Look at what is in this issue:

- IBM PC Line Drawing,
  by Roy Soltoff
- AMORT1: Amortization Table,
  by Andrew M. Kunz
- Yet ANother HiRes Graphics Format,
  by Hans de Wolf
- Reusing NEC Laser Cartridges,
  by Roy Soltoff
- BACKUP Basics,
  by Scott Toenniessen
- New price list for TMQ subscribers,
PRICE LIST for TMQ Subscribers - effective Jan 1, 1993

TRSCROSS (transfer to Mod III/4)

TRSCROSS (transfer to Mod III/4)

MSDOS Software (** Indicates Closeout)

Hardware (** Indicates Closeout)

Hardware (** Indicates Closeout)

The Fine Print

The Fine Print

MISOSYS, Inc.
P.O. Box 239, Sterling, VA 20167-0239
703-450-4181; Orders only: 800-MISOSYS
The MISOSYS Quarterly is a publication of MISOSYS, Inc., PO Box 239, Sterling, VA 20167-0239, 703-450-4181.

Unless otherwise specified, all material appearing in herein is Copyright 1992 by MISOSYS, Inc., all rights reserved.

THE MISOSYS QUARTERLY

subscription rate information

Each issue of TMQ has information on MISOSYS products, programs and utilities, patches, significant messages from our CompuServe forum, and articles on programming. Not only that, TMQ will keep you up to date with information, news, and announcements concerning our entire product line and related machine environments. Subscription cost varies by rate zone as follows:

A = $25; United States via 3rd class bulk mail
B = $30; Canada, Mexico, via 1st Class
C = $32; Colombia, Venezuela, Central America via AO Air
D = $35; South America, Europe, & North Africa via AO Air
E = $40; Asia, Australia, Africa, Middle East via AO Air

TMQ Toolbox

The MISOSYS Quarterly is published using the following facilities:

The hardware used to produce the “camera ready” copy consists of an AST Preium/386 computer (20 MHz) with 9 Megabytes of RAM, a Seagate ST-4096 80M HD, ST25140M, Expans! card; a CMS DJ10 tape backup, a NEC Multysync II monitor driven by a Video Seven VGA card, an AST TurboScan scanner (Microtek MS300), and a NEC LC-890 PostScript laser printer.

Text is developed, edited, spell-checked, and draft formatted using Microsoft WINWORD Version 1.1; Submissions on paper and letters are scanned and converted to text using ReadRight optical character recognition software by OCR Systems. Final page composition is developed using PageMaker 4.0 by Aldus.

The Blurb

Letters to MISOSYS

TMQ Appearance ....................................2
Points to Ponder ....................................2
Trade-in Policy ....................................5
In this issue ........................................5
TMQ Schedule .......................................5
MISOSYS Forum .....................................5
DISK NOTES 7.1 ....................................6
DOS Manuals ........................................6
MS-DOS Products ..................................6
Seagate ST351A/X IDE Drive ....................6
Power Supplies ......................................6
Address Change .....................................6
FAX Number .........................................6
LB Printing Hint ...................................7
Closeouts ...........................................7
Hardware Clearance ...............................7
Binders ..............................................7
Passages .............................................7

Inside TMQ

IBM PC Line Drawing ................................15
AMORTI .............................................20
Yet Another HiRes Graphics Format .............22
SMALARCH/CMD Small File Archiver ..........23
Reusing NEC Laser Cartridges ....................38
BACKUP Basics ....................................40

List of Advertisors

MISOSYS, Inc. IFC,42-54
Pacific Computer Exchange 22
TRSTimes magazine 14

List of Patches in this Issue

LB8230A/FIX For LB8/CMD 8
LB8230B/FIX For LB8/CMD 8
HDPACK3/FIX For HDPACK 10
TMQ Appearance

Just when you were ready to accept TMQ's appearance, here I've gone and changed it again. As subscription levels continue to shrink, and as I've gotten terribly dismayed at having to throw out all the extra copies of previous issues, I have moved total control of TMQ's publication in-house. TMQ is now being printed on our in-house copier. It is also being bound in our new GBC thermal binding machine. In this way, I can control the duplication of copies to exactly what is needed. There's no more waste of an excessive print run winding up in the landfill.

I also will no longer be making back issues of TMQ available; however, I will be making re-prints of articles available. There will soon be a complete index of TMQ content available. The index will list the number of pages included in each re-print; a standard fee per page will prevail. This index will be available in three forms:

- A bound printed copy - fee to be determined;
- An LB Database file set on floppy disk - fee to be determined;
- An LB Database file set available for download from our CompuServe Forum (as long as the forum remains available).

It is expected that the index covering TMQ Volumes I through VI will be available by late January 1993.

The Blurb by Roy Soltoff

Points to Ponder

SBT Corp., a software-consulting firm, has coined a new term which should add headaches to work-place businesses as well as home-based businesses using computers. I'm referring to the futz factor. Futzing relates to the wasted time spent by computer users playing or tinkering with their computers. According to SBT, futzing includes striving for perfection, playing with fonts, re-doing projects, and most likely, getting into computer games on business time.

But as we waste more time on computers, the hardware continues to add features. For instance, Sejin-America and Home Row announced a new 101-key PC keyboard which incorporates Home Row's J-Mouse. This non-traditional mouse is fabricated with resistive pressure sensors located beneath the "J" key (hence it's name). By depressing the key and pushing it in any of the four directions, software converts the pressure changes to mouse movements.

Early versions of the J-Mouse are already being used on Everex and Leading Edge note book computers.

Turning away from computers, here's something to cure the never-find-a-parking-spot parking blues. Mazda came up with a prototype one-passenger vehicle that folds into a suitcase; the suitcase is essentially the body of the vehicle. Power comes from a one-cylinder, 1.7 horsepower engine. Top speed is a scant 12 mph. However, options include such creature comforts as an AM/FM stereo cassette radio as well as a 3.3 inch TV. There's something to keep you occupied while you avoid being run over on the freeway. Then again, some of the roads here in the D.C. area run at less than 12 MPH during rush hour - an oxymoron if ever I saw one!

Talking about driving, there's been a lot of recent developments in the drive business - disk, that is. Toshiba introduced a 3.5" hard drive with 1.2 gigabytes of capacity. List priced at $2395, the MK-538FB includes a SCSI-2 controller and consumes only 10 watts.

IBM, on the other hand, which is struggling to stay at the top of the heap in the computer marketplace, introduced a 2 gigabyte 3.5" hard drive. It is also bundling two of these drives together into a 4 gigabyte frame to offer 8 gigs of storage. That's known as the Allicat package.

Coming down in size, Aura's 1.8" 126 megabyte 19 ms drive should be in production soon. These drives use the PCMCIA-standard interface, an interface currently seeing a lot of action.

Seagate also has a new series of 1.8" drives with 40, 65, and 80 megabyte capacities. The 65 meger consumes only 1.3 watts in read/write mode, 0.13 watts in standby, and a scant 0.09 watts in sleep mode.

Now why aren't hard drives usable on the TRS-80 coming down in price? That's because very few manufacturers are building any more MFM drives - the ones typically used on the older machines. I attended a presentation by Seagate Technology recently and was amazed as to their projections on the shift in drive size.
Right now, their shipments of 3.5" hard drives exceed the shipment of 5.25" drives. And 2.5" drives will overtake the 3.5" ones in a few years.

Seagate has also discontinued many of the older drives. The only MFM drives still in production at Seagate are the ST225 (20MB), the ST251 (40MB), and ST4096 (80MB). All 3.5" MFM drives are discontinued. Even the SCSI-1 and early IDE drives are out of production. Kalok discontinued production of their 3.5" 20MB drive quite some time ago; I used to use those in my MHD734 hard drive package. But it's more sensible these days to make larger capacity drives.

Of course the Seagate presentation was important to me for two items: (1) I got a look at the new ST351A/X IDE drive which has a combination 8-bit/16-bit jumper-selectable interface. I grabbed onto one of those recently and replaced the old Kalok in the family's 1000TL/2 with the snazzy 351A/X. Not only is there now twice the storage capacity, but the 351 is considerably faster and quieter than the Kalok 320. I'm now selling 351A/X drives (see announcement later on).

The second thing I found useful about the Seagate presentation was the ability to talk directly to a Seagate technician. Some of you who read between the lines may have noticed quite some time ago that I was working on a hard disk driver to use my host adaptor with SCSI drives. The MISOSYS host adaptor currently being sold was designed for the SASI/SCSI interface; but I use a supply of SCSI-1 to MFM hard disk controllers (the Xebec St1421 and Adaptec 4010) to utilize MFM drives. I had purchased a Seagate ST157N SCSI-I drive to aid in developing a driver. Unfortunately, I could never get the driver to work the drive properly.

Last Spring, I poured on the time to fully understand why the driver would not work. My research disclosed that the driver was able to operate the drive perfectly using a sector size of 512 bytes. But that sector size is not useful for TRS-80 users. Phone calls to Seagate's technical support, followed up by letters detailing the problem, led to no response. Until I nailed that tech! Meeting eyeball to eyeball solved the problem.

It appears that the Seagate SCSI Interface Manual left out one vital piece of information. In the Format Unit command, there is a data bit (FMT DATA) which if set, indicates that a list of defective areas on the disk will be passed to the controller; a zero indicates that the data-out phase of the command will not occur and no defect data will be supplied to the drive.

Such a list was not normally used on TRS-80 hard drives, and modern smart drives (those with imbedded controllers) maintain their own lists internally. In fact, another bit in the Format Unit command tells the controller to use only the original manufacture's list (internal list). What the Interface Manual failed to disclose was the requirement to set that bit if the sector size was being changed. Even if the list was void, the bit had to be set! That's so the controller would be able to re-calculate the new sector numbers corresponding to the existing list.

What was happening in my driver was the controller generating a continuous error the first time it accessed what it thought was a bad sector.

Now that the problem was solved, I went back to my prototype SCSI driver, made the change to set the FMT DATA bit in the format command, and it performed flawlessly. Now there's just a little cleaning up to do with the error handling and the SCSI driver is complete.

The end result is that my SCSI driver can be used with my T80 to SCSI host adaptor and any Seagate SCSI drive. This eliminates the need for a separate hard disk controller. Putting a hard drive internal is now less of a problem. With modern hard drives using less power than the old floppy drives, converting a floppy slot to a hard drive use will be easier - no separate controller to worry about for power consumption, heat generation, cabling, and mounting. I'll soon be popping that ST157N into a 4P. The only drawback is that now the ST157N has been discontinued. That was a 3.5" drive. The smallest capacity 3.5" SCSI drive Seagate now makes is a 248 MB SCSI-2 drive (still should work). They do have an 84 MB ST296N 5.25" drive. So I'll be looking around for compatible drives. If you're still looking to add a hard drive internally, talk to me.

Floppy drives haven't been taking a back seat to capacity inroads. Iomega Corp, makers of the Bernoulli Box, have introduced a variant on the Floptical drive licensed from Insite Peripherals, Inc. This 20 megabyte floppy drive has a 65 ms average seek rate - about the same as the venerable Seagate ST-225 hard drive. Iomega's lO20S 3.5" drive is read/write compatible with 1.44 MB and 720K floppy disks.

Improvements in the standard 3.5" floppy drive are coming from Epson. Their SMD-1100 drives run on 3.3 volts, the next standard for computer powering of laptops and notebooks. Epson also announced an SMD-800 series of dual 3.5"-5.25" floppy drives which incorporate both mechanisms into the same 5.25" half-height housing; the same electrical mechanism is used for both devices. A similar drive, the FD505, is available from Teac.

Other hardware advances include Dallas Semiconductor's EcoRAM. This item is a 256-bit read/write serial memory device in a 3-pin transistor package. Address, data, and control lines are all multiplexed on a single pin. A variant on this device is their DS2400 Silicon Serial Number - which is a read-only form of the SRAM. The SSN contains a one-of-a-kind number which can be used to positively identify electronic equipment.

Is a step back in time considered regressive? Not if you happen to be old enough to be a vacuum-tube audiophile. Transistors have replaced the use of vacuum tubes in virtually all forms of electronic
The Blurb

Eric Pritchard has now devised a Tube Emulator which he claims sounds like the real thing. Sample circuits were built up and presented to a group of professional musicians for ear evaluation. With that test successfully passed, Pritchard formed a company called Deja Vu Audio to market the Tube Emulator.

If your hearing needs improvement, how about your smelling? According to a brief article in EBN, Arrow Electronics is considering moving from their Melville New York headquarters to Dallas because of the discontent arising over an adjacent compost facility which has soured the local air quality.

One of the recent merchandising developments at Tandy is their requirement for bar coding products purchased for resale at their Computer City, Incredible Universe, and Radio Shack chains.

Speaking of Tandy, they just got some good press over their new Video Information System (VIS) multimedia player. The system includes a 25-MHz 386SX CPU, 2 MB memory, CD-ROM, SVGA graphics, stereo audio, and an infrared remote control. There’s no keyboard as of yet as this thing hooks up to your TV set. The VIS includes Tandy’s newly designed RAMDAC and VGA color-blending chips which make graphics look like broadcast video.

Now for a look at the horizon in memory storage, how’s 50 gigabytes in a 5 cm cube sound? All that, and 10 ns access time to any location in the cube. Those figures are the promise of research by Robert Birge, of the Syracuse University Center for Molecular Electronics, in photoreactive protein harvested from bacterium – halobacterium halobium, to be exact.

Peter Rentzepis, of the University of California at Irvine, is also researching the use of organic molecules in three-dimensional memory. His goal is 100,000 gigabytes in a peanut-sized, transparent cube. Does this sound like Superman’s crystal palace? Both of these systems employ laser-based optical reading and writing.

Speaking of lasers, in a previous issue of TMQ I mentioned a new blue-laser developed by Sony which was claimed to eventually triple the capacity of CD-based systems. One user asked me to comment on that. The increase in optical capacity is, in a way, no different than that which has occurred in the field of magnetics. The capacity of a recording medium is based in part on the size of the medium necessary to store one bit of information, and the ability of the read/write head to target an area of matching size. If, for instance, it took one square centimeter of area to record one bit, then a read/write head with a field area of two cm² would be incapable of resolving single bits. Tape recorders increased their storage density as tape mediums improved (finer particulate for the magnetic coating) and magnetic heads improved to reduce the distance between the tape and the head (allowing for smaller magnetic fields).

Optical recording with lasers takes the same approach. Recollect that the majority of lasers produce red light. Red, at the low end of the visible spectrum, has the longest wavelength of visible light, 610-750 nm. It is the actual length of the wave itself (or width if you prefer) which controls the area of medium being discerned. By using a laser light with a much smaller wavelength, the bit density can be increased. Blue has a wavelength of 450-500 nm. Thus the average areal density of blue is one half that of red. The problem was that blue lasers were very difficult to fabricate with sufficient intensity of light to be useful. Thus the advance had technological merit.

Another advance, though not a technological marvel, is the inclusion of a small CCD video camera as part of a pen-based electronic notebook being shown by Fujitsu. Not only does the unit have the capability of capturing handwritten input, it can also capture video images directly. Still black and white images are instantly captured and stored on its 2.5“ hard drive.

Now a look at the result of software bundling. In the past, users of LDOS and/or LS-DOS have asked for the inclusion of all sorts of applications within the DOS. Recognize that the DOS already includes a keystroke multiplier, minimal text editor, and a communications program. In the MS-DOS world, the same kinds of requests have been made - in fact, users have complained that the DOS was a poor product because it did not contain functions such as Superkey (keystroke multiplication), a shell processor, and a compression utility.

When DR-DOS came out, it was lauded for including file compression which MS-DOS 5 did not subsequently include (DOS 5 does include a shell and DOSKEY, a keystroke multiplier). Well now that MS-DOS 6 is to include bundled data compression, the stock of Stac Electronics, a leading supplier of compression software, dropped 75% in value.

Compression software companies are not the only ones who took a hit. When MS-DOS 5 was released with a shell facility, how many users of SoftLogic Solutions SoftwareCarousel gave up on it? I know mine is sitting on the shelf never to be upgraded again. Even though its switching performance was faster since it used RAM in contrast to MS-DOS 5’s disk-only task switcher, Carousel would have required an upgrade to the next version which would work with MS-DOS 5.

A DOS can’t include everything, lest it put independent add-on vendors out of business.

Some things get bundled with a DOS, and applications, that a user does not want. I am referring to computer viruses. These buggers have gotten quite pervasive over the past year or two as the ranks of computer crackers have swelled, and the extent of possible damage has increased. To date, most of the cures have been either
preventative (i.e. don't use any software disk except from a reliable source), or post operative (by using one of the many anti-virus programs which have come on the market. It has even been alleged that many viruses already exist which are designed to infect the next version of MS-DOS (6.0).

Now comes a hardware solution to the problem. Since viruses are primarily passed unbeknownst to the user by attaching themselves to executable programs thence to the DOS, having a means of protecting against fraudulent disk writing to regions of the disk containing programs could very well guard against virus infection.

According to Robert McCarroll of Western Digital, there exists about 1000 known computer viruses and they are growing in numbers to the tune of 50 per month. That's what prompted WD to develop the WD7855LP/LV system controller chip which works with the 386SX system management interrupt (SMI). The SMI, which is a highest level interrupt, is activated by WD’s controller on every disk write. Software in protected memory then checks to see if the write request is to an area of the disk previously protected by the user. If so, the operation is assumed to be a virus and the user - or a special virus-inoculation program - is notified.

And the other front for criminals - the software piracy front - has finally been attacked in Congress to provide bigger teeth on the punishment side. Commercial software piracy - where folks spring up and copy packages en masse for resale - has been elevated to a felony from a misdemeanor. It is said that this will now encourage federal prosecutors and other law enforcement agencies to pursue these cases of theft. The law defines commercial pirates as individuals who willfully copy software for commercial advantage or private financial gain. Prison terms of up to five years and fines of up to $250,000 can now be imposed on persons convicted of infringing at least ten copies of a copyrighted software program, or any combination of programs with a retail value greater than $2,500.

**Trade-in Policy**

The policy for trade-ins of an equivalent non-MISOSYS software product for a MISOSYS-published software product is to just send in an original Table of Contents page with the trade-in fee which is 50% of the price of our product. So for LB 2.3, trade in any other database product and you can purchase LB or LB-86 for $49.50 plus S&H. How's that for a deal? It doesn't matter for what system or operating environment your trade-in was designed for. This offer does not extend to products re-sold by MISOSYS or products on sale.

**In this issue...**

Last issue, I said that Volume seven would start our focus on the C language. I'll defer that an issue as many other articles were contributed or internally developed which I felt would have greater priority to TMQ readers. So look for the C coverage to begin next issue.

I did receive a big article covering an introduction to FORTH from Peter Knaggs; however, I need to do some work on it before publication in TMQ. I am still keeping open my offer for someone who wants to get involved with learning FORTH and writing about it in a series of articles - for either MS-DOS or TRS-80; MISOSYS will provide you with a free copy of either HartFORTH for the Model I/III or Model 4, or even HartFORTH-86.

From time to time I am told that there is not enough on BASIC in TMQ. Well, you'll find an amortization program in BASIC within this issue. Also, an algorithm for drawing lines with PC line-drawing characters is illustrated in a BASIC program.

For the assembly language buffs, Frank Slinkman provides another example of his expertise with a disk-quantity-optimized archival program for hard drive users. Along the theme of backing up hard drives, Scott Toenniessen provides techniques for using known tools in the hard drive backup process. Odds and ends fill out the issue.

**TMQ Schedule**

The MISOSYS Quarterly is mailed approximately every three months. Note that your mailing label usually has the expiration date of your subscription. For instance, those with "92/11" complete their subscription with this issue. The renewal fee to continue with another four issues is covered on page 1.

**MISOSYS Forum**

I sponsor a forum on CompuServe. You can reach some "experts" on TRS-80 and MS-DOS subjects by dialing in, then GO PCS49, or GOLDOS. This is probably the oldest forum still-surviving from the MicroNet days. If you want to see it continue, how about popping on for a chat or a question.

The forum contains many programs to download, as well as lively discussions which thread through the message system. You can direct a message to me at 70140,310. Post a message in private if you don't want it "broadcast".

I handle support via that facility. You can also submit an order either by a message...
saved as PRIVATE, or via EMAIL. MS-DOS users can even request an order for selected products be EMAILED as a ZIPed file for nearly instant service (manuals to be shipped separately).

**DISK NOTES 7.1**

Each issue of *The MISOSYS Quarterly* contains program listings, patch listings, and other references to files we have placed onto a disk. Where feasible, the text accompanying an article is also on DISK NOTES. DISK NOTES 7.1 corresponds to this issue of TMQ. The disk is formatted usually for TRS-80 LDOS/LS-DOS users at 40D1 (that’s 40 tracks, double density, one sided). MS-DOS users can request a 5.25” 360K disk. If you want to obtain the fixes and the listings, you may conveniently purchase a copy of DISK NOTES priced at $10 Plus S&H. The S&H charges are $2 for US, Canada, and Mexico, $3 elsewhere.

**MS-DOS Products**

MISOSYS is a reseller of products purchased from Ingram Micro; thus, we have access to a huge array of MS-DOS products. So if you are looking for some hardware or software to go with your MS-DOS system, why not get in touch with us for a quote. Call, write, or FAX.

