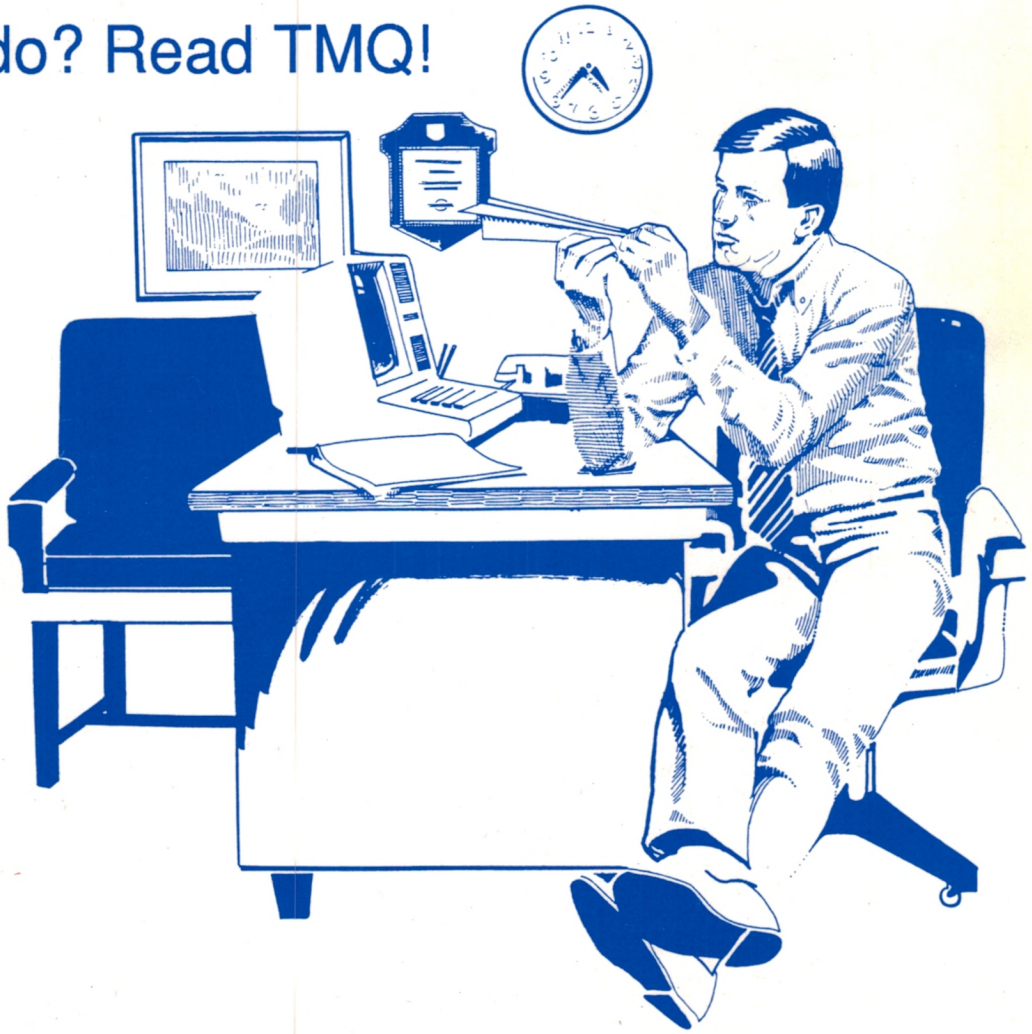
THE MISOSYS QUARTERLY

In this issue:

Nothing to do? Read TMQ!



Volume III.ii **$10** **Fall 1988**

TRS-80 Software and Hardware from MISOSYS, Inc.

PRICE LIST effective October 1, 1988
Prices subject to change without notice

Product Nomenclature	Mod III	Model 4		Price	S&H
6 foot M-M printer cable	same	R-06-CMM		$19.95	D
BSORT / BSORT4	L-32-200	L-32-210		$14.95	
CON80Z / PRO-CON80Z	M-30-033	M-31-033		$29.95	
diskDISK LS-diskDISK ...	L-35-211	L-35-212		$39.95	
DSM51 / DSM4	L-35-204	L-35-205		$59.95	
DSMBLR / PRO-DUCE	M-30-053	M-31-053		$34.95	
EDAS / PRO-CREATE	M-20-082	M-21-082		$74.95	D
EnhComp / PRO-EnhComp ...	M-20-072	M-21-072		$99.95	D
Filters	L-32-053	n/a		$14.95	
GO:Maintenance	**n/a**	**M-33-100**		**$59.95**	**F**
GO:System Enhancement ...	**n/a**	**M-33-200**		**$59.95**	**F**
GO:Utility	**n/a**	**M-33-300**		**$59.95**	**F**
Hardware Interface Kit ..	n/a	M-12-110		$29.95	
HartFORTH/PRO-HartFORTH..	M-20-071	M-21-071		$59.95	B
Lair of the Dragon	same	M-55-021		$29.95	
LBMU-M4	n/a	L-50-515		$29.95	
LDOS 5.1.4 User Manual ..	L-40-020	n/a		$30.00	D
LDOS 5.3 Max Upgrade Kit.	M-10-833	n/a		$34.95	
LDOS 5.3 Mod3 Upgrade Kit	M-10-033	same		$34.95	
LED / LS-LED	L-30-020	L-30-021		$29.95/$34.95	
Little Brother-M4	n/a	L-50-510		$74.95	F
LS-DOS 6.3 Upgrade Kit ..	n/a	M-11-043		$39.95	
LS-DOS 6.3 Site License .	n/a	M-11-143		varies	
LS-Host/Term	n/a	L-35-281		$59.95	
LS-UTILITY	n/a	L-32-150		$29.95	
MC / PRO-MC	M-20-064	M-21-064		$124.95	D
Mister ED	n/a	M-51-028		$39.95	B
MRAS / PRO-MRAS	M-20-083	M-21-083		$89.95	D
PRO-WAM	n/a	M-51-025		$74.95	F
" Programmer Toolkit....	n/a	M-51-225		$29.95	
Programmer's Guide DOS 6.	n/a	M-60-060		$25.00	B
QuizMaster	L-51-500	n/a		$24.95	
RATFOR-M4		M-21-073		$99.95	F
RSHARD - R/S HD driver ..	M-12-013	same		$29.95	
switch box, AB 36	same	R-SB-PAB		$40.00	E
switch box, ABCD 36	same	R-SB-PAD		$50.00	F
switch box, cross 36	same	R-SB-PX2		$50.00	F
TBA / LS-TBA	L-21-010	L-21-011		$39.95	D
TeleTrends TT512P modem .	**same**	**H-4P-512**		**$79.95**	**E**
The Gobbling Box	same	M-55-020		$19.95	
THE SOURCE 3-Volume Set .	n/a	L-60-020		$40.00	**
UNREL-CPM		M-32-054		$49.95	
UNREL-T80	same	M-30-054		$49.95	
UTILITY-I	L-32-070	n/a		$29.95	
XLR8er e/w 0K RAM	n/a	R-MB-003		$175.00	F
XLR8er e/w 256K RAM	n/a	R-MB-004		$255.00	F
LC-890 Laser Toner	n/a	R-LP-910	-	$20.00	D
LC-890 Laser OPC Kit	n/a	R-LP-920	-	$110.00	H

** Final closeout price including surface freight worldwide

Freight codes: A = $2.50; B = $3.00; C = $3.50; D = $4.00; E = $4.50; F = $5.00; G = $7.50; H = $10.00; All unmarked are $2.00 each
Canada/Mexico add $1 per order; Foreign use US rates times 3 for air shipment

Virginia residents add 4.5% sales tax. We accept MasterCard and VISA; Checks must be drawn on a US bank.
COD's are cash, money order, or certified check.

MISOSYS, Inc.
P.O. Box 239
Sterling, VA 22170-0239
703-450-4181 Orders only: 800-MISOSYS (800-647-6797)

THE MISOSYS QUARTERLY

Volume III.ii **Fall 1988**

Table of Contents

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

The Blurb by Roy Soltoff

Points to Ponder

The **year-end holidays** are coming up, and as has been typical for many years now, MISOSYS closes up shop and vacates the premises. This year is no different. Please be advised we will be **closed for business** starting on Friday **December 23rd** and will reopen "fresh as a daisy" on Monday January 2nd 1989. I wish you and yours a happy and safe holiday season. We'll be off to the mountains of North Carolina, stuffing 16 family members into the in-laws cabin near Edneyville.

Turning now to other topics, I'll try to keep this brief. For one thing, I am down to the wire in getting this TMQ prepared for the printer. So what else is new? I always find myself short on time and long on work. But its great to be busy.

I thought it was time to pop a **new machine** in to here to help in speeding up the computing process. So I went out and got an AST **Premium/386** machine which hums along at 20 MHz. A stock machine comes with 1 Meg of RAM and a 1.2 Meg 5.25" floppy. I added a small 20 Meg drive for now as I intend to build this machine up over next year. The memory card takes the small in-line memory modules (SIMMS). Because of the volatility of the DRAM market these days, most computer manufacturers are supporting both 256K and 1 Meg SIMMS; the AST 386 does. The memory board handles four banks of SIMMs, four SIMMs per bank. The first bank holds only 256K SIMMs (that's the 1 Meg); the other three banks hold either type. You need to add SIMMs in groups of four to fill a bank. Now the AST machine uses 70 nanosecond static column SIMMs which go for about $595 per SIMM; the 256K SIMMs go for $140. But anyone who gets into the big machine market will find that 2 Megabytes of RAM is totally insufficient; you need 4-5 Megs! So I'm waiting until prices drop a bit before taking that plunge. $2200 for memory is a bit steep! An NEC Multisync II color monitor and a Video Seven VGA Deluxe provide the viewport.

The other point about a "big" machine is the disk capacity and speed. I am planning on doing a lot of graphics work and publishing on this machine. I expect soon to get PageMaker for publishing and advertising workup. WINDOWS/386 is another thing on the list to support multi-tasking. All of that takes memory and fast disks. So I'm looking at something in the order of 100-140 Megabytes of fast disk access. That also doesn't come cheap.

Of course, if I recollect the old Model I days, no one ever went and bought a "full system". You grew it gradually. Most of us "oldtimers" started with a 16K Level II (or even a 4K Level I and ran Microchess!). The next advance was an expansion interface with possibly 16K additional memory. Most of the time you waited until you could get memory from third party sources as Tandy was charging about $295 for 16K. Perhaps $595 for 1 Meg isn't so bad after all. The E/I enabled you to add a second cassette. Few did; the E/I afterall was the device to get you into a disk drive with all of its 87.5K of storage. A disk drive cost you about $400. So building up a full 48K 4-drive system could set you back somewhere in the neighborhood of $4000. It's interesting that a full Model III with two external drives was about $4000 when it first came out. A Model 4 was about $3000 with two more drives. I spent about $4000 on my Model 2000. It seems like $3000-$4000 has been par (not including printers). But this 386 will be more. Now will it last longer than the Model I?

Here's some good news for Model 4 **LB Data Manager** users. I have enlisted the services of RicLin Computer Products (nee Rich Deglin, author of MC) to work on our LB package. This work is long overdue. My plan is to develop a new version for release sometime next year. This work is to be broken up into phases. The first phase was to groom all of the source code to compile under PRO-MC on the Model 4 side of the house, and Microsoft's C compiler 5.1 on the MS-DOS side of the house. I am happy to report that this first phase is complete. A benefit of this completion is that the Model 4 version no longer has the memory restraints which the Aztec C-compiled version had. Thus, **LB4 can run with PRO-WAM installed!** Since LB is a "behaved" program, it works hand in hand with PRO-WAM; you can pop open a WAM window from LB and import/export to your heart's content. This is what I have been waiting for. Note that if you are limited to a 128K Model 4 and are using the extra 64K as a MemDisk, you cannot have PRO-WAM installed. You either need more than 128K (XLR8er or Alpha Tech board) or are operating with more than two floppy drives or a hard drive.

Now phase one was not supposed to incorporate changes; however, Rich and I discussed some small changes and introduced two. We added the View/Modify Path command to the Model 4 which you use to specify what disk drive the various data files are on. This speeds up the process of getting to your data. And we reduced the number of keystrokes needed to specify sorting criteria.

LB is a very large and complex set of programs. As such, it requires a great deal of field test. Therefore I have not cut this new version into production. But I will make a copy available to any current LB user so you can try it out. All you need to do is send me your existing LB executable disk and I'll return it with a copy of this beta version of LB.

As a service to the Model III/4 community of users, I'd like to set aside some space in TMQ to publicize the following three things. First, a list of **phone numbers** of companies still servicing and supporting this market. If you are a company who sells Model III/4 products, send me the information. This must come from the company, not a statement from a third party. Second, I would like to publish a list of **public computer bulletin boards**. If you run a BBS that supports the Model III/4, please forward the particulars to me. Third, I want to publish a list of **computer clubs** which support the TRS-80 user. Please forward your input. I don't envision this as a one-time report, but a continuing publication.

MISOSYS Holiday Savings!

Everybody likes a bargain at holiday time. So here's one I think you will like. The TMQ coupon accompanying this issue to subscribers gives you 30% off all software and book items and 15% off all hardware items (DRAM excluded) sold by MISOSYS which are not already on sale. So bundle up your needs and get them in on that one order. Check with your friends who don't subscribe to TMQ and order for them.

MISOSYS Hot List

This list represents what products in our catalogs have been the most popular in terms of sales units. The list excludes TMQ subscriptions and Disk Notes. LS-DOS 6.3 was added to the list in June 1988. The current month is October 1988; prior 3 months is Jul-Sep 1988; and prior 12 months is Oct87-Sep88.

MISOSYS HOT List

Current Month	Prior 3 months	Prior 12 months
LS-DOS 6.3	LS-DOS 6.3	LDOS 5.3
LDOS 5.3	LDOS 5.3	LS-DOS 6.3
Lair	*The Source*	PRO-WAM
PRO-WAM	MRAS	*The Source*
The Source	DSMBLR	diskDISK
XLR8er	PRO-WAM	Little Brother

TMQ Schedule

Our target for mailing the *THE MISOSYS QUARTERLY* is the last week of the respective month as follows: Winter issue in February, Spring issue in May, Summer issue in August, and Fall issue in November. This schedule may place your TMQ late in the season based on the cover date; however, it follows from the mailing of issue I.i on August 19th, 1986.

In order to avoid the Thanksgiving-Christmas rush, this issue was prepared early. It goes to the printer on November 8th; with luck, it will get out in the mail by November 15th. I hope to continue to pull up the mailing date of succeeding issues until the target delivery is close to the beginning of the season.

The TMQ coupon has a field for entering your receipt date. If you are returning the coupon, please note the receipt date on the coupon and get it back to us. Unfortunately for the last issue, a few folks recorded "yes" instead of the date of receipt. If you send me back the coupon, I certainly know that you got the issue. Boy I really have to make things simple; the DATE OF RECEIPT, folks.

TMQ advertising

If you are interested in reaching a dedicated TRS-80 audience, consider *THE MISOSYS QUARTERLY*. If you have a TRS-80 Model III or 4 related product to sell, you can reach these buyers by placing your advertisement in our publication. TMQ is read world-wide. Our subscribers are predominantly in the United States; however, we do have a significant number in Canada, Europe, and Australia. Space rates are as follows:

Full page	$250
Half page	$150
Quarter page	$100

We accept only black & white ads; however, ads for our inside covers are printed in the same color as the cover (TMQ alternates between PMS colors: green 354, purple 266, blue 293, and red 199). If you would like to place your ad in *THE MISOSYS QUARTERLY*, give me a call.

PD Software Librarian

Vic McClung has volunteered to be the librarian for the collection of TRS-80 public domain diskettes. Henceforth all requests and contributions be directed directly to him at:

Vic McClung
914 Crescent
Sikeston, MO 63801
USA

DISK NOTES 3.2

Each issue of *THE MISOSYS QUARTERLY* contains program listings, patch listings, and other references to files we have placed onto a disk. DISK NOTES 3.2 corresponds to this issue of TMQ. If you want to obtain all of the patches and all of the listings, you may conveniently purchase a copy.

DISK NOTES is priced at $10 **Plus S&H.** The S&H charges are $2 for US, Canada, and Mexico, $3 elsewhere. If you purchase DISK NOTES 3.2 with the coupon which accompanies this TMQ issue, you can save $2.50; the cost then being only $7.50 + S&H.

Fixes in this issue

I did away with *The Patch Corner* because the number of patches had dwindled to a handful. This issue again has only a handful, but here's a reference to them.

P. 20 CED/TXT

P.21 SYS6H/FIX - For LDOS 5.3 DIR P
P.22 SYS053F/FIX - LDOS 5.3 & French III/4
P.28 VIDTXB/FIX - For Tandy's VIDTEXT
P.30 AL1/FIX - For Prosoft's Allwrite 1.13
P.57 LBSORT2/FIX - For LB's LBSORT module
P.58 MCOPTSBx/FIX - For MC's MCOPT/CMD
P.64 PDSxxx/FIX - For PRO-PaDS and LS-DOS 6.3
P.66 LSFEDII/FIX - For LSFEDII and LS-DOS 6.3
P.80 XLR8xx/FIX - For XLR8er and LS-DOS 6.3

Out of print TMQ's available

For out of print issues, we are providing back issues of *THE MISOSYS QUARTERLY* via copier reprint. The price is **$12.50** plus $2.75 S&H in the U.S. and CANADA. For foreign zone D, the S&H rate is $5.50; zone E is $6.50. The price for regular back issues still in print is $10 + S&H. We are currently out of print on all issues except II.iii. Here's a synopsis of past issues:

Volume I See the index in issue III.i. An index to Volume II prepared by Elmar von Muralt couldn't fit into this issue; it will definitely be in the next issue.

II.i David Hall on the 64180; Gary Phillips on XLR8 & 4P; Doug Tittle on sorting PRO-WAM data; WORD with DW II.

II.ii Extended DATE$ of Model I LDOS 5.1.4; Input SUBroutine for QuickBASIC; BASIC Interface to @EXMEM; HIRES Graphics for MC; Focus on speed.

II.iii Headline driver for I/III LDOS; XLR8er installation; CTL255 filter for PRO-WAM; FIXBANKS for your XLR8er; 4P boot ROM disassembled; C bit fields.

II.iv Tapes, Disks, and CMD files; Searching with DSM4; EXMEM<>BASIC interfacing; Testing printer ready; Revision to HIBANKS; WAMDUMP: convert a PRO-WAM /APP file to /CMD file; SYSDRV: A system drive other than :0.

III.i Reading NEWDOS/80 disks; An LB archival utility; Popup Application Window; XMODEM in C; Getting into computer math, part I; TMQ Volume I index.

Market Research

Finally, here's some feedback on our external hard drive market research request. I have very little to report over and above what was stated in the last issue. I am going forward with the project, but must wait until the host adaptor is finalized before making any public announcement. Anyone providing a request for information on the proposed package will get a mailed announcement in addition to any posting in TMQ.

To recap the proposed hard drive package, here are the tentative specifications: 20 megabyte half height drive (Seagate ST225), PC-type hard disk controller (Western Digital WDXT-GEN) with on-board MS-DOS driver & formatter, MISOSYS designed TRS-80 host adaptor, Leadman drive case for two half height drives e/w 60-watt power supply (switchable 115V/230V) and fan, device drivers for both LDOS (model III mode) and LS-DOS (model 4 mode) along with archive/restore utility, and our diskDISK software. Our target price is still $495.00. We are also planning a hardware clock on the host adaptor as a low-cost option. The beauty of this configuration is that everything except the host adaptor is re-usable in an MS-DOS PC environment. Thus, if you can your TRS-80, you have a 20 Megabyte drive with controller totally reusable in a PC.

MISOSYS is ready to ship the off-the-shelf components now to those folks looking to acquire a hard drive, controller, or external case. Perhaps you just want to add a 20 Megabyte drive to your PC. Or maybe you are a hacker wanting to upgrade your Radio Shack 5 Meg drive to a 20 Meg drive. Check out the next section of *The Blurb.*

New Product Announcements

Golden Oldies: Maintenance

The *GO:MTC* product is a collection of programs designed to provide maintenance support services for your computer operation. The programs in this group have been rewritten for Model 4 LS-DOS 6.3. You get **DIRCHECK** to perform an integrity check of your disk's directory and repair certain kinds of errors; **FIXGAT** to re-construct a corrupted Granule Allocation Table; **IOMON** for trapping disk input errors; **MAPPER** to check the granulization of files stored on your disk; **RAMTEST** to perform an exhaustive test of all DRAM memory in your computer; and **UNREMOVE** to restore a file inadvertently deleted. All documentation has been revised and is printed in a convenient 5.5" by 8.5" format. Order M-33-100 for $59.95 ($5 S&H US).

Golden Oldies: System Enhancement

The *GO:SYS* product is a collection of programs designed to provide additional features to LS-DOS 6.3 operation. The programs in this group have been rewritten for Model 4 LS-DOS 6.3. You get **DOCONFIG** for manipulating

CONFIG/SYS files; **DOEDIT** to provide command editing; MEMDIR to get a memory directory; **PaDS** for the provision of Partitioned Data Sets; **PARMDIR** to obtain parameterized directory information for listings and Job Control Language processing; **SWAP** to switch drive assignments; **WC** for wild card command invocation; and **ZSHELL** to provide command line I/O redirection, piping, and multiple commands on a line. All documentation has been revised and is printed in a convenient 5.5" by 8.5" format. Order M-33-200 for $59.95 ($5 S&H US).

MISOSYS will accept as trade, any of the following older versions of individual items: PRO-GENY, ZSHELL, PaDS. Each item traded in is worth $10 towards the purchase of GO:SYS. Just send your old diskette back. Trade in offer expires March 31, 1989!

Golden Oldies: Utility

The ***GO:CMD*** product is a collection of products designed to provide additional utility for your computer operation. The products in this group have been rewritten for Model 4 LS-DOS 6.3. You get **FASTBACK** and **FASTREAD** for hard disk large file archive/restore; **PRO-CESS** to manipulate executable command files; **COMP** to compare two files or disks; **FED2** to investigate and zap disk or file sectors on a full-screen basis; **IFC** updated with new features for interactively copying, moving, renaming, deleting, and invoking files; **ZCAT** for cataloging 6.3 diskettes. All documentation has been revised and is printed in a convenient 5.5" by 8.5" format. Order M-33-300 for $59.95 ($5 S&H US).

MISOSYS will accept as trade, any of the following older versions of individual items: LS-FED-II, PRO-IFC, PRO-ZCAT; PRO-CESS. Each item traded in is worth $10 towards the purchase of GO:CMD. Just send your old diskette back. Trade in offer expires March 31, 1989!

1200 baud internal modem for Model 4P

That's right, you heard it. MISOSYS has acquired the TeleTrends 1200 baud Hayes compatible modem which slips right into the Model 4P's internal modem slot. As soon as we get our feet wet with this unit, we'll be working up a package for the desktop Model 4. For now, you Model 4P folks can upgrade to a real Hayes-compatible modem operating at 1200 baud. Our introductory sale price is just $79.95 ($5 S&H in US). It's just what you needed for your 4P.

Hard Disk Drive Components

Because of our pending development of an external hard drive package, MISOSYS is now stocking components for its assembly. We want to make these available separately. Here's what we have along with your cost:

1. Seagate ST-225 hard drive, 20 Megabyte formatted,
   $225 + (5 lbs) $7.50/$10.50 S&H
2. Western Digital WD-XTGEN drive controller,
   $75 + (1 lb) $4.00/$6.00 S&H
3. External hard drive case e/w 60 Wt ps; for 1 full or 2 hf ht,
   $125 + (10 lbs) $13.00/$18.00 S&H
4. Items 1&2 plus connecting cable set (data & control)
   $285 + (6 lbs) $8.50/$11.50 S&H

Note that S&H figures are US 48 States/Hawaii&Alaska. Outside US please write for shipping charges.

Standby/UPS Power Supply

If your area is subject to unexpected outages, surges, brownouts, and other nuisances of clean power, I suggest you consider a standby system or an UPS. The difference between a Standby Power System (SPS) and an Uninterruptable Power System (UPS) is that an UPS is always feeding your system with regulated, battery driven power whereas the SPS "rides the line" and switches over to the battery backup within milliseconds of a brownout or blackout. The PTI Turbo/2 line of SPS' we're carrying switch to backup under 1 millisecond and also have surge protection. Most, if not all, computer power supplies should be able to hold their DC voltages for much longer than a millisecond when the power is cut. So a SPS is sufficient for most folks, and economical to boot. Our pricing on the SPS packages recommended by us are:

PTI Datashield PC200A	$299 + S&H
PTI Datashield Turbo/2 300A	$469 + S&H
PTI Datashield Turbo/2 450A	$499 + S&H
PTI Datashield Turbo/2 625A	$549 + S&H

These weigh about 37 pounds each so shipping will be by UPS ground only, and only in the 48 United States. Shipping will vary between $5 and $18. The PC200A unit is recommended for floppy based systems; the Turbo/2 450A will handle a bigger 286 or 386 class machine. I have one 450A unit feeding Brenda's AST Premium/286 (40Meg drive) and Radio Shack Line Printer V. Another handles my AST 386.

Family Update

by Roy

Thanks for all the well wishes Brenda and I have received for our newest Soltoff, **Benjamin**. He'll be five months now in two more days, so you know when I am writing this. "Benj", as Stefanie calls him, has recently started moving around a bit while on the floor. He's not quite crawling yet, but he is turning over and rolling. So the "pick up your toys from the floor" lectures to Stacey and Stefanie have begun. Benjamin's on the move...

It's difficult to acknowledge just how lucky we've been with him sleeping through the night. We have not had problems with colic like some of our friends have had. Benjamin is just a happy baby. He goes to sleep happy, he wakes up happy, and he's mostly happy during the day. Benjamin does like being held, to the point that either Brenda or I have had to hold him during dinner time. But now that he's getting some mobility, and his world is expanding, that doesn't seem like a requirement anymore. He's even noticing the TV!

The only negative aspect of his appearance on the scene has been a considerable amount of spitting up. He's been gaining weight and hasn't been sick, so our pediatrician finds nothing alarming. He does go through a bunch of cloths every day. Here, we use cloth diapers as "cloths", and the Huggies are for the "bottom".

The alarm now goes off at 6:00 am, but we are not always up then - it's just too early. **Stacey started kindergarden** in September and has really gotten into the swing of things. She goes five days a week for a morning session. Based on some of the things she has said to me, I have the feeling she has aged a few years during the few months in school so far. Things like, "you're treating me like a baby", and "boy, you don't know anything" seem a little too soon. There was this story about the high school graduate went off to college thinking how little his father knew, and was shocked to find out how much his father learned while he was away at college. I did tell Stacey that if I was as smart as she when I was young, why did I forget everything I knew? Ah youth, why we waste it on the young!

I am pleased to see the development in small motor control which has occurred over the past few years with her. Starting from crayoning in coloring books where the whole page appeared to get the same color to now where each segment of the picture has its own correct color - and all within the lines, seems like a miracle. She's even starting to draw freehand. She did a crayon sketch of me which just has to be saved. No, I won't print it in TMQ.

The biggest step I see in development is the beginning stages of reading. Now I never grew up with ***Doctor Seuss***. But we started getting those books a few years ago. Of course we had to read the words, then. I can now sit down with Stacey and get her to read most of the words of some of the easier books. I think learning to read is a big leap in development. Now our job is to nurture that ability so that readng becomes more important than the "tube".

Stefanie is now in the 4-year old preschool class. Most folks still think the two girls are fraternal twins, not 16 months apart. But it's been great having them close together in age. Stefanie and Stacey have the same friends, and most of the time they are friends with each other.

Stefanie is still the problem eater. But then, she's only four. We had a birthday party for her in October and there were about 16 kids altogether - some crowd! We made it a pizza party, making everything from scratch. **Pizza Stefanie eats!** I made up a big batch of dough and had each kid make their own little pizza. We then put all 16 little pizzas in the oven, in order, to bake. When done, each child got to eat the pizza they made. It was fun, and everybody had a good time. We usually get about eight parents to stay, and we had pizza, too.

We may turn out to have a crowd up here for Thanksgiving. Brenda's folks are coming. We were planning on taking an extended weekend and visit the cabin this weekend, but just couldn't break away. We had hoped to meet with Brenda's sister and her family while down there. Since we couldn't, they may come up here for Thanksgiving as they want to see Benjamin before Christmas. We also extended the invitation to Brenda's Brother and his family. They may get here as well, but since they live all the way down in Gainesville Florida, it's *iffy*.

One last thing to report. Near the end of Summer, we bought a lot down at Lake Anna where we go boating. We also picked up an old 24 foot trailer from a neighbor and had it "trucked" to our lot. So we now keep our small boat down there, and have the option of breaking away from the hustle and bustle of life. I think that it is important to be able to get away. Don't let life slip by without enjoying some of the leisure activities.

Letters to the Editor

Discontinued Software

Fm Bruce J Hutchison: I missed out on some of your now discontinued products. Knowing your position about copied/bootleg software, would you have any objection to my putting a message on CIS offering to buy used original disks and documentation from anyone who has no need of them anymore. I would also offer to trade original legitimate packages that I either never used or do not use anymore. If you have no objection and assuming someone is willing to sell/trade could I then send the original disk(s) to you for refreshing and/or updating with the appropiate fee?

Thanks for staying in the TRS-80 market. I use my Mod 4 as a king-sized tinker toy and have no interest in the overpriced MS-DOS world. Long live the Trs-80!!

Fm MISOSYS, Inc: Buying used software is a legitimate activity. You buy used machines, don't you?

Phone Numbers a la TRS-80

Fm Jim Beard: Here's my phone list for Tandy products. I would advise you to contact a couple and get what you want now, while the availability and price are right.

Powersoft, 214-733-4425; Aerocomp, 214-637-5400; MISOSYS, Inc., 703-450-4181, order line 800-647-6797; Ft. Worth Computers, 800-433-SAVE Tandy franchise; Elek-Tek Computer & Supply, 312-677-7660; Tandy National Parts, 817-870-5600; Tandy switchboard, 817-390-3700

Testing: Alpha, Beta,...

Fm Adam Rubin: May I ask your help with definitions of alpha-test, beta-test, gamma-test, and so on, as they apply to programs? Rightly or wrongly, here's how I've been interpreting them:

[no name] - Testing by original programmer(s); Alpha-test - Testing by in-house test staff; Beta-test - Testing by outside group of testers; Gamma-test - Bug reports by end users; Delta-test - (No idea); Epsilon-test - ?

Fm Ray Reyes: Adam, Delta test -- alpha testing of the revised version resulting from Gamma testing of the original version. Epsilon test -- beta testing of the above.

Fm Adam Rubin: Ray, Does that mean my first four definitions are more-or-less correct?

Fm Ray Reyes: Well, yes, except that there's really no such thing as "gamma" testing. It all stops with beta, and then the product is thrown to the wolves, whereupon the process starts all over again as bugs are reported. Only this time the version number changes.

Fm Adam Rubin: Ah Ray, I see. So for most programs, the version number changes only with public releases of the product, and other methods are used to distinguish the various beta-test releases?

Fm Hardin Brothers: Adam, Some companies (Word Perfect comes to mind as a good example) don't change version numbers with each update. Instead, they use the file date to identify which version you have. As I remember, their reasoning goes something like this: "We don't want our users to think that there's anything wrong with the software so, unless the user finds a bug, there is no reason for him/her to know that a newer version exists."

Fm Shane Dawalt: Yeah Hardin, but if a newer version exists, doesn't that imply the older version had some sort of defficency -- cosmetic as well as operational? Seems that when any piece of software is of a newer version, the company selling the software would want EVERYONE to be up-to-date. The reasoning you stated sounds to me that the company is actually saying "To keep from being bothered by a mob of people wanting updates to a newer software version, we will hide the fact that we indeed have a newer software version, unless the version fixes a bug which we consider serious."

Fm Joe Kyle-DiPietropaolo: Well Hardin, the way I heard the reasoning was:

"Gee, when we find something wrong, we should fix it right? Should we follow the same policy as others do, and only make

these corrections available once a year or so? How about if we make bug fixes continuously available to anybody who wants them or has a problem?"

"No, we can't do that - everybody else will make fun of us for having a version number that keeps changing - they'll use it to accuse us of having buggy software even though we don't have as many as they do - they just don't release the fixes except at fixed intervals."

"Well geez, what if they user really needs a fix, with those other guys they'd be screwed, but can't we offer some better way?"

How about if we announce and offer upgrades to the version number as major features are added, but offer anybody the ability to upgrade to the latest production version with the latest bug fixes whenever they want to?"

"That sounds peachy. How do we control and keep track of releases though?"

"Hey, I got it, let's use the date....

Same mechanism, slightly different attitude.

Fm Joe Kyle-DiPietropaolo: Adam, Just to show I'm not totally useless when it comes to useful suggestions, I'll throw in a serious note. During testing, be it alpha, beta or whatever, some folks will refer to testing as improving the "quality" of the software.

Not so. Fixing bugs doesn't improve quality of software, it merely decreases the number of observed defects currently present in the product. Quality cannot be "tested into" a product, it must be designed into a product, be it hardware or software.

Fm Adam Rubin: That's a good point, Joe. It would seem to me, though, that there's some relationship between the bugs in a released software product, and its "quality". Hmm... perhaps it's one of those "worst links in the chain" things.

If the design were rated at, say, 7 points out of 10 on the "quality" scale, the final product could end up with anything from 0/10, if unusable due to excessive bugs, up to 7/10, meaning no significant bugs. (Of course, if the documentation's quality is only 3/10, then the package as a whole could be no better than that.) Does that make any sense? Or should I go and read the dozens of books available on software quality first?

Fm Joe Kyle-DiPietropaolo: Well Adam, let me say that this is certainly not a point that is "accepted" by all concerned, and is strongly influenced by your interpretation/definition of the word "quality".

The argument goes something like this: Let's define quality as the inverse of the likelihood that a user will encounter one or more uncaught software defects in the field. Since you can't prove that a program is defect free (Turing machines, reduces to the halting problem, NP-complete), all we can do is find and eliminate defects through testing, and we can never prove that we just killed the last bug. You can never prove there aren't any more, but you can easily prove there is another by finding it.

Now, given two software packages of a similar nature, but different production environments. The first was not produced with the benefit of modern structured design techniques, and the latest in software engineering concepts applied throughout the entire development process, while the second one was.

For the first product, an average of some number of bugs per thousand lines of code were found during the final testing stages. For the second, an order of magnitude fewer bugs ere found. Which is the higher "quality" product? Which is likely to have more bugs left? How many bugs do I need to remove from the lower quality product to raise it to the same quality level as the other?

The metrics of software quality assurance are a whole 'nother issue.

Fm Hardin Brothers: Well Shane, I generally agree but can see some problems. Would you want to pay (even a nominal fee) for monthly upgrades if your present version did everything you ever asked of it? Or bi-weekly upgrades? Or should they keep all minor bug fixes in-house for 6 months because most of their users will never find them necessary?

I would like to see all companies publish a list of known bugs and provide upgrades for a nominal fee when the user felt he/she needed it. But I only know of one company that even claims to provide a semi-public bug list that is up-to-date (I'm talking about MS-DOS products here -- Roy is certainly an exception). Others seem to take the tact that they should stonewall all bug reports until they have a new version to release or at least a maintenance release that fixes several problems at once.

Fm Shane Dawalt: Well Hardin, I would expect that the user would have the RIGHT to hold off on purchasing monthly upgrades until the exact information on the upgrade was known. What does it fix? Why IS it fixed in the first place? That sort of thing. The user shouldn't be FORCED to buy a upgrade simply because the company decided it was time to fix it. If the company is actually going to keep the fixes in-house for a long period, they should AT LEAST acknowledge the fact that the bug was recognized and a fix was generated. Not respond, "Maybe or maybe not." to the question was the bug fixed.

Fm Hardin Brothers: But Shane, how can a software house afford to support multiple (especially many past) versions of a

product? "You have version 48.7, right? Well, boot up and tell me what the version number says -- yes, I'll wait while you go in the other room and turn on the computer <tap, tap, tap, tap tap te dum dum> Hi -- okay, what version did it say? Oh, you have to uninstall the autorun macro so you can see the boot-up screen. To do that, simply delete ARUN.MAC from your drive -- or better yet, rename it temporarily. Huh?

Okay -- exit from the program. No, don't turn off your computer, just hit F10 to exit. Then type REN ARUN.MAC ARUN.BAT. Then start the program again.

Oh -- which shell are you using? Does it let you get to a DOS prompt? You know, the letter C followed by a greater-than sign? You've never seen that? Look, do you have a system administrator at your site? Oh, he's on vacation until a week from Monday. Well, look, unless you can get to the DOS prompt, I'm afraid I can't help you because I need to know which version you are using. No, it doesn't say in the documentation, only on the disk. What happens when you press F10? Oh, the screen blanks and then the program starts over. Well, see if you can find someone to help you get a DOS prompt and then give me a call back, will you?"

Fm Shane Dawalt: Wouldn't it be so much easier Joe, to simply increment the version number when something is changed? It just seems dumb to put a number into a package which is suppose to represent it's version, then not use that version number unless it is absolutely necessary. Are they trying to hide the fact their software DOES change a lot?

I recall a big todo over on the Borland Programming B forum (BPROGB) when Turbo C 1.5 was released. Nobody could understand why Borland jumped from version 1.0 to 1.5. Borland sited the reasons for this: most importantly, every time bugs were repaired during in-house testing, they would simply update the version number ... not the file date. This seems as it should be.

Fm Joe Kyle-DiPietropaolo: Shane, The upgrade cost to WordPerfect 5.0 was the same whether you had 4.0 (three years old), 4.1 or 4.2 and any date release of those versions. WPCorp will continue to supply and support 4.2 in parallel with the new 5.0 version.The major reason I'm tossing in all this stuff is that WPCorp was specifically named as using this particular version naming convention, regardless of which they are probably **the** single most reasonable large software vendor when it comes to support and upgrades. I'd rather not see them get smeared when their behaviour doesn't warrant it.

The only reason that the file dating scheme is used is because they do indeed make **optional** upgrades available as bugs are corrected or (small) features are added. They **don't** wait (perhaps years) to fix problems or add minor enhancements as virtually all other software houses do. The version number is incremented for major enhancements.

Fm Joe Kyle-DiPietropaolo: Shane, I already explained this. Yes, incrementing the version number would be easier, but when? At the time of fix, or at the time of release? Everybody else in the word processing business does the latter (TurboC doesn't count). Easier, but not practical in the harsh world of the PC software marketplace --- unless you restrict **when** and **how often** fixes are made available.

Me, I'd rather have a new version whenever I care to ask for it, that specifically fixes a bug that keeps me from completing a project. I can't count the number of times I've heard/read a Microsoft representative say "That bug is already known and fixed in the next release." The response to "When can I have it?" is "When the next version is released."

Fm Shane Dawalt: TurboC doesn't count, Joe? I realize word processing software is much, MUCH different than compilation software. But it all boils down to the same thing ... software is software and bugs are bugs and fixes are fixes, regardless of whether it is compilation, word processing, database management, DOSes, etc. Besides, when Tandy released patches for their SuperScriptsit, they changed the version number. Seems everyone should change any version number on ANY piece of software at the time of fix because it ain't the same version as was available before the fix was placed into it. It may be a trivial change ... but it's a change to the code nonetheless.

Fm Joe Kyle-DiPietropaolo: Shane, Both Turbo C and a word processor are both software packages, and similar maintenance procedures can be applied to both. The difference is in the nature of the product, the nature of the user and marketing to that target user. User perceptions of a product's quality are just as important as the actual quality of the product itself, if a product is to survive in the software marketplace. That's the key factor being considered here.

The reason I said that Turbo C doesn't count is that a compiler is aimed at an entirely different market than a word processor would be. User perceptions are different. Programers are used to working around compiler problems, ask any Microsoft or Lattice 'C' user that writes large or complex programs. In many cases, they may choose to continue to work with an existing version whose behaviour they understand rather than move to a new version. Moving to the new version would require re-learning the new patterns that generate broken code, and the risk of breaking code that works with the old version. That's the cost - do the features in the new version warrant the change? Each case must be considered independently.

Back to users perceptions - Sure, Tandy incremented the version number for SuperSCRIPSIT each time they made a change. Were they released one at a time? No - they were batched together and released once every six months or once a year. Even that didn't help them, SuperSCRIPSIT still got a reputation as being a piece of junk. Why? Because it **deserved** it - it was a piece of junk.

Fm Shane Dawalt: I will certainly agree with you on SuperScripsit being a piece of junk, Joe. We both know that is true. And I suppose your reasonings for this entire thread do make sense. I suppose I was placing all software into it's group and not making note of the fact that different types of people will be using the software in different ways.

I can see your point about programmers working around compiler problems. It just seems more important to use a compiler which is more "stable" in terms of less bugs than simply working around a known broken compiler. Of course, I don't get down and dirty with a compiler bug. If I find one, I report it and the project gets delayed until a fix comes out. Unfortunately, companies can't do this.

Fm Joe Kyle-DiPietropaolo: Shane, In a world full of totally rational people, many decisions could be made based on doing the "right" thing, as they are now, but with the advantage that "right" could be defined quite a bit differently...

RE: using a "compiler wich is more 'stable' in terms of less bugs than simply working around a known broken compiler". Given that choice, I'd pick the "fixed" compiler too.

Unfortunately, those aren't the two options you have, except in rare instances. The choice is typically between the older, better understood product, and a newer roduct that has as many or more bugs in unknown locations. The ones you found in the old version may or may not be fixed, or may have merely been moved.

This brings up an interesting discussion I read recently, I think in the latest issue of *Byte*. An engineer developing a safety-related product was discussed. The engineer picked the Intel 8008 microprocessor product, used in a dual redundant CPU design for a control application.

The 8008 hasn't been in production for years, and is basically the second microprocessor ever developed. It has an address space of 16K (bytes) and an internal hardware stack limited to eight levels. We're talking antique here. To ensure supply, he made a one-time buy for an estimated ten year sales and repair period. Why? Because he understood the cpu, its limitations and bugs. He felt that there weren't any surprises left to find.

Fm Adam Rubin: Joe, perhaps the term "software quality" is more clearly defined than I'd realized. It seems to me that there's more to the quality (in the traditional sense) of a piece of software than the number and seriousness of bugs found. For example, I couldn't consider a word processor with neither bugs nor an "insert" function very high on the quality scale.

Maybe I'd better read some of those dozens of books on software quality before I start getting confused here.

Model 4 speedup

Fm Shane Dawalt: Roy, By now, you are probably tired at hearing how nice the TMQs look. Well, this last issue is no exception. As I read from the front to the back cover at 3AM last night (night person you know) many things caught my eye. One that almost pulled them out of their sockets was the reprint about speeding up the model 4 by bypassing the incorrectly programmed PAL chip. I had pondered this chip (U3) before. Wondering why Tandy had done this weird thing and I had hypothesized that to get full speed, all I need to do is bend pin 7 up. Unfortunately, I wasn't in the 'experimental mood' so this was never done. The article in TMQ reminded me about that thought plus gave me confirmation on my suspicions. It's 1:30AM and just got done buttoning up the M4. I ran my CPU speed test program & now my M4 is whisteling away at 4.05 MHz. (Before the hardware change, it was at a slugish 3.31 MHz.)

