THE MISOSYS QUARTERLY

Look at what is in this issue:

- ☞ Add an internal 40MB hard drive to your 4P,
  by Roy Soltoff
- ☞ How to make a Model 4 FILTOMAT,
  by Christopher Fara
- ☞ The "C" Language,
  by Earl C. Terwilliger
- ☞ An Environment for MC and LDOS/LS-DOS,
  by Richard N. Deglin and Roy Soltoff

The "Sea" Language

Volume VII.ii **$10** **spring19/93**

CLOSEOUT PRICE LIST - effective March 1, 1993

TRS-80 Software (Items on Closeout: NO RETURN)

Product Nomenclature	Mod III	Mod 4	Price S&H
AFM: Auto File Manager data base	P-50-310	n/a	$10.00 D
BackRest for hard drives	P-12-244	P-12-244	$10.00
BSORT / BSORT4	L-32-200	L-32-210	$5.00
CP/M (MM) Hard Disk Drivers(drive specific)		**H-MM-???**	**$10.00 B**
CON80Z / PRO-CON80Z.	M-30-033	M-31-033	$5.00
diskDISK / LS-diskDISK	L-35-211	L-35-212	$10.00
DoubleDuty		M-02-231	$25.00
DSM51 / DSM4	L-35-204	L-35-205	$10.00
DSMBLR / PRO-DUCE	M-30-053	M-31-053	$10.00
EDAS / PRO-CREATE	M-20-082	M-21-082	$10.00 D
Filters: Combined I & II	L-32-053	n/a	$5.00 B
GO:Maintenance	n/a	M-33-100	$15.00 B
GO:System Enhancement	n/a	M-33-200	$15.00 B
GO:Utility	n/a	M-33-300	$15.00 B
Hardware Interface Kit	n/a	M-12-110	$5.00
HartFORTH/PRO-HartFORTH	M-20-071	M-21-071	$10.00 B
LDOS 5.3.1 Mod1 Upgrade kit	M-10-133	n/a	$20.00 B
LDOS 5.3.1 Mod3 Upgrade Kit	M-10-333	same	$20.00 B
LS-DOS 6.3.1 Upgrade Kit - M4	n/a	M-11-043	$20.00 B
LS-DOS 6.3.1 Upgrade kit - M2/12/16		M-11-002	$25.00 B
LED / LS-LED	L-30-020	L-30-021	$5.00
LS-Host/Term	n/a	L-35-281	$10.00
LS-UTILITY	n/a	L-32-150	$10.00
MC / PRO-MC	M-20-064	M-21-064	$79.95 D
Mister ED	n/a	M-51-028	$10.00 B
MRAS / PRO-MRAS	M-20-083	M-21-083	$30.00 D
PowerDot (Epson or Tandy)	P-32-21?	n/a	$5.00
PowerDraw	P-32-220	n/a	$5.00
PowerDriver Plus (Epson).	P-50-200	P-50-200	$5.00
PowerMail Plus	P-50-003	P-50-004	$15.00 D
PowerMail Plus TextMerge	P-50-100	P-50-100	$5.00
PowerScript	P-50-142	P-50-142	$10.00 B
PRO-WAM	n/a	M-51-025	$50.00 D
PRO-WAM Toolkit	n/a	M-51-225	$15.00 B
QuizMaster	L-51-500	n/a	$5.00
RATFOR-M4		M-21-073	$10.00 D
SuperUtilityPlus	P-32-132	P-32-104	$15.00 D
Supreme HD Driver (PowerSoft-RS)	P-12-113	P-12-113	$15.00
TBA / LS-TBA	L-21-010	L-21-011	$5.00 D
THE SOURCE 3-Volume Set	sold out		
Toolbox/Toolbelt	P-32-203	P-32-245	$10.00 B
UNREL-T80	same	M-30-054	$5.00
UTILITY-I	L-32-070	n/a	$5.00
XLR8er Software Interface Kit (M3 mode)		M-12-11X	$5.00 B

TRS-80 Software

DISK NOTES from TMQ (per issue)			$10.00
HDPACK: Disk De-fragger	**n/a**	**M-33-400**	**$29.95**
LB Data Manager-M4 (Ver 2.3)	n/a	M-50-510	$99.00 D
LDOS/LSDOS Reference Manual	M-40-060	M-40-060	$30.00 D
LDOS/LSDOS BASIC Reference Manual	M-40-061	M-40-061	$25.00 D
LDOS 5.3.1 Diskette - M1	M-10-110	n/a	$15.00
LDOS 5.3.1 Diskette - M3	M-10-130		$15.00
LS-DOS 6.3.1 Diskette - M4	n/a	M-11-243	$15.00
EnhComp / PRO-EnhComp Diskette	M-20-072	M-21-072	$23.98
RSHARD - R/S HD driver	M-12-013	M-12-013	$15.00
SSCSI - SCSI driver for H-HD-MHA	H-HD-SWS	H-HD-SWS	$25.00 B

MSDOS Software ("*" Indicates Closeout: NO RETURN)

LB Data Manager 2.3	M-86-510	$99.00 D
DED-86 [Disk/Memory sector editor]	M-86-020	$29.95 D
*RATFOR-86	M-86-073	$10.00 D
*HartFORTH-86	M-86-071	$10.00 D
*SAID-86 [Text Editor]	M-86-040	$15.00
Super Utility PC	P-86-407	$29.95 B
TRSCROSS (transfer <> Mod III/4	P-86-212	$89.95 B
*FM-86 (File Manager)	L-86-050	$10.00
*Lair of the Dragon	M-86-021	$10.00

TRS-80 Game Programs (Items on Closeout: NO RETURN)

Cornsoft Group Game Disk: Bouncezoids, Crazy Painter, Frogger, Scarfman, Space Castle (M3)	M-55-GCA	$20.00
Kim Watt's Hits (M3)	P-55-GKW	$10.00
Lair of the Dragon (M3/M4)	M-55-021	$10.00
Lance Miklus' Hits (M3)	P-55-GLM	$15.00
Leo Cristopherson's (M3)	P-55-GLC	$10.00
The Gobbling Box (M3/M4)	M-55-020	$10.00

MSDOS Game Programs

Lair of the Dragon	M-86-021	$10.00

Hardware ("*" Indicates Closeout: NO RETURN)