I still have some Tadiran TL-5296 6V lithium batteries usable in most AT-class machines. Don’t wait for your battery to fail and lose your configuration data. A spare’s shelf life will probably out last your machine.

**Seagate ST351A/X IDE Drive**

One of the hot new drives on the market for low-end systems is Seagate’s new 42-Megabyte ST351A/X drive. This 28ms IDE drive features a MTBF of 150K hours. What I found very useful is the 351A/X is jumper selectable for either 8-bit (i.e. Tandy 1000TL/2) or true 16-bit bus systems. I replaced the Kaylok 320L and WDXT-GEN controller in my 1000TL/2 with this new 351A/X and the machine’s response is now fantastic. Besides a souped-up performance, the new Seagate drive is absolutely quiet! I can’t hear the thing.

So if you are looking to pop in a 40M IDE drive in your new low-end PC, consider this new Seagate. On hand now at only $195 + $7S&H (domestic U.S.)

**Power Supplies**

MISOSYS has a limited supply of replacement power supplies for Model III or 4 computers. The Astec AC8151-01 40-watt supply provides +5V @ 2.5A, +12V@2.0A, and -12V@0.1A. It’s size is 6.25”x4”x1.75”; mounting holes are 3.125”x4.75”. The Astec AC12310 68-watt supply provides +5V @ 7.3A, +12V@2.5A, and -12V@0.1A. It’s size is 7.69”x4.125”x2”; mounting holes are 3.75”x7.25”. This supply is a direct replacement for the Tandy Model 4 power supply (Tandy’s was based on the Astec design). The 40 watt supply is $40 and the 68 watt supply is $50. Surface S&H for either is $5. Currently in stock are seven of the 68-watt supplies and nine of the 40 watt supplies. When these are gone, we will not be obtaining more.

**Address Change**

The United States Postal Service has seen fit to change the ZIP code of our Post Office Box (mailing address) effective July 1, 1992. The new ZIP is 20167. Please update your records; I don’t want to lose any of your orders.

**FAX Number**

If you want to reach us by fax, try 703-450-4213.
LB Printing Hint

Here's a hint for LB 2.3.0 users looking for a way to remove blank lines from mailing labels. If you have received any recent mailings from us, you would be able to see the effect.

At MISOSYS, our general format for mailing labels is:

First-Name Last-Name
Company-Name
Street
City, State ZIP
Country

Most of our database records do not have a company-name entry; thus, our labels would have previously been printed with a blank line between the name row and the street row. But now there is an alternative.

With the inclusion of the Suppress Blank Lines feature of the Print parameters, you can eliminate the rows which have nothing printed on them. This can only be useful for one-across labels, however. Also, on mailing labels, make sure you set your print parameters for 6-line forms and set the I-Rec parameter in the defining of the screen. That's to ensure that all six lines of a typical 6-line mailing label will be used for each record (you do not want your printed image riding up the labels).

Closeouts

After evaluating the past few years sales of MISOSYS TRS-80 software products, the time has come to cease the manufacture of selected program packages. This issue includes a list of products which will no longer be available from MISOSYS after the Spring of 1993. In case anyone still wants to acquire some of these products, I am making this announcement to TMQ readers first. I will subsequently publicize this list to any takers. In order to make it easy, the price list in this issue reflects those products under clearance at prices reduced approximately 80% from normal. Available quantities of some products may be limited. Note that no warranties, returns, or support will be offered on these items.

Hardware Clearance

Over the years, MISOSYS has accumulated TRS-80 hardware and related equipment in excess of current needs. The following items are now classified as surplus and are available for sale to the first takers (shipping charges are additional):

- Tandy 2000 e/w color monitor, floor stand $150
- TRS-80 Model II (Video needs tweaking) $50
- TRS-80 Model III (working) $35
- TRS-80 Model III (video bad) $20
- Model 4, 128K (non-gate) $75
- TRS-80 4P (working) $75
- MAX-80 $75
- Amdek Video-300 Monitor $40
- BMC Monitor (for MAX or M1) $25
- (3) 5 Meg Radio Shack HD $75
- 10 Meg Radio Shack HD $100
- 12 Meg Secondary (bad drive) $50
- DW-II Daisywheel $50
- Printer III (no printhead) $25
- Radio Shack Modem II $15
- Lobo dual floppy drive box with two 80T1 drives $25
- CTR-41 Cassette (new, in box) $5
- CTR-80 Cassette (new, in box) $5

Binders

I was recently cleaning out the warehouse and came across a box of LSI 1" binders. These are the beige colored D-ring binders which were used with some LSI products years ago. First come first served at three for $10 + $5 S&H (continental U.S. only). Note some imperfect.

I also found, tucked away in the back room, a large box of old MISOSYS-imprinted 0.75" dark blue three-ring binders, and a handful of 1" MISOSYS binders. Both are imprinted with our old Alexandria address. The 1" binders are bundled three to a box, and the 0.75" are four to a box. $10 per box + $5 S&H.

And for the nostalgia buffs, I found a box of old Galactic Software Inventory Manager User manuals still in the Yellow three-ring binders. There's probably about 8-10 binders. Again, I can pack three to a box.

If you want a quantity of these binders, let me know. I can find a larger box to package them and save on shipping charges.

Passages

I close this Blurb with a sad note. Chuck Tesler recently informed me that Ron Malo, a former employee of Prosoft, passed away this past August. Ron was a key member of the early PC community, as well as the radio-controlled model airplane community. Those of you familiar with Prosoft's Alwrite word processor should well remember Ron.

The Blurb - 7 - The Blurb
LB 2.3.0 EDIT-FIND Bug

The following information was sent directly to LB 2.3.0 users. In case I missed someone, here’s the scoop.

It appears that there was a bug introduced at the recompilation of the LB8 module when version 2.3.0 was created. This bug renders the EDIT-FIND command inoperative when a FIND is performed on a database without an index file attached, or if one with a single key field is attached. This bug was located and supposedly corrected on the LB/OV8 file dated November 19, 1992. Unfortunately, after testing a hand patched module, a patch was applied to the master disk. The patch had the last “F” line mis-typed as a “D” line. Unfortunately, the PATCH command not only does not apply the corresponding D-line, it does not flag this as an error. As a result, the FIND command with the patch applied would most likely crash LB.

If you have a TRS-80 Model 4 version of LB with a file date earlier than 11/19/92 for the LB/OV8 file (disk 1), then please apply the following patch entitled LB8230A/FIX; the patch as listed here is correct. If your LB/OV8 is dated 11/19/92, then please apply LB8230B/FIX.

- LB8230A/FIX - Patch to LB/OV8 of release 2.3.0
  - Patch corrects bug in unsorted find command
  - Apply via PATCH LB/OV8 LB8230A
  D1E, 03=CD F7 70
  F1E, 03=22 CA 70
  D46, 58="Data not found "
  F46, 58="Search data not"
  D46, 67=00 22 CA 70 22 D4 70 C9
  F46, 67=20 66 6F 75 6E 64 20 00
  . Eop
  - LB8230B/FIX - Patch to

LB Printer codes

Fm MISOSYS, Inc: Release 2.3 of LB added an automatic print definition report generator. This print autogen module references a PRINTER DEFINITION file of printer codes to automatically construct a report definition for narrow and wide table reports, form reports, and 11 address label reports plus Rolodex and 3x5 card. The file uses the following control codes:

- codes to place printer in cpi’s 10, 12, 14, 15, and 17, where applicable;
- codes to enable/disable bold, where applicable;
- codes to enable/disable underline, where applicable;
- codes to enable/disable italic, where applicable.

The following codes have been submitted by our users since the release of the PRINTER.DEF file.

Fm David Huelsmann: The following ASCII codes are for an Epson L-1000 Action Printer.
I can get some of the samples in the manual to work using S but I always have to key in line numbers. When I tried the example on page 254 I get 0 compilation errors but nothing happens. When I try the sample on page 86 there are no compilation errors but the computer hangs up.

I am probably overlooking some small detail but would appreciate any help you can give me in this matter.

Fm MISOSYS, Inc: To begin with, Enhcomp programs do not have to have a BASIC line number on every line (see page 34). Only lines which are the target of GOTO's or GOSUB's, etc., require either a number or a label. If you are starting a program from scratch, then the easiest way is to invoke the S/CMD supervisor. It automatically loads the CED editor. From there (either CED invoked from S or directly from DOS), you can simply type "I", for input. Enhcomp won't automatically insert BASIC line numbers because most lines will not need that number. <BREAK> will cease the input mode. That is essentially the same as interpreter BASIC's AUTO command.

The sample program on page 86 will absolutely lock up your computer since you are using the Model 4 version and that program is noted to be for Model I or III (not the 4). The program as written pokes machine language code into memory - code which is Model I/III specific. What you see on page 254 is not a complete program but simply a program fragment - a program fragment, so to speak. It's sort of like a defined function in interpreter BASIC. If you wrote a BASIC program which consisted of nothing but a DEFFN, it wouldn't do anything, either!

Look at the short program on page 255 - it should work. So should the one on page 163 if you make sure that you either have a drive "A:" or change the OPEN to reference a valid drive.

Incidentally, your reference to your existing BASIC appears to be referencing the BASIC Interpreter provided with the DOS, not a compiler. There is a major difference in the two. Interpreters need to look at each source statement each time a statement is reached and convert the logic to interface with machine language functions. Compilers do that once and convert the source program to machine language.

HDPACK Queries

Fm JFS: Since I use my HDDE utility to create extra directories for my 40 MB drive, DIRCHECK, of course, reports many "granules allocated but not used". This is due to the fact that the GAT must record all space used on all directories.

Will this prevent HDPACK from working properly, or will it just avoid using the granules DIRCHECK thinks are improperly allocated (as I would think it should)?

Fm MISOSYS, Inc: Any granule showing up in the GAT as allocated but unused by a directory entry will be locked in place and both unused and unmoved. Your presentation with HDDE would appear no different than any other "bad" granule locked out by a hard drive formatter as unusable - but the only way such a formatter can lock it out is to either allocate it in the GAT, or construct a file entry allocating it. You should be safe.
software. I'm referring to HDPACK, I hadn't realized how "flaky" a hard-drive can get after using it even for a week. I knew reading and writing to the hard-media will fragment a file after several access's, and now I can really appreciate on a weekly basis, to optimize it and know all will be 'smooth' on a regular basis. Fantastic, and a job well done!

I used DISKOPT, by D. Goben, and since then have stuck it in the back of the disk box. I've run into inconsistencies with it, and seems a clash every now and then would occur, I didn't like it after I found out that it caused some files to "grow". I noticed a 19.5K/HR file would turn-out to be 24K when I finished optimizing, I don't know what happens, but it must me re-sizing the granules or something.

At any rate, I have one comment on your HDPACK also. I am using LS-DOS 6.3.1, and have created a JCL file to optimize my hard drive; it's very simple:

```
HDPACK :0
HDPACK :1
HDPACK :2
HDPACK :3
```

One thing I noticed is now that I've cleared partition 1, (for a blank dummy drive), with no files on it, when the JCL file gets to the "empty" partition, HDPACK hangs up the system and I have to re-boot. Is this the JCL doing this or HDPACK? Do you have a fix it if so? Please drop me a line if you can Roy. Thank You and have a nice Christmas out east there. I'll be watching for your next new piece of software!

Fm MISOSYS, Inc: Well it turns out that the problem you refer to is due to HDPACK trying to clean up after the last file was restructured. Of course, with an empty directory, there is no last file (HDPACK doesn't test for that). The HDPACK3/FIX will correct for that event.

```
. HDPACK3/FIX - 12/15/92
 - Patch to HDPACK
 - Corrects hang on attempt to pack a disk with no files
 - Apply via, PATCH

HDPACK HDPACK3
D06, C9=33
F06, C9=32
D08, C3=AD 2C
F08, C3=D6 26
D08, D9=2A 5A 27 7C B5 C8
C3 D6 26
F08, D9=72 7B 27 7C B5 C8
C3 D6 26
24 24 24
24 24 24
Eop
```

LDOS Version?

```
Fm MB: What is the latest version of LDOS for a Model I? I'm currently on 5.1.4. I've seen references to 5.3 (?) but that also seems to be mainly for a Model III.

By the way, is it at all possible to load the RDUBL driver in LOW memory somewhere? My word processor ignores the memory flag and grabs it all (COPYART), so I have been running it with SD all these years. It almost worked with TRS-DOS 2.7/8, with just a bit of general unreliability, so it must be possible to do something. Ideas?

Fm MISOSYS, Inc: Latest version of LDOS for the Model I is 5.3.1 (that was a relatively recent upgrade). You cannot run the double-density driver in low memory as there is not enough space available. Why not dig into COPYART to fix it to honor HIGH$?
```

