, and:

```
(B)
struct junk pile;
```

would then create an instance of the structure.

```
(C)
struct {
      char line[100];
      struct junk *nextjunk;
     } junk;
```

is a definition and allocates the memory for storage. Normally you would put a declaration like (A) in the header file and create the structs using syntax (B) where needed in each module. You probably only have syntax (C) in one module, with nothing to identify the structure in the other.

(Fm: Nate Salsbury) Hardin, Your comments regarding 'extern' and 'struct' made eminent sense and, even better, were right! The module compiled without a hitch as soon as it found the template!

Now, I have read a REAL LOT about 'C' (even including my MC manual if anyone is willing to believe THAT) and I have NEVER seen any caveat about THAT problem before (e.g. having to include a "struct" template in every file using said structure.)

As I read all the comments, a 'struct' is a type of variable and variables can be declared as 'extern' and presto - all the 'problems' would be cleared up at "link time". This omission is not just in the MC docs - I have never seen a hint about it in ANYthing I have read.

Well, NOW I know (one more thing about this weird 'C' language)! I wonder if I'll ever get enough knowledge to feel as comfy with 'C' as I do with B*S*C.

Les - I think I've got it. Thanks for that lucid explanation. One reason for my confusion is that I have NEVER seen the illuminating sentence you wrote.... to wit: "Preprocessor #defines, structure declarations and typedefs must be repeated in each file."

I would have intuitively decided on the #defines but have NEVER seen even a HINT that structures and typedefs needed repeating as you suggested with the #include technique.

Your 'guess' was exactly right - the structure definition was in another, separately compiled module. When I added it to the one in question, all smiles... until the next 'mystery' comes along!

(Fm: LDOS Support jjkd) See the new Dr. Dobbs (not the Baby Duck issue, the next one) for a clever macro solution so that you can use a single external file to handle both situations.

[Editor's note: it was the Flotsam and Jetsam article on page 138 of the April 1987 issue.]

PRO-MC and Variable Initialization

(Fm: Mark Harris) Is there an easy way to have MC initialize all variables to contain zeroes when they are created? A fair amount of C programs assume zero initialization. If I've missed something in the manual, I offer my apologies. By the way, after several months with PRO-MC, I remain impressed. It's a very solid package.

(Fm: MISOSYS) Check out the "-Z=Y" option in MLINK. It will "generate hex zeroes for DS regions in DSEGs...". If you want that, it

will make the linking a little slower and take up a lot more /CMD file space since all your declared buffers will be zeroes in the output file instead of a "DS" null space. If you happen to have a "char buffer [5000];" in your program, your /CMD file will be 5000 bytes+ bigger. That what you want? If you do, then add the "-Z=Y" switch to the JCL line which invokes MLINK immediately following "MLINK".

(Fm: LDOS Support jjkd) No 'C' program should ever assume that a variable not explicitly initalized contains anything of significance. I can't even think of a single 'C' compiler that this would work with. Can you give any examples?

(Fm: Mark Harris to MISOSYS) Perfect. Now don't get me wrong. Programs written in any language that assume some default initialization of variables make me ill. But reality being what it is, I have come across a lot of public domain C programs that assume all pointers and ints are initialized to zero. As you might imagine, it's a bit of a drag to go through 200 lines of declarations and initialize all variables.

(Fm: Mark Harris to LDOS Support) Well actually, I've never used a C compiler that didn't have all variables initialized to zero. But you see, I've always used C on mainframes/minis and the OS of course zeros out all pages when your process initially receives them. (Security y'know) So, while the compiler itself is not initializing the data area, the end result is that everything starts out as zeros. And the fact remains that a lot of programs take advantage of that.

Examples: pcc on Vax, 3B2, Perkin Elmer, NCR Tower. C on DG MVs, IBM 370s etc etc. To be honest, I don't know how the seemingly thousands of C compilers available on the PC clones behave.

(Fm: Les Mikesell) My K&R is always in the wrong place when I want to look things up, but "The C Programmer's Handbook" from AT&T says: "Any permanent variable is initialized to 0 (NULL) unless otherwise specified." I'm pretty sure K&R says roughly the same. ANSI might not guarantee this since there are some machines where the same bit pattern does not represent zero for all data types.

(Fm: Les Mikesell to MISOSYS) Or the correct way, which is to have the compiler do it for you. The AZTEC compiler does it at run-time when you use their linker but generates all the zeros in the /CMD file if you use the M80 format. Seems like it shouldn't be too hard to to generate labels for the start and end of the area and add the code to zero it at start-up. Of course you then have to keep variables initialized to other values separate.

(Fm: MISOSYS) Aztec does it when you use their linker because they support a third segment type (code, uninitialized data, initialized data). I wanted to stay compatible with Microsoft - M80 REL format supports only one CSEG and one DSEG; Rich and I looked at handling initialized data in a COMMON but that just wasn't feasible. My easy solution was to add the "-Z=Y" statement in the linker command line and that fixes it! What's wrong with that solution?

(Fm: Les Mikesell) The "-Z=Y" option is adequate in that it makes the generated code conform to the definition of the language. However, it would seem more intuitive to follow the language definition by default and require the option for the nonstandard optimization. Or, better yet, find some way to initialize at run-time. This might be impossible to link with L80, but if MLINK were able to link all Microsoft /REL files, it wouldn't matter much.

(Fm: MISOSYS) Yes, it would have been impossible with M80, and we had imposed the specification that MC was required to work with M80. That was a given. It then became impossible to collect all static uninitialized vars into one "segment" for run-time initialization purposes. We had considered using a COMMON but that would have been much too dificult. I guess that is where the "GROUP" operator came from in MASM. Thus, since the linking tool we had to use did not support a method, we chose to deal with it as we did. We also recognized that the machine environment was better off without automatically filling zeroes. I can't stand to see those files generated by L80 with sector after sector of wasted garbage or zeroes. At least with MRAS, you have the capability of keeping a smaller OBJ module.

(Fm: Byron P. Peebles) Static and external variables that are not initialized are guaranteed to start off as zero. Automatic and register variables that are not initialized are guaranteed to start off as garbage. (on 3B2 UNIX running AT&T C)

(Fm: LDOS Support jjkd) That certainly is bad coding practice. I don't know of any micro implementation that zeros memory. I can't imagine any implementation of lint not screaming bloody murder about that kind of usage.

K&R says (pp.37): "Automatic variables for which there is no explicit initalizer have undefined (i.e., garbage) values. External and static values are initalized to zero by default, but it is good style to state the initalization anyway."

(Fm: Mark Harris) You're right. Lint will complain if it can positively determine that a variable is not initialized before usage. Still, next time you're on a UNIX machine that has sources, run Lint on a couple of the programs. You will find several programs that do not initialize sources, mix pointer types, etc. wc, for example, doesn't initialize some of the counters.

(Fm: Mark Harris to Byron P. Peebles) I thought I remembered reading it in K&R so I looked it up again today. It' on page 80-something. (If you think about it, you don't want automatic local variables initialized to zero. Too much overhead everytime a function is called.) Anyhow:

```
#include <stdio.h>
static int i;
main(){
   printf("%d\n",i);
}
```

printed out 12699 when run on my machine. Certainly not initialized to zero. Granted, not a major, earthshaking issue, but since I was mildly flamed for asking the original question, I thought I ought to support my case.

Truth be known, I really do not like to program in C. I earn a living supporting an Ada compiler. I find the language well thought out in most cases. I wait for the day when one can obtain an Ada compiler for a PC that (1) doesn't cost thousands (I want a validated compiler), (2) doesn't need massive amounts of memory, (3) doesn't take minutes to compile the simplest of programs and (4) is available for the model IV.

(Fm: mark harris) Another thing. I realize that I've caused a lot of confusion by not saying what I really meant. When I said variables, I didn't mean all variables. I really meant variables declared at the outer lexical level (global). Sorry, it was late.

(Fm: Byron P. Peebles) I found the whole thing enlightening. I am relearning C after several years away for it. It sent me back to the books, since I could vaguely recall some guaranteed zeros, nulls, and some guaranteed trouble. One thing this all did for me was force me to locate the books. No small task around this office.

(Fm: H. Brothers) MSC guarantees that it will zero all unitialized external and static variables.

(Fm: Les Mikesell to LDOS Support) That quote can be taken as an absolute guarantee that globals and static values (equivalent to the "permanent variables" mentioned in the AT&T book) are always supposed to be initialized to zero. K&R's comments about style may be safely ignored. Lint doesn't even object to this assumption. A sample program:

```
/* test for uninitialized global */
#include <stdio.h>
int x;
main()
{
int y;
if (x==0) puts ("X=0");
if (y==0) puts ("Y=0");
return (0);
}
```


The lint output:

```
==============
(8) warning: y may be used before set
==============
function returns value which is
always ignored ... puts
```

Note that there is no complaint about x being used before set. A test run on the 3B2 showed both x and y as zero, though the y value was, of course, accidental.

(Fm: LDOS Support jjkd to Mark Harris) Many examples does not a good programming technique make. As you indicated, this is referring to using automatics, not statics or externals. As Les and yourself pointed out, there is a big difference in the way these two categories are handled.