*Power Supply, 40WT Astec AC8151	H-PS-A40	$40.00	D
*Power Supply, 68WT Astec AA12310	H-PS-A68	$50.00	D
*Floppy Disk Controller M3/M4	H-MM-FDC	$35.00	F
*Double Density Controller (DDC) M1	H-MM-DDC	$40.00	F
*RS232 Serial Card M3/M4	H-MM-SPC	$40.00	F
*RS232 Serial Card Kit M3/M4	H-MM-SPK	$45.00	F
*TeleTrends TT512P modem (M4P)	H-4P-512	$49.95	E
Floppy drives (5.25" 360K 1/2 ht)	H-FD-360	$75.00	D
Floppy drives (3.5" 720K 1/2 ht)	H-FD-720	$85.00	B
*Floppy Drive Case (2-1/2 ht drives)	H-FD-2SV	$30.00	F
MSCSI HD, 20Meg M3/M4	H-HD-020	$395.00	?
MSCSI HD, 40Meg M3/M4	H-HD-040	$495.00	?
MSCSI Hard Drive joystick port option	H-HD-JSO	$20.00	
MSCSI Hard Drive hardware clock option	H-HD-RTC	$20.00	
Aerocomp HD - 20 Meg M3/M4	H-MM-020	$350.00	?
Aerocomp HD - 40 Meg M3/M4	H-MM-040	$450.00	?
Hard drive: Seagate ST225 (20M)	R-HD-020	$200.00	G
Hard drive: Seagate ST251-1 (40M)	R-HD-040	$250.00	G
Hard drive: Seagate ST-351AX (IDE)	R-HD-I40	$195.00	G
*Cable: dual floppy extender	H-FD-2EX	$10.00	
Cable: 4Ft floppy (1 34EDC each end)	H-FD-C04	$12.50	
*Cable: 4Ft M3/M4 printer	H-RC-PM4	$20.00	
Cable: 4Ft Radio Shack hard drive	H-HD-CT4	$20.00	
Cable: 4Ft MISOSYS hard drive	H-HD-C04	$20.00	
Cable: 26-1069 internal floppy	H-FD-2NG	$20.00	
Cable: 26-1069A/26-1080 internal floppy	H-FD-2GA	$20.00	
Cable: 26-1080/A internal floppy	H-FD-24P	$20.00	
*Standby Power System: 200VA	sold out		
*HD Controller: Adaptec 4010A	H-HD-CA4	$45.00	D
*HD Controller: Xebec S1421A	H-HD-CX2	$45.00	D
*HD Controller: WD1002S-SHD	H-HD-CW2	$45.00	D
T80 to SCSI host adaptor	H-HD-MHA	$75.00	D
PC-Compatible Hardware			
ZOFAX 96/24 Fax/Modem (PC XT/AT)	R-Z1-FAX	$125.00	G
*Infochip Systems Expanz! (PC)	R-IC-EXP	$99.00	G
DJ10 Tape Backup (PC)	R-TD-D10	$199.00	G
DJ20 Tape Backup (PC)	R-TD-D20	$265.00	G
AB11 Tape Adaptor (PC)	R-TD-A11	$45.00	D
KE10 External tape adaptor/case (PC)	R-TD-K10	$110.00	F
Tadiran TL-5296 AT 6V lithium battery	R-PB-TL6	$19.95	B

The Fine Print

Freight codes: A = $3.50; B = $4.00; C = $4.50; D = $5.00; E = $5.50; F = $6.00; G = $7.00; H = $12.00; ? = varies; **All unmarked are $3.00** each; Canada/Mexico add $1 per order; Foreign use US rates times 3 for air shipment. Virginia residents add 4.5% sales tax. We accept MasterCard and VISA; Checks must be drawn on a US bank. COD's are cash, money order, or certified check; add $5 for COD.

MISOSYS, Inc.
P.O. Box 239, Sterling, VA 20167-0239
703-450-4181

The MISOSYS Quarterly is a publication of MISOSYS, Inc., PO Box 239, Sterling, VA 20167-0239, 703-450-4181.

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 (renewal for two issues is 1/2 cost shown):

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 Premium/386 computer (20 MHz) with 9 Megabytes of RAM, a Seagate ST4096 80M HD, ST251 40M, Expanz! card; a CMS DJ10 tape backup, a NEC Multisync 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.

Table of Contents

List of Advertisors

List of Patches in this Issue

TMQ Index

With the introduction of the TMQ Index, articles and other material which has appeared in past issues of TMQ can be ordered as a re-print. The index lists the number of pages associated with each re-print; a standard fee of $1 per page will prevail. By the time you read this, thes index will be available in three forms:

- A printed copy - $10 + $4S&H;
- An LB Database file set on floppy disk - $10 + $3S&H;
- An LB Database file set available for download from our CompuServe Forum (as long as the forum remains available).

Upcoming at MISOSYS

Over the past six months, I have been working off and on at porting the LB86 Database Manager to the Tandy DeskMate environment (that's the MS-DOS version). Version 3.3 of the DM development system has been used; however, version 3.5 has been acquired to continue the effort. I am currently targeting a release of the DM version of LBDM by summer.

Now Tandy has discontinued all DeskMate development for at least a year; their emphasis currently is on multimedia under Microsoft Windows. If you keep your eye on the PC marketplace, you should easily see that Windows as taken over like the Blizzard of 93. In any event, an LBDM² has been planned for quite some time and I wanted to complete that as a steppingstone to a Windows version of LB.

The Blurb

by Roy Soltoff

From my observation, DeskMate was modeled as a chareacter-based version of Windows. So I feel that a good grasp of development under DM will aid in grasping the intracies of Windows. If readers feel that they would be interested in reading of my DeskMate development experiences, give a shout.

Points to Ponder

Here's help for free-lance programmers: A new Programmers Registry has been established by Solutions Software Corporation to create a database identifying part-time at-home and full-time programmers. The Registry will contain information on the types of programming services that these individuals can offer on a part-time and/or full -time basis to clients. Retirees, students, homemakers and others with the spare time and the skills available to providing programming services are asked to request a free Listing Questionnaire. The Solutions BBS may be accessed at no charge by calling 407-321-6119 (8-N-l). Go to File Library and download QUEST.ASC. The completed listing may be uploaded to the BBS or via mail or fax. For more information, contact: Solutions Software Corporation, 1511 Kastner Place, Sanford, FL 32771; 407-321-7912; Fax 407-321-3098 or 407-323-4898.

Remember the Randy Cook copyright message accessible in Model I's TRSDOS 1 by holding down certain keys during booting? Well according to ShadowRAM's column in Computer Reseller News, the developer's list of names for Microsoft's Access can be obtained by the following weird procedure: *Create an Access table, save it as Cirrus. Select Help About, hold down the control-shift, put the cursor on the key icon and click the right mouse button twice. You'll see ducks sitting on a pond. The idyll is shattered when two lightening bolts annihilate the ducks. Paradox - get it? Then you get the developers.*

With the downturn in the world military market, many defense industrials are being forced to rapidly migrate to consumer products. Programmers, who seem to survive on Jolt cola and pizza, may soon be able to get that pizza as quick as a Pentium processor. Texas Instruments Inc.'s Defense Systems & Electronics Group have been hard at work trying to perfect an oven that will cook restaurant-quality pizza in about a minute. The ovens were designed by Turbo-Chef president Philip R. McKee, who said, "Forging into areas that previously were not considered high-tech or defense oriented will become increasingly necessary as this country attempts to reap the peace dividend and pull in the reins on defense spending."