OPREG bit-3 Confusion

```
Fm MISOSYS, Inc : There appears to be some confusion concerning the software handling of reverse video in the Model 4 DOS. The TRS-80 Model 4 is not capable of reverse video of the 256-character set - only the lower 128 characters can be displayed in reverse. The Model 4 uses character bit-8 to indicate a character should be displayed in reverse video. Once reverse video is activated in hardware, any character stored in the video RAM which has bit-7 set (values 128-255) is displayed in reverse. Characters with values of 0-127 are displayed in normal video. Bit-3 of the OPREG (78H) designates hardware reverse video. Therefore, when you activate reverse video with CHR$(16), the video driver enables the hardware by setting bit-3, and "turns-on" the driver's bit-8 setting routine. Any character sent to *DO from then on will have bit-8 set by the driver - even if it is a character already with a value above 127; the hardware will display it as an ASCII character in reverse video since the hardware is enabled. The DOS manual documents CHR$(17) as "Set high bit routine off". That's exactly what it does. Once the driver receives a CHR$(17), it ceases further setting of bit-7 for each character subsequently sent. But since the hardware is still reverse video active, any reversed characters still on the screen are left in reverse video.

Now, when you want to stop displaying characters in reverse, you send CHR$(17). However, since the previous characters (in reverse) still appear on the screen, resetting bit-3 of OPREG will display those characters as their 8-bit component. Now what were ASCII characters in reverse, will automatically appear as special or alternate characters. That's not what you want. So CHR$(17) does not reset the hardware; it only turns off the driver's bit-8 setting routine so subsequent
- and the second one was too long to fit on
a single line - until I left off the second
scr17.

After reading your write-up of PRO-NTO/
PROWAM, I can see why you are not
familiar with the Tandy programs of
SuperScripsit or Deskmate. Now if I
switched to PROWAM instead of using
Deskmate:

What would I need to go along with the
plain Jane PROWAM to duplicate Tandy's
Deskmate? (I have SU4 and Doubleduty.)
Does PROWAM2 include MisterEd? Is the
Text Editor useful as a simple word
processor as is Deskmate? Would it ac-
cept ASCII files without modification?
(Deskmate only requires the filename to
have the extension /DOC for it to be listed
on the menu.) I have many thousands of
words stored in ASCII Deskmate files.
Would it allow transfer of ASCII files
from one disk to another without being
in the word processor program? Would it
allow importing an ASCII file to the file
under work, from another file on that disk or on
another active disk? Will it allow a speci-
fied paragraph to be exported to another
file on the same or a different disk, and
have it automatically appended to that
file?

Will I have to have 128K on my Model 4
to run it? (one of my 4's does the others
don't.) Easy enough to fix.

How to obtain PRCTL? How much? (As
on p.35 VI.iv). Same for CTL255?

And on other subjects: Why can't LDOS
5.3 do a CAT? It does a DIR in algebra-
ical form and it does a four column DIR,
un-alpha when asked for DIR under
SuperScripsit (Mad3), so the routine must
be there somewhere - perhaps only in the
SS? It sometimes makes for easier print-
out to be taped to the disks. The DIR :0(P)
Printout is longer than the CAT :0(Pr).

Fm MISOSYS, Inc: The format I use to
publish patches in TMQ is as a /FIX file.

This means that you should type the lines
into a file, save it under the name speci-
fied, and apply it to the program as speci-
fied. You really only need to type in the
"D" and "F" lines, however, if you care to
preserve the file for future reference, it is
a good idea to type in the comment lines as
well - or obtain the corresponding
DISK NOTES to save you the typing. For
instance, that patch should have been
typed in and saved under the name
"SCR17/FIX". Then it would be applied
by the command printed on patch line 4.

I think you missed the point of all the
information I printed in the previous issue
concerning PRO-WAM. DESKMATE
and PRO-WAM are two completely dif-
terent types of packages. DESKMATE
attempts to be an operating environment
with easily-invoked applications: i.e. word
processor, spreadsheet, communications,
and so on. This is somewhat similar to the
T/MAKER product which was sold for many
years - and still should be available by T/
Maker Company (now T/Research, I
believe). The problem with DESKMATE is
that there is not sufficient memory in the
TRS-80 to implement fully-functional
applications.

PRO-WAM, on the other hand, is similar
to Borland's SIDEKICK - which is a
memory-resident pop-up windowing appli-
cation program. None of the applica-
tions in SIDEKICK - nor in PRO-WAM -
pretend to be fully-functional applica-
tions. Their claim to fame is their ability
to be available for use when you are
running another application. This gives
your computer the task-switching capa-
bility. I thought that message was quite
clear.

Therefore, it makes no sense whatsoever
to compare the applications in PRO-WAM
- or the package as an entire package - with
the DESKMATE application. However, that
aside, I can make some comments to
increase your awareness of PRO-WAM's
applications.

MisterEd is a program package separate
from PRO-WAM. From time to time
MISOSYS has bundled the two together
for a promotion. This combination was
available and advertised at a special price
in the last issue of TMQ.

As far as TED is concerned, the PRO-
WAM article said that, "the LS-DOS 6.3
TED/CMD editor was derived from TED/APP". Since you have TED/CMD with
your LS-DOS, you know exactly what its
capabilities are. TED/APP omits the print
command since the application is ex-
remely tightly written and has no memory
available for additional commands. Thus,
TED/APP accepts ASCII files (in fact
only ASCII files), uses a default extension
of "TXT", allows transfer of files be-
because being a PRO-WAM application,
you can pop up the PRO-WAM library
executive to perform DOS library com-
mands, And just as TED/CMD operates,
TED/APP allows files to be concatenated.

PRCTL as well as CTL255 are both part of
the Wammes Application Disk. The price
for this disk was published on page 5 of
the TMQ you referenced (see the PRO-WAM
Special in column three). The disk was
(and is) $10 + $2 S&H.

Now, why can't LDOS 5.3 do a CAT?
Well it can! The CAT command is simply
an automatic invocation of DIR with the
(A=N) parameter. Fortunately for LS-
DOS, the command interpreter overlay
(SYS1/SYS) has room sufficient for a
larger command table. It's overlay region
extends from 1E00H to 23FFH - a full
1.5K. In LDOS, the overlay region
extends from 4E00H to 51FFH - only 1K.
Thus DOS overlays in release 6 have more
memory space available to code function-
ality in the overlays.

If you have access to a Programmer's
Manual, you will note that the DOS in-
cludes a service call named @DODIR.
This is available under both LDOS and
LS-DOS. It's function is to perform a
mini-directory either to the display or a
user buffer. That's what SuperScripsit is
using to display the directory.

Finally, on the subject of CAT, issue Liii
@PUTs to *DO won’t have bit-8 set.

The only way to ensure that no characters remain on the screen which were displayed in reverse is via a CLEAR-SCREEN, i.e. HOME (28) followed by Erase to end of display (E2EOD = 31). It works that way by design!

**LS-DOS 6.3.1 on 6000HD**

Fm DY: I’m curious about the Model 2/12/16 version of LS-DOS 6.3.1. Will it work on a 6000HD? And what about the Hard Drive? Is it usable under 6.3.1 via your RSHARD drivers or some other way? How Model 4 compatible is it?

Fm MISOSYS, Inc: Yes, and No/Yes. The LS-DOS 6.3.1 which is for the Model II will run on the 6000HD (I have one right next to me). A driver is available for the II/12/16/6000HD but it is TRSHD2/DCT and TRSFORM2/CMD. The seems to not work on some Model 12 computers; however, the problem has never been isolated.

**Converting a Database to Clipper**

Fm RD: The task that I have to perform is to move a single database file (written in TRSDOS BASIC) onto a MS-DOS format and finally into a DBASE DBF (Clipper) file structure. Any other hints you may have would be gratefully received.

Fm MISOSYS, Inc: Okay, one thing is our TRSCROSS file conversion utility for MS-DOS. It reads TRS80 Model III and 4 files directly on your PC.

If your Model 4 database in BASIC is using a fixed record length file (i.e. fielded records), then an additional solution is to use the LBCONV database conversion utility which is bundled with our LB86 Data Manager. It can convert a fixed record length fielded file to LB format - then LB format can be converted (using LBCONV again) to dBASE-III format. LB86 is $99 + S&H. A bit much for just the LBCONV utility. You may find a PC-NATUGU member in the UK with a copy of LB86 who may be able to do that conversion once the file(s) is/are moved to MS-DOS.

**PRO-WAM Use**

Fm JPJ: PRO-WAM was received and installed. I use five Model 4’s, only two with hard disks. Due to lack of floppy space, I made a PRO-WAM application disk that can then be removed (since /APL’s are in bank memory for that application). I first used a /JCL file to install PRO-WAM and copy the proper WAMx/APL to WAM0/APL (/APL number on the command line would be helpful). But I had trouble remembering the defaults. So I created PROWAMx/CM files from the original PROWAM/CMD, but with the proper defaults for each application (x now runs from 1 to 5). I then used FED/ APL to change WAM0/APL to WAMx/APL on each PROWAMx/CMD file (Record 19). A /JCL (copy attached) installs the proper application, then I can remove the disk and insert the data disk. I thought that others may want to try this.

Fm RJS: Thanks for the patches to Superscript which you published in Volume VI.iv page 9. They worked fine - after a little experimenting. I forgot that patches needed parentheses around them.
of The MISOSYS Quarterly published a short article on extracting the DIR library module from LDS’ SY6/SYS module, patching it for “A=N”, and naming the resulting file CAT/CMD. It’s also a simple matter for a modest programmer to code a CAT for LDS which uses the @DODIR service call. In fact, based on my discussion of the Superscripsit patch to SCR17/CTL which was in the last issue, most of the code is right there. All you need to add is a parsing of the command line to get the drive number to CAT. TRSTimes of Nov/Dec 1991 had such a program.

### International Keyboards

Fm LR: I purchased the French version of LS-DOS 6.3.1 some time ago, your Invoice 92-10404 dated 05/08/92. So far I have been unable to find the letter M, which makes it rather difficult to type in commands such as MEMORY, SYSTEM, etc! Typing an upper case “M” gives me a question mark, the lower case “m” gives me a comma. I’ve tried all possible combination of keys with no success, although did manage to get a lower case “m” on my printer using CONTROL; but that doesn’t give me an “m” on the screen.

At three score years and ten my old brain isn’t what it used to be, can you help me out?

Fm MISOSYS, Inc: The international versions of LS-DOS, specifically the French/Belgium (6.3.1F) and Deutsch (6.3.1D) versions, require the use of a TRS-80 equipped with the respective keyboard. The international DOS versions are not conversant in the foreign language; each merely addresses the specific keyboard and video. The printer driver is also specific. Everything else about the DOS (except for the day-month-year sequence of date entry) is identical to the domestic variety DOS.

In your case, you cannot enter the letter “M” because your keyboard does not have that key! Neither can you enter the following characters: { }, %, *, @, ^, “, and ".

The sketch shows a keyboard matrix for the French/Belgium keyboard. The area in a dotted box reflects the keys which are only in place on the international keyboards. The Deutsch keyboard would have those keys installed with slightly different keycaps. In your case, “M” is one of those keys. Note also that if you compare the keycaps with the English keyboard, you will note that there are differences from the regular key positions as well. I have placed the English alphabetic keys which are different from the international keyboards just outside of the keycap image; there are also differences in some of the special keys which I have not illustrated. You may as well resort to using the English version of LS-DOS unless you can locate a French/Belgium keyboard.

If I have to slow down, I might as well use TAPE100. It may be slow, but it is absolutely reliable. I very soon became weary of M100->tape->M4. Besides, the tape would get dents on it from the parked capstan and cause drop-outs.

So I tried to use a little pocket radio as an amplifier to allow direct M100 -> M4 transfer. It chugs along at about 1500 BAUD, if the manual can be trusted. That is a lot better than the MODEM speed of 300. It is very quick to set up and it works super reliable, and it works just as well M4 -> M100. Naturally, it won’t work for the 4P.
For anyone interested, few though there may be, the equipment is simple for anyone handy with a soldering iron. Just take a little battery powered, pocket radio and add a mini-jack to the input of the audio section. (Find the volume control. Ignore the two leads to the power switch. See which of the other three leads goes to ground. It will be the one on the "bottom" end of the potentiometer. Open the other end and insert the jack. If it is done correctly, the radio should still work as a radio and also work as a little utility amplifier. Test it using a cheap crystal type microphone or crystal type earphone.)

Now make up two cables using mini-plugs and DIN plugs for input and output. Label them IN and OUT according to the TECH manuals. (Watch those DIN plugs the numbering is not smooth. Enough to say, the ground is the one opposite the locating tab and the IN is the pin next to it on one side and the OUT is the next pin to it on the other side. Just make up a cable using ground and one of the adjacent pins going to a mini-jack and another cable using the other adjacent pin going to it's mini-jack.)

Set the volume control to about moderate loud from the speaker. (annoyingly loud.) Plug the input cable into the output of the radio, call up TAPE100 and watch for the symbols. Reverse the cables to the computers and the plugs in the amplifier for transfer in the other direction. That's all there is to it, and it works every time - and transfers reliably.

I tried to get Mark Reed to modify his program Load 100/CM to work both directions, but he has trouble with his CASS port and begged off. I suspect his problem is with the tape system, not with the CASS port. Every time I have tried to use tape, I have eventually had trouble. The direct "cable, amplifier, cable" method works super. If we could get a program for the Model III that would work both ways in LDOS and also one in TRSDOS, then we would be able to use the M100 to transfer anything to anything in the TRS80 realm. Besides allowing students and researchers to have a super-low budget tool for notes and full word processing. The inexpensive Model 100 does great for portable work, but the Model III or Model 4 does much better managing files on disk and exchanging files with others by disk. (The Model III's are very cheap, now, and still run quite reliably, and thanks to you, Roy, they still run efficiently. Roy, if you decide to distribute on an annual basis, I'm interested.)

Thanks again for helping to keep us going for as long as you have. I try to do as little program development as I can. It interferes with my other writing and life is too short to do all the things that I want to do, as it is.
During the development of version 2.3 of the LB Database Manager, I wanted to provide the operator with the capability of drawing lines and boxes in the view screen. To do this, two hurdles had to be overcome. The first hurdle was a means to support a 256-byte character set; this was related to the internal design of LB which I'll touch on briefly here. The second hurdle was the development of an algorithm for easily drawing lines and boxes.

LB was originally developed jointly for the TRS-80 and MS-DOS operating environments and released back in 1985 by Logical Systems. The beginning stages of planning and development were probably done in 1983. LSI originally was working on developing an MS-DOS-only relational database system which internally was called "Big Brother" (BB). That proposed product was noted in their LSI Journal of April 1984. BB was being developed to LSI specifications by an out-of-house development team. Unfortunately, the team fumbled the ball and could not produce even a portion of the specs.

LSI scaled back the specifications to a flat-file database manager and brought the efforts in-house. My supposition is that the term "Little Brother" stemmed from the junior-sized version of the larger effort. They also ported the package under development to the TRS-80 Model 4 environment. Since the design goal was to achieve identical operating characteristics across both MS-DOS and TRS-80 environments, certain compromises had to be made.

The PC video environment allows character attributes, such as reverse video and blinking, to be distributed on a character-by-character basis throughout the entire video screen. The PC uses two eight-bit memory cells for each video display character position; one stores the character value while the other stores the attributes. The TRS-80 Model 4 video hardware uses a single eight-bit memory cell per character position to store only a character value. Also, it does not inherently support blinking video; where used, that must be achieved through continuous software updating of the screen by displaying blank strings alternately with the characters to be blinking. Reverse video is available on a character-by-character basis; however, reverse video is designated by activating the eighth bit of a character value (i.e. by adding 128 to the value) along with a bit change in a CPU port to activate reverse video hardware. Since the eighth bit is used to distribute reverse video on a character basis, this then restricts reverse video to the lower 128 characters; graphics and special characters cannot be used simultaneously with reverse video.

The LB Data Manager view screen - as does the entire presentation of LB's screens - uses reverse video for highlighting. Reverse video can also be used by an LB user to highlight the view screen when it is created or modified. Since box and line drawing characters have character values above 127D, they would be unusable on a TRS-80. In the quest to make both versions equivalent, my supposition is that LSI either LSI did not consider the use of line drawing characters or decided it could not implement their use.
The LB view screen is created by a user by means of a drawing technique using cursor movement keys and character entry. Fields may be placed on the screen anywhere - the choice is entirely at the discretion of the user. The screen image is saved in a data file using two character arrays. The screen array is for storing the actual screen characters (note that a string of periods is actually stored in the screen array for each position of a field). The eighth bit of the screen array cell is used to designate that cell position is to be displayed in reverse video if the bit is set. The attribute array contains field numbers in any array cell where the corresponding screen position is a field. LB supports a maximum of 64 fields numbered 0-63. To differentiate field 0 from a non-field position, each field number has a mask value of 128 added to it. This scheme uses bits 0-6 to store a field number with the high-order bit (bit-7) set to indicate a field.

On the surface, it would appear that the view screen storage arrays provided no further room for adding any other attribute, or of expanding the displayable character set to include line drawing characters for the PC. That was the first challenge to overcome; I did not want to alter the screen array which would require a conversion procedure to upgrade existing screen files and an inability to be downward compatible.

In solving the problem, I realized that even though the attribute array used all eight bits, they were used only when a field cell was specified. Since line drawing has to be restricted from passing through a field displayed on the screen, fields and line drawing were mutually exclusive functions. If I limited line drawing bit uses to bits 0-6 and keep bit 7 RESET, then the attribute array could share its use with both field specifications and line drawing specifications. Also, since the TRS-80 could not simultaneously display graphical line drawing characters along with reverse video - which was always present - I had to either not support line drawing on the TRS-80 or convert...
line drawing symbols to some form of ASCII character counterpart. The latter method was chosen. One additional point was to keep the view screen file identical across implementations for ease of porting data file sets across implementations. This was done by storing the IBM PC line draw graphic character and having the TRS-80 version perform the code conversion during the screen display.

The technique used to differentiate reverse video character codes from graphic codes is to use bit-2 of the attribute array SET with bit-7 RESET to designate that the corresponding screen array position is to be used as is. The screen cell can store a graphics character with bit-7 SET. Additionally, reverse video of line drawing characters is designated by setting bit-3 of the corresponding attribute cell.

Now it certainly would have been simpler and more straightforward to revise the screen arrays' use to store strictly character values using all 256 values and simultaneously revise the attribute array use to designate either fields or reverse video attributes; however, to do that would have again created a problem in backward compatibility as well as requiring a screen conversion utility to upgrade from a previous version. I chose to keep the upward and downward compatibility. Besides, since the view screen background consists simply of static strings, the additional time required for array processing is unnoticeable compared to the time to open, read, and process the view screen file. With that first challenge out of the way, I moved to the second challenge - what algorithm could be developed to use in forming line connections?

The PC supports what could be called two sets of line drawing characters - including junctions where lines intersect: one set is made up of single lines; the other is made up of dual lines. There also exists a set of joining characters which establish the junction points where single lines meet dual lines. In the implementation of line drawing in LB, I chose to support only the dual-line line drawing characters. This, of course, is dashed to make the dual lines distinguishable.
course, made the job easier. But it took some time to discover an simple algorithm which would calculate the proper joining character where an intersection occurred.

Microsoft WORD provides a line drawing mode in which you can simply move the cursor around the screen and a line would be drawn in its trail. This is not unlike many of the block graphics programs TRS-80 users have seen in the past, except that WORD’s line drawing used the PC line drawing characters. Figure I displays a facsimile of these dual-line PC characters along with their values, in decimal. I was always fascinated how WORD always displayed the correct joining character when the cursor was moved through an existing line (which created a cross junction) or when the cursor was moved at rightangles (perpendicular) to a line (which created a T-junction). When I first sat down to consider a technique, it was quite a puzzle.

My first attack was to explore the character values themselves in hexadecimal, octal, and binary. For instance, an examination of the Z80 operation codes in octal will reveal a great deal about their underlying operation as groups of codes. But that didn’t provide any insight in the case of line drawing characters.

I next attempted to study the codes from a state diagram. Such a diagram reflects the change in state of a component to a given output caused by specific inputs. But the state changes were utterly complex. I naturally assumed there had to be an easier way.

I then stumbled onto the idea of decomposing each line draw character into subcomponents. A close inspection of every character in Table I will reveal that each character is composed of, at most, four pairs of components: a top vertical; a bottom vertical; a left horizontal; and a right horizontal. I could thus create a table of component values which would then represent equivalent characters. Value selection was arbitrary. I chose to use binary values of 8, 4, 2, and 1 to represent the top vertical (TV), bottom vertical (BV), left horizontal (LH), and right horizontal (RH) components respectively. For instance, a plain horizontal line has both an LH and RH component; a top left corner has both an RH and BV component; and a leftfacing T-junction has three components: TV, BV, and LH. With the line drawing characters converted to a table of subcomponent values, it is quite easy to perform a simple lookup of the character value to a given code.

The next piece needed to put together the line drawing algorithm was a method to create a code value as the cursor was moved about the screen. Careful thought produced the following idea. The cursor is always moved by means of an ARROW key which denoted the direction of movement. Thus, if the direction of movement is UP, the current character cell should add a top vertical component. Similarly, if the direction of movement is RIGHT, the cell should add a right horizontal component. It is easy to see then, that moving DOWN adds a bottom vertical component and moving LEFT adds a left horizontal component. But what should these components be added to? The current cell value? No, it turns out.

You need to examine the adjacent cells for components closest to the cell in question. As an example, let’s consider moving UP. A cell value is built then, by adding 2 (LH code) if the cell immediately left has an RH component, plus a 1 (RH code) if the cell immediately right has an LII component, plus a 4 (BY code) if the cell immediately below has a TV component, plus an 8 (TV code) if the cell immediately above has a BY component. If the cell immediately left, for instance, has no RB component, then you know there is no line coming from the left.

As it turns out, there is also no problem with checking for and including a component value from the adjacent cell in the direction of movement. This is because
the component value will always be added because the direction of movement requires the corresponding component value. However, considering that adjacent cell allows us to consider all four adjacent cells as a common function prior to any function needed to be executed for a specific direction.

I chose to use the screen borders as an enclosing surface; thus, if the cursor is currently positioned at the border, the cursor cannot be moved into the border to wrap around to the opposite side of the screen. Neither does the imaginary cell on the other side of the border add any component.

Another question to consider is the simultaneous display of line drawing and ASCII characters. I chose to implement in LB the priority use of line drawing. This means that a line being drawn through a character string will replace any ASCII characters with line drawing characters. Also, ASCII characters provide no line drawing components. This is the same concept in TRS-80 block graphics when displaying both graphics characters and ASCII text characters on the same screen.

The completed algorithm simply examines the four cells adjacent to the current one, temporarily accumulating the proper codes for the four adjacent components if they are present: BV for above; LH for right; TV for below; and RH for left. It does this by sequentially searching a character to code conversion table using each of the four adjacent cells in turn. Any character not in the table provides a zero result. The algorithm then adds the proper component based on the direction of movement. By constructing a code to character conversion table with all possible sixteen binary values, the line draw character can be extracted simply by using the code as an index value. Actually, the table has seventeen values since a zero value should imply no component (i.e. display a SPACE). Note that since initial movement would accumulate no components from adjacent cells, it is important to have codes of only a single component produce a character of two components. For example, an initial movement to the right from a blank screen accumulates only an RH component. But no character has that kind of appearance; the proper character is a horizontal line. So a code of 1, as does a code of 2, is translated to a character value of 205 just as a code of 3 which has both an LH and RH component. Table II presents the code to character translation.

Table II

<table>
<thead>
<tr>
<th>TV</th>
<th>BV</th>
<th>LH</th>
<th>RH Character</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>32</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>205</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>205</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>205</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>186</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>201</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>187</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>203</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>186</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>200</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>188</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>202</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>186</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>204</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>206</td>
</tr>
</tbody>
</table>

Since this algorithm was developed specifically for the PC line drawing character, I chose to illustrate its use in an MS-DOS program. The line draw program is written using Microsoft Quick BASIC.

Of all the Microsoft MS-DOS BASICS, none provides a way to read extended key codes through the INKEY$ function; thus, a program somewhat compatible across TRS-80 and MS-DOS is an impossibility to program. MS-DOS BASIC supposedly allows you to sense (trap) any key code through use of the ON KEY GOSUB statement. The linedraw program uses that facility. On the other hand, I could not get it to work correctly using user KEY values (15-20). The program illustrates the algorithm using <F2> to toggle between draw mode and erase mode, and <F10> to terminate.

The program first dimensions an array used to store the screen character, and two additional arrays for the conversion tables. A multi-line function, ABTVAL, is defined which is used to obtain the sub-component, if any, of an adjacent cell. The conversion tables are then filled with the values read from DATA statements.

The LOCATE statement positions the cursor to the northwest corner and turns on the cursor. The program allows for both drawing and erasing the line, functions which are toggled with an <F2> key depression; drawmode is initially turned on. Note, by the way, that LOCATE uses screen position values which are 1-based (i.e. col 1, 2, ..., 80) whereas the program uses a zero-based value for its ROW and COL variables; the cursor re-positioning statement makes the adjustment by adding one to the row and col values.

The video array is next initialized to SPACE characters to match the clearing of the actual screen.

The ON KEY(2) statement sets up BASIC's trapping of <F2> for toggling draw/erase modes. The ONKEY(10) traps the <F10> key for use as an exit from the line draw program. The KEY(n) statements activate trapping of the cursor movement keys. A drawback of this key utilization is
that QB traps only the cursor characters on the keypad - not on an auxiliary cursor pad as found on 101-key keyboards.

The ON KEY statements will invoke the subroutines only when the corresponding key is pressed. The line numbered 100 will be continuously executed until a trapped key action is detected. <F2> stops this program.

Each of the four direction movement actions call subroutine 150 first. This is the part of the algorithm which examines the four adjacent cells and accumulates a temporary code value based on the adjacent cells participating components. Each direction of movement code block then adds the proper component and calculates the new cursor position - all based on a valid movement. Finally, they each call a common subroutine to index the correct character for the cell and display the character. The cursor position is then updated.

Note that the direction movement DOWN introduces a spurious repetition of the character from which the cursor was just moved - at least it does on my system. This does not appear to be a bug in the program. For some reason, moving down seems to draw two characters.

The same concept can be applied to single-line line drawing characters. In fact, it should be possible to handle a mixture of both single- and dual-line characters and junctions by using encoding bits of 4-8 and larger conversion tables.

I explored adapting the algorithm to the TRS-80 Model 4 environment; however, a few snags reveal themselves. A close examination of the PC line drawing character set will reveal that true T-junctures exist as character values both in the vertical and horizontal directions. In the TRS-80, line drawing must use graphics characters which are fabricated in a 2 by 3 (H,V) cell matrix. Thus, you have true T-junctures as graphic characters only in the horizontal direction. For example, a right-T can be displayed with character value 157D and a left-T with character value 174. As a further drawback, the block graphics cell is developed using an underlying image only 10 dots in height. The cell is divided up into blocks of 4, 4, and 2 dots; thus, the graphics blocks are not of equal height.

One could use this algorithm by maintaining an array of values equivalent to that for the PC, and convert to available displayable characters. That is left as an exercise.

Now if someone has an answer as to the access of keystrokes in MS-DOS BASICS using ON KEY, I would be interested in seeing it.

```bas
10 '**********************************************************************
11 **
12 ** This program will estimate (very closely using double precision) 
13 ** your interest and principal at any time during the period of 
14 ** a loan with monthly compounded interest. It will request that 
15 ** the user supply an initial balance, interest rate, and monthly 
16 ** payment, and/or the duration of the loan, and from that compute 
17 ** the rest. This program may be used to budget your expenses. 
18 **
19 ** Variable Dictionary 
20 **
21 ** E - Balance remaining on the loan at any time (in dollars). 
22 ** D - Duration of the loan in months. Used only to calculate the 
23 ** monthly payment when user enters 0 for monthly payment. 
24 ** PP - Final payment on the loan. This amount may cause an extra 
25 ** month if it is significantly more than the monthly payment. 
26 ** HALF - Always 0.5, Used to compute monthly payment, and to round. 
27 ** ID - Interest paid to date. 
28 ** IF - Interest computed on the balance for the current month. 
29 ** M - Monthly payment (in dollars). 
30 ** MO - Current month number. 
31 ** PD - Principal paid to date. This value is computed as if the 
32 ** payment for the month has already been made, not at the start 
33 **
```

Inside TMQ - 20 - Inside TMQ
** of the month. The value for the previous month is then the **
** correct one. **
** PP – Principal paid in the current monthly payment. **
** R – Interest rate. This value may be entered in either the **
** annual rate (10.5) or decimal adjusted annual rate (.105). **
** TP – Total paid to date. This shows you how much your loan is **
** really costing you. **
** US$ – This string is used to format the output to pretty print **

```vbnet
100CLS:DEFDBL A-Z
105ON ERROR GOTO 1000
110DEF FNINTEREST(B, R) = INT(B * R * 100 + .5) / 100
120HALF = .5
130MO = 1
133PRINT TAB(28):"Loan Amortization Program";TAB(70):"AMORT1/BAS"
135PRINT:PRINT:PRINT
140INPUT "Initial Balance: ":B
150INPUT "Annual Interest Rate: ":R
160IF R > 1 THEN R = R / 100
170R = R / 12
180PRINT:PRINT USING "Your first month's interest is $###.###":FNINTEREST(B, R)
190PRINT USING "Monthly Payment (Enter 0 for automatic computation): ":M
200IF M = 0 THEN PRINT USING "Enter term of loan in months: ":D: IF D = 0 THEN
210PRINT USING "You must enter either Loan Term or Monthly Payment":SOUND 0, 0:
220GOTO 190 ELSE M = INT(HALF+B*(1-(1-R)^(-D)))/120)/100
230IF FNINTEREST(B, R) > M THEN PRINT USING "Error - you are not paying enough to
240PRINT USING "cover interest of $##.##.":FNINTEREST(B, R)
250PRINT:PRINT:GOTO 190
260LPRINT:PRINT USING "Printing amortization schedule started at ":TIME$.
270LPRINT:PRINT USING "Payment schedule at ":TIME$.
280LPRINT:PRINT "<--- Before Payment --- After Payment --- >":LPRINT
290LPRINT "Month's Month's Interest Principal Total Pd.".
300LPRINT "<--- to Date to Date to Date to Date "
310LPRINT
320US$ = "####.####.####.####.####.####.####.####.####.####.####.####.####.####.####.####"
330WHILE B + FNINTEREST(B, R) >= M
340TP = TP + M
360IF = FNINTEREST(B, R); ID = ID + IF
370PP = M - IF; PD = PD + PP
380LPRINT USING US$; MO, B, IF, PP, ID, PD, TP
390B = B - IF
400MO = MO + 1
410WEND
420IF B <= 0 THEN LPRINT STRINGS(LEN(US$), "--");LPRINT:GOTO 500
430IF = FNINTEREST(B, R); ID = ID + IF
440PP = B; PD = PD + PP
450TP = TP + PP
460LPRINT USING US$; MO, B, IF, PP, ID, PD, TP
470LPRINT STRINGS(LEN(US$), "--");LPRINT
490LPRINT USING "Last month's payment: $###.####.####":PP
500LPRINT
510LPRINT USING "Your total payment for the duration of the loan is: $###.####.####":TP
520LPRINT:LPRINT "Processing complete at ":DATE$; "":TIME$.
530LPRINT CHR$(12):
999END
100PRINT "Error - ":IF ERR = 57 THEN PRINT "Printer not ready" ELSE PRINT ERRS$.
1010PRINT:PRINT "Press any key to restart . . .":;
1020IF INKEYS = "" THEN 1020
1030RESUME 1040
1040RUN

---

Inside TMQ
Yet Another HiRes Graphics Format
by Hans de Wolf

Extending /CHR

When I was working on the Monosave program I discovered that the /CHR format was capable of storing images that use the full 32k RAM of hires memory by ignoring the limit of 80 bytes per line, just as the /SHR format does. Monosave creates such files if the X parameter is specified on the command line. In order to distinguish these files from /CHR and /SHR formats I introduced the /XHR extension (for xExtended High Resolution). Who wants to write a program that can read and display or print this new format?