(Fm: LDOS Support jjkd to H. Brothers) Thank you, as Les indicated I didn't point out the difference between the handling of statics and externs vs. autos. Gosh, you just never know where that nasty ol' stack has been <<grin>>.

(Fm: LDOS Support jjkd to Les Mikesell) Yep, thanks for emphasizing the difference in the handling of statics and externals as compared to automatics.

(Fm: mark harris) Huh, without beating a minor point into the ground, this all started out with the question of how to have the compiler initialize variables to zero since plenty of public domain C code depends on this.

I never claimed that this was a good programming technique. It obviously is not. Furthermore, K&R (I do not know about the upcoming ANSI standard. Although I should. I am an alternate on the committee.) specifies that statics and externals are initialized to zero. PRO-MC does not follow this. No big deal. It merely makes porting C code a somewhat tedious procedure.

Anyhow, thanks for the help and the answers to my question. It's nice to be able to get a quick, intelligent response to a query on such a solid product.

Using the linker option caused the static variables in a small test program to be initialized to zero, so I guess it did the trick.

All this was started by me trying to get a version of fgrep, egrep, or grep up and running on the IV. (*grep is a UNIX style pattern matching program. Useful for finding strings in files.) None of the several versions I've tried seem to work. Don't really have the time as of right now to try and figure out why. *grep is a fairly complicated program.

The problem is not neccessarily PRO-MC's. I've come across a lot of programs that take advantage of compiler feature (like stack architecture) or machine architecture. The hard part is finding where in the code the program screws up. (Hey MISOSYS, wanna do a source level debugger?)

I'm currently putting the finishing touches on a command line editor written with PRO-MC. The program is in the same sort of vein as the one that Hardin Brothers did in a column some months back, but a good bit more extended. Something like SCREENEDIT found on Data General's AOS/VS machines. Maybe I'll upload it when I'm satisfied with the way it looks. It's a nice alternative for people who don't like to use SHELL's menus.

If anybody out there has a version of *grep that compiles and runs under PRO-MC, I sure would appreciate it if you would upload it.

SIDEWAYS with MC

(Fm: Jerry Wagers) Have you been wanting to run SIDEWAYS on your Model III/4/4P? If you've seen this interesting program running on other machines, you had to be impressed. Now a public-domain version of it has just been uploaded to the database, called OFFSID/ACH.

OFFSIDE will allow you to print any ASCII file SIDEWAYS with your EPSON compatible or GEMINI printer, with a line length of up to 255 characters and 80 lines per 'page'. It also permits multiple pages, so virtually any ASCII file can be printed. All ASCII characters are in the character table, and are well-formed and properly spaced. No more 'funny-looking' printouts! Even better, OFFSIDE is written in C and was compiled on Misosys PRO-MC, *and* the source code is included in the archive for you 'hackers'! If you want to run it in model III mode, just recompile it, as is, with MC instead of PRO-MC.

Full instructions are also included in the package; although, the program is extremely simple to use. Additional instructions are included on changing the character pitch of the printout if desired.

PRO-MC and __LINE__

(Fm: Michael Johnston) Roy, I am trying to implement some debugging macro's for my system and I'm having a few problems. First, can I use a macro from within a macro? What I am trying to do is something like this:

```
#define TRON printf("Line:%u\n",_LINE_ -1);
```

and then sprinkle TRON statements in the program. I tried this and the pre-processor expanded the TRON statement but not the _LINE_ macro like it says in the manual. Then I figured that I might have to run through the pre-processor twice to get the full expansion performed. When I do this and get to the compile stage and I get errors on the lines with the _LINE_ thingy in it. I figured that the second pass would expand it to the line# but no-go. I then tried searching all the files on both MC disks with GREP for _LINE_ but I came up empty handed.

(**Fm:** MISOSYS) Page 2-40 states that "macro text replacement may be nested to a depth of 20"; however, I am not sure if the predefined macros (such as '__LINE__') fall into that class. I do note that your example only showed '_LINE_' which is not the predefined macro '__LINE__'. Do you catch the difference? Or did you just drop one of the underlines on each end of "LINE" in your message? Page 2-39 clearly shows TWO underlines required on either side of the "name".

(**Fm: Michael Johnston**) Roy, your right. I was only using _LINE_ instead of __LINE__. Thats a major distinction anyone could miss at a quick glance. I'll try it again and see if it works.

PRO-MC and EOFs

(**Fm: Shane Dawalt**) I wrote a program which uses the FSCANF() function in PRO-MC to obtain an integer value from stdin. I have set O_KBECHO using the OPTION() function so that stdin will echo to stdout, i.e. keyboard input will be displayed. Here's the kicker: when I press <BREAK> at the FSCANF() function, the program locks up in an infinite loop redisplaying the prompt, then an error message over and over again. The routine is designed so that integers less than 1 and greater than 1000 generate an error message after which the routine loops back and redisplays the prompt. I know that an EOF returned by PRO-MC is 0xFFFF. I assume that FSCANF() is returning 0xFFFF as the result. My routine sees it and displays the error message then goes back for more. But, why doesn't the EOF condition remove itself after the next FSCANF()? Also, how can the EOF be changed from <BREAK> to the <SHIFT><whatever><whatever> sequence? I tried the IOCTL() function (with the sgtty header included of course) but I kept getting an "Illegal logical record number." The exact IOCTL() syntax I used was:

```
static struct sgttyb sg = { 0, 0}
/* other statements which have nothing
 to do with ioctl() */
if (ioctl (stdin, tiocgetp, &sg))
{   perror ("IOCTL() error");
    exit(-1);
}
/*  and the program continues ... */
```

Am I missing a simple point in the above code ... or am I completely wrong?

(**Fm: Les Mikesell**) You have to use an explicit clearerr(stream) to reset the EOF condition. A couple of points:

(1) If there is any chance the program will be run with redirected input (isn't there always?) you should be sure to test that the program is being run from a terminal (isatty(0)) before clearing EOF. Otherwise, the proper response to EOF is to exit gracefully.

(2) Unless you are parsing computer generated output (predictable), it is better to avoid using fscanf() directly for input. Instead, use fgets() into a buffer, then sscanf() to parse the buffer. It is fairly difficult to recover from an error in fscanf(), and its concept of "newlines are whitespace" seldom matches reality.

(**Fm: MISOSYS**) I assume that you really specified "TIOCGETP" and not "tiocgetp"; there is a difference, you know!

MC and C variables

(**Fm: Shane Dawalt**) I'm puzzled on the operation of 'C' when variables are passed to a function. Specifically, what do I do after they are passed? Consider the following hypothetical function:

```
void myfunc(buf_ptr,count,length)
int count, length;
char *buf_ptr;
{
   ... the function coding goes here
   return;
}
```

Can I use the variables defined outside of the "{}" of myfunc() or must I define a new set of variables inside the "{}" and set them equal to the variables which contained the passed information? I ask this because, at times, PRO-MC acts as if it cannot access those variables passed to the function unless they are assigned to variables defined within the function. This happens most of the time in a situation such as (using the variable definitions of myfunc()):

```
if (*(buf_ptr+count) == '\0')
    {
    ....do something
    }
continue with function...
```

In this case, the 'C' function will not go to the "....do something", but will continue on past the end of the if statement. Yeah, I know, have you checked your logic and punctuation ... YES, YES, YES. Nothing is wrong except this small idea that 'C' doesn't like to use it's passing variables as working variables.

(Fm: Les Mikesell) The variables passed to a function act exactly like conveniently initialized local variables. Your function looks reasonable, but of course I can't tell if the: "if (*(buf_ptr+count) == '\0')" statement should be true or not without knowing what is passed to the function. I usually add a printf() to display the variables for debugging when something doesn't work the way I expect. It is pretty easy to have an "off by one" error working with Ø based arrays.

(Fm: Shane Dawalt) Actually, that's what I wanted to hear. I didn't like the idea of creating variables inside a function for variables passed to the function. Now ... what's wrong with that dumb thing?

(Fm: jeff brenton) Well, the function looks ok, but.... did your calling function know that buf_ptr was a pointer, and not an int? Is it possible you passed it as &buf_ptr, which would send a pointer to the pointer to your buffer, rather than the pointer itself?

As for the variables themselves, you CAN use them within the function, without copying them around. Those values, plus the ones defined as local to the function you are writing, are available to the function block.

Last thing - are you SURE there is a \0 in the buffer?

(Fm: Shane Dawalt) In writing that message, I was assuming that the calling function was correctly written and assumed the proper things. As for my program, everything is fine. Your idea about \0 has already been thought of and I have placed an initialization statement at startup to take care of this. Actually, it was declared extern and public to all modules/functions which should have cleared the space to \0 automatically. Or am I wrong to assume this?

(Fm: LDOS Support jjkd) The variable should be declared as a static in exactly one module, and as an external in all others. You may initalize it when declaring it as static if desired. If it were external in all modules, there would never be any storage allocated to it.

(Fm: Shane Dawalt) Perhaps I'm not using the correct "language" to explain how I have declared the variable. The main module has the variable declared BEFORE the main() function. The variable is not preceeded by extern. It is declared with it's data type, then the variable name. I have the impression, right or wrong I don't know, that a variable declared at the very BEGINNING of the program is automatically assumed static. Of course, I could have my wires crossed ... again.