I would like to suggest that pin 7 (the pin which is bent up) be terminated in a 1K ohm resistor tied to +5V. This keeps the possiblity of noise on this line from thrashing about the logic threshold & causing spurious operation. There's a capacitor near the chip with a 5 volt pad. Quite handy ... you'd almost think it was put there on purpose. On my system, it's C98. Rats, wanted to fetch the board's serial number while I was in there and forgot it. Anyway, thanks for printing that article.

Fm Joe Kyle-DiPietropaolo: Shane, Not that this hasn't been discussed here [LDOS forum] before, but note that the Z80 (not just the Z800) has more stringent memory cycle requirements during the M1 memory cycle. Some machines may not work (or worse, have rare, almost indetectable pattern sensitive memory fetch errors) with the M1 wait removed.

Fm Shane Dawalt: Joe, I am fully aware of the Z80 op-code fetch requirements. In fact, that's why I hesitated when I first decided it was pin 7 of U3 generating the slower clock speed. Since it was documented to actually work on some machines, I decided there was a better probability it would work on my machine. Therefore, I modifed the hardware. As I stated, I ran my memory tester (MEMTEST) and decided data storage wasn't affected by the hardware modification. Just to see if things stayed stable, I left the computer run for 8 hours and retested the speed and memory again. Again, no problems. Of course, MEMTEST doesn't test execution problems yet [that's in the works], but at least the system didn't crash totally.

I'll certainly be on the lookout for errors at any rate. And, ya know, if you think about it, why does that WAIT need to be in there? If the RAM is speedy enough, which I'm almost sure Tandy designed for that, then the only other problem it could be is support circuits. Maybe there are some PALs which cannot run as fast as Tandy wanted them to during the design. I don't have my PAL info handy so I cannot lookup the max speed of the PALs used. I'm sure Tandy used all LS devices

for TTL chips on the discrete logic. Those are good to around 10-25 MHz depending on the chip complexity. If I start having errors, I think I'll start looking at the discrete logic.

Fm Joe Kyle-DiPietropaolo: Shane, I'm certainly not saying that you are sure to have a problem, only that the Z800 may not have been the only reason for the delay. Tandy had a heck of a time getting some of the early CPU boards to run reliably, and there are a number of errors out there. Another one is that some PALs are coded such that you can't run code reliably within one of the alternate banks. Data storage and retrieval works fine, but code won't run. M1 is the only real difference there. Whether it is a lack of M1 waits on those machines in the back banks or not I don't know.

Fm Shane Dawalt: I would be tempted to blame a CAS/RAS/MUX generation problem for those M4s who cannot successfully execute code in the banks. Although I haven't tried it yet, my M4 used to run back bank apps ok.

1988 Calendar

Fm Joe Kyle-DiPietropaolo: I found this and thought you might find it handy. Aggie Hacker's Hexadecimal Gig'Em Texan Calander for 1988

SUNNDI

JA	FE	MA	AY	ME	CH	CH	AW	SE	AW	NO	DE
3	7	6	3	1	5	3	7	4	2	6	4
A	E	D	A	8	C	A	E	B	9	D	B
11	15	14	11	F	13	11	15	12	10	14	12
18	1C	1B	18	16	1A	18	1C	19	17	1B	19
1F				1D		1F			1E		

MUNNDI

JA	FE	MA	AY	ME	CH	CH	AW	SE	AW	NO	DE
4	1	7	4	2	6	4	1	5	3	7	5
B	8	E	B	9	D	B	8	C	A	E	C
12	F	15	12	10	14	12	F	13	11	15	13
19	16	1C	19	17	1B	19	16	1A	18	1C	1A
	1D			1E			1D		1F		

TEWSDI

JA	FE	MA	AY	ME	CH	CH	AW	SE	AW	NO	DE
5	2	1	5	3	7	5	2	6	4	1	6
C	9	8	C	A	E	C	9	D	B	8	D
13	10	F	13	11	15	13	10	14	12	F	14
1A	17	16	1A	18	1C	1A	17	1B	19	16	1B
		1D		1F			1E			1D	

WEDDSDI

JA	FE	MA	AY	ME	CH	CH	AW	SE	AW	NO	DE
6	3	2	6	4	1	6	3	7	5	2	7
D	A	9	D	B	8	D	A	E	C	9	E
14	11	10	14	12	F	14	11	15	13	10	15
1B	18	17	1B	19	16	1B	18	1C	1A	17	1C
		1E			1D		1F			1E	

THIRSTY

JA	FE	MA	AY	ME	CH	CH	AW	SE	AW	NO	DE
7	4	3	7	5	2	7	4	1	6	3	1
E	B	A	E	C	9	E	B	8	D	A	8
15	12	11	15	13	10	15	12	F	14	11	F
1C	19	18	1C	1A	17	1C	19	16	1B	18	16
		1F			1E			1D			1D

FRAWDDI

JA	FE	MA	AY	ME	CH	CH	AW	SE	AW	NO	DE
1	5	4	1	6	3	1	5	2	7	4	2
8	C	B	8	D	A	8	C	9	E	B	9
F	13	12	F	14	11	F	13	10	15	12	10
16	1A	19	16	1B	18	16	1A	17	1C	19	17
1D			1D			1D		1E			1E

SATTADI

JA	FE	MA	AY	ME	CH	CH	AW	SE	AW	NO	DE
2	6	5	2	7	4	2	6	3	1	5	3
9	D	C	9	E	B	9	D	A	8	C	A
10	14	13	10	15	12	10	14	11	F	13	11
17	1B	1A	17	1C	19	17	1B	18	16	1A	18
1E			1E			1E			1D		1F

Shrink Wrap

Fm Shane Dawalt: You'd probably know, I've often wondered about that shrink wrap. Do you have machines in-house to wrap the product and shrink the wrap or is the product boxed up in some way and taken to a place that can shrink wrap stuff? (This is the kind of stuff I wonder about at 3 in the morning ... Egads.)

Fm MISOSYS, Inc: We have a shrink wrap machine here. Actually, that's two things. One is called an "L" sealer and the other is a shrink oven which has a conveyer belt running through a metal housing (oven) which has plenty of heating coils in it. The oven runs off of a 230V 30A line. The "L" sealer is also used to poly wrap *THE MISOSYS QUARTERLY*.

Fm Shane Dawalt: Oh. You wrap the product, seal it with the "L" sealer, put it on the belt & bake it rare. (I bet you don't let that oven run too long between wrapping parties do ya?!!)

Fm MISOSYS, Inc: Actually, I rarely use the oven except when I go to batch production. It takes too much power and generates too much heat. Of course, it does feel good in the winter when its quite chilly down in the dungeon.

MAXDOS 6.3 for the MAX-80

Fm Bryan Headley: Roy, I see you forgot to mention MAXDOS in the last issue. If you don't mind, I'd like to upload a short text description of MAXDOS into one of the data libraries. I have had good sales, selling to the CP/M-based Max-80 community. I'd like to reach the LDOS/6.3 crowd now.

Also, in this issue there's talk of modifying Model III HD Profile+ to the Model 4. Why, when Small made a Model IV version already? The Model 4 has a page to itself in the new catalog, with all the Model 4 software still available. Profile should still be there... (or did I miss something?)

Fm MISOSYS, Inc: We were pressed for time in getting that issue together. Didn't you read that? Apparently you also misread the discussion a la Profile III. I printed the request from a customer. My response pointed to the impracticality of the job. You read in haste.

Fm Bryan Headley: MaxDos 6.3 is a 100% TRSDOS-6 compatible port for the Max-80. Due to incompatibilities with the Model 4 vs. Max-80, you might be looking at 95% compatibility. Specifically, these involve things like the character sets (inverse video is handled by a filter), playing with banked memory through hardware, and reading the keyboard matrix.

A substantial portion of software works. Exceptions are VidText, Scripsit Pro, and Double Duty (for the reasons I outlined above).

Here's the deal. Since we draw on the original 6.3 (for utilities, BASIC, and system overlays before they are patched), we have to require for legal reasons that you own a TRSDOS 6.3 (this way, MISOSYS & LSI gets paid). Send that 6.3, two blank 5-1/4s and a check for $10 to me, and I'll send you a MaxDos (you get the TRSDOS 6.3 back, natch). I'm at:

Bryan W. Headley
1609 Homedale # 2008
Fort Worth, TX 76112

My New Hardware

Fm Theodore Masterton: So I could not wait forever. My 4p now has a neighbor; a Leading Edge D2 with Econo Monitor, Zuckerboard monographics, Seagate ST-225 (no snickers!) and a WD 1003 WAH controller. Short of an Out and Out clone, it was a pretty good deal; besides, we use Leading Edges at the agency and I have GOT to flatten that learning curve anyway possible.

The best fun was buying and installing the HD subsystem all by myself. If it wasn't for forums I do not know how I would have learned enough to decide on the equipment, let alone find out that I could increase the performance of the Chevy Six like Seagate 50% just by dropping from an interleave of 3 to 2! The HD now does quicker transfers at 65ms than our 3 to 1 interleaved ST-251's at 40ms.

So it sure was a great 4 month shopping spree/ordeal. I must have 40000 cubic feet of computer shoppers at me feet here in my office. Ironically, my copy of *THE MISOSYS QUARTERLY* and *Computer News 80* both arrived the same day as the 286. Kinda made me feel funny. Sorta like Jimmy Swaggert caught with his lens cap off.

TMQ Artwork

Fm Bryan Headley: We are **very** curious about your graphics art subscription service (the one you subscribe to and use in the *Quarterly*). We have customers wanting icons etc for PageMaker. Could you tell me whom/how much/phone number for these guys? We'd be happy to refer customers to the guy. Also, you might want to dabble with a killer package called Arts & Letters. It works in Windows; outputs TIFF. You got 120 various drawings with the package; you can place, rotate, stick shadows on, etc. graphics. Since output is TIFF, Ventura and PageMaker read without a hassle. Very nice! Are you still using Word to do the Q?

Fm MISOSYS, Inc: The company you want to know about is Dynamic Graphics, Inc., 6000 N. Forest Park Dr., Peoria, IL 61614-3592. 800-255-8800. I subscribe to a service called "Clipper". They also have other services. Clipper runs about $350/year for 12 monthly mailings. The service is excellent and quite worthwhile.

More Market Research feedback

Fm L. R. Boatman: Hello Roy, Concerning your request for Market Research on a 20 Meg Hard Drive and price as you stated in The Quarterly ($495.00) you have my interest and I will look forward to the availability of the system.

A few months ago I had a problem with the XLR8er and Prowam. You hit the nail on the head with your reply. Seems that I failed to install the faster RAM (150ns) in banks 1 and 2 of the 128K. I now load PRO-WAM into Bank 10 "PROWAM (BANK=10)" and have not encountered any further problems. The Ram is not available in 150ns in this area. Would you please tell me of a third party vendor who handles these chips and has directions for their installation on the Model 4P, also the price if known.

The following is a short narrative on my use of the Model 4 (I have two (1 with XLR8er and 1 with 128K, converted Model III). I use the machines during three months of the year to do taxes (JAN-APR). The addition of the XLR8er on the DS Disk Drive Model 4 and the use of MultiPlan reduced the amount of time the customer is serviced in half. I also do payrolls and book keeping for a number of customers. My children and I also do a lot of our school work (Learning is a never ending process), I keep a mailing list and copies of the clients taxes (1000+) and a Data Base. The use of PRO-WAM has and is a valuable tool in the development of both.

The above information is sent in reference to the survey or Market Research concerning the HD for the Model 4P. I don't know how much longer the Tax software people are going to continue to support the Model 4P, but with your proposed package and with it being reusable in a PC environment, it has my interest.

Fm Jane A. Layman: This letter is in response to your market research query in both the cooperative mailing and Volume II.iv of *The MISOSYS Quarterly*. I would be very interested in the external 20 Megabyte hard drive you are contemplating marketing particularly as it will come with MISOSYS' software. For the last 5 and 1/2 years I have been using a 5 meg Radio Shack drive with my vintage Model 4. While the drive (and the Model 4) have performed flawlessly all this time, one has to consider that the drive cannot go on forever. Also, with only a 5 meg drive and two operating systems installed, my hard disk space is somewhat at a premium.

Another place where I find space at a premium in my system is in high and low memory. In addition to a hard disk, I also purchased an XLR8er Board (from H.I. Tech just days before MISOSYS announced it would distribute the board). I have been very satisfied with the board but would appreciate improved versions of its accompanying software. (I have been using HIBANKS with FIXALL/FLT, which for some of my operations has been a distinct improvement over FIXBANK.) I am not particularly sanguine, however, about being able to discard FIXALL/FLT even with the revised version of HIBANKS.

My Model 4 is among the first to come off Radio Shack's assembly line. It is a non-gate-array version that originally came with only one floppy drive. Over a period of time I had Radio Shack install the extra 64K, an RS232 board, and the 2nd floppy drive. Six months ago I had the XLR8er Board added and the 128K of 200ns chips replaced with faster ones. Aside from occasional cleaning and realignment of the floppy disk drives, my computer has never seen the inside of a repair shop.

I can hardly complain that the hardware Radio Shack sold me was unreliable. My keyboard, however, is not up to XLR8er speed. After reading in the Model III documentation that came with the XLR8er Board that I could treat the LDOS version of FIXALL/FLT as optional provided my Model 4 keyboard did not behave erratically under LDOS at the XLR8er's top speed, I was disappointed to discover that my vintage Model 4 keyboard behaved very erratically indeed. That means that under LDOS I must choose between running at top speed (with FIXALL) or using KI4/DVR. Is there any chance that the FIXALL/FLT function could be incorporated into KI4/DVR, e.g., in place of the Delay and Rate options? I cannot guess at the feasibility of the programming involved, but sales of the Hardware Interface Kit reported in TMQ suggest this might be a strategic product enhancement.

As long as I'm this far advanced on a wish list, a final comment on the XLR8er software. Ramdisks that installed in the x.3 format would be very nice. Thus far my extremely uninformed attempts at "hacking" along these lines have not met with any success. The reality is they aren't likely to in the foreseeable future.

Last but not least, I would like to offer a rebuttal to Charles A. Ainsworth's Story #3 on p. 16 of Vol II.iv of TMQ re the friend who purchased VisiCalc from Radio Shack and found it only ran under the system (TRSDOS 6.01) on the distribution disk. I, too, purchased version 02.09.02 of VisiCalc on a 6.01 system disk at close-out sale prices. I have used and continue to use the program frequently under TRSDOS 6.2.x and now LS-DOS 6.3. VisiCalc also takes kindly to the XLR8er Board and the optional patches to LS-DOS related to the Hitachi chip's math instructions. If there's a conflict between VisiCalc and any version of TRSDOS/LS-DOS, I have yet to find it.

In closing I will say that I very much enjoy the coverage of the XLR8er Board in *The MISOSYS Quarterly* and will look forward to further coverage on this and other topics.

Fm MISOSYS, Inc: I believe that Michel Houde's new software interface will definitely impact your needed low memory reduction in usage. The code added to enable the XL8er's added memory banks takes up 121 bytes. Using the revised ERAMDISK adds 116 bytes for the page extended memory manager plus 96 bytes for the RAM driver. Rex's HIBANKS module at 205 bytes was an improved rewrite of the original FIXBANK module at 246 bytes; I don't have room to load RAMDISK with my hard disk driver in either of those cases. I'd say that Michel's efforts will be useful.

Be kind to beginners...

Fm Rema Lou Brown: Dear Roy, I must praise you for the usefulness and the increasing quality of your publication. The Summer 1988 issue of *The MISOSYS Quarterly* arrived today. I had just finished re-reading my four issues of Volume 2 of *TMQ*. The more I learn, the more I discover I missed the first time I read these. (My order is enclosed for the back issues of Vol. 1 of *TMQ*.)

I'm not a hacker, probably for the same reason Picasso's wife didn't paint. Our family life revolves around computers. In 1961 we decided food really was a necessity, so my husband quit teaching high school math to become a programmer analyst. He is now a manager for Unisys nee Sperry. He, our two sons, and one of our daughters-in-law work in the JSC NASA environment. They converse in Basic, Cobal, Fortran, Pascal and, now, Ada. As a former high school English teacher, I wish they were as fluent in English!

My first computer was the TRS 80 III. My first applications were Radio Shack's *Scripsit*, *Time Manager*, and *Project Manager*. Tandy exchanged my Mod III for a Mod 4 (1069), and I entered a new world. Yes, you read that right! I have 128K RAM, two SSDD 5 1/4" floppies, a 10 meg hard disk from Hard Drive Specialists, and an Epson RX 80 printer. I divided my HD into 4 partitions: 3 LS-DOS 6.3, and 1 LDOS 5.3. I added two exterior DSDD floppy drives last week.

I did try to learn how to use and how to maintain my computer at my local RS Computer Center. RS offered some business training courses, *Profile 4*, *Visicalc*, etc., but these seemed designed to sell additional applications not to teach me how to use those I already owned. I could have taken computer courses at our local university or local junior colleges, but I did not want to become a technician or a language specialist.

I had to opt for self-education with some coaching from the family. After I discovered its existence, I subscribed to *TRS 80*. When Tandy let it go belly-up, it passed me to *Micro 80*. I dropped that when it became a tub-thumper for RS's MS-DOS machines and a vehicle to expound Tandy corporate policy. Thank you for creating *The MISOSYS Quarterly*! You have helped me see uses for my computer that I didn't know were possible.

I do not have the knowledge, the patience or the interest to write my own programs, but I'm a willing customer for applications that allow me to do what I want to do. I read each manual and "Readme" file carefully before I ever use the application! It is especially helpful if the author writes in clear English, does not presuppose the reader knows the jargon, and includes an index as well as a table of contents in a manual. I highlight operating information with yellow and warnings with orange as I read so I can find those important points later.

After I added my hard drive, I had to retire *TiMgr* and *PrjMgr* because the RSCC folk couldn't help me overcome the fact neither program recognized the HD as Drive 1. They sold me *Deskmate* 4 to replace *TMgr*. In addition to *DM* 4's calendar, I thought it would give me a filer, a communications package, a dialer, and a simple spreadsheet. It was the worst purchase of my life.

I became disenchanted with *DM 4*'s limitations in a matter of weeks. I didn't expect a Cadillac, but I did hope for a Cheverolet. It turned out to be two steps below a Hyundai. (No disrespect to Hyundai intended.) The end came unexpectedly. I had to change the *DM* 4 calendar frequently. I would edit it in its sort mode because it was more efficient for planning and scheduling a sequential series of activities. After one editing session, I accidentally edited the calendar directly from the sort mode without first returning to unsort, then returning to the day page, and then returning to the main menu. Everything turned to manure. The calendar entries were pure chaos and every free granule on the HD partition was filled with crud. No where in the manual did it warn of this danger. Thank heavens for backups!

I then began a campaign to resuscitate *TMgr* and *PrjMgr*. I Suffered third degree frustration trying to discuss possible solutions to my problems with our RS CC "experts." I finally phoned Ft. Worth. It took eight different long distance calls to discover someone at Tandy who knew that RS had a version of both *TMgr* and *PrjMgr* that worked with a HD set-up. Goodbye, *DM 4*!

If local RS CC people knew their own products I could have saved money, time, and anguish!. Surely RS could create a mailing list of customers from a data file based on product registration cards, and notify us of the availability of new versions of their software. They knew how to contact us to demand that we update our TRSDOS. Oh, well, someday I'll buy a new computer -- BUT IT WON'T BE FROM RS!

I use ProSoft's *Allwrite* and *Dotwriter*, Cornucopia's *Electric Webster*, PowerSoft's *Super Utility* and *Tool Belt*, and MISOSYS' *PRO-WAM* in Mod 4, and RS's *TMgr* and *PrjMgr* and Powersoft's Tool Box in Mod III. I've just purchased PowerSoft's *Back/Rest* so I can backup individual Mod III data files that are too large for a floppy disk. I've also acquired a modem and, after I'm certain of what I'm doing with it, I'll probably signup with CompuServe. My wish list includes *Little Brother* (order enclosed) and a good thesaurus that's compatible with *Allwrite*.

I began this letter to respond to the comments of some of your Compuserve correspondents concerning the "stupidity" of some TRS 80 users (*TMQ*, Vol. 2, iii and iv). I know you've experienced a terrible pain in the posterior over the date business and the public's confusion of your LDOS 5.3 with LSI's LSDOS 6.3. The antagonism this confusion has generated, yours and ours, was misdirected. Tandy should be the target, not you, and not their customers!

It's easy for your correspondents to be critical of others for not being as "in the know" or as "informed" as they. Without having to explicitly state it, each has managed to point out how superior he is to those whom he criticizes. (Did anyone else notice that they directed their criticism almost exclusively at women? Ah, hem!)

Your correspondents need to remember that they weren't born knowing what they now believe they know. They learned through experience, through school, or through work. Surely they, too, asked "Stupid" Questions of someone sometime somewhere along the way! They should be glad that no one disparaged them the way in which they now disparage others.

My best wishes to Brenda, Stefanie, Stacey and Benjamin Charles! Ah, yes, newborns are wonderful even though there are 2:00 am feedings, colic and wet diapers. Of course my memory may be faulty after twenty-seven years! Four years ago I learned that a newborn is best when its your granddaughter. Now we're expecting again -- our second grandchild is due in December.

Fm MISOSYS, Inc: I suppose all of us are guilty at one time or another of being unable to revert to novice level. It takes a great deal of skill to teach, and perhaps a certain kind of personality. There are those of us who have spent years developing an "expert" skill level but have a difficult time in relating to those newcomers with the questions we have long ago forgotten. Perhaps those of us who you tried to reach in your letter will reflect on what you said, and come to grips with this problem [should that have been "whom you tried"?].

Now as far as grandparents are concerned, according to my mother-in-law as related by my wife, nothing brings a bigger smile to a grandparent than seeing their now-grown-up-child struggling with the same problems in dealing with the new offspring as they dealt with with us when we were little. Bill Cosby's TV dad raises that issue a number of times on the Cosby TV show. I see it at home. I'm forever telling my children to "turn off the light when they leave their room", and close the door after you go outside", things my dad had to constantly remind me of. And how often do I have to remind them to "brush your teeth". Just wait until I get to the years where they start telling me, "Dad, you don't know anything"!

Fm Mark Allen Reed: Dear Mr. and Mrs. Soltoff, Congratulations on the birth of your son! I once read that miracles wouldn't be miracles if they happened every day; but children are being born somewhere on the planet every minute, and that doesn't make birth any less miraculous. I'm sure that all three of your children will continue to bring you happiness and joy in the coming years.

I just received *The MISOSYS Quarterly*, volume III.i; as usual, you did an excellent job. On page 18, Carol Welcomb described some problems she has been having with OKIDATA about her Okimate 10 printer. Since her problems seemed similar to the ones I have experienced in the past (see *The MISOSYS Quarterly*, volume II.iii, page 79), I wrote to her with some suggestions and explanations. I am enclosing a copy of the letter; you may want to publish it if you feel it would help some of your other readers out of similar predicaments.

Everyone in volume III.i is talking about your "*Required Reading*" flyer, and I'm a little puzzled as to why I didn't receive it. I'm a *MISOSYS Quarterly* subscriber and a regular customer. The only reason I can come up with is that I'm not a registered owner of LS-DOS 6.3. I own two TRS-80 Model 4D's, and each came with copies of TRS-DOS 6.2 and LS-DOS 6.3. TRS-DOS had a registration card, but LS-DOS did not. I am writing my serial and customer service numbers at the close of this letter. Would you please add me to your LS-DOS 6.3 data base:

Fm MISOSYS, Inc: As you surmised, the "*Required Reading*" flyer was mailed specifically to the LSI data base of registered LS-DOS 6.3 users. We didn't want to combine that list with our normal data base. But I'll be sure to get your ID numbers into that data base now. I would expect to have another mailing as soon as I have a firm announcement on our hard drive package. You'll get that one.

Update on printer woes

Fm Mark Allen Reed to Carol L. Welcomb: I was sorry to read (in *The MISOSYS Quarterly*, volume III.i) that you, too, are having problems with an OKIDATA printer. I'm not an expert on OKIDATA's product line, but in my experiences with that company I picked up quite a bit of information, and I think I can help you out of your predicament.

OKIDATA sells Okimate printers as two components. The first -- which most people think of as "the printer" -- is the actual mechanical unit that transfers ink onto paper. The second -- the "Plug 'n' Print" module -- is the brain of the printer which allows it to communicate with the computer.

The Okimate is useless without the correct "Plug 'n' Print" module, in the same way that your Model 4P would be useless without its video display and keyboard.

I'm not familiar with the Commodore 64, but your description of its "Plug 'n' Print" socket makes me think it drives printers through a serial port. For that reason, the "Plug 'n' Print" module you already have will not work with your 4P. You need a "Plug 'n' Print" which supports your 4P's Centronics parallel port. I don't think OKIDATA makes "Plug 'n' Print" modules for TRS-80 computers -- the rest of the world seems to think that TRS-80 computers don't exist any more -- but OKIDATA's "Plug 'n' Print" module for the IBM PC should work without problems. I tested that theory by unplugging my Okimate 20 from a Tandy 1000 SX and reconnecting it to a TRS-80 Model 4D. Everything worked perfectly. The only

adjustment I had to make was to flip a DIP switch to enable automatic linefeeds after carriage returns -- but that's nothing unusual. You will have to use Tandy's printer cable instead of the one that comes with the "Plug 'n' Print" module; I doubt that the computer end of an IBM printer cable will fit into a TRS-80.

I think you can order "Plug 'n' Print" modules from OKIDATA Supplies (P.O. Box 573, Chester, NY 10918). Their telephone number is 1-800-524-8338. If you would rather look for a module locally, you can call 1-800-OKIDATA for the name of your nearest OKIDATA dealer; their information isn't very up-to-date, but its better than nothing. Wherever you buy the module, expect to pay somewhere around a hundred dollars. And whatever you do, don't let someone give you an IBM PCjr module by mistake! That happened to me when I first bought my printer. You need an Okimate 10 "Plug 'n' Print" for an IBM PC with a parallel printer port.

OKIDATA makes good printers, but dealing with the company is an unpleasant experience. I hope this letter has helped solve some of the mystery surrounding the Okimate 10. If you have any further questions, I would be happy to try to answer them.

P.S.: I am sending a copy of this letter to *The MISOSYS Quarterly*. When you solve the last of your Okimate problems, you may want to send them an account of the whole incident. It would be educational and might help other people with similar problems.

Fm Carol L. Welcomb to Mark Reed: Dear Mark, I thank you very much for your letter concerning the Okimate 10 printer. I can tell you exactly what has happened and what I've decided to do.

Firstly, I finally got the Okidata people to respond to my detailed letter about the problem. They were quite rude, I feel, as they pointed out the only computer that the Okimate 10 will work with is a Commodore. The Okimate 10 was designed specifically for the Commodore 64 (I didn't know this at the time nor did any of the literature on the Okimate 10 confirm this).

I assumed that when the owner's manual stated that when you decide to upgrade your system, simply upgrade your "Plug 'N Print" module. The Okidata people let me know in no uncertain terms that it was meant only for a Commodore 64.

I have a friend who now owns the printer I was given. She has a Commodore 64. I do understand that the Okimate 20 will work with my 4P, if I used an IBM module. I, however, will never own an Okidata/Okimate printer, even if it is given to me!

I appreciate your time and trouble in coming to my aid. I am not that great at knowing the language(s) of this old 4P yet, but I have a vast background in electronic work. Being that I am permanently disabled now and unable to work, I have the time necessary to become great friends with this machine.

I have a DWP-220, which is a great daisy wheel printer. I plan on getting a Star Micronics (with IBM/Epson modes) in the very near future. I have learned a great deal about the Okidata 10, but there was no practical way to hook it to the 4P. I feel good about going with the Star Micronics, for it has a wide carriage.

Again, thank you for your concern and your time.

Some basic questions ...

Fm James L.Lopez: Dear Roy, First let me thank you and Congratulate you for your article "Getting into Computer Math" in Volume III.i. Although I already have a smattering of your subject matter in my head this was such a good presentation that I am recommending it to one of my grandsons who is Computer Minded. I learned something too. I am glad you are going to continue this subject.

I have a TRS-80 Model 4, Model 26-1069, Ser. #0038747 with 128K RAM. And am ordering an XLR8er, for the expanded RAM memory. As per your request I took off the top of my Computer and looked for the revision of the motherboard. Since I am not all that knowledgeable about Computer nomenclature, I deduced that the board on the back side of the Computer was the Motherboard and I believe what you wanted was: 8769296 REV A. Below that appeared the numbers 700219.

As I looked at this board I remembered the remarks in Volume II iv, page 88, from Mark Vandemeulebroucke so I made some notes about my Computer: In socket U72 there is a PAL16L8CN chip and in socket U71 there is a SN74LS245N chip, and a yellow wire W3 runs from U48 to U52. Does this mean my Computer can't use the XLR8er?

Will the XLR8er (what do those letters mean ?) give me improvement with my word Processor Programs, Basic Programs, data files? Would I be able to fit a Grafyx Solution board in there also?

Now let's get some things straight here. I retired after 36 years with Exxon Overseas. My experience is with Training, Industrial Safety, Fire Protection, Security, and I cut my teeth, for over 20 years, on Industrial Instrumentation. After retiring I was an Instructor in the Instrument Technology Department at San Jacinto Junior College here in Pasadena for 5 years. I am what you could call an expert in my field. My next birthday I will be 74. I am writing a book "Our Aruba Story" and have not had the time to do much more than write with this Computer since I obtained it 6 years ago.

I have done a little with Basic in order to build up some Address and Labeling Programs. I subscribe to "Code Works" Magazine and have learned something about Basic. I have a Lazy Writer Word Processor "Piggy-backed" on a LSDOS63L System Disk which I use as well as a Superscripsit Word Processor. Both word Processors have their good points and I use one to complement the other.

A friend of mine replaced the 2 Single Sided Disk Drives with Double Sided Disk Drives in my Computer about six months ago. Otherwise my knowledge about what goes on inside the Computer is somewhat limited. I think it is Stupid not to ask questions (even if they appear Dumb or Stupid to you). How else am I to learn something I should know? And as a customer I know you will be patient and try to give me answers without getting bent out of shape. This was what I tried to do in my classrooms.

A copy of "Mod-4 By Jack" written by Jack and Marie Klein, Crest Software, 2132 Crestview Drive, Durango, Colorado 81301 has saved my hide many times. Explanations in this 6" x 9" Owner's Manual type of publication is certainly easier and more helpful than the original received with this Computer. However Jack only "touches" on some material because it is rather complicated.

Volumes II.iv and III.iv of "*The MISOSYS Quarterly*" have been received and I have been pleased with what I have seen so far. What does MISOSYS stand for? However I have noticed that most of the notes seem to be by and for people who have considerable experience with programming which is out of my league. Nevertheless I remember when I first started learning how to bowl I found that if I played with more experienced Bowlers and even Scratch Bowlers (when they would deign to play with me) I learned to play a better brand of Bowling. So maybe I will learn something with your help.

I notice that most, if not all, of the LRL entries, as seen in the Directory when you have an LSDOS63L System Disk in Drive "0" and are at LS-DOS Ready and type DIR (S,I,All), are 256. a) Why not 255 or 257 ? I have seen some reading 64 or 1 and wondered whether I should change them. b) What happens if you do change them to 256 for example! c) Is it possible that some mistake in operation would change them? d) Does this reserve a space of 256 bytes for this file?

I have noticed that, with an LSDOS63L SYSTEM disk, in Drive 0, if you are at LS-DOS Ready and you type: LIST MOVE/CTL or any entry which ends in CTL you will see the contents of that file. What should appear in Move/Ctl - what is it supposed to do? Mine seems to have picked up the partial contents of a Document.

If you type LIST DOS/HLP on my computer you get the first half of the file with white letters on black and the last half reversed (black letters on white). Is there any way you can make this file all black on white or white on black? Also this file, Dos/Hlp, appears on the screen with no separation between words and apparently contains some sort of code. Why no separation between words?

Lazy Writer will give you the Checksum of a file. What is Checksum?

QExactly what steps do you go through to apply a "Patch"? I notice this procedure is mentioned rather frequently in your discussions and it appears to me that an awful lot of patches get applied. QIf the program was designed for a specific Computer why would a "Patch" have to be applied ? I can understand if the said program is being adapted to work with an different Computer that you might have to apply a "Patch" to make it work.

Perhaps my questions will give you some ideas for the type of articles that are needed. After all I rather imagine there are a flock of customers or potential customers out here with me who are in just about the same boat I am. Weak on Computer knowledge, but willing to learn.

I have more questions, but will have to close and get this in the mail. Thank you for your kind attention.

Fm MISOSYS, Inc: Your 26-1069 Rev A non-gate array machine cannot use both the XLR8er and a hires graphics board. Since your board is an early one, I can't be positive that the XLR8er will work. If it doesn't, we will refund your money. Incidentally, I believe H.I.Tech chose "XLR8" as a moniker for the board since you prounce it "accelerate". The XLR8er makes your machine run faster; thus, it will do that for every program you run. You cannot presently fit a graphics board in there in addition to an XLR8er board. The problem is a physical one; your 26-1069 cannot work the XLR8er with a connecting cable longer than 3.5". The graphics board installation prevents the mounting of the XLR8er. I currently have a Model 4 Graphix Solution board which I am trying to install into a 26-1069 Rev C along with an XLR8er board. If I succeed, I will advise folks in TMQ.

I have published that answer before. See TMQ II.iv (Spring 1988) page 19. For those who missed it, MISOSYS is pronounced "MY-SEW-SIS"; it is an acronym for Microcomputer Software Systems. I originally wanted to use the three letters, "MSS", but Michael Shrayer Software beat me to the punch. How many remember him? Electric Pencil, hint hint.

The LRL can be any number from 1 to 256. That's all that the DOS supports. Most folks, if not most high level languages, use a fixed length of 256, which is the size of one sector, and do their own blocking/deblocking. The latter terms are used to denote the operation of working with a fixed record length less than 256 characters when disk I/O has to be accomplished. The DOS supports its own read/write blocked record I/O of LRL between 1-256 and does its own blocking/deblocking. For some reason, probably rooted in the failure of the original Model I TRSDOS to support blocked

record I/O, most folks don't use it. Don't change the LRL of a file unless you know why you are doing it. If its other than 256, the program which created the file and the one which needs to operate on it most likely needs to utilize that LRL. Of course, since the program which OPENs a file declares the LRL at open time and the one in the directory is not used for other than reporting the number of records, it probably would cause no error to occur if you "zapped" the directory.

I haven't the faintest idea what MOVE/CTL does or is. It is not a file provided with the DOS or any MISOSYS product. Sounds like something associated with SuperScripsit. That program uses files with an extension of "CTL". Good programs should have a description of the various file names making up their package. Check your SS manual, and you will probably find that information.

The file "DOS/HLP" is not a file which is to be LISTed. It is the data file used by the HELP utility. For instance, the command **HELP DOS** displays the DOS commands for which help is available. Information on the HELP system covers 2.5 pages of my TRSDOS 6 manual. You could probably make good use of that command.

See previous answer. Since DOS/HLP is not a plain text file, the result of LISTing it in ASCII provides nothing useful.

If you add together all of the byte values making up a file and discard overflow beyond 255 on each addition (i.e. modulo 256 addition), the resulting value is a one-byte checksum. If you add each byte modulo 65536 (i.e. using a 16-bit unsigned accumulator), the resulting value is a 16-bit checksum. The 16-bit checksum is usually what's dealt with since the Z80 does have 16-bit registers for addition.

Okay, so what are checksums good for? Typically they're used in communications transfer of files. As a modest level of checking the integrity of a transmission, the sender checksums the file before sending, then sends the file followed by the checksum. The receiver then again checksums the file and compares the calculated checksum to the sender's transmitted checksum. If they don't match, then there was a transmission error. If they do match, there could have been an undetectable error. A 2-byte checksum is generally more reliable than a 1-byte checksum. There are more elaborate schemes.

An entire field of science is devoted to the design and analysis of error correcting codes. This is usually termed, "Information Transmission Theory". That field is extremely important to the ability of receiving usable transmissions of data from satellites and spacecraft.

Please read about the PATCH command in your DOS manual. I see five pages of information there. A patch is not just to convert a program to run on a different computer than the one it was designed for. Most patches make alterations to a program to either fix bugs or provide enhancements.

Letters to the Editor

The Reviewer

Pro-EnhComp:

Mark Allen Reed
Reeds' House of Color
Glen Road Plaza
West Lebanon, NH 03784

Pro-EnhComp is an excellent BASIC compiler for the TRS-80 Model 4. The extensions it makes to the BASIC language are just what every programmer has been waiting for. Its unique in-line assembly language capability eliminates DATA statements, string-packing, and USR routines forever. Once you've seen your BASIC programs operate as stand-alone CMD files, you'll never want to go back to the interpreter.

Of course, Pro-EnhComp isn't perfect. The CMD files it creates are often slow by compiled standards. Also, the CED editor supplied with the Pro-EnhComp package contains a few quirks that irritated me when I used it. Overall, however, Pro-EnhComp's many good points far outweigh its few bad ones. I would recommend Pro-EnhComp for any Model 4 user who is serious about programming in BASIC.

I reviewed Pro-EnhComp, version 2.6, on my 128K TRS-80 Model 4D. MISOSYS also sells a Model I/III version of EnhComp. Since it uses the same instruction manual as the version I used, I think almost everything I write about Pro-EnhComp will apply to the Model I/III version as well.

The Programs

The Pro-EnhComp system consists of four programs and a support file.

CED/CMD is the tokenizing text editor used to create Pro-EnhComp programs. It is a versatile line editor with a massive text buffer and a full array of editing and file maintenance commands. With CED, you can evaluate integer expressions (such as 1234+2345), add or remove BASIC line numbers from program lines, copy or move lines within the program, suppress the display of BASIC line numbers, list program lines to the screen or printer, insert and delete program lines, renumber all or part of the program, display a directory of any disk drive, display lines from a disk file, search for and replace text, and define parameters for print-outs. Of course, you can also edit individual program lines. My only complaints are that the editor expands all tabs to eight spaces (which is far too many), and that it capitalizes the text of remark statements. Aside from those quirks, CED is very powerful and makes program modification simple.

BC/CMD is Pro-EnhComp's BASIC compiler; it takes a file created by CED or another editor and translates it into a stand-alone machine language program. BC can accept a wide range of parameters, such as suppress BREAK key checking, suppress error checking, suppress line number information, wait for the user to press a key after a compilation error, display the program on the screen, send the program listing and all messages to the printer, and suppress generation of a CMD file. In addition, BC can "include" other program files in your compiled program. That makes it much easier to write programs in modular format.

S/CMD is a supervisor program that links CED and BC together. Using S, you type and edit your BASIC program using CED. Then you type RUN, and BC compiles it. If any errors come up, S returns you to CED to correct them; otherwise, your compiled program executes, and you are returned to CED when it completes. Under S, program development is simple.

SUPPORT/DAT is a specially-constructed file containing support subroutines used by the BC compiler. BC displays the number of each subroutine it uses as it creates your program.

REF/CMD is a cross-referencer that details the variables, user-defined functions, user-defined commands, symbols and labels, and program line numbers in your compiled program. I don't use REF very often, but the cross-reference report it produces is detailed and thorough.

The Manual

The instruction manual is a good reference guide for experienced BASIC programmers. It could provide more examples, particularly in areas where Pro-EnhComp differs from interpreted BASIC, but any question you have can be answered either from what the manual implies or by trial and error. The writing style of the manual is direct, informative, and often humorous.

The manual does contain a few errors; they are corrected in the README/TXT file supplied on the Pro-EnhComp disk. You'll probably want to print out a copy of README/TXT and refer to it as you read the Pro-EnhComp manual.

The Language

Pro-EnhComp's implementation of BASIC is powerful. It combines the standard features that everyone expects with unique enhancements that make the programmer's job easier.

For instance, Pro-EnhComp allows in-line assembly language programming. At any point in your program, you can switch the compiler to "Z80-mode" and start programming directly in assembly language. Furthermore, the assembly language portion of the program has full access to all of BASIC's variables! Gone are the complex procedures which used to be necessary to link assembly language with BASIC. You'll never have to POKE from a DATA statement again.

Pro-EnhComp also supports user-defined functions and commands, and both constructs can span as many lines as necessary. Strings can now contain as many as 32,767 characters. Pro-EnhComp includes a new disk file mode that permits direct manipulation of FIELD variables (no more LSET and RSET). SET, RESET, and POINT have been restored to BASIC, along with the powerful new PAINT, PLOT, and DRAW commands. Pro-EnhComp contains a multitude of printer control commands, such as PAGELEN, LMARGIN, RMARGIN, and PZONE. That last statement is especially useful in business programs that print out columnar reports.

Programs can now GOTO and GOSUB labelled statements; line numbers are no longer necessary. Since GOSUB "SAVEFILE" is much more informative than GOSUB 5100, programs written with Pro-EnhComp can be self-documenting. In addition, Pro-EnhComp contains a built-in sort command that is very flexible. Model 4 users can stop lamenting the loss of CMD "O".

Speed

Some *MISOSYS Quarterly* readers have complained that programs compiled with Pro-EnhComp are slow. I have read published comparisons which state that some programs run more slowly under EnhComp than under interpreted BASIC! That surprises me, since I've never noticed that my compiled programs were sluggish. Then again, I bought EnhComp more for its extensions to BASIC than to create fast CMD files.

The compiler does a lot of disk access as it works. That takes time and puts wear and tear on your disk drive. If you have one, I suggest you place SUPPORT/DAT on a RAM disk. I did, and now I can compile most programs in under a minute.

Conclusion

Pro-EnhComp is an excellent buy. Its few faults are far outweighed by its impressive features. The in-line assembly language feature alone is worth Pro-EnhComp's purchase price. If you are serious about BASIC programming on your Model 4, you owe it to yourself to buy Pro-EnhComp.

CED/FIX

by Mark Allen Reed

I enjoy using Pro-EnhComp on my TRS-80 Model 4D. It is an incredibly powerful BASIC compiler system, and its in-line assembly language capability is unrivalled even on MS-DOS compilers.

However, the CED line editor (version 1.4) supplied with Pro-EnhComp has two quirks that annoy me. First, it expands every tab to eight spaces, which is far too many to be useful. Second, it capitalizes text following REM statements. That is,

```
'This is a remark statement
```

becomes

```
'THIS IS A REMARK STATEMENT
```

I wouldn't call these bugs, but they bothered me enough that I started looking through the program with DEBUG to see how I could correct them.

The results of my labor are presented below. Create the file using the BUILD command; I named it CED/FIX. Then, make a backup copy of CED/CMD just in case. Finally, apply the patches by typing

```
PATCH CED/CMD USING CED/FIX
```

Now every tab expands to a more reasonable four spaces, and all text following an apostrophe remark statement is immune to capitalization.