Texas Instruments is in the news from another new angle - angling mirrors. It recently demonstrated live video based on its digital micro-mirror (DMD) technology. The DMDs, which are micro-mechanical spatial light modulators, are aluminum mirrors about 17 microns wide, attached to poles at two ends, and suspended over static RAM. The mirrors tilt 10º to the right or left, depending on whether the SRAM cell under the mirror is holding a 1 or a 0. A large 640 x 480 array of the mirrors is assembled which provide pixel resolution of a VGA screen in a projector device. In the projection system, light is beamed through a

condensor lens, and then through a red-green-blue filter wheel which is synchronized with the video information fed to the DMD chip. The filtered light is then projected onto the DMD chip, whose mirrors are deformed according to the digital information that has been written to the chip's SRAM. The image is reflected from the mirrors through a projection lens onto a large screen. TI has plans for DMD arrays up to 2,048 x 1,152 mirrors for use in HDTV systems.

Video compression schemes are hot! According to *Electronic Engineering Times*, a bold breakthrough was demonstrated via closed-circuit TV at this winter's Consumer Electronics Show by video startup, 3DO. The 3DO interactive multiplayer system "took a still image, pasted it onto the visible faces of a cube, then rotated the cube. It then made the cube transparent, so that the still image could be seen on all six faces as the cube rotated. It then bounced the rotating cube around the screen, distorting it to make it appear to be a graphics-laden block of Jell-O." They "also took a live feed from a camcorder, then manipulated 15-frames-per-second video, squeezing, twisting, peeling and stretching multiple representations of a 3DO employee's face. Next, the machine presented an image of a revolving globe made of 120 facets, and sent it bouncing around inside a three-dimensional room; as this was going on, it orbited a graphic representation of a light bulb around the globe, rendering the appropriate shading in real time. The architecture capable of performing such feats at consumer prices is an indication of what is to come."

"The core of the system is a pair of animation-cel engines working in parallel, each a complex graphics pipeline, capable of 3-D transformation, shading and lighting, transparency control and texture mapping. The engines can manipulate a theoretical peak of 64 million pixels a second, according to 3DO president and CEO Trip Hawkins.

Lining up behind 3DO are many of the world's largest electronics and entertainment companies including: AT&T, Panasonic, Time-Warner, MCA, and Electronic Arts

And for those 3D-fans out there who are tired of the red-green glasses, *Electronic Buyers' News* reports that Reveo Inc. has a technology for recording 3D color-image data in one image, called a spatially multiplexed image (SMI). The technology is known as multi-mode stereoscopic imaging (MMSI), and it produces 3- color images using only one display device, such as a projector, a video monitor, or even a printer. The user demultiplexes the image using passive glasses. Underlying MMSI is a device called a micropolarizer (µPOL), invented by Dr. Sadeg Faris, Reveo's founder. The µPOL is a polymer sheet with an orthogonally polarized checkerboard pattern. The µPOL, placed over the CCD (charge-coupled device) imaging array in a camera, for example, modulates the pixel image the camera is seeing into left and right pixels; the camera has circuitry that does the modulation into left and right images. Multiplexing circuitry then combines the modulated images into one - the SMI. The displayed full-color image can be viewed in 3-D using a pair of passive glasses equipped with a µPOL, or directly when a µPOL is laminated over a printed image. Reveo has built a 7-pound flat-panel overhead viewing screen priced at $7,995 that can display 260,000 colors; competing devices cost upwards of $30,000.

My points to Ponder column seems to always have something to say about hard disk drives. That's because drive technology is zipping by. This issue is no different. Western Digital's new *Caviar* drive is claimed to be the first to market to pack 212.5 megabytes into each platter of a 2-platter 3.5" sub-13ms seek time drive. That's 425 MB for an OEM single-unit price of $700.

Meanwhile, 42.6 MB are being stored per platter on a 2-platter 85.3MB 1.8" drive by Aura Associates. The drive, destined for lap-tops, withstands an operating shock of 10g.

Integral Peripherals, who introduced the world's first 1.8" drive well over a year ago, is up to 85MB in their Cobra model - which incidentally withstands an operating shock of 100g. At $475 in small quantities, the drive may prove attractive for PCMCIA-compatible peripherals [don't ask about the mnemonic - it has to get changed so it can be pronounced!].

Stepping up in size, Toshiba is sampling a 2.5" drive with a capacity of 340 megabytes at 12ms average seek and an PCAT or SCSI interface. TMQ's columns are a hair less than 2.5 inches from hairline to hairline.

If you want to store gigabytes, Micropolis has a 5.25" full-height 8-platter drive which stores 3.6 gigabytes. That company rings nostalgic for me; my first floppy drive pack was made by Micropolis. It consisted of two 35-track one-sided floppies in a beautiful blue case. I haven't used it in years, but it was so pretty, that it may never be discarded. Two gigabytes are also available in a 3.5" drive from Hewlett-Packard

Blue lasers jump out again in the news [see TMQ VII.i, page 4]. IBM, looking for ways to stem the tide of red ink, is now prototyping a high-density optical storage device using a new blue laser capable of producing 50 mW of 428-nm blue light. Since the diameter of the blue light is capable of being focused to one-half the diameter of previously used infrared light, storage density should quadruple. Coherent, Inc. has licensed the technology hoping to produce a commercial product in about a year.

High-cost optical drives also appear in pleasant packages. Ricoh Corp has recently released a recordable compact disk (CD-R in lieu of CD-ROM). At an average access of 500ms and 150 kilobytes/second data transfer, its no speed demon. But disks recorded on it can be played back on standard CD-ROMs. For those who need the capability, its yours for a little over $4,000.

Now for an environmental pickup, Toshiba Corp has developed a highly efficient, non-polluting method of turning plastics into fuel oil. One industry observer in Japan said, "Turning mixed, chloride-based products into oil is the modern equivalent of turning lead into gold." It is estimated that Japan currently generates 5.6 million tons of plastic wastes a year, approximately 86% of thermoplastic suitable for the process. With the process claimed to turn 1 kilogram of plastics into one liter of gasoline, kerosene, or a mixture of the two, [get your calculators out for this one], Japan alone could recover 11.54 billion gallons of fuel per year.