(Fm: MISOSYS) See page 2-14 of the MC users manual (through 2-17). This defines "scope of variables" and "storage classes". True that variables defined outside of any function are static [main() is a function].

(Fm: Shane Dawalt) Well, what I call external is what should be called static. Like I said, the language I use is not what should be used. [I'm having a rather hard time in programming in "C". The language barier I'm building around me isn't helping ... as you can see.]

Perhaps my explanation above wasn't a good one. I call variables external when they are declared outside of a function. I don't "see" variables as static unless they have an absolute declaration of STATIC. This is wrong. I'll have to take Roy's advice and try to reread the pages he outlined.

(Fm: Les Mikesell) No,no... Static means something else in 'C'. A variable declared as static has permanent storage, but limited scope. That is, it is only known within the block where it is declared. A static variable declared within a function will retain its value between calls to the function, but cannot be accessed outside of that function and will not conflict with another variable of the same name inside another function. A static variable declared outside of any function may be accessed by any function within the same file but would not conflict with another variable of the same name declared as static in another file which might be linked into the same program. Global variables (defined outside of functions without being declared static) are accessable by functions in all files, but they must be declared extern in all files where they are used, except the one where they are defined. **Most** compilers will resolve multiple definitions of the same variable into a single global, but don't count on it.

(Fm: LDOS Support jjkd) Yes, "external" refers to "defined, allocated storage and possibly initalized **external** to **this** source code file". The tag "extern" is not optional.

"Static" is "defined, allocated storage and possibly initalized **in** this source code file, but not within any function". The tag "static" is optional.

(Fm: Les Mikesell) Static varibles may also be declared within functions (or inside any block) to provide permanent variables that are only accessable within the block.

MC: Structures and bit fields

(Fm: Nate Salsbury) I've been reading the referenced file in order to learn more about structures and bit fields. Two questions:

(1) It seems to me that the part labelled 'DCT byte 4 flags' is really going to be reading what the Tech Manual calls 'DCT+3' and the part labelled 'DCT byte 3 flags' will really be overlaying what the Tech Manual calls 'DCT+4'.

It also seems to me that the two sections 'DCT byte 8' and 'DCT byte 7' will actually overlay 'DCT+7' and 'DCT+8'respectively Is that an error in the DCT/CCC file or in my understanding of what a structure is/does?

(2) In main() there is a line of code: "dct = (struct dct *) regs.IY". Do I read that to say that "the value of regs.IY is being cast to be a pointer to a structure of the type dct" and then the result is assigned to dct which was created as such a pointer earlier?

(Fm MISOSYS) Careful reading of the README file supplied with MC should clear up the confusion. Let me cite the applicable text. "All bit fields are treated as unsigned short integers; there are no signed bit fields. Bit fields are ordered from high (most significant bit) to low (least significant bit) in a machine word (16 bits). The ordering is such that the first declared bit field will start in bit 15 of a double Z80 register (e.g. bit 15 of HL, or bit 7 of H), and so on. Note that when a word containing bit fields is fetched from or stored into memory, the high and low bytes are reversed! This reversal must be taken into account when creating a set of bit fields to describe an externally imposed format (e.g. a bit structure maintained by the operating system). The reversal is of no importance if the structure is wholly contained within the C program, and not used outside of it."

Now let's take a look at that "dct" structure as used in DCT/CCC (abbreviated).


```
struct dct
{  uchar dct_instr;
   char *dct_driver_addr;
   /* DCT byte 4 flags: */
/* bit field declaring 8-bits */
   /* DCT byte 3 flags: */
/* bit field declaring 8-bits */
   uchar dct_curr;
   uchar dct_high;
   /* DCT byte 8 allocation data: */
   unsigned dct_gpc: 3;
   unsigned dct_spg: 5;
   /* DCT byte 7 allocation data: */
   unsigned dct_head: 3;
   unsigned dct_spc: 5;
   uchar dct_dir;
} *dct;
```

The "uchar dct_instr;" declaration references a single byte which happens to be the first byte of the DCT (DCT+0). The "char *dct_driver_addr;" declaration references the driver vector address which is DCT+1 and DCT+2; note that this is declared as a pointer-to-char which takes up two bytes.

Now the fun begins. You are confused over why the structure says that the flags at DCT+4 are declared next followed by a declaration of the flags at DCT+3. The cited reference stated that bit fields are unsigned short integers (16 bits) and that the first declared bit field will be starting in bit 15 of a double Z80 register, say HL. Now that's the high-order bit of register H. Since Z80 "words" are stored in low-high order, it means that the first bit field declared will wind up in the high order storage location when placed into or read from memory. Thus, the first bit field declared must be referencing the fifth byte of the DCT which is DCT+4; similarly, the ninth bit being fielded starts referencing the low-order register and thusly references DCT+3 (the fourth dct byte). A lot of the confusion comes from not recognizing that you are really declaring up to 16 bits of a "machine word" once you start a bit field - whether you use the entire 16-bits or not. The DCT/CCC example program separates the 16-bits into two 8-bit byte bit fields of declarations to follow along with the documentation for the DCT itself. The second bit field you asked about follows the same reasoning.

It's important to understand the thrust of the the final words cited from the README file. Not all processors reverse a short integer in memory. Ever wonder what swab() was designed for? The storage of double words is even more of consternation; some CPU's reverse the high and low word and then reverse again the individual storage of each word (80x86, for instance). Some just reverse the high and low words while others don't reverse a double word at all. Then there's long words...

You are right on about your question concerning "dct = (struct dct *) regs.IY" Let's take a look at what it is doing. "regs.IY" is declared in the Z80REGS.H header file as an "unsigned short integer". MC is rather tight on permitting pointer operations only on variables declared as pointers. The "CALL(GTDCT,&regs);" statement issues an @GTDCT service call which loads register IY with the DCT address of the drive being referenced. Within C, this "address" is conceptually used as a "pointer to a structure of type dct"; thus, that's the reason for the cast.

MC and #include variations

(Fm: Byron P. Peebles) I think the documentation should add a brief description of the differences between: '#include <file.h>' and '#include "file.h"' when used in the UNIX environment.

As I'm sure you're aware, K&R indicate the quoted include searches first in the directory of the source file before searching the sequence of "standard places", while the bracketed include does not search the directory of the source file. (Standard places usually meaning "/usr/include"). While this means nothing under LS-DOS implementations, I could impact the systems developed under LS-DOS/MC once migrated to UNIX, or other environments.

(Fm: MISOSYS) Since MC is not used in a UNIX environment, why should it describe the differences between <stdio.h> and other forms?

(Fm: Byron P. Peebles) It's my humble <rightfully so> opinion that since you suggest that programs developed under LS-DOS with MC will be compilable under UNIX, you should point out areas where the handling of things (like forms of include) differs. I would have kept quiet on the entire issue, except your documentation, which does appear to be excellent, uses several forms of include for the same declaration on the same page. Just seems it might cause confusion, that's all.

(Fm: MISOSYS) That's to show that in MC, all of those forms are equivalent. That's for the folks trying to port stuff from other systems.

MC and bug reports/fixes

(Fm: Roger A. Sabin) While using the PRO-MC compiler, I came across a problem. Some of the float routines do not work properly. Specifically, the following routines do not work: fraise(), fexp(), and fsqr(). Also, the sinh() function returns -HUGE when the argument is zero.

(Fm: MISOSYS) We had a number of little problems pending with MC that we have corrected. I located the root of the problem you were having with fexp() and fsqr(). The problem also carried over to fraise(). Of course, fsqr() uses fraise() which uses fexp()! The latter was where the problem lay. Apparently one line of data from the table of constants associated with the polynomial expansion was corrupted. Fixing fexp() cleared up the problem with all three.

I also located the bug in sinh(0.0). The problem was actually with the _ddv2() function; it didn't test for a zero value before doing its thing. That function divides by two a float or double by decrimenting the exponent byte (which is kept in excess 128 notation). Since a zero exponent is used to designate a zero value float, an arbitrary decrement from 00H to 0FFH would arbitrarily change the float from a value of zero to some fraction times two to the minus 127th. That's why you got a large negative result from sinh(0.0) [calculated as _ddv2(exp(x)-exp(-x))].

Note to our readers: corrected versions of the appropriate library modules were put on DISK NOTES 7 as suggested in TMQ I.iii.

(**Fm: Jeffrey McLean**) I have found more bugs in MC. External and static variables are not initialized to zero. K&R states on page 198: "Static and external variables which are not initialized are guaranteed to start off as Ø." Scanf and fscanf(stdin,...) do not echo characters that are typed in to the screen. Scanf() and fscanf() do not handle non white space, non conversion characters correctly. The program TEST/CCC demonstrates this problem. The FIXBUFS option does not work. One last bug occurs in expressions where UNSIGNED LONG values are converted to DOUBLE (see enclosed test program).

(**Fm: MISOSYS**) Here are the solutions. As far as guaranteeing non-initialized statics and external variables to be zero, that essentially requires them to be initialized to zero. The Aztec assembler does this by collecting all uninitialized statics, et al, into a third segment type supported by their assembler (code, data, uninitialized data). Their runtime support can then zap that region with zeroes. By doing so, they don't take up disk file space in the object program for the zeroed region. We decided to support Microsoft's REL bit stream which restricted us from implementing such a method. The alternative was to either ALWAYS initialize such data to Ø explicitly or not initialize (i.e. leave initialization subject to whatever was in memory by using DEFS for declaration).