Load Module Graphics

Although the /XHR format does its job (storing 32k pictures efficiently) I did not like it very much; it adds yet another graphics format to the already existing chaos. This made me think: I would like to have a graphics format that combines the strong points of all other TRS-80 graphic formats: the easy loading and writing of /HR files, the efficient storage of the /CHR format, a maximum image size 1024 by 255 pixels like /XHR, and support for image cropping like in the /BLK files. I came up with the following solution to this problem: Load Module Graphics. At this moment Load Module Graphic files are just an idea, but maybe the TRS-80 programmers out there could make it reality. Let me explain ...

LMG files are constructed along the design of the LDOS Load Module file formats, like used for /CMD files, the /SYS library files and PaDS data sets. If there is any accepted system standard for TRS-80 file formats this is it. All these files are constructed from Load Module blocks: packages of 256 bytes or less, each preceded with a type indicator and byte counter (more information can be found in the Pro-Cess/CMDFile manual). A single byte is used as type indicator, allowing for up to 256 types of load modules. The actual number of types in use is much less than this (01-08, 0A, 0C, 0E, 10 and 1F), leaving enough room for extensions to the format. Why not introduce a few new load module blocks to contain graphics data instead of memory load data?

An example: the Object Code load module block (type 01) consists of a byte 01 indicating the type, a single byte indicating the number of data bytes in the block and a two-byte load address, followed by the actual bytes to be loaded. Let’s use the same structure for a new Load Module type 21H, and use the 2 bytes in the load address as the values for the X and Y address where the load should start. The data bytes in the load module should be the bytes along a row (like a line in a /BLK or /HR file). Remember, a single row of a screen image can contain 1024 pixels, but no more than 128 bytes and will always fit in a load module. Because each block contains information about length and load address also parts of a screen can be stored in this format (the changing parts in an animation!).

We can even improve on this: introduce a 22H load module that is constructed in the same way, but do not store the raw data bytes for the image in data part of the module: store the image data bytes in the compressed RLE format that is used in /CHR files. While we are working, we can also add a 23H type block which is the same as 21H, but contains a column of an image instead of a row, and a 24H type for an RLE-encoded column.

In order to finish the job, we should patch the system loader in L($)DOS to recognize these new load module types.

Then add load modules for control purposes (switching video modes) and imagine what would be possible: complete animated slide shows could be stored in a single /CMD file. HiRes images belonging to a program would be loaded automatically by running the program. Any program would be able to load an image or start an animation by executing a DOS command, like SYSTEM “LOAD MOVIE/LMG” in BASIC ...

There may be technical reasons why this concept may not be feasible, but if anyone is looking for a programming project: why don’t you give this idea a try?
If your Model 4 is a floppy-only system, you only have to keep “backups” to guard against the inevitable loss of data. But when you graduate to a hard drive, Murphy’s Law requires you to start keeping “archives” of your files. To this end, MISOSYS, Inc., supplies the programs ARCHIVE and RESTORE with their hard drive systems.

ARCHIVE makes it possible to store, on floppies, copies of hard disk files which are too large to store on a floppy via normal means. It’s a bit slow, but it does the job, and does it well. But it is not efficient when dealing with small files enough to fit on your floppies. It adds extra sector to each file, and the speed is such that (F)BACKUP would be much faster. However, (F)BACKUP does not store the files very efficiently. Whenever the next file to be copied does not fit on the target floppy, you are prompted to insert another disk, even if the free space is only one granule less than the file size. This usually results in a great deal of wasted floppy disk space.

The accompanying listing, SMALARCH/ASM, is one solution to this problem. It makes use of the LS-DOS COPY command; so is reasonably fast, and it makes nearly optimum use of floppy disk space. To do this, it functions as follows:

1. It analyzes the target disk, and determines its capacity.
2. It analyzes the files on the source disk (hard drive partition), and determines which files to copy. It will not copy any file too large for the floppy, nor any .SYS, invisible, empty or password-protected files.
3. It sorts the files to be copied in descending order by size, and in ascending order alphabetically by file name within size. “Size,” in this case, means the number of 6-sector granules of floppy disk space.
4. Foreach floppy in turn, it goes through the sorted list of files and selects a set of files which most efficiently fills the floppy’s unallocated free space.
5. The selected files are then copied from hard drive to floppy and removed from the sorted list, (go to Step 3, above) until all files are copied.

Assuming the source disk contains files with a fairly broad range of file sizes (and/or lots of small files) SMALARCH will virtually always create floppy archive disks with zero free space. In my own experience, I have never seen SMALARCH leave a single granule of wasted disk space on a target floppy. However, the theoretical worst case is if every file to be copied is larger than half the size of the target floppy. In this case, SMALARCH would write one file per floppy, and waste just as much disk space as (F)BACKUP would.

To give you an idea of how SMALARCH’s BEST_FIT algorithm “thinks,” one of the tests I ran consisted of a set of 25 files: 5 each of 78 sectors (13 floppy granules), 5 each of 210 sectors (35 granules); 5 each of 300 sectors (50 granules); 5 each of 360 sectors (60 granules); and 5 each of 450 sectors (75 granules). All of these were to be archived to 360K DSDD floppies which hold 233 granules of data, after the requirements of DIR/SYS and BOOT/SYS.

The human brain would typically select five sets of 75 + 60 + 50 + 35 + 13 = 233 each to exactly fit the data on five diskettes.

The original SMALARCH merely stepped through the sorted list and picked the largest file which would fit on the disk. Using the above file sizes, this would result in the first disk having 75 + 75 + 75 = 225 granules used, and 8 granules wasted. The second disk would have had 75 + 75 + 60 + 13 = 233 granules used, and 10 granules wasted, etc., spreading five disk’s worth of data over six disks.

Obviously, a more sophisticated approach was required. Enter the BEST_FIT algorithm. I originally wrote BEST_FIT rather differently than it appears here. Originally, it was part of an order-filling system for a manufacturer who wanted to fill distributor orders as efficiently as possible without “breaking” boxes of the items to be shipped. Thus it had to pick the smallest combination which was equal to or greater than the quantity desired by the customer.

Obviously, this new version of BEST_FIT, since it tries to fill available space, must pick the largest combination which is equal to or less than the requested total quantity.

Using the set of test files described above, the algorithm (because it prefers to use the largest files first) created two floppies of 75 + 75 + 75 + 35 + 13 = 233; one floppy of 75 + 60 + 50 + 35 + 13 = 233; and two floppies of 60 + 60 + 50 + 50 + 13 = 233. It succeeded, in other words, in fitting exactly five disk’s worth of data on exactly five disks. Essentially, BEST_FIT is a “brute force” approach to the problem. The recursive nature of the algorithm
creates, in effect, up to one FOR-NEXT loop for each quantity in the list. Thus, if you have 15 files to copy, there could be as many as 15 nested FOR-NEXT loops created in the program.

The following BASIC pseudo-code is a ROUGH representation of how this works:

```
TOTAL = 0 : BEST = 0
FOR XO = 0 TO 14
    TOTAL = TOTAL + QUANTITY(XO)
    IF (BEST < TOTAL AND TOTAL <= GOAL) THEN BEST = TOTAL
    IF TOTAL < GOAL THEN
        FOR X1 = XO + 1 TO 15
            TOTAL = TOTAL + QUANTITY(X1)
            IF (BEST < TOTAL AND TOTAL <= GOAL) THEN BEST = TOTAL
            IF TOTAL < GOAL THEN
                FOR X2 = X1 + 1 TO 15
                    TOTAL = TOTAL + QUANTITY(X2)
                    IF (BEST < TOTAL AND TOTAL <= GOAL) THEN BEST = TOTAL
                    IF TOTAL < GOAL THEN
                        FOR X3 ... (etc.)
                        TOTAL = TOTAL - QUANTITY(X3)
                        NEXT X3
                        IF TOTAL = GOAL GOTO JOBDONE
                        TOTAL = TOTAL - QUANTITY(X2)
                        NEXT X2
                        IF TOTAL = GOAL GOTO JOBDONE
                        TOTAL = TOTAL - QUANTITY(X1)
                        NEXT X1
                        IF TOTAL = GOAL GOTO JOBDONE
                        TOTAL = TOTAL - QUANTITY(X0)
                        NEXT X0
```

As can be seen from the above, if no combination of quantities in the list adds up exactly to the goal, the search can be quite extensive and therefore extremely time-consuming (something like five or ten minutes to go through a list of 40 files). Normally, the only time this will occur in SMALARCH is for the last disk, when the total size of all files is less than the goal.
To avoid such "endless" searches, this special case is handled very quickly by code in BEST_FIT.

To see how this concept is implemented, let's go to the program listing, starting at the label BEST_FIT.

Because all references to the variable arrays in the program are base zero, the number of the last array element in use will always be one less than the number of elements. For example, if there are 100 files to copy, array elements 0 through 99 will be in use. For convenience later on, the variable LAST (which refers to the last array element in use) is set to one less than COUNT, which is the number of not-yet-copied files in the sorted list.

Whenever the algorithm finds a better BEST solution to the problem, the files selected are recorded in an array of one-byte flags which are related by position to the elements in the main ARRAY of files. That is, the 10th flag relates to the 10th element of ARRAY, etc. Obviously, these must be initialized to zero (FALSE) upon entry.

Next the sum of all files in the list is accumulated. If this sum is less than the goal (which is the free space on the target disk), it makes no sense to make a long search for a sum which can never be found. Also, if the sum is exactly equal to the goal, there is no need to make a search. Thus, in these special cases, the BST_FLG flags for all remaining files will be set, and the sum of all files returned to the calling routine in the HL register.

In the only other case is that the sum of all file sizes is greater than will fit on a floppy, which means a selection of a set of the remaining files must be made, which is done by CALC_FIT.

Now that we know we'll be invoking CALC_FIT, and since the BST_FLG array is filled from the contents of the TMP_FLG array whenever a better value for the BEST solution is found; all TMP_FLGs are now initialized to zero

```
ORG 0D600H

.BIT EQU $8

/* these messages are disposable after syntax checked: */
HELLO$  DM LF, 'SMALLARCH - (c) 1990/92 by J.F.R.
SYNTAX$ DM LF, 'Syntax is: SMALLARCH -d
        DM 'where :d is the drive number of the
        DM 'Hard drive partition to copy from', LF
        DM TAB+7, ' :d is the number of the floppy', LF
        DM 'drive to copy to', LF, LF
        DM 'Parameters:', LF, LF
        DM TAB+7, '/update=no' (u) preserves set MOD'
        DM '/flags', LF, TAB+7, '/verify=yes' (v)
        DM 'causes verified disk writes', LF, CR
HDTARGET$ DM 'Target drive is not a floppy - function
        DM 'aborted', LF, CR
STARG$  DM 'Target drive cannot be system drive -'
        DM 'function aborted', LF, CR

; STACKEQU$ ;program stack after HIT is read

; variables:
PTRO DS 2*254 ;pointers to array elements
ARRAYDS 13*254 ; struct ( int grades;
                 |     char filename[13]; )

ARRAY SIZE EQU $-ARRAY
BST_FLG DS 255; one flag for each array element
TMP_FLG DS 255; plus one
BEST DS 2
ACCUMDS 2
LAST DS 2
FREE DS 2 ; storage for free space on floppy
DIRBUF DS 20 ; buffer for directory records
SVCMSB DS 1 ; storage for msb of SVC tbl addr
COPYCTR DB 0 ; inits to zero

; messages:
BREAK$  DM 'Break key detected - program
        DM 'terminated', LF, CR
BPTERR$ DM 'best_fit function error - program
        DM 'terminated', LF, CR
NEWDSKS DM LF, LF, TAB+19; 'Insert new disk in drive
DRIVNO DM 'x and hit <ENTER>', CR
COMANDS DM 'copy filename/ext:s :d', CR
FILNAM EQU COMAND+5
DRIVES$ DM 'x : x', CR
SOURCES EQU DRIVE$+1
TARGETS EQU DRIVES$+4

; subroutines
PERFORM PUSX IX ; issue LS-DOS COPY command
CALL SWAPCD ; change @EXIT code
LD [PRGSTK], SP ; save program stack ptr
LD SP, &$-8
SYSTACK EQU &-2 ; load SP w/system stack
LD HL, COMAND
SVC@CMNDI ; issue copy command
```
CALC_FIT uses two kinds of variables which, in C parlance, are called “global” and “local”. The ARRAY of file sizes and names, the array of pointers (PTRO), ACCUM and BEST are examples of “global” variables. They can be accessed by code anywhere in the program, including CALC_FIT.

The “local” variables, on the other hand, are known only to CALC_FIT, and cannot be accessed by any code outside of CALC_FIT. Because CALC_FIT is recursive (i.e., it calls itself), its local variables must be stored in such a way that each level of recursion has its own unique set of local variables. The best way to do this is to store the variables on the stack, à la C. Each invocation of CALC_FIT uses six levels of stack, and since there can be as many as 254 files in a list, there must be about 3K of stack space available. This is provided for elsewhere in the code.

Before calling CALC_FIT, the calling routine must push three variables on the stack, namely the number of array elements to search, the array position at which to start the search (initially element number zero), and the sum of sizes to find. In CALC_FIT itself, these are called “number”, “start”, and “target”, respectively. CALC_FIT also uses two other “local” variables, namely “total” and “local_best”, which it also pushes onto the stack. The order of appearance on the stack is of crucial importance, as this is how CALC_FIT accesses them.

Of the six levels of stack required, five contain variables (the sixth being the RET address), and are accessed via their location relative to the stack pointer, as follows:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>number</td>
<td>SP +10</td>
</tr>
<tr>
<td>start</td>
<td>SP + 8</td>
</tr>
<tr>
<td>target</td>
<td>SP + 6</td>
</tr>
<tr>
<td>RET address</td>
<td>SP + 6</td>
</tr>
<tr>
<td>total</td>
<td>SP + 2</td>
</tr>
<tr>
<td>local_best</td>
<td>SP + 0</td>
</tr>
</tbody>
</table>
The references for the next recursive iteration of CALC_FIT will access its five local variables the same way, but these five will be different from the five used by the previous level of recursion, and each will be six levels of stack (12 bytes) lower in RAM than the corresponding variable used by the calling level. Each successive recursive call will generate a new set of five variables and one RETurn address.

When a level of recursion returns to a higher level, the value found is returned to the caller in the HL register, and the stack pointer increased by 12 through a series of POPs so it once again references the local variables of the current level. The CALC_FIT listing is commented with the original C source statements, and is well enough commented in general that it would be redundant for me to go through the listing step by step in this article. The only area that needs explanation is the logic behind the C statement:

```c
if ( total < target )
    total += calc_fit( target - total, start + 1, number - i );
```

the implementation of which begins at the label CFT030. At this point, total holds the value of array[start].grans — the size of the file at the top of the current portion of the list being searched.

If total is too small, then it must be increased, which is done by adding the size of some file further down the list. To do this, a new search is initiated starting with the next array element, with the target being the difference between what we need and what we've got, and the values for "start" and "number" adjusted to describe where to start the search and how many elements to search. Then CALC_FIT calls itself to perform this search.

The new iteration has a new total (the size of the first file for this search) and a new target, as described above. If the total is too small, then CALC_FIT calls itself again, with new values for target, start and number.

Thus the program will loop from the label...
CALC_FIT to the CALL CALC_FIT instruction until a sum greater than or equal to the target is accumulated. In each of these iterations, the total of all sizes held in ACCUM is compared to BEST, and BEST and the BST_FLG array updated each time a better fit is found.

Once a total equal to or greater than target is found, the program will loop from the first POP DE instruction after CALL CALC_FIT to the first RET instruction executed by the code as many times as CALC_FIT called itself (i.e., if CALC_FIT calls itself ten times, it must return to itself ten times).

Finally, when either an exact solution has been found (i.e., ACCUM == GOAL) or all possibilities have been exhausted, return is made to the calling routine with the solution value in the HL register and the component parts of the solution identified in the BST_FLG array.

To illustrate further, consider an example where there are 3 files in the list, with the sizes 10, 5 and 3 grans respectively, and a “target” of 18 grans. The calling routine would push a “count” of 3, a “start” of 0, and a “target” of 18 onto the stack, and then call CALC_FIT.

The first iteration of CALC_FIT would make “total” equal to 10 (the size of the first file in the list), and then call itself with a “count” of 2, a “start” of 1, and a “target” of 8.

The second iteration would assign 5 (the size of the 2nd file on the list) to “total,” and then call itself with a “count” of 1, a “start” of 2, and a “target” of 3.

The third level of recursion would return a value of 3 (the size of the LAST file on the list) in BL. This 3 would be added to the “total” in the second level of recursion, which would then return the value 8 in HL to the highest level of recursion. This 8 would be added to the first level’s “total” of 10, and the value 18 would be returned to the calling routine in HL. Now consider what would happen if,
in the above example, the target is 9 instead of 18. In the first level, the “total” of 10 is larger than the “target” of 9, so the routine does not call itself, but rather restores ACCUM to zero, and starts a new search with “count” = 2, “start” = 1, and “target” = 9.

This time through, “total” becomes 5; so the routine calls itself with count = 1, start = 2, and goal = 4. The second level of recursion returns 3, which the first level adds to its total of 5. Because the accumulated 8 is not equal to the goal of 9, a new search is started at the 3rd file, which returns 3.

All possibilities are now exhausted; so return is made to the caller, which loads BEST into the HL register, and returns to its caller with the second and third BST_FLG flags set and 8 in HL.

In this way, BEST_FIT tells the caller the closest it could come to the goal of 9 was 5 + 3 = 8, and the 10-gran file will have to be written to another disk.

One final example: 5 files with sizes of 16, 11, 8, 7 and 3 gran, and a goal of 18.

In the first level, count = 5, start = 0 and target = 18. The first total is 16; so the routine calls itself with count = 4, start = 1 and target = 2. Since subsequent levels of recursion cannot find a value <= 2, zero is returned to the top level, and a new search started with count= 4, start= 1, and target = 18.

The first total becomes 11; so the routine calls itself with count = 3, start = 2 and target = 7. In this level, total becomes 8—too large—so ACCUM reverts to 11, and the next file size is checked. Total now becomes 7, which is equal to target; so it is returned to the top level of recursion, where the 7 is added to 11 to get 18. Because 18 = goal, the routine returns to the caller.

Note that in this case there is another solution: 8 + 7 + 3 = 18; but the algorithm found 11 + 7 = 18 first; so that’s what it
settled on. In general, it's better to use up larger files first anyway, because they're harder to "fit" than smaller files.

O.K. Now that we know how the data is processed, let's go to the label BEGIN to learn how the program obtains the data, and then disposes of it.

First, the pointer to the remainder of the command line (everything after "SMALARCH" plus the following space) in HL is pushed.

Then, so the state of the system can be restored at the end, the value of the current state of the D flag is stored, and the vectors for the @EXIT, @WRITE and @VER SVC's are retrieved from the SVC vector table and stored various places in the program code. The same is done with the high memory pointer.