Incidentally, to make room for my modifications, I shortened CED's sign-on message from "EnhComp Line Editor," etc., to "CED 1.4." I hope that doesn't bother anybody. I was careful to leave all copyright and authorship notices intact.

```
D00,4F=43 45 44 20 31 2E 34 00
F00,4F=45 4E 48 43 4F 4D 50 20
D00,57=FE 27 20 02 0E 01 FE 22 C2 99 32 C3 A8 32
F00,57=4C 69 6E 65 20 45 64 69 74 6F 72 20 56 65
D08,BF=C3 53 2A 00
F08,BF=FE 22 20 F1
D09,49=00 00
F09,49=CB 3F
D09,50=00
F09,50=87
```

DOS Subjects

Model III LDOS

LDOS 5.3 DIR

Fm Richard VanHouten: I found another little bug in the LDOS 5.3 library commands, this one in the DIR library command. It is counting the "Filespec...Time" line twice, once when it displays this 63 character string terminated in 03H, the second time when it sends a carriage return. I suspect the string was originally 64 characters long, and the carriage return was sent to the printer only, and then was patched. This can be corrected by patching the CALL 58EC at 591D to NOP NOP NOP; actually, if the 03 at 5AD6 is changed to 0D, the 8 bytes from 5918 to 591F can be eliminated.

Fm Roy Soltoff: You hit the nail on the head; the LDOS 5.3's DIR command was only displaying 14 lines because it was counting the header line twice. That string was originally terminated with an ETX because it was supposed to be 64 characters long like the others. That means it must be ETX terminated to avoid a double line feed. The subsequent CALL to put the CR was also supposed to go only to a routine which output the CR to the printer if the PRINT parameter was active.

As it was, the string was only 63 characters and the CR was always output. The string was counted as a line and the CR was counted as a line. I prefer to fix things with the shortest possible patch. Thus, the following one byte change will allow DIR to output 15 lines; that gives one more line of filespecs.

```
. SYS6H/FIX - Patch to LDOS 5.3 DIR command.
. Apply via,
 PATCH SYS6/SYS.SYSTEM (D0C,1D=21:F0C,1D=CD)
```

That's all it takes to do the job.

Non-stop scrolling

Fm William Chao: Roy, I am just wondering if you can provide a simple patch for the display of the directory in LDOS 5.3 to scroll instead of 'paging' one screen at a time. This 'paging' happens when you list a file on the screen also (using LIST filename) and I don't like it as much as the LS-DOS's scroll feature.

Fm MISOSYS, Inc: A simple patch to permit the LDOS 5.3 DIR to scroll? How about using the nostop parameter? DIR partspec (N) should do the trick. You could also patch the parameter to default to N=ON. I don't have the patch for that handy. With FED2, it would take about 10 seconds to find it if I had a Model III LDOS booted up, but I don't.

Now that I do as I am composing this issue of TMQ, the "N" parameter (nostop) of DIR in LDOS 5.3 is at address X'589F'; thus, defaulting nostop to ON which would effectively keep the display scrolling, would be:

```
PATCH SYS6/SYS.SYSTEM (D0B,FD=FF FF:F0B,FD=00 00)
```

Now if you just want to get rid of clearing the screen before each page (I really prefer to clear the screen as it is so much easier on the eye), just change the X'CD' byte at X'5907' to a 21H effectively eliminating the CALL @CLS. That's:

```
PATCH SYS6/SYS.SYSTEM (D0C,0F=21:F0C,0F=CD)
```

In order to default the LIST command to nostop, apply the following patch:

```
PATCH SYS6/SYS.SYSTEM (D29,4A=FF FF:F29,4A=00 00)
```

RSHARD and drive step rate

Fm Bruce J Hutchison: I bought RSHARD from you and it works great on a used 5 Meg drive I bought. On page 6 of the manual it explains how to set the step rate. The default is 10 microseconds. It goes on to say if there is documentation with a different value to use that instead. The manuals for both my drive and a friends 15 Meg drive give a track to track access time of 3 milliseconds and that is the value I used on mine. (My friend had her drive set up by RS using TRSHD6. Who knows what value they used). I usually like to use defaults as I assume the programmers know what they are doing when they

write these things. (You did, didn't you?) In TMQ 1.4 page 30 you say (in reference to the RS driver, NOT RSHARD) to use zero or the smallest value given in the prompt. Also I remember in one issue of TMQ it was stated that the controller uses buffered stepping. That implies that I can use any value and the controller will feed it to the drive at the correct rate so I should use the default instead of the value in the manual. Is this a correct belief on my part?

Fm MISOSYS, Inc: We usually know what we are doing. But don't forget that the RSHARD disk driver can work with many different drives, not all of which would use the same step rate. There is only one default, which I believe is 3 milliseconds - the figure used by the 5 Meg drive.

International keyboards resolved

Michel HOUDE
8, rue du Docteur Roux
60200 COMPIEGNE
FRANCE

I was very pleased to see my name mentionned by Fred Pieters, in the latest TMQ. As he was my first and unique customer, I get a 100% satifaction rate! Actually, after my May'87 letter was published in TMQ, I got one phone call from a guy in Paris, and two letters from Belgium, one was Fred Pieters' you forwarded to me, the other was sent to 'Michel Houde, Teacher-Researcher, Chemical Engineering, Compiegne University, France', I received it! It came from Dirk Vandenbossche. All 3 of them were, of course, interested by my adapting LS-DOS 6.3 to a French keybord. To all 3 I said: "Please ask Tandy". Tandy finaly released LS-DOS 6.3F in october (more on that later), but Fred Pieters wanted LDOS 5.3 to work properly on his Model_4. I asked him to send me a 'QFB' copy of his master and patched him as needed, adding a KIAZ4/DVR, a PR/DVR (for proper European character conversion), and even a special TWOSIDES/JCL to let him build a double-sided optimized BOOT disk. I did not hear of him since I sent him the diskette, so I thought no news is good news. I'm glad to know he's satisfied. I asked him a 300 French Francs fee. (That's $40-50, depending on current rate and what the bank actually charges). These were immediately spent on LS-DiskDISK. Which means that the few cents you spent on forwarding Fred's letter returned 3 sales: LDOS 5.3 and Mod4 HIK to Fred, and DiskDISK to me, and 2 happy customers of yours. But I wouldn't do it any more, as I spent hours writing a 'README' text explaining what he could do and what he could not with patched LDOS. You never know how people use their computer, and what program they use, some of which are so tricky, I didn't want my work to be blamed for someone else's errors. Now I understand how difficult it is to write accurate and comprehensive documentation!

As Fred Pieters praises my job, I decided to sent it to you. All the needed explanations are included in the SY053F/ASM file. All there is to do is: PATCH SYS0/SYS.SYSTEM using SYS053F/FIX.

```
.SYS053F/FIX
.Patch to enable LDOS 5.3 to work on French Model
III/4
.Patch SYS0/SYS.SYSTEM using SYS053F/FIX
.x'4dca'=08, KFLAG$ scanner, LD A,(3808H)
d0c,b3=08:f0c,b3=01
.x'4dcd'=67, BIT 4,A
d0c,ba=67:f0c,ba=47
.x'4e25'=30, European chars
d0d,12=30:f0d,12=38
.x'507e'=70, European chars
d0f,73=70:f0f,73=78
.x'4024', no space comp
d01,97=01:f01,97=00
.x'4e4e'=CD 00 5F, call 5F00H, install high memory
patch (KI and PR)
d0d,3b=CD 00 5F:f0d,3b=21 ED 41
.x'4ec3'=cd b0 5f, call 5FB0H, enter DATE (dd.mm.yy)
at Boot prompt
d0d,b1=B0 5F:f0d,b1=28 42
.these last patches must be last for PATCH to
function correctly
.note that the only restriction on the length of a
line is the number of bytes
x'5f00'=21 ED 41 00 E5 3E 46 32 3B 3C 2A 11 44 22 68
5F 01 2C 00 AF ED 42 22 11 44 22 11 44 23 E5 C5 11
66 5F D5 B7 ED 52 44 4D FD E5 FD 21 5C 5F FD 6E 00
FD 66 01 7C B5 28 0F 5E 23 56 EB 09 EB 72 2B 73 FD
23 FD 23 18 E7 FD E1 E1 C1 D1 ED B0 21 8B 5F 22 E8
41 21 66 5F 22 F1 41 E1 C9 4F 5F 55 5F 6F 5F 76 5F
00 00 18 06 91 5F 03 24 4D 48 CD 74 5F CD 24 30 F5
21 88 5F 11 1D 42 06 03 4E 1A 77 79 12 23 13 10 F7
F1 C9 00 00 00 DB F8 E6 F0 FE 30 C9
x'5fb0'=21 12 3F E5 11 D2 5F CD 28 42 E3 D5 F5 22 20
40 2A 26 42 7C 65 6F 22 26 42 3E 1E CD 33 00 F1 D1
E1 C9 1E 44 41 54 45 20 28 64 64 2E 6D 6D 2E 79 79
29 20 3F 20 03
.eop
```

Instead of sending you the KIAZ4/DVR file (AZ stands for AZERTY, as you have come to know), I decided to muscle it a bit, and came out with KEYB/CMD, which is an universal international keyboard driver, you invoke via:

```
KEYB (FR,M4,Type,Jcl,Delay=d,Rate=r,Wait=w)
```

Type, Jcl, Delay, Rate, are the same parameters as standard KI/DVR, with same effects, including re-using memory if already resident. Wait is a special parameter which changes the value used when calling @PAUSE, it is aimed at Model_4 with XLR8er, if needed for debouncing purposes. M4 means Model_4 (Model III is default). FR means French keyboard, it may be replaced by GR for German, and even US for English (US is the default). It is installed as a CMD file, because it did not fit in the SET driver region, and it is for *KI anyway, the DCB being a documented address. The beauty of this is that all 6 different cases (cross US, GR, FR vs M4, M3) are handled by a unique driver (with minor mods being performed

at load time) which always reinstalls itself in same memory when invoked subsequently, and as it does not use x'4df2', x'4df4', x'4df8', it does not interfere with KI/DVR or KI4/DVR modules, although it does set KFLAG$ bit(s). Which means that it is possible to try them all, with RESET *KI between them, but prepare 2 one-line JCL files, one with RESET *KI, another with MEMORY, as you will loose access to the asterisk in both GR and FR modes, and the letter M in FR mode. By the way, I used the undocumented @GTMOD rst system call. As usual, a lot of error checking is performed. Now the module is longer than $KI or $KI4 (596 bytes vs 366 or 401), but as I reduced the type-ahead buffer to 128 bytes (64 key strokes, one line, should be adequate) and optimized the code, my Type and JKL modules are 287 and 71 bytes long, vs 436 and 86 for $TA and $JKL, which totals up to 954 bytes for KEYB (T,J) vs 923 for KI4 (T,J), not that bad!

Note that I put great efforts in making the up-arrow key code available to SAID, a one byte patch to patched SAID should be ok (IKI module name vs $KI), same offset for both Model III and 4 modes as Model III mode KI. I did not support toggling between x'5B' and x'1B' for up-arrow, sorry Roy!

If you wonder how I got the German data, well here in France, when we bought either the graphics board or the hard disk kit, we got 3 versions of the supporting software: the regular TRSDOS_6.x version, plus the TRSDOS_6.xF and TRSDOS_6.xD versions. I said that in the past tense, as I wonder if anyone would care to buy them now: the Model_4 is an unknown product in Tandy stores or Computer Centers. By the way, we never had a French 4D, neither clustered arrows Model_4s (they were made in France), but the few last 4Ps had them (I got one!). While speaking of French models, yes Roy, Tandy omitted the shields, as no one cares about RFI here in Europe, I wonder who ever heard of it! I say in Europe, because in those European Community days, that would be an ECC concern. More and more standards are nowadays ruled on an European scale.

To answer another question, yes we have a French SuperScripsit, since Model III days, and it does its keyboard scanning via CTL 255. I did not even attempt to translate Lynn Sherman's CTL255 filter for the very reason that I consider PROWAM Version 2 to be essentially incompatible with European characters, which are treated as 'protected'. With PRONTO's CARD/APP, one could enter all characters, export or import them. You can't do that with PROWAM 2. Now I admit that there are so few people interested, that nobody could blame you, Roy, for the design choice you made. Rather blame Tandy for the hardware, which is the real culprit. Now, I must say that French Superscripsit and French Multiplan do care for European characters in inverse video, but these are huge programs, not a resident desktop manager.

Don't misunderstand me, I like PROWAM. How do you think I did to generate those long patches? Well I used TED/APP and FED/APP, exporting the hex data pairs from FED to TED.

I recall someone asking how to display DIR in 24 hour format. Here are the patches to LS-DOS 6.3 SYS6/SYS:

```
d09,11=00;f09,11=0C
d09,17=00;f09,17=0C
d09,32=20;f09,32=61
d09,36=20;f09,36=70
```

Now a last word about LS-DOS 6.3F, which I understand, Fred Peters sent you. When I asked Tandy-France about it in October 87, they told me they were seriously thinking about it, but no release date could be stated. When I phoned again in December, they sent it! FREE! But no documentation, no explanation. Just a "Date > 88" on the label. All 'new' (as compared to 6.2) are password protected with NO access, which is probably safer when no docs are provided. All files are dated 1-Sep-87, and the serial # displayed on the Boot screen is replaced by "18-Oct-1987". After doing cross comparisons between 6.2, 6.3, 6.2F, 6.3F, I can say that LS-DOS 6.3F as sent by Tandy is equivalent to 6.3 level K/L (not L+), with no CS# displaying when typing ID, and no dummy code in the last sector of SYSn/SYS files.

Well, that should be enough for today. I should write more often. I like TMQ's new look very much. Please keep the Family Update. I would not write such a letter to an anonymous PO Box. Best wishes to you and your family.

Fm MISOSYS, Inc: I appreciate tremendously the amount of work you have done, Michel. I will put your KEYB international keyboard driver for LDOS in this issue's DISK NOTES. Thanks again for the efforts.

Now here's one for you. If you want to get rid of the "protected character" facility in PRO-WAM's import/export functions, a simple 2-byte patch will do that. At X'44EA' of PROWAM you will find a `BIT 7,A` instruction, That tests the high order bit of the character being moved. If you don't want that test to eliminate all extended characters, change the code to NOP followed by CP A. This leaves the Z-flag set for the proper handling as if the character to be moved was not an extended character. The patch would be:

```
D12,CE=00 BF;F12,CE=CB 7F
```

The downside of this is that the forms facility within the CARDX application will no longer be able to protect any fields, although you will be able to utilize the reverse video.

Model 4 LS-DOS

Programs with LIB names

Fm Ralf Folkerts: Roy, I have a little problem with my LS-DOS 6.3: I'm just 'upgrading' my 'Mark IV Pro-PaDS' to a 1.3 Version that supports dating to 1999. I've extracted the members from the PDS/CMD file and add '13' to the filespec when I assemble the 1.3 Versions (DIR13/CMD,...). That worked fine with BUILD and DIR, but when I've tried that with APPEND the DOS always executes it's append LIBrary command. I can even add '/CMD:1' to the APPEND13 - it doesn't work. It's not too bad - I've renamed it to 'APP13/CMD', but I'd like to know if there is a patch to prevent this! I've tried that with TRSDOS 6.2 and LS-DOS 6.3D, too - same result. I hope you have a tip.

Fm MISOSYS, Inc: No patch needed. Just a little reading of the manual; but I doubt that Tandy put it in theirs. If you want to execute a program that has the same name as a DOS library member, you have to prefix it with an exclamation point. Thus, in order to invoke the "APPEND" module which was extracted from PaDS, enter the command line as,

```
!APPEND program padsfile (...
```

Drive searching

Fm Robert G Strickland: Roy, A few questions about the XLR8 board, now installed on my 4P. With all the LC files loaded on Ramdisk and with Ramdisk operating as the System (drive :0) drive, I notice that the compiler/DOS still looks for files on :1 & :2 before using :0 (the Ramdisk). After a successful compilation, if I run the same compilation again :1 & :2 are not interrogated. Any thoughts on this ?

I imagine that it may have something to do with the speed of the memory chips. I am running the XLR8 at (1,2,80) until I install the faster chips.

Fm MISOSYS, Inc: It may still look for files on drive 0, however, without a red light on the RAMdisk which comes on when the drive is accessed, how do you know it is being accessed? You don't. Folks fail to appreciate that when you @INIT a file, the DOS does a global search if the file does not exist. It will stop as soon as it finds the file. [If any program, LC included, @INITs a file which doesn't exist, the DOS has to search all drives if a drivespec is not part of the filespec. Once it finds the file is nowhere to be found, it then creates it on the first available (un-write protected) disk drive. The second and subsequent times you run that compile, the DOS finds the file on drive :0 for the @INIT and thus needs no global search. Make sense?

Fm Robert G Strickland: Roy, Your reply suggests that I may have not been clear. With Ramdisk designated as drive :0 and with all necessary LC files copied to the Ramdisk drive :0, the system still looks to the two floppies to run the compile and assemble process. This seems obvious to me (?) in that at the start of the compile process, before any writes are made, the activity lights on the disk drives come on. For example, the compile JCL is on Ramdrive :0, yet the system seems to be looking for in on :1/:2 before loading it from :0. This is evident in that if there are no disks in :1/:2, the lights still come on in sequence. Then the compile runs, evidently "finding" the required files on the Ramdisk :0. Is this right?

Fm MISOSYS, Inc: You misunderstand me! When you are doing a compile, the compiler has to create (via @INIT) an output file. The DOS will always search all disk drives for an existing file of the name you designate until it finds one. If none are found, then the DOS will search all drives starting with 0 until it finds a drive which is available for writing. In your case, the object file was not available so the DOS looks on drive 0, drive 1, then drive 2. I hope you don't think that when the DOS wants to write a file, it creates it on drive 0 without looking first on all drives. TRSDOS and its derivatives are not MS-DOS. We don't use the concept of a current directory (or current drive) where all files are written unless told otherwise. Do I make the answer clear now?

Fm Shane Dawalt: Robert, When you run a JCL, it must compile the JCL first. LDOS uses a file called SYSTEM/JCL. The system will search all drives for SYSTEM/JCL on the first execution of any JCL. Since you are running a RamDrive, most likely, there is no SYSTEM/JCL on the drive when you first start up your machine. Hence, DOS looks for SYSTEM/JCL first and, not finding it, creates it on drive 0, in your case the RamDrive. If your JCL doesn't need compiled every time, execute it like this:

```
DO * filename
```

The '*' inhibits the JCL compiler from compiling filename/JCL. Also, note, that using this syntax inhibits the JCL compiler from searching for and creating SYSTEM/JCL.

How do you know if you don't need to compile it? If your JCL has any lines beginning with '//' then it MUST be compiled. Similarly, if parameters are entered on the DO command line, it MUST be compiled.

Invoking programs from BASIC

Fm Dave Spiceland: Is it possible to run a /CMD program in BASIC? I know some LIB commands can be run by: SYSTEM"TOF for example. But is it possible to run a

program which prints something on the computer and then return to BASIC?

Fm Shane Dawalt: Dave, I strongly doubt it. If I remember my M4 BASIC correctly, upon startup, BASIC sets a special flag in the operating system. This flag tells the operating system that only library commands may be executed. If you attempt to run a program other than a library command, an error message will be displayed and you will be returned to BASIC.

Fm Jim Beard: Dave, I believe that the system overlay area, 2400H to 3000H, can be used from BASIC via the SYSTEM command. You can check where a /CMD file loads using CMDFIX.BAS in DL1 (I think) of the LDOS forum. If you compile languages yourself, you can probably force them to load at 2400H. You have 3K to work with. If you are using commercial products, check them out with CMDFIX.

Fm Joe Kyle-DiPietropaola: Dave, The other replies give you some of the story, but the key is this: BASIC sets the "only execute this command if it is a internal system command" bit before executing SYSTEM arg$ statement. So, if you **know** that your program will fit in the X'2400' to X'3000' area, you can force it though by using the RUN verb. As in: SYSTEM"RUN BSORT blah, blah" where BSORT/CMD is an external utility.

Hah, folks scoffed and said "what's this silly command good for..."

Fm Shane Dawalt: Jim, If I recall correctly (my famous phrase of the year), BASIC sets the LIBRARY execution only bit (bit 4 of CFLAG$) in LSDOS. I don't believe the system allows execution of /CMD files when this bit is set, regardless of their load origin.

Fm Shane Dawalt: One question, Joe, what if the program you RUN loads outside the overlay area? I suspect its crash city huh?

Fm Joe Kyle-DiPietropaola: Shane, Fireworks eventually, depending on how much later the overlaid code gets used.

Fm Jim Beard: Shane, I believe you are safe if you keep to the 2400H to 3000H range. Joe can tell you more about how to exit back to DOS prompt. I once got him all bothered by suggesting to someone that they exit BASIC by SYSTEM "BOOT". The question was "how do I exit BASIC without closing my files?" Also, someone who wanted to make his MS-DOS disks readable under LS-DOS got my suggestion to FORMAT :1 (q=n). I shouldn't do that. Joe doesn't think it's funny, either.

Fm Joe Kyle-DiPietropaola: You could have a stubby that loads into the 2400H to 3000H region, off-loads 3000H to HIGH$ to disk, moves itself to HIGH$, protects itself and then invokes whatever you really wanted via @CMNDR (without the system command bit set) and then puts things back later. Messy.

Model 3 LDOS BASIC is smart enough to crunch itself up against HIGH$ moving all loose space low before invoking a command. So, you can execute anything you want. Of course, don't forget that the system loader doesn't honor HIGH$ (and no, I don't want to bring that one up again!

Fm Jim Beard: You're right about some people trying FORMAT :1 instead of HyperCross. I wouldn't call that stupidity, since lack of information would be the real cause. I felt guilty as soon as I posted the message. A couple of hours later, I saw your reply and left it up, but I was going to yank it.

It wouldn't be hard to write a loader which (1) fit between 2400H and 3000H, (2) honored HIGH$, and (3) would save BASIC and add command-line arguments. Something like JCL would have to be invoked to call another module to restore BASIC to memory and return, I suppose. Hmmm. ---

Fm Dave Spiceland: Shane, Aha!! Therefore, running /CMD programs in BASIC is impossible. I had thought it could be done in Model III NEWDOS. Thanks!

Fm MISOSYS, Inc: Yes it could. And it could also be done in Model III LDOS BASIC, DOSPLUS, and about every DOS except Tandy's. And you can run certain CMD programs under Model 4 BASIC if they adhere to the restrictions noted. Don't forget that Model 4 BASIC is stock Microsoft. What Microsoft did at the least, was to allow you to access LIBrary commands via the *SYSTEM command-string* statement. On the other hand, Microsoft originally had BASIC ORG'd at 3000H - you would have lost another 2.5K of program space. We convinced them to ORG BASIC at 2600H and separate out that portion of BASIC from 2600H-2FFFH into an overlay, BASIC/OV1. Upon return from a SYSTEM command, BASIC reloads the overlay. This lets you access any CMD program which runs entirely within the library overlay region by using the library RUN command as a program executer, as Joe noted.

BASIC FOR-NEXT loops

Fm Steve Brewster: Two questions... (1) what is the difference in the way model 3 DOS and LS-DOS 6.3 BASIC handle for next loops? If I run this simple sample program:

```
10 For I = 1 to 30
20 If I<21 then next
30 next
40 Print "all done"
```

LS-DOS 6.3 returns an error "next without for in 30".

Second question. Why do you get an error "duplicate definition in line xxxx" when a LS-DOS 6.3 basic program halts because of an error and then when you type run the error message is printed. Line xxxx contains a DIM statement. The only way I seem able to run the program (which runs ok on the first time through) is to return to system mode and the invoke basic again.

Any insight into these questions would be most helpful and appreciated.

Fm Shane Dawalt: In LS-DOS 6.3 (BASIC 1.1.1), only one NEXT can occur for any FOR NEXT loop. This is a standard programming practice. Model 3 BASIC allowed multiple NEXTs for one FOR statement. This should be illegal. After all, the statement is a FOR NEXT loop, not a FOR NEXT NEXT NEXT NEXT ... loop. (I got bit by that nasty programming incompatibility too. But it certainly forces one to keep up with structured programming.)

Your second question needs a code example. Typically, when you type RUN, BASIC clears all variables so no duplicate definition error will occur. Have you tried to enter CLEAR at BASIC Ready when you get this duplicate definition error? CLEAR also clears the variable definitions from memory. Does the program do any weird pokes into RAM?

Fm Michael Rubio: Ah! Finally I get to help someone out! The first question, Steve, on the for next looping and you get a next without for is caused by the If statement. Take line 30 and make it a part of line 20 using else : If I<21 then next else next. And the duplicate definition usually means that a variable was DIM'd twice. Since basic sets aside the memory for that variable, it can't adjust it at anytime unless there has been a clear statement. But, the clear statement is not always in order, say you need only to erase one variable, then use : DIM A$(100):ERASE A$:DIM A$(200) for example.

Fm Steve Brewster: Shane, This is the first time I have run into this error message, if the program is run only once, all is well, but when execution fail from an error typing RUN gives this error. I do not see any duplicate code in the program and it does not use any pokes. I will try typing clear, and let you know.

Fm Shane Dawalt: Actually Michael, making line 30 part of line 20 won't work cause you'll have:

```
10 FOR .....
20 IF .... THEN NEXT ELSE NEXT
40 PRINT "done"
```

This will still give an error, only this time it will be a "NEXT without FOR error in line 20". In BASIC 1.1.1 (or 1.1.0), you cannot have multiple NEXTs linked to a single FOR. You can only have one NEXT for one FOR. This was valid in Model III BASIC (ROM and TRSDOS1.3). It is not valid for BASIC 1.1.1.

I recall when I first found this out. 'Twas a painfully long program ported from TRSDOS1.3 to M4. I quickly threw the program away as it had FOR NEXT loops written everywhere.

Fm Michael Rubio: Shane, That's interesting, it works on my system. 6.2 Model 4. That is the solution that I found when I ran into the same problem.

Fm Shane Dawalt: Michael, I've tried 'em both and I get NEXT without FOR on BASIC 1.1 and 1.1.1. I'll reread your reply, but I'm 95% sure I entered it correctly. Here's what I entered, as a test program, for this illusive critter:

```
10 FOR T=1 TO 100
20 IF T<50 THEN NEXT
30 NEXT
```

This generates a NEXT without FOR in 30, as I expected. Now, change line 20 to be:

```
20 IF T<50 THEN NEXT ELSE NEXT
```

Delete line 30 and RUN. This generates a NEXT without FOR in 20, as I expected. This was run under BASICG 1.1.0. I have purged all prior 6.3 releases except BASICG. I think when the IF construct fails, BASIC pops the first NEXT it comes to from the stack then when it reaches the second NEXT, there is no other NEXT entry on the stack to be popped. The interpreter screams at the programmer for this.

Fm Joe Kyle-DiPietropaola: Actually Steve, the difference is in the BASICs, the DOS doesn't know or care from FOR/NEXT loops. Anyway, you have quite a few replies

already that probably have you covered, but I'll read further on and see what develops...

Allocation Methods

Fm Adam Rubin: After reading "Ramblings" in TMQ II.iii, p.2 and the BYTE article mentioned there ("Extra Edition", Vol. 12 No. 12, p.185), I decided to compare the various allocation methods. I took an empty logical drive on my HD (1206K free, 1K granules), and devised a program to repeatedly create and delete files of random sizes. With the same test under LS-DOS 6.3 (patched for allocation method when necessary), the final results were:

> Random allocation (as in 6.0.0) -- average 1.634 extents per file; First-fit allocation (as in 6.3) -- average 1.824 extents per file; Next-fit allocation (described in articles) -- average 1.059 extents per file.

So, this seems to uphold the conclusions found in the BYTE article -- namely, next-fit allocation results in much less file fragmentation than first-fit allocation. (I don't know why "random" came out slightly ahead of "first-fit", though.) Questions or comments, anyone?

DiskDISK & drive configuration

Fm Dave Spiceland: I have a Model 4 with an external drive & using LS-DOS 6.3. I'd like to change my drive configurations to make drive 2 act as drive 1 and drive 1 will become drive 2. Is there an easy way to switch the drives after boot-up?

Fm MISOSYS, Inc: An easy way to reverse the effect of drives 1 and 2 would be to use the SWAP utility we used to sell. It's now available in the **GO:SYS** product bundle.

Fm Joe Kyle-DiPietropaola: Dave, In addition to SWAP, you can use the commands

```
SYSTEM (DRIVE=1,DISABLE,DRIVER="FLOPPY")
3
SYSTEM (DRIVE=2,DISABLE,DRIVER="FLOPPY")
2
```

and SYSGEN the result. Note that SWAP would be much faster and easier to use, as you can flip and flop at will. This technique requires multiple commands, and answering prompts or using JCL.

Interrupts and tasks

Fm Shane Dawalt: Here's a question which has been nagging at me: If a code segment is written and linked to the task processor, that code segment is not allowed to issue an EI instruction. (I'm not sure why it would want to in the first place, but humor me.) Something along those lines, can I assume ANY non-disk I/O SVC will not execute an EI instruction?

Fm MISOSYS, Inc: I don't think you can make that assumption. If I read you correctly, you didn't limit your query to interrupt tasks. Any program or routine which is not an interrupt task has total freedom to issue either DIs, or EIs. A program can have a DI without an EI. But if it issues a disk I/O request which is satisfied by the floppy driver, that does an EI. The @CMNDI does an EI to ensure that interrupts are on following a program's conclusion and @EXIT. So you can EI anywhere in any program as long as it is not an interrupt task. The reason is simple. When any interrupt is generated, the Z80 services it and essentially inhibits further interrupts. If the task processor is executing, you don't want to allow the CPU to accept another interrupt say from the RTC, as that would force another entry into that task processor. That in itself is not destructive as the task processor is re-entrant; it utilizes no static storage. On the other hand, the interrupt processing is very stack intensive. A recursive interrupt handling may very well cause the stack to overflow. That could be problematic. Clear now?

Fm Joe Kyle-DiPietropaola: Shane, No. All the byte I/O routines (@GET, @PUT, @CTL), for example, end up turning the interrupts back on. This caused Les quite a bit of trouble in LS-Host/Term, as I recall.

Even if there is no file I/O involved, the byte I/O calls will turn interrupts back on if they are currently off. Where this comes in as a problem was in the following scenario.

With the $CL driver wakeup vector set to your handler routine, you will be called whenever a character is available. The character will be in C, but that is the "raw" character. To honor any filtering on the device, you must use @GET on the device instead. Problem is, you were called with the interrupts disabled. If you get the character via @GET, the interrupts are turned back on.

At high baud rates, this can cause your interrupt code (and the system interrupt handler) to be re-entered before finishing processing of the previous character. Messy at best.

Fm Les Mikesell: Shane, The problem is that the byte i/o routines normalize the memory banks and make sure the video ram is swapped out, because the drivers in high memory may be called. To do that safely, it has to disable interrupts so the previous state can be saved. When it gets done, it enables

interrupts even if they were off coming in. This was a problem for COMM (remember 6.0) because all the foreground devices (keyboard, printer, screen, disk) share a common buffer and buffer management routine with the background comm/dvr. I ended up duplicating most of the byte i/o routine inside of COMM just to avoid the EI. Of course the interrupt processing routine also adjusts memory, so I didn't have to worry about that part.

DATECONV those RAMDISKs

Fm Frank Slinkman: Roy, I just noticed the RAMDISK utility not only doesn't support time stamping, it doesn't support dates after 12/31/87. On any file which is created on, copied to, or backed up to the ramdisk, all 1988 dates get changed to 1980.

I can live without time stamping (although it is helpful in many cases), but correct dating is very important to me in keeping track of the text files I write for a living!

Can you please supply a patch to bring RAMDISK/DCT date-stamping in conformity with LS-DOS? And if there's a patch to enable time stamping, that would be much appreciated as well.

Fm MISOSYS, Inc: Please pay attention now. DATECONV has been provided on LS-DOS 6.3 to convert disks which use pre 6.3 dating conventions. If you use a JCL to install the RAMDISK, just add the DATECONV command line. That's totally transparent. No patch needed, per se. Certainly one could be constructed. But you do have a solution...

Fm Frank Slinkman: Well Roy, it ain't quite as simple as that. I tried several times, BEFORE putting the question on the board, and kept getting the "Can't convert dates on non-6.3 SYSTEM disk" message.

Finally, AFTER reading your reply to the question, I tried it AFTER backing up the /SYS files to the RAMDISK, but before establishing it as the system disk, and that worked. However, your answer DID push me into experimenting a bit; so the problem is now solved, and I thank you for that.

Fm MISOSYS, Inc: Don't know why you would have gotten that message from DATECONV. On the other hand, LSI's implementation has been known to fail like that. That's why they put in the parameter, CS. If you do a:

```
DATECONV :d (CS)
```

then that will force the date conversion regardless. I believe that it is noted in one of the addendum. It may have also appeared in a TMQ.

VIDTXB.FIX

Fm Joe Kyle-DiPietropaola: Here is the complete text of the patch for comparison purposes:

```
. VIDTXB/FIX - 12-Jan-1988 - Developed by Joe Kyle
. [76703,437], SYSOP of the LDOS/TRSDOS 6 Forum
.
. For the latest in information about your TRS-80,
. GO PCS49
.
. This patch allows Radio Shack's Model 4 VIDTEX
. Version 01.00.00 to accept the variations in
. COMPUSERVE's "B" protocol introduced to
. improve performance circa December 1987.
. The program will now support reduced quoting of
. control characters, but not full QuickB protocol.
.
. Install using the command:
.     PATCH VIDTEX/CMD USING VIDTXB/FIX
.
D04,9A=00 00 18;F04,9A=FE 20 30
. Remove restrictions on quoting range for data
.
D04,F7=00 00 18;F04,F7=FE 20 30
. Remove restrictions on quoting range for checksum
.
D2B,7B=31;F2B,7B=30
D2B,D6=31;F2B,D6=30
. Change version number to 01.00.01
.
. End of patch
```

*DESKMATE and *CL*

Fm Dave Spiceland: Joe, I've got another question about DESKMATE: I'm trying to setup a configuration for DESKMATE. When checking DEVICE I get:

```
*CL <=> X'0FF4'
```

How do I duplicate that from the keyboard? I know this is a silly question, but I haven't yet mastered this part of the DOS. Any help you could offer would be appreciated!

Fm Joe Kyle-DiPietropaola: No problem Dave, you need to do a

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

followed by a

```
SETCOM (Q)
```

if the defaults (displayed with a plain ol' SETCOM) weren't suitable.

180K or 184320 Bytes?

Fm Samuel A. Robbins: Why are the disk drives on the Model 4's listed as being 184K drives when they only format to 180K? Thank you for your patience and a BIG thanks for LAIR OF THE DRAGON; It is a really superb game, even though I haven't figured out how to kill any of the nasties yet.

Fm MISOSYS, Inc: Your confusion over disk size is in confusion over the term, "K". A 40 cylinder floppy has 40 tracks times 18 sectors per track times 256 bytes per sector. Multiply that out and you get 184320 bytes. But a "K" is 1024 bytes, not 1000 bytes. So divide 184320 by 1024 and that equals 180K. The size values of the DIR command, for instance, are given in units of 1024 bytes, "K". That's where the difference appears between 184 and 180. It is really wrong to use a term such as "184K" when referring to a 40-track floppy's capacity. We reference capacity, be it disk space or memory space, in units of "K" which is a term steeped in binary. Unfortunately, the world of engineering uses "k" as an abbreviation for "kilo", meaning thousand (as in kilobaud, not kilobyte). The abbreviation for "kilo", meaning 1000, should also be lower case whereas the "K" for indicating 1024 should be in upper case.

Fix for Alwrite and PRO-WAM

From Brad Stiles: Congratulations on the arrival of Benjamin C. I predict that in less than two years your "fatherly" experiences will begin to change dramatically. We have Stephanie (5) and Jeffrey (18 mths), and although they are similar in many ways, the gender difference is also the result of vast differences which we had not believed possible. We are very pleased for you, and by we I mean both I and my wife, Pam. When the MQ arrived, she called to announce its arrival and also the good news. Pam is not at all interested in computers, but she has an enduring interest in the Family section of the MQ.

[Here's a] plea for help with an Alwrite/Electric Webster/Pro-wam, conflict. My normal system configuration (except for the printer routing) is indicated in the following DEVICE (B) and MEMDIR listings [Note: listings omitted to conserve space as the end result indicates they are not needed for publication -editor]. I have an Alpha Tech 512K Ram Board and use the RAMDRIV4/DCT driver from RAMDRV/LQR on Compuserve. The configuration also includes your KSMPLUS2/FLT. The DOS is at LS-DOS 6.3.0.K (CustID# 15815) with all patches obtained from the LSI-FIX files on the DISKNOTES disks. The DOS is also patched with your AT patches, from DISKNOTE VI, with the correction from MQ, Vol. I.iii, pg 110 applied.

I have always noted that when using Alwrite (Rel. 1.13) together with Electric Webster and/or PRO-WAM (2.x, Ser# 010265), the `Device (B)` command will eventually result in the normal display, followed by a bunch of sporadic characters and junk. This never resulted in any serious inconvenience until a few months ago, when I began using the machine on my job, 8-10 hours a day. At this time I noticed unusual behavior with the software, and occasionally a computer hang, generally late in the day. The computer hangs generally (but not always) involve the use of PRO-WAM, where on the return from PRO-WAM, the previous AL screen is displaced after which the machine becomes comatose. I have since replicated this behavior at the office, at the house and on a 4P (no ramboard or AT patches); although I have not been able to develop a sure-fire "hang" procedure. I have also hypothesized that the problem relates to the passing of control from AL to and from the other programs, eg., AL to ALF, AL to PRO-WAM, AL to EW, etc. I have carefully tested this hypothesis by working for extensive periods with AL, without ever calling ALF, EW or Prowam from within AL. The result was a clear `DEVICE (B)` display and absolultely no trouble. This is not, however, a very satisfying procedure. I should also mention that I have had absolutely no trouble between PRO-WAM and VisiCalc, MultiPlan, PFS-File, PFS-Report, SAID, BASIC, etc.

I have contacted PRO-Soft (Ron), and received absolutely no satisfaction, It was the "*.. it is the other guy's fault ..*" song and dance, with a little of the "*.. even if we knew what the problem was, we wouldn't fix it ..*" melody. I bought Alwrite, because of generally favorable discussion of the PRO-WAM/Alwrite compatibility in the MQ. I like the software, but it seems to be the core of a problem that is very inconvenient. I have also noticed, with some interest, similar or possibly related discussions in the MQ: Charles Ainsworth, Vol II.iv, pg 92 and your Blurb in Vol III.i, pg 3, re., the system stack evolution.

I have included a self-addressed-stamped-envelope, in the hopes that you can shed some light and solution on this situation. I realize that you are backed up with activities right now, but if you would Q this up for future consideration, I would really appreciate it.

Example of unusual behavior with software: KSM filter codes getting through into Alwrite text instead of the normal Alwrtte macro stored under same code (eg., KSM <Clear>E = "device (b", Alwrite <Clear>E = "to end of file")

Fm MISOSYS, Inc: Brad, The biggest clue as to unfortunate behavior is your statement about the DEVICE (B) command display. When the device map produces a junked up display at the conclusion of the correct device streams, it is caused by a corruption of the page of memory which contains the device control blocks (DCBs). It so happens that the DCBs are stored in memory from 200H through 2FFH. The system stack is placed at 37FH and is used towards lower memory. It is also split into two parts, one from 37FH to 340H and a second

from 33FH to 300H. This provides a 64-byte stack which is acceptable for system use. I purposely designed the system stack to immediately follow the DCB page so as to incur the least possible damage from a stack overflow (there is no provision in the Z80 to trap a stack overflow).

Unfortunately, some programs do not provide their own stack space as would be expected in proper program design of complex, stack intensive, operations. I know that ALLWRITE does not provide its own program stack area; I feel strongly that it should. The fact that it uses the system stack is the reason why it became necessary to alter the system stack split from 360H to 340H. While writing this letter, I checked the AL/CMD file which I have with FED2. AL does use the system stack entirely. I then checked ALF/CMD; it saves the contents of the SP register then sets it to its own area. It then restores to the previous SP setting, I assume when it exits or returns to the calling program.

The flexibility of device installation and filtering introduced into LS-DOS 6.0 additional use of the stack (be it program or system) since a filter usually pushes the IY index register and issues an SVC to chain to the downstream filter or device. This procedure would use four bytes of stack plus any additional used by the filter. Thus, each additional filter in a device stream adds to the stack usage. If a memory bank other than zero is resident at the time the initial device call is invoked, then four additional bytes of stack are used.

Enough folks have used PRO-WAM with ALLWRITE without being exposed to any problems. In your case, it may be the additional usage of the KSMPLUS filter. I checked the code on that filter; it uses eight bytes if no pending string is available. The interrupt handler is going to be active. If an interrupt occurs at the lowest point in a device call, the maximum amount of stack will be used. The task processor uses 20 bytes minimum. Maximums may add another 8-10 bytes. The Alpha Tech patches I released will also use about four bytes on every interrupt. Certainly, excessive usage of the system stack is causing you problems. PROSOFT may not be able to do anything about that without a revision to ALLWRITE. It could be that the EW program also uses the system stack, and shouldn't.

Okay, what can you do about it. You also mention a problem in KSMPLUS keys interfering with ALLWRITE's macro keys. That will always occur. There is nothing which can prevent that. The two programs are mutually exclusive. You will have to RESET the *KI device prior to invoking ALLWRITE in order to disable KSMPLUS. If you want to re-activate PRO-WAM, then `FILTER *KI *WM`. That will provide the correct filtering for PRO-WAM. You can do the same kind of thing for KSMPLUS after using ALLWRITE. Perhaps with KSMPLUS out of the way, you may have less stack usage. Also, try to avoid invoking the ancillary programs from other than AREA 0 of ALLWRITE. When you use the AREA command of AL to get to a second or third text buffer, AL "permanently" resides that memory bank. This, of course, will work since the DOS has provided all of the memory management routines necessary to automatically reside bank 0 on interrupts, device calls, and disk I/O requests. But that uses additional stack space.

When I developed LS-DOS 6.0, I provided for a full page of memory for the system stack. This allowed for two stacks of 128 bytes each. LSI was "forced" to shrink that to 128 bytes total in order to introduce additional features. The implication is that programs must provide their own stack space if they have more than a few words of stack usage. PRO-WAM switches to its own stack area as soon as it takes control; I provide a 288-byte stack for PRO-WAM in order to support up to the four nested applications capable of being invoked.