We decided that to force the compiler to generate zeroes would not necessarily be best for the target environment; a Z80 with 180K drives. That's why we went to such great extent in our linker to provide an ability to keep DS regions declared in data segments! On the other hand, we provided a linker switch, "-Z=Y", to allow the programmer control over the generation of zeroed space. You can edit the MC/JCL file to add that switch to the MLINK command line. You can also make a change in the MCMACS file. The macro, $VAR, is used to declare space for statics (local or global). The space is defined by the statement;

```
DS      #SIZE
```

If you want to have those regions explicitly initialized to zero, change the statement to;

```
DC      #SIZE,0
```

You could even get fancier by adding conditional code centered around a #option. We may discuss that in TMQ I.iv.

That's the best we can do with uninitialized data. The echoing of characters during stream input from the keyboard is governed by the KBECHO option. There was a bug in fopen which effected KBECHO as reported in TMQ I.iii. If you add the statement "option(O_KBECHO,1);" to your code, you will get echo. That fixes up your second problem.

We have revised _scan() (which is used by the higher-level scan functions) to agree with the System V handbook on termination conditions. It was a major revision; however, in so doing, we have cleared up a number of bugs in that function. We even added code in scan to properly terminate float fields.

The FIXBUFS option did not work correctly. Corrections had to be made to fopen() as reported in TMQ I.iii.

I tracked your "unsigned long to double" problem down to an obscure bug in the @UL2DST routine. This routine in LIBA converts an unsigned long number on the stack to a double on the stack. Fifty percent of the time it would work correctly. That's because the error would occur if the return address of the @UL2DST caller was an even number!

(**Fm: Richard Deglin**) I came across a bug which was interested when I fixed some other problem associated with structure initialization. Here are patches which correct a problem encountered when initializing nested structures. They are in The Patch Corner as INSTRU??/FIX.

I also noted the discussion in TMQ I.iii which concerned itself with the initialization of automatic variables. This is something I overlooked when implementing MC. Here are some patches which extend the syntax of automatic scalar variable initialization to include any arbitrary expression as the initializer. In doing this, however, the code generated will often be less efficient than the old way; nevertheless, the patch should stand because it brings MC into conformance with standard C. They are in The Patch Corner as AUTOIN??/FIX.

MRAS assembler

(Fm: Michael R. Johnston) I received the fall TMQ and it's great! I can't say enough about it. I sure hope that subscriptions pick up - I'd hate to lose this one! Your comments about spending more time with the family really hit a soft spot here. All too often in the height of writing this program or debugging that program, we forget about the ones we love. If more programmers would remember this, maybe there wouldn't be so many of us divorced. Conclusion: Keep the copmments about your family in TMQ; it makes it much more of a 'friendly' publication.

I would like to congratulate you on PRO-MRAS! I never thought that writing in assembly could be so much fun. I used to use ALDS but I got tired of it's limitations, i.e. DS's of 255 bytes or less per line among other things.

I have a quick BUG REPORT for you on PRO-MRAS version 1.0a. The following statements do NOT report an error.

```
ETX     EQU     03
UNDRLN  DC      80,'_',ETX
```

The code generated will be 80 underlines followed by a zero byte even though ETX is defined as 3. Using this with DSPLY really had me going. I couldn't figure out why I was getting garbage after it printed the string.

(Fm: MISOSYS) It is probably wrong of MRAS not to complain about the syntax of your statement:

```
UNDRLN  DC      80,'_',ETX
```

since the statement does not adhere to the correct syntax [the DC pseudo-op does not permit repetition]. You would need to recode the statement as a pair of statements:

```
UNDRLN  DC      80,'_'
        DB      ETX
```

I don't feel it is necessary to work up a patch for this. It will be corrected when MRAS gets reassembled to another release.

(Fm: John M. Pantone) I have been using your MRAS assembler now for several months, and I am extremely pleased. MRAS is, without a doubt, one of the finest assemblers, micro or mainframe based, I have ever used.

I have, however, run into a rather obscure bug which I would like to bring to your attention. It relates to the DW pseudo-op; specifically when multiple arguments are provided, and where the first arguiment is the address of a label.

If the first of many arguments to DW is a constant, or if DW is used with single arguments only, the bug doesn't appear. I have enclosed four figures which illustrate the problem. I have tried to isolate the bug to as small an example as possible, while still illustrating the situation.

In the process of preparing this letter, I ran into 2 other, less disturbing, bugs: (1) The -CI flag of MRAS appears to do nothing - the REL file produced (not a CIM) file as documented) is identical to one produced without the -CI flag. (2) The -I=y flag of MLINK, while it does, in fact produce a core-image file, does not use the CIM extension (it continues to use the CMD extension) as documented.

I hope you are able to track down this DW bug - it is rather nasty, since the listing appears just fine - it is only the CMD file generated with MLINK which shows the error; not too many people are likely to look there for such a thing. The -CI and -I=y bugs are, perhaps, really just documentation changes.

(Fm: MISOSYS) I greatly appreciate the detail of your bug report and the extent to which you have narrowed down the problem. It certainly made it easy for me to find and rectify the problem. Actually, the bug was more obscure than you even thought. It, of course, was isolated to specifically the case of multiple operand DW's. Next, you had to have a relative value (in any operand position) immediately followed by an absolute value which was negative; particularly one in which the negativity was indicated by the presence of a

minus sign (or another relative value of different mode).

For instance, if, in your case, you coded the first line rather than the second :

```
DW      DUMLAB,0FFFFH,0
DW      DUMLAB,-1,0
```

the correct values would be generated. The bug was that the mode of reference was not being initialized to absolute for the 2nd through nth operands of the DW statement since that was being done only preceding the parsing of each statement. The unary minus sign is a special case in expression analysis. Needless to say, MRS612/FIX (Model 4) and MRS512/FIX (Model III) are a short patch which will correct the bug.

As for your first note, it is true that the "-CI" flag of MRAS must be entered in addition to the -GC switch. We do feel that the workaround precludes the necessity of attempting a program patch.

Point two is well taken. Although MLINK does indeed accept the "-I=y" flag and generates a core image file, the filespec generation doesn't switch to "/CIM" from the "/CMD" default. Again, I don't feel it necesssary to bother with a patch. MLINK has been patched enough. We do have a plan for a "possible" re-assembly later on this year if time permits. It should be correctable then. Again, thanks for making our job easier.

(**Fm: Dirk Vandenbossche**) I've been using your software packets, PRO-WAM, PRO-MRAS, and PRO-MC for some time now and they all seem to work well. Yet I've met some problems.

In the 80-MICRO issue of May, an assembly language program 'PRSET/APP' that works with your PRO-WAM, is explained. As I don't know much about assembly language, I've typed in and compiled the program with MRAS. As you can see in the listing added, the assembler points out several error messages. As I considered the program to be correct, I assumed something had to be wrong with MRAS. So I tried out the TYPER program, you enclosed in your PRO-WAM packet. When compiling, I met the same error messages. Do you see a possible explanation/solution for this?

(**Fm: MISOSYS**) There's an easy explanation for the problems you were having with assembling PRSET/ASM using MRAS. Your solution is to directly generate the /APP core image file by a command such as:

```
MRAS PRSET +O=PRSET/APP -GC-CI
```

That's because trying to generate a REL file imposes certain constraints on the operand fields. The errors you experienced can be explained by simply reading page 2-15 of the MRAS manual. A statement such as

```
DC .HIGH.$<8-$+256,0
```

has at least three operations not permitted in the expression when assembling a relocatable object module. It is permitted when assembling an executable command file. I believe that Hardin Brothers used our PRO-CREATE assembler: MRAS would behave similarly when the "-GC" switch is used.

PaDS - Partitioned Data Set Utility

(**Fm: Irwin B. Burton**) I have tried to use PaDS with LDOS 5.3. I found that by eliminating the PASSWORD protection I can make up a PaDS of my utilities. However, I have trouble adding any 5.3 utilities to this file. As an example, HITAPE from 5.1.4 loads in and runs fine under 5.3. I'm sure this has to do with the PASSWORD - DATE/TIME changes in the directory but I'm not enough of a hacker to figure out just what is happening. I would appreciate any help you can give.

(**Fm: MISOSYS**) You guessed wrong. The reason that HITAPE (or other utility) from LDOS 5.3 shows up as a DATA member of PaDS has nothing to do with passwords (or the lack thereof in 5.3). What it has to do with is a test that PaDS makes to ensure that the appended member is a CMD file. It rejects as being data, any file which does not start with either a 01H (load record) or a 05H (comment).

In LDOS 5.3, the header comment record was suppressed to save disk space (just in case a file went to a granule boundary without the header); this makes 5.3's files start with a 1F record. Thus, PaDS assumes those are not programs.