Now the high memory pointer is saved in case of an error exit.

Next the command line pointer is restored, and the command line parsed. The code looks for the required ":s" parameter, and aborts either if it does not exist or if the specified source drive number is out of legal range.

Next, the required ":d" parameter is sought, with an abort if it is not present or contains an illegal value.

Finally, the optional UPDATE and VERIFY parameters are checked by the @PARAM SVC. If the VERIFY=YES parameter is used, then the @WRITE SVC vector in the SVC vector table is replaced with the @VER vector, and the verify bit in the D flag set to tell the system about this change.

Because SMALARCH will not copy /SYS files, and there are a minimum of two /SYS files per directory and a maximum of 256 file slots per directory, the maximum possible number of files to be copied is 256 - 2 = 254. Therefore, SMALARCH sets up 254 array elements, each initialized with NULLS, a value which signals

```
LD D, (HL) ;DE = array[last].grans
LD HL, 6
ADD HL, SP
GETWORD
EX DE, HL ;DE = target; HL = size
INC DE
OR A
SBC HL, DE ;size <= target?
JR NC, CFT020 ;go if not
{: local_best = array[last].grans;
ADD HL, DE ;HL = array[last].grans
POP DE
PUSH HL ;replace local_best
: best flag[last] = TRUE;
LD DE, (LAST)
LD HL, $B7_FLG
ADD HL, DE
LD (HL), -1
;
;while ( start < last )
CFT020 LD HL, 8
ADD HL, SP
GETWORD ;start
LD DE, (LAST)
OR A
SBC HL, DE ;start < last?
JR C, $+5 ;go if so
;
;if start == last, search done - return local_best
POP HL ;local_best
POP AF ;clear stack
RET
;
{: memset( temp_flag[start], FALSE, number );
ADD HL, DE ;HL = start
LD DE, TEMP_FLG
ADD HL, DE ;HE -> temp_flag[start]
LD (HL), 0
LD D, H
LD E, L ;DE -> temp_flag[start]
LD HL, 10
ADD HL, SP
LD C, (EL)
INC HL
LD B, (EL) ;BC = number
LD H, D
LD L, E ;HE -> temp_flag[start]
INC DR ;DE -> temp_flag[start+1]
DECC
JR Z, $+4 ;no LDIR if none to copy
LDIR
: temp_flag[start] = TRUZ;
LD HL, 8
ADD HL, SP
GETWORD ;start
LD DE, TEMP_FLG
ADD HL, DE
LD (HL), -1
: accum += total = array[start].grans;
LD HL, 8
ADD HL, SP
GETWORD ;start
ADD HL, HL
LD DE, PTR0
```
Therefore, the next two small sections of code fill the array data area with bytes of OOH, and create 254 pointers, one to each array element. Each 13-byte array element contains the following data:

- **size (in floppy grans)**: 2 bytes (lsb,msb)
- **file name**: 8 characters, right padded with blanks
- **file extension**: 3 characters, right padded with blanks.

Next, the Hash Index Table (HIT) of the source drive is read. Each non-zero entry in this table is associated with a file; so each is checked in turn, and the directory entry for each existing file is read. The file type is then checked, and only visible, non-protected normal and PaDS files are accepted.

When a file passes these tests, its size in sectors is checked. If it is too large to fit on the floppy, or it's an "empty" file (i.e., one with no records), it is skipped.

Five is then added to the number of sectors, and the sum is divided by 6, the number of sectors per floppy granule. This yields the number of floppy grans required to hold the file. This value is written (in lsb,msb order) to the first two bytes of the next unused array element. Then the file name and file ext are copied from the directory record to the array element, and the next HIT entry checked until all 256 have been checked.

Finally, a count is made of the number of files to be copied, with the result placed in the word COUNT in the shell sort subroutine. Now that we have dispensed with the need for the HIT table and the initial informational messages, the stack is switched to STACK (the program stack) to accommodate the large stack requirements of CALC_FIT, as well as cope with...
the effects of using the @CMNDI SVC to invoke the LS-DOS COPY command.

At DSKDONE, the array is sorted, and a check made to see if there are any more files to be copied. If so, COUNT is updated, and then, at PROMPT, the user is prompted to insert a diskette in the drive specified in the “:d” parameter.

The diskette is then checked, and the amount of its free space in granules is calculated and stored in the word labeled FREE.

If there is no free space, the user is prompted to insert another disk. Otherwise, BEST_FIT is called to select which files to copy.

Now the BST_FLG array is stepped through in order. If a flag is set, then the associated file’s size is set to zero to mark it as “used”.

Since the files are sorted by file size in descending order, and the value of COUNT updated after each sort, a size of zero will cause the array element to be sorted out of the active portion of the array the next time the list is sorted. This will, in effect, remove the used files from the list. Next the file name and file extension are copied in “FILENAME/EXT” format to a portion of the COMANDS$ string, and the source and target drive numbers added to the end of the string in “:s:d”+CR format. The file is then counted in the COPYCTR byte pointed to by IX, and the PERFORM subroutine called to issue a system COPY command.

At PERFORM, the IX register is preserved, and SWAPCOD is called to replace the code pointed to by the @EXIT SVC vector to a JP to PFM010. Then the program stack pointer is saved, and the system stack pointer switched in. The COPY command contained in COMANDS$ is now issued via invocation of the @CMNDI SVC.

Normally this SVC does not return, but in this case it does, because the program has
changed the code for the @EXIT SVC. Return is to PFM010, where the program stack is switched in again, the original code for the @EXIT SVC restored, and any error code from the COPY command and the IX register are restored before return to the caller.

Upon return, the value in HL is checked for an error code, and, if none, a jump is made to SCT010 to get the next file to copy. In case of any error, or end of job, exit is via the code at EXIT.

Here any error code in HL is preserved, and the original high memory pointer restored. If SMALARCH was invoked with the (verify=yes) parameter, both the @WRITE SVC vector and the D flag are also restored to their original values.

(Note that if the system is already in "verify=on" mode, SMALARCH will not change it. The program's "verify=yes" parameter will turn verify on, but the "verify=no" parameter or default does not turn verify off. In other words, SMALARCH uses the status of verify upon entry as its default.)

Finally, the error code in HL is restored, the system stack is switched back in for the last time, and return to LS-DOS is made via the @EXIT SVC.

The positioning of SMALARCH/CMD in RAM is such that 4K+ have been preserved for high memory routines by the ORG address. I would think (hope) that 4K should be adequate for virtually any system configuration.

However, if it is not (i.e., the MEMORY command shows a himem pointer of x'0e03' or lower) you need to change the ORG statement (reducing it by an even multiple of 100H) to accommodate your himem needs.

SMALARCH could be rewritten to automatically relocate itself to high memory, but including that code in this article would have greatly lengthened and complicated the listing, making it harder to
To understand the actual functions taking place, take the disks, in order from first to last, and execute a `DIR` command.

To see how SMALARCH organizes the files on archive floppies, take the disks, in order from first to last, and execute a `DIR` command. A typical directory appears below:

Note the files appear in descending order by size. Look at the two 25.5K (17 gran) files, ELVIS2/GIF and GOTHIC17/GIF. They are not the same size in sectors and bytes, but they are the same size in gran. This is why they were written in alphabetical order. The same is true of the three 11-gran files (BRINKLEY/GIF, CAPITOL/GIF AND JUNE/GIF) and the two 10-gran files (COKE/GIF and JANUARY/GIF). The last file, AUDREY/GIF, was obviously at the top of the list of 3-gran files; and was selected because it was the first file encountered which exactly filled the remaining 4.5K of free space on the floppy.

I have found SMALARCH to be very useful in making my own archives of hard disk files, and hope and trust TMQ readers will find it equally useful.
CP BLANK
JR Z,BGN020 ;skip spaces
CP CR ;end of command line?
JR Z,BGN030 ;no params if so
CP '(' ;list char = ')?
JP NZ,SYNTAX ;error if not
LD D2,PARAMS
SVC@PARAM ;parse params
JP NZ,ERROR
LD A,(URESP)
OR A ;was update param used?
JR Z,$+6 ;go if not
AND$0E ;correct type of response?
JP Z,SYNTAX ;error if not
LD A,(VRESP)
OR A ;was verify param used?
JR Z,BGN030 ;go if not
AND$0E ;if so, correct type?
JP Z,SYNTAX ;error if not
STIT2,(IY+$D$-'A') ;set verify bit in D flag

LD A,(VERIFY)
OR A ;verify requested?
JR Z,BGN040 ;go if not
LD A,(SVCMSB)
LD R,A
LD A,@WRITE
ADD A,A
LD L A,;HL -> WRITE vector
LD DE,$-$
$VEREQ$b$-2
LD (SL),E
INCHL
LD (HL),D ;replace w/$VER vector

LD C,6
JP Z,STARG ;target can't be sys drive

LD A,DRIVNO
AND7
JP Z,STARG ;target can't be sys drive
LD C,A
SVC@CDRV ;is target drive ready?
JP Z,ILDRIVE ;abort if so
LD A,(TY+0)
CP OC9E ;is drive disabled?
JP Z,ILDRIVE ;abort if not
LD C,MODR ;get space of target drive
LD C,(TY+6) ;C = # of cylinders - 1
LD A,MY+8) ;(omits DIR/SYS cyl)
AND EOH ;isolate grains/cyl/side
RLCA 
RLCA 
RLCA 
INCA ;base 0 adjust
LD L A
LD B,0 ;EL = # grains/cyl/side
BIT5,(TY+4) ;double-sided drive?
JP Z,$+3 ;go if not
ADD HL ;els double grains/cyl
SVC@MUL16 ;multiply * # of cyls
LD E L,;HL = # of free granules
LD L A ;on an empty disk
DEC E ;remove 1 gran (BOOT/SYS)
LD C,6
SVC@MUL16
LD L A ;EL = # free sectors on
LD L A ;empty disk of this type
LD (SLIMIX),EL ;store result
fill sort data w/nuls (size of x$00000' means EOD)
LD HL,ARRAY
LD DE,ARRAY+1
LD BC,ARRAY_SIZE-1
LD (HL),0
LDIR
build 254 pointers to array elements
LD B,254
LD IX,PTR0
LD HL,ARRAY
LD DE,13
BGN050 LD (IX),L
INC IX
LD (IX),H
INC IX
ADD HL,DE
DJNZ BGN050
read source drive HRT, and check all files for copying
LD HL,HRT
LD DE,1
LD A,(SOURCE$) ;p/u source drive $
AND7 ;convert to binary
LD C,A
SVC@RDSSC ;read the HIT
JP NZ,ERROR
EX DE,HL ;DE -> HRT
LD IX,ARRAY
BGN060 LD A,(DE) ;p/u HRT entry
JP A ;slot in use?
JR Z,BGN100 ;try next slot if not
LD B,E ;DEC to B
SVC@DIRRD ;read directory entry
JP NZ,ERROR
PUSH HL
POP H
;RT -> directory record
LD A,(HL) ;p/u 1st byte
AND$11111111 ;0 means "don't care"
; (re PADS files)
CP 108 ;exclude FXDE, SYS, INV.
; and protected files.
; require file to be ACTIVE
JR NZ,BGN100 ;go if not to be copied
PUSH DE ;else save HRT pointer
LD Z,(TY+20)
LD D,(TY+21) ;DE = # of sectors (ERN)
LD A,D
OR E ;empty file (zero size)?
JR Z,BGN070 ;don't copy if so
LD HL,$-$ ;# free sectors on empty
SLIMIT EQU $-2 ; floppy disk
SBC HL, DE ; will file fit on disk?
JR NC, BGNO80 ; go if so
BGNO70 POP DE ; else restore HIT ptr
and
JR BGN100 ; try next DEC

BGNO80 PUSH BC ; save DEC & drive #
PUSH BC ; twice
LD HL, 5 ; value for rounding
ADD HL, DE ; HL = # sectors + 5
LD C, 6 ; # of sectors/floppy-gran
SVC$DIV16 ; HL = # floppy grans req
LD (IX), L ; store 1st
INC IX
LD (IX), H ; store 2nd (last word of
INC IX ; sort key is size [grans])
EX DE, HL
POP BC ; DEC & drive #
PUSH IX
LD A, (UPDATE)
OR A ; update flag set?
JR Z, BGN090 ; skip next section if not
RES 6, (IX+1) ; else reset MOD flag
POP IX
POP HL ; HL -> DIR record
SVC$GRWR ; rewrite DIR entry
JP NZ, ERROR ; w/MOD flag reset
BGNO90 POP HL ; HL -> DIR record
LD DE, 5
ADD HL, DE ; HL -> filename + ext
PUSH IX
POP DE
LD BC, 11 ; length of filename + ext
LDIR ; copy to sort key
PUSH DE
POP IX ; IX -> start of next sort
key
POP BC ; DEC & drive #
PPOPDE ; restore HIT pointer
BGN100 INC E ; DE->next HIT entry
(falls
JR NZ, BGN060 ; thru when 256
checked)

; determine how many files to copy (for
; sort & search)
;
LD IX, COUNT ; -> lab (cannot exceed
254)
LD HL, ARRAY ; -> first element
LD DE, 12 ; array element size - 1
LD B, 254

BG110 LD A, (HL) ; p/u lab of size
INCHL ;
OR (HL) ; is size zero?
JR Z, BGN120 ; end of data if so
INC (IX) ; else bump COUNT
ADD HL, DE ; HL -> next element
DJNZ BGN110 ; check next
BGN120 LD IX, COPYCTR ; COPYCTR init to
00H
LD (SYSTACK), SP ; switch over to pgrm
stack

DSKDONE CALL SHELSRT ; sort in
size/alpha order
LD A, (COUNT) ; p/u # files in last disk
SUB (IX) ; sub # copied to last
disk
JP Z, EXIT ; done if none left to
copy
LD (COUNT), A ; else update COUNT
LD (IX), 0 ; and reset COPYCTR
PROMPT LD HL, NEWDK$ ; prompt user for
new disk
SVC$DISPL

PMT01 SVC$KEY ; wait for keystroke
CP 80H ; break?
JP Z, BREAK
CP CR ; enter?
JR NZ, PMT010 ; #f if not
LD A, (DRVINO) ; p/u target drive #
AND $ ; convert to binary
LD C, A
LD B, 4
LD HL, DIRBUF
SVC$DODIR
JR Z, PMT030 ; device not available?
JR Z, PROMPT ; if so no disk - reprompt
JP ERROR ; else report error, quit
PMT020 LD HL, (DIRBUF+13) ; p/u free K
LD A, H
OR L
JR Z, PROMPT ; go if no free space
ADD HL, EL ; free K * 4 + 2 is maxi-
mum
INCLH ; possible of # of free
ADD HL, EL ; sectors on this
floppy
LD C, 6 ; divisor is sectors/gran
SVC$DIV16
LD (FREE), HL ; store # of free grans
; Call best_fit() to determine which files
to write to
; floppy. Step through best flags and set
; sizes of marked
; files to x'00000000' (causes elements to sort
; out of active
; portion of array next sort), and COPY the
; file.
;
SELECT CALL BEST_FIT ; flag files for
COPYing
LD A, B
OR L ; did function return NULL?
JP Z, BFTERR ; something wrong if so
LD HL, BST_FLG
LD BC, (COUNT)
INC BC
SCT010 LD A, -1
CPX A ; search for set BST_FLGs
JP PO, DSKDONE ; go if none left
PUSH HL ; save search start
PUSH BC ; and search count

Inside TMQ - 36 - Inside TMQ
15. Properly align the previously removed hub into the left hand side of the drum. You may need to pull back on the spring clip using a thin tool (such as a popsicle stick) to fully engage the hub.

16. Re-insert the previously removed screw into the hub.

17. Insert the drum cover plate into the top of the assembly. Insert the right hand side first. Position so that the extra cutout on either side of the channel is positioned towards the handle. Note that the plastic fingers on the right are harder to correctly position.

You now have a refurbished PC cartridge to use in your machine. Depending on the condition of the photoconductive drum, your old cartridge may be good for another 3000 copies - at least as far as the capacity of the waste toner receptacle!
Long-standing readers of The MISOSYS Quarterly will recollect that I use a NEC LC-890 Laser printer (actually an LED printer). I have commented on this in the past, as the printer is used in production of TMQ as well as to produce manuals for MISOSYS products.

The printer has held up well over the years, with just a few problems. However, when problems occur, they appear to be expensive to fix. The predominate expense is labor, as the local NEC authorized repair center has charges of about $60 per hour. Therefore, when the machine produces a dirty image, it can be significantly less expensive to learn not only how to clean it, but why it gets dirty.

This article will discuss two issues concerning the LC-890. The first issue will be two reasons why a dirty image can result. The second is a means whereby you can recondition your own Photo Conductor (PC) cartridge.

To begin with, other machines are known to use the same print engine; and as such, this material applies as well. Based on the label of a toner cartridge I purchased from Quill, the NEC engine is used on the AES L5, Datasouth Pagewriter 8, Interface Systems Laser 8, Kaypro Page Printer I, NEC LC850, 860, 890, Nixdorf, Siemens PT10, and Telenorma Model ISY-70. All of these machines use a toner cartridge separate from the PC drum cartridge. I have always been impressed with the print quality - when the machine is clean and properly working.

One problem of a dirty image appears as incomplete coverage of toner - especially on large black areas (solid graphics or a large type size). The incomplete coverage may be on one side of the paper. If cleaning of the components as discussed in the User Manual do not clear up the problem, it is off to the shop for an expensive cleaning. Or is it? After one such trip, I discussed the issue with the shop technician.

It turns out that the LC-890, et al, is plagued with a problem of toner clumping depending on the frequency of use, and the environmental conditions (temperature and humidity). I have always been of the belief that a car will last longer the more it is used. When it stays parked a great deal of time, it will rust out. The same principle holds true with the printer - although rust out may not be the exact terminology to apply.

Toner is itself an extremely fine granulated material. When the toner particles clump together, they will cause the image to be incomplete as the clumps will not properly transfer as particulate to the paper. At this point, the toner clumps need to be removed. You can determine if your machine has toner clumping by examining the developer assembly once it is removed from the machine. That's simple matter to do. Simply open up the toner cover used to add toner, then pull up on the cover to remove it; it's held in place by channels at either end. Then unscrew the two large screws located at either end of the developer assembly and lift the developer unit straight up. It may appear to not budge at first - the fit is tight. Incidentally, the screws will not disengage from the assembly.

As you look at the front of the assembly, notice the appearance of the toner on the roller. If it's not a fine particulate, the toner has clumped. You may even want to slightly reverse the normal direction of the roller by rotating the roller gear to see how the toner appears. When it is clumped, you may be able to just vacuum away the clumps (using one of those small hand-held vacuums). If the problem is with the toner throughout the developer unit, all of the toner will have to be removed and the unit cleaned up. Since I did not have to go that route, I won't explain that here. I was able to remove the clumped toner just by vacuuming away the surface clumping. Once that was done and the normal areas cleaned, my machine performed nearly as good as new.

Now a laser printer is quite similar in operation to a copying machine. A charge is placed on a drum which rotates and attracts toner; a sheet of paper traverses the drum transferring the image to the paper. The paper is then passed through a heater which fuses the toner to the paper. Not all of the toner which comes in contact with the drum is used for the image; some is wasted (only the toner which adheres to the charge on the surface of the drum is used). When it comes to the wasted toner, there are two classical methods to deal with the waste. One method is to recycle the waste toner back into the supply toner. The other is to catch the wasted toner into a bag or bottle to be subsequently discarded.

I had a Minolta copier which recycled its toner; thus, there was never any need to discard a toner bag. I also have had - and still have - two Toshiba copiers which maintain a receptacle for accumulating the wasted toner. When the bag becomes filled, a warning light appears which renders the machine inoperable. I never thought about the method used to handle wasted toner on my laser printer until one day when I was having considerable difficulty keeping it clean.

PC cartridges are relatively expensive.
Quill's price is $129.88. These are supposed to last about 7000 copies; a toner bottle lasts for about 3000 copies. I always thought that the 7000 limit was based on the usability of the photoconductive drum. However, it turns out that this is not the case. The drum can last anywhere from 0 to 10,000 or more copies - depending on how many imperfections you want to live with; minute scratches on the surface can cause dot blemishes to appear on the image transferred to the paper. The other dependency is impacted by the method used to accumulate wasted toner. With $130 for PC cartridge replacement, there is a tendency to want to use a PC cartridge past the 7000 point - especially when using the machine to print out program listings.

I was sitting one day at the printer trying to determine why it was dirtying up continuously after just one page of printing when it hit me; the printer must somehow accumulate the wasted toner in the PC cartridge itself and the chamber used to bottle the waste was filled up - there could simply be no other mechanism because the print engine was too simple a device! Since I have saved every PC cartridge I purchased (never know when someone would start recycling them), I proceeded to examine one of my used cartridges.

A close inspection revealed an easy method to disassemble the cartridge. Once the drum was removed from the cartridge, it revealed a compartment which was fill of toner. My problem was the cartridge I had been using had its waste toner compartment filled up; thus the new waste toner had no place to go but spill over onto the paper dirtying up the image. The machine provides no warning when this happens - you're supposed to replace the PC after 7000 copies! All I had to do was clean out the waste toner compartment, and I would then have an almost good as new cartridge. The following procedure can be used to perform the job. I have cleaned two cartridges using this procedure which have provided me an economical way to extend the life of a cartridge. Pay close attention to the use of the plastic bag.