One last point about stack use is that it could be possible for a program to be so misbehaved on system stack usage that the upper stack is corrupted by the lower stack utilization even in spite of the switch to 340H as the split point. As previously noted, ALF uses its own stack; AL does not. It would not be problematic to develop a patch to AL which places a stack at 2600H. Although the area from 2400H through 25FFH is reserved for future expansion of the DOS, that utilization is not manifest in the 6.3 release; thus, it could be possible for AL to be patched so that it would use that region for a stack. There may be a number of versions of AL out. Mine is V1.03 at Update Level 11/29/84. The first thing it does is save the current stack contents at 6CF4H via a LD (6CF4H),SP. Using FED2, or similar, look at the last sector for the transfer address record (02 02 xx xx). Make note of the address (xx xx) and find that location. If the code is as mine, then change the 4-byte LD (nnnn),SP instruction to 31 00 26 00 and then change the targeted location of the original instruction to "00 26". This will force AL to use the area immediately below the AL program as a stack. Then test out the program. You may have to also examine Electric Webster for the same kind of stack problems as I don't have that package.

Fm Brad Stiles: Thank you for your prompt reply letter and help over the telephone. I have implemented your suggestious re., modifying Alwrite to alleviate overflow of the system stack and eventual crashes. Specifically, I have applied the following patch:

```
.AL1/FIX: Patch to AL/CMD, version 1.13, dated
10/16/87,
.Relocates the Alwrite program stack to 2800H
.  Eliminates Alwrite's use of the normal system
stack space
. Ref: Letter, Roy Soltoff, MISOSYS, to BT Stiles,
28. 09.88.
. Usage: Patch al/cmd all/fix
D48,CA=31 00 26 00
F48,CA=ED 73 17 6D
D47,68=00 26
F47,68=00 00
. EOP
```

Since applying this patch, the system has performed abolutely flawlessly, The "DEVICE (B)" table has not been clobbered. I have had no further hung computers, etc. Naturally, I am extremely pleased.

One final note. In response to your suggestion, I have investigated the stack usage by Electric Webster. With the help of DEBUG, I have deduced that M/EW (Electric Webster as it loads from Alwrite) does not have its own program stack either. However, it is not necessary to modify M/EW, since it uses the same stack that AL/CMD had used and thus it functions correctly from the H2600 stack space by virtue tf the AL/CMD patch.

Thanks again!!

Desktop Publishing & The Model 4

Some Personal Reflections

Lee C. Rice, Ph.D.

I serve as one of the co-directors of the Marquette Philosophy Department's Microcomputer User Area, one of about a dozen facilities on the MU campus which serve specific departmental needs. The Philosophy User Area has a shared-resource arrangement with a similar facility in the Theology Department; and the two facilities have joint holdings which include about 20 Model 4 systems, 6 MSDOS machines of varying kinds, two multi-pen plotters, a digitizer, about 20 printers, and a large base of software for both programming development and new users. About half of the Model 4 systems are also hard-wired (via multiplexor at 9600 baud) to the university's central system: a VAX cluster consisting of a VAX 8650 and four smaller VAX minis with shared disk and other resources. Many of our Model 4s find frequent use as VAX terminals, and also for uploading and downloading of data files. The central VAX system also offers several on-line word processors, including Digital Standard Runoff, TEX, and TPU; and a variety of devices (laser printers, tape drives, plotters, etc.) are accessible through the VAX system either in batch or on-line mode. I provide these data only to make it clear that, whenever we are considering a particular project which uses computer resources, we are able to consider a rather wide variety of options, and are not restricted to micros or particular software environments.

Two publishing projects affiliated with the department, which had been handled for years using commercial typesetters and printers, are the Mediaeval Text Series (which publishes approximately one volume of 150-400 pages annually) and the Aquinas Lecture Series (one smaller volume annually). Three years ago we added a quarterly journal, *Philosophy & Theology*, to this project list; and also decided, in the interest of economy and of possible expansion of the two series, to change the text preparation to an in-house procedure. Laser printers at that time were available on the central VAX system. In order to test the feasibility and efficiency of in-house document preparation without expending money for additional hardware, we worked for one year on the central system (using TEX, which is a powerful formatter, but fairly difficult to use). At the end of the year, successful trials and a variety of other factors indicated that we should continue in-house preparation, but under a local system of printing and formatting instead of the VAX system.

This decision meant purchasing a laser printer of our own, and placing it on-line to one of our systems. We opted for the Hewlett-Packard laserjet on the basis of reliability and minimal down-time (it still excels in these respects), and then were faced with the question of what formatter to adopt as the "official" standard. In current use on our own systems were LeScript (multiuser license for both TRSDOS and MSDOS versions), Word-Perfect, and a variety of single-user licensed formatters for both Model 4s and MSDOS machines. LeScript is not a particularly powerful formatter, although it remains popular with new users; and tests with Word-Perfect and several other formatters suggested that their promises fell far short of their performance, especially in the laser environment.

Before indicating why we ended up with Allwrite under TRSDOS, I should make a couple of distinctions. The first is between desk-top publishing and routine (academic) publishing. "Desk-top" software seems to be all the rage right now, but its primary claim to fame comes down to the ability to produce short documents with integrated graphics which can be laid out in a screen preview mode. This is the sort of thing businesses often need, but it is not for everyone. Ability to handle graphics and special document layouts was not something we needed, and software which provides these abilities does so at a price: it is usually less efficient, more expensive, and much more demanding in memory and CPU resources. Standard academic publications have a simple format, few graphics, and fewer surprises.

The second distinction worth bearing in mind is between "What-You-See-Is-What-You-Get" processors (WYSIWYG) and simpler text formatters. The WYSIWYG processors must produce layouts on the CRT screen itself, which typically also makes them slower and places heavier demands on memory. Another feature of WYSIWYG processors is that their files are never Ascii in format; and can usually NOT be moved from one host system (micro or mainframe) to another without extensive editing and conversion. This presented a serious defect for us. Even though we wanted to produce our camera-ready copy on a micro system (whether Model 4 or IBM-clone), we did want **portability** for the source files, so that

they could be easily moved among micros and mainframes without special conversions.

Text formatters, as opposed to WYSIWYG processors, typically post-process a file in Ascii form, which contains embedded printing commands, and output to selected devices either on-line or in batch mode. They are a bit harder to work with, since you don't see your final document until it is actually printed; but they are typically fast and portable for file maintenance. If you are doing a great deal of graphics work or format composition on a page-by-page basis (e.g., special newsletters, fliers, etc.), WYSIWYG is definitely what you want. If your formatting needs are simpler and more routine, as ours were and are, Ascii formatters make a lot of sense.

If you are looking for enhanced document preparation facilities, my first advice, therefore, is to decide just what you need. If graphics and special layouts will be a major part of your priorities, you want WYSIWYG. You can then forget the Model 4: its screen controls are simply too primitive, and its native memory too limited, for such software. Incidentally, 8088 IBM clones are a bit better here, but not much. I own a Z-158 home system, running at a clock speed of 12 mgz, with 640K of standard and 2meg of extended memory. I have several WYSIWYG formatters on this home system (which has a 40meg hard drive), including FONTASY (a product by Pro-Soft, the people who produce Allwrite). The screen-formatting is powerful, but the programs are all very slow. I find myself using Pro-Soft DOTWRITER on the Model 4 as a matter of choice where graphic fonts are needed: the MSDOS stuff is a bit more powerful, but nowhere near as fast as DOTWRITER. If you are considering a graphics oriented publication project, my advice would be to consider the minimal system to be an 80286, better yet an 80386 - and best of all an 80386 running under XENIX rather than OS-2 or MSDOS.

If your needs do not include lots of graphics (or if you don't mind doing pasteups where graphics are required), then consider Pro-Soft's ALLWRITE. After an extensive evaluation period of almost a year, we ended up adopting this formatter. Since it is not available for MSDOS, that committed us also to the Model 4. The laser package available as a separate add-on for ALLWRITE (and which, incidentally, is also available for the Model 3 and works quite nicely under LDOS5.3) provides the necessary utilities and font tables for most needs. Supported fonts include both Times-Roman (sizes 6-30 point) and Helvetica (same sizes), as well as all of the HP internal fonts. These fonts are available in regular, bold, and italics. Hewlett-Packard does not support bold italics, so the Allwrite people thoughtfully added an internal Allwrite command to produce that as a fourth font.

Also an add-on, but one worth the small purchase price, is a set of programmer utilities for font management and for the creation of special font table files for customized use. Allwrite supports just about every capability of the standard laserjet or laserjet-plus. Rough copies can be printed on just about any standard dot matrix printer also. This is an important consideration. Laser printers do not have ribbons, but the cartridges and toner packs run anywhere from $100 to $200 depending on brand.

Will Allwrite do anything that you could possibly want to do with a laser printer? No, and the Pro-Soft people make no such claim for it. If you want a text formatter which can literally do anything, then EROFF is probably the best choice. This is a full port of the UNIX TROFF text formatter, and is available for a variety of minicomputers and also for MSDOS systems. We purchased this for our Zenith PC-clone which is on-line to the laser printer. It is rather expensive ($695, versus about $200 for Allwrite), but provides you with 10 full disks (double-sided) and over 800 pages of documentation, not to mention a macro library containing over 400 built-in macros. EROFF, it should be mentioned, is a formatter just like Allwrite. It does NOT provide an editor, and expects Ascii source code produced any way and any where you want. It provides integration with graphics, special fonts, statistical libraries, and also a host of utilities for doing anything you could ever want to do.

On the down-side, EROFF is slower than Allwrite, and its syntax is also a bit clumsier. Both systems have been available now in our User Area for quite some time. 90% of our projects are done under Allwrite, which is easy, fast, and friendly. The remaining 10% are done under EROFF, which is powerful and virtually complete. Note the advantage of the Ascii source files expected by both Allwrite and EROFF. Users can write, proof, and correct their files on ANY computer (mainframe or micro) under ANY editor (which generates plain Ascii files) using ANY DOS. When it's time to print, the source is simply ported to TRSDOS (for Allwrite) or MSDOS (for EROFF) systems.

If you are looking for something in the MSDOS world which is easier to use than EROFF, consider Microsoft WORD. Unlike most of the systems which claim to support laser printers, but do so only incompletely (Word Perfect is a good example), WORD is quite powerful, supports 90% of the HP fonts, and also runs at a credible speed if you turn OFF the WYSIWYG screen graphics. WORD is also bug-free so far as we have discovered, and the successive versions of it from Microsoft are good evidence of this company's intention to stay with it and support it with future enhancements.

If what you have is a Model 4, and are wondering about laser printers, you need not be ashamed of TRSDOS or the Model 4. Allwrite is a formatter more powerful than almost anything currently available even in the IBM world, and it is easier to use than anything that comes close to it in power on **any system at all**. On a personal note, I have a home workroom which contains a Model 3, Model 4, and a Zenith Z-158 system (all with hard drives, all in frequent use).

I purchased last year a Tandy LP1000 laser printer (which emulates a Hewlett-Packard laserjet). I had a technician build me a parallel selector box which would enable connection of all three micros to the laser printer. I have purchased personal copies of WORD and a host of other MSDOS software, as well as personal copies of Allwrite for both LDOS and TRSDOS6. I still do 90% of my laser printing using Allwrite from the Model 4. It's fast, it's efficient, and it's easy; and that is about the best thing that can be said for any text formatter, regardless of DOS or hardware.

My wish-list for the future includes an 80386 system with an 80meg hard drive, 20mgz CPU, and XENIX. Then, and only then, I'll consider WYSIWYG formatters seriously. The 8088 just doesn't have the guts, and MSDOS is a step backward from LDOS/TRSDOS, hardly a step forward.

For more information on Allwrite (or on FONTASY for MSDOS), contact PRO-SOFT at: P.O. Box 560, North Hollywood, CA 91603. For more information on EROFF, write to: Elan Computer Group, Inc., 410 Cambridge Avenue, Suite A, Palo Alto, CA 94306. I can be reached at Marquette University (Dept. of Philosophy, Milwaukee, Wisconsin 53233), or via BITNET to 6802ricel at MUCSD.

MSDOS

MS-DOS Modems & interrupts

Fm Shane Dawalt: I had a rather nasty problem when I installed my mouse (LOGITECH bus mouse). My modem was on COM2 and COM1 is reserved for my external communication link. Hence I needed COM3 or COM4. The mouse's board supports all the com channels so I placed it in COM3. The mouse worked fine as did the modem. Unfortunately, I couldn't read a single character from COM1.

After talking with the Zenith repair, they told me that there is not a standard for COM3/4 accessing. Therefore, any manufacturer can use "any" interrupt line they want. This will cause adverse operation of the respective COM port. After that, I decided the $45 for the com board was worthless.

Fm Joe Kyle-DiPietropaola: Shane, That's why I recommend bus rodents, you'll never ever have more than two serial ports that work "generally" with software. More are easy with custom software support, but the vast majority of available software will barf.

Fm: H. Brothers: Shane, The problem with COM3/4 is that IBM's designers didn't figure out how to share an interrupt until the MCA (something which the Model I designers knew well). Even if you have 4 COM ports, only two can be active at once (and they must use different IRQs because the IRQ's on the PC/AT bus are edge-triggered and it is impossible for driver software to know who is requesting service. Generally, COM1 & 3 share one IRQ, and COM2 & 4 share the other (although you can often set that with jumpers). The possibilities of I/O port conflicts further complicates the situation, especially since IBM crippled the 8088's I/O capabilities severely and made it difficult to find unused ports (hmmm -- similar to the way Tandy severely crippled the Z-80's interrupt capabilities, I guess).

Fm Shane Dawalt: I still think that having a modem and a bus mouse sharing COM2 is still bizarre. What's even worse is the fact that they actually WORK! Note I'm not complaining, but is does seem ridiculous. It is obvious Zenith repair knows more than I do as they said it would work from the start. Seems this would be function dependent however.

So only two ports may be active at once. Strange. When you're talking to, say, port 3, then port 1 cannot be accessed via interrupts. Evidently, that's why I could write to port 1 but couldn't read from it. Well that was simple enough. I wonder if it would be possible to move my comm link to COM3 and move everything else down. I've got some generic C code to handle COM1-4 I/O. All I need is to specify the appropriate channel.

Fm Joe Kyle-DiPietropaola: Shane, You can **sometimes** share IRQs, it is entirely dependent on how "smart" the programs are. Most aren't.

Fm Daniel L. Srebnick: Shane, The problem is with the sharing of interrupts. COM1 uses IRQ4 and COM2 IRQ3. Some boards with COM3 and 4 then try to reuse IRQs 4 and 3 respectively. So, you have COM1 and COM3 in contention for the bus; the same for COM2 and COM4. This is the reason that intelligent controllers such as the DigiBoard are used in multi user Unix systems to support multiple terminals. The communications board itself must manage interrupt contention.

Fm H. Brothers: But leaving alone the question of chaining to the previous owner of the ISR, how does a driver know which device signalled the IRQ? Some devices may have status registers that can be checked, but that is entirely device-dependent. I think it would take smart software, the right devices, and a specific order of loading the device drivers with the smartest ones loaded last.

MSDOS shells

Fm Shane Dawalt: Hardin, Quick question; Do you know if any books exists on how to write a shell for MSDOS? I have an idea and it could be done without using a shell, but I believe a shell implementation would be the easiest. Do you know if using C for shell writing is acceptable or not? I finally got a book on the 80286 for assembly language, but I'm not

comfortable with assembly on the 80286 yet and, besides, I don't have an assembler yet!! (Suppose I could hand assemble and poke the code in DEBUG ... ARRRGGGGG!!!)

Fm Michael Stein: Shane, The best introduction I know of to MSDOS internals and the requirements of a shell is to be found in Advanced MSDOS by Ray Duncan, published by Microsoft Press. Certainly could be written in C.

Fm H. Brothers: Shane, Any program you can create in .EXE or .COM format can be a shell. Simply use the Shell command in Config.Sys, and the program will be installed instead of Command.Com. Ray Duncan's "*Advanced MS-DOS*" (a must-have book!) has a simple shell example. Certainly, you could write it in C, QB, assembler, Pascal, almost anything (but have a floppy available to boot from so you can get to Config.Sys and remove the shell). Some shells, btw, are nothing more than programs that run on top of Command.Com as normal applications.

About assemblers: you did know that Debug has a simple assembler built into it, didn't you? Also, I think there is one you could start with in IBMSW. If you want a high-quality assembler, I'd suggest two to think about: MASM 5.1 from Microsoft (includes a decent editor) and OPTASM from SLR Systems.

Fm Bryan Headley: Shane, How about Ray Duncan's Advanced MS-DOS book? Throughout, he documents the DOS calls and quietly puts together a command interpreter. Also good is Holub's book On Command, from Dr. Dobbs. The latter comes with the shell Holub wrote...

Don't like assembler? Sounds reasonable. Go get a C compiler, then. QuickC and Turbo C are right bargains of the day.

Fm Shane Dawalt: Oh? What about disk operations? Doesn't Command.Com take care of that (I'm referring to opening and closing of files or what about specifying the path. Also, doesn't Command.Com take care of the environment variables, i.e., read them in and set them in memory?) Anyway, I've been pointed to Ray's book by Michael (the last reply). I'll certainly be peeking under books and on top of book shelves for it.

Yup, I have MASM 5.whatever on my list. ZENITH is selling it, so I'll probably get it through them. Last I checked, Zenith's programmer's pack was $168.75. 'Course, that was before the current MASM 5 series. It may have went up, down or done a curly-Q. And no, I didn't know Debug had a simple assembler. I haven't been reading that part of the MSDOS manual yet. I assume OPTASM from SLR Systems is an optimizer for ASM source. (As if my code isn't optimal already!<grin>)

I have been running Turbo C and I've been playing around with the spawn functions. Now, without getting into the nitty-gritty, I was under the impression that child processes which are spawned can only be of the .COM file type. Can they also be of the .EXE file type??? Are there any memory restrictions on child processes (except the obvious restriction that they not attempt to use memory which is not available).

Fm Joe Kyle-DiPietropaola: In addition, note that you can get a book and disk with sample source code for a shell already written in 'C' by Allen Holub (used to be with Dr. Dobb's), available from M&T Books. Check any recent issue of Dr. Dobbs for more info.

Fm jeff brenton: Shane, Turbo's spawn functions simply call DOS, so everything DOS supports can be spawned. BAT files, of course, require a copy of COMMAND.COM be spawned to run them, but both .COM and .EXE are supported otherwise.

OPTASM is a MASM-compatible assembler that claims to be 4-10 times faster than the fastest version of MASM available. It does NOT optimize assembly code, as that would be considered a bug!

Of course, 80x86 assemblers **ARE** free to rearrange certain instructions at will...

Fm H. Brothers: Shane, The word "shell" has several different meanings in the MS-DOS world, which seems to lead to loads of confusion. It can mean a replacement for Command.Com, a program that runs as a user interface on top of Command.Com, and also, as a verb, the process of invoking a new copy of Command.Com from inside an application.

In one sense, Command.Com is itself nothing but a weak shell. Most of the insults hurled at MS-DOS by users (as opposed to programmers) should really be directed at Command.Com. It prints the A> prompt, interprets user commands, maintains a "master" copy of the "environment" strings, and runs, as child processes, all of your other applications. That's the important part: Command.Com simply spawns everything else you run.

To reduce its memory requirements, Command.Com does some strange things like loading part of itself at the very top of memory. Some of it stays memory resident at all times (like any other parent program); some of it stays in high memory. If an application overwrites Command.Com's high memory section, Command.Com must reload itself when it regains control from a child. That's what the Comspec= environment string is for.

Command.Com interprets everything you type or every line in a .BAT file. First it looks to see if the command is in its own internal command list (things like Copy, Dir, Set, etc.) If not, it starts searching through the path (an environmental string that it has control of) looking for a .Bat, Com, or .Exe file that meets your command. If it finds a .Bat file, it does the interpreting. Otherwise, it simply spawns your app as a child.

The important point is that Command.Com is nothing more than an application. It has more knowledge of "undocumented" DOS features than most programmers, but otherwise it is nothing special.

So ... it is not exceptionally difficult to write a replacement for Command.Com, and there are many good reasons for doing so. Some folks simply disable some Command.Com commands so that users can't do things like Dir or Type for security reasons (or have to run a custom .Exe or .Com program instead of Command.Com's built-in function). Some folks replace Command.Com entirely. Some people run shells above Command.Com, which, once they start, never let the user see Command.Com at all.

File access, directory control, etc. are part of MS-DOS, not command.com. Certainly, when you type CHDIR xyz, Command.Com interprets what you have done and then invokes the DOS commands (usually through Int 21h) to change the default directory.

It's important to realize that anything Command.Com can do, your programs can do also. Every program has a copy of the environment for example and can create a new environment for its children (which is what Command.Com does, although its copy is called the "master" environment). Every program can spawn any .Exe or .Com program including a new copy of Command.Com. Any competent programmer could write a better batch interpreter than Command.Com's and include it as part of a new shell.

If you want to write a new shell to replace Command.Com, it has to be able to provide the interface that programs expect: keeping an environment, spawning processes, interpreting .BAT files and user commands.

But other than that, it can do almost anything you wish. Do you want to give MS-DOS the "look and feel" of LS-DOS? Simply write a shell which implements the commands in LS-DOS's LIBrary. Do you want your DOS machine to act like a Mac? Use Windows as a shell. Do you want Command.Com to be more forgiving of typographical errors in a command line? Simply replace or supercede it with a shell which can handle typos (neat algorithms for doing so in the latest Dr. Dobbs and AI Expert). Do you want your computer to ONLY run WordPerfect? Simply replace Command.Com with a small shell which only and always invokes WordPerfect. You could designate WP as the shell program itself, but then there wouldn't be any environment strings set up for it to find its dictionary and macros. Besides, Command.Com doesn't receive any environment from MS-DOS since it is in charge of maintaining the environment itself.

BTW, spawn() should be able to spawn ANY .EXE or .COM program assuming enough memory space. It simply calls the same MS-DOS EXEC function which Command.Com uses and, optionally, sends a new environment to the child process.

The Tower of Babel

Writing Interactive RATFOR/FORTRAN Programs

by Jane A. Layman

Much as I like poring over virtually every word in *The MISOSYS Quarterly*, I find that my favorite programming languages are grossly under-represented in its pages. This article comes with a sample program, DESCRIP, to demonstrate the techniques discussed in the text. The techniques work equally well with the Model 4 and MS-DOS versions of MISOSYS' RATFOR. I use the Model 4 version myself. An MS-DOS version of the RATFOR source (file name DescripM /RAT) is included. It has been compiled and successfully run under MS-DOS by a friend with an MS-DOS machine and the MS-DOS version of RATFOR.

The following files make up this example:

DESCRIP/RAT: The Model 4 RATFOR source for the sample program. **DESCRIP/CMD**: The Model 4 executable program. **DESCRIPM/RAT**: The MS-DOS RATFOR source for the sample program. **DESCRIP/HLP**: Instructions for using the sample program. **CLS/MAC & /REL**: A "bonus" for Model 4 users. CLS/MAC contains the (M80 syntax) source code for LS-DOS' CLS library command. /REL is the relocatable object code that can be linked with the DESCRIP(other)/REL code to clear the screen any time the module is called. Instructions for adding it to DESCRIP are included in CLS/MAC source. **DUMMY1 & 2/DAT, /MAP**, and **/OUT**--very short dummy data files for trying out the program. The DUMMY2 set is identical to DUMMY1 except that the last 3 variables were added to test the DESCRIP program's error trapping.

I hope the information in the article will be of interest to *TMQ* readers and that some of them, at least, will have RATFOR FORTRAN programming tips of their own to share.

Jane A. Layman
20165 Davidson Road
Waukesha, WI 53186

When it comes to speed and efficiency in number crunching, FORTRAN remains the star performer among computer languages. The addition of RATFOR preprocessing raises FORTRAN to new heights, particularly in the area of program structuring. One's RATFOR/FORTRAN programming becomes even more enjoyable when one master's the art of writing interactive and reusable FORTRAN programs.

The program that accompanies this article (DESCRIP/RAT written in the Model 4 version of RATFOR) contains a number of features that allow the user to pass information to the pre-compiled program at run time. This information includes input and output file names, labels for the data, AND the format statement by which the data file is to be read. In addition, the program contains a routine to structure numerical output in an approximation of a BASIC PRINT USING statement.

The concepts in this program will work equally well in both the TRS-80 and MS-DOS environments. MS-DOS users will need to make changes to the syntax of the I/O statements to convert the program to MS-DOS use. LS-DOS users, whose I/O statements are subjected to LRL (logical record length) checking by the operating system, may wish to consider applying the patch discussed below to their working RATFOR/FORTRAN system disk(s).

LS-DOS LRL Checking

As a sometimes researcher who makes considerable use of RATFOR/FORTRAN, I have had abundant occasions on which to create ASCII data files of an LRL (logical record length) other than the system default of 256. When attempting to view or edit these files in a text processor, I was prevented from doing so by LS-DOS's LRL checking when opening a file. Copying the file back and forth between the default LRL and its actual one, or zapping and rezapping the correct byte in the directory entry, or just switching to a text processor running under LDOS (which does no such checking) got to be a pain. Ergo, I examined "The Source" (Vol. I, p. 161) for a way to bypass this error checking without interfering with the DOS in any other way.

The fix I came up with can be installed as follows:

```
PATCH SYS2/SYS.LSIDOS (D02,C6=DD BE
09:F02,C6=FD BE 04)
```

With this patch installed the DOS now compares the LRL value it has just placed in the File Control Block with itself rather than with the LRL listed in the directory and, never finding a mismatch, goes merrily on its way. To set up an ASCII file of any desired record length, I now CREATE the file with one record of the desired LRL, load the file into a text processor, and edit and extend the file at will.

The DESCRIP Program

The DESCRIP program presented here computes descriptive statistics (Mean, Standard Deviation, etc.) for anywhere from 1 to 60 variables for each run. It will read up to a 1000 line formatted data file in which each line contains the 1 to 60 observations for each case. [Instructions for redimensioning the program for even larger data sets are included in the source.] Zero or negative observations are treated as missing data and deleted from the computations. [The reader could incorporate a different missing value routine into the source code and/or add a prompt that would allow for the specification of missing values at run time.] Output is to a disk file specified by the user.

After reading the data file and completing the initial computations, the DESCRIP program conducts error trapping for too few observations and zero variance prior to attempting the final calculations. RATFOR's ability to correctly interpret IF . . . ELSE code blocks nested within larger IF . . . ELSE code blocks is a tremendous asset for constructing this type of code.

Incorporating Data Specifications at Run Time

The program uses a combination of keyboard and "Map File" input to pass user information at run time. Keyboard prompts ask for file names [Use your operating system's syntax for specifying them.], the LRL of the data file (for LS-DOS users), and the number of variables in the data set. The program makes use of a formatted "Map File" which contains the FORMAT statement of the data file and labels for each of the variables in the set. [Additional information about this file is contained in a comment block at the beginning of the source code.]

The FORMAT statement and file names are read into byte arrays; labels are read into a double precision array. To OPEN a given file, the array containing the file name is specified in the CALL OPEN statement in place of the file name itself. E.g., CALL OPEN(6,OUTFIL,0). The name of the array is NOT enclosed in single quotes. You want the FORTRAN compiler to interpret it as a variable, or variable location, NOT as a literal.

Similarly, the FORMAT statement is stored in the first record of the "Map File", minus the syntax provided for the compiler's information--the word FORMAT and a statement label. The FORMAT statement itself consists of field specifications which begin and end with parentheses. When you want FORTRAN to use the FORMAT statement (stored in the DESCRIP program in the FMT byte array), the name of the array in which the format is stored is used in the READ (or WRITE) statement in place of the statement label.

While the program presented here uses a separate disk file in which to store the FORMAT statement, this is not a FORTRAN requirement. FORTRAN would input both the format and label information via the keyboard in the same manner as it accepts the file names and numeric information. The program could also have been written to read the FORMAT statement from the first record of the data file, provided the user placed it there in advance of running the modified program. In fact, this would be the preferred approach for programs that have no need of extras such as variable labels.

Reducing Decimal Places to Manageable Proportions

When the number crunching is complete, the number of significant digits that were desirable at the computation stage can seem a liability at the output stage. The ability to format numerical output in something akin to BASIC's PRINT USING statement would simplify matters considerably. FORTRAN can be persuaded to suppress any number of unwanted digits during output, provided the unwanted digits have first been set to 0.

The sample program includes a routine to limit output to the first 3 or 4 digits after the decimal place for 5 separate REAL arrays. I.e., the results are printed in F12.4 or F11.3 format without any runtime **FW** (format width) warnings. Setting the unwanted decimal positions to 0 is accomplished in the following manner. Each completed computation is first multiplied by 1,000 or 10,000. [In your mind's eye, move the decimal point 3 or 4 places to the right.] The variable is then truncated at the new decimal point by the AINT (as opposed to the INT or IDINT) FORTRAN function. Finally, the decimal place is restored to its original position by dividing the variable by the original multiplier. You can truncate fewer or more decimal positions by adopting fewer or more zeros in the multiplier/divider. Rounding up rather than truncation can also be achieved if one uses the "trick" suggested for the Model III version of BASIC, i.e., by adding .5 to the multiplied value prior to using FORTRAN's AINT function. E.g., MEAN(I)=MEAN(I)*10000 becomes MEAN(I)=(MEAN(I)*10000)+.5.

The Finishing Touch

When the DESCRIP program has completed its initial run, the user is asked whether or not he/she wishes to rerun the program (Y/N). The user's alphabetical response [Do NOT enclose the keyboard response in single quotation marks.] is stored in a single BYTE variable.

While FORTRAN inputs the response as a literal (format A1), it does so by converting it to a number in the range of -127 to 128, a set of numerical values that encompasses the ASCII character set. FORTRAN will not understand what you mean if you later attempt to reference the byte's contents with a statement such as: IF (RERUN.EQ.'Y'.OR.RERUN.EQ.'y'), but it will behave as you want if you substitute the numerical equivalents of the ASCII characters in the IF statement. In short, once one learns how to speak RATFOR/FORTRAN'S language, it can become very amenable to one's requirements.

DESCRIP computes the Mean, Variance, S.D., Min and Max for up to 60 variables. 0 or negative values are treated as missing data.

Prior to running DESCRIP, a "Map File" must be prepared. This file should have a logical record length of 81 (80 characters per record, including blanks, and a carriage return in column 81). Record number one of the MAP file should contain the format of the data file starting in column one.

```
For example:    (5X,6F2.0,3X,20F3.2)
```

Use F fields to input both integer and real data into the X array. [G or E fields will also work if you prefer either of their input characteristics.] Records 2 through 7 of the MAP file should contain room for 60 variable LABELs, 10 per record in A8 format. All 6 records should be present. (Optional) records 8 through n can be put to any purpose the user finds appropriate. While running DESCRIP, you will be prompted by the program to type in the following information:

The name of your data file
The LRL of this file [LS-DOS users only]
The number of variables for this run
The name of your MAP file
A file name under which to write the program's disk output.

```
# ************************************************
# DESCRIP/RAT--This program computes descriptive
# statistics (Mean, Variance, Standard Deviation, N,
# Min and Max for set of up to 60 variables.  When
# the program encounters a 0 or negative value on an
# item, it assumes the data is missing and deletes
# the observation.  Program output goes to a
# user specified disk file.
# ************************************************
# This program compiles with over 29,000 bytes free.
# To dimension for a larger variable set, increase
# all but the BYTE array variables by the same
# constant.  In the event you wish the program to
# read more than 1000 cases, extend the upper limit
# on the DO I=1,1000 loop.
# ************************************************
# N.B.:  This program is intended for use with a
# carefully structured "Mapfile" with an LRL of 81.
# The file should be structured as follows:
#  1) The format of the data file (including the
#     beginning and ending parentheses) should begin
#     in the first column of record number 1
#     and may be up to 80 characters including the
#     parentheses.  Use F fields to input both
#     integer and real data into the X array.
#  2) Records 2 through 7 of the "Mapfile" should
#     contain room for 60 variable LABELs, 10 per
#     record in A8 format.  All 6 records should
#     be present. (Optional) records 8 through n can
#     be put to any purpose the user finds
#     appropriate.
# ************************************************
INTEGER I,J,K,NX,NS(60),LRL
REAL X(60),XSUM(60),X2SUM(60),MEAN(60),
VAR(60),SD(60),MIN(60),MAX(60)
DOUBLE PRECISION LABEL(60)
BYTE MAPFIL(23),INFILE(23),OUTFIL(23),FMT(80),RERUN
# Title and prompts for keyboard input
WRITE(3,30)
5 WRITE(3,40)
READ(3,90) INFILE
WRITE(3,50)
READ(3,100) LRL
WRITE(3,60)
READ(3,100) NX
WRITE(3,70)
READ(3,90) MAPFIL
WRITE(3,80)
READ(3,90) OUTFIL
# Open Mapfile and read in info.
CALL OPEN(6,MAPFIL,81)  # N.B.: Unless the Mapfile
READ(6,110) FMT         # conforms to the specifi-
READ(6,120) LABEL       # cations above, this block
ENDFILE 6               # of code CANNOT run.
# Initialize program accumulators; MIN array set
# arbitrarily high; remaining
DO I=1,NX                       # variables set to 0
   {
   NS(I)=0
   XSUM(I)=0.0
   X2SUM(I)=0.0
   MIN(I)=5000.0
   MAX(I)=0.0
   }
# Read data file and perform initial calculations
WRITE(3,200)
CALL OPEN(6,INFILE,LRL)
DO I=1,1000                     # DO I=1,1000 loop
   {                            # READ statement
   READ(6,FMT,END=10) (X(J),J=1,NX)
   DO K=1,NX
      {
      IF (X(K).GT.0)      # Check for missing values
         {
         IF (MIN(K).GT.X(K)) MIN(K)=X(K)
         IF (MAX(K).LT.X(K)) MAX(K)=X(K)
         NS(K)=NS(K)+1
         XSUM(K)=XSUM(K)+X(K)
         X2SUM(K)=X2SUM(K)+(X(K)*X(K))
         }
      }
```

```
   }
10 ENDFILE 6
WRITE(3,210)
# Compute descriptive statistics; includes error
# trapping for too few observations for a variable
# and zero variance.
DO I=1,NX
   {
   IF (NS(I).GT.1)
      {
      MEAN(I)=XSUM(I)/NS(I)
      VAR(I)=(X2SUM(I)-
(NS(I)*(MEAN(I)*MEAN(I)))))/(NS(I)-1.0)
      IF (VAR(I).LE.0)
         {
         SD(I)=0.0
         }
      ELSE
         {
         SD(I)=SQRT(VAR(I))
         }
      }
   ELSE
      {
      MEAN(I)=XSUM(I)
      VAR(I)=0.0
      SD(I)=0.0
      MIN(I)=XSUM(I)
      MAX(I)=XSUM(I)
      }
   }
# Routine to limit REAL variable output
# to 4 decimal places
DO I=1,NX
   {
   MEAN(I)=MEAN(I)*10000
   MEAN(I)=AINT(MEAN(I))
   MEAN(I)=MEAN(I)/10000
   VAR(I)=VAR(I)*10000
   VAR(I)=AINT(VAR(I))
   VAR(I)=VAR(I)/10000
   SD(I)=SD(I)*10000
   SD(I)=AINT(SD(I))
   SD(I)=SD(I)/10000
   MIN(I)=MIN(I)*1000
   MIN(I)=AINT(MIN(I))
   MIN(I)=MIN(I)/1000
   MAX(I)=MAX(I)*1000
   MAX(I)=AINT(MAX(I))
   MAX(I)=MAX(I)/1000
   }
# Write out results
CALL OPEN (6,OUTFIL,0)          # Output file
WRITE(6,300)
DO I=1,NX
   {
   WRITE(6,310)
LABEL(I),NS(I),MEAN(I),VAR(I),SD(I),MIN(I),MAX(I)
   }
ENDFILE 6
WRITE(3,220)                                # Job done
WRITE(3,230)
READ(3,240) RERUN
IF (RERUN.EQ.89.OR.RERUN.EQ.121) GO TO 5
# 89 & 121 = the decimal equivalents of ASCII
# upper & lower case 'y'.
STOP
# Formats for Title and keyboard prompts and input
30 FORMAT(26X,'DESCRIPTIVE STATISTICS'//28X,'by Jane
A. Layman'/)
40 FORMAT('0','Data File name (up to 23 characters):
')
50 FORMAT(1X,'What is the LRL of this file?  ')
60 FORMAT(1X,'How many variables for this run?  ')
70 FORMAT(1X,'Map File name (up to 23 characters):
')
80 FORMAT(1X,'Output File name (up to 23
characters):  ')
90 FORMAT(23A1)
100 FORMAT(I4)
# Mapfile formats
110 FORMAT(80A1)
120 FORMAT(10A8)
# Screen message formats
200 FORMAT('0','Reading data file'/)
210 FORMAT(1X,'Initial calculations complete'/)
220 FORMAT(1X,'Job done.'/)
230 FORMAT('0','Rerun program (Y/N)?  ')
240 FORMAT(A1)
# Output file formats
300 FORMAT(26X,'DESCRIPTIVE
STATISTICS'//'Variable',7X,'N',8X,'Mean',_
 4X,'Variance',8X,'S.D.',4X,'Minimum',4X,'Maximum'/)
310 FORMAT(A8,I8,3F12.4,2F11.3)
END
```

Sample data

```
Person A 4 3 3 4 3 4
Person B 3 4 4 3 3 4
Person C 2 1 1 2 2 3
Person D 1 1 2 1 3 2
Person E 4 3 4 3 0 3
Person F 2 2 1 2 4 2
Person G 1 1 2 2 3 1
Person H 3 3 4 4 2 4
Person I 4 3 4 3 4 3
Person J 2 2 1 2 1 2
Person K 1 1 2 2 1 2
Person L 3 4 3 4 2 4
```

Sample MAP file (note: each data row has four additional columns of "UNUSED")

(8X,6F2.0)

```
ITEM 1  ITEM 2  ITEM 3  ITEM 4  ITEM 5  ITEM 6
UNUSED  UNUSED  UNUSED  UNUSED  UNUSED  UNUSED
UNUSED  UNUSED  UNUSED  UNUSED  UNUSED  UNUSED
UNUSED  UNUSED  UNUSED  UNUSED  UNUSED  UNUSED
UNUSED  UNUSED  UNUSED  UNUSED  UNUSED  UNUSED
UNUSED  UNUSED  UNUSED  UNUSED  UNUSED  UNUSED
```

DESCRIPTIVE STATISTICS

Variable	N	Mean	Variance	S.D.	Minimum	Maximum
ITEM 1	12	2.5000	1.3636	1.1677	1.000	4.000
ITEM 2	12	2.3333	1.3333	1.1547	1.000	4.000
ITEM 3	12	2.5833	1.5378	1.2401	1.000	4.000
ITEM 4	12	2.6666	.9696	.9847	1.000	4.000
ITEM 5	11	2.5454	1.0727	1.0357	1.000	4.000
ITEM 6	12	2.8333	1.0606	1.0298	1.000	4.000

FORTRAN/RATFOR

Complex data types

Fm Mark Mueller: HELP! Does anyone know how to implement the COMPLEX data type in Microsoft (Model 4) FORTRAN? My version is 3.44.00. The docs say "may be included in a future release". HA HA HA. Seriously, any help will be appreciated. (p.s.: If we can do this, I'll be on the phone tomorrow to order RATFOR!)

Fm Jim Beard: Mark, You needn't offer to buy RATFOR to get an answer to a question, pal. If I can stop grinning long enough, I'll try to help. Ahem.

Using regular FORTRAN stuff, I have implemented the COMPLEX data type using double statements for add/subtract, etc., and using function and subroutine calls. F80 is VERY efficient if you use three or less arguments, which is possible with complex arithmetic. You can use subroutines for multiplication and division, if it will save a lot of code. The calling sequence is so fast compared to the floating point arithmetic (even if you had an 8087 on your Model 4) that using a subroutine is OK for speed and efficiency. And, if you get it right once, it is always right.

With RATFOR, you can define macros to perform your complex arithmetic. The compiled code will be as efficient as you can get.

Even with FORTRAN compilers that support the type COMPLEX, I usually implement complex arithmetic this way. I find it more natural than the use of the CMPLX, REAL and AIMAG intrinsic functions to go back and forth.

Fm jeff brenton: Mark, You might do it as it is usually done in C - an array of two reals, with functions to operate on the arrays. In fact, I seem to recall reading a FORTRAN tutorial years ago, that said that was the SAFEST way to use COMPLEX numbers in a "portable" FORTRAN program, simply because not every compiler handled them properly, and some just didn't [as you know!].

Fm Mark Mueller: Jim, Well... it wasn't really a desperation ploy (yet), but I have a FORTRAN program that the FCC uses to calculate space & ground loss for AM signals that uses COMPLEX extensively. It's not a real big program and should fit nicely in my model 4. Trouble is I CAN'T USE IT!! If you could pass along the info I would be eternally grateful, and go dust off my FORTRAN books and get to it!!

And, yes, I'll buy RATFOR, for which I have had no pressing need until now. (grin). I'm even willing to pay for the info (well, not TOO much). Let me know if we can work something out, ok?--mark (tired of doing it the hard way)

Fm Jim Beard: Mark, I'll have to work up some stuff and test it before answering. Look for COMPLEX.FOR and COMPLEX.RAT uploads soon. Your biggest weapons are that you can look for the COMPLEX statement at the beginning of each subprogram and do globals on the variable names. For now, I'll chance it with a real-time program (I may regret this later):

```
subroutine cmult(m1,m2,p)
real m1(2),m2(2),p(2)
p(1)=m1(1)*m2(1)-m1(2)*m2(2)
p(2)=m1(1)*m2(2)+m1(2)*m2(1)
retr
end
subroutine cdiv(num,den,ratio)
real num(2),den(2),ratio(2),mod2
mod2=den(1)**2+den(2)**2
ratio(1)=(num(1)*den(1)+num(2)*den(2))/mod2
ratio(2)=(num(2)*den(1)-num(1)*den(2))/mod2
return
end
```

On ANSI bugs

Fm Jim Beard: Here is my nomination for an ANSI bug:

Fm Joe Kyle-DiPietropaola: No go, bugs gotta have six legs. Or did they leave that out of the ANSI spec?

Fm Jim Beard: The ANSI draft spec leaves the number of legs to the implementation. Some bugs have no legs, some have hundreds.

Fm Joe Kyle-DiPietropaola: Ah, I found the problem. I had

```
#define BUG INSECT
```

in my code.

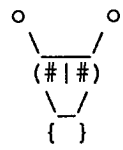
Fm Jim Beard: I would check out a listing of <stdio.h> and make sure that INSECT wasn't a macro defining a STREAM.

Fm Joe Kyle-DiPietropaola: You were right, it was a combination of that, the previous #define and an

```
assert(BUG == SIXLEGS);
```

P.S. Your bug is missing the fiendishly menacing pincers traditional in the "classic" 'C' bug. Or did they drop those from the ANSI draft too?

Fm Jim Beard: The pincers are left to the implementation.