Here's a little fix you can make to PaDS which will make it more lenient (it will accept any char less than 20H as valid.

```
PATCH PDS.PDS (D20,75=20 30:F20,75=05 20)
```

While you're about it, add the PDS553/FIX patch to the PDS(BUILD) module. That revises PDS(BUILD) to the password convention of 5.3. A similar patch for PRO-PaDS was in TMQ I.iii. That should fix you up. A similar patch for making PRO-PaDS more lenient is:

```
PATCH PDS.PDS (D20,55=20 30:F20,55=05 20)
```

PRO-WAM Window Application Manager

(Fm: Tom Gallaudet) The Sort key in the first record of ADDRESS/DAT only affects the PSORT sorting. Changing it has absolutely no effect on the "Search for ?>" key. I'm looking to search by Company name, the third field, 20 characters long.

(Fm: MISOSYS) It seems like ADDRESS is hard coded to use the last&first name fields for searching; the header key field(s) are only for PSORT. I changed this in PRO-WAM Release 2 so that ADDRESS can pick up the information on the first key and use it. If you go for the upgrade, you'll have what you were asking for (and more).

(Fm: Alan H. Pesetsky) Been fiddling with a copy of Pro-NTO (now WAM) that I bought from a RS store some time ago (#00549). I seem to have trouble using Shift f1 & Shift f2 & shift f3 when I have KSM Plus installed - f1 & f2 & f3 work however. KSM Plus allows the use of both sets and I do. Is there a way around this conflict, if it is a conflict. Also are there patches or updates I should have?

(Fm: LDOS Support jjkd) If you have KSMPlus installed with definitions for the function keys, they can't be used to control Pro-WAM. You must "undefine" those keys before Pro-WAM can see them.

(Fm: Bob Haynes) Here's a mini-tip for anyone using Roy's PRONTO/PROWAM program... Would you prefer not to use a multi-key (control-P, or whatever) command to activate your PROWAM system? Just found out that you can specify an Fx key as the activation character, even though the docs seem to imply you cannot. Works just fine, apparently since PROWAM does not take over those keys under AFTER it activates. When doing assembly programming, a quick double tap on F1 takes me directly to my SVC/APP screen; once for PRONTO, once for the /APP. I love it!

(Fm: MISOSYS) The only thing you would lose by using a function key as the PRO-WAM activation "hot key" is the ability to get the PRO-WAM menu displayed once you got into an application. Some folks forget what they have assigned to a key. If you are already in one application, pressing the hot key still brings up the menu - unless the hot key happens to be a function key - then you would invoke the application assigned to that function key.

(Fm: Dirk Vandenbossche) There is a problem concerning the file WINDOW/CCC also included in the PRO-WAM packet. When compiling this program with MC, I came across error messages. In the manual, you describe on page 2-12 the pointer operations that can be used. The operations <<, >>, &, and | seem not possible. Can you please indicate how you did compile this program?

(Fm: MISOSYS) Your problem with the WINDOW/CCC functions is because they were written for our LC compiler; long before MC was released. MC, being a full compiler, is tighter about your use of identifier types. MC flags an error when you try to use a pointer in a non-pointer operation - even though on the particular implementation, an integer and a pointer are of the same bit length. It just makes good sense for portability to insist that pointer operations use pointers and normal arithmetic operations using pointers (except those permitted for pointers) be disallowed. I have attached a sheet showing the two functions which need "updating" by the addition of a cast to unsigned int. Edit your WINDOW/CCC file to include these casts and you will have no further problems. These are the revised lines 11, 12, 15, and 28 of WINDOW/CCC:

```
*(++buf+((unsigned int)regs[BC]>>8))='\0';
if (((unsigned int)regs[AF] & 1) == 0)
return(((unsigned int)regs[BC]&0xff)+1);
return((unsigned int)regs[AF] >> 8);
```

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 and LDOS 5.3 patch formats use 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 (it does support a semicolon which is used under LDOS to signify a trailing comment); LDOS 5.3 supports a colon separator both in a command line patch and a fix-file patch. 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 for TRSDOS 6.