Fuzzy logic has come a long way since I first made mention of it a few years ago. For those really interested, there's a bulletin board maintained by Motorola which specializes in fuzzy logic. Try 512-891-3733 at 300-9600 baud, 8,N,1.

Did you know that Smith-Corona is the only domestic supplier of portable electric typewriters - not that they're manufactured here in America. SC's now required to pay a dumping duty of 16% on typewriters it imports from its production facility in Singapore. Who's that Russian comedian with the famous line, 'America, what a country?' Brother International, a Japanese company has moved their typewriter production to Tennessee while our American Smith-Corona moves production to Asia!

According to *Electronic Products*, the Oak Ridge National Laboratory with joint research by Eveready, has developed a thin-film battery which can be fabricated on semiconductor processing equipment to be included directly into an IC such as a computer memory chip.

According to *Electronic Engineering Times*, Sarnoff Research Center, after ten years of research, has spun off a unit to market their "Smart Sensing" vision technology. At the heart of the operation is a pyramid image-processing chip named the PYR-1. The goal is to create low-cost board level products which can be plugged into plain vanilla PCs to enable real-time computer recognition of objects which are in motion..

And Computer Reseller News reports that HP will shortly be releasing a 3-pound notebook computer with Microsoft Windows 3.1, DOS 5.0, Winword, and Excel - all in ROM! Nicknamed the *Lion*, the portable will have two PCMCIA slots, one of which will support HP's Kittyhawk personal storage module. Pricing? How's $1,500-$2,000?

Now for the truly bizarre, the Hitachi Cambridge Laboratory and the Cavendish Lab at Cambridge have demonstrated a memory device structure utilizing the Couloumb Blockade effect they believe will eventually be used to store a bit of information with just one electron! This was reported in *Electronic Engineering Times* of February 22nd - not April 1st! To put this into perspective, current 16-Mbit memories use about 500,000 electrons to store one bit. Where 100 m^2 of space and 10 kW of power would be required to store one terabits of memory using 16M-bit devices, a C-B device would need but one square inch of space and 0.1 W.

Finally, the production of 386-based computers is grinding to a halt. With Intel dropping the 386DX chip, and X86 customer demand of 90-95% for 486-based systems, expect to see few 386 machines by fall 93.

Trade-in Policy

With the closeout of most TRS-80 products, our trade-in policy exists solely for our LB database and remaining MSDOS-related products. The policy, where applicable, is to just send in an original *Table of Contents* page from an equivalent non-MISOSYS software product 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...

The long-awaited re-print of Earl Terwilliger's six-part series on the C language finally appears in this issue. This C introduction should help put the language into perspective. As the original series terminated prematurely after the 6th part because the LSI Journal folded, perhaps I may just continue the series next time with new material on data structures and functions.

Also sitting in the TMQ input box for a few years has been Rich Deglin's implement of an environment for LDOS and LS-DOS. Primarily designed for MC, Rich put together stand-alone DOS commands which could access the environment. All the source code is on Disk Notes; thus, non-C-based programmers could introduce the environment access to their programs - if they so choose. Rich did the code, and I put together a short piece of text to present the concept of an environment.

Chris Fara, of Microdex, also begins exposure of his expertise in TMQ. Look for his article demonstrating a different slant on filter installation..

Finally, in the last issue of TMQ, I piqued your hard drive interest by my reference to the upcoming SCSI driver. Well it's complete, and this issue provides complete specifications on how I turned my Model 4P into a 40 megabyte machine. With its XLR8er speed and 40M capacity, all I need do now is find a way to add a hires graphics board...

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 "93/03" complete their subscription with this issue. The renewal fee to continue with another four issues is covered on page 1.

Some folks have asked me if TMQ is going to continue past issue VII.iV. Right now there are about 260 subscribers - 40 or so non-US. To mail at 3rd class bulk rate for US subscribers, it requires a minimum of 200 pieces. Each annual renewal has resulted in a 20% reduction in subscribers. That trend implies that Volume VIII would not have a sufficient base to continue 3rd-class mailing. So unless something changes to project a reversal of the downward subscription trend, consider that TMQ will cease publication with issue VII.iv.

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 GO LDOS. 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. MSDOS 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 7.2

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 7.2 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.**

LB Templates

Please note the availability of the following LB database templates:

LB Template Disk 1

DRA	Dragon magazine article index
GAMEINV	Role playing game inventory
LEAP	Parent group address roster
PROP	Valuable property record
PTA	PTA roster
STAMPS	Stamp collection
STREK	Star Trek collection inventory
VID	Video Tape and Laser disk library

LB Template Disk 2

AUD	Catalog of audio disk/tape collection
COMPUTER	Catalog of owned computer equipment
CREATURE	Catalog of adventure game creatures
LIB	Library card catalog
MAILFILE	Address mail list / LB database example
MISOSYS	Customer information database

To use any template, simply copy the files to your data drive, create a path file using LB menu option 14, then add your data. To create a template for others, simply use LBMANAGE to duplicate your database, then copy the new set of files to another disk. Submit your templates to MISOSYS for publication. They are available at $10 per disk + $3S&H, or free for download from our CompuServe forum. An MS-DOS 360K disk will hold a pair of template disks.

DOS Manuals

Don't forget that with our "LDOS™ & LS-DOS™ BASIC Reference Manual", which 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, you can purchase the disk version of EnhComp for $23.98 plus $3 S&H. If ordering the EnhComp disk, please note which version: Model I/III or Model 4!

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.

SCSI Driver

MISOSYS has a SCSI driver, H-HD-SWS, now available for use with our H-HD-MHA Model III/4 host adaptor. The SWS driver is for directly supporting a Seagate SCSI drive or exact equivalent; it can handle a drive up to eight heads and 1226 cylinders (approximately 80 megabytes). Seagate drives which are in this capacity range include the 48MB ST157N, the 60MB ST177N, and the 84MB ST1096N. These drives are out of production, however, re-furbished drives should be available at reasonable prices (one source is listed in the 40M4P article in this issue). Drivers for both Model III and Model 4 modes are included.