C and standard files

Fm Shane Dawalt: Les, I'm thrashing about in a problem which I can't seem to find any reference too in any book I read. What is the storage type of the standard device names, i.e., stdin, stdout, stderr? I have had the notion these are integers. Are they? Reason why I'm asking is, I'm trying to redirect files inside the program. I'm using handles. What I'm trying is this:

```
int handl;
 handl = open(filename, O_RDONLY|O_TEXT);
    /* Lets assume it's ok */
  dup2(handl,stdin);
  close(handl);
```

As I understand it, I open the file and get handl then I do the dup2() which will automatically close stdin and dups handl onto stdin. Finally I close handl to freeup the file space. Unfortunately, the compiler gets quite angry at the notion of using "stdin" as a handle. It insists this is a type mismatch error. The function prototype of dup2() is: `int dup2(int oldhandle, int newhandle);` If I use dup() instead of relying on dup2() to close stdin for me, then I must close(stdin) myself. The compiler complains about the type mismatch on this too. Have I missed something somewhere??? I was always under the impression that the standard device names were actually handles and, thus, were integer types.

Fm Joe Kyle-DiPietropaola: Shane, The stdin, stdout, stderr names are streams of type FILE * rather than int file handles ala open, dup2, close. fdopen() can turn a handle into a stream, but I think that you should just look at freopen().

Fm Les Mikesell: Shane, stdin, stdout, and stderr (and the others peculiar to msdos) are of the same type as FILE *. You can either use the integer values 0, 1, and 2, respectively (some compilers define STDIN, STDOUT, etc. as the int fd values), or use fileno(stdin), or use the freopen() function instead of doing the dup() and fdopen() yourself.

Fm Shane Dawalt: Well Joe, I'm have a rather nasty problem with the data being buffered when I don't want it to be. I'm talking to a printer device. When I send it initialization information, I MUST be sure it was dumped to the hardware and not lounging around in a buffer. Streams imply an internal buffer. I was under the impression that handles, and assorted functions, are nearer the system level. I reasoned, then, that there is less chance of buffers getting in the way if I use handles rather than streams. Also, I'm attempting internal redirection ... so handles make a little more sense than streams at the moment. I'll tackle streams at a later date. <sigh> Combine that with the fact that I'm not comfortable with dup() and dup2() so I'm sorta programming in the dark. Then, I had convinced myself the standard device names were handles rather than streams. I can really throw myself in the toilet and flush at the same time can't I??!!

Fm Joe Kyle-DiPietropaola: That should be fflush() rather than a simple flush.

Fm Shane Dawalt: Les, Thanks for the reply. Some bird told me that ... evidently wanting to confuse me. I checked with the supplied grep program (which took me a while to choke down the syntax) and couldn't find any other STDIN defines anywhere. I placed some in the STDIO header so if I need them later on, I can use them.

As for freopen(), stream I/O is buffered. My application (which dumps initialization info to the printer) must not be buffered. If I used block I/O with the handles, it is not buffered, correct? If the user wants to direct output to the printer, a printer setup string is sent ... but the string MUST be output WITHOUT buffering. This buffering business is what's getting in the way.

Fm Les Mikesell: Shane, Under unix, using the stdio buffered file routines will often give a 10x speedup by reducing system calls. With DOS, it is not nearly so dramatic so you might want to work with the unbuffered routines. However, you can simply call fflush(fp) whenever necessary to flush the buffer to the device, or use setvbuf() to unbuffer the stdio routines. If you are using the same fp to initialize a printer as the one used to send data, I don't see why buffering should hurt. If you send a form-feed you would need to flush the buffer to make it happen, though.

Fm Shane Dawalt: Les, Here's the deal (so I you don't keep hearing bits and pieces from me): My program accepts filenames from the command line (and switches). These files are opened and redirected to stdin. A switch can inhibit this redirection and allow input from stdin, e.g. for piping.

The program redirects stdout to the output device. The default is PRN. A switch will allow this redirection to go to a file or to another device. Another switch inhibit redirection and allow stdout to flow normally, e.g. for piping.

The problem is, what if I dump text to the PRN device and stdin detects EOF? Stdout may still have some output buffered, but it doesn't appear on the printer. This appears as though the file being printed is incorrect, when, in reality, the rest of the data is still setting in the RAM for no good reason. When the output is redirected to another device or file, everything is closed properly and, of course, the buffer is dumped to the printer. But it is not a good feeling to see only part of the output on the printer and seeing the interactive prompt pop up on the screen (which indicates command line processing has concluded). [I knew about fflush(), but thought it weird to be working with handles then call a stream type function. I expected a function for use with handles.]

I didn't know you could change the buffering modes. I found them in the reference book. Thanks for the hint!

Fm Les Mikesell: Shane, If you have the file/device names, why bother redirecting stdin/stdout? Just open your own files and handle them whichever way you like. Just remember that single-character read()s are expendable. The stdio routines buffer the i/o into convenient sizes for the machine. I prefer to use the stream functions unless I need more control than just buffering (which is easy enough), for example reading in fixed size blocks or twiddling with terminal parameters. For your printer problem you can just call fflush if you are working with streams, but if you anticipate that the program might be used on a network, you might want to reopen the printer (or dup the fd and close one of the copies). This will flush the job into the network spooler on most networks.

Fm Shane Dawalt: Les, Ya know how they say that you learn by doing? By golly, they're right! I found the syntax for setvbuf() [similar to setbuf()] and implemented it in my program. Didn't take too many modifications (maybe 10 minutes). Anyway, I just ran it a couple of minutes ago and WOW ... single character read is quite slow! (Not that I didn't believe you. <grin>)

Hmmm. I guess I was too set on using handles. I didn't think of simply placing the required stream pointer into the input/output stream pointers for the I/O functions. Gee. That would have been so much easier. That saying about not seeing the forest for the trees applies here.

C: Why | and || ?

Fm Adam Rubin: Theoretical-type question from somebody **very** new to C (namely me):

Why does the || (logical OR) operator exist? Why couldn't | (bitwise OR) also serve as the logical OR function? (In other words, why was it necessary to create the logical OR function?)

Fm Jim Beard: Adam, If you are stringing together logical expressions, the | and ||, and the & and &&, are interchangeable. However, the C language does not define the order of evaluation of logical expressions of the form `if(a<b | c==d) ...` If your program defines c=d when a is undefined, you can make sure that the equality of c and d is checked first by writing `if(c==d || a<b) ...` The double operators force left-to-right priority in evaluation. The double && does the same, but demonstration of its utility is not as straightforward.

Fm Les Mikesell: Adam, There are several differences in the bitwise operators &,| and their logical counterparts &&, || . First, the logical operators will always evaluate to 0 or 1 regardless of the components of the operands. Second, the order of evaluation is defined for the logical operators. Also, the evaluation of expressions containing logical operators will short-circuit as soon as the answer is known (evaluating left to right). Thus, they may be used as flow control statements.

```
x = func1() ||func2();
```

will not execute func2 if func1 returns a non-zero value. The value 1 will be assigned to x if either func1 or func2 returns a non-zero value.

Fm Hardin Brothers: Adam, The || and && are handy when you want to count a variable or function return true if it contains any non-zero value: if(strlen(gets(s)) && a). An expression like this will return TRUE if both halves generate non-zero values, but if you replace && with &, the value depends on the specific length of s and value in a.

Fm Joe Kyle-DiPietropaola: Adam, I would guess that there's also something to be said for the code generation. The logical OR requires no need for intermediate storage, whereas with the bitwise OR there's always the possibility that the resulting value may be needed in the surrounding expression. Of course, the need for storage could be optimized away given a smart enough compiler if the results aren't used.

Fm Adam Rubin: Thanks for your reply, Jim. I think I see what's going on. So:

```
if ( exp1 || exp2 || exp3 ...
```

will stop at the first "true", and immediately execute the statement after the "`if(...)`", but

```
if ( exp1 | exp2 | exp3 ...
```

will execute "exp1," "exp2," "exp3," etc. and then determine whether the whole "`if(...)`" expression is true or false.

So,

```
if ( a==b | 2==c++ | 0*puts('Test\n') )
```

will always increment c and send 'Test\n' to standard output, whereas

```
if ( a==b || 2==c++ || 0*puts('Test\n') )
```

will only increment c if a==b, and only display 'Test\n' if c had been 2.

Hey, I think I just figured out how to combine

```
if(a=b)
  {
     c++;
     if(c==3)
        {
         puts('Test\n');
        }
```

into one expression! (That example doesn't look quite right somehow, but the idea is the important thing.

Fm Adam Rubin: Thanks for your reply, Les. I think I see what's going on with | and ||, and when you'd want to use each. (I have an "example" in my previous message, to Jim Beard.) Most important, I see why both were included in the language, as C doesn't seem to include any "redundant" operators.

Hmm...the course instructor told us not to assume that "true" would be any specific value. I gather some of IBM's mainframe C compilers return -1 for "true" or something...

Fm Adam Rubin: Thanks for your reply, Hardin. I can see where {non-zero} & {non-zero} could often be zero (e.g. 2 & 4), but {non-zero} | (non-zero) will always be non-zero, or "true". Now I see, though, that && and || stop at the first false or true respectively, but & and | will cause all the expressions to be evaluated regardless. A few more programs, and I think everything that was covered in the two-day (!) course will be a lot clearer.

Fm H. Brothers: Adam, Another thing (if you assume that True == 1): Since || always returns True or False, it is easy to count the occurrences of true statements in a loop if you use ||. Code could go something like this:

```
true_count = 0;
for (i = 0; i < max_val; i++)
    true_count += (a[i] || b[i]);
```

(If you want to allow for the possibility that True == -1 on some machines, simply do abs(true_count) as you exit the loop). This is the kind of condensed expression that tends to drive Basic and Pascal programmers up the wall, but I don't find it any more difficult to read and understand than, say,

```
TrueCount = 0
FOR I = 0 TO MaxVal
  IF A(I) <> 0 OR B(I) <> 0 THEN
     TrueCount = TrueCount + 1
  END IF
NEXT I
```

In fact, I think the C expression makes it clearer that the loop is testing for non-zero values in either of two arrays.

Fm Jim Beard: Adam, One last comment: the associativity of the logical operators | and & is left to right, but you can't depend on executing all expressions in a logical expression involving a string of |'s. This is left to the implementation, and is explicitly discouraged in K & R section 2.12, "Precedence and Order of Evaluation". Side-effect expressions are explicitly discouraged where implementation may affect program execution. This is done so that implementation can be determined on the basis of hardware architecture. So, a given implementation MAY increment c and/or send 'Test\n' to standard output.

Your example of the correct usage of || is right on the mark, though. The purpose of || and && is to allow the usage of side effect expressions in logical expressions.

The usual logical value for TRUE is -1 (all 1's) so that NOT TRUE will evaluate to 0. NOT 1 will evaluate to -2, which will be taken as TRUE in the final checking. The best policy is (1) not to assume a value for TRUE, as you stated, and (2) use the language keyword for TRUE if available and -1 if not.

Fm MISOSYS, Inc: MC, as an example, uses a value of 1 to indicate TRUE and 0 to indicate FALSE. This is per the stdio.h header file. However, it will treat any non-zero value as true when evaluating a logical expression. Also, NOT non-zero-value will always be evaluated as FALSE in MC. It doesn't blindly go ahead and just complement the value but evaluates its logical value and then sets the result to TRUE or

FALSE respectively. But yes, different implementations may treat it differently.

Fm H. Brothers: Jim Not in most micro implementations of C, Jim. TRUE is usually 1, FALSE is 0. The reason for the values, if I remember correctly, is so that a true expression can be added to another expression in order to increment it. Notice that true xor true is still false, and that taking the logical not of any non-zero value will produce false (or 0). Like the distinction between the logical && / || and bitwise & / |, C draws a distinction between a logical not (!) and a bitwise not (~); I wish Basic and Pascal were as smart!

Fm Les Mikesell: Adam, The logical operators return either 0 or 1 but logical tests interpret 0 as false, anything else as true. It is best to maintain this convention since you may be testing the return value from a function as well as a logical test.

Fm Les Mikesell: Jim, In C, logical operators return 0 or 1, and logical tests interpret 0 as false, anything else as true. The "!" is a logical NOT and is different from the "~" or bitwise not. There are no keywords for true or false and they are not needed, since the implicit value of an expression can be used logically instead of having to explicitly compare with a constant.

`if (expr)` is identical to `if (expr != 0)` and `if (!expr)` is identical to `if (expr == 0)`.

Fm Shane Dawalt: Les, I was always under the impression that TRUE was always !FALSE which implies TRUE is -1. Then again, my reasoning could be flawed as "!" is a logical NOT and "~" is a bitwise negation. Now I wonder ...

Yes, I've been screwed up for the last 2 years. Page 187 of K&R (1st Edition) says, in paraphrase, that "!0" is "1" and "!" of any non-zero value is "0". Now, if that's changed with the ANSI Draft then I quit!

But the keywords are handy for initializing with program readability in mind.

Fm Les Mikesell: Shane, Yes but the keywords are wrong if they consist of anything but defining the constant 0 for your false value. That is the only thing guaranteed to work in a comparison unless the data types of the things you are comparing match. And even the zero needs to be cast to the correct type if you pass as an argument to a function (unless the function wants the default int type). Testing the implicit value of an expression with a logical operator is the same as comparing with the constant zero so it will work even if the expression returns some other data type.

Fm Jim Beard: Hardin, Only C has the distinct and separate functions of the bitwise and logical operators, to my knowledge. Perhaps Modula 2 does, but I would be surprised if ADA did.

Fm Jim Beard: Roy, Fail-save logical operators and expressions are available in all languages I know about, except BASIC. Large mainframes use ones complement notation, so that a bit-for-bit not-false is -0. In micros and minicomputers, not-false is -1. Languages which do not distinguish between bit by bit operators and logical operators have a keyword for FALSE, except for BASIC.

Fm H. Brothers: Jim, It's not too difficult to implement FALSE and TRUE in Basic, however. In interpreted Basic, my programs often begin:

```
10 DEFINT A-Z
20 FALSE = (1 = 0)
30 TRUE = NOT FALSE
```

In QuickBasic, it's a little neater to used defined constants:

```
DEFINT A-Z
CONST FALSE = 0
CONST TRUE = NOT FALSE
```

I suppose one could argue that using TRUE and FALSE in Basic slows down program execution slightly, but I think that is far outweighed by the clearer meaning of the program. And besides, if execution speed is that important, Basic is probably not the language of choice anyway.

Fm Jim Beard: It's amazing to see that. Your solution is pretty much what I used the one or two times I felt that a BASIC program had to be transportable. The integer data type is necessary in ones' complement machines (large mainframes) because TRUE is -0, which stores as 0 (the same as FALSE) in floating point. The program will work in Microsoft BASIC 3.x on either the Model 4 or MS-DOS because of the rather through work that went into these implementations. This may not be true of other BASIC's though, because it is not necessarily true that floating point variables are converted to integer for bit-by-bit operations.

SORTING

Fm Pete Betz: I have a question about sorting. I recently wrote a Forth program that has to do some string sorting, so I cooked up a simple insertion sort that seemed to work reasonably well. Then, just for the heck of it, I snitched a heapsort routine from MMSForth, modified it to handle strings, and was "thrilled and delighted" to find that it sorted the same material in almost exactly one-fifth the time!

Anyway, the problem is that, even after making extensive modifications, I still can't really picture how the bloody thing

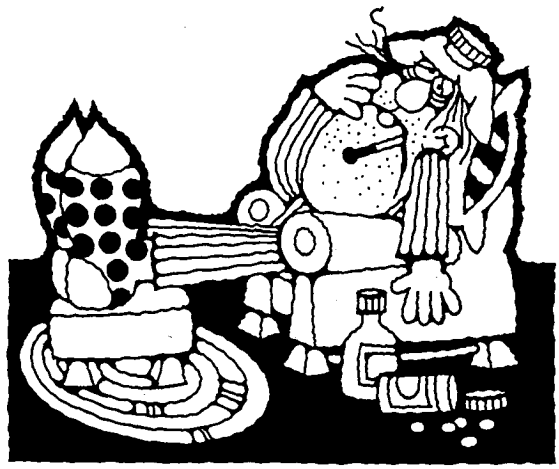
works. Can any of you blokes recommend a book that concentrates on computer sorting? I'm not looking for code or any specific language, but rather descriptions, theory, strategy, algorithms, and comparisons of everything from bubble to quick. Any suggestions?

Fm Joe Kyle-DiPietropaola: Pete, Two of the best sources for information regarding algorithms in general, including sorting, are the standard "texts" on the subject of programming. These are: (1) Knuth's "*Art of Computer Programming*" series. (2) "*Algorithms*" by Sedgewick.

The former is a three volume set, volumes one and two are available in a second edition, and volume three is still in its first edition. Volume three, entitled "Sorting and Searching", is the specific one you'd want, though all three volumes are top-notch. Hardbound only, currently priced at around $43 each. This is **the** definitive reference when it comes to algorithm and analysis.

The latter is a single volume that covers the same range of subject matter, but from a different point of view, and in less detail. Sedgewick will refer you to Knuth for a detailed mathematical analysis of the material in question, but there is certainly enough detail to implement almost any of the algorithms presented. "*Algorithms*" is currently in its second edition, which is a sorta pink/purple color. I have the first edition, which is blue. If you are paying full-boat (circa $35), insist on the second edition. "*Algorithms*" is often offered by the Library of Computer and Information Sciences Book Club, and is one of the books I picked up for a buck each when I joined.

Fm H. Brothers: Sedgewick's Algorithms is a good start. The code is in Pascal and pseudo-Pascal, but it is generally easy to understand. I have the first edition (a second edition is out now, but I haven't bought it yet). He explains a selection sort, insertion sort, shellsort, bubble sort, quicksort, a couple of radix sorts, direct and indirect heap sorts, and a couple ways to do a merge sort. Sorting is just one of the seven sections of the book.



Roy's Technical Corner

Getting into computer math

by Roy Soltoff

Part 2: Binary multiplication

Floating point math... No, I'm not ready to jump into that quite yet. The fundamentals are quite important to grasp; one just cannot jump into complicated mathematics without understanding the basic concepts. That's true in any field; you can't begin to excel at blackjack unless you understand the 52 cards in a deck and the mathematics of probability associated with card combinations. So we're not going to grasp computerized floating point mathematics until we get our feet wet with the basic four functions.

Before I start into this issue's coverage of computer math, let me diverge to a corresponding topic. In a recent issue of *Computer Language*, noted author P. J. Plaugher referenced a book on floating point mathematics by Pat H. Sterbenz, *Floating-Point Computation*, Englewood Cliffs, N.J.: Prentice-Hall, 1974. Plaugher quoted R. W. Hamming's words concerning that book, "Nobody should ever have to know that much about floating-point arithmetic." Based on that recommendation, I, as I'm sure other CL readers, proceeded to locate a copy of the book.

I called Prentice Hall in New Jersey where anyone can order one of their books. According to my scratchings, the number is 201-767-5937; they take plastic. Unfortunately, the book was out of print and no longer available through them. But they were kind enough to steer me to a couple of book dealers specializing in locating out of print books. I called *The Strand* first, as they were the first on the list. "We don't deal in books on mathematics" or something similar was their reply. So I turned to *Continental Book Search*. Al Katz, who appeared to be the chief honcho behind the company, was quite helpful. He took down all of the information concerning the book and said it would take a few weeks to locate a copy; he would get back in touch with me. A few weeks later, I received a notice in the mail from him that a copy of good quality had been located. The price was $60. Of course anything no longer available can sometimes command a higher price; I immediately phoned Al to accept the book. I don't know if he was trying to "twist my arm", but he did inform me that he had other parties interested in the book and if I didn't want it, they did. I bought the book. Actually, Continental Book Search was pleasant to deal with; they did get results. Anyone in need of that service can contact them (him) at Box 1163, New York, NY 10009 [212-254-8719].

The copy of Floating-Point Computation turned out to be a surplus duplicate copy of the Library of Congress, and at one time had been at the US Air Force Technical Library at Edwards Air Force Base, California. I've got a classy book! When I get into floating point math in this article, I'll be able to comment more about the book.

In the last issue of TMQ, I laid the groundwork for this heavy topic. I covered the basics of the binary number system, touched on converting numbers between our familiar decimal system and binary, and discussed the first two functions: addition and subtraction. It's time to move on the the other two basic functions: multiplication and division.

Again as before, since our computers are steeped in binary operations, multiplication will be discussed using binary mathematics. Binary, you shall see, is quite a bit easier to deal with than our decimal system. But before I go further, let's take up a quick review of decimal multiplication. Everyone put away their calculators. Get your brain back in gear.

Terminology, first, for those who have forgotten the terms, and for those who were never taught them. Multiplication has three components: a multiplicand, a multiplier, and a product. In mathematics, multiplication generally follows the *commutative* law, which means that A times B is equal to B times A. But that is not always the case in implementation, especially with floating point! But that will come later. In any event, for our discussion here, when multiplying any two numbers, which one is the multiplicand and which one is the multiplier is arbitrary; it involves the position. As an example let's multiply 13 by 5.

```
 13   multiplicand
x 5   multiplier
===
 65   product
```

I added the terminology of each component. Notice I said multiply 13 **by** 5. That puts "13" on top. But that was a simple

multiplication. You didn't need your calculator for that, did you? Let's look at something more complicated, multiplying 232 by 127. Can you do that in your head? Usually not. You work it out on paper performing what is called *partial products*. Here's how that looks.

```
     232  multiplicand
x    127  multiplier
========
    1624  partial product
     464  partial product
    232   partial product
========
   29464  product
```

Each "partial product" was calculated by multiplying the multiplicand by a different place digit of the multiplier (232x7, 232x2, 232x1). How would we do it if we had only one accumulator for the partial products? We could multiply the multiplicand by the rightmost multiplier digit, accumulate the partial product, then shift the multiplicand left one place and the multiplier right one place, then repeat the process until we have dealt with all digits of the multiplier. Here's that concept again using the example.

```
Step 1
     232  multiplicand
x    127  multiplier
========
    1624  partial product
       0  accumulator
========
    1624  accumulator
```

```
Step 2 - Shift L/R
    2320  multiplicand left
x     12  multiplier right
========
    4640  partial product
    1624  accumulator
========
    6264  accumulator
```

```
Step 3 - Shift L/R
   23200  multiplicand left
x      1  multiplier right
========
   23200  partial product
    6264  accumulator
========
   29464  product
```

The difference is one of mechanical manipulation, but it breaks the problem down into operations which computers are good for. In the binary case, multiplication is as straightforward as the previous decimal example, but the arithmetic is exceedingly simple. That's because the binary multiplication table is quite small. Here it is:

```
0 x 0 = 0
0 x 1 = 0
1 x 0 = 0
1 x 1 = 1
```

It's important to recognize that when the multiplier bit (binary digit) is a zero, the partial product is zero, and when the multiplier bit is a one, then the partial product is identical to the multiplicand. You really have no multiplication at all, just a simple case of testing a bit for zero or one, then either adding the multiplicand to the partial product or skipping the add. It's as simple as that!

Now with this information in hand, let's go back to the case of multiplying two numbers in binary. I'll use the original case of multiplying 13 by 5. In binary, that's:

```
Step 1
    1101  multiplicand
x    101  multiplier
========
    1101  partial product
       0  accumulator
========
    1101  accumulator
```

```
Step 2 - Shift L/R
   11010  multiplicand left
x     10  multiplier right
========
       0  partial product
    1101  accumulator
========
    1101  accumulator
```

```
Step 3 - Shift L/R
  110100  multiplicand left
x      1  multiplier right
========
  110100  partial product
    1101  accumulator
========
 1000001  product
```

Doing our shifting this way may cause some problems when implementing a multiplication algorithm in software. Remember that the Z80 chip in our TRS-80 computer has no hardware multiplication instruction (the 64180 used in the

XLR8er, does). The problem we could run into stems from the fact that if you multiply an 8-bit number by an 8-bit number, the product can be a 16-bit number. If we use an algorithm of shifting the multiplicand left, we need a storage register the same size as the one used for the product. If we are multiplying two 8-but numbers, both the multiplicand and the multiplier would need 16-bit registers. That seems kind of wasteful. Is there another way we can do our shifting?

Let's examine an algorithm where we shift the multiplier and the product left. Since the product has to already be stored in a "double-wide" register, we save one "double-wide" register by not shifting the multiplicand. In this example, I'll shift the partial product (initialized to zero) left. Then I shift the multiplier one bit left and test it for zero or one. If a one, I'll add the multiplicand to the partial product; If a zero, I'll skip. Then I loop until all bit positions of the multiplicand have been examined. The reason that I shift the partial product left first, is that I have to wind up adding the multiplicand as the last thing I do, assuming the last multiplier bit is a 1. So I need to shift the partial product before I test the multiplier bit. In this example, I have "built out" the multiplier to 4-bits so it's easier to see I have to loop four times. I am also noting the letter "C" if the left-shifted multiplicand bit is a one (meaning add) or an "NC" (meaning skip) if the bit was a zero.

```
Init    0  partial product
     1101  multiplicand
     0101  multiplier
```

```
Step 1 - left shift
       00  shift product left
NC   101   multiplier left
========
     1101  multiplicand
     skip
========
       00  partial product
```

```
Step 2 - left shift
      000  shift product left
C    01    multiplier left
========
     1101  multiplicand
      000  partial product
========
     1101  partial product
```

```
Step 3 - left shift
    11010  shift product left
NC  1      multiplier left
========
     1101  multiplicand
     skip
========
    11010  partial product
```

```
Step 4 - left shift
   110100  shift product left
C          multiplier left
========
     1101  multiplicand
   110100  partial product
========
  1000001  product
```

This result is the same as before. However, I only had to use a double-wide register for the product. This is important when considering a specific software implementation - especially so when we start looking at 16-bit by 16-bit as well as 32-bit by 32 bit multiplications.

For those familiar with Z80 operation codes, I'll follow up this discussion with an example from LDOS and LS-DOS which have an 8-bit by 8-bit multiplication routine available through a service call (@MUL8); however, the provision is that it only supports an 8-bit product.

```
; Register A = Multiplicand
; Register E = Multiplier
; Product returned in Register A
@MUL8    PUSH    BC        ;Save some regs
         LD      D,A       ;Multiplicand to D
         XOR     A         ;Init product to 0
         LD      B,8       ;Set loop index
MEA1     ADD     A,A       ;Shift product left
         SLA     E         ;Shift multiplier left
         JR      NC,MEA2   ;Skip if a '1' bit
         ADD     A,D       ;Add in multiplicand
MEA2     DJNZ    MEA1      ;Loop for 8 bits
         POP     BC        ;Finished, restore regs
         RET
```

This routine uses the exact algorithm as I just related. Recognize that in binary, doubling a value by adding it to itself (as in `ADD A,A` above), is the same as shifting left one bit position. The SLA instruction is a logical shift left which shifts all bits of the register by one position. The bit which "falls off the end" is caught in a bucket called the "carry flag". The JR NC,MEA2 instruction causes a "skip" of the ADD instruction following it if the carry flag contained a 0 bit. This tracks with our algorithm detailed above. The reason why this routine is limited to an 8-bit product, is due to the single accumulator, "A", being used to hold the product. If you want

to code a complete 8x8 multiplication routine, it could be done as follows:

```
; Register E = Multiplicand
; Register A = Multiplier
; Product returned in Register HL
MUL8    PUSH    BC      ;Save some regs
        LD      HL,0    ;Init product to 0
        LD      D,H     ;Zero reg for adding
        LD      B,8     ;Set loop index
MEA1    ADD     HL,HL   ;Shift product left
        RLCA            ;Shift multiplier left
        JR      NC,MEA2 ;Skip if a '1' bit
        ADD     HL,DE   ;Add in multiplicand
MEA2    DJNZ    MEA1    ;Loop for 8 bits
        POP     BC      ;Finished, restore regs
        RET
```

I have actually used a "rotate" instruction, RLCA, to examine the leftmost bit of the multiplicand. In this case, all I really want to do is examine the bit; I am not concerned as to the value remaining in the register - provided I keep getting each successive leftmost bit, which I am.

Here's one final variation on this routine. Notice that the 16-bit accumulator starts out needing only the right 8-bits; the left 8-bits are zero (yes, I know, the right 8-bits start out as zero too, but the first ADD can load them with 1's). Each time through the loop, the accumulator can use at most one additional bit. We can take advantage of this fact and have the leftmost 8-bits hold the multiplier. Then the left-shift of the product register not only shifts the product one bit position, it shifts the multiplier as well.

```
; Register E = Multiplicand
; Register A = Multiplier
; Product returned in Register HL
MUL8    PUSH    BC      ;Save some regs
        LD      H,A     ;Set H=multiplier
        LD      L,0     ;Init product to 0
        LD      D,L     ;Zero reg for adding
        LD      B,8     ;Set loop index
MEA1    ADD     HL,HL   ;Shift multiplicand
                        ;  & product left
        JR      NC,MEA2 ;Skip if a '1' bit
        ADD     HL,DE   ;Add in multiplicand
MEA2    DJNZ    MEA1    ;Loop for 8 bits
        POP     BC      ;Finished, restore regs
        RET
```

As the multiplicand is shifted out of the high order 8-bits of the product accumulator, the partial product is shifted in. That saves the processing time for one rotate instruction. Let's step through this routine showing the results of the register operations.

```
00000101 00000000  HL
00000000 00001101  DE
After 5 iterations:
10100000 00000000  HL
00000000 00001101  DE
After 6 iterations:
01000000 00000000  HL
00000000 00001101  DE
01000000 00001101  HL
After 7 iterations:
10000000 00011010  HL
00000000 00001101  DE
After 8 iterations:
00000000 00110100  HL
00000000 00001101  DE
00000000 01000001  HL
```

The result in HL agrees with the previous result. It will also work correctly under the worst case situation with both the multiplicand and the multiplier equal to 11111111B.

```
11111111 00000000  HL
00000000 11111111  DE
After 1 iterations:
11111110 00000000  HL
00000000 11111111  DE
11111110 11111111  HL
After 2 iterations:
11111101 11111110  HL
00000000 11111111  DE
11111110 11111101  HL
After 3 iterations:
11111101 11111010  HL
00000000 11111111  DE
11111110 11111001  HL
After 4 iterations:
11111101 11110010  HL
00000000 11111111  DE
11111110 11110001  HL
After 5 iterations:
11111101 11100010  HL
00000000 11111111  DE
11111110 11100001  HL
After 6 iterations:
11111101 10100010  HL
00000000 11111111  DE
11111110 10100001  HL
After 7 iterations:
11111101 10000010  HL
00000000 11111111  DE
11111110 10000001  HL
After 8 iterations:
11111101 00000010  HL
00000000 11111111  DE
11111110 00000001  HL
```

MISOSYS Products Tidbits

LB Data Manager

Little Brother-M4 L-50-510

LB is a **flat file data management system** where ease of use is its primary goal; you don't need to program anything or remember complicated command sequences to manage data. Even for the most complex data management needs, LB produces results quickly; EVERY function in LB is menu driven and comes with complete on-line HELP information.

To set up a data base, you just define the record layout. For each field, enter a descriptive name, type, and length. LB handles up to **65534 records**; each can contain up to **1024 characters**. LB supports up to **64 fields per record**; fields may be up to 254 characters long. There are seven types of data fields available: alpha, numeric, right justified, literal, dollar, float, and calculated (add, sub, mul, and div); any of which may be a *Protected Field*, so that its data will not be displayed unless the proper *Password* is entered.

You next establish a *screen*, and you are ready to begin entering data! You may view or edit any record at any time. Find information quickly. You can even create an *index* to your data so any record can be accessed within seconds.

Simply define a print format screen, and LB will print records according to your specifications; 10 different formats can be created. You can print with headers/footers, date, time, page numbering, totals and sub-totals if desired, mailing labels format, and even form letters. You select what records get printed and can use an index for printing in *sorted* order as well; great for organizing your report.

For **automating your processing**, LB can be run in an *automatic* mode; frequently used procedures (such as selecting, sorting and printing) can be saved for future use.

LB requires a minimum of two floppy disk drives and 128K of RAM or hard disk, 64K, and one floppy disk drive).

LB Page Skips

Fm Pete Betz: Roy, I just came across the mention in the *Quarterly* by Elmar von Muralt of his problems with LB skipping pages during prints. That's for real -- I've been having the same problem. My "cure", believe it or not, is to always use the <T>of? feature before starting the print, whether I really want it or not. If I ditch a page of paper with the <T> before printing, the page skipping never happens -- if I don't, the problem frequently arises. (We're talking no other variations here; same printer, same PR0 file, same FORMS setup, same curse words, same everything....)

Fm MISOSYS, Inc: He did find his problem. It was a conflict with the FORMS/FLT parameters and the PRTPRMS in LB. Once he corrected the parameters, the problem disappeared.

LB file structure

Fm John Tollini: Roy, I have an application I need to set up for a client that end up generating a data base of names & addresses. Will the file layout of LB cause any difficulties in creating a data file from basic to be maintained by LB. The basic programs would only be adding to the data base, all other maintenance and use would be done with LB. I seem to remember some discussion either here or in TMQ about the subject, but I can't find it now. (The LRL would be less than 256)

Fm MISOSYS, Inc: I think you'd have problems in using BASIC to add data. The data/LB file has a field behind the last record which records number of records. BASIC doesn't give you any convenient file access method which can handle that. The data record portion of the file could be handled with "R" or "D" access since all records are fixed-length blocks. That trailing field is a duplicate of the same data that's in the data/DEF file and both are read and compared to ensure data integrity. The data/DEF file may also be kind of tough. As long as your BASIC program is just going to set up data for adding to a data/LB file, why not just have it output a file in AUTO ADD mode format? That was discussed some time ago in TMQ. We also have a little booklet which discusses that file structure which we include with LB (actually formed from TMQ excerpts). I would recommend that approach.

LB Patch

Fm Daniel L. Srebnick: I have an auto file for Little Brother (M4) that creates several indices. It worked fine, until I recently applied the patch to the sort module that fixed an open file problem (I myself never encountered this problem). It seems that after creating the first index, LB closes all files, including the JOB file, causing the auto sequence to terminate.

Fm Roy Soltoff: Daniel, I have found the problem you were experiencing with the AUTO facility prematurely terminating after you applied the LBSORT1/FIX to LBSORT (of LB). It wasn't a case of the auto file being closed. Actually, the auto file is already closed earlier in the sort/select module with the state of the auto facility preserved in the /DEF file. What actually was the problem was that I forgot to add a PUSH HL into the patch code. That PUSH was supposed to insert a 0 return code at the conclusion of processing the LBSORT module. Since a 0 was not on the stack, LBSORT was passing an error return. This is what kept the AUTO function from being re-enabled. There's a simple patch, LBSORT2/FIX, to apply to fix this up after the LBSORT1/FIX has been applied.

```
. LBSORT2/FIX - - 09/15/88
. Corrects application of LBSORT1/FIX
. Apply via, PATCH LBSORT LBSORT2
D10,F3=21 00 00 E3
F10,F3=D1 21 00 00
```

I fixed it up by getting rid of the preceding POP DE (which cleaned the stack) and changed it to a EX (SP),HL after loading the 0 into HL.

Converting PFS:File data to LB

Brad Stiles
Zugemachtes Feld 11
D-8551 Hemhofen
Federal Republic of Germany

I thought I would pass along some lessons learned in the process of converting several PFS:File data bases to Little Brother. PFS:File is a rather nice data base, simple to use and very flexible in regards to expanding field or record length on the fly. But it is these same capabilities that make it difficult to deal with when printing reports or dumping data for other applications, This is because both PFS:File and PFS:Report determine the report column length from the longest length of the corresponding data field (which is never fixed or limited) being printed and they don't allow truncation in the report. To make matters worse, the data is stored as binary so that BASIC programs can not access it. I have converted most of my applications to LB, and am very pleased with the results and with the increased power of Little Brother. The following procedure outlines the methods which I used:

Print FILE data using REPORT

It is important to use REPORT, since this is the only way to get data out of FILE with a fixed length for each field. Before starting REPORT, it is also a good idea to add a 1 character field at the end of the each record in FILE (using FILE's DESIGN FILE/CHANGE DESIGN commands). This field should be populated with a "." or other convenient character. The field is then printed at the end of each report, to force REPORT to make all the lines the same length. Otherwise REPORT truncates the report line length after the last character printed from each record.

Once in REPORT, use the HEADINGS command to adjust any headings which might be longer that the data contained in the corresponding fields. Establish a report with the PRE-DEFINE REPORT command that dumps all of the data fields in sequential order. REPORT sorts on columns assigned as 1 and 2. If no sort is wanted, use only columns 3 - 16 in the report definition.

Using REPORT's PRINT a REPORT command, specify the report name just defined, the output to a file (eg., OLDDATA/PR:d), lines per page that exceeds the number of records being printed, and a page width wider than the total length of all the fields, plus at least one character between each two columns. With FILE, the exact length of the fields are usually not known, so use the maximum of 256.

If an error message is received that the output is wider than the page width (256), allow the report to finish anyway. The Partial report will be valuable in determining the next step. If the data base and report are large, REPORT may hang on the last record. Reboot and use the program, for recovering routed printer files (MQ, Vol II.ii, pg 30, Jeff Brenton), to recover OLDDATA/PR.

Check and Edit Report:

REPORT writes the file to disk with a LRL=1. In order for SAID to read the file, "COPY OLDDATA/PR:s /TXT:d (LRL=256)". Then inspect OLDDATA/TXT with SAID or other word processor. Determine how long each printed column is and how many blanks were used to separate the columns. Delete any leading blanks which were used to center the report (the SAID Macro and Macro Repeat commands can be used here). If all of the requested fields were not printed ("Report too Wide" error), use this information to go back into REPORT and try again with fewer fields (but including the dummy field)., The fields left out can be addressed after the first report is safely in Little Brother.

On large reports, OLDDATA/TXT may be too large for SAID to edit. To split the last half of the file into a smaller file:

```
ROUTE *PR OLDDATAB/TXT:d
LIST OLDDATA/TXT:s (P,LINE=xxx)
RESET *PR
```

where xxx is a record number approximately in the middle of the file. Edit OLDDATA/TXT, and truncate the file (block delete) at a convenient, recognizable record that is greater than

xxx. Edit OLDDATAB/TXT and block delete the first portion of the file down to the same record. When the files have been edited, checked and are ready for further processing, they can then be recombined with "APPEND OLDDATAB/TXT to OLDDATA/TXT". For very large files, this procedure can be repeated incrementally to get the file into small enough chunks.

Preparing AUTO/JOB for LB

At this point, it should be noted that the file OLDDATA/TXT is simply an ASCII file with data in columnar format, The remainder of this procedure can be used on any such file for loading data from one application into a LB data base for storage and future retrieval.

Roy has printed a BASIC program, which reads a data file and generates an AUTO/JOB file suitable for auto loading into LB (MQ, Vol. I.i, Summer 86, P. 68, not on DISKNOTES V). With a few small changes, this program will also work on OLDDATA/TXT, created by REPORT. LRL should be calculated to equal the sum of all the columns and the spaces between the columns in the report, plus 1 for the carriage return at the end of the report line. When printing the data back: to AUTO/JOB (lines 60 - 80),

```
PRINT#2, LEFT$(field variable,LEN(field variable>-
x);CHR$(13);
```

Should be used, where x is the column separation added by REPORT. Be sure to field the dummy column and the carriage return, but don't bother to print them to AUTO/JOB.

If the BASIC program corresponds accurately to OLDDATA/TXT, the program will generate an AUTO/JOB file that will show each record in the same configuration. It can be edited and examined for consistency.

Populating the Little Brother Data Base

a) The LB data base should be defined to correspond to the original FILE data base. If fields are added or deleted, allowances must be made in the reporting or AUTO/JOB generation phases of this procedure. Before using the LB "6) Run automatically" command, the appropriate screen format must be defined (Define Screen Formats), and the format must also be set as the default screen (Set Screen/Add Index).

Populate the LB data base with the FILE data by choosing "Run Automatically" and entering "Use" and "AUTO)".

When Data exceeds the Capacity for one Report

As noted earlier, when the FILE data base has greater than 14 fields or the report exceeds the maximum report width (256), the data base must be split into two or more separate reports. The second report should include the fields not included in the first report and the dummy field as the last field in the report. This file should be processed in the same fashion as the first one, up to the generation of the AUTO/JOB file.

The BASIC program which generates the AUTO/JOB file should be modified as before, except for the following additional lines or changes:

```
 40 PRINT#2,"3";CHR$(13);:'enter update mode of LB
 60 PRINT#2,"e";:'enter edit mode for each record
 61 PRINT#2, LEFT$(field variable,LEN(field
variable)-x);CHR$(13);: ' Was line 60
 95 PRINT#2,CHR$(11);:' 70 move to the next record
for editing
```

In addition, the printing section (lines 60-80) should be adjusted to include a "PRINT#2, CHR$(13);" for each, field in the LB data base that must be skipped over in order to reach the position for the new data being added.

Following the generation of the AUTO/JOB file, LB can be started, the data base name selected and "6) Run Automatically" selected again. This operation edits each previously populated record and adds the data that did not make the transfer on the first operation.

Fm MISOSYS, Inc: Thanks for the procedure on porting PFS-FILE data over to LB. Although quite involved, you appear to have made excellent use of system resources. That LIST using a starting line number with the *PR device redirected to a disk file is one that I didn't think of to split a file. I'll use your writeup in TMQ. Perhaps Pam will read that.

HartFORTH

HartFORTH [DOS6 M-21-071][DOS5 M-20-071]

HartFORTH is a *full FORTH* that conforms to the 79-STANDARD. The Model I/III version is an indirect threaded version; the DOS 6 version is a direct threaded implementation providing greater execution speed of 10%-40% depending on the details of the actual program. The kernal contains some additional useful words and utilities which turn HartFORTH into a full-fledged development system.

HartFORTH is designed to *run under an operating system* which is totally transparent to the programmer or user. The virtual Memory that it accesses for storage and retrieval purposes is a normal DOS file that is requested by the FORTH system when it is first entered. Doing this has several advantages in that it provides for FORTH files to be used in other language application programs and vice-versa. Enhancements have been built into the kernal in the form of functions to call the operating system file handling routines so that other files may be created or accessed.

HartFORTH supports double length integers, string handling, cursor manipulation, graphics, random numbers, and floating point.

FORTH Compiler

Fm Kevin R. Parris: What type of language is FORTH? What applications is it considered best suited for? PRO-HartFORTH from MISOSYS is down to a very reasonable (too good to be true, maybe) price, but I do not want to get something I may have no use for. Advice please!!

Fm Bill Brandon: Kevin, FORTH is a threaded, interpretive language of considerable power. It is considered to be best for interactive-type applications, less good for number-crunching (though it's not too shabby). It is very fast (10 times faster than compiled BASIC) and produces very compact code (the same application in FORTH will often be smaller than in m/l, because of threading). However, it tends to be difficult to learn, especially if your first language was BASIC or FORTRAN. The code drives conventional programmers up the wall, because it just LOOKS weird, even though (if well done) it is highly structured.

Go by B. Dalton's and get a copy of Brodie's book, *Starting FORTH*. What good is FORTH? Well, there seems to be an awful lot of commercial *processcontrol applications done with it (industrial controls, etc), and many of the games at your favorite video palace are programmed in FORTH as well. I understand there's lots of openings on the West Coast and in Florida for good FORTHers.