If you use the FIXES?/TXT file from the DISK DNOTES corresponding to this issue, please separate out the individual fixes which you need by use of any text editor you find convenient to use.


```
-----
. ALTBANK/FIX by Paul M. Bradshaw  03/22/87
. Patch to cause ALTRES to use bank number 4.  If you wish to use a bank other
. than bank 4, replace the 4's with whatever bank number (in hex, up to 0F) you
. wish to use.  Note in the 7th line, that the number is '84' and NOT '04'.
D01,1A=04:F01,1A=02:D01,9E=04:F01,9E=02:D02,1F=04:F02,1F=02:D02,28=04:F02,28=02
D03,83=04:F03,83=02:D03,F2=04:F03,F2=02:D07,EC=84:F07,EC=82:D08,32=04:F08,32=02
D08,50=04:F08,50=02
-----
```

```
. AUTOIN51/FIX - patch MC51/CMD - RND - 05/19/87
. allow arbitrary expression as initializer of an automatic scalar variable
D1E,EB=04 00 B9 75 03 00 B9 75 05 00 B9 75 01 00 B9 75
F1E,EB=04 00 BB 75 03 00 BB 75 05 00 BB 75 01 00 BB 75
D23,B8=21 06 00 39 7E 23 B6 28:F23,B8=CD B9 AD 22 67 70 21 01
D23,C0=13 21 D7 54 E5 E5 E5 CD AC 7E F1 CD 4C 95 F1 CD
F23,C0=00 22 79 B4 2A 77 B4 22 69 70 CD 65 AC 22 59 70
D23,D0=36 C3 F1 C9 CD B9 AD 22 67 70 3E 01 32 79 B4 2A
F23,D0=21 00 00 22 79 B4 2A 59 70 7C B5 CA 2D 76 2A 59
D23,E0=77 B4 22 69 70 CD 65 AC 22 59 70 AF 32 79 B4 7C
F23,E0=70 E5 CD AB BF F1 22 65 70 7C B5 20 0B 2A 59 70
D23,F0=B5 CA 2D 76 E5 CD AB BF D1 7C B5 20 0A D5 CD 9B
F23,F0=E5 CD 9B BF F1 22 65 70 7C B5 CA 1B 76 2A 65 70
D24,00=BF F1 7C B5 CA 1B 76 22 65 70 11 08 00 19 7E FE
F24,00=11 08 00 19 6E 26 00 E5 CD 09 70 F1 7C B5 28 11
D24,10=09 20 11 2A 59 70 E5 CD 39 D5 F1 2A 69 70 22 77
F24,10=2A 59 70 E5 CD 39 D5 F1 2A 69 70 22 77 B4 C3 2D
D24,20=B4 C3 2D 76 21 06 00 39 7E 23 B6 20 06 CD 50 76
F24,20=76 21 06 00 39 CD AC E4 CD A4 E4 7C B5 28 05 CD
D24,30=C3 19 76 2A 65 70 11 09 00 19 6E 62 E5 CD 2B 6F
F24,30=50 76 18 75 2A 65 70 11 09 00 19 6E 26 00 E5 CD
D24,40=F1 E5 C1 21 6B 70 CD EE E4 2A 65 70 11 04 00 19
F24,40=2B 6F F1 E5 C1 21 6B 70 CD EE E4 2A 67 70 E5 2A
D24,50=CD 8B E3 C5 D5 CD A1 6E F1 F1 CD A4 E4 ED 5B 67
F24,50=65 70 11 04 00 19 CD 8B E3 C5 D5 CD A1 6E F1 F1
D24,60=70 CD 7E E4 7C B5 C4 56 BE 00 00 00:F24,60=CD A4 E4 D1 CD 7E E4 7C B5 C4 56 BE
------
. AUTOIN52/FIX - patch MC52/CMD -  RND - 05/19/87
. allow arbitrary expression as initializer of an automatic scalar variable
D19,B3=08 72 05 00 05 00 00 72 04 00 EE 71 03 00 EE 71 02 00 E9 71 01 00 E9 71
F19,B3=38 72 05 00 05 00 30 72 04 00 17 72 03 00 17 72 02 00 0D 72 01 00 0D 72
D1F,57=CD 09 72:F1F,57=CD D4 AB
D1F,D9=18 09 F1 CD 6E CE F1 C3 BE 71 00:F1F,D9=F1 C1 E1 E5 C5 F5 7C B5 C4 35 BF
D20,3C=18 09 CD 6E CE F1 C9 00 00 00 00:F20,3C=F1 C1 E1 E5 C5 F5 7C B5 C4 40 C5
D20,4F=C9 21 0F 6B E5 CD A0 9C D1 F1 E1 E5 F5 E5 D5 CD C8
F20,4F=C9 21 0F 6B E5 CD A0 9C F1 F1 E1 E5 F5 E5 21 0F 6B
D20,60=A6 F1 C1 21 4B 6B CD 36 E6 2A 0F 6B 18 0C 2A 0F
F20,60=E5 CD C8 A6 F1 F1 F1 C1 E1 E5 C5 F5 7C B5 28 11
D20,70=6B CD FA 71 CD 48 C5 2A 11 6B E5 CD 6D C5 F1 C9
F20,70=21 0F 6B E5 F1 C1 E1 E5 C5 F5 E5 CD B6 97 F1 F1
D20,80=01 02 00 72 21 0F 6B E5 CD 90 9A F1 C9 21 06 00
F20,80=01 02 00 72 C9 F1 E1 E5 F5 E5 C1 21 4B 6B CD 36
D20,90=39 7E 23 B6 CA D4 AB F1 21 D0 54 E5 E5 E5 CD FE
F20,90=E6 2A 0F 6B E5 CD 6D C5 F1 18 21 2A 0F 6B E5 CD
D20,A0=7E F1 CD 61 94 F1 21 DD 54 E5 DD E5 E5 CD 1D 95
F20,A0=6D C5 F1 CD 48 C5 21 0F 6B 23 23 CD F4 E5 E5 CD
D20,B0=F1 F1 21 D0 54 E5 CD 22 90 C3 5B 71:F20,B0=6D C5 F1 C9 21 0F 6B E5 CD 90 9A F1
------
. AUTOIN60/FIX - patch MC60/CMD -  RND - 05/19/87
. allow arbitrary expression as initializer of an automatic scalar variable
D20,47=59 4D 05 00 05 00 51 4D 04 00 3F 4D 03 00 3F 4D 02 00 3A 4D 01 00 3A 4D
F20,47=8F 4D 05 00 05 00 87 4D 04 00 6D 4D 03 00 6D 4D 02 00 63 4D 01 00 63 4D
D26,70=CD 5A 4D:F26,70=CD 31 92
D27,29=18 09 F1 CD 2C BA F1 C3 F2 4C 00:F27,29=F1 C1 E1 E5 C5 F5 7C B5 C4 90 A7
D27,8C=18 09 CD 2C BA F1 C9 00 00 00 00:F27,8C=F1 C1 E1 E5 C5 F5 7C B5 C4 24 AE
D27,B7=C9 00 2A 97 45 7C B5 C2 BA:F27,B7=18 76 2A 97 45 7C B5 28 05
D27,C0=4D 21 8B 45 E5 CD F4 7F D1 F1 E1 E5 F5 E5 D5 CD
F27,C0=CD BA 4D 18 6A 21 8B 45 E5 CD F4 7F F1 F1 E1 E5
D27,D0=C4 8A F1 C1 21 C7 45 CD 23 DC 2A 8B 45 18 0C 2A
F27,D0=F5 E5 21 8B 45 E5 CD C4 8A F1 F1 F1 C1 E1 E5 C5
D27,E0=8B 45 CD 4B 4D CD 36 AE 2A 8D 45 E5 CD 89 AE F1
```

```
F27,E0=F5 7C B5 28 12 21 8B 45 E5 F1 C1 E1 E5 C5 F5 E5
D27,F0=C9 21 8B 45 E5 CD C0 7D F1 C9 21 06 00 39 7E 23
F27,F0=CD D7 7A F1 F1 18 38 F1 E1 E5 F5 E5 C1 21 C7 45
D28,00=B6 CA 31 92 F1 21 09 29 E5 E5 E5 CD 15 5C F1 CD
F28,00=CD 23 DC 2A 8B 45 E5 CD 89 AE F1 18 22 2A 8B 45
D28,10=36 77 F1 21 16 29 E5 DD E5 E5 CD 03 78 F1 F1 21
F28,10=E5 CD 89 AE F1 CD 36 AE 21 8B 45 23 23 CD E1 DB
D28,20=09 29 E5 CD AA 71 C3 8F 4C 00 00 00 00 00 00
F28,20=E5 CD 89 AE F1 18 08 21 8B 45 E5 CD C0 7D F1
-----
. AUTOIN61/FIX - patch MC61/CMD - . RND - 05/19/87
. allow arbitrary expression as initializer of an automatic scalar variable
D1E,DA=04 00 A8 49 03 00 A8 49 05 00 A8 49 01 00 A8 49
F1E,DA=04 00 AA 49 03 00 AA 49 05 00 AA 49 01 00 AA 49
D23,A7=21 06 00 39 7E 23 B6 28 13:F23,A7=CD A8 81 22 56 44 21 01 00
D23,B0=21 C8 28 E5 E5 E5 CD 9B 52 F1 CD 3B 69 F1 CD 22
F23,B0=22 68 88 2A 66 88 22 58 44 CD 54 80 22 48 44 21
D23,C0=97 F1 C9 CD A8 81 22 56 44 3E 01 32 68 88 2A 66
F23,C0=00 00 22 68 88 2A 48 44 7C B5 CA 1C 4A 2A 48 44
D23,D0=88 22 58 44 CD 54 80 22 48 44 AF 32 68 88 7C B5
F23,D0=E5 CD 97 93 F1 22 54 44 7C B5 20 0B 2A 48 44 E5
D23,E0=CA 1C 4A E5 CD 97 93 D1 7C B5 20 0A D5 CD 87 93
F23,E0=CD 87 93 F1 22 54 44 7C B5 CA 0A 4A 2A 54 44 11
D23,F0=F1 7C B5 CA 0A 4A 22 54 44 11 08 00 19 7E FE 09
F23,F0=08 00 19 6E 26 00 E5 CD F8 43 F1 7C B5 28 11 2A
D24,00=20 11 2A 48 44 E5 CD 08 A9 F1 2A 58 44 22 66 88
F24,00=48 44 E5 CD 08 A9 F1 2A 58 44 22 66 88 C3 1C 4A
D24,10=C3 1C 4A 21 06 00 39 7E 23 B6 20 06 CD 3F 4A C3
F24,10=21 06 00 39 CD 5C B8 CD 54 B8 7C B5 28 05 CD 3F
D24,20=08 4A 2A 54 44 11 09 00 19 6E 62 E5 CD 1A 43 F1
F24,20=4A 18 75 2A 54 44 11 09 00 19 6E 26 00 E5 CD 1A
D24,30=E5 C1 21 5A 44 CD 9E B8 2A 54 44 11 04 00 19 CD
F24,30=43 F1 E5 C1 21 5A 44 CD 9E B8 2A 56 44 E5 2A 54
D24,40=3B B7 C5 D5 CD 90 42 F1 F1 CD 54 B8 ED 5B 56 44
F24,40=44 11 04 00 19 CD 3B B7 C5 D5 CD 90 42 F1 F1 CD
D24,50=CD 2E B8 7C B5 C4 42 92 00 00 00:F24,50=54 B8 D1 CD 2E B8 7C B5 C4 42 92
-----
. AUTOIN62/FIX - patch MC62/CMD - RND - 05/19/87
. allow arbitrary expression as initializer of an automatic scalar variable
D19,CE=23 46 05 00 05 00 1B 46 04 00 09 46 03 00 09 46 02 00 04 46 01 00 04 46
F19,CE=53 46 05 00 05 00 4B 46 04 00 32 46 03 00 32 46 02 00 28 46 01 00 28 46
D1F,72=CD 24 46:F1F,72=CD EF 7F
D1F,F4=18 09 F1 CD 86 A2 F1 C3 D9 45 00:F1F,F4=F1 C1 E1 E5 C5 F5 7C B5 C4 50 93
D20,57=18 09 CD 86 A2 F1 C9 00 00 00 00:F20,57=F1 C1 E1 E5 C5 F5 7C B5 C4 5B 99
D20,6A=C9 21 2A 3F E5 CD:F20,6A=C9 21 2A 3F E5 CD
D20,70=BB 70 D1 F1 E1 E5 F5 E5 D5 CD E3 7A F1 C1 21 66
F20,70=BB 70 F1 F1 E1 E5 F5 E5 21 2A 3F E5 CD E3 7A F1
D20,80=01 02 00 46 3F CD F7 C1 2A 2A 3F 18 0C 2A 2A 3F
F20,80=01 02 00 46 F1 F1 C1 E1 E5 C5 F5 7C B5 28 11 21
D20,90=CD 15 46 CD 63 99 2A 2C 3F E5 CD 88 99 F1 C9 21
F20,90=2A 3F E5 F1 C1 E1 E5 C5 F5 E5 CD D1 6B F1 F1 C9
D20,A0=2A 3F E5 CD AB 6E F1 C9 21 06 00 39 7E 23 B6 CA
F20,A0=F1 E1 E5 F5 E5 C1 21 66 3F CD F7 C1 2A 2A 3F E5
D20,B0=EF 7F F1 21 ED 28 E5 E5 E5 CD 19 53 F1 CD 7C 68
F20,B0=CD 88 99 F1 18 21 2A 2A 3F E5 CD 88 99 F1 CD 63