Cleaning a NEC PC Cartridge
1. Obtain a large plastic bag - one in which you can place the PC cartridge and a receptacle to receive the waste toner (I use an empty toner cartridge to receive the waste).
2. Roll up your sleeves and remove any jewelry (you want to minimize the chance of getting toner on you).
3. Open up the empty toner cartridge or other receptacle you will use to receive the waste toner and place it in the bag.
4. Looking at the top side of the PC, there is a plastic panel with a channel cut into it on the top front just over the drum. Four plastic fingers (two per side) keep it engaged. Slightly depress the left two fingers (there's rectangular holes on the side to reach the fingers) and pull up on the panel. Do the same for the right side. The panel can now be removed.
5. A Philips screw is located on the left side of the drum. Remove the screw and save it. Using a thin blade screwdriver, letter opener, or pocket knife, pry loose the drum hub end and plastic cover (they're a one-piece assembly).
6. From this point on, always hold the PC such that the side with the drum is facing up - the drum must be at the highest point. Otherwise, toner will come cascading out of the assembly from the used toner compartment. Also, avoid touching or bumping the reflective surface of the drum!
7. Disengage the drum from the side where the hub was just removed. Pull up and out at an angle until it clears the retainer; the right side will follow.
8. Place the removed drum on a soft surface. A sheet of bubble wrap saved for this purpose is fine.
9. Immediately below where the drum was located is the reservoir for holding the waste toner. This toner is what needs to be removed.
10. Place the PC assembly in the large plastic bag with the opened toner cartridge.
11. Carefully pour the toner from the PC into the empty receptacle. This will be a messy operation as the waste toner will not pour in a controllable flow. The plastic bag keeps the messy toner which you spill from getting onto everything. Toner is tough to clean up. I don't recommend re-using this waste toner as it may have some clumps in it. Nevertheless, it's your choice.
12. After all of the toner has been evacuated from the PC, you need to clean up the cartridge. You can vacuum any remaining toner misting on the surfaces. You can also wipe the assembly with a damp cloth - but I found that vacuuming is still required after wiping.
13. At this point, you should clean the drum. I use ordinary rubbing alcohol on a soft cloth for this. By positioning the drum vertically at the edge of a table and holding it at the top, you can rotate the drum while cleaning it with the cloth. Once clean, handle carefully by the edges of the drum.
14. Re-insert the drum into the cartridge. The end with the shaft goes towards the right as you look at the housing facing the handle. Place that end in first by tilting the drum, then inserting the left end to the inner side of the cartridge. Note the spring clip on the left side of the cartridge housing; the clip goes into the inside left of the drum.

continued on page 37
So, you've just installed that new MISOSYS 40 megabyte hard drive. You've got all your applications loaded up and running; everything's set, right? Wrong! You forgot the backup "plan".

On large computing systems, extensive backup plans are used to ensure the safety of user's files. Files need to be protected not only from hardware failures and crackers', but from the user's themselves. It's easy to delete a file accidentally or purge several unnecessary files that become necessary two or three weeks later. I'll use the backup system in place at the State University of New York at Buffalo, which I attend, as an example. This 25,000 student university has computing systems used for administration, research, and other important work on several platforms, including, an IBM 3090 mainframe, UNISYS/Sperry 1100/90 mainframe, a cluster of Digital VAX mini's, and a large UNIX network based on SUN workstations and file servers. Under the backup plan, all new and modified files are backed up daily. This set of backups is kept for 2 months. All files are backed up weekly and kept for 4 months. Once a month, backups of all files are again made and this set of tapes kept for 4 months. Once a month, backups are made and these kept for 32 months. This will also vary from system to system. Generally, I would do them as soon as a new system is configured and running, after installing (and debugging) a major upgrade (operating system, major change in the system set up, etc.) and when all the minor modifications found on the daily backups have added up to major ones since the last full backup.

For full backups I use PowerSoft's BACKREST (available from MISOSYS). This program creates an image of a hard drive partition on multiple floppy's. Other similar utilities are available, take your pick. I first run my MOD3PART/JCL to enable my LDOS partition, then run BACKREST on each of my 5 partitions. BACKREST informs me of the number of disks I need before starting the backup, so if I need more I abort the program and format them (I usually format one more than BACKREST calculates it needs as it is occasionally off by one disk). One problem with BACKREST is that it doesn't reset the modification directory flags, so even though all files have been backed up, the directories indicate otherwise. To correct this, I wrote RESMOD/JCL and RESMODA/JCL. RESMOD/JCL (execute with "do resmod") executes ParmDir to create DOFILE/JCL and then the PARBACKA/JCL file is executed. PARBACKA/JCL backs up all modified files on each hard drive partition. Backup will ask for a new disk if the current one runs out of room (the first PATCH in the JCL modifies BACKUP so that it won't abort if it needs a new destination disk. Normally BACKUP doesn't allow a destination disk change if it was executed from a JCL. The second patch reverses the first). After the partitions are backed up, DOFILE/JCL, created earlier, is executed. DOFILE will backup modified and new files found in every DiskDISK file on the system. Note that PARBACKA/JCL turns on job logging. Now we have a running list of backed up files. Neat! When the JCL is done, mark the disks as modified backups with the date and put them away. Do this as often as you feel is sufficient. If backing up daily seems like too often, do it every couple days, or even once every week. It all depends on how many files are created/updated on your system. Just remember, more backups are better than less.

I usually do full backups about every six months. This will also vary from system to system. Generally, I would do them as soon as a new system is configured and running, after installing (and debugging) a major upgrade (operating system, major change in the system set up, etc.) and when all the minor modifications found on the daily backups have added up to major ones since the last full backup.

For full backups I use PowerSoft's BACKREST (available from MISOSYS). This program creates an image of a hard drive partition on multiple floppy's. Other similar utilities are available, take your pick. I first run my MOD3PART/JCL to enable my LDOS partition, then run BACKREST on each of my 5 partitions. BACKREST informs me of the number of disks I need before starting the backup, so if I need more I abort the program and format them (I usually format one more than BACKREST calculates it needs as it is occasionally off by one disk). One problem with BACKREST is that it doesn't reset the modification directory flags, so even though all files have been backed up, the directories indicate otherwise. To correct this, I wrote RESMOD/JCL and RESMODA/JCL. RESMOD/JCL (execute with "do resmod") executes ParmDir to create DOFILE/JCL which clears mod flags on all DiskDISKs. Then RESMODA/JCL is called which clears mod flags on all partitions and executes DOFILE/JCL. The CLRMOD/CMD program I use to clear the flags is a public domain program I acquired somewhere (CompuServe?). Because I don't have the source code or the author's name, I decided not to include the file. If you need such a program, look around on BBS's or CompuServe. If you happen to own Logical System's FM program, you can use 'FM: d (mod,cm)', where d is the drive number, to do the job.

I would suggest keeping two sets of full backups so that you have "latest full back-
ups” and “next to latest full backups”. I would keep all the modified file backups made between these two full backups until the next full backup. For example, following this system you would have:
Full backup made Dec 10, 1991; modified file backups made between Dec 10, 1991 and June 10, 1992; full backups made June 10, 1992; modified backups made since June 10. Now any file created for over 6 months will be available (if you did enough modified backups). If necessary, keep backups for a longer period of time.

Now, you may have noticed a couple of things about my JCL files. First, they are split into two parts. This is because I use the ‘//include’ JCL command to execute DOFILE/JCL (created by ParmDir). ‘// include’ actually inserts a copy of another file in place of the ‘//include’ when the JCL is compiled by the DO command (DO compiles the file before running it). If I created DOFILE/JCL in the same JCL file as the ‘//include’, it would not be available at compile time. So, why use ‘//include’, why not use ‘do dofile/jcl’? The answer is that I want control to pass back to PARBACKA/JCL after DOFILE/JCL is executed. The only way to do that is use ‘//include. Also notice that I remove DOFILE/JCL in the same JCL file as the ‘//include’, it would not be available at compile time. So, why use ‘//include’, why not use ‘do dofile/jcl’? The answer is that I want control to pass back to PARBACKA/JCL after DOFILE/JCL is executed. The only way to do that is use ‘//include. Also notice that I remove DOFILE/JCL in the same JCL file as the ‘//include’, it would not be available at compile time. So, why use ‘//include’, why not use ‘do dofile/jcl’? The answer is that I want control to pass back to PARBACKA/JCL after DOFILE/JCL is executed. The only way to do that is use ‘//include. Also notice that I remove DOFILE/JCL in the same JCL file as the ‘//include’. Remember that the ‘//include is used in the compilation phase. By the time the REMOVE is executed, DOFILE/JCL has long since been inserted into PARBACKA/JCL.

If you don’t own ParmDir, I would recommend getting it. You may not use it a lot, but when you need it, you’ll be glad you have it. Just the other day I was converting about 10 disks from Multidos to LDOS. Copying the files left garbage in the LDOS directory date fields. Instead of typing ‘reset filespec (date=on)’ for 75 or so files, I typed ‘parmdir :7 dofile/jcl (a="reset ‘x=(date=on)’ once for each disk followed by ‘do dofile’. This helped do what computers are supposed to do, make life easier! (I’ve also included CHECKDIR/JCL and CHEKDIRA/JCL which check every partition’s and DiskDISK’s directory for errors using MISOSYS’ Toolbox/Toolbelt utilities.2 CHECKDIR produces the file ‘checkdir/log’ which contains any errors found).

As a parting suggestion, I would highly recommend getting a 720K floppy drive for your system if for nothing more than backing up your hard drive. MISOSYS has 720K 3.5” drives. If you want 5.25” 720K drives, I have included instructions for configuring Teac 1.2 megabyte drives to work as 720K drives.

### Configuring the Teac FD55GFV

From TRSLink Online Magazine by Gerry Stuteville

This configuration takes the TEAC FD55GFV 1.2 Meg “AT” floppy disk drive and sets it to work with the Western Digital Controller used in the TRS-80 series of computers. It sets the drive as 80 track double side double density unit to give 720 K of storage. The jumpers on the board have to be set up in the following manner in order to work with our controllers. The first jumper has to be installed on the main logic board in Position “LG” this jumper is behind the drive select row pins. When I bought these drives this jumper was not provided with the drive, to jumper this you can either get a .01” space shorting block and place across the pins or solder a piece of wire across. The second jumper that has to be set is in a cluster of four pins set up in a “T” formation behind the “LG” jumper on the main board. These jumpers are labeled as SPEED and have roman numeral markings. You want to jumper the pair of pins marked as roman numeral one “I”. Another way to identify the pins that need to be jumpered is that the pins are the vertical portion of the “T” shaped formation.

Once these pins have been set up set the Drive Select pins to what ever drive position the drive is in. Simply 0, 1, 2, 3 no other pins need to be set. One final note: the drive select does work slightly different on the Models III and IV. If these drives are external to the computer drive 2 would have to be configured on the drive as drive position 0 and drive :3 configured as drive position 1.

After setting your drives up you should now have the 720K storage available to your machine.

To set up your drives on LDOS and LS-DOS, type:

```
SYstem (drive=d, cyl=80) <ENTER>
```

System (drive=2, ENABLE)

After this has been set you can then make it part of your CONFIGURATION by typing:

```
SYstem (sysgen) <ENTER> LDOS
SYsGEN <ENTER> LSDOS
```

Then the final part is to format your disk by typing:

```
format :d (sides=2) <ENTER>
```

This will format the disk both sides after you have answered the other questions in the format routine.

### Using a TEAC FD-55GFR

Courtesy Vince Seifert

The Teac FD-55GFR 5.25inch 1.2M drive can be used as a 720k drive: Put jumpers on the pairs of marks marked LG and HL; they’re at right angles to each other about 2 cm forward of the termination resistor SIP. Also, make sure there are jumpers on the FG posts over by the power connector, and on the RY posts (two of three posts in a right triangle near the termination resistor SIP - the other possible pair of the three is marked DC), and on the appropriate D0 – D3 drive select between the power connector and the termination resistor SIP.
### Window Application Popup

**PRO-WAM**

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:

- **An ADDRESS file** data base prints cards and mailing labels. Throw away that black book and your Rolodex file.
- **HEAD** pipes formatted address data into your letters.
- **BRINGUP** tickler file schedules up to 12 items per day by time. Now print module. Remember those appointments.
- **CALender** gives you a month at a glance; covers 4000 years. Flags days with BRINGUP items.
- **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 keeping a small data base.
- **PHRASE** is a KSM from disk for lots of automation.
- **A telephone list and autoDIALER** for Hayes modems.
- **CALculator** 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. That's an online HELP facility!

PRO-WAM works with all programs which use standard DOS kernel requests and honor the DOS high memory pointer. It 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!

### Z80 Assembler

**EDAS**

This powerful combined disk-based line editor and Z80 macro assembler assembles from one or more nested source disk files or memory buffer; features nested conditionals with ten pseudo-ops, nested 7-level MACROS with parameters both positional and by keyword, cross reference listings; and a separate full screen text editor.

The expression evaluator supports left-to-right evaluation of add/sub/mul/div/1mod/shift, logical AND/OR/XOR/NOT, binary ops EQ/GE/LT/LE/SHL/SHR; unary HIGH/LOW. Labels may be up to 15 characters long; start with A-Z, '@', or ';'; positions 2-15 may also use 'a' and 'q'.

A sorted symbol table listing is available during the assembly. A complete CROSS REFERENCE listing is performed by the XREF utility.

Line edit text in memory and use a command syntax identical to BASIC with block move/copy; with string change/search. Invoke DOS commands within the editor.

If you are writing system software, support software, applications - big or small, EDAS will provide the power to make your job easier, faster, and more worthwhile.

### 8080 to Z80 Translator

**CON80Z**

A source translator to help you convert your Intel 8080 files to Zilog Z80 files. Converts CR-LF sequences to a single CR. By using the CR="c" parameter in the command line, the character "c" will be interpreted as a logical line end.

Translates "M" to "(HL)", extended instructions (LDX); B, D, H, and PSW are changed to BC, DE, HL, and AF; changes <DB/DS/ DW/SET> to <DEFB/DEFS/DEFW/DEFL>.

### Z80 Assembler

**MRAS**

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. Macro support includes REPT, IRP, and IRPC as well as standard macro parameters by both keyword and position. It supports nested includes and a full range of nested conditionals. MRAS incorporates a fast binary-searched symbol table and the ability to enter symbol values from the command line. Labels can be any length with 15-character significance. It has flexible output redirection of listing and symbol table.

MLINK supports virtual memory bit-stream buffering, REL, and IRL library searching, direct generation of complex program overlays, and does not generate disk space for DEFs regions in DSEGs and COMMONs. The linker can generate either a normal executable object file (CMD) or a core image file (CIM). MLINK supports the following special link items: 0-3, 5-7, 9-11, 13-15.

MLIB maintains both relocatable (REL) and Indexed relocatable (IRL) module libraries. You can add, delete, extract, or replace a module; and get module maps.

SAD is an advanced full screen text editor. It can be used to generate your assembler source code, C-language source code, or edit any type of ASCII file. Model 4 128K operation provides easier, faster, and more worthwhile.

### Z80 Disassembler

**DSMRLR**

This disassembler provides extensive capabilities such as direct disassembly from CMD disk files, automatic partitioning of output disk files, data screening for non-code regions, and full label generation. It even generates the ORGs and END statement - the complete ball of wax. You will find that the use of this disassembler even by a beginning assembly language programmer will be paying handsome rewards with the ease of its use and clarity of the documentation. It's a professional tool for your use.

The labeling disassembler produces an assembler source from in-memory code or directly from a CMD-type disk file. Labels are generated for 16-bit references; a reference is any relative instruction target address or a 16-bit target for load, call, jump, add, or subtract instructions.

The disassembler allows you to build a screening data file listing what segments of the program are to be interpreted as data regions. You enter the addresses of the "segments" after analyzing the target program's disassembly.

GRT output is in screen-sized pages. PRINTER output is pagged with column headings, page numbers and titles for print-out that look identical to an assembler listing. Output to DISK produces files suitable for MRAS/EDAS (configurable for others), and is automatically segmented into manageable file sizes. You will even be prompted to change the output file drive when the disk becomes full.

### REL Disassembler

**UNREL**

Here's one of those rare utilities designed for the programmer. UNREL will decode a relocatable object module which has been assembled by either Microsoft's M80 or MISOSYS' MRAS assemblers. The output is an assembler source file compatible with MRAS and M80. UNREL assumes anything in a code segment is code, and anything in a data segment is data. It supports special link items: 0-3, 5-7, 9-11, 13-15.

We bundle in SPLITLIB which can be used to split a library into separate modules. We also include DECODREL, for displaying the bit stream of a REL file. This can be used to more fully understand the actual bit stream.

UNREL should be the perfect professional assembler's tool for your bag of tricks.
### Communications Terminal

**LS-Host/Term**

This communications' package gives you the tools needed to get communications chores done quickly and effectively.

**ADDSS5** is set up to look like a Radio Shack DT-1 emulating an ADDS-25 terminal. Full cursor positioning, reverse video, and blinking fields are supported.

**TERM8** allows one Model 4/4P to be used as a remote terminal to another running HOST portion of LS-Host/Term.

**HOST** lets your 4/4P operate remotely with password access for log-in from another 4/4P using ADDSS5. All video effects are properly transferred to the remote system.

We include a version of XMODEM for file transfer between systems using the MODEM7 protocol, as well as a utility that converts to/from binary and HEX-ASCII binary reper-sentation, to/from INTEL Hex format and checksum files.

### Full C Compiler

**MC**

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 280 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 specifcations, parsing for UNIX **"** extensions in file specifications, overtype support (requires MRAS), a full pre-processor, lots of options, and is designed for the program-mer 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-90 or MRAS, 2 disk drives, and upper/lower case.

### Structured BASIC

**The BASIC Answer**

The BASIC Answer is a text pre-processing utility that allows programmers to generate program code in a structured manner. Source code is created with your text editor. TBA is then used to process this source code into ordinary interpretive BASIC code that uses a minimum of memory. TBA utilizes labels in lieu of line numbers; supports variable names to 14 significant characters; allows the use of pseudo Global and Local variables (local variables retain their value only in a unique subroutine); and introduces the concept of Conditional Translation. This last feature allows co-existence of "machine-dependent" or other variable code within the same program source with the irrelevant sections ignored when processing in source to executable code.

### BASIC Compiler

**EnhComp**

This is an enhanced BASIC compiler released in 1986 and reviewed in the March 1987 issue of 80 Microcomputing and October 1987 issue of COMPUTER SHOPPER. This compiler has lots of great features. It handles the bulk of Modell 3 Micro BASIC and supports additional commands and functions. Standard floating point with both single and double precision functions; random file access ("x" mode for randoms to 32767); turbo graphics, pixel graphics, keyboard array sort, multi-lined functions, user commands, IF-THEN-ELSE, REPEAT-UNTIL, printer control, sequential file positioning, line labels and more.

A supervisor program automates the edit-compile-test phases inherent when using compilers; this makes using EnhComp almost as easy to use as your BASIC interpreter. You also get CED, a line editor with string search/change, partial load/save, renumber, copy, and move.

Enhcomp has a built-in Z80 assembler. You can easily create hybrid programs of BASIC statements and in-line assembly code which completely eliminated compile switched printing and DATA statement high-memory module techniques for your BASIC program to access a machine code module. Z80-MODE accesses BASICS's variables!

You'll have to edit existing BASIC programs, but the power and completeness of EnhComp make that an easy task. Requires our BASIC Reference Manual

### BASIC Sort Utility

**BSORT**

Here's a high speed sort for almost any number of one or two-dimensional BASIC arrays: string, integer, single and double precision. When invoked from your BASIC program, BSORT will perform the indicated sort, and execution will continue with the next statement in your program.

Multiple key arrays may be specified; the sorting on each key can be done in either ascending or descending order. Tag arrays that do not affect the sort, but merely follow along may also be specified. BSORT can also create an integer index array, without affecting the actual order of the elements in the "sorted" array. For string arrays, "matching" parameters allow sorting based on a portion or "substring" of the key array elements. BSORT goes far beyond CMD'O in capabilities and performance.

### Disk Sort Utility

**Disk Sort Merge (DSM)**

A high speed, disk virtual sorting utility that eliminates the burden of sorting from your applications software development project. DSM will create and maintain index files for you. Since the disk is disk virtual, your only limitation is the amount of available disk space, not available memory!

Sorts almost any type of field in a random access file: integer, single and double precision, and strings. Files can have 65,535 records with an LRL up to 1024. Spacily up to 24 select fields (12 for DSM51). Relations (EQ, LT, etc) may be applied to your criteria; operators AND/OR may be used.

Sort ascending or descending; skip records that match a deleted record value. Save a template of the specifications to disk to automate the sort. This allows you to set up a sort operation that is transparent to even a novice user.

DSM is intended for use with user-developed applications software. Please note that DSM creates an index file, as opposed to actually re-ordering the records in the data file.

### Quizzes and Answers

**QuizMaster**

QM is an educational question and answer program that can also be used as a game. It displays a question and four possible answers and scores the operator's response based upon speed and accuracy. QM comes with five subject files of up to 100 questions each derived from grades 6-9 textbooks: U.S. Information; Geography; Math; Genral trivia; as well as Fantasy and Science Fiction trivia.