Oh, if you learn FORTH, be prepared to get a really odd look from other programmers when you tell them what you're up to. It's the same look, every time. Somewhere between moral outrage and utter disgust.

The MISOSYS FORTH is first-rate, and that's according to professionals (I'm just a neophyte myself.)

Fm Pete Granzeau: Bill, difficult very is Forth, opinion humble my In.

Fm Bill Brandon: Pete, Did I detect "that look" on your face just then? Yeah, you're right. I didn't point out the nice features like RPN (ever wonder why they don't call it NPR? Now you've got **me** doing it!), and the ever-lovin' stack.

Fm Kevin R. Parris: Bill, Thanks for the response. But this term "threaded" is not totally clear to me. Does it have to do with multi-tasking, by chance?

Fm Bill Brandon: Kevin, No - it's just an obscure way of saying that each command or function in FORTH is defined in terms of other commands or functions. Commands, functions, and the like are called "words" in FORTH, by the way. What this means is that execution of a program consists of calling a single word, which incorporates all the words in the program. It's analogous to subroutines, though not the same.

Get the book - FORTH, like most elegant solutions, takes a long time to explain well, and I'm not sure I can do it justice. Another alternative to getting the book - if you can find a copy of the next issue of Science Software, published by John Wiley & Sons, you will see an article on FORTH by yours truly. Not that I'd ever stoop to plugging my own writing, of course!

Fm Hardin Brothers: Pete, A bumper sticker that the Forth user's group was handing out at the West Coast computer Faire a couple years ago read (this is true):

> Forth Love If Honk Then

Fm Bill Brandon: Hardin, What a GREAT piece of bumper clutter! I wonder if the group has any left? Was that the FIG bunch in Mountain View, Hardin?

Fm Hardin Brothers: Bill, As I remember, that was a year with several Forth groups at the Faire, and I honestly don't remember who had the bumper stickers. However, it does sound like something that the Mountain View group would come up with -- or perhaps someone there could steer you to the right folks if you really want one.

Fm Joe Kyle-DiPietropaola: Kevin, I'll add my two cents - FORTH has been described as a "write-once" or "write-only" language. That is, you write a "clever" program in FORTH, and say it works. If somebody else comes along to maintain the program (or yourself a year later), it just might be easier to re-write the thing from scratch as compared to trying to figure out exactly how the original program worked...

Fm Bill Brandon: Joe, Documentation seems to be especially important with FORTH. I've heard the comment you made

from everyone who's ever programmed in it, including Charles Moore. What we haven't told Kevin (but he'll figure out soon enough) is that many of the words in FORTH are typographic symbols or strange combinations which are pronounced very differently from what you'd think (fetch and store still give me a lot of trouble! Reading code in a hurry, I frequently get them backwards. What's the term for programmer's dyslexia? grrr).

The proficient FORTHers I've met say this is no problem, since the language is so efficient to program that re-writes are trivial. I'm not proficient ...

Fm Kevin R. Parris: Well Bill, since it turns out to be RPN logic based, I am less interested than before, since I am (I suppose) a "traditional programmer". I spend the workday tending a large mainframe operating system (IBM MVS/370) in assembler language. Thanks for the explanations.

HARTForth manual errata

Fm Pete Betz: Roy, Has anyone called your attention to the printing flaw in HARTForth's documentation? In the word-reference section, the printer decided not to print any @'s or #'s, leaving blanks instead. Sooooo...

```
page: 26 -- R should be R@
      28 -- (blank)     @
            C           C@
      29 -- <           <#
            (blank)     #
            S           #S
            >           #>
      35 -- IN          #IN
            D IN        D#IN
      36 -- 2           2@
      37 -- "           "@
      41 -- F           F@
```

(A thousand pardons if you already know all this.)

Fm MISOSYS, Inc: Not sure whether anyone called out all of the omissions. I know that some of the characters were supposed to be handwritten in before duplication. They didn't get in. An errata was never prepared. Here's the errata in TMQ and I worked up the same data for the README file which is on the HartFORTH diskette. Thanks for the prodding.

MC C-Compiler

MC [DOS 6.x M-21-064] [LDOS 5.x M-20-064]

If you are looking for a **full C compiler**, look no further. If you are looking for a well stocked UNIX System V standard library, look no further. **MC**, reviewed in the January 1987 issue of *80 MICROCOMPUTING*, is a complete C compiler which adheres to the standards established by Kernighan and Ritchie. The library of functions is extensive and System V compatible. The compiler generates Z80 relocatable macro assembler code (M80 or our MRAS). The libraries are files of relocatable object modules. **MC** is a full-featured compiler for the discriminating programmer!

MC supports command line *I/O redirection* for compiled programs, *wild-card* file specifications, parsing for UNIX "." extensions in file specifications, *overlay* support (requires MRAS), a full pre-processor, lots of options, and is designed for the programmer wishing the ultimate in C compilers. The package is supplied with the compiler, pre-processor, an optimizer, assembler macro files, C libraries, a Job Control Language file, the header files, and a 400+ page user manual. MC requires the use of either M-80 or MRAS (available separately), 2 disk drives, and upper/lower case.

Pro-MC and structs

PRO-MC - Difftime()?

Fm Ralf Folkerts: Roy, I'm having problems with my Pro-MC. Due to lack of time I've started 'playing' with my Pro-MC. But almost every program won't LINK (I'm using M80/L80). This happened with the DCT program and the DHAMP program from one of the later TMQ's. The assembler does it's work without reporting any errors, but L80 generates a '1 Undefined Global(s)'. With the DCT program this is 'CALL', with the DHAMP it is 'DIFFTIM'. I've added a '#include m80.h'. Where may the error be ? When I try to LINK 'manual' I only see a 'LIBA RQUEST' and a 'LIBC RQUEST' but no MATH RQUEST. Btw.: I had problems with the MCMACS/MAC file because the '.LIST' command starts in col. 1 but must start at 8 with my M80 !

P.S.: I plan to trade in my Mod. 4 ALDS for Pro-MRAS. If the error is with the m80/l80 combination let me know. I'll send you the ALDS Disk by Air-Mail tomorrow; the Manual will follow by surface-mail ! Thank you for your help !

Fm MISOSYS, Inc: I think you have a local problem. The difftime() function was listed on page 555 of TMQ II.ii; you would have to either link that in with each program, or add it to a library. Sounds like you also do not have the MC 1.6 update as difftime() was included in that release already in the library. Check out that same issue, page 18. I can't imagine why you would have an undefined global on 'call'. Better make sure that you put in an '#option INLIB' into that code as the call() function is in the installation library. -Roy

Fm Ralf Folkerts: Roy, I found the Problem: The /MAC Version does contain conditional Requests for the MATH/REL, IN/REL and USERLIB. It seems that the IF won't work correct with the M80, because the Programs will link correct, when I remove the IF/ENDIF for the necessary LIB and then assemble and link 'manual'. Any suggestion how to correct that ?

PRO-MC Overlays

Fm Richard Watkins: I am working on a fairly large program, which is entirely written with PRO-MC and I am going to have to break the program up into either separate programs and 'chain' them with cmdi() or use overlays. I have never used overlays before. I have read the part in the manual several times and a couple of questions come to mind. If I understand it right, whenever an overlay exits the return code is ignored. Does this mean that I can only return information from the overlays with global variables? I also would like to know if anyone else has used PRO-MC with overlays and could tell me if there are any bad things lurking around.

Fm David Huelsmann: Richard, XARC4 Version 02.00.01 uses overlays in PRO-MC. No bad things lurking about to my knowledge with overlays. Just be sure that ALL library routines that are to be used across the overlays are declared extern in the root. The overlay can access any function in the root, passing the appropriate values that the root function might need. However, return values from ovmain are ignored. In other words, when your current overlay has finished execution and is returning to the root, any 'return NOERR' type coding will be ignored.

MCOPT Bug?

Fm David Huelsmann: Roy, I may have found a problem when using MCOPT (1.6a). When the following code is compiled without using MCOPT, the expected value of 39 is obtained for numnodes. However, when using MCOPT, numnodes receives a value of -256!

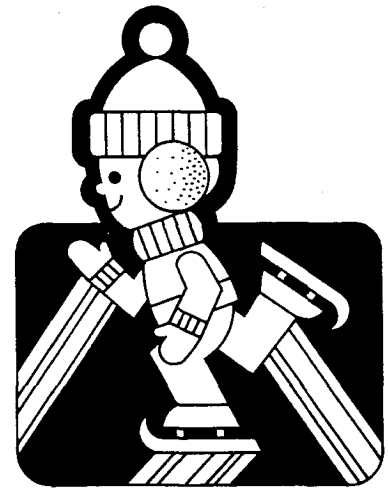
```
#define NUMVALS 257
int dctreehd = 295;
main()
{
int numnodes;
numnodes=(dctreehd<0) ? 0 : dctreehd-(NUMVALS-1);
}
```

Since this problem didn't seem to be arising from one of my own coding errors for a change, I went ahead and dumped both the /ASM file and the OPT file for comparison. I think that you will probably see enough from the /OPT code to deduce what the problem might be. By the way, it doesn't make any difference if you use the conditional operator or if-else terms for the code.

Fm MISOSYS, Inc: Here's the fix. Rich had already stumbled recently on that bug and just supplied me with a fix.

```
. MCOPTSB5/FIX - 09/28/88 - patch MCOPT5/CMD
. correct subtract operation
D25,B5=00 00 00 00 00 00 00 00 00 00 00 00 00 00
F25,B5=2A E1 6D 11 02 00 CD B4 98 7C B5 C2 94 80
D26,29=2A E3 6D 7C B5 20 04
F26,29=2A E3 6D CD B2 99 7C
D26,30=21 E7 6D 34 2A E1 6D CD EF 97 11 03 00 EB CD C0
F26,30=B5 28 07 2A E7 6D 23 22 E7 6D 2A E1 6D E5 CD EB
D26,40=98 7C B5 28 2E F1 C1 E1 E5 C5 F5 E5 CD E6 88 21
F26,40=97 F1 11 03 00 EB CD C0 98 7C B5 28 26 21 C2 6B
D26,50=C2 6B E3 2A E1 6D CD A6 99 E5 CD 3B 96 F1 E3 21
F26,50=E5 2A E1 6D CD A6 99 E5 CD 3B 96 F1 F1 E5 21 03
D26,60=03 00 E5 CD 9E 88 F1 21 AE 6E E3 CD EF 5D F1 2A
F26,60=00 E5 CD 9E 88 F1 F1 21 AE 6E E5 CD EF 5D F1 2A
D26,70=E7 6D C9
F26,70=E7 6D C9
. eop
```

```
. MCOPTSB6/FIX - 09/28/88 - patch MCOPT6/CMD
. correct subtract operation
D25,CE=00 00 00 00 00 00 00 00 00 00 00 00 00 00
F25,CE=2A FE 41 11 02 00 CD B0 6C 7C B5 C2 B1 54
D26,42=2A 00 42 7C B5 20 04 21 04 42 34 2A FE 41
F26,42=2A 00 42 CD AE 6D 7C B5 28 07 2A 04 42 23
D26,50=CD F1 6B 11 03 00 EB CD BC 6C 7C B5 28 2E F1 C1
F26,50=22 04 42 2A FE 41 E5 CD ED 6B F1 11 03 00 EB CD
D26,60=E1 E5 C5 F5 E5 CD 03 5D 21 DF 3F E3 2A FE 41 CD
F26,60=BC 6C 7C B5 28 26 21 DF 3F E5 2A FE 41 CD A2 6D
D26,70=A2 6D E5 CD 3D 6A F1 E3 21 03 00 E5 CD BB 5C F1
F26,70=E5 CD 3D 6A F1 F1 E5 21 03 00 E5 CD BB 5C F1 F1
D26,80=21 CB 42 E3 CD F3 31 F1 2A 04 42 C9
F26,80=21 CB 42 E5 CD F3 31 F1 2A 04 42 C9
. eop
```



MRAS Assembler

Z80 RELocatable Assembler

MRAS [DOS 6.x M-21-083] [LDOS 5.x M20-083]

An advanced Z80 assembly package for the programmer who wants a powerful and flexible development system. It includes a macro assembler which generates either **relocatable object code** modules or CMD files directly, a linker, a librarian, a full-screen text editor, a utility for converting to/from line-numbered files, and a cross reference tool for directly generated CMD files.

MRAS generates M80 compatible /REL files. Supports REPT, IRP, and IRPC macros; nested includes, and a full range of nested conditionals. It has flexible output redirection of listing and symbol table.

MLINK supports virtual memory bit-stream buffering, REL and IRL library searching, zero disk space for DEFS in DSEGs and COMMONs, generation of program overlays, special link items: 0-3, 5-7, 9-11, 13-15.

Includes **MLIB**, our REL module librarian, and our **SAID** advanced full screen text editor which can be used to generate your assembler source code, C-language source code, or edit any type of ASCII file.

PRO-MRAS Bug?

Fm Shane Dawalt: Roy, PRO-MRAS(v1.0a) insists there is an ILLEGAL ADDRESSING MODE error with the following instruction:

```
LD   A,80-.LOW.SYS_SET.SHR.1+.LOW.SYS_SET
```

In theory, this should subtract the LSB of SYS_SET from 80 then shift that result right by 1. The shifted result should then be added to the LSB of SYS_SET. This final result should then be loaded into A.

I thought MRAS was choking on the expression length so I broke it up as follows and rerun MRAS:

```
TMPY  EQU     80-.LOW.SYS_SET
TMPY1 EQU     TMPY.SHR.1
      LD      A,TMPY1+.LOW.SYS_SET
```

MRAS gave the same error on the line with the symbol TMPY1. I retyped the long expression as:

```
LD      A,80-.LOW.SYS_SET/2+.LOW.SYS_SET
```

and this assembled fine. There is no difference in meaning, just in syntax. Why doesn't MRAS like the previous coding? The immediate mode operand was calculated correctly (with the .SHR.1 replaced by /2). The program runs perfectly. Sorry, didn't say what SYS_SET was did I? SYS_SET was an EQUate in a definition file which was INCLUDEd way in advance of it's usage. Before I uploaded the message, I checked for all my usual 'clutsy' mistakes. Didn't find any so I though I'd see what your diagnosis was.

Fm MISOSYS, Inc: It is a bug, and an old one. I thought that was published before in TMQ. The problem is, according to the very old note written into the listing, that the dot vector table (the table of vectors in MRAS which handles the pseudo-ops which require surrounding periods, is 15 entries. The ".SHR." pseudo-op is entry number 9. After the expression analyzer obtains the result of the operation code decoding, it checks to see if a separator character has been sensed (at that point it doesn't know if it got a character or a coded result of a character stream). But since the code for ".SHR." is 9, the routine checking for a separator sees that as a TAB! That's why you get an "Illegal addressing mode error". I can't have any more than 15 entries in the table. I'm not sure if I ever looked at finding a way around that, but since I'm working on a new release of MRAS, I should be able to come up with something. What this means is that the ".SHR." pseudo=op doesn't work, ever. You could use the alternate form of shift right, "<-".

MLINK Problem

Fm David Huelsmann: Roy, I have run across a problem using MLINK when linking 7 overlays to a root that I cannot fathom. The root is a 16.5K /REL file, 2 overlays are 12K, one is 1.5K and the rest are 3.0K. The -v=:0 switch is used to an AlphTech 1 Meg ram disk with 190K free. The longest chain is AB3AH with an org of 2600H for the linker.

If I link 6 of the overlays and the root - no problem (also doesn't matter which six). If I try to link all 7 overlays with the root, MLINK doesn't report any errors but it fails to write any of the globals, variables, or text to the root in the first sectors. And, of course, the program will fail to run. The root is the same number of sectors in length whether 6 or 7 overlays are linked.

I have tried compiling and linking a small root with seven overlays and there were no problems. This rules out the number of overlays as a factor. (I suspect size of the overlays themselves or something wrong with the handling of the symbol table). I have recovered MLINK/VMF and it is 70.5K in size. I have also linked it using a disk as the VMF location (boy, you are right, that takes a looong while) with no difference in output.

I can link the longest chain of 6 overlays, saving the biggest /OV? file and then relink using the one overlay left out previously. After renaming appropriately, everything works correctly.

The language is ProMC and MLINK is 1.0b (I think) with all patches to the MC programs, MRAS, and MLINK current. I am out of ideas, how about you?

Fm MISOSYS, Inc: Well, David, is my face red! Turns out that the problem (read as bug) was easy to find, but difficult to fix. The virtual memory file uses a logical record length (LRL) of 1. All access of data to and from the VMF is via @GET/@PUT single character I/O functions. That makes it very easy to position to an exact byte location when the LRL is 1. Unfortunately, the @POSN SVC can only correctly deal with a file which is 65535 bytes long when the LRL is 1. Besides that, the internal data handling of segment sizes, absolute VMF locations, and byte offsets are done with 16-bit registers - what else?

It turns out that ARC4 has a ton of data - especially in that arc4hlp/rel module. When you linked arc4 with the root and the first six overlay modules (omitting arc4sq), the VMF was 58558 bytes long. That worked out just fine for MLINK. But when all of the seven overlay modules were linked, the VMF was 70K. What happened was that positioning into the VMF for that last module which exceeded the 65535th byte in the file wound up at the beginning of the file; MLINK has no code to trap that 16-bit overflow.

So for the time being, it stands as a bug; if the VMF exceeds 64K during the link session, the resulting CMD file will be corrupt. I guess when I was implementing the Virtual Memory File, the thought of a big overlay process would exceed the limitations of the VMF handling just didn't occur to me. On the other hand, I just dug out my notebook which I used during development of MRAS and MLINK. I did implement the virtual memory file facility before I implemented the overlay facility. The overlay facility was an afterthought developed specifically to use for MC (we eventually decided against using overlays for the MC job processes). So its entirely possible that the implementation of overlays was subject to the 64K limit due to a design oversight. I would think it impossible to have a VMF exceed 64K for the linking of a single CMD file which could run on a Model 4.

Since I have a new version of MLINK on the drawing board, one which handles Microsoft FORTRAN modules, I'll include a fix for this problem. There would be no easy patch to supply, other than one which detected the 16-bit overflow and aborted with an appropriate error message. The fix may not be easy. I'll probably have to look at using 24-bit register handling, switch the VMF to LRL of 256, and use internal page buffering routines.

PRO-MRAS/MLINK

Fm Jim Beard: Roy, You're right, of course. Nothing as complex as a compiler can be entirely free of bugs. Remember Beard's Law: If the probability of writing a line with a bug is p, the probability of writing n lines without a bug is (1-p)^n. Thus, if you write a line with 99% probability that the line is bug-free, a program with 69 lines will have a 50% probability of having a bug. A program with 2000 lines will have about one chance in a billion of being bug-free. Find them all? The wheel has to squeak to get some oil.

For p small and n large, a good approximation of this expression is exp(-n*p). It's all a matter of degree. Since there is so much more money in mainframe hardware and software, more manpower is used for writing and testing compilers, not to mention support for bug fixes on early versions. Microsoft grosses about $150 million a year; CDC spent a lot more than that on every FORTRAN compiler they ever released. Bugs? Sure, but we need to understand that when we buy and use a microcomputer compiler, we are part of the process of developing future versions. That is a reasonable price to pay to have that kind of power at your fingertips at home. I would have a real problem writing my IEEE papers if I couldn't try out my ideas at home; a lot of valuable stuff has come from my Model III F80, Model 4 RATFOR, and MS-DOS RATFOR.

Linker program listings

Fm Shane Dawalt: Ok, the answer to this question must be quite obvious ... cause I can't seem to figure it out. I'm running PRO-MRAS with a rather involved program. The program has been split into 4 modules and I'm linking each with MLINK. Now, I know how to get an assembly listing when I use the -GC switch in PRO-MRAS. And I know that if I use the same procedure (without -GC) for relocatable modules, I will get an assembly listing, but I'll get reference addresses only, not absolute addresses. And that's reasonable since I'm requesting relocatable code generation for MLINK. How do I generate an absolute assembly listing using MRAS and/or MLINK?? I'm sure it can be done.

Fm Jim Beard: Shane, When you produce relocatable code, the assembler goes away before absolute addresses are determined; therefore, you need the relative address map produced by the assembler. This should give you the addresses of code and variables relative to the beginning of the code or the entry point(s).

The linker will determine the absolute loading addresses; use the -M=filespec option to write a loader map to a disk file. This will give you the absolute addresses of the entry point(s) of your code, and you can then use the assembly listing of relative addresses to find the absolute addresses of all the variables and the code.

Fm Shane Dawalt: Well ... I tried that. It was not what I intended to see. All it gives is the absolute addresses of the CALLs (functions, what-have-you). I was intending on seeing more in the line of all labels used within the program with

absolute addresses. Not EQUates, just labels used within the program. (Any ideas on this?) Ya know ... it's kinda hard to go from absolute assembly operation which tells all to relocatable assembly operation which give hints.

Fm Les Mikesell: Shane, The linker map gives you the address of all the symbols it knows about, which are those that have been declared public. Thus if you want an address to show up in the map, just put a label there and declare it as public (or is it global in mras?). If you want to have an absolute listing to help you step around with a debugger, just change it to ASEG and add an ORG to where you want it. You can change it back later if you want. This works best if you can INCLUDE all your modules into one assembly but it should be possible to ASEG any single module and still have the linker resolve other modules to it. Keep in mind that the purpose of a relocating linker is to connect modules that may have been created by different people and may contain identical labels internally. Only the labels declared public need to be seen by the linker (and known by the person working on the other pieces).

Fm Jim Beard: Shane, When you use a relocatable intermediate stage, you drop back to DOS after the /REL file is written without forming the absolute addresses. There is nothing in the /REL file that keeps the information on locations of the program labels. The /LST file will have them as relative addresses.

The loader keeps a table of all global addresses which it uses to link modules and search libraries. This table is available as the loader map. Using this to find program globals such as entry points can give you the base address (first byte) of the program. The sum of the relative address and an offset address gives you the absolute address of the label.

Lots of calculators nowadays give you the ability to do hex arithmetic. The Casio CM-100 does it and lots more, need no batteries (runs on light), and costs about $10. Or, you can invoke BASIC and do it at the keyboard.

Fm Shane Dawalt: Les, The program I'm working on is huge. Not so much in executable code as it is in the source code. I attempted the INCLUDE assembly and got no-where fast. Just too slow. I went to the relocatable assembly since I need only reassemble the source modules I modify then relink them all. The linking doesn't take but a couple of seconds. The assembly was taking upwards of 2 minutes and when you're fixing bugs, the wait just makes all that more time for me to go crazy in. BTW ... it is either PUBLIC or GLOBAL in MRAS. MRAS accepts either directive so you're safe.

Fm Shane Dawalt: Jim, I've got a hex calculator 'round here, but it's nice to just look and see rather than performing the extra operation of fetching the calculator and adding (and hoping the correct buttons are pressed which I tend to blunder sometimes). I may look into a program which using the LST from MRAS and the SYM from MLINK to generate an absolute output. Now that I know the code .. I can debug by simply watching the registers change. It's just the first few debuggings I get lost on.

Fm Jim Beard: Shane, The software to convert a relative map to an absolute map using data from a loader map should be easy, time consuming, to write. But, I haven't seen one lately. The problem with uploading or marketing one is that a version change in either the compiler or loader will require a rewrite. You will be save in the Z80 market, which is static now, but the MS-DOS languages are just beginning to really shine and versions upgrades come on a yearly basis now.

PRO-WAM

Window & Application Manager

PRO-WAM 2.0 M-51-025

This desktop manager gives keystroke access to 4 memory resident **pop-up** applications and disk access of others. A Function Key lets you invoke DOS library commands. PRO-WAM turns your TRS-80 into a powerful machine because it comes with many useful and powerful time savers and desk organizers. Here's some of what you get:

4 An **ADDRESS** file data base prints cards and mailing labels. Throw away that black book and your Rolodex file.

4 **HEAD** pipes formatted address data into your letters.

4 **BRINGUP** tickler file schedules up to 12 items per day by time. New print module. Remember those appointments.

4 **CAL**endar gives you a month at a glance; covers 4000 years. Flags days with BRINGUP items.

4 A 3x5 **CARD** filer for a free-form scratch pad of 40 columns by 12 rows. Or use the new **CARDX** with ***forms*** capabilities. It's great for small data base.

4 **PHRASE** is a KSM from disk for lots of automation.

4 A telephone list and auto**DIALER** for Hayes modems.

4 **CALC**ulator gives you 4-functions at your fingertips. **RPNCALC** gives 7-functions in bin, oct, dec, and hex.

PSORT puts your PRO-WAM data files in sort order. *EXPORT* and *IMPORT* functions allow you to move data across windows between applications and programs. There's even an online **HELP** facility!

PRO-WAM works with all programs which use standard DOS keyboard requests and honor the DOS high memory pointer; requires one 32K RAM bank, about 2K of high memory, and a small piece of low RAM. If you have a model 4, then you must have PRO-WAM!

A better TERM/APP

Fm Damian Kelly: I have a problem with TERM in PRO-WAM. The COM/DVR is installed as per page 117 of the PRO-WAM manual. I have a Model 4P used as a terminal to an SCO Xenix V system. COMM works perfectly with or without PRO-WAM loaded. The 4P works well as a terminal to Xenix. However, if I try to use TERM rather than COMM, the 4P drops characters and prints lots of garbage on the screen - graphics, etc.

Why does COMM work perfectly, but TERM in PRO-WAM does not? Is speed (9600) a problem? The manual is unclear as to whether TERM sets its own speed, ignoring SETCOM. What have I done wrong?

It is very handy to be logged in to Xenix and still have PRO-WAM available - this aspect works well, providing I use COMM in LS-DOS. Note: a direct line is used to Xenix, (no modem) like a normal terminal.

Fm MISOSYS, Inc: Damien, Let me relate to you why TERM behaved as it did. Excuse the technical aspect of the discussion, but if you have gotten used to XENIX, you must be more than a "plain user". One of the features of PRO-WAM is that it supports up to four nested applications (up to four windows with one window per application). Thus, it is entirely possible that TERM could be overlayed by another application which you invoke. It is because of that possibility that TERM had some curious code.

The Model 4 DOS uses a serial driver which makes use of the available interrupt which occurs when a character is received. Any program which then uses the COM driver, as does TERM, must provide a routine which takes over the serial driver's WAKEUP interrupt vector. That vector stores the address of a routine within your program which the serial driver CALLs, and which should issue a @GET of the character just received. That interrupt routine is within the program area of TERM. There is no other place for it. If that routine were overlayed by another program because of a subsequent PRO-WAM application invocation, and a character were received from the COM driver, the interrupt would occur and the machine would go off to never never land. To eliminate that from occurring, TERM was programmed to disengage the wakeup vector prior to every keyboard strobe and re-engage it after the keyboard strobe. That means that during the execution time where it was possible to invoke another application, the wakeup interrupt task was removed from the TERM program area. TERM also had code to try and grab any character which may have been received while the wakeup vector was disabled. Unfortunately, this process just wasn't fast enough to keep up with a character stream. In fact, even at 300 baud, a stream of characters would result in some character loss.

I recently cleared enough things from my desk to find the time to attack the TERM problem. I tried a few solutions until arriving at one deemed successful. I first tried to establish a small wakeup interrupt routine in the PRO-WAM stack area. The purpose of that routine was to spot check a few bytes in the application program area to see if TERM was still resident. If it was, then it would continue the interrupt handling otherwise ignore it and return. But that wasn't successful for a number of reasons.

I next considered the possibility of temporarily disabling the interrupt task by having TERM maintain a flag condition indicating it was doing an @KBD. Thus, during the time of exposure to a PRO-WAM application invocation, the task could inhibit itself. But that was considered to be back in the same boat of character loss.

Finally, I arrived at the solution of disabling PRO-WAM while within TERM! But since I want folks to be able to invoke other PRO-WAM applications while within an application, I designed this version of TERM to examine the keystroke returned from @KBD, if one was entered. If that key code was a function key value, I would enable PRO-WAM for one keystroke, allowing the operator access to PRO-WAM. This whole scheme was simplified by the easy method I used to disable PRO-WAM. Since I intend to write this up in the next issue of *THE MISOSYS QUARTERLY*, I won't elaborate on that here. The bottom line is that because PRO-WAM is disabled for normal key input of TERM, you can't export from PRO-WAM back to TERM. But why would you want to? There is no reason whatsoever. Exported text couldn't be sent to the COM line. So there is absolutely no degrading of the interface.

I have tested out the revised version of TERM at 9600 baud and it appears to function properly. Now for the other PRO-WAM readers, I am including the source code of that TERM here in TMQ for a number of reasons. First, It illustrates another PRO-WAM application in source form for those programmers desiring to learn more about PRO-WAM applications. Second, it illustrates a useful technique of disabling PRO-WAM temporarily. That may prove useful for other applications. Here's the code:

```
;TERM/ASM - 08/15/88
;*=*=*
;       Mini-Terminal application for PRO-WAM
;       Copyright (c) 1988 MISOSYS, Inc.,
;       All rights reserved
;*=*=*
TYPBUF  EQU     2700H
BREAK   EQU     80H
ETX     EQU     3
CR      EQU     13
CURON   EQU     14
LF      EQU     10
DOWN    EQU     26
HOME    EQU     28
        TITLE   '<Mini-Terminal>
        OPTION  CI
*GET    SVCMAC
*LIST   OFF
```

```
        ORG     2700H
        DB      'PROWAM'
        DB      'MiniTerminal',3
        DC      13,0
        DW      IROW,ICOL
        DC      .HIGH.$.SHL.8-$-48+256,0
        DATE
        TIME
        DB      'Copyright (c) 1988 MISOSYS, Inc.'
*LIST   ON
ENTRY   LD      HL,0            ;Window origin
IROW    EQU     $-1             ; (save screen)
ICOL    EQU     $-2
        LD      (TYPBUF),HL     ;Reset CL buffer
        LD      DE,24.SHL.8.OR.80
        @WCREAT
        JR      Z,OK
NOISE   LD      B,2             ;One beep for
        @@SOUND                 ; no window
        RET
OK      LD      DE,'LC'         ;Get *CL
        @@GTDCB                 ; DCB pointer
        JP      NZ,EXIT2
        LD      (CLDCB),HL
        PUSH    IY              ;Save Reg IY
        @@DSPLY HELLO$
;*=*=*
;       See text
;*=*=*
        LD      DE,WAM$         ;Find PRO-WAM
        @@GTMOD                 ; resident
        JR      Z,GOTWAM        ;Go if found
        @@DSPLY NOWAM$
        JR      KIBGN1          ;Set to not
alter WAM!
GOTWAM  INC     HL              ;Point to
        LD      (WAMHK),HL      ; JR offset
KIBGN   INC     (HL)            ;Disable PRO-WAM
by bypassing
        INC     (HL)            ;  filter test
KIBGN1  CALL    SETWAKE         ;Set our wakeup
vector
        PUSH    IY              ;Save previous
wakeup
KILOOP  @@KBD                   ;Scan for key
        JR      NZ,CLLOOP
;*=*=*
; See if the keystroke is a PRO-WAM function key
;*=*=*
        CALL    TESTKEY
        LD      DE,$-$
CLDCB   EQU     $-2
        JR      NZ,KILOOP1      ;Not FKey, don't
hook
;*=*=*
;       If we found WAM module, then unhook wakeup
and enable WAM
;*=*=*
        CALL    ENAWAM          ;Enable PRO-WAM
        JR      Z,KILOOP        ;Ignore if no
WAM hook
        POP     IY              ;Restore old
        PUSH    HL              ; wakeup
        LD      C,4
        @@CTL
        @@DSPLY ENABLD$
        @@KEY                   ;Now allow a keystroke
        POP     HL              ;Recover WAMHK pointer
        JR      KIBGN           ;re-engage
;*=*=*
;       input loop continuation
;*=*=*
KILOOP1 CP      '='.OR.80H
        JR      Z,EXIT1
        CP      BREAK           ;If BREAK, send
        JR      Z,SNDBRK        ;  modem break
        LD      C,A
        @@PUT
        JR      CLLOOP
SNDBRK  DI
        LD      C,1
        @@CTL
        LD      BC,2000H
        @@PAUSE
        @@PUT                   ;a zero
        EI
        @@CKBRKC
        JR      KILOOP
CLLOOP  CALL    CLBGN   ;Scan for CL char
        JR      NZ,KILOOP       ;Back if no char
        OR      A       ;Have NULL, ignore it
        JR      Z,KILOOP        ; (i.e. no char)
        CP      0CH
        JR      Z,HOMECLR
CLLOOP1 LD      C,A
        CP      LF
        JR      Z,KILOOP
        @@DSP
        JR      KILOOP
HOMECLR LD      C,28
        @@DSP                   ;Home is direct
        LD      A,31            ;Set CLREOF
        JR      CLLOOP1
;*=*=*
;
;*=*=*
EXIT1   POP     IY      ;Restore old wakeup
        CALL    SNDCTL4
        POP     IY              ;Restore reg IY
        CALL    ENAWAM          ;Enable PRO-WAM
        JR      CLOSE
EXIT2   CALL    NOISE
        LD      B,5
        @@PAUSE
        CALL    NOISE
CLOSE   LD      BC,8.SHL.8.OR.0
        @WINDOW
        RET
;*=*=*=*
;       Routine to enable PRO-WAM
;*=*=*
ENAWAM  LD      HL,$-$          ;Pick up WAMHK
WAMHK   EQU     $-2             ; pointer
        LD      A,H
        OR      L
        RET     Z
        DEC     (HL)            ;Enable WAM
        DEC     (HL)            ; by restoring
        OR      H               ; filter test
        RET
;*****
; Check the type ahead buffer for any character
;*=*=*
CLBGN   LD      HL,TYPBUF       ;p/u start of
type buffer
        LD      A,(HL)          ;Pick up PUT
pointer
        INC     HL              ;bump to GET
pointer
        CP      (HL)            ;the same?
        JR      NZ,CLBGN1
        OR      H
        RET
CLBGN1  PUSH    HL      ;save pointer to GETPTR
```

```
        LD      E,(HL)    ;p/u offset to buffer
        INC     HL        ;pt to buffer start
        LD      D,0       ;add offset to start
        ADD     HL,DE     ;to point to char posn
        LD      A,(HL)    ;GET the stored char
        POP     HL        ;Rcvr GETPTR
        INC     (HL)      ;Bump by 1 for char
        JP      P,TYPAHD1         ;Ck on bump to
128
        LD      (HL),D    ;Reset to start of buf
TYPAHD1 CP      A                 ;Set Z-flag
        RET
;*****
;       no character in CL buffer - get from CL
;*****
CLSCAN  LD      DE,(CLDCB)
        @@GET
        RET
;*****
; scans CL & saves char - called when scrolling
;*****
TYPTSK1 CALL    CLSCAN            ;scan for a char
        RET     NZ                ;Ret if no char
        LD      HL,TYPBUF         ;p/u type switch
        LD      B,A               ;put char in B
        LD      E,(HL)            ;p/u PUTPTR
        LD      A,E               ; & compare
        INC     HL                ;  to GETPTR
        CP      (HL)              ;jump if key
        JR      Z,TYPTSK3         ; buffer empty
;*=*=*
; Char to stuff - check if buffer will overflow
;*=*=*
        INC     A         ;If the next loc'n wraps
        CP      (HL)      ;  to the GET loc'n,
        RET     Z         ;  don't permit overrun
TYPTSK3 PUSH    HL        ;save ptr to GETPTR
        INC     HL        ;pt to start of keybuf
        LD      D,0       ;& calculate PUT loc'n
        ADD     HL,DE
        LD      (HL),B            ;store the char
        POP     HL        ;rcvr ptr to GETPTR
        DEC     HL        ;backup to PUTPTR
        INC     (HL)      ;Bump past the char
        RET     P         ;Back if not to 128
        LD      (HL),D    ;  else reset to lst
        RET               ;  position in buf (0)
;*=*=*
;       Routine to set the wakeup interrupt vector
;*=*=*
SETWAKE LD      IY,TYPTSK1        ;Add our wakeup
SNDCTL4 LD      DE,(CLDCB)
        LD      C,4
        @@CTL
        RET
;*=*=*
;       Routine tests for any PRO-WAM function key
;*=*=*
TESTKEY LD      HL,KEYTAB
        LD      B,6       ;Init for 6 values
TKEY1   CP      (HL)
        INC     HL        ;Bump to next value
        RET     Z         ;Back with Z if match
        DJNZ    TKEY1
        RET               ;Else ret NZ on no match
;*=*=*
;       Key sequence in code order for PRO-WAM
;*=*=*
KEYTAB  DB      93H,83H,92H,82H,91H,81H
;
HELLO$  DB      CURON,HOME,'Mini-Term'
        DB      LF,'The real small terminal
program'
        DB      LF,LF,'<CLEAR><SHIFT><=> to
exit',CR
NOWAM$  DB      'Can''t locate resident PRO-WAM,
functions inhibited!',CR
ENABLD$ DB      LF,'*** COM line disengaged: '
        DB      'PRO-WAM enabled for one keystroke
only ***',CR
WAM$    DB      'WAM',ETX
;
LAST    EQU     $
        IF      LAST.GT.3000H
        ERR     'Memory overflow!!!'
        ENDIF
        IFLT    LAST,3000H
        DC      .HIGH.$.SHL.8-$+256,0
        ENDIF
        END     ENTRY
```

The resident module of PRO-WAM is a filter program; it filters the keyboard device. Having a standard DOS 6 module header, its resident code starts out as follows:

```
BEGIN   JR      START             ;Go past header
        DW      $-$               ;Last byte used
        DB      3,'WAM'           ;Module name
MODDCB  DW      $-$
        DW      0
BNKSAV  DB      0
START   JR      C,FILTER          ;Filter if @GET
PUTOUT  PUSH    IX                ;Chain to next
        LD      IX,(MODDCB)       ;  driver/filter
        @@CHNIO
        POP     IX
        RET
FILTER  CALL    PUTOUT            ;Get KI char
```

Here's how TERM deactivates PRO-WAM. Notice that the module only takes over if an @GET device function is being requested, i.e. an input request. If you don't know that and are interested, better consider a copy of *The Programmer's Guide to TRSDOS 6*. This input request is tested by the `JR C,FILTER` statement. If not input, the code falls through to the device chaining routine. If we examine the relative byte offset at BEGIN, we find that an offset to PUTOUT differs from an offset to START by only two bytes. If we temporarily add two to the relative branch offset in the first jump relative instruction, it would branch directly to START and thereby bypass the test for filtering.

The code illustrated in the TERM program first makes sure it can locate the WAM module in memory. If so, it positions the pointer returned by GTMOD to the address of the jump relative offset. It then saves this address in the ENAWAM routine. It does that so that the routine can later ensure that it has a valid pointer before it starts altering code in memory. Any PRO-WAM application can use this same trick to temporarily disengage and re-engage the PRO-WAM facility.

The whole purpose of disabling PRO-WAM is to ensure that while TERM has established an interrupt routine within its code space (for COM wakeup vector processing), you cannot invoke another PRO-WAM application which would overwrite that interrupt routine. This version of TERM still allows you to invoke an application while in TERM, say the DIALER facility to autodial a telephone number, by testing keystrokes for the function key code values normally interpreted by PRO-WAM. If it senses that you pressed a function key, it then disables the interrupt routine thereby avoiding a system crash, then re-enables PRO-WAM for one keystroke. If you then enter a function key code a second time, PRO-WAM sees that and invokes the corresponding function.

This version of TERM/APP appears on DISK NOTES 3.2; it has already been placed on the PRO-WAM distribution disk along with an appropriate note.

PRO-WAM and SuperLog

Fm MISOSYS, Inc: In the previous issue of TMQ (III.i, page 3), I noted that John Coyne, of the NATGUG user group in Great Britain, had succeeded in discovering the root of the problem in using Superlog together with PRO-WAM. Gordon Collins, editor of NATGUG NEWS, the official publication of the group, was kind enough to send me the text file of John's article for publication in TMQ. For those who may have an interest in corresponding with NATGUG, here's their address:

NATGUG NEWS
c/o Gordon Collins
11 Elizabeth Road
Sutton Coldfield
West Midlands B73-5AR
Great Britain
Tel: 021-354-3299

Moving SuperLog 4 into the XLR8 Board

by John Coyne

I have been using Pro-Wam for some time now and must confess it is a fine utility but, it has what I consider one shortcoming, it lacks a large window filing system, database, call it what you like. Superlog, on the other hand, does have a large window, but lacks the many facilities of Pro-Wam, however, these two utilities can be made to compliment each other. Superlog is a database which can be called up through a windowing facility similar to Pro-Wam, but each of these two utilities fulfill the windowing task in a different manner. Pro-Wam always works from bank 0 and uses the allocated ram bank as a data store. Data is extracted from this bank when required. Superlog, however, resides and runs in its allocated bank. Superlog was designed for use with the standard 128k machine and works fine in those two extra banks, but trying to run Superlog from any of the additional banks provided by the XLR8 causes problems. I found the inability of Superlog to move to a bank greater that 2 to be a bit limiting now that I had the extra memory, also, I normally try and keep the lower two banks reserved for DDuty. It was my intention to find out if both DDuty and Superlog could be made to run together, therefore, something had to move. At present they both want to use the same banks. Superlog appeared to be the easiest to try and move and I decided that was the way to go. A brief look at the inner workings of Superlog revealed that Superlog did actually work in the XLR8 banks, only the screen I/O ended up confused.

The additional memory of the XLR8 does not support VDU shadowing (the switching in of the VDU ram while the XLR8 memory bank is selected), however, this is possible with the standard 128k and, Superlog was designed to make use of this facility. Although the XLR8 driver ensures memory bank 0 is always switched in when VDU i/o occurs while a XLR8 bank is being accessed, this does not resolve the problem of a block move to or from the VDU when the sending or receiving area is in the XLR8 bank. For instance, if Superlog is running in, say bank 4, and wanted to sent a string to the VDU, Superlog would simply point to the string in its own memory (bank 4) and request the display supervisory call (SVC @DSPLY). The operating system will then deal with the request, but first the XLR8 driver will switch in bank 0 and in doing so switch out the string to be displayed. The SVC has been given a pointer which now points to an area in bank 0, which will probably contain rubbish, and that what gets displayed, however, once the SVC is finished, control will, very correctly, be passed back to Superlog in bank 4.

It was not to difficult to solve that little problem. It was just a matter of transferring any message strings to be passed to the VDU into a work area in low memory first. Surprisingly that was all that was required to get Superlog to run in banks greater than 2. There is a limit though, without major surgery Superlog will only work upto bank 7. I considered that to be sufficient and left it at that.

The purpose of this whole exercise was to get Superlog to work with Pro-Wam and DDuty, and that is where the problem started. Solving the problem once it was found was easy, but finding it first was a long and tedious task. I nearly gave up a couple of times, but curiosity got the better of me.