D20,C0=F1 21 FA 28 E5 DD E5 E5 CD 38 69 F1 F1 21 ED 28
F20,C0=99 21 2A 3F 23 23 CD B5 C1 E5 CD 88 99 F1 C9 21
D20,D0=E5 CD 3D 64 C3 76 45:F20,D0=2A 3F E5 CD AB 6E F1
-----
```

```
. BC53/FIX - 05/19/87 - Patch to EnhComp BC/CMD
. Corrects error trap of IF exp THEN missing-clause
. Apply via, PATCH BC BC53
D2C,9A=17 5B:F2C,9A=98 63:D05,2F=B7 C2 98 63 C3 BD 66:F05,2F=00 00 00 00 00 00 00
-----
. BC63/FIX - 05/19/87 - Patch to PRO-EnhComp BC/CMD
. Corrects error trap of IF exp THEN missing-clause:. Apply via, PATCH BC BC63
D2C,AD=11 2F:F2C,AD=A6 37:D05,29=B7 C2 A6 37 C3 CF 3A:F05,29=00 00 00 00 00 00 00
-----
. INSTRU51/FIX - patch MC51/CMD to correct initialization of nested structures
. RND - 05/24/87
D22,2B=21 08 00 39 6E 26 00 7D B7 C4 71 AD E5:F22,2B=F5 21 00 00 39 E5 CD 71 AD D1 CD B1 E4
-----
. INSTRU52/FIX - patch MC52/CMD to correct initialization of nested structures
. RND - 05/24/87
D1C,A6=21 08 00 39 6E 26 00 7D B7 C4 B4 AB E5:F1C,A6=F5 21 00 00 39 E5 CD B4 AB D1 CD F9 E5
-----
. INSTRU60/FIX - patch MC60/CMD to correct initialization of nested structures
. RND - 05/24/87
D23,9E=21 08 00 39 6E 26 00 7D B7 C4 E9 91 E5:F23,9E=F5 21 00 00 39 E5 CD E9 91 D1 CD E6 DB
-----
. INSTRU61/FIX - patch MC61/CMD to correct initialization of nested structures
. RND - 05/24/87
D22,1A=21 08 00 39 6E 26 00 7D B7 C4 60 81 E5:F22,1A=F5 21 00 00 39 E5 CD 60 81 D1 CD 61 B8
-----
. INSTRU62/FIX - patch MC62/CMD to correct initialization of nested structures
. RND - 05/24/87
D1C,C1=21 08 00 39 6E 26 00 7D B7 C4 CF 7F E5:F1C,C1=F5 21 00 00 39 E5 CD CF 7F D1 CD BA C1
-----
. KSMA/FIX - 03/30/87 - Patch to LDOS 5.3 KSM/FLT
. Apply via; PATCH KSM/FLT.FILTER KSMA
D00,60=2A,FC,4D,01,47:F00,60=FD 6E 02 01 51:D00,BD=59:F00,BD=57
-----
. PATCH to correct MAPPER (24-Oct-84) from PRO-MACH2 - M. Houde
. MAPPER displays 4 granules in a row, which means that when disk allocation is 5 or more
. granules/cylinder, 2 lines/cyl are used, and 9 cylinders are displayed on the screen,
. whereas 18 cylinders are displayed when only one line/cyl is needed, filling the screen
. in each case. Unfortunately, when allocation is exactly 4 grans/cyl, only 9 cylinders are
. showed, leaving half of the screen blank. Cure is easy, as the culprit is an erroneous
. CP 4;JR C,n which should be CP 5;JR C,n (X'335B'-X'335C')
D03,74=05:F03,74=04
-----
. UNREM63 for UNREMOVE (12-May-84) - M. Houde
. This patch enables UNREMOVE/CMD (version 1.0a) from PRO-ESP to operate
. on 6.3 and 5.3 disks, as well as 6.x and 5.1 disks.
. This patch must be added as an X-type patch, as no room is available inside the file.
X'311D'=CD C0 35 ; was 'E E6 07
X'35C0'=3A CD 36 CB 5F 20 04 7E E6 07 C9 7D C6 11 6F 7E
X'35D0'=E6 1F 00 C6 50 CD 2E 31 0E 20 3E 02 EF 3E 02 EF
X'35E0'=2B 7E E6 F8 0F 0F 0F CD 2E 31 0E 3A 3E 02 EF 7E
X'35F0'=E6 07 23 4E CB 21 17 CB 21 17 CB 21 17 D6 50 C9
-----
. ALTDSK63 for ALTDISK (7-Feb-85)
. change DOS7 6.x to 6.3 (X'32D1')
D03,2D=33:F03,2D=78
. GAT set DOS version to 63, set bit 3 of byte 0CDH (X'376C')
D07,DE=63 00 8B:F07,DE=62 00 83
. Boot sector 2, DOS version (X'39C1')
D0A,3B=63:F0A,3B=62
-----
```

```
. NAME63 for NAME (19-Jul-84)
. Change check for year<88 ti year<100 (X'30C8')
D00,FD=64:F00,FD=58
. Change 6.x to 6.3 (X'31E7')
D02,20=33:F02,20=78
. Since TRSDOS 6.2 Boot sector 2, bytes x'18'-x'1F' are no longer used for
. date storage, but rather Levelxx (system disks), it would be better to
. prevent NAME from changing this area. Change JR Z, to JR (x'3165')
D01,9E=18:F01,9E=28
-----
. MRS512/FIX - 05/19/87 - Corrects DW expansions with mixed modes
. Apply via, PATCH MRAS MRS512
D02,46=7C 8F:F02,46=75 8F:D1C,03=75 8F:F1C,03=C4 62:X'8F75'=AF 32 9B 65 C3 C4 62
-----
. MRS612/FIX - 05/15/87 - Corrects DW expansions with mixed modes
. Apply via, PATCH MRAS MRS512
D02,2A=67 62:F02,2A=60 62:D1B,C6=60 62:F1B,C6=BB 35:X'6260'=AF 32 92 38 C3 BB 35
-----
. PDS553/FIX - Patch to PaDS's BUILD module for 5.3
. Apply via; PATCH PDS.PDS PDS553
D15,D1=70 3D 52 45 29 0D 00 00 00:F15,D1=61 3D 2C 70 3D 52 45 29 0D
-----
PaDS: PATCH PDS.PDS (D20,75=20 30:F20,75=05 20)
-----
PRO-PaDS: PATCH PDS.PDS (D20,55=20 30:F20,55=05 20)
-----
. RSH61/FIX - 05/27/87 - Patch to RSHARD6/DCT - Apply via, PATCH RSHARD6/DCT RSH61
. enable JCL
D00,C1=C3 C1 68:F00,C1=B7 20 27:D08,ED=B7 C2 DF 60 C3 BB 60:F08,ED=00 00 00 00 00 00 00
. ASCII to binary conversion
D03,E9=C3 C8 68:F03,E9=83 18 E6:D08,F4=83 57 C3 BA 63:F08,F4=00 00 00 00 00
. check step value for initialization
D00,D6=06:F00,D6=6E
-----
. RSH61/FIX - 05/27/87 - Patch to RSHARD6/DCT - Apply via, PATCH RSHARD6/DCT RSH61
. test direct invocation
D00,1A=C3 09 39:F00,1A=CA 81 35:D09,39=CB 5F CA 81 35 C3 11 30:F09,39=00 00 00 00 00 00 00 00
. enable JCL
D00,9C=C3 11 39:F00,9C=B7 20 27:D09,41=B7 C2 BA 30 C3 96 30:F09,41=00 00 00 00 00 00 00
. ASCII to binary conversion
D04,0C=C3 18 39:F04,0C=83 18 E6:D09,48=83 57 C3 DD 33:F09,48=00 00 00 00 00
. check step value for initialization
D00,B1=06:F00,B1=6E
-----
. RSH62/FIX - Patch to RSHARD6/DCT - 06/11/87 - Apply via, PATCH RSHARD6/DCT RSH62
. Corrects installation into high memory when low is not available
D02,CD=CD 1D 39:F02,CD=22 2B 35
D09,4D=21 00 00 45 3E 64 EF 22 2B 35 C9:F09,4D=00 00 00 00 00 00 00 00 00 00 00
-----
. SDAT53/FIX - 04/14/87 - Patch to EnhComp SUPPORT/DAT
. Corrects USING on certain single/double fractional values
. For LDOS 5.3; PATCH SUPPORT/DAT (D60,14=21:F60,14=C2)
. For LDOS 5.1; PATCH SUPPORT/DAT (D60,14=21)
-----
. SDAT62/FIX - 04/14/87 - Patch to PRO-EnhComp SUPPORT/DAT
. Corrects USING on certain single/double fractional values
. PATCH SUPPORT/DAT (D5F,BF=21:F5F,BF=C2)
-----
```

```
. SYS6D/FIX - 04/23/87 - Patch to LDOS 5.3 ROUTE Command
. Corrects re-use of previous buffer/FCB
. Apply via;
PATCH SYS6/SYS.SYSTEM (D36,31=6F:F36,31=67)
-----
. SYS6E/FIX - 06/18/87 - Patch to DIR
. Corrects obscure problem in space calc
. Apply via, PATCH SYS6/SYS.SYSTEM SYS6E
D0A,4D=FF
F0A,4D=00
-----
. TED5B/FIX - 05/19/87 - Patch to LDOS TED
. Fixes text deletion on LOAD if DELETE mode is active
. Apply via, PATCH TED.UTILITY TED5B
D04,CF=DB
F04,CF=B9
-----
. TED6C/FIX - 05/19/87 - Patch to LS-DOS TED
. Fixes text deletion on LOAD if DELETE mode is active
. Apply via, PATCH TED.UTILITY TED6C
D04,B1=AC
F04,B1=8A
-----
. INSTALL3/JCL    Thomas L Keister Jr        15 Jun 83
. LDOS 5.3 change by Luis M. Garcia-Barrio    7 May 87
.                 8/N/1 #4, 215/848-5728
.
. Following a Global RESET, this JCL file will install
. SUPERLOG into a minimal floppy-based system including
. KI/DVR, Type-ahead, and JKL screen print.
.
. Valid for MISOSYS' LDOS 5.3.
. For LDOS 5.1.3 restore the original PATCH listed below.
.                SUPERLOG series 'L'.
//PAUSE Type <ENTER> to proceed. <BREAK> to exit.
. Installing SUPERLOG.  Screen Storage to MEMORY.
SUPERLOG (LOAD,MEM)
. Configuring System...
SET *KI TO KI/DVR (TYPE,JKL)
VERIFY (OFF)
. Disable (shift)<BREAK> option of LDOS.
MEMORY (A=X'44DD',B=X'D4')
.
. The LDOS 5.1.3 patch was PATCH SYS0/SYS.SYSTEM:0 (D03,37=D4)
. The new LDOS 5.3 patch follows:
.
PATCH SYS0/SYS.SYSTEM:0 (D03,2F=D4:F03,2F=CD)
.
. Now add any additonal drivers, filters, options you desire.
. Then type: SYSTEM (SYSGEN) to save this configuration.
. SUPERLOG will be operational each time you reBOOT.
//EXIT
```