QM randomizes both the order of the questions and the order of the answers to prevent memorization. The question sequences is never the same. Extended play provides a sudden death mode feature for the skillful user. QM includes all the programs necessary to establish and maintain your own series of multiple choice questions on any subject. Five support programs are provided to create, extend, edit, print, and maintain the question & answer files. All features are easy to use and easy to operate.
MegAdventure is unlike virtually any other interactive fiction adventure that you have ever played, for it will more than just paint its pictures upon the canvas of your imagination - it will stop the sweat right onto your forehead. Block mode allows the manipulation of large text areas. Search and Replace are also provided.

Hex mode allows characters to be input as two hexadecimal digits; makes possible the direct editing of graphics.

**Action Game**

*The Gobbling Box*

This fast-paced action arcade-type game runs on the TRS-80 Model I, III, and 44/P4D. The game generates a variety of special sound effects and music which complement the action on the screen. The arrow keys or Alpha Products joy-stick control the movements of the GOBBLER in this game.

You want your GOBBLER to eat as many dots as possible, while trying to avoid the ZONKERS who won't stop chasing your GOBBLER until one of them eats it or until the GOBBLER eats all dots on the GameBox. The GOBBLER's reward is a new Box; there's no skill. The GOBBLER can tame the ZONKERS for a short while by eating one of the ENERGIZERS on the board. Then it's the GOBBLER's turn to chase, catch and eat the ZONKERS.

The game has two skill levels: the pace is fast; the sound is great; the action is continuous. You can't beat this bargain of a game. Even Stacy plays it!

**Adventure-type Game**

*Lair of the Dragon*

If you thought the TRS-80 was dead, think again. Our Lair of the Dragon MegAdventure is unlike virtually any other interactive fiction adventure that you have ever played, for it will more than just paint its pictures upon the canvas of your imagination - it will stop the sweat right onto your forehead!

If you truly believe that discovery is one of the finest points in life, if you would like to test your ability to think logically to the fullest extent of your ability, if you would like to take on the largest adventure ever written in the genre of interactive fiction, and if you have the guts to face that which would make any other mortal cringe in fear, then Lair of the Dragon is your cup of poison; for reward is a hard-earned commodity here, not given easily to the timid and the faint-hearted. If you are an old hand at adventuring then be pre-pared for a worthy opponent.

MegAdventure rips the door to adventure right off its hinges!

---

**TRS80 Software and Hardware from MISOSYS**

*Text Editor*  
**LED**

A full screen text editor for almost any type of ASCII files, including ASCII program source code for BASIC programs, TBA source, as well as JCL and KSM files. The command menu may be displayed while editing text. This display includes all command keys, the filename, the cursor column, the character hex value and the available memory.

Cursor positioning uses the arrow keys. <CLEAR> key combinations move the cursor to the top, bottom, left or right. Has the following modes: overwrite, insert, insert line, and delete. Block mode allows the manipulation of large text areas. Search and Replace are also provided.

Hex mode allows characters to be input as two hexadecimal digits; makes possible the direct editing of graphics.

*TRS80 6 Source books*

**THE SOURCE, 3-Volume Set**

This will be the last time that these books will be made available for a giveaway price. THE SOURCE contains a vast wealth of information for the assembly language programmer. THESOURCE is not only informative, but also an excellent learning tool.

These books contain the complete, commented assembler source code for TRS80 6.2, excluding hard disk support, the Microsoft BASIC and the HELP utility. Each book is softbound, 6-1/2 by 11. The complete set totals over eleven hundred pages of clearly commented, elegant source code. Volume 1, The System, covers SYS5 to SY8 and SY8 to SY82. Volume 2, The Libraries, covers all of the library commands making up SY58, SY87 and SY89. Volume 3, The Utilities, covers all utilities, drivers, and filters.

*Hard Disk Driver*

**RSHARD**

Finally for your Radio Shack hard disk drive is this hard disk driver package from MISOSYS - at a reasonable price. You get support for both LDOS 5.3 and LS-DOS 6.3.

- RSHARDx driver partitions by both head and/or cylinder; supports two 8-headed drives up to 1024 cylinders.
- RSFORMx formatter adds both low level and high level formatting to your drive's partitions.
- HDCHECK checks the performance of your drive.
- ARCHIVE lets you backup some or all of the files on your hard drive to multiple floppy, BIG files and small files.
- RESTOREx lets you selectively restore some or all archived files to your hard drive.

All ten modules come fully documented and are ready to install into your LDOS 5.3 or LS-DOS 6.3 system (or both).

*Hard Disk sub-partitions*

**diskDISK**

Do you have a hard disk? If so, you need diskDISK. The diskDISK utility allows the creation of logical disk partitions as files on a physical disk drive. This is indispensable for hard disk users. Once a diskDISK file is installed into a logical drive slot, the diskDISK can be used just like any other physical drive; diskDISK provides for easy swapping of any currently active diskDISK file.

With diskDISK, you can easily group related files for ease of maintenance. DiskDISK files can also be set up as images of physical drives to allow mirror image backups.

DiskDISK drives allocate in granule sizes smaller than your hard disk system. Five inch diskDISK images allocate just like floppy drives. Also, there are special diskDISK types that allocate in one or two sector granules for maximum storage efficiency.

*Hard Disk De-fragger*

**HDPACK**

When your hard drive files become fragmented with excessive directory extents, access speed degrades. Your program will finish in less than the optimum time. Now with HDPACK utility, you can restore that ZIP to your computer. HDPACK will automatically, and intelligently, re-pack the fragmented files on your drive which will improve the performance of file access time.

HDPACK provides a visible display of its defragging operation, which in minutes can restore a ten-megabyte directory of files to a minimum number of extents. HDPACK can even work on floppy diskettes, too.

**PUTS ZIP IN YOUR DRIVE**

*Hardware Interface Kit*

**Mod 4 features for Mod III**

This will allow you access to your Model 4 hardware features while using LDOS 5.3. Here's what you get:  
- K14 keyboard driver uses CTRL key, CAPS key, and function keys.  
- SET2RAM switches to Model III RAM mode. @BANK provides bank switching capability.  
- MemDISK provides a one or two bank RAM disk.  
- MemDISK/DCT and the memory management facility.

---

**DoubleDuty doubles your Model 4**

DoubleDuty, published previously by Radio Shack (cat 26-2231), is now available from MISOSYS. DoubleDuty divides your 128K or greater TRS-80 Model 4 computer's memory into three complete and independent partitions. Two partitions each operate as if they were there own 86K Model 4! This lets you run two programs concurrently switching between either at the flick of a function key. It doesn't support multitasking, so only the foreground application receives CPU time. The third partition can be used to execute DOS library commands.

Our new 2.6.0 release also works with expanded memory known to the DOS (such as our XLR8er board). A BANK parameter lets DoubleDuty use any pair of adjacent memory banks. With expanded memory and DoubleDuty, run Scripsit Pro along with other programs. If you thought you needed another computer, think again. With DoubleDuty, you can now have two for the price of one DOS Manual.
TR-80 Software and Hardware from MISOSYS

Model III Utilities

Utility Disk #1

14 utilities useful to novice and experienced LDOS users.

- COMP is a file and/or byte-for-byte comparison utility.
- DCT allows you to view or modify the Drive Code Table.
- DIRCHECK checks the directory on a diskette and corrects most recoverable directory errors.
- MAP displays or prints the allocation (granules or cylinders and sectors) of a file on a diskette.
- RAMTEST is a self-reloading RAM memory test.
- READ40 allows access to a 407 disk in an 807 drive.
- TYPEIN combines the functions of JCL and KSM. Allows programs such as Profile 3+HD to be totally automated.
- UNKILL recovers files accidentally KILLED or PURGED.

FOURTH Compiler

HartFORTH

HartFORTH is a full FORTH that conforms to the 79-STANDARD. The Model III version is an indirect thread ed 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 kernel 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. HartFORTH supports double length integers, string handling, user manipulation, graphs, random numbers, and floating point.

Model III Filters

FILTERS

This combines 23 filters and utilities from FILTER Disks 1 and 2 with assembly source code at a clearance price.

- XLATE translation for I/O devices
- LISTBS print formatting for BASIC pro grams
- STRIP7 removes high-order bit off all characters
- STRIPNT replaces output >127 or <32 with a # symbol
- MONITOR displays control chars in string form (90x)
- TITLE prints a title after form feed
- UPPER converts lower-case character to upper case
- LOWER converts upper case character to lower case
- SLASH translates zero to zero-backspace slash
- TRAP discards any user-defined character
- LINESPACE adds or removes a linefeed after return
- PAGEPAWS pauses after formatted for <ENTER> key
- CALC performs hex/dec/bin conversion; hex add or sub
- REMOVE removes occurrences of a byte from a disk file
- COMM tests for modem carrier
- DICTATE toggles cassette on/off from the keyboard
- DOSPEED regulates output device speed from keyboard
- KMSPLUS features key re-definition on the fly
- LOCATE adds a line number before each line of output
- MARGIN sends a 2-char control before margin spaces
- MAXIMIZE translates one character to a group of chars
- SLOSTEP for drives that require additional settling time
- VIDSAVE saves the current video screen in high memory

Maintenance

GO:MTC

The GO:MTC program collection provides maintenance support services for your computer operation. DIRCHECK performs an integrity check of your disk’s directory and repairs certain kinds of errors; FIXGAT re-constructs a corrupted Granule Allocation Table; IOMAN traps disk input errors; MAPPER checks the granularity of files; RAMTEST performs an exhaustive test of RAM; and UNREPLACE restores a deleted file.

System Enhancement

GO:SYS

The GO:SYS program collection is designed to provide additional features to LS-DOS 6.3 operation. DOCONFIG manipulates CONFIG.SYS files; DDEDIT provides command editing; MEMDIR gets a memory directory; PRD6 provides Partitioned Data Sets; PARMDIR obtains parameterized directory information for listings and JCL processing; WC for wild card command invocation; and ZSHELL for command line I/O redirection, piping, and multiple commands on a line.

More Model 4 Utilities

GO:CMD

The GO:CMD program collection provides additional utility for your computer operation: FASTBACK and FASTREAD for hard disk large file archive/restore; PRO-CESS manipulates command files; COMP compares two files or disks; FED2 reads disk or file sectors on a full-screen basis; IPC to interactively copy, move, rename, delete, and invoke files; ZCAT catalogs 6.3 disks.

Model 4 Utilities

Filters and Utilities for DOS 6.x

- PRCODES gives control of boldface and underlining
- TRAP discards any user-defined character.
- MAXIMIZE is a translation filter system for I/O devices, Doork 1:1 or many: includes EBCDIC and DVORAK cables.
- KMSPLUS improves on the DOS: allows key re-definition on the fly; defines strings for the function keys.
- READ40 allows access to a 407 disk in an 807 drive.
- TYPEIN combines the functions of JCL and KSM. Allows programs such as Profile 4 to be totally automated.

TR-80 Mod I & III GAMES

Lance Micklus’ Greatest Games

This 3-disk set is a great collection as it features space games (Space Trek), adventure games (Dog Star Adventure), gambling games (The Mean Craps Machine, which also includes a Craps tutorial booklet on disk), board games (Mean Checkers Machine), as well as some diam useful programs that you might use for real purposes. Also has some educational games for the kids.

TR-80 Mod I & III GAMES

Leo’s Greatest Hits

Animated Game Disk with sound

This is one of the greatest values in games ever produced. Leo Christopherson wrote the very first animated game for the TRS-80 and the country went wild for it. Android Nim will make everyone laugh to watch these life-like creatures as they shake their heads up and down or side to side and blink at you stupidly as they wait for you to make a move. Then Leo invented how to make the TRS-80 produce sound and added it to Nim. He then followed Android Nim with the other games, even getting Radio Shack to sell Dancing Demons, which is a real scream. The disk contains the famous games: ANDROID NIM, BEEHARY, DUELING DROIDS, DANCING DEMONS, SNAKE EGGS, and ANIMATED LIFE. All games feature full sound effects and some of them are even in 3-part harmony: You and your family will just love this disk! Dancing Demon even features saving your song and dance routines to disk and four of them are included! The possibilities are endless and it is always entertaining. A great way to "show off" what your computer can do and always fun.

TR-80 Mod I & III GAMES

KIM WATT GAME DISK

Contains: Space Colony, Symon, Capture, Horse Race Slots

These are some games that Adventure International published back in the early ’70’s. Originally these were sold on three separate diskettes (or tapes), but we have combined them all on one disk for your collectors.
RS232 Serial Card or Kit

MISOSYS acquired the remaining stock of genuine Aerocomp Serial card kits for your Model II or 4. Polish your broken serial card with this brand new work-alike replacement; or get a kit to install a serial port in the computer without an existing one.

Floppy Disk Controller (FDC)

MISOSYS acquired the remaining stock of genuine Aerocomp floppy disk controller boards, 100% compatible with the original. Replace your broken FDC card with a brand new one. Complete kits with plated steel mounting towers and all necessary cables are available to convert a cassette machine into a disk powerhouse.

Floppy Drives and Parts

5.25" 360K 1/2-height; 3.5" 720K in 5.25" frame; 2SV5 drive case & P/S; Single drive cable; Dual floppy extender cable

Our Model 2SV5 dual vertical external floppy disk drive case will hold two 5.25" half-height disk drives. Needed for one or two drives; ** needed for two drives.

Computer Power Supplies

Astec AC8151-01: 40 -watt supply

Provides +5V @ 2.5A, -12V@2.0A, and -12V@0.1A. Size is 6.25"x4"x1.75"; mounting holes are 3.125"x4.75".

Astec AC12310: 68 -watt supply

Provides +5V @ 7.3A, +12V@2.5A, and -12V@0.1A. Size is 7.69"x1.25"x2"; mounting holes are 3.75"x7.25". Direct replacement for Tandy Model 4 power supply.

XLR8er’d Model 4 in III mode

This is based on our Hardware Interface Kit but is for the XLR8er board operating under DOS 5.3. The package includes: XLR2RAM, XMEmDisk, BANKer, VIDx, and SETx.

XLR2RAM adds a bank handler for the extra 64K of memory and the extended 256K of XLR8er board memory. The RAM disk driver can create a RAM disk of 2to 272 banks; however, only 10 are supported through XLR2RAM. The VIDx expands scroll from 0 to 15 lines rather than just 3; supports reverse video, and supports the @DSP of character codes from 0-31 and 182-255.

Model I/III Data Base Manager

Automation File Manager (AFM)

AFM is a language that you can program your database in! The package consists of three integrated modules that can be called from within each other. AFM - Auto File Manager - is the actual data-base program, AFR - Auto File Reporter - is obviously the module that prints out your reports, and AFU - Auto File Utility - recovers space and keeps the files integrity intact.

A Few is a free-form entry system, which means that you can enter your data in any way you want! You are not limited to a particular screen format. In fact, each record can have its own individual display format! Really! A unique feature is the ability to define the same field repeatedly within a record. AFM's search routines will find every occurrence of a field within a record, no matter how many times it is repeated.

An AFM record can be up to 4096 bytes long (including field name information), but each AFM record takes up only as much space as it needs. If the record must be expanded, then more space is taken up on the disk. AFU will later allow you to recover unused space within an AFN database file. The size of an AFM database file can expand to accommodate a full fifteen megabytes.

AFM allows you to test printer parameters through a FORMS command very similar to the FORMIC command of many popular disk operating systems. This allows you to control your report formats. Totally independent of FORMS, is a COLUMN parameter which is used for defining column widths on display. Thus you have complete control over the formatting of your data for printing and display.

Sorting within AFM is automatic. Once the sort field has been defined, every record entered with that sort field will automatically be placed in the correct position relative to the other records in the file. However, you can list the contents of ANY field in sorted order at any time. In addition, searching can be performed on major key fields through the use of powerful relational operators including ANY, OR, GREATER THAN, LESS THAN, EQUAL, NOT EQUAL, etc.

AFM has a second module included called AFR (Auto File Reporter). It is one of the MOST powerful report generators ever made available for the TRS-80 computer. Fully relational reports can be generated by applying any set of parameters to your data base. AFR also contains a BCD MATH Package which will allow you to have a running total on numeric fields in reports if desired, using +, -, * operators. Accuracy is assured by using BCD math instead of floating-point and output is automatically formatted.

AFM contains numerous HELP screens, which can be invoked at the touch of a key. The help screen is user-controllable and user-definable; you may set up your own help screens for a particular application if you wish. It is very easy to set something up for people who don't use the system all the time.

• Output of AFR is fully sorted by all fields and is fully relational. We can't imagine ANY report that cannot be performed by AFR. You can feed your data each with different file lengths.

• FORM-LETTER output of AFR will allow you to write documents and have the data filled 'in automatically by AFR.

• RELATIONAL LOOKUP report allows multiple passes through the database while generating reports. You can utilize information from ALL NINE drawers in the same report.

• MAIL labels etc. are a cinch with AFR's t-text option which allows selected field output without fudging the information before hand.

• Forms filter parameters include Word Wrap, Page Numbering, Headers, and Vertical Justification.
TRS-80 Software and Hardware from MISOSYS

Floppy Disk Repair Utilities

SUPER UTILITY PLUS
- Reads, repairs and works with all the popular TRS-80 operating systems Models I, III, 41

If you use a TRS-80 with disk drives, then this is a must-have program that you will wonder how you did without for so long! Super Utility is completely menu-driven with the most common defaults built right in. It is configurable for all the popular TRS-80 operating systems and will even allow you to set one drive for one system and another drive for a different operating system and copy files easily between the two. Even between Models I and III or 4, regardless of density, track number, number of sides, or system used.

Super Utility removes or decodes passwords, reformats a disk without erasing the data, fixes problems, backs up most protected disks, etc. Super Utility has over 65 functions and features. Does not work on hard disks. Our ToolBox or ToolBelt has similar features for hard drive use, as well as floppy. Super Utility does not support Newdos/60/double-sided disks.

Super Utility Plus 4/4P/4D

The Model 4 version of Super Utility has all the features of the Model III version, but uses the larger amount of memory for quicker operations, plus utilizes the three function keys. It boots right up in a Model 4P without having to first load the MODELA III ROM file.

Hard Disk Check, Repair, & Modify

LDOS ToolBox
If you own a hard disk and use LDOS, this is the perfect insurance policy for your data. The LDOS TOOLBOX is like a Super Utility+ for hard disks. Features Disk Check and Disk Repair, Sector Modification, plus many, many other useful utilities that makes using a hard drive even easier. Each program contains a built-in help command, so many times you don’t even need to look things up in the manual — just press <Enter> for help! A very wise buy for hard disk users.

Model 4 ToolBelt
This is similar to the LDOS TOOLBOX, except it is for the Model 4 TRSDOS 6 operating system (all versions).

Super Fast Hard Disk Backup and Restore

Back/Rest
BACK/REST has proven to be a great time-saver for thousands of TRS-80 hard drive users. BACK/REST can back up 10 megabytes in about 10 minutes and 20 meg in about 30-40 minutes. It also tells you how many disks are needed. Works under LDOS or TRSDOS 6 (both versions on same disk). Great utility for hard disk users!

Hard Disk Drivers for Tandy disk systems

Supreme HD Driver - RS
These hard disk drivers out-perform the Tandy drivers in many ways. Our Powersoft drivers allow you to combine LDOS and TRSDOS on the same drive and boot from either system (with floppy disk). They run faster and take much less memory from the system. Only for use with Tandy Hard Drives.

SuperSCRIPSIT Printer Driver

PowerDriver Plus
- Allows EPSCN or compatible printers to be fully utilized with SuperScript and SCRIPSIT PRO.

This is a replacement driver for the ones you got with SuperScript. It fully supports the various Epson and Epson compatible printers to the limits of their capabilities. Model I, III or 4 is supported in the same package. Easy to install.

A Major Enhancement for SCRIPSIT 4, III and I

PowerSCRIPT
This modification for Radio Shack's SCRIPSIT program turns it into a POWERHOUSE! Your program merges with your copy of SCRIPSIT to create a new program! PowerScript allows you to add printer control codes directly in the body of your text! Now it is easy to add underlining, bold face, the different sizes of print, etc. Initially set up for the EPSON type dot-matrix printers, it is configurable just to about any printer during set-up. If you have more than one printer type, then just set up a copy of PowerScript for each printer you have. PowerScript adds the ability to see an alphabetized directory without exiting the program seeing how much free space you have, and others. It works on the Model I, III or 4 versions of SCRIPSIT. It will even make a Model I version of SCRIPSIT work on a Model III or 4 (in the III mode).

PowerDraw
PowerDraw allows you to create graphics (mixed with text if desired) and save them to disk. It allows you to create up to 99 "frames" of animation and "play" them like a movie. It allows you to save the graphics in several modes, including BASIC listings, CMD file format, etc. These can then be merged into your programs, either in BASIC or machine language. Many of PowerSoft's opening screens were created with PowerDraw. In fact, it even creates animated opening screens to really pep up the program. It also allows you to print the screens on Epson-type and several other types of printers.