FAX Number

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

Closeouts

MISOSYS continues the closeout of most of our TRS-80 software products. The closeout products listed in this pricelist will no longer be available from MISOSYS after June 15th 1993. Note that no warranties, returns, or support are 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 color 2000 e/w stand $150
- Tandy 1000 & mono monitor $125
- TRS-80 Model III (working) $35
- TRS-80 Model III (working) $35
- Tandy DT-1 Data Terminal $25
- 25-1061 3.5" External floppy for Tandy 1000EX (new) $50
- Amdek Video-300 Monitor $40
- BMC Monitor (for MAX or M1) $25
- 12 Meg Secondary (bad drive) $50
- DW-II Daisywheel (26-1158) $50
- Line Printer III (no printhead) $25
- Radio Shack Modem II $15
- DW-II Tractor Feed (26-1446) $10

Used Software

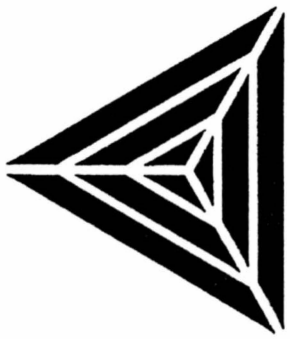
The following items of used software packages are available for purchase. These are items accepted as trade-ins or otherwise accumulated:

- Radio Shack C, 26-2230 $15
- pfsFILE, Model 4 26-1518 $15
- pfsFILE, Model 3 26-1515 $15
- Series I EDTASM I/III 26-2013 $10
- Radio Shack ALDS 26-2012 $15
- Profile 3+ 26-1592 $15
- Compiler BASIC (III) 26-2204 $15
- ZEN EDTASM $5
- Quikpro+ $5
- ZBASIC 2.21 $5
- Level I BASIC Instruction Course $5
- Sargon II (cassette) $5
- Interlude (Model I) $5
- Gambiet 80 (mod-I tape) $5
- Macro-Mon (Model III disk) $10
- Personal Finance 26-1602 $10
- Blackjack/Backgammon (mint) $5
- Microchess 1.5 26-1901 $5

Binders

I have one carton of MISOSYS 8.5" 3-ring binders; these are the 1" ones we used for our small User Manuals. First come first served at **three for $10** + $5S&H (continental U.S. only).

Letters to MISOSYS



LDOS 5.3.1 Type Ahead

Fm LE, Arlington,TX: Roy, I still have trouble with the type ahead buffer in 5.3.1.

```
10 A$=INKEY$:IF A$ =""THEN 10
20 PRINT A$;:FOR I=1 TO 5000:NEXT
30 A=A+1:IF A<6 THEN GOTO 10
```

If you run this program and hit any key 5 times very quickly, it should print that key five times. The program produces the desired effect with a Model 4 and 5.3.0. It also works with the KI4/DRV and 5.3.x.

Fm AM: Roy, I am convinced that Type Ahead on LDOS 5.3.1 does not work. I am loosing characters when I type at a normal typing speed. This does not happen using LDOS 5.3.0 with either its own KI/DVR or using KI4/DVR. Type ahead will work on LDOS 5.3.1 using KI4/DVR. All of this is on a Model 4 running in Model III mode. Since I discovered this, I have been testing LDOS 5.3.1 on my Model I's, and Type Ahead DOES work fine on the Model I with LDOS 5.3.1. Apparantly, there is some call within KI/DVR that is simply bypassing the designed function. Would you please look at this?

Fm MISOSYS, Inc: Okay guys, you made your point. It involves a change made with the release of the 5.3.1 KI/DVR. There was a short piece of code which was conditionalized FALSE for the Model III version but TRUE for the Model I version. I have re-assembled KI/DVR and have it available on the Disk Notes accompanying this issue of *The MISOSYS Quarterly*. I'll also make it available for download on CompuServe. By the way, it's not easily patched.

SAID and LDOS 5.3.1

Fm DJK, Richmond, VA: For some time I have been using my SAID program with its built-in keyboard driver on my Model III. The documentation suggests that I should be able to use KI/DVR that is bundled with LDOS 5.3; but everytime I install SAID and by-pass the built in driver, I get a "Can't find KI/DVR" message even though the driver is loaded and active - at least as far as I can tell. I have attached a short Job Log printout to demonstrate the problem.

My SAID program came bundled with my EDAS program (Version 4.3.a, Serial #821280). The documentation update history notes that the disk/program was upgraded to SAID 1.1 on 2/10/86, and patch SAIDIN51/FIX was applied to SAIDINS/CMD on 4/17/85. No other patches or upgrades pertaining to SAID are mentioned in the README/TXT file.

I would really like to use the KI/DVR with SAID rather than having to reset *KI before using it. (Nasty things happen with KI/DVR resident even with type ahead off.) TED has no problem recognizing the presence/absence of KI/DVR. Is something wrong or am I missing something.

Fm MISOSYS, Inc: The problem is one of syncronization: The LDOS 5.3.1 keyboard driver was updated subsequently to last implementation of SAID. SAID can't find the LDOS 5.3.1 KI driver because of a number of reasons. First, the module name of the 5.3 driver was "KI3" or "KI1" for the Model 1 5.1.4 version; the 5.3.1 driver has a module name of "$KI".

Next, SAID needs to index the special character conversion table to change the code for the up arrow key from 5BH to 0BH; the table was moved in release 5.3.1.

The solution to the problem is to patch SAID to introduce the search changes necessary to find the $KI driver of 5.3.1. The following patch should work for Model III operation:

```
. Patch to Model III SAID
1.1
. Allows finding $KI
driver
D2A,35="I";F2A,35="3'
D2B,38="$KI":F2B,38="KI1"
D2A,C8=A1 FE;F2AC8=E7 FF
. Eof
```

DIRCHECK & BACKUP

Fm HAB, Jacksonville, FL: Dear Mr. Soltoff: By now you probably have had several, queries from readers of *Computer News 80* in which David Goben said in the February 1993 issue that your DIRCHECK is dangerous to use because it frees locked-out granules that should have remained locked out. As I am an owner of DIRCHECK, I'd like to know your response. I could see this as being a problem with floppies, which give me far more problems with locked-out granules than your hard drive ever has. Could this have contributed to my abort-and-trashing complaint that you have never been able to duplicate? I have found that it has freed up hard-drive granules that never should have been locked out in the first place; the only problems I have ever had with your hard drive since I bought it in 1989 have been the result of directory glitches.

Second, Goben says BACKUP has been flawed for years, but that it never mattered until it encountered subdirectories created by his software. It seems that the problem has to do with the drive number register being properly set rather than ignored. I haven't bought his subdirectory software, so it doesn't really matter to me. But someday I might, so I thought I'd ask.

Third, there was a post on Roy Beck's BBS in L.A. that you are getting out of the TRS-80 business altogether and selling off your inventory at close-out prices. If so, I have received no such mailing. If you are leaving the TRS-80 world, I'm concerned about continued support for my MISOSYS hard drive and related software, Naturally, I don't want you to leave, but if you must, I trust there will be sort of formal announcement, coupled with suggestions as to where people like me can turn for support

Fm MISOSYS, Inc: By now I have had only one query concerning Goben's CN80 piece - yours. [How many readers do they have?] Since I have not read what he wrote, I cannot comment on that.

To my knowledge, DIRCHECK does not free up locked out granules on floppies which were locked out as flawed. In fact, DIRCHECK warns you if a file is allocated to a granule which is locked out due to a flaw.