[Editor Note: Contrary to The Blurb, I found room to squeeze in this section]

Our Compuserve Forum - PCS49

FORUM commands

The following commands may be entered at the main forum FUNCTIONS menu. To find out more about a particular command, enter the command at the prompt below. If you'd like a listing of the entire online HELP displayed at your terminal, enter "..." (three periods without the quotes), at the next prompt. The listing is approximately 5 pages long. Remember to use CONTROL-A to stop the output from scrolling, and a CONTROL-Q to start the output. For general information about using a forum, type "INstructions" at the main forum FUNCTIONS menu.

Please note that "H" may be entered at most Forum menus and prompts to obtain help specific to that menu or prompt. More information is available for:

INstructions	Leave	Read
Scanning	BROwse	COnference
DLibrary	Bulletins	MDirectory
SSubtopic	OPtions	Help
EXit	USTatus	SENd
NEW	Information	Userlog
SNames	LNames	VERsion
HIgh	RForward	RReverse
RThread	RSearch	RMarked
RIndividual	SThread	WHO
DAY	TIME	

Forum policy on deleting messages:

(Fm: LDOS Support jjkd) (1) Please delete all (P)rivate messages after reading. A (P)rivate message is denoted by the characters (P) in the header. (P)rivate messages don't help the general membership, and do take a valuable message slot.

(2) Delete all single line or quickie "Thanks!" types of messages after you read them. This will free up a message slot for another message. If any information is in there along with the thanks, go ahead and leave the message. It may help other readers.

(3) Of course, if you are not satisfied with the appearance or content of one of your messages after sending it, go ahead and delete it when posting another copy.

Other than these situations, messages should not be deleted. Thanks!

SIG paging

(Fm: LDOS Support jjkd) You may temporarily set paging off via the command: **SET PAG OFF** at the main Forum prompt. To turn paging off permanently, GO CIS-9 and set your terminal parameters to have paging off. While you are there, you can set the proper number of lines per page and your terminal type if desired.

Garbage Messages

(Fm: LDOS Support jjkd) You can delete a message that is to or from yourself by using the Delete command from the main Forum prompt or menu. You will then be prompted for a message number to delete. Supply the correct number and away she goes. You will not be allowed to delete a message if you are not the sender or recipient.

Problems accessing a BBS

(Fm: LDOS Support jjkd) Assuming that you are using the exact same equipment to call our Compuserve forum as a BBS you are having problems with, then there are three possible sources of problem:

(1) Phone lines from you to them. Since your dialed route to CompuServe is likely not the same as to the BBS, there easily could be a problem from you to them. If long distance, try another carrier (AT$T, SPRINT, etc). If not, try taking your equipment to a friend's house who lives more than five miles or so from you, and see if that makes a difference.

(2) Baud rate, parity, stop bits, modem mode. Make sure that you match what the other end is expecting. Most common is seven bit word, even parity and one stop bit. You are limited to 300 baud, and should be in originate mode.

(3) Make sure that the other end really is a computer with a modem. When I called the number you gave ("insert number here"), I got a message indicating that the number was disconnected. Always try calling with a regular phone in addition to a modem to make sure that a person or message isn't answering. Repeatedly calling a normal residential phone number mistakenly thought to be or published as a BBS number can cause untold grief at the other end.

Unsqueezing Files

(Fm: John Bode) How do you unsqueeze a file that is squeezed when you download it and then it won't run? Please I need to know how to undo the things that you do!

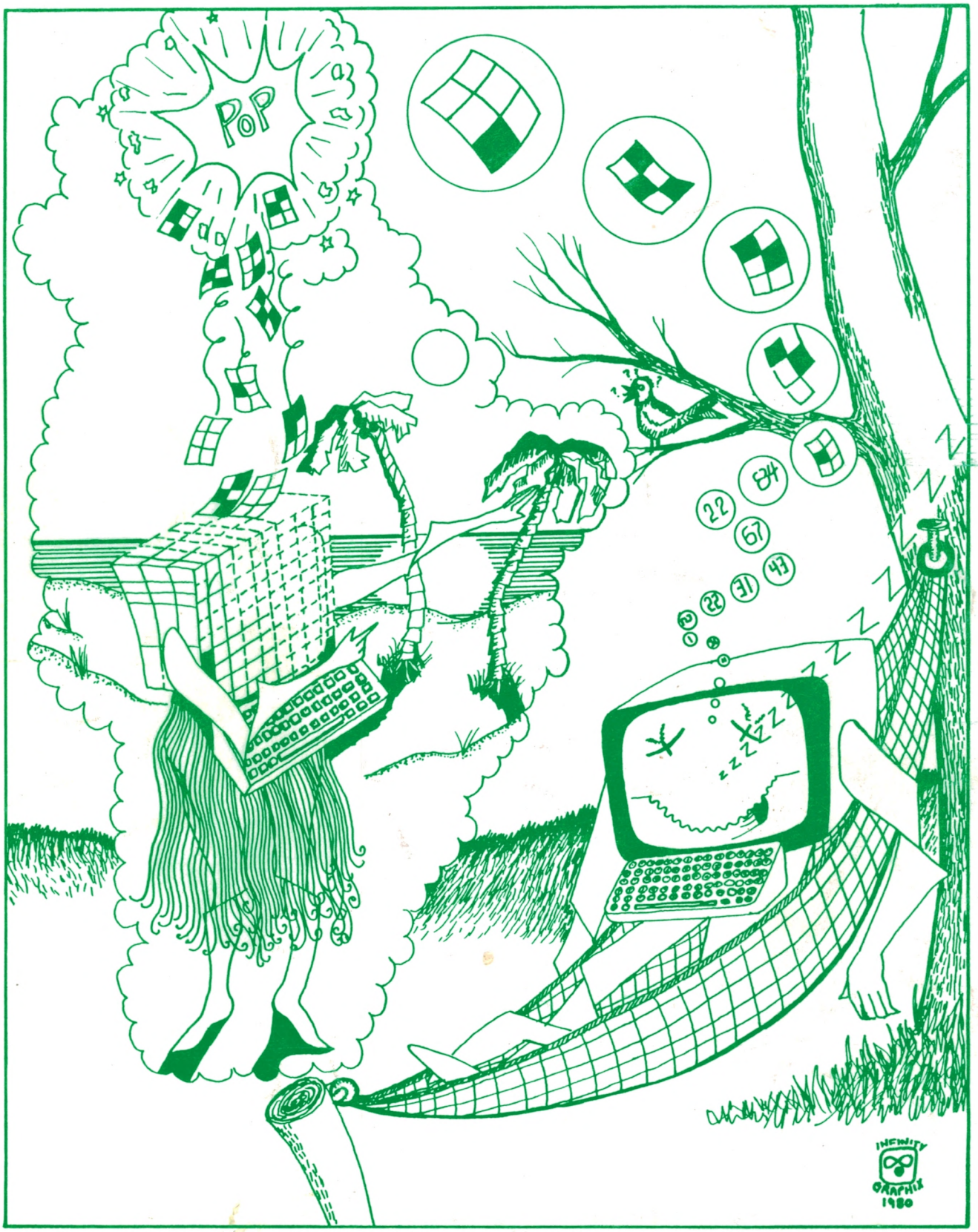
(Fm: Michael Johnston) John, to unsqueeze or de-arc a program you must have one of the several archive or squeeze utilities. They will extract the file to its usable condition. Just download the following: (1) ARC4/CMD, (2) XARC4/CMD, (3) USQ4/CMD, (4) LU4/CMD, (5) ARCH4/CMD.

The first two are the newest and preferred files. the middle two are the old style that are used rather infrequently now. The last one also fits into this catagory as it has been replaced by arc4/cmd. If you don't want to download all that stuff (who can blame you?) Just send a disk in a mailer with a return address label and postage to: Michael R. Johnston, 155 DeKalb Ave, Brooklyn , N.Y. 11217 Better send two disks and I will copy ALL the files you need onto them OK?

(Fm: LDOS Support jjkd) It depends on how the file has been made smaller. The breakdown is as follows:

Ext	Program needed
ACH	ARCH3 or ARCH4, depending on OS
ARC	XARC4 or ARC4
?Q?	USQ
LBR	LU and possibly USQ
LQR	Both LU and USQ
ESS	PROCES, not PRO-CESS

The above utilities are available for TRSDOS 6 in DL 6, with some new arrivials here in DL 0.


POP
22
84
67
31
47
INFINITY
GRAPHIX
1980