Using Superlog and Pro-Wam together the first time proved to be a disaster. Going into Superlog and then into Pro-Wam was fine, and as long as you remained in either of these utilities, no problem, however, when returning from Superlog to the system I was never quite sure what would happen. After the first crash I made sure there was nothing in the system other than a backup of my boot disc. Debugging in the extended memory banks is very frustrating and I would be very pleased to hear from someone who has found an easy way of doing it, anyway that is another subject. Furthermore, the problem was

not really a bug, the software was doing exactly as it was designed to do, it was just the inevitable when asking a number of programs to interface with each other. You may argue that windowing programs were designed to work with other programs; yes they are, but everything has its limits. Because Pro-Wam is a popular program and it contributes to this particular problem and, because the problem may reoccur when interfacing with other programs, I considered it worthwhile giving a summary of what can happen.

Although Superlog is hooked into the keyboard for input, it is also task driven to allow it to be invoked under any condition. When Superlog is installed another task is added to the system. My system also has a clock running, which is task driven, therefore , I have, in addition to the system tasks, two further tasks running. Under these conditions my stack can reach half capacity (0340H). This is an important point as can be seen later. Superlog uses its own stack, but because it is task driven the Superlog driver needs to store the interrupt information before passing control to Superlog main. This information, comprising mainly of return addresses, is held on the main system stack and, depending where the interrupt occurred, this information may take the stack below 0340H. When using Superlog on its own this presents no problem, however, using Superlog and then calling Pro-Wam will overwrite anything on the stack below 0340H (and that is not a bug). At this stage it will not affect either Pro-Wam or Superlog and you are able to switch between them as much as desired. The problem occurs when trying to return to the system. You never know what will happen, in most cases the system will crash.

I will try and explain in simple terms (I hope) the sequence of events leading to the problem. I have mentioned above, when Superlog is invoke the return address of the calling program is stored on the stack before passing control to Superlog. Depending what else is running in the system this may push the main stack below 0340H. Superlog maintains its own stack (07E0H for those interested) in its own work area and has no further call on the main stack. At this point everything is still OK. If you now call Pro-Wam which, incidentally, is already installed and is quietly sitting in high memory, in most cases above 0F300H, but always in bank 0. Pro-Wam will now take control of the stack and move it into its own work area. The first thing that Pro-Wam does after initialization is to present the user with a menu and prompt for a keyboard input using the system SVC facility. The use of any SVC facility will pass control to the system while the SVC request is being processed. Readers familiar with LS-DOS may have notice that Pro-Wam keeps its stack in the VDU/keyboard overlay area and if the stack stayed there while the VDU/keyboard is switched in the stack would disappear, probably resulting in a crash. The system is clever enough to realize this and if it detects that the stack is above 0F300H then it will place the stack somewhere safe. There is only one place where the system can place the stack and that is in its own stack area, however, because it is an SVC and SVCs can be called from anywhere, the system will preserve its own main stack by place this second stack half way down the main stack area, you will have guessed by now, at 0340H. Anything else in that area will be overwritten which, in this case, is the return information for Superlog to get back to the system.

Now, you may argue that any windowing program must preserve the stack of the foreground task. Both Pro-Wam and Superlog do exactly that, however, who expected one windowing program to run inside another. The system does the right things and there must be limits to the stack, so its really up to the application program. Each program (Pro-Wam and Superlog) work fine on their own, yet they were designed to be used while other programs are 'running'. So where does the responsibility lie. I will leave the question there.

As I was already digging around in Superlog, and there appeared to be plenty of space, I found it was the easiest to take care of the stack problem within Superlog. Superlog and Pro-Wam can now work happily together without any disasters.

It is also possible to have DDuty, Superlog and Pro-Wam working together, however, this requires a more cautious approach. It is possible to use both DDuty and Superlog on their own, but only if the Superlog driver is installed in low memory and only if a Superlog window is opened and close in the same Dduty partition without changing a partition. Any attempt to change a Dduty partition with a Superlog window open will cause a problem. To be able to change partitions with open windows would require some serious modification to Dduty. I am not sure it is worth the effort. DDuty and Superlog will, nevertheless, work very well together if Pro-Wam is used as a stepping stone, but again, only if Superlog driver is in low memory. What it really comes down to is that if you do need to change partitions and leave a Superlog window open you must pass control to Pro-Wam first. In other words open a Pro-Wam window inside of a Superlog window and then there is no difficulty in swapping partitions and you can still open more Pro-Wam windows in the next partitions, but no further Superlog windows. Before returning to the original partition any Pro-Wam windows must be closed. Once in the old partition, close the Pro-Wam window and continue where you left off. This procedure is a bit makeshift, but it can get you into the next partition without closing a Superlog window. I am not so sure there are many member who use Superlog and probably even fewer who use Superlog and a XLR8, however, I hope the discussion on the stack problem may prove to be informative. Should anyone require a patch for Superlog to run in a XLR8 machine please get in touch.

The Sorcerers Apprentice

Submitted by a NATGUG member...

'Twas in the Sorcerers spell-room one dark eve, the Sorcerer and I were looking at some spells. I was endeavouring to learn much, but my brain was slow and my mind agog at the Sorcerers vast library of spells. One thing I did recall later, alone in my hovel. "Get ye the spell 'Prowam'," said the Sorcerer, "it is powerful magic indeed."

The next day I spent much time searching for the magic number of the Wizard 'Soltoff', who lives in a place called 'MISOSYS', where night is day, and day is night.

That evening with trembling hands I reached for the communicator and the magic number. I need not have feared so for my lucky talisman 'Visacard' was just the trick to release a copy of this spell, 'Prowam' to me.

Some weeks later it arrived, I opened the package with nervous fingers, there it was inside, complete with spell-book! I was indeed happy. There followed many hours of learning, 'Prowam' is indeed powerful. The Sorcerer was pleased with my increasing skills and gave me a new spell, 'Superlog' to try out.

With my skills now rapidly advancing I endeavoured to use the power of 'Prowam' to influence the magic of 'Superlog'. How could I produce a sorted 'Superlog' file? The answer came one day as I was mixing potions in the kitchen (a powerful drug called 'Coffee'). "Of course," I spake aloud, several small furry creatures running panic stricken at the sound of my voice, "use 'Psort'."

The spell 'Psort' is powerful and useful magic by itself, if only I could fool it into influencing 'Superlog'. After much pondering I came up with a plan so cunning even the Sorcerer was impressed. My plan for sorting /LOG files is chronicled below;

The following spells are needed to perform this operation, they are; FED/CMD (or FED6/CMD) better still is 'Elesfed', BASIC/CMD, PERASE6/CMD or ERASE/CMD or similar spell, PSORT/CMD which comes with PROWAM.

The course of action is as follows; first create a file of 1024 bytes. The way I chose was with a BASIC program along the following lines;

```
OPEN "O",1,"filespec/ext.password:d": FOR n=1 TO
512: PRINT #1," ":NEXT n: CLOSE
```

Why only n=1 TO 512, you might ask? Well the `PRINT #1," "` produces the following output; 20 0D since 0D is a carriage return. So now we have a file of 1024 bytes, which is the length of a single 'Superlog' record. I used ERASE/CMD filespec to 'zero' all the bytes in the file but this is superfluous wizardry.

Next, using the great 'Elesfed' I took the 'zero-file' and changed its nature. By changing certain bytes 'Psort' would be fooled into thinking that this file was the start of a 'Prowam' data file. The bytes to change are as follows;

02 - this is the number of pages in the /LOG file (in hex).

10 & 11 - these must be changed to 00 04 (this is 1024 decimal the LRL).

12 & 13 - 1st Rec as Data, we need 01 hex.

14 & 15 - Relative location for SORT, we need 40 00 hex (64 dec).

16 - Length of SORT 3C hex (60 dec).

These changes produce a strange result, the spell 'Psort' thinks that the 'zero file' is the start of a 'Prowam' data file. The spell will look at this file and SORT on the 3rd line of each 'Superlog' record, (one whole line). The resultant file will be read by 'superlog' and appear in normal format.

Now that the file has been created, how do we use it? Well, 'Elesdos' the spell maker has a useful function called APPEND, a kind of sub-spell. We use this to add our 'Superlog' file to the newly created header thus;

```
APPEND     filespec/LOG.password:d     [to]
filespec/ext.password:d
```

In this manner the 'Superlog' file remains in its original form, whilst the other file is now almost in a format ready to be SORTED. Using FED/CMD I looked at the new file and saw that it was good. I then changed the byte 02 from 00 to nn to reflect the number of pages in the 'Superlog' file. Now the file was ready to be tested, would my magic be good enough? I issued a new spell;

```
PSORT filespec/ext.password:d
```

[pack is NOT applicable] Lo and behold! It worked. I examined the new sorted file holding my breath in anticipation. Fortunately I did not have to strain myself, it had worked perfectly.

The first record of the file is in a language not understood by mere mortals, it can be partly erased using <SHIFT><CLEAR> but the remainder must remain, as this is used in later SORTs.

I used my quill to enter a warning on this first screen, to the effect that the file must be changed to reflect the number of pages before attempting to use 'PSORT' again.

Also the page numbers do not reflect their true value, if the file is small it would be possible to change each one. Perhaps some other apprentice could write a BASIC spell to do this, if so I would very much like to see it.

```
300 LSET COUNTER$=CHR$(Z) 'COUNTER$ = number of
records
310 LSET LRL$=CHR$(64) 'LRL$ = Logical Record Length
(64 bytes)
320 LSET FSTDAT$=CHR$(8) 'FSTDAT$ = First record
that contains data
330 LSET KEYLEN$=CHR$(8) 'KEYLEN$ = key length -
used by PSORT for sorting
340 PUT 1,1: CLOSE 1
350 SYSTEM
```

Generate DIALER/DAT

Fm Rev James R. Peet: Dear Roy, I wrote this little BASIC program that creates the DIALER/DAT file used by PRO-WAM's DIALER application. It enables me to quickly recreate my DIALER file from my CHURCH DATABASE system files. I thought that if you thought it profitable you could print it in your MISOSYS Journal.

```
10 'By Rev. James R. Peet
20 '7017 S.  Fairfax Street
30 'Littleton, CO 80122
40 '(303)-721-9707
50 'September 9, 1988
60 'Creates DIALER/DAT file for PRO-WAM's DIALER
application
70 'Need to modify lines 100-110 to suit your file
needs
80 CLS
90 Y=3
100 OPEN "d",1,"famrec00/key",84
110 FIELD 1, 3 AS FILL1$, 8 AS L.NAME$,6 AS
FILL2$,28 AS COMMENTS$,30 AS FILL3$, 8 AS PHONE$,1
AS FILL4$
120 OPEN "d",2,"dialer/dat:6"
130 FIELD 2, 64 AS DATA1$,64 AS DATA2$,64 AS
DATA3$,64 AS DATA4$
140 Z=LOF(1)
150 FOR X=1 TO Z STEP 4
160 GET 1,X
170 LSET
DATA1$=L.NAME$+"   "+PHONE$+"       "+COMMENTS$+"
       "
180 GET 1,X+1
190 LSET
DATA1$=L.NAME$+"   "+PHONE$+"       "+COMMENTS$+"
       "
200 GET 1,X+2
210 LSET
DATA1$=L.NAME$+"   "+PHONE$+"       "+COMMENTS$+"
       "
220 GET ',X+3
230 LSET
DATA1$=L.NAME$+"   "+PHONE$+"       "+COMMENTS$+"
       "
240 PUT 2,Y:Y=Y+1
250 NEXT X
260 CLOSE 1;CLOSE 2
270 'For explanation of lines 280-330 see page # 29
in PRO-WAM manual
280 OPEN "d", 1,"DialeR/dat"
290 FIELD 1, 2 AS FILL1$, 1 AS COUNTER$, 13 AS
FILL2$, 1 AS LRL$, 1 AS FILL3$, 1 AS FSTDAT$, 3 AS
FILL4$, 1 AS KEYLEN$, 233 AS FILL5$
```

PRO-PaDS & LS-FED-II

John Grindey
1 Chapel Terrace
Beeley, Matlock
Derbyshire DE4-2NT
Great Britain

Here are some patches for a couple of your discontinued product's namely PRO-PaDS and LSFEDII that have been asked for by other TMQ reader's.

Patches for PRO-PaDS for 6.3 dating: With these patches PRO-PaDS directory is compatible with all 6.x dating. I have used PDS dir entry byte-9, bit's 4 and 5 (which were reserved for future use) to store the year's two extra bit's as in the standard directory, and the rest of the directory is as before so it will read the directory of PDS files created before the patches were applied.

The patched version of PRO-PaDS will only operate with 6.3 (it uses SVC 95 which is not in 6.2) and should only be used with it, and no testing for pre 6.3 is included.

There is a choice of dating types either the original type where the date is taken from the system, or use of the file's original creation date (this patch uses code from Notes from MISOSYS 4, Patch PPADSD/FIX).

ALL PATCHES are to the members that have been copied from PRO-PaDS not to PRO-PaDS itself. As the patching is fairly lengthy I have created a JCL file (PDSMOD/JCL); Apply by `DO =PDSMOD`.

The LSFEDII/FIX is a patch for the UPDATE command. This updates the date but only sets the time to 12:00am. After this patch is applied the programme will only work with 6.3; I have included a routine to trap pre 6.3 DOS.

```
.Patches to PDS to allow date's up to 1999
.Also patches PDS(BUILD) for 6.3's APPEND
.Files
.       PDS/CMD.PDS    NOT THE ORIGINAL COPY
.       PDS663A/FIX
.       PDSDIR/FIX
.       PDSAPP1/FIX
.       PDSAPP2/FIX
.       At least 10k. of disk space on drive 0
.       MUST be available to the system
//PAUSE Press ENTER to continue BREAK to abort
pds(c) pds(append)append
pds(c) pds(build) build
pds(c) pds(dir)dir
pds(k) pds.pds(append)
pds(k) pds.pds(build)
pds(k) pds.pds(dir)
pds(p) pds.pds
y
patch build/cmd pds633a (o=n)
patch dir/cmd pdsdir
.The next patch's give a choice of date type,
. either the date the file was Appended to the PDS
. file, or the files original creation date
//ALERT 1,1,2
//KEYIN Select Date type < 1 > System date , < 2 >
File Creation Date ?
//1
patch append pdsapp1
//2
patch append pdsapp2
///
!append append pds.pds
!append build pds.pds
!append dir pds.pds
remove append/cmd build/cmd dir/cmd
cls
.Let's see if it work's
pds(d)pds
//exit
```

```
.PDSAPP1/FIX - Patches to PRO-PaDS APPEND to make it
. compatible with both old and new
.dating systems, using system date.
.
X'26DD'=C3 75 2D
X'2D75'=4F E6 07 47 13 1A 07 07 07 B0 77 2B 13 79 E6
18
X'2D85'=07 47 1A B0 77 C3 E9 26
.EOP
```

```
.PDSDIR/FIX - 06/05/88 - Patch for Model 4 PRO-PaDS
.Patches to PRO-PaDS DIR to make it compatible with
. both old and new dating systems.
.
x'2800'=E1 7E E6 07 47 CD 40 2B 00 00 00
X'2B40'=E5 2B 7E E6 30 0F B0 C6 50 26 00 6F 06 02 3E
5F EF E1 C9
.EOP
```

```
.PDS663A/FIX - 02/14/87 - Patch  to PRO-PaDS for
. LS-DOS 6.3's ATTRIB
.Fixes PDS(BUILD) to revamp the internal ATTRIB
. command string. This patch to be applied only to
. the extracted BUILD member. Apply for LS-DOS 6.3
. only via, PATCH BUILD.PDS using PDS663A
D01,E9=70 3D 52 45 29 0D
F01,E9=75 3D 2C 70 3D 52
.Eop
```

```
.PDSAPP2/FIX - Patches to PRO - PaDS APPEND to make
. it compatible with both old and new
.dating systems , using file original creation date.
.this version uses code Published in Notes from
. MISOSYS Issue 4 (PPADSD/FIX)
.
X'26CE'=18 16
X'27DC'=CD 75 2D
X'2D75'=C2 EE 29 ED 4B 7F 2C 3E 57 EF C2 EE 29 79 07 07
X'2D85'=07 4F 3E 46 B1 32 90 2D FD CB 18 46 23 D5 11 02
X'2D95'=2D 7E 20 09 E6 0F 12 23 13 7E 12 D1 C9 E6 0F 4F
X'2DA5'=23 7E E6 F8 47 D5 11 11 00 19 D1 7E E6 18 07 B1
X'2DB5'=12 13 7E E6 07 B0 12 D1 C9
```

```
.LSFEDII/FIX -06/05/88- Patches to LSFEDII/CMD
.Patch to LSFEDII/CMD to make the "U" command work
. with 6.3 date type
.Set's date to system date and time to 12.00am
.
.CALL EXTRA ROUTINE
D04,16=CD 90
F04,16=3E 58
D04,1C=42
F04,1C=EF
.ALTER VERSION NO.
D10,BD=33
F10,BD=30
.EXTRA CODE
D12,77=4E 6F 20 4D 65 6D 6F 72 79 20 74 6F 20 6D 61 70
F12,77=49 6E 73 75 66 66 69 63 69 65 6E 74 20 4D 65 6D
D12,87=20 6C 6F 61 64 20 6D 6F 64 75 6C 65 20 66 69 6C
F12,87=6F 72 79 20 74 6F 20 6D 61 70 20 6C 6F 61 64 20
D12,97=65 03 4E 6F 20 4D 65 6D 6F 72 79 20 74 6F 20 6D
F12,97=6D 6F 64 75 6C 65 20 66 69 6C 65 03 49 6E 73 75
D12,A7=61 70 20 64 69 72 65 63 74 6F 72 79 03 D5 11 10
F12,A7=66 66 69 63 69 65 6E 74 20 4D 65 6D 6F 72 79 20
D12,B7=00 19 36 00 23 3E 00 77 D1 3E 58 EF C9 00 00 00
F12,B7=74 6F 20 6D 61 70 20 64 69 72 65 63 74 6F 72 79
.END OF DATE ROUTINE
D1C,30=32 FD 33 7E D6 50 32 99 42 00 00
F1C,30=4F 7E D6 50 30 01 AF B1 32 FD 33
.CHANGE MESSAGE ADDRESS
D1E,58=75
F1E,58=7F
.AS THIS VERSION IS COMPATIBLE WITH 6.3 TYPE DIRECTORY
.THIS CODE CHECKS FOR IT AND ABORTS IF NOT
D00,16=C3 90 64 00
F00,16=ED 73 08 30
X'6490'=ED 73 08 30 3E 65 EF FD 7E 1B FE 63 D2 0E 30 21
X'64A0'=A9 64 3E 0A EF 21 FF FF C9 4D 75 73 74 20 62 65
X'64B0'=20 4C 53 2D 44 4F 53 20 36 2E 33 0D
.EOP
```

The Hardware Corner

Adding floppy disk drives to
TRS-80 Models III, 4 and 4D
by Charles A. Ainsworth

Introduction

There are several reasons why one might wish to add or change floppy disk drives: One may have a pair of drives in the computer and wish to add two external drives or change from the computer's original drives to others of greater capacity or better quality, adding or not external drives (up to a maximum total of 4, numbers zero through 3), etc.

I first became involved in the matter when using model III and the quality of the original drives I got was unbearably bad and each computer had been to a service shop about five times for drive adjustment and alignment before I realized that they either didn't know how to fix them or just plain couldn't due to the poor quality of the drives. I was also having problems reading disks generated in other computers, probably indicating severe misalignment. That led me to disconnect the original drives and use others I bought, at the same time adding a pair of external drives to increase the available disk KB on line.

My experience is limited to models III, 4 and 4D. I am not covering the 4P on which I have had no experience; there was an article on adding drives to the 4P in an 80 Micro some time ago. Caution: There was an error in showing how certain connections had to be soldered to a board, which was corrected in a subsequent issue.

When deciding to use drives purchased separately for numbers zero through three, I also wanted the maximum possible capacity; at that time there was no 3-1/2" drive I knew of that would be suitable so I concentrated on 5-1/4". A comparison of prices showed me that for a reasonable extra I could use the largest capacity then generally available, 720K, double-sided, 80-cylinder 5-1/4" units and that was what I got for all four drives. I'm very satisfied with the results and the extra for double-sided 80-cylinder was very well worth it. One always seems to be needing additional disk capacity. (Would be nice if someone came up with a model III/4 driver for the 3-1/4" 1.4 MB drives!)

Later, when I installed XLR8er boards from MISOSYS on my 4D computers, I got a bonus in available drive capacity. I had four 720K drives, of which my system and program disk used one and my data disks three. By putting the system and program files in the Ramdisk the XLR8er provides, I now have four drives available for data disks.

Anyone wishing to follow my ideas and install new drives and interested in the newer 3-1/2" size, should not fail to read Roy Soltoff's clarification of types and drivers in TMQ II.iv., p83 where some uncertainties and confusion were dispelled.

Caution

(1) Technology and equipment designs change constantly and what I found may differ from what you will encounter. So my purpose is, hopefully, to point you in the right direction, but the ultimate responsibility for results rests entirely with you, so I would advise you to study your own case on its own particular merits and be sure you know exactly what you want to achieve and thoroughly understand how to do it. Carefully read any instructions provided with the drives and ask your supplier to clarify anything you don't understand. That's much cheaper than blowing something for not having read or understood the instructions!

(2) Most suppliers ship floppy drives with a cardboard insert where the disk normally goes. This keeps the mechanism and heads from thrashing during handling. In double-sided drives it also keeps the heads from knocking together with possible damage if the drive were to be jolted. Keep this cardboard in place as much as possible until the drives are in working position but do not forget to remove it before you apply power to the drives.

Supplier and brand

I happen to favor a specific brand of drives. That doesn't mean there are no other good brands, it's just that I have had very good experience with it and it's what I 'm familiar with.

I will be covering the use of half-height TEAC 5-1/4" 2x80 floppy drives from Aerocomp, which I have used exclusively and which I have recommended to friends who have used them to their entire satisfaction. I have been told Aerocomp "burns them in" for a number of hours and also gives them a thorough check for adjustment and performance. For

newcomers to the game who don't know what "burning in" means: Electronic components, such as resistors, capacitors, diodes, rectifiers, chips and so on, are tested during manufacture. Sometimes a part which is marginal in quality might manage to sneak past inspection and get installed. Experience shows that such parts will very probably fail within a few hours of being put to work, so the "burning in" consists of running them under power to weed out any weaklings. I value such a service greatly as it can save headaches caused by component failures after the equipment is commissioned.

I have also found Aerocomp very willing and competent in answering any questions on the phone.

TEAC drives from Aerocomp were also endorsed in the LSI Column in TMQ II.ii., p32, left column, third paragraph. I have used them several years in some extremely intensive work and have never had a single problem, not even one input/output error! Yes, I mean that literally. Furthermore, I am averse to head-cleaning disks, and extremely allergic to their scratchiness so never use them. I am afraid of the heads getting scored which could lead to all sorts of operating problems; the heads come with a highly polished finish that obviously needs respect. So how did my heads stay clean? Probably because I use brand-name disks manufactured as double-sided, no brand X or cheapies, just the regular Tandy disks for the number of sides and tracks that I use. I would guess that they have very good coating which I have never known to peel, score or flake, which doesn't scratch the heads or leave deposits and which probably has a built in lubricant. Avoid using both sides of a disk sold as single-sided; the "other" side may not have the necessary qualities.

There are other brands and capacities of drive available on the market from other suppliers. If anyone were to decide to use them, they should bear in mind that I am talking of a specific brand and supplier and of 80-cylinder 5-1/4" half-height double-sided units and that they may have to adjust things accordingly if they go a different way from mine.

Computer heating

I have always felt, since I got my first model III, and later with the 4 and 4D, that the internal drives provided with the computer: 1) Tend to overload the computer power supply especially on longish write or read sessions, (I have seen model 4's where the readout jittered whenever a drive started up), 2) The power-supply load due to the drives, particularly the rotation and stepper motors, generates additional heat and 3) The drives themselves, especially the motors, give off heat inside the computer. I recollect, when the III was introduced, several magazine reviews stressed the possibilities of overheating due to those drives inside the case. Electronic components don't like heat and can fail easily if it becomes excessive. So when I got my own drives, I simply disconnected the original drives zero and one and left them there in case a computer ever had to go back to a service center, as Tandy might not service a unit with alien drives inside. My own drives zero through three were all installed outside and I used drives provided in their own enclosures (two half-height drives per enclosure) with their own independent power supplies. At times I have run them all so hard for hours on end that I feared overheating so I set up a household fan blowing on the cases to keep them cool. If I had used internal drives that way, maybe the computer innards would have been fried! It is interesting to note that many newer computer models are provided with internal fans to avoid component overheating.

Bootup disk

If you are changing drives zero and one from single to double-sided and/or from 40 to more cylinders in the case of III's or 4's, or perhaps from double-sided 40-cylinder to double-sided 80-cylinder on a 4D, you will require at least one bootable system disk formatted to suit the new drives. Don't work yourself into a corner and end up with new drives and no disk you can boot up with! If you are installing four new drives, install numbers two and three first and format and prepare a bootable disk in drive two, for instance, with the existing system disk in drive zero. If you are only installing new drives as zero and one, hook one up temporarily to prepare your boot disk.

Although it has been described before in several places, there may still be folks who don't quite remember how to set up a bootable double-sided disk. From TRSDOS 6.2, LDOS or LS-DOS, format your new system disk in a new drive specifying number of sides and cylinders to suit. For example, to format double-sided 80-cylinder in drive 2: `FORMAT :2 (SIDES=2,CYL=80)`. Then `BACKUP SYS0/SYS:0 :n (S)` (n=newly installed destination drive). That must be the very first thing you put on your new system disk (FORMAT will already have put BOOT/SYS on cylinder zero and DIR/SYS on a central cylinder, unless you had some reason to place it elsewhere with the DIR=n parameter); the system needs SYS0/SYS to be able to boot and if it gets put somewhere else it may end up on the wrong side of the double-sided disk and the system may not find it at bootup, when it first starts up expecting to find a single-sided disk in drive zero. SYS0/SYS is not required on a purely data (non-system) disk.

Then `BACKUP :0 :n (NEW,I,S,Q=Y)` still using your DOS system disk as source in drive zero. Answer the prompts with Y or N according to the files you need on your new bootup disk. Copy everything you may conceivably need including SYSn/SYS, BASIC, utilities, etc. That will give you your new double-sided system disk. If you are not familiar with what each SYSn/SYS file does (for the 4's its in the Technical Reference Manual), copy them all just to be sure. You can easily delete any unwanted ones later. DOS files are password protected. If you have the passwords handy, you can use REMOVE, otherwise PURGE will do it.

Once you have your new drives set up, you can then boot up with your new system disk, check over the configuration you want to use, and then SYSGEN, as you will have lost your original SYSGEN by backing up by class instead of mirror image.

Those who are using drives 2 and 3 should also remember, before SYSGENing the bootup disk, to enable then when the DOS requires it (see, for example, the SYSTEM (DRIVE=n, ENABLE) command in the TRSDOS6/LSDOS manual). As an alternative, LSDOS6.3 users may care to use the patch shown in TMQ I .iii, p61.

Obviously, you should remember to copy necessary files from your other working and data disks to your new disk format.

If you are changing over all drives to 80-track, remember that at times you may need to read 40-track disks, such as when you purchase new software and want to back it up to the 80-track drives. There are several ways you can do this. Personally, I have RD4O/CMD, part of the MISOSYS PRO-ESP package and READ40/CMD, part of the MISOSYS LS-Utility package, which allow an 80-track drive to read (not write to) 40-track disks. I have also known friends buy one external 40-track drive or adapt one removed from a computer and share it among them whenever they have occasion to work 40-track disks.

Getting prepared

If you decide you can live with the original drives in your computer, and only wish to add external drives two and three, you have it fairly easy. If you decide, as I did, that you want all four to be new drives, or if you only wish to replace drives zero and one, you have to go further and open up your computer and get involved with its insides.

If you have previously been inside computers or if you are a good dabbler in electronics, you should have no problem, as what you have to do is fairly simple. There is no soldering and no leads or connections to cut, so it is easy to restore things to their original condition if ever the machine has to go to a service center for repairs. However, if you feel perhaps a bit scared of getting entangled, enlist the assistance of someone who can help you, guide you or even do the whole job for you. In opening the computer, you will be breaking the sticker that covers one of the screws, which will void any existing warranty.

Tools and Supplies. These are simple and vary according to the details of the job. Specific items are stated under each heading. You will need a medium Phillips screwdriver, perhaps a simple set of jeweler's screwdrivers available at many hardware stores (mainly for prying at close quarters), a grounding wrist strap available from Tandy for a couple of dollars or so to prevent any static discharges which might damage the electronics, which you should wear while working on the open computer, some thin hookup wire (bare or insulated) available at Tandy (for extending the wrist-strap connection to a ground) and perhaps a pair of long-nosed pliers on certain computer models as described later. Don't be tempted to improvise that grounding strap; the commercial one has a resistor in circuit to limit shock in case you were to touch a live part while grounded; it is unsafe to connect yourself direct to ground without an intervening resistor.

Control cables. If you make your own drive interconnections to the computer, you will need flat 34-conductor cable but if Tandy doesn't have it you can get 36-conductor and remove two. It comes in rolls of a few feet (six, if I remember correctly, which should be ample for the job). If you can only get flat cable with more than 34 conductors, simply cut lengthwise with a pair of sharp scissors, at one end, exactly between the conductors you won't need and the rest of the cable. This only needs to be a short cut, about 1/2" long, but be sure you cut exactly in the "valley" between conductors. Then peel off the unwanted conductors, as easy as peeling a banana. However, before you go too far, be sure your cut was correct; an incorrect cut may cause the insulation to be torn, baring the copper strands, which is totally unacceptable. When you cut the cable to length, use a pair of sharp scissors; avoid running a knife across the cable to cut it, as this tends to tear the strands and may produce accidental short circuits between conductors, sometimes hard to detect. Cut at a right angle to the cable length: if you don't have a craftsman's eagle eye, mark the cable first with a soft pencil, crayon or tape and a square, to guide you. A correct cut will help when you fit the connectors.

Whenever you handle those flat cables treat them tenderly, as crushing or tugging may render them unusable. Whenever you plug a cable underneath the computer, be sure it isn't twisted and carefully lay the computer down again to avoid damage.

I found that the most convenient arrangement with twin drives purchased in an enclosure is to mount them so the disk slots are vertical and, in fact, the drive cases came with rubber feet to position them this way, which allows cooling air to circulate freely through the grilles. When inserting a disk, you would then do it with the label facing left, so probably the best place to sit the drives is to the right of the computer. Left-handed users may prefer the left side but then the disk is inserted with the label out of sight. Whichever way it is done, think of this when estimating the length of cable to cut. The cable has to be just comfortably long enough, with a few inches to spare, to connect computer to drives without tugging, but no longer. This cable may pick up or radiate RF interference, so avoid excess length.

If you have drives outside the computer, in more than one enclosure, keep the enclosures separated at least an inch from each other and from the computer case to allow cooling air to circulate around them and through any side grilles.

You will need connectors for the cable ends, described later. When installing them, previously check carefully to see that

all teeth are accurately aligned. The teeth bite into the cable insulation, displace it and make contact with the wire inside; tolerances are close and alignment is important. Poor connections can cause problems and may be hard to trace.

The Tandy system for control cables on many of these computers requires the omission of certain pins from the cable connectors for drive selection. TEAC drives (and perhaps other brands) do it differently and accept all pins. See the section on Drive Jumpers.

To install the connectors, open them by gently prying the end tabs that hold them together. Then place one section of the connector near one end of the cable, inserting the cable carefully so that it fits exactly within the space assigned to it in the connector. One of the connector sections is made so the cable fits accurately so it is properly aligned with the teeth without sideways play; this is the first of the two sections you should insert the cable in. Be sure the connector is square with the cable length; then holding connector parts and cable together firmly and carefully, put the connector in a vise with parallel jaws (be sure the jaws can't damage the connector) and tighten evenly until the locking end-pieces snap into place. If you don't have a vise you may be able to use adjustable mechanic's pliers (keeping the jaws as parallel as possible), squeezing in various places until you get the same result. Then repeat the procedure at the other end of the cable with the other connector. Be gentle, those connectors are a bit delicate. Don't be tempted to join the connector sections by hammering; unless you are an expert tool user, you'll probably end up with a smashed connector.

When fitting connectors, leave about 1/16" of the cable projecting beyond the connector.

Getting the connector correctly plugged into the drive. There's a right way around to plug into the drives and there's also a wrong way. The manual I got with my drives states that a reversed connector can't damage any circuits. As you test your new drives, keep your finger on the power switch and instantly switch off if the drives grumble or make any strange noises. At least, that's the way I did it and it worked fine. But don't push your luck and switch off instantly at the least symptom of trouble or queer noises, reverse the connector and try again. Don't make this initial test with a valuable disk in the drive as a reversed connector may erase something. Once you have found the correct way around, mark the connector (e.g., dab with a felt pen or whiteout fluid) so you will be sure of reinserting it correctly after any disconnection. If your drives run continuously with the light on, see "Terminating Resistors", below.

Grounding. If you purchase all external drives in their own enclosures with power supplies, the drive frames will usually be grounded via the enclosure and the ground pin in the power supply plug. If drives were to be mounted any other way, care should be taken to ground them by means of the lug which usually is fastened to the drive frame at the rear. Wire size is not critical; use insulated wire just in case it were to make accidental contact with any live parts. Grounding is required to prevent the sensitive drive electronics picking up any neighboring electrical interference which might upset reading and writing. In the case of drives mounted inside the computer, grounding is usually to the nearest computer ground terminal.

Installing drives two and three

If you are only purchasing external drives 2 and 3, you have a fairly easy job and won't need to open up the computer case. Order your two drives, complete with case and power supply, and specify with your order the type of computer and the numbers you will be using the drives as. If you further want to avoid work, get a price from the drive supplier for a cable for connecting your drives to the computer. I found I could save a few dollars by making my own, so I cut a length of that cable I got from Tandy and attached a 34-position card-edge connector, Tandy cat. #276-1564A, at each end.

Plug one end of the cable onto the card edge underneath the computer, marked as the external drive connection (so the cable exits naturally towards the computer rear without bending around the connector) and plug the other end onto the card edge behind the drive case. Note previous statements about establishing the correct way around to plug the connector and item (2) under Caution.

Installing drives zero and one

Planning the job. You are switching drives zero and one which came originally with the computer. I explained above that I recommend leaving these installed in case the computer has to go to a service center. But there are other reasons. If you buy bare TEAC 5-1/4" drives with the intention of installing them inside the computer, you may find on some computers that they are longer than the originals and that rather severe surgery has to be performed on the tower that holds the drives, which in turn may mean further disassembly and removal of the tower for attack with a hacksaw. That may not be a tough job on some models where parts of the tower are plastic; but I've seen 4D's where the tower is heavy gauge sheet steel all riveted together, and that is really tough to carve. The screw holes in the tower may not fit the drives and will need some very careful work to drill accurately. Drives must be mounted so the mountings don't put the drive frames under tension, otherwise the frames may warp and throw the drives hopelessly and incurably out of adjustment. Then you may find that the mounting screws you need have metric threads and are hard to find. The length of the screws must not be excessive as otherwise they can contact and ground live parts inside the drives which will go up in smoke. If you are a master mechanic you may just get the fronts of the drives lined up with the original windows in the case, but in any event you will have to use cardboard or something else to cover the surplus window space to prevent dust entering; dust, disks and drives don't go well together. And you must

remember to leave enough clearance at the front of the computer to enable the cover to be replaced. You may have severe problems leading the flat cables to the rear of the drives due to the extra length of the TEAC drives. You may save something by not buying the external case and power supply but end with a patched up job and headaches which more than offset your savings. My attitude is, unless you are an expert at these things and very adventurous, don't do it!

Another reason for using all external drives, and it has saved me money: My work requires hardware reliability, and I must always have a standby computer to keep going in case one breaks down and spends some time in the shop. If a working machine breaks down, I can move all my four drives over to a standby and keep going.

If you may buy another computer in the future, judicious planning ahead may enable you to buy drives you can use both now and on whatever other computer you plan on, in which case satisfy yourself of compatibility by checking with your intended supplier(s) as necessary.

So I hope you will put new drives zero and one outside, leaving the old ones in, and I will continue on this basis.

However, some of my statements regarding heating may not apply in your case and you may feel that your usage is not heavy enough for it to be a problem, that you are skilled enough to handle the job and insist on putting your drives zero and one inside. In that case, you may have an easier job with 3-1/2" drives. Check with your supplier for availability of brackets to adapt the drives to the computer and of faceplates for the drives to mimic 5-1/4" full height drives to suit the windows in the computer front. Your drive supplier may also be able to provide the correct mounting screws. But, as I stated, my own experience and this write-up are based on 5-1/4" units. There may or may not be similarities with your 3-1/2" drives.

Making the change. To go about your job, you need ample space all around the computer. It's frustrating and annoying to work when there isn't enough room. So arrange things to give you plenty of working room and space to place a seat to work comfortably. Discomfort and cramped quarters are no sort of ingredients for working on computers and can cause distractions that in turn cause errors.

Switch off the computer at its own power switch; remove the plug from the wall outlet and don't plug in or switch on again until you are positively sure it's perfectly safe to do so. Talk for kids, you may say; yet I have known an experienced worker blow an expensive motherboard for forgetting that!

Before you begin opening up, have your machine switched off for about half an hour to prevent electrical shock. The CRT (image tube) operates at high voltage and there are capacitors that may retain a charge after switching off and which may take time to discharge. In any event, avoid poking around unnecessarily with fingers or tools in the upper section that contains the tube, just in case.

Also, before you begin opening up, have pencil and paper handy to make notes of everything you remove, to be completely sure of replacing things as they should be and in their right places. It's easy to forget to replace something and infuriating to close up and find you have to start disassembling again because you forgot an item.

Get the grounding strap on your wrist and extend the cable with a length of hookup wire until it comfortably reaches the nearest ground connection with plenty of spare length to allow you free movement; if your premises have a ground connection as many do, perhaps the handiest grounding point will be the metal screw that holds your power outlet faceplate on. If you don't have a grounded system, or if you are not sure, continue that grounding hookup wire to the nearest water faucet (**not** a gas fitting).

In this paragraph and the next, right and left side mean when facing the front of the computer. Turn the computer on its left side to obtain access to the bottom. Unplug any peripheral cables from the computer such as printer cable, RS-232 interface cable, etc. Remove three machine screws from the front edge and two from the sides near the front. Remove five sheet metal type screws from the back of the sides and the rear edge of the computer (one is usually under the warranty tag). Some screws are recessed. If there's a screw in the computer back, remove it also. Note the positions the screws come from for subsequent reinsertion.

Now, if you have never done this before, go extra carefully and gently. Lay the computer on the table again in normal operating position holding it from underneath and also holding down the top, which is now loose. Then grasp the upper part by both right and left sides and lift it straight up, slowly and carefully as you will be also lifting the image tube which is glass, contains a high vacuum and can implode and scatter glass if broken. When you have lifted the upper part a distance a little less than the computer height, swing it over to the left, rotating it counterclockwise, so it comes to rest on its left side on the table close to the left of the computer base. Your movement will be restricted by the cables that connect the base to the video system, which usually have rubber band(s) attached to guide them back into place when you reassemble. Don't disturb the rubber band(s). One thing that may hinder you is the lock knobs on the disk drives which may catch on the edges of the drive windows, and you may have to wiggle the upper part to get it past them; some types of drive may need the knobs in locked position and others in open position.

From here onward, "right" and "left" sides mean as viewed from the rear of the computer from which you will be working most of the time.

At the rear of the open computer you will find, occupying most of the back, the sheet aluminum RFI (radio frequency

interference) shield, which has to be withdrawn. Several screws must be removed from the top and ends, usually 6. Then gently swing the top of the shield towards you and lift it away. Sometimes there is adhesive tape on the bottom edge which must be separated for removal. On some models, there may be one or more plugs that have to be disconnected. Don't forget to carefully note their locations and orientations and to replace them when reassembling!

For the benefit of those who are unfamiliar with the innards: You have now bared the motherboard; treat it carefully and with respect; all those parts are quite delicate and should be left alone and kept away from unless specifically necessary to work on the board. Motherboard repairs can be expensive!

Now locate the 34-conductor flat control cable that leads to the rear of the two disk drives, towards the left side, usually quite visible. One end will have two connectors plugged in at the rear of the drives. Do not confuse this cable with the keyboard cable which goes downward and under the drives to the front of the computer, and which does not concern us. Depending on the computer model, the drive cable will be plugged in, at the non-drive end, in one of two ways: (a) At the upper edge of a vertical board located behind the motherboard (typical of III's and some 4's), or (b) at a header type connector, consisting of 34 pins in two rows with the pins at right angles to the motherboard, in the lower left-hand corner.

If you have (a), you will probably need a pair of long-nose pliers or hefty tweezers to unplug the cable. Make a temporary mark somewhere with soft pencil or pieces of tape to show exactly where the connector is plugged as you will be replacing it with another and it's not too easy to see the location (a flashlight is almost a must). Unplug the connector; it may not come readily so try easing it loose gradually first from one end and then the other several times. Pull on the connector, not on the cable which is rather delicate and easily damaged. Don't nip the cable with the tool.

If you have (b) try pulling the connector off with your fingers and if necessary ease it loose very gently from both ends, a little at a time, prying with a small screwdriver, taking care not to distort the pins. Be careful on some models where components are mounted hard up against the header pins and might be damaged by careless insertion or removal of the connector. Pull on the connector, not on the cable. Also bear in mind that you are pulling on the main board which may break if you overdo it, so prefer gentle prying to hefty tugging.

Once the cable is removed, either as in (a) or (b), fold it out of the way and tape it in place so it won't interfere with the replacement of the computer cover or the RF shield, and simply forget it.

There are still the two power connectors, usually white plastic female four point plugs, that connect the power supply to the rear of the drives, one per drive. These should be removed if reasonably possible. It may be a very awkward job on some models, and remember you may have to replace them if ever you take your machine to a service center. If this looks too impossible, just let them be. The drive electronics will still load the computer power supply, but the rotation and stepper motors won't run (since the control cable is disconnected) and the motors are the major part of a drive's load.

Those power connectors are usually below the board, pressed in tight. If you do attempt to remove them, go easy on the amount of force you apply; the receptacles are attached to the printed circuit board which might break if you overdo it. Prying may help.

If you decide to remove drive-mounting screws to get at these connectors, take care to prevent the drives crashing around and into each other as the screws are removed; use padding material such as cardboard, plywood, etc., between and below the drives. I have found it reasonably easy to loosen and move the drives on III's and 4's but 4D drives seem much harder to work on.

You now have to cable up for outside drives zero and one. You will need a length of 34-conductor cable to connect them to the computer innards. See previous remarks about establishing the length, removing surplus conductors and fitting the connectors. If you have arrangement (a) you will need, at the computer end, a Tandy 34-position card-edge connector, cat. #276-1564A; if you have arrgt. (b) you will need a Tandy 34-position computer connector, cat. #276-1525. In both cases (a) and (b), at the other end, for plugging onto the drive you will need a 276-1564A. Install the connectors on the cable as previously described and at the computer end carefully plug in, in the same location as the original drive zero and one cable was removed from. Particular care is needed in the case of arrangement (a) as things may not be too visible or accessible, so be sure the connector is driven home correctly aligned and all the way in.