In the case of hard drives, there is no lock-out table. What has been common in the TRS-80 community of hard drives, is to allocate a flawed granule during an integrity check of the drive partition. But since this is done in the single allocation table, there is no way for any utility or the DOS to subsequently know that the granule is flawed compared to a corrupted directory with a granule allocated but not used. DIRCHECK will indeed deallocate any granule on a hard drive which is unused but allocated - however DIRCHECK prompts you to make sure of your intended action.

The manual states, "Hard drive directories differ from those found in floppy drives in that they have no lock out table for flawed cylinders during formatting". The manual further states, "It is not recommended to use a diskette with a flawed directory. ... recover files by copying to a known good diskette." What I would recommend is to (1) save a copy of the FREE space map immediately after formatting a hard drive so that you are aware of any "bad" granules. (2) Next, if you have a problem and use DIRCHECK to attempt repair, you know the state of granules allocated because of defects. (3) Lastly, you should run HDCHECK (LOCK) after using DIRCHECK if granules allocated-for-defects were de-allocated. HDCHECK is provided with some of our hard disk driver packages.

Now with the diskDISK subdirectory facility available for $10, why would anyone need someone else's? DiskDISK has been available since 1983 (that's a good 10 years) and BACKUP has never had a problem with DiskDISK's sub-disks! Perhaps Goben needs to make sure that he writes software which works correctly with the DOS? Or when he comes upon a problem, he could report it - I find it a difficult job to fix problems which remain un-reported!

Finally, concerning Beck's BBS post, MISOSYS is closing out most of its TRS-80 software. This was announced in the TMQ VII.i issue. You're a subscriber. The issue was mailed in early January. Had you not received it by the time of your letter (February 9)? There will be no further support of software being closed out. One can only speculate on other items. If we continue to stay in business, you have a source; if we close down, it is no different than thousands of other business establishments which have closed down.

Fm HAB, Jacksonville, FL: First, I want to say that, although received the winter issue of TMQ, I didn't pick up on the significance of what you were doing with the closeout prices. I hope that, in some way, you will stay with us TRS-80 users as long as you can, especially people like me who bought your hard drive and related software, and who at other times have purchased some of your fine utilities.

I can add more detail to what David Goben was saying now, having read his warning against your DIRCHECK a second and then a third time: He was saying that your DIRCHECK inadvertently makes granules within the directory available for

accepting files. Here's the way he put it in CN80:

When a hard disk is formatted, the unallocated granules on a directory cylinder are locked out so that files can not store data on that cylinder. This is the proper thing to do...

The DIRCHECK program supplied with the HDPACK optimizer will note if granules are locked out on the directory cylinder, and if it finds any it will then turn around and free them. What this means is that a file can later have this freed space allocated to it, and this spells trouble!

I have difficulty believing this would happen, as your DIRCHECK recognizes when it is being asked to work on a hard drive, and I have to specify the H option if I want it to proceed. Further, when I looked at my partition directory areas, I found no residue of files that don't belong there. So if I'm right, he's maligned your product. But if he's wrong ...

On the TRS-MOD134 national BBS echo, I have encountered hard drive owners who didn't know about your DiskDISK and SubDisk and thereby were unable to utilize all of the storage space if the directory slots were full. I've told them about MISOSYS' solution. This shows you that I'll be in your corner as long as you have a corner. And I hope you'll continue to reserve a corner for us.

Fm MISOSYS, Inc: I guess I will just have to disagree with Goben. DIRCHECK will indeed de-allocate any granule either not allocated to a file or belonging to a cylinder speciified with the "H" parameter. However, a properly constructed directory DIR/SYS entry will show that an entire cylinder of granules is allocated; thus, when DIRCHECK analyzes such a directory, it should find all granules of the directory cylinder allocated to the directory. If you look on page 169 of *THE SOURCE*, Volume 3 - The Utilities, you will find that lines 5350 through 5400 are initializing the number of granules allocated to DIR/SYS; this number is the total number of granules on the cylinder. DIRCHECK cannot possibly de-allocate the unused allocated space beyond that actually used by the directory under these circumstances! I also examined the code in my other hard disk driver implementations, MSCSI and RSHARD. Both construct the DIR/SYS entry in a similar manner.

Okay, so how could Goben come up with this statement? One way could be if he has written a hard disk formatter which does not properly construct the directory entry for DIR/SYS. Another way could be if he has done something which corrupted the DIR/SYS entry. In the past, I have heard of folks corrupting the DIR/SYS entry by opening it up as a file, then writing to it using standard DOS file access service calls. Such methods are neither documented nor proper.

Finally, I would offer this suggestion: if someone believes they have a problem with one of my products, the most sensible thing to do would be to report the problem. Publishing what may turn out to be erroneous statements in *CN80* about someone elses product without any prior notice or report just shows the lack of respect some folks stoop to.

Product Queries

Fm RLH, Albion, NY: I was just rereading the March 1988 CN80 where they said that you were getting out of the TR8-80 market. I know that you had to get into MSDOS to continue eating but I'm very thankful that you didn't completely abandon us.

I'm using MSDOS 2.11, 3.3 & 4.0. I like some of the MSDOS software but the DOS still hasn't caught up to LDOS which we have had for more than a decade. At least once a week, I am reminded of an LDOS feature that I took for granted (when it is impossible or difficult to do with MSDOS).

I just bought LDOS 5.3.1 for the Model I that I use to replace the SOLEd LDOS 5.1.3/5 that was used on both Model Is. I sent a check for two 5.3.0s for the Model I when it came out for the III. I used MultiDOS 2.1 for a while until I got LDOS patched to 5.1.5. I still use it occasionally because of a couple of utilities (ZAP, VFU & FMAP) that I'm not aware that exists for LDOS. I have and use FED/CMD. I'm looking for a disk zapper to handle the double sided drives. One of the features that I would like is to be able to format & verify a single cylinder. Does MISOSYS have one or did I mss it. I also have enclosed a printout of FMAP. Do you have something like that. MAP/CMD doesn't tell me what file is in specific granule. If a disk crashes (very rare today), I could look for that file on another disk or archive copy.

The Model 4 is my primary machine using LDOS 5.3.0 90% of the time, LS-DOS about 8%, MultiDOS 1% and other DOSes 1%.

I have wanted DOTWRITER for years which isn't available. I heard that PowerDot is something like it. Where was it reviewed or where can I get more information on its use and potential? I read about GRASP that you sold years ago. Could this be used to enhance my Model I/III/4 printing? Do you still sell it?

Fm MISOSYS, Inc: As Mark Twain once said, "*the report of my death was an exaggeration*". I don't know how that rumor of me getting out of the TRS-80 market started circulating. Perhaps that's what they wished!