PowerMail Plus
This program was because all the other mailing list/database systems couldn’t keep track of all the types of data most folks wanted to keep track of. You needed speed, you needed hard drive support, and you needed a crash-proof data structure. PowerMail+ was top-rated (5 stars) in several publications and has never been topped. Works on floppy or hard disk under all popular TRS-80 operating systems. Allows importing of data from several other once popular mailing systems to avoid re-entering data. Written in machine language by the author of PowerSync, this program is FAST and sorts up to 10 levels very quickly. If you keep track of names and addresses along with associated data for any situation, this is the one to use. Many churches, organizations and businesses use PowerMail+ for all the different kinds of lists they need to pull from. Each record has 24 user-definable "flags" to allow total customization for your exact needs.

Create customized "form letters" and Labels with PowerMAIL+. This optional module for PowerMail allows you to create customized "form letters" or custom labels, lists, etc. with PowerMail Plus and any word processor that saves text in ASCII format. Very easy to use and really gets the effect you want. Allows completely definable report generating from your PowerMail+ data.
Choose LDOS 5.3.1 or LS-DOS 6.3.1

Both Model I and Model III LDOS support similar commands: DOS commands are virtually similar to Model 4 LS-DOS 6.3.1 syntax where possible.

The DATE command, "Date?" prompt on boot, and the @DATE: SVC now support a date range of 32 years; from January 1, 1980 through December 31, 2011.

Enable or disable the printer time-out and error generation with SYSTEM (PRTIME=ON/OFF).

Both ASCII and hexadecimal display output from the LIST command is paged a screen at a time. Or run it non-stop under your control.

MEMORY displays (or prints) the status of switchable memory banks known to the DOS, as well as a map of modules resident in I/O driver system memory and high memory.

Specify SYSTEM (DRIVE=d1,SWAP=d2) to switch drive d1 for d2. Either may be the system drive, and a Job Control Language file may be active on either of the swapped drives.

The TED text editor has commands to print the entire text buffer, or the contents of the first block encountered. Directories from TED, too!

Have extended memory known to the DOS? The SPOOL command now permits the BANK parameter entry to range from 0-30 instead of 0-7.

Alter the logical record length of a file with "RESET filespec (LRL=n)"

Specify "RESET filespec (DATE=OFF)" to restore a file's directory entry to the old-style dating of pre-6.3 release. Specify "RESET filespec (DATE=ON)" to establish a file's directory date as that of the current system date and time.

SYSTEM command supports removable and reusable BLINK, ALIVE, and UPDATE memory modules.

Double-density BOOT support for Model I with embedded SOLE and FORMAT (SYSTEM). Supports mirror-image backup, too. Reworked DUBL driver eliminates PDUBL and RDUBL and takes less memory; enhanced resident driver eliminates TWOSIDE.

Model III version auto-detects Model 4 for installation of K14 keyboard driver; supports CAPS, CTRL, and function keys.

SPOOL command offers Pause, Resume, and Clear parameters. (OFF) attempts to reclaim memory used.

Both Model I and Model III support similar commands: all features of Model III 5.3.0 are in Model I 5.3.1. That includes such facilities as DOS and BASIC help files, SETCOM and FORMS library commands, TED text editor, BASIC enhancements, etc. All DOS commands have been groomed for Model 4 LS-DOS 6.3.1 syntax where possible.

Felt uncomfortable with the alleged protection scheme of 6.3? LS-DOS 6.3.1 has no anti-piracy protection! Neither does LDOS 5.3.1. MISOSYS trusts its customers to honor our copyrights.

Best of all, a 5.3.1 or a 6.3.1 diskette is available as a replacement for your 5.3.0 or 6.3.0 diskette for $15 (plus $3 S&H in US and Canada, $4 elsewhere). There's no need to return your current master.

The 5.3.1 or 6.3.1 diskette(s) come(s) with a 30-day warranty; written customer support is available for 30 days from the purchase date. Versions of 5.3.1 for the Model I and Model III are available. Versions of 6.3.1 for the Model 4 and Model II/12 are available; Model 4 French and German versions are also available (specify 6.3.1 F or 6.3.1 D). Some Model I 5.3.1 features require lower case or DDEN adaptor.

DOS and BASIC Reference Manuals

Two new reference manuals are available from MISOSYS. First, we have the the 349-page "LDOS™ & LS-DOS™ Reference Manual", catalog number M-40-060. This single manual fully-documents both LDOS 5.3.1 and LS-DOS 6.3.1 in a convenient 8.5" by 5.5" format. If you use one, or the other, or even both DOS versions, you may want to bring yourself up to date with a single manual. Gone are the many pages of update documentation. Price is $30 plus $5 S&H.

We also publish the "LDOS™ & LS-DOS™ BASIC Reference Manual". This 344-page book, catalog M-40-061, covers the interpreter BASIC which is bundled with LDOS 5.3.1 (even the ROM BASIC portion), the interpreter BASIC which is bundled with LS-DOS 6.3.1, and both Model I/III-mode and Model 4-mode EnhComp compiler BASIC. One convenient 8.5" by 5.5" manual covers all four BASIC implementations for $25 plus $3.50 S&H. Since this new manual covers our compiler BASIC, you can purchase the disk version of EnhComp for $23.98 plus $1.50 S&H when purchased along with a BASIC Reference Manual, or the disk version by itself for $29.98 plus $3 S&H if purchased separately.

MISOSYS, Inc.
P. O. Box 239
Sterling, VA 20167-0239
703-450-4181
[orders to 800-MISOSYS (647-6797)]
Let LB Data Manager solve your data storage problems

LB Version 2.3: Modern up-to-date features provide this newest release of our Flat File Data Manager with a greater degree of flexibility and an increased level of ease-of-use. LB still provides strong database capabilities with absolutely no user programming!

NOW WITH COLOR SUPPORT FOR PC USERS

We've added many features asked for over the past few years by LB users; yet LB is still about the easiest, most flexible data manager you can use for managing your data. Absolutely no programming is needed to create a database with up to sixty-four fields; construct input screens for adding, viewing, and editing data; and create your own customized report. Quickly you define your data fields in response to LB's prompts, and then draw your data input screen using simple keystrokes - or have LB automatically create your input screen. In no time at all, you're entering data. Customize your printed reports with user-definable print screen definitions. Or use LB's define print autogen module to automatically create Table or Form reports, or over a dozen different address label configurations including a Rolodex™ card and a 3" by 5" index card. LB is just what you need in a data manager! We even have many database templates available for download on our CompuServe forum!

Data capacity per database:
LB supports up to 65,534 records per data base; 1,024 characters (64 fields) per record; and up to 254 characters per field.

Field types supported:
LB allows ten field types for flexibility: alphabetic [A-Z, a-z], calculated (operations on "numeric" fields using +, -, *, /, with 2-level of parentheses), date last modified [YYYY/MM/DD automatically maintained], dollar (±dddddddd.dd), floating point (±dddddddddddd.ddddddd, literal (any ASCII character), numeric [0-9,-,], right-justified numeric, upper case alphabetic [A-Z, automatic conversion of a-z], and upper case literal (literal with automatic conversion of a-z). All field types utilize input editing verification so invalid data cannot be added to a record. Field name strings can be up to 19 characters long.

Data entry and editing:
LB allows you to design up to ten different add/update view screens to provide extreme flexibility for selectively viewing your database fields. You can customize the appearance of any view screen by a simple drawing process, or use LB's built-in autogen capability. View screen definition even provides an intelligent line-drawing mode so you can create lines and boxes to enhance the appearance of your screen image. If your computer supports a color video adaptor, each view screen can be configured to a distinct foreground/background color arrangement to increase the distinction of it's data viewing.

Using a database password provides the capability of selectively protecting fields from being displayed or printed without entry of the correct database password, or they can be protected from being altered. This is quite useful in a work-group environment. Fields may be selectively established to require a data entry before a record being added or edited is saved. You can enable a special index file to keep track of records being added. This can be subsequently used, for example, for a special mailing to newly added customers. Flexible editing includes global search and replace with wild-card character match and source string substitution. Search and replace can be performed on all records, or on records referenced in an unsorted or sorted index file.
Record selection and sorting:

You can maintain up to ten different index files to keep your data organized per your multiple specifications. Records may be selected for reference in an index file by search criteria using six different field comparisons: EQ, NE, GT, GE, LT, and LE. You can select on up to eight different fields with AND and OR connectives. Index files can be left unsorted, or you can sort in ascending or descending order. By associating a sorted index file, any record can be found within seconds - even in a very large database.

LB even includes a special Dup command which uses the sophisticated Ratcliff/Obershalp pattern recognition algorithm for automatically finding duplicate or near-duplicate records! Duplicates can then be either manually deleted or automatically purged using the provided LBMANAGE utility program.

Automatic operation:

For automating your processing needs, LB can be run in an automatic mode, without operator intervention. Frequently used procedures can be saved by LB's built-in macro recorder for future use. Entire job streams may be produced, so that LB operations may be intermixed with literally any DOS function that can be batch processed. These named procedures are easily invoked via a pop-up list-box.

Maintenance utilities:

To make it easy for you to grow your database as your data needs grow, we provide three utility programs for managing your database. LBREDEF allows you to construct a new database with an altered data structure and populate it with data from your existing database. This facility is great for adding new fields, or deleting fields no longer needed. Or you can use LBREDEF to redefine the field type of an existing field and convert the existing data. Another utility, LB MANAGE, allows you to duplicate your database structure, copy or move records from one to another, or automatically purge un-needed records.

A third utility, LBCONV, converts to LB from pfsFILE4, Profile4, DIF, dBASE II, dBASE III, and fixed record. It also converts to DIF, dBASE, and tab or comma delimited files to enable easy porting of LB data to other systems.

Report generation:

Report generation incorporates a great degree of flexibility. Your report presentation can be totally customized through print definition formats which you define on the screen as easily as you define the add/update view screens. You can truncate field data, strip trailing spaces, or tab to a column. You control exactly where you want each field to appear on your report. LB provides for a report header complete with database statistics: database name, date, time, and page numbers. A report footer provides subtotaling, totaling, and averaging for calculated, dollar, floating point, and numeric fields; print number of records printed per page and per report.

Many report formats can be automatically created by LB's define print autogen module. You specify your printer type and character size from a pop-up list-box. Select one of four canned report formats: narrow or wide carriage Table reports; a Form report; or an address mailing label using one of six different sizes of labels including labels printed two, three, and four across. Label formats also include formatting for a Rolodex™ card and a 3" by 5" index card. Label formats automatically select the needed fields from your database definition.

For printing, associate any of the ten index files and you control exactly what records get printed; even a subset of indexed records can be selected for printing to give you a means of recovering from that printer jam halfway through your 30-page printout. You can even force a new page when the key field of an index file changes value. Up to ten different printout definition formats can be maintained for each database. Reports may be sent easily to a printer, the console display screen, or to a disk file - useful for subsequent printing or downstream data export to other programs. Report formatting allows for multiple across mailing labels, multiple copies of the same record, or even Form printing one record per page for sales books. You can easily generate mail/merge files of address or other data for your wordprocessor. Or you can use LB's built-in form letter capability.

Help is on the way:

The main menu even provides a shell to DOS so you can temporarily exit LB to perform other DOS commands. LB provides extensive on-line help available from almost every sub-command. A200-page User Manual documents every facet of LB's operation.

Competitive Trade-up policy:

Send in an original Table of Contents page from any existing database program and get LB Version 2 for half price. That's only $49.50 + S&H!

Ordering Instructions

Specify MS-DOS (and media size) or TRS-80/4 version. LB is priced at $99 + $5 S&H US ($6 Canada; $7 Europe; $9 Asia, Pacific Rim, and Australia).

MISOSYS, Inc.
PO Box 239
Sterling, VA 20167-0239
703-450-4181 or orders to 800-MISOSYS
TRSCROSS™
(Pronounced TRISS-CROSS)

TRSCROSS runs on your PC or compatible, yet reads your TRS-80 diskettes! Copy files in either direction!

The FASTEST and EASIEST file transfer and conversion program for moving files off the TRS-80™ and over to MS-DOS (or PC-DOS) or back

TRSCROSS™
Copyright 1986, 1987 by MISOSYS, Inc.
All rights reserved

1 - Copy from TRS-80 diskette
2 - Copy to TRS-80 diskette
3 - Format TRS-80 diskette
4 - Purge TRS-80 diskette
5 - Display directory (PC or TRS-80)
6 - Exit

Shown above is the Main Menu displayed when running TRSCROSS on your PC or compatible.

TRSCROSS is as easy to use as it looks to be! The program is very straightforward, well thought out, and simple to operate. TRSCROSS has several "help" features built into the program to keep operation as easy as possible. Just pop your TRS-80 disk into your PC and copy the files right to your PC data disk or hard disk. It can't be any faster or easier! All steps are detailed in the instruction manual. Advanced features, for those that desire to use them include executing menu options right from DOS or from a batch file or macro. This can really speed up transfers when similar operations are performed frequently.

TRSCROSS allows you to "TAG" all files to be moved in ONE pass!

TRSCROSS converts TRS-80 BASIC programs and SuperSCRIPSIIT files in ONE PASS while COPYING to MS-DOS!

No need to save your programs or files in ASCII or run a separate conversion program first before transferring. TRSCROSS reads your tokenized BASIC program or SuperSCRIPSIIT files directly off your TRS-80 disk and performs the conversion all in ONE pass while being transferred directly to your PC or compatible computer. Automatically converts most BASIC syntax, and lines that need special attention can be listed to a printer. (Does not convert PEEKs, POKEs, graphics, machine language calls or subroutines.)

TRSCROSS will even FORMAT a TRS-80 disk right on your PC! (Handy for those who use both machines!) Former TRS-80 users who no longer have their TRS-80, but still have diskettes with valuable data... this is exactly what you've been waiting for!

TRSCROSS will READ FROM and COPY to the following
TRS-80 double-density formats:
TRSDOS 1.2/1.3, TRSDOS 6.2*, LDOS 5.3*, DOSPLUS, NEWDOS/80*, & MultiDOS.

DOS formats listed above flagged with * signify that earlier versions of these DOS's are readable as well, but one or more sectors may be skipped due to a format problem in that version of the DOS. (Disks that were formatted with SUPER UTILITY™ or SUW4/4PM™ do not have this problem.) TRSDOS 6.02.01, or higher should not have this problem. Disks formatted in any 5.25" 80 track format, or single density are not supported; 3.5" 720K disks are readable in a 720K 3.5" MSDOS disk drive.

TRSCROSS Requires: PC or compatible computer, 128K and a normal 360KB (40 track) PC or 1.2MB (80 track) AT drive. Double-sided operation is fully supported. If you have more than one disk drive, fixed drive, or RAM disk, operation will be much smoother. TANDY 1000 requires more than 128KB memory (DMA). TANDY 2000 is not supported at this time due to a difference in disk controller and floppy drives. "Special" data files (like PROFILE+) would need to be converted to ASCII on a TRS-80 first before they would be of use on a PC or compatible.

If you use both types of computers, or you plan to retire your TRS-80, this is for you. TRSCROSS will allow access to your TRS-80 diskettes for years to come. Copy your TRS-80 word processor data files as well as your Visicalc data files over to MS-DOS and continue using them with your new application.

Only $89.95

Plus $4 S&H (U.S.) or $5 Canada or $6 Foreign
Virginia Residents must add appropriate sales tax.

MISOSYS, Inc.
P.O. Box 239
Sterling, VA 20167-0239
Phone: 703-450-4181 (Orders only: 800-MISOSYS)
Super Utility™ for MS-DOS™

A powerful file recovery and disk media utility for floppies, internal fixed, and many external fixed and cartridge storage devices.

For IBM PC™, XT™, AT™ and many other PC compatibles including COMPAQ™, AT&T™, and TANDY 1000, 1200, 2000, and 3000.

MS-DOS users... your wait is over for a GREAT new disk utility! Accidental deletions and disk directory problems can happen to anyone at anytime... a power spike, fingerprint, speck of dust, hardware problem, or simply typing DEL in the wrong sub-directory can destroy critical data in a moment without warning. It's gone. Therefore we introduce Super Utility™ for MS-DOS an easy to use disk utility containing many functions sorely needed in today's MS-DOS computing environment.

With Super Utility™ you can restore damaged or deleted files using two different methods (one easy, the other a little tougher). Even clusters of an erased file assigned to another file can still be restored, unless the user has physically written over every byte of the original data (especially useful in word processing files).

In addition to file repair and recovery, Super Utility™ provides sector verify, sector editing, modification of sectors in Hex or ASCII, ease of renaming of files and setting their attributes, string search, copying sectors to a file, diagnostic sector checking, mapping of the FAT table of a file or an entire drive, visual graphics pertaining to your system, and full directory and sub-directory editing without endless menu-hopping - all in one program. The sector display mode displays all 512 bytes on-screen at one time and allows you to fully explore your disks. SEARCH and CHANGE are nice here! Compatible with DOS versions 2.0 - 3.1 on most systems. Some computers may require the use of PC-DOS. Color, composite, or monochrome video are supported. A great tool for fixed disk users as well as floppy. Also compatible with IOMEGA's Bernoulli Box storage device (also sold by Tandy as the Disk Cartridge System).

Super Utility™ is a program that fills the gaps that PC users have most need of and have asked us for. It's aimed at the beginner, the "office user", hobbiests, students of the PC, and programmers alike.

So, why not be certain about the safety of your data (and your peace of mind) when you can have Super Utility disk insurance right on hand at an unusually low price? Make your new computing life easier, more fun, and knowledgeable all at the same time. Super Utility is easy to use, unprotected, and great insurance against disaster if and when it strikes!

MISOSYS Special
Only $29.95

MISOSYS, Inc.
P. O. Box 239
Sterling, VA 21067-0239
Phone: 703-450-4181 Phone Orders Only: 800-MISOSYS (800-647-6797)

Add $5 S&H in U.S.; Add $6 S&H in Canada & Mexico; Add $9 S&H all others. Virginia residents add 4.5% Sales Tax. VISA and MasterCard accepted. Checks not drawn on an American bank are not accepted.

Requirements: IBM PC or compatible running PC/MS-DOS 2.x-3.x, min. 128K memory, one disk drive. PC-DOS may be required for use.

IBM, PC, XT, and AT are registered trademarks of International Business Machines Corp. MS is a registered trademark of Microsoft.
More Bytes, Less Bucks.

Now you have a choice when selecting a backup system from Colorado Memory Systems, the company that sets the standard for value in data storage.

More Bytes-Jumbo 250
With 250 Megabytes of storage at $499 SRP ($265 at MISOSYS) you can save more while spending less. Jumbo 250 delivers superior performance while using all applicable QIC industry standards for format and data compression.

Less Bucks-Jumbo 120
At $399 SRP ($199 at MISOSYS) Jumbo 120 continues to provide jumbo performance for peanuts. With 120 Megabytes and QIC compatibility, Jumbo 120 is the backup of choice when less storage is needed.

Jumbo 250 and Jumbo 120
Both work with IBM® PC, XT, AT, 386™, 486, PS/2™ and compatibles and the price includes software that will let you backup nights and weekends with an unattended scheduler. Network compatibility is standard, you can also get optional Xenix®/UNIX® software offering an easy to use menu system, another first from Colorado Memory Systems. While both drives use the standard floppy controller, a variety of optional jumperless controller cards offer easy to install performance enhancements.

Available from:
MISOSYS, Inc.
PO Box 239
Sterling, VA 20167-0239
800-MISOSYS
With a 20 or 40 MB MISOSYS Hard Drive connected to your TRS-80 Model III or 4, your computer will sail through data access.

MISOSYS has been shipping complete drive kit packages since September 1989 which plug into Model 4/4P/4D and Model III computers; let us build one up for you! Our host adaptor, which interfaces the 50-pin expansion port of the TRS-80 (host) to the 50-pin SCSI port of the HDC, sports a hardware real time clock option using a DS1287 clock module. With its internal battery lifetime in excess of 10 years, never enter date and time again. It even adjusts for daylight saving time! Another option available is a joystick port and Kraft MAZEMASTER joystick with a port interface identical to the old Alpha Products joystick; thus, any software which operated from that joystick will operate from this one.

Software supporting the S1421 and 4010A controllers includes: a low level formatter; an installation utility and driver; a high level formatter; a sub-disk partitioning utility; utilities to archive/restore the hard disk files onto/from floppy diskettes; a utility to park the drive's read/write head; a utility to set or read the hardware clock; a keyboard filter which allows the optional joystick to generate five keycodes; and a utility to change the joystick filter's generated "keystroke" values after installation. Optional LDOS 5.3 software is available.

Twenty megabyte drive packages are currently built with a Seagate ST225 hard drive; Forty megabyte packages use a Seagate ST251-1 28 millisecond drive. Drive packages are offered as 'pre-assembled kits'. Your 'kit' will be assembled to order and fully tested; all you will need to do is plug it in and install the software. Drive kits include a 50-pin host interface cable and the hardware clock. Full implement of status lights included: power, ready, select, read, and write. Add a joystick or hardware clock for but $20 additional per option (see price schedule).

Aerocomp Hard Drives now available from MISOSYS
MISOSYS is also the sole source of remaining brand new Aerocomp hard drives. All Aerocomp drives include status LEDs, software driver and formatter, power and host cables, and installation Job Control Language. We are building their 20M and 40M drives. We also have Montezuma Micro CP/M Hard Disk Drive drivers available.

Order any hard drive kit or unit from MISOSYS and we'll pre-install either LS-DOS 6.3.1 or LDOS 5.3.1 at no extra charge.