The cable has to be arranged to go down to the rear edge of the computer baseboard, quite vertically, than exit towards the rear between base and top, and has to be shaped a bit so it will fit comfortably in the convolutions between them. If this is done carefully, there will be no problem when the top is replaced. Note that the way things are shaped, the top may tug at the cable a bit when it is replaced, so try to leave a little slack inside to prevent this. With arrangement (b) you may have to trim the bottom edge of the RF shield slightly with shears where the cable exits from the header connector; be sure to remove all burrs or sharp edges with a fine file to avoid cutting into the cable. Also, with arr. (b), plug the connector onto the header so that the cable exits downward out of the connector. Whichever arrangement you use, before closing up the computer, tape the cable in place so it won't get displaced when you replace the cover.

Replace everything you removed when you opened up, being sure not to forget any plugs, and you should be ready to test.

See previous cautions about getting the connector the right way around at the drive and item (2) under Caution, above. Again, when reassembling be very careful of the image tube to prevent very unpleasant and expensive consequences from breakage. Before closing up, check to see that the socket on the neck of the image tube, at the extreme rear, hasn't been displaced accidentally, and see it is firmly seated by pressing gently. Reread the item in this section re possible shock and remember that the socket sits on pins which are embedded in glass, so be gentle.

Also check that there has been no displacement of the card-edge connector on the video board near the image tube and snug it up if necessary.

Opening the drives

If you get drives complete with case and power supply, and have to open them, be very careful. Remove and replace the outer cover slowly and carefully, straight up or down taking care that the edges or corners of the covers can't jab any sensitive parts inside which are quite close; keep fingers and tools far away from the electronics and mechanism, except where strictly necessary to make any changes. Note that the screws that hold the drives to the base are special for those locations and that they must not be exchanged for any others lest the drives go up in smoke as mentioned previously. If you remove the drives, handle them with great care by their edges and avoid contact with the electronics or mechanism. Do not touch any mechanical parts, which are very carefully adjusted and will serve you as loyal workhorses if respected. Remember that perspiration may be acid and corrode sensitive mechanical parts. Do not attempt to adjust or align anything unless you're an expert and have suitable equipment and a service manual! If you have to change any jumpers on the board, grasp them carefully with suitable pointed pliers or tweezers and don't drop them. They are small and one dropped, especially on a carpet, is almost certainly a goner.

If you were to leave the drives outside the case for any length of time, be sure to cover them carefully with a lint-free dust cover. Dust, cigarette smoke and ash and dirt in general are arch-enemies of drives and disks. Which, incidentally, might also be a good reason to regularly protect all your hardware with dust covers when not in use for any length of time (and another reason to persuade smokers to quit!).

Drive jumpers

When I installed these drives some years ago, I received instructions regarding jumper linkages on the drive boards. I have also noticed, on drives purchased by friends, that some of the instructions vary from single to double-sided drives and from 40-cylinder to 80-cylinder. Some drives come with head-loading solenoids and others don't. In any event, each purchase I have seen has come with an instruction booklet or leaflet that clarified things, but which may require opening up the drives and changing jumpers. No problem, provided it's done carefully.

When I first got drives with head-loading solenoids, I thought it might be a good idea to set jumpers so as to have the heads load (come in contact with the disk) when a drive was selected rather than when the motor-on signal went out to all drives, thus preventing a head rubbing against a disk except when the drive is selected. That didn't work well in practice as sometimes when a file was being copied sector by sector from one disk to another, or when DISKCOPY copied a disk track by track, drive select has quickly changing back and forth and the heads were being rapidly loaded and unloaded in quick succession, making a loud clatter and showing the drives were obviously being used in a manner they were never meant to be used. So I changed head load to respond to motor on instead of drive select, even though some heads were now sometimes rubbing on disks that weren't being read or written to. As it turned out, those drives treat the disks so gently that there was never any problem with disk wear or scratching.

On some TEAC drives I have seen over several years, the manufacturer recommended using jumper U2 only on TRS-80's or IBM PC's; that was for TEAC FD55B 2x80 cyl. drives. More recently, I have encountered a later version, FD55BR 506, where jumpers are somewhat different; I had good results in setting head loading as I wanted it by removing jumper HS. In any event, if you purchase your drives from Aerocomp and can't figure out the jumpers, a call to their tech will probably fix you up quickly.

Regarding the setting of jumpers for drive selection: Note that the TRS-80 system requires, on many models, that the card-edge connectors be short of certain pins for drive selection. The TEAC system does it another way, by moving jumpers on the drive board, so all connector pins can be active. Just in case you do get to moving jumpers on the drives, be advised that, on all TEAC drives I have seen, the jumpers have to be placed in locations DS0 and Ds1 for drives zero and one, respectively, and also in locations DS0 and Ds1 for drives 2 and 3, respectively. This didn't seem right when I first saw it, but that's the way it works. The reasoning is probably that drives zero and one are the first two on one cable and that drives two and three are the first two on another cable. Those locations may be labelled DO and D1 in later versions of Teac drives.

CAUTION: If you change drive-select jumpers around, be quite sure you don't have the same drive number on two drives on the same cable, which would thoroughly confuse the drive controller and produce totally unpredictable results, some of which just might be harmful.

Terminating resistors

Another point that may need watching: On a three or four drive system, the last drive viewed by the computer as external (#2 or #3) must have a terminating resistor. Typically

the absence of one is shown by drives revolving continuously with the drive-select light on. This resistor is located on the drive electronics board towards the rear. On my drives it's a 14-pin DIP plug-in pack marked IAM E3317, plugged in so the marking is in readable position when looking from the rear of the drive. I have seen, through the years, conflicting opinions as to exactly where the terminator should go and if there should be one on each drive or not. On my own 4-drive setup, it doesn't seem to matter where it goes as long as there's one present, but I can't guarantee that will work with every setup. Perhaps a good starting point would be the last external drive in the system. In the specific case of drives zero and one, any terminating arrangements on models III, 4 and 4D are made on the computer drive controller so one doesn't need to be concerned with terminating them.

Drive alignment and rotational speed

Drives may drift out of alignment after a time. Typically, that shows as I/O problems when switching disks between drives. But there can also be a treacherous situation which may pass unnoticed: If a disk is formatted, written to and read in a drive which is appreciably misaligned, and always used in that same drive, one may not notice that anything is amiss until one attempts to use the disk in another drive.

There are several ways to check drive alignment. There are commercial disks for testing it, which some users swear by and others swear at. Probably the only really accurate way to check it and correct it is a competent technician equipped with brains, full test data, an oscilloscope and a precision test disk created in a lab. For my purposes, I created the following homespun setup solely as a first alert to possible misalignment (not for curing it):

When I installed my set of four new drives, and while they were still brand new and presumably perfectly aligned, I formatted a disk and wrote to it an assortment of files; I copied several files to it in the lower numbered tracks (the outer ones). Then I used CREATE to install a dummy file and fill blank tracks so that I could next copy a number of other files to somewhere around the center of the disk. Again CREATE and a number of files on an inner portion (high-numbered tracks) of the disk. Then I REMOVEd those CREATEd dummy files, leaving me with a disk with an assortment of files at the periphery, center and interior of the disk where alignment is usually checked.

Then I made a number of tests with that disk for file readability in all my drives, which, as would be expected from new and properly aligned drives, were completely successful. That became my alignment checking disk which I filed away for future use. When I wish to test my drives for alignment, about once or twice a year, I use it to test for readability in all drives assisted by RDTEST/CMD, part of the MISOSYS LS-Utility Disk. It hasn't happened yet, but if ever I were to get readability problems, I'd send the drive(s) to an expert shop for realignment, probably Aerocomp who supplied them.

As I said before, my system is homespun and I claim no particular accuracy for it as an alignment check; I simply regard it as advance warning of misalignment.

There may be other faults in a drive, apart from alignment, that may cause input/output errors. However, drive alignment is very important.

Another important feature is drive rotational speed. In the interchange of data between computer and floppies, there are conditions which demand that the drive rotate at the standardized speed of 300 RPM within narrow tolerances. Drives maintain their speeds in various ways. Some allow adjustment within certain limits, others have systems that monitor and adjust their own speed internally; in the latter, speed cannot be adjusted.

One problem that an accurate drive speed causes is that, at times, the drives seem to fall asleep and turn with the light on when nothing is happening, due to synchronization between system interrupts and drive rotation. Before getting my TEAC drives, while I was using the original computer drives, I never noticed such a thing, perhaps because they were off speed. But when the TEAC's were installed, falling asleep became immediately evident. There are two things one can do: Either live with it or fiddle with the drives to avoid it. Personally, my attitude is to let sleeping dogs lie, so I left things as they were. Those wishing to change drive speed should be very sure of what they're doing, provide themselves with means for measuring the speed accurately and be certain that the drive is not the self-regulating type which cannot be adjusted for speed. They should not overdo adjustments lest they jump out of the frying pan (sleeping drives) into the fire (I/O errors). [TRSDOS 6.2 and LS-DOS 6.3 as well as LDOS 5.3 all have an operating system command to overcome the problem of floppy drives aligned to exactly 300 RPM. That's the SYSTEM (SMOOTH) library command. What that command does is force the DOS to disable CPU interrupts sooner than would otherwise occur in the floppy disk driver. The side effect is that the communications driver, being interrupt driven on received characters, will tend to lose characters occasionally even at 300 baud during a COMM session with the dump-to-disk parameter on. If you use COMM and want to concurrently write to disk during receipt of characters from the communications line, you will have to invoke SYSTEM (SMOOTH=OFF) during the communication's session. - editor]

Drive speed can be checked with software available on the market. I have Howe's System Diagnostic and there may be others. In several years I have used my TEAC drives there has never been any visible deviation from the original of exactly 300 RPM.

Model 4 Video Bloom

Fm Jim Beard: Joe:, My Model 4 is showing a strange problem in the video. The brightness has increased gradually over the last few months, and for the last week, I can't lower the brightness enough to blank the retrace. The contrast is low, but not terribly so. The last day or so, it has gotten worse as I work, to the point tonight that it is obvious that within a day or two the screen will completely white out. From the technical manual, I estimate that the contrast is a simple gain pot on the video in, and the brightness pulls down a CRT grid with a pot to ground.

So, I don't see an obvious cause. Troubleshooting the video board can be tough without a test Model 4 hacked up on the bench because it is so hard to reach when in place. What would you suggest? I could always get my other video board out of the parts bin, which hasn't been touched in 3 years.

Fm Joe Kyle-DiPietropaola: Jim, I seem to recall a problem like this discussed before, but I recall the solution as being one of the fixed resistors shifting in value or maybe one transistor going. You can get enough slack in the cables to at least probe some points by flipping the case top on its side, immediately to the left of the computer and leaving the screws out of the video driver board.

Fm Jim Beard: Sounds like motherhood to me. There is only one transistor, a high voltage video amp supplied by an 80 volt source. I'll start with it if no resistors check bad.

Fortunately, my brightness and contrast controls work OK. In fact, I can temporarily improve things by turning to full brightness for a few seconds and then turning the brightness all the way down. I finished my DWP 230 printer driver for Scripsit Pro that way. Once I fix it, I can then USE my new hot-shot printer driver.

Fm Fred Oberding: Jim, you have described a classic case of a Model 3 RCA video board going south. The RCA board was used in early Model 4's also. At the opposite end of the board from the connector, just on the other side of the flyback transformer, you will find two resistors; R-516, a 1 meg 1 watt & R-518, a 470k 1/2 watt in a voltage divider ckt. Either one or both have changed values or have opened up altogether. Replacing them should put you back in business.

Fm Gary Phillips: Jim, Check your power supply before the video, Jim. If the 12v out from the supply is creeping upward, you would get those symptoms.

Fm Joe Kyle-DiPietropaola: Ya, as I recall, part of the fix was to replace with the next larger wattage.

Fm Jim Beard: Fred, I replaced R516 with four 1 Meg 1/2 watt in series-parallel, and R518 with a 470K 1/2 watt. R518 was originally a cheap carbon 470K paralleled with a 3.3 Meg. It read open on an ohmmeter, until a little handling got it read a shakey 1 Meg. It is like new now. I owe you one.

Fm Steve Lorenz: Fred, I saw your reply to another user here having trouble with his Model 4's video and recently my Model 4P has developed a similar problem. I am wondering if you can help? The problem is as follows: If you boot up the system without a system disk in drive 0 then the message about closing the drive door appears on the screen and this looks fine! But if you boot with a system disk in the unit once the unit activates the disk drive the video goes bad. After the drive completes its access the video continues to shake from side to side slightly. I have tried cleaning the pots on the video board and adjusting them but it didn't help. I suspect the trouble to be on the power supply. Any help you can give will be appreciated. I am electronic technician so tearing into the unit is no problem and I also having the tech. manual around here somewhere!

Fm Fred Oberding: Steve, it appears that you probably have a video Phase Locked Loop circuit going out of sync. It reads OK in 64 character mode but not in 80 character. Your 4P is a non-gate array I believe. There is a small trimmer capacitor, C-231, usually orange, near IC, U-148 that needs to be tweaked with a nonmetallic screw driver. Near the trimmer cap you will find a jumper on stakes E1-E2, move it to the stakes marked E9-E10, and adjust the trimmer until it locks or nearly locks in, and then move the jumper back to E1-E2. The display should stay locked in.

In rare instances, you may need to change the trimmer out for a negative temperature coefficient type. Now the difficult part, getting at that trimmer with the unit running! You will need to have the case off and the pan that holds the main logic board on-screwed and ready to drop. Boot up a disk, and very gingerly pull the power from the drives and remove the drive cable from the logic board - watch your fingers near the fan blades Now with the FDC cable loose, you can swing down the logic board pan; (assuming you already have the unit resting on its side opposite the drives) and you are ready to make the adjustment.

Model III Power Supply

Fm Jeff Hunsinger: Can anyone give me info on the Model III Power supply. I know that it supplies +12, -12, and +5 volts, but at what amperage? Could I use an IBM style supply without damaging my III?

Fm Joe Kyle-DiPietropaola: Jeff, The Model 3 uses two Astec-style 38 watt power supplies. I believe that the specs go:

+05v - 2.5A
+12v - 2.0A
-12v - 0.2A

for each supply. I may have the amperages for +5 and +12 reversed, but that really doesn't matter much for what we're considering here. Almost any IBM power supply will more than cover these needs, though the original 63 watt IBM PC supply might be a bit light. Of course, you'll have to figure out how to shoehorn the thing in there, or extend the cabling so you can mount it outboard. Make sure that you run the whole shebang on the new supply, don't try to run just original supply and the new supply at the same time. You may not meet the minimum load spec for regulation on the IBM supply, and grounding problems could get you into real trouble.

Model 4P Power Supply

Fm Robert G Strickland: Where can I pick up an extra PS for my 4P, aside from a direct order from National Parts? I know they were made by an outfit in Texas. I used to know the name, but now it escapes me. I have also seen some advertised thru wholesale outlets. I bought one once, but the +5v line had too much ripple on it for my Teletrends Modem; I sold it to somebody who wanted to trouble shoot it.

Fm Joe Kyle-DiPietropaola: Robert, the supplies were made by both Tandy and Astec, Astec was the original vendor and Tandy eventually produced them themselves from Astec's design. Both Jameco and Timeline used to have them (Astec 65watt version), but I don't know offhand if either still does. Jameco is (415) 592-8097, Timeline is (213) 217-8912. A surplus unit from either will run you around $30. A new one from Tandy or elsewhere will be more like $100.

XLR8er comments

Fm Frank Slinkman: Roy, concerning the XLR8er associated software, while they're obviously a great improvement over the Mod 4 without them, there are still a couple of deficiencies:

1. The BOOT command no longer functions properly. It puts you in Model III casette mode instead of booting up the system in Model 4 mode. Is this because the speeded-up system can't read the slow bootstrap ROM? This is a fairly minor problem, since there IS that orange button there.

2. BUT, after a boot, RAMDISK does not recognize the fact that the ramdisk is already formatted. If you type "N" to the "do you want to format" prompt, it refuses to install the ramdisk. I consider this a SERIOUS deficiency, because it allows no (reasonable) possibility for recovery of data if the program using/creating it crashes or locks up, forcing a reboot.

3. Sometimes, under LS-DOS, typing the letter "a" brings forth a whole passel of other characters as well. There doesn't seem to be any pattern to the characters -- either as to their quantity or nature. This hasn't caused any problems from JCL, nor does it occur using Scripsit or TED; so it appears to be a problem with the way the DOS accesses the keyboard. Of course, it's possible my "a" key is at fault. Has anybody else mentioned any similar problems to you?

As a heavy user of Super Scripsit, which still has the odd bug or two (under-statement of the year?), number 2 above is of great concern to me. Now if I could only find a way to cut down on the 2 minutes 38 seconds it takes my JCL routine to copy all the system, Super Scripsit and spelling checker programs and data to my ramdisk <hint, hint)...

Fm MISOSYS, Inc: The problem with the BOOT command going into the cassette mode prompt is due to the faster speed of the machine. I don't know if its due to not being able to read the ROM (probably not because that's what's displaying the message), but if you slow down the XLR8er prior to issuing the BOOT command, it reBOOTs fine. My guess is that the timing loops in the ROM boot code are executed too fast with the XLR8er running at maximum; thus, the ROM boot loader doesn't "see" the disk drive.

I looked at the BOOT command code. It executes an SVC_0. That SVC_0 does a few things. It first moves a small routine to 4300H. That routine switches the machine in to the ROM Model III mode, then does a RST-0. The RST-0 is then executed from ROM. If you have a 4P, then the RST-0 is executed from RAM becasue the routine noted above does nothing on a 4P. The RST-0 code in DOS 6 switches in the 4P boot ROM which then continues to execute the code starting at address 5. Therefore, in order to have the BOOT command switch back to a slower XLR8er speed, you ould have to come up with some code to replace the BOOTCOD stub routine. I don't think that has a great deal of priority to use up valuable low-memory space. However, you could probably add that code to SYS1. There's some space at the end. Grab THE SOURCE and hop to it.

The RAM disk should not "disappear" during a reBOOT. It may be your Model 4 doing that. Try to not lean on the RESET button too long.

There are also fast loading/dumping utilities in the LDOS forum which can load/restore a RAMDISK image. That may be (has to be) faster than BACKUP. Also see Michel Houde's contribution in this issue.

Fm Frank Slinkman: Roy, I press the reset button (EXTREMELY gingerly) always when all 10 banks are used and it is being used as the system disk.

Since I wrote that message, though, I've noticed that sometimes (not always), it stays intact if I use only banks 3-10. Let me experiment a little bit more with it, and I'll give you a more detailed report later.

Fm Roy Soltoff: Frank, Some Model 4s lose their refresh if the RESET button is depressed for too long a time. Don't know why. Maybe Joe knows.

Fm Frank Slinkman: But I DON'T press it too long! I barely touch the damn thing. I've noticed that if I use the top 8 banks (instead of the top 10) the ramdisk survives more often, but not all the time. Does make it sound like refresh problem, doesn't it. Or does it?

Fm MISOSYS, Inc: It does. Sounds like a local machine problem. Some 4s are more touchy to the RESET button than others.

Fm Joe Kyle-DiPietropaola: Frank, I've never really done an analysis on this, but the problem is that the Z80 (and 64180) supplies the refresh in software, via refresh cycles during M1 (opcode fetch cycles). That is, if the CPU ain't running code, you don't get RAM refresh.

On the Model 4 systems, the reset button is tied to the CPU RESET input, rather than generating an NMI as was done on the Model 1. I believe that the Model 4 hardware merely enforces the minimum spec on the width of the reset pulse, but does not limit the length. Since we're talking about a refresh period of two milliseconds, how long you **intend** to hit the button probably matters very little.

Fm Shane Dawalt: Joe, I wonder if it could be memory chip dependent too? I have held my RESET button on my pre-gate-array M4 for 5 seconds without killing the MemDISK image. It does go into reset as the screen clears but nothing else occurs until the RESET is released.

XLR8er Xperiments

Fm Daniel L. Srebnick: I would like to eliminate the need to use FIXBANKS or Rex Basham's HIBANKS in order to free up low memory, if this is possible. In TMQ V II.i p81 Dick Newman states "I recently installed an XLR8er" ... "ALPHA1.FIX" ... "allows me to remove FIXBANK." On page 83 of the same issue, Bill Schaper states that he also uses the AT patches with an XLR8er. On the other hand, in TMQ V II.iii, John Tollini asked you if the AT patches are usable with the XLR8er. You replied, "No!"

In light of the above contradictions, I thought that it would be worth trying out the AT patches (the revised LS-DOS versions with the two byte correction applied) with my XLR8er. After all, saving a couple of hundred bytes of precious low memory can be useful. I applied the patches, and sysgened a disk with RSHARD6 and the forms filter, both in high memory. The floppy still remains as the system disk after sysgen. An auto JCL file installs RAMDISK from banks 5 thru 10. The JCL ends and the fun begins.

A peek with MEMDIR shows that the high memory modules are gone! Instead, there is just a module labeled < unknown >! That is the best case scenario. There was another variance of events which I don't exactly recall at the moment when any attempt to access the hard drives resulted in a file not found -- because the driver was wiped out of Bank 0.

Fm MISOSYS, Inc: Anybody who thinks that the Alpha Technology memory board patches will work with the XLR8er is dreaming! Please don't perpetuate any rumors. On the other hand, there are now patches for the XLR8er, courtesy of Michel Houde. See the next subject...

A new XLR8er Software Interface

by Michel Houde

Fm Roy Soltoff: The following submission by Michel Houde has my gratitude and appreciation. The material is specific to those Model 4 owners with an XLR8er memory/speedup board installed in their computer. Michel has also provided a universal RAM disk software operating under the extended memory handler published previously in TMQ as well as a utility providing fast load/save capabilities of the RAM disk. Due to the length of the submission, this issue of TMQ will provide only the XLR8er interface patches so folks can get started on using them. Source code, for those who are interested, has been placed on DISK NOTES 3.2 which corresponds to this issue of *THE MISOSYS QUARTERLY*. If you have a 128K Model 4, you can use this RAM disk with EXMEM; if you have an XLR8er, you should switch over to utilize the interface patches presented herein rather than the software provided with your XLR8er. I have. Even if you have an Alpha Technology memory board, use the AT patches previously published in TMQ and this RAM disk.

I am making up a special Model 4 XLR8er software interface disk as suggested by Michel. Just return your existing Model 4 TRSDOS 6 XLR8er disk (and a return label) for a copy of that disk.

When I ponder at times whether it is worth it to still poke around in this TRS-80 marketplace, with sales declining and the future opportunities dim, it is the generosity and helpfulness of folks like Michel who just make it all worthwhile. I congratulate Michel on his work, and the obvious amount of time he has borrowed from his family will surely be respected by all those who benefit from his accomplishment.

Michel HOUDE
8, rue du Docteur Roux
60200 COMPIEGNE
FRANCE

Dear Roy, Congratulations on your new born baby. You must be glad it's a boy, aren't you? Family life is what makes it all worthwhile. I wonder if you would say that in English, but I hope you understand what I mean. You may have noticed it already, but when it comes to children, it seems that 2+1 is more than 3, as far as family time is concerned. Perhaps you recall I told you we have one girl and two boys, currently aged 7, 4 1/2 and 2 1/2. Remember: I am a teacher-researcher in Chemical Engineering at Compiegne University. You printed my letter in TMQ (Spring 87), and you also had me on the phone sometime in December 87 when I ordered an XLR8er.

I felt compelled to write when I read the new Summer issue of TMQ, received August 24th. Perhaps I can save you some work, as I resolved way back in January the problems everyone is having with the software as supplied by HiTech.

I have developed a set of patches similar to those you wrote for the Alpha Tech board. Only BOOT/SYS and SYS0/SYS are patched specifically for HD64180 use. That was a design requirement. The mods to SYS2, SYS12 and BU you published in TMQ are also required, but as they are related to timing, they are not CPU specific.

Let me outline what has to be done, what I did, and how I did it.

Let's first deal with the memory allocation scheme, because it will have an effect on the way we do other things. After a reset, memory is logically split between Bank Area x'0000'-x'efff' and Common Area 1 (CA1) x'f000'-x'ffff', the CBAR value being x'f0'. These are also physical addresses as BBR and CBR are reset to zero. The normal bank switching set by Model 4 hardware imposes a logical split at x'8000'. As we can't get rid of CA1, it seems logical to make it start at x'8000' and attribute x'0000'-x'7fff' to either Bank Area or Common Area 0 (CA0).

HiTech chose Bank Area, but it makes more sense to split between CA0 and CA1, as CA0 is bound to physical address zero, which is what it must be. Unfortunately, they also chose to make CA1 take up all memory, while switching in banks 0,1,2 (let's call these alternate memory, versus extended memory for the additional 256k). That's unfortunate, because it needs 2 ports to be modified, CBAR and CBR, which means more code. Therefore I decided to split logical memory between CA0 and CA1, by loading x'88' into CBAR at Boot initialization. Translation: make both Bank Area and CA1 start at x'8000', then CA0=x'0000'-x'7fff' and CA1=x'8000'-x'ffff'. These value will never change! Bank Area will not be used.

How do we access all that memory (384k)? Let's inventory it. First we have 2 standard 32k memory banks, split at x'8000', which are normally active. These are the Lower and Upper banks. The Upper bank is labelled Bank_0 by the DOS. Then we have 2 alternate 32k banks, labelled Bank_1 and Bank_2. Either or both can be swapped with either or both standard banks. Actually, the Lower Bank is never changed by the DOS (how could it function?), and only the Upper Bank (Bank_0) may be swapped with Bank_1 or Bank_2. As far as the MMU is concerned, these are always mapped at physical addresses x'8000'-x'ffff', with CA1 starting at x'8000' and CBR set to x'00', the swapping being handled by the standard OPREG$ port. It should also be mentioned that any of the banks 0,1,2, whichever is active, can be shadowed by the keyboard and video memory at x'f400'- x'ffff'.

Now comes the 256k extended memory located at physical addresses x'40000'- x'7ffff'. It can be accessed by setting CBR properly. The value held by CBR may be considered as an offset as it is shifted 12 bit-positions left (that's 3 nybbles) and added to the starting address as set by CBAR. For instance, x'38' shifted gives x'38000', add x'8000' and you get x'40000', start of extended memory. As ~~512~~ 256 divided by 32 equals 8, we can logically define 8 banks, labeled Bank_3 to Bank_10. Physical addresses are linked to bank numbering very easily: x'40000'-x'47fff', x'48000'- x'4ffff', x'50000'-.... up to x'78000'-'x7ffff'. If we take the 2 high nybbles of each starting address and subtract x'08', we get the value to load into CBR. Acceptable values are x'38', x'40', x'48', x'50', x'58', x'60, x'68', x'70'. Computation from bank number is easy: subtract 3, multiply by 8, add 38H, load CBR.

I decided not to change speed while accessing the keyboard. After performing numerous tests (including Rubin's test), I found out that adding one memory wait state and setting refresh rate to 40 T would lower maximum speed by about 8% (see report after discussion).

The preceding remarks lead us to the initialization part of SYS0, which happens to provide a 32 byte patch area at x'2194' and a convenient place to call a patch from x'1e79' (x'2147' and x'1e70' in 6.2). That's where we will set CBAR, zero CBR (warm reboot) and set wait states and refresh rate.

Byte I/O and physical disk I/O both bring in Bank_0 when called. As they use @BANK, that's all right. Now the tasker, as the interrupt handler is called, also needs Bank_0, but this time without the help of @BANK, which is too slow (must check bank number, poll function request, check BAR$, compute bank switching values, etc.). Surprisingly, it is very easy to handle, all we have to do is set CBR to 0, making Banks 0,1,2 accessible, actual bank 0 is brought in by the standard routine. We must of course save CBR current value. That's where our allocation scheme shines, as there is only one byte to save. Have a look at THE SOURCE, Vol 1, p.102. At x'1c0b' we find LD HL,LBANK$; LD A,(HL); LD (HL),0 then PUSH AF. We will instead do LD H,A; then IN0 L,(CBR); PUSH HL. Later on is the counterpart at x'1c3c':

POP AF; LD (LBANK$),A; we will do: POP HL; OUT0 (CBR),L; LD A,H; LD (LBANK$),A. The actual code is a little more intricate, as there are more bytes to deal with and we CALL two patches, which is more space saving than JP's. Please note that no additional stack space is needed, compared to the standard TASKER, which might be valuable in some cases.

Now is the problem of ENADIS_DO_RAM, the routine that deals with keyboard and video RAM. The first idea is to bring in bank 0, either on each call or on the first of subsequent calls. I did it, but it slowed things down significantly. After careful checking and deep thought, I decided it was not necessary to do that. Keyboard routines are: standard driver, type-ahead task, KFLAG$ scanner task. All 3 routines bring in Bank 0 before calling ENADIS_DO_RAM, either byte I/O or tasker. With one exception, the same is true with video routines: standard driver, cursor blinking, clock, trace, alive. The exception is @VDCTL, we must handle it. As with the tasker, we only have to set CBR to 0, after saving its current value. There is no need to save/change (LBANK$) as the @VDCTL routine is closed, with no alien calls. The worst that could happen is an interrupt, but then the tasker sets bank 0 anyway. However, in those days of stack space scarcity, I thought it would be useful to first check the current value of CBR. If it is null, it is not necessary to change it. As stated before, alternate banks can be shadowed by KB/DO RAM. This saves 2 levels of stack, and I think that the most frequent case is Bank_0 active. The cost is 4 bytes, but they might prove worthwhile.

One last point: although it costs 9 bytes, I did include a memory header, instead of enlarging the $FD module. It looks nicer when running MEMDIR. Anyway, total length of low memory additions is only 121 bytes.

You will find 3 files dealing with this subject, XLR8I/ASM, XLBOOTA/FIX, XLSYS0A/FIX. The 'I' stands for International as I have provided the data for French and German versions of LS-DOS 6.3. Hence the 'A' for American.

Like anyone, I was annoyed by RAMDISK/DCT taking up so much space in low memory. I wrote a program called ERAMDISK/CMD which has a few interesting features: its invocation is parameter driven, it supports 3 drive types, like DiskDISK, up to 30 banks, and a (HIGH) parameter can either force high memory usage, or force low memory usage (H=N) or let the program try lomem first and use himem if no lomem. It is even possible to disable it and recover memory, should the need arise. Typing ERAMDISK without any parameter will display a syntax help screen. At the very beginning of the source file you will find a full list of all features. If you examine the source code, you will see that a lot of error checking is included. I almost forgot to mention that it uses SVC 108 (@EXMEM), being kind enough to set the SVC vector, in the event the @EXMEM module being sysgened and not yet reactivated. Roy, if you had numbered it 125 (or 126, or 127) you would not have all those questions about @EXMEM not being SYSGENable. You forgot that SVC vectors 124-127 are SYSGENed, since 6.2. (124 is @WINDOW). [*Actually, I really didn't forget. I considered @EXMEM to be a SYSEM function; thus, it should have a system SVC number, not a USER number. I then cleared it with LSI. The problem you run into when using a USER SVC number is that someone else may have decided to use that same number.*] Which leads me to another program I wrote: PEXMEM (Page EXtended MEMory Management) which is a reduced @EXMEM, reduced to page handling, functions 3 and 4, same entry conditions as @EXMEM. It is SYSGENABLE as it uses SVC 125 and 108, and it does re-use its module if invoked and already there. More important: it checks the buffer address and uses double buffering only when needed. This is important when using it in conjunction with a RAMDISK, especially when RAMDISK is system drive, with most disk accesses being in system buffer loading overlays, directory sectors, etc. I say it is important because double buffering IS time consuming.

To have a comprehensive package, I include ERAMLD/CMD (Extended RAM Load and Dump) to transfer data between banks 1-30 and disk. It is parameter driven. Typing ERAMLD without parms will display a help screen. Your own BANKER/CMD should of course be part of such a package.

I also include SETX/CMD which is a replacement for SET180, but with a difference, it load at x'2600', and runs in the library overlay region. Can't understand why SET180 runs at x'7000'! Also, SETX will display previous and new status, when a modifying parameter is invoked. It does not try to modify FIXALL module as SET180 does.

Also included is HITACHI/ASM which builds macros for some of the new opcodes: IN0, OUT0, TST, MLT. XLR8I *GET's this file. As some other /ASM files need it I included SVC/ASM, which is a list of all 6.3 SVC equates, plus SVC macro definition.

As you can see, I wrote a whole set of programs to go with the XLR8er. I wrote them in January '88 (got the board on 31-dec-87) because, as I said, I was not satisfied with the original ones. If you find them good, especially the patches to SYS0 and BOOT, take them. I give them to you, commented source code included. As I said last year, I am not in the software business. I wrote the programs for me, but when I compare them with some that are published, either in TMQ or commercially, I find they are rather better. Take them as a gift for the birth of Benjamin.

Here is the short report on speed tests: From TMQ II,2 p54, Rubin's test (interrupts enabled) yields 131_s for "newer Model 4", 112_s for "Model 4P", 70_s for "4P with XLR8". I ran the test (interrupts disabled) on a "non-clustered Model 4" and a "clustered 4P", both with French keyboards. Old Model 4 rated 124.6_s and newer Model 4P 108.0_s. Theoretical timing should be 107.6_s with no wait state and 124.1_s with one wait state (standard Model 4/4P with Z80 running 4.055

MHz). The results show that both Model 4's have 1 wait state, and both Model 4P's have none. Same tests with interrupts enabled brought exactly the same values as yours. After installing the XLR8er on the "Old" Model 4, I ran the tests again with different settings (remember I'm a scientist).

Memory waits:	0	1	2	3
Rfrsh rate: 80T	68.3	71.2	85	111
Rfrsh rate: 40T	71.1	74.0	89	115

The result at max speed is close to yours (with EI). As can be seen, 1_ws and 40T lower speed by about 8%. But my keyboard works reliably, and my screen is clean. By the way, theoretical timing (6.144 MHz and new CPU T-states count) should be 56.6_s (0ws,80T) and 68.0_s (1ws,80T). Food for thought.

```
.*** XLR8A63/JCL ***
//.Procedure to patch an LSDOS 6.3 diskette for use
//. with an XLR8er. Drive to be patched
//. must be given on JCL command line.
//.If needed parameter (O=N) may be included,
//. same meaning as PATCH
//IF -d
//.DO XLR8A63 (D=d [,O=N])
//QUIT
//END
//IF O
.DO XLR8A63 (D=#d#,o=#o#)
//ELSE
.DO XLR8A63 (D=#d#)
//END
.Diskette in drive #d# is about to be patched
. Press ENTER to continue, BREAK to abort
//ALERT (7,7)
//IF O
patch sys0/sys.lsidos:#d# using xlsys0a/fix (o=#o#)
patch boot/sys.lsidos:#d# using xlboota/fix (o=n)
patch sys2/sys.lsidos:#d# using xlr8s2/fix (o=#o#)
patch sys12/sys.lsidos:#d# using xlr8s12/fix (o=#o#)
patch backup.utility:#d# using xlr8bu/fix (o=#o#)
//ELSE
patch sys0/sys.lsidos:#d# using xlsys0a/fix
patch boot/sys.lsidos:#d# using xlboota/fix (o=n)
patch sys2/sys.lsidos:#d# using xlr8s2/fix
patch sys12/sys.lsidos:#d# using xlr8s12/fix
patch backup.utility:#d# using xlr8bu/fix
//END
//ALERT 0,0,1,5,0,2
.eop
```

```
.XLSYS0A/FIX To use XLR8 special features (03-Sep-88)
.PATCH sys0/sys.lsidos USING xlsys0a/fix
.must also PATCH boot/sys.lsidos xlboota/fix (O=N)
d06,f0=57 ED 5C 7B C9;f06,f0=C5 57 AF 06 08
d06,f5=ED 29 38 7C 32 02 02 C9
f06,f5=87 CB 23 30 01 82 10 F8
d09,74=CD 77 08;f09,74=3E 66 EF
d0b,97=67 CD D6 06;f0b,97=F5 21 78 00
d0b,c2=E1 CD 0F 19;f0b,c2=F1 32 02 02
d0d,07=CD 94 21;f0d,07=00 00 00
d10,2e=3E 40 ED 39 32 ED 38 36 E6 FC F6 02 ED 39 36 AF
f10,2e=00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
d10,3e=ED 39 38 ED 39 39 3E 88 ED 39 3A C9
f10,3e=00 00 00 00 00 00 00 00 00 00 00 00
.eop
```

```
.XLBOOTA/FIX To use XLR8 special features (03-Sep-88)
.PATCH boot/sys.lsidos USING xlboota/fix (O=N)
.must also PATCH sys0/sys.lsidos USING xlsys0a/fix
d00,06=6D 10;f00,06=F4 0F
d04,c9=57 5D 6F ED 6C ED 5C 7B 5A 16 00 19 C9
f04,c9=C5 EB 4F 21 00 00 7D 06 08 29 17 CB 01
d04,d6=ED 28 38 E3 E5 AF ED 39 38 21 78 00 C9
f04,d6=30 03 19 CE 00 10 F5 4F 7D 6C 61 C1 C9
d06,77=79 C3 FD 0F 00;f06,77=E6 7F FE 03 D2
d09,9a=58 10;f09,9a=42 0D
d0b,df=2E 50 ED 6C 00 00 00 00
f0b,df=21 50 00 CD C9 06 65 6F
d0d,26=24;f0d,26=12
d0d,f4=18 FE 6C 10 04 58 4C 52 38 E6 7F FE 0B 30 2D D6
f0d,f4=00 00 00 00 00 00 00 00 00 00 00 00 6d b6 6d b6
d0e,04=03 30 0A AF ED 39 38 79 E6 7F C3 7E 08 E5 04 05
f0e,04=6d b6 6d b6 6d b6 6d b6 6d b6 6d b6 6d b6 6d b6
d0e,14=28 30 6F 2C 3E 80 07 2D 20 FC 21 7B 08 05 28 17
f0e,14=6d b6 6d b6 6d b6 6d b6 6d b6 6d b6 6d b6 6d b6
d0e,24=05 28 1A 05 28 0A 05 3A 02 02 E1 C8 3E 2B B7 C9
f0e,24=6d b6 6d b6 6d b6 6d b6 6d b6 6d b6 6d b6 6d b6
d0e,34=ED 34 20 0A B6 18 02 2F A6 77 AF E1 C9 A6 3E 08
f0e,34=6d b6 6d b6 6d b6 6d b6 6d b6 6d b6 6d b6 6d b6
d0e,44=E1 C9 21 05 80 39 E1 38 E3 07 07 07 C6 38 ED 39
f0e,44=6d b6 6d b6 6d b6 6d b6 6d b6 6d b6 6d b6 6d b6
d0e,54=38 C3 DB 08 ED 38 38 B7 CA 42 0D F5 AF ED 39 38
f0e,54=6d b6 6d b6 6d b6 6d b6 6d b6 6d b6 6d b6 6d b6
d0e,64=CD 42 0D E3 ED 21 38 E1 C9
f0e,64=6d b6 6d b6 6d b6 6d b6 6d
.eop
```

```
.XLR8S2/FIX - Patch to SYS2/SYS
.Apply via, PATCH SYS2/SYS.LSIDOS XLR8S2
D00,E4=12;F00,E4=09
D00,EE=40;F00,EE=20
D01,04=40;F01,04=20
.eop
```

```
.XLR8S12/FIX - Patch to SYS12/SYS
.Apply via, PATCH SYS12/SYS.LSIDOS XLR8S12
D03,7C=12;F03,7C=09
D03,86=40;F03,86=20
D03,9C=40;F03,9C=20
.eop
```

```
.XLR8BU/FIX - Patch to BACKUP/CMD
.Apply via, PATCH BACKUP.UTILITY XLR8BU
D17,22=12;F17,22=09
D17,28=40;F17,28=20
D17,3C=40;F17,3C=20
.eop
```

MISOSYS

APPLICATION SOFTWARE TO STRETCH YOUR TRS-80 MODEL 4

PRO-WAM™ Version 2

Window & Application Manager

Our applications turn your 128K Model 4 into a sophisticated business or personal machine rivaling the best of them. Because easily installed PRO-WAM comes with many useful and powerful menu-driven time savers and work organizers. PRO-WAM is accessed with a single keystroke; its **export** and **import** functions allow you to move data across windows between programs.

- Address **CARDS**, **LABELS**, and new **HEADINGS** for display and export
- Improved **BRINGUP** tickler file; new **PRINTING** and sorting
- Improved **CALENDAR** flags BRINGUP items visually on screen
- Ten 3 x 5 CARD files with **FORMS** and **FIELDS** using reverse video
- New virtual **PHRASE** access for export
- New **TODO** list manager with "who does it"
- Plus many other vital applications!

PRO-WAM [M-51-025] . . .$74.95 + $5S&H

MISOSYS has been supplying the TRS-80 community with professional quality software since 1978; that's over nine years of experience captured in a host of other software products ranging from language compilers and assemblers, fine crafted utilities, other application software, and operating systems. We also publish a magazine, *THE MISOSYS QUARTERLY*, which is available on a subscription basis for just $25 per year in the U.S. Call or write us for a catalog of our complete product line.

Now activate PRO-WAM from newly compiled LB release (hardware restrictions apply!)

M/C and VISA accepted
S&H: $5, $6 Canada, $15 Other

LB Data Manager

A flexible data manager

LB is easily used by anyone for managing their data. It's menu driven for ease of use; absolutely no programming needed. Requires a Model 4 with 128K or a hard drive. LB86™, an MS-DOS version is also available.

- Store up to 65534 records per data base
- Up to 1024 characters per record
- Up to 64 fields per record
- Nine field types for flexibility
- Select and sort on up to 8 fields
- Keep multiple indexes for accessing data
- 10 input/update screens per data base
- 10 printout formats per data base
- Extensive on-line help available

LB [L-50-510] . . . . . .$74.95 + $5S&H

MISOSYS, Inc.
P.O. Box 239
Sterling, VA 22170-0239
800-MISOSYS or 703-450-4181

TRS 80 MODEL 4 • MODEL 4D • MODEL 4P

Drive your Model 4 into the future with the XLR8er

(accelerator)

The XLR8er provides the following:

- ***Improved speed** — up to 8 MHz Z80 equivalent*
- ***Expanded ram** — 256KB additional high speed ram memory — optional*
- ***Expanded I/O** — optional*
- ***Software utilities** — TRSDOS, CP/M, or LDOS — one included with the XLR8er. Additional $15.00*
- ***Simple plug-in installation***
- ***Full one year warranty***

New pricing!
$175.00 with ØK RAM
$255.00 with 256K RAM
(add $5 S&H)

When ordering, please specify computer model (i.e. 26-1069, 26-1069A, 26-1070) and if HIRES graphics is installed. For 26-1069, note Revision # of motherboard.

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