The exact capability of FMAP is available for the Model 4 with our GO:MTC product [M-33-100]; at it's closeout price of $15 + S&H, it's a steal because it also includes DIRCHECK, FIXGAT, IOMON, RAMTEST, and UNREMOVE.

To format and verify a single cylinder, your best bet is Super Utility (where have you been all these years). SU should allow you to also re-format a floppy disk without losing any data.

GRASP was discontinued moons ago. PowerDOT is a drawing tool using block graphics for display but can print in graphics mode to certain printers (mostly old ones but Epson compatibility keeps you current).

sscanf() bug in MC

Fm Richard VanHouten: TEST1/CCC and TEST1/CMD illustrate the bug in sscanf(). As you can tell by running TEST1, it locks up when trying to scan a 2 word string using "%s %s %s". I disassembled sscanf and found that it passes pointers to 2 subroutines to the main scan function, one for getting characters from the buffer and the other from ungetting them. I found that the get subroutine returns 0x0d when it finds end of string, instead of 0xffff (EOF) as documented in the manual. When sscanf() is processing the space (skip whitespace), the 0x0d is considered whitespace and the program locks up trying to get past EOF. I have tried this with and without #include <math.h> and got identical results.

TEST2/CCC and TEST2/CMD illustrate the bug in frnd(). frnd() is documented as returning a random float between 0 and 1 if passed a (float) 0, and to return a random float between 1 and the passed value if passed a float between 1 and 32767. I have not tried to trace this one down.

Fm MISOSYS, Inc: Looks like we have a trade-off. You were right about the sscanf() function. Checking back to the source code, the code does not match up with the comments. I have included a revised copy of sscanf/rel on this disk of yours. You need to replace the copy in your libraries (LIBC/REL and MATH/REL). Disk NOTES will also include the relevant files.

Since you have to deal with LIBC, which is a biggie, you may need to split it into two pieces. I provided a utility to do this a long time ago; it is called "splitlib". A copy of it is also included. I am giving you a copy of the text page from the TMQ issue. Split rel libraries can be re-combined with the DOS APPEND file1 file2 (STRIP) command.

As far as the frnd() problem you were having, it is caused by your attempt to use a single precision function in double precision mode. The compiler "float" switch is quite clear on this aspect on page 3-7 where the "+f" switch is documented. It also refers you to chapter 5 where on pages 5-5 and 5-6, the matter is discussed in extreme detail. What was happening was the float argument was actually being converted to a double, but the function was expecting a single; the value used by the function being the zeroed part of the number; hence your result.

I edited your test2/ccc program to cast the value returned from frnd to a double, then re-compiled the test program using the "+f" compiler option and it worked as advertised. I suppose in hindsight, those functions specifically supported only in single precision could have had a note in their function documentation. Sorry for the confusion.

```
#include <math.h>
main()
{
      int x;
      printf("\x1c\x1fThis
should print 20 random numbers
between 0 and 1\n");
      for(x=0;x<20;++x)
            printf("%f
", (double)frnd((float)0));
      printf("\nThis should
print 20 random numbers between
1 and 10\n");
      for(x=0;x<20;++x)
            printf("%f
", (double)frnd((float)10));
}
```

PRO-MC dint() fix

Fm FS: I now have a fast assembly language fix to the limitations of the present dint() in PRO-MC. It requires replacing the FDFIX module in LIBA. However, when I load liba/rel into MLIB, I get a "symbol table overflow" error. Is there any way to replace this module without resorting to the kludge of putting the new FDFIX in a USERLIB?

Fm MISOSYS, Inc: You would need to split the library in two (use splitlib which was given in a past TMQ). Then adjust the half containing the revised module. Then append the two modules with the DOS APPEND command (use STRIP).

As an aside, the STRIP parm of APPEND was put in specifically to append REL modules - that was its exact purpose. It, of course, could also have been used to append two SCRIPSIT documents by allowing you to strip off the NULL from the first one.

LOBO MAX80 HD Boot

Fm BP: Hello, Roy! I have the LOBO MAX80 hard drive working, although not bootable, as mentioned in my other messages. I'll give that a fresh try shortly.

But I have been using much of my old MISOSYS on the LOBO, including PDS and Elsie. Having a bit of fun porting K&R utilities, like SORT, just for the refresher in C language.

Do you have any kind of list of software that is known to work on the LOBO? Of course, I refer to your own products, as shown on your price list. I'm most interested in programming tools, like DSMBLR. (My copy expects a tape).

Also, for my Model 4P, does RSHARD contain your drivers, as M80HARD does? I got confused after reading the literature. I have a 5MB TRS80 unit.

Fm MISOSYS, Inc: You're confusing me! All MISOSYS Model III software should work on the MAX-80 except SU - there is a specific MAX-80 version. RSHARD is a MISOSYS product, thus it contains our drivers.

As for the problem of direct booting from the MAX-80 hard drive, it appears that the original work was not complete for certain configurations. Since I have recently put together a complete MAX-80 system for testing prior to sale - including a hard drive - I decided to dig into the difficulty. Follow these steps:

1. Apply the M80DVR/FIX which is on the 5.3.0M disk. Apply via,

PATCH SYS0/SYS.SYSTEM M80DVR (O=N)

This will bring the driver up to Lobo's WIN product instead of the earlier UVC product.

2. Apply the following patch to the SYSGEN module:

PATCH SYS7/SYS.SYSTEM (D0A,75=0300773A:F0A,75=00000000)

3. Follow the install procedure for M80HD. Don't forget to build the BASIC "ROM image" onto SYS0/SYS if you want it. Put the DOS onto the drive partition with head 0.

4. Backup to the HD, then switch to it as the SYSTEM drive. Then SYSTEM (SYSGEN) to place a config file on it.

5. Run the M80BOOT command. This changes the BOOT/SYS for M80.

6. Apply the following FIXBOOT/FIX to SYS0/SYS but only to the hard drive:

PATCH SYS0/SYS.SYSTEM FIXBOOT (O=N).

D00,4D=21 DE 04 11 3F 10 01 07 00 ED B0 21 3C 10 00

D00,67=C3 94 10

7. If your hard drive has other than 4 physical heads, apply the following fix:

PATCH SYS0/SYS.SYSTEM (D0B,D7=XX:F0B,D7=04)

where "XX" is the number of physical heads (02, for instance).

8. Move DIP switches 6, 7, and 8 to the UP position. BOOT from the hard drive.

Code updated?

Fm MF: Roy, Has SuperUtility+ or ToolBelt/ToolBox ever been updated to handle the 6.3.x directory structure (i.e., new dating)? Mine are from Powersoft days, and if the code's been updated I should get new ones?

Fm MISOSYS, Inc: They were never updated. I wouldn't touch the code in SU+ with a ten-foot pole. As far as the toolbelt/toolbox, we have other products which operate similarly and are x.3 compatible.

FDC Card Installation

Fm RTM: I purchased a Model III/4 FDC Card from you on 01/19/93. There wasn't any insructions on how to install the card. Help please.

Fm MISOSYS, Inc: They aren't kits. To venture into the world of electronic cards for the TRS-80, you need to get your hands on a Tech Manual for the machine. The FDC card we sell is simply a replacement. It takes a few pages of instructions to convey the best way to open up your desktop Model III or 4 and lift off the top without risking breakage of the CRT. If you do not know what to do, either get the Radio Shack Tech Manual (should still be available from RS), or get the assistance of a knowledgable assistant. Conversion from a cassette machine requires many other parts.

To help you along, the following information was provided by the LDOS Forum's hardware *expert*:

Fm JB: If you don't have 2 or 3 spare power supply connectors (little 4 pin jacks), you will need either an extra 38 watt power supply or a new 60 watt which replaces your existing 38 watt. A single 38 watt power supply is on a 4"x5" PC card on the front of your motherboard case. A 60 watter is an 7"x9" card. If you have the large power supply but no connectors, you may need cables. Check when the top is off, but before you remove the CPU. If you need power supply parts, put the computer back together until you have what you need to complete the job.

1. Place the computer on its rear panel, using a towell to prevent scratching the case. Remove cables from the bottom and rear of the computer. Remove 10 screws from the bottom, making a diagram of where the go by type and length.

2. Place the computer back on its feet and remove the black screw from the rear of the compuer. This screw (a) prevents the motherboard from breaking the monitor tube socket if the computer is dropped and (b) provides a ground connection. Don't lose it, and put it back when you are through.

3. Very carefully remove the top, with monitor tube, by lifting slowly straight up and setting it aside to the left on a towell. The cable to the video board is short and requires that the top be laid down just to the left of the comuter. Find the plug, note its connection, and unplug it. Set the top aside.

4. The motherboard, FDC board, and RS-232 board are inside the vertical shielded box at the rear of the computer. Remove the 6 screws which attach the cover/shield to the rear of this box. Carefully peel away the tape at the bottom of the shield; this tape is to be re-used on reassembly.

5. Mark and remove the cables to the CPU board (power supply, video, keyboard, cassette, RS-232, sound board). Remove the 8 screws holding the CPU board, 3 at the top and bottom and one at each side. Remove the CPU board gently. Use needle-nose pliers to compress any nylon retainers which hold the CPU board.

6. Install an RS-232 board if your computer doesn't have one. Get it from Roy. Don't put your computer back together without an RS-232 board. It's cheap, and you have to remove even the FDC board to install it.

7. Install the FDC board using #6 x 1/4" screws and plastic spacer mounts. Attach an unused power supply connector to J3. Press the flat 20 conductor mylar flex cable gently into J2, making sure that it is aligned and all connections are made. Use TV tuner cleaner on the cable to ease the installation and ensure good connnections. Bend the cable so that it will go over the top of the CPU board.

8. Replace the CPU board. Carefully press the mylar flex cable into J7, which will be near the upper left corner of the PC board (looking from the rear of the computer, which is the component side of the board). J8, near the top and to the right of center, is for the RS-232 board. Pin 1 is on the left side of both J7 and J8.

9. Finish installation of the CPU board by reversing the disassembly process. Use your diagrams and wiring tabs to make sure that all screws and wires are properly replaced. Check your work. Have a cup of coffee or a soft drink and recheck your work again. Don't forget the sound board. Remember all the grounding tabs.

10. Connect the FDC cable to the FDC board. Pin 1 on the FDC board is on the left, looking from the rear. Pin 1 on the cable is marked as the colored wire on one edge of the ribbon cable.

11. Put the shield back on the rear of the CPU board. Use the floppy drive manual to set the jumpers on the floppy drives properly. Jumper settings may vary according to whether your floppy cable has a twist in a few conductors between the connectors or not. Jumper settings are identical to those for an IBM PC clone. Install the disk chimney and mount the floppy drives. Break out the dummy floppy drive covers from the comuputer top, being careful not to harm the paint, case, or monitor tube. Connect the power supply cables to the floppy drives. Connect the FDC ribbon cable to the back of the floppy drives. Attach all mu-metal shields to the disk chimney.

12. Finish assembly of your computer by reversing the rest of the instructions. Use your diagrams and recheck your work. Be sure and replace the little black metal screw in the back.

13. Boot your computer without a floppy. It should boot to ROM BASIC as before. Check out your computer to make sure that it works as before. If not, disassemble and redo any missed connectors or bad connnections. Remeber, I told you to double check your work as you go.

14. Use an LDOS, MS-DOS, or TRSDOS system disk in the bottom drive and reboot your computer using the RESET button. If it doesn't boot, see step 13.

Discontinued Products

Fm AM: Well, Roy, I am sad, but I know you have been very faithful to those of us that are genuine "die hards". Thank you for all (and I do mean ALL!) your continued fine support for the TRS-80's! I hope you will at least continue to fill orders for the older I/III products as long as you have stock available. When you released the LDOS 5.3.1 for the Model I, you truly thrilled my heart, and I know you revived the interest of at least a few in the original Radio Shack jewel that started all this frenzy. Know that you will not be forgotten, and as long as I draw breath, I will still be tapping out .RS0LT0FF to gain access to some file! Although I have never met you in person, I feel that I know you. Happy New Year to you and yours! Will you continue to publish TMQ?

Fm MISOSYS, Inc: TMQ will continue through issue VII.iv (7.4) for those who don't speak Roman). The products listed in TMQ VII.i as "closeout", will be eventually removed from market by late spring even if any manuals remain. I have to stop bothering with this TRS-80 market (as small as it is) and get a life (i.e. job) to make some money. Many of the products listed as closed out have not even sold one copy in the past year, so there just cannot be any legitimate fuss from anyone.

You or others may be interested in some other old stuff I have been digging up in trying to clean out the back room. I came across a Data Dubber originally sold by The Peripherals People. I also found a brand new Omikron Mapper. I have a Micromation board with 8" floppy drives

which was used to turn a Model I into a CP/M machine. There's the Stringy Floppy. I also have a System 9710 by Micropheral Devices. That interesting box connects up to a specially adapted IBM Selectric II typewriter to turn it into a printer. The thing connects up to the parallel printer port. This was used by some of the early types in the business for doing User Manuals. My early manuals were done on it. The Selectric II is still used here as a typewriter. Some of this stuff may be offered to the Smithsonian. There's an old 10 Meg 8" IMI hard disk drive and a power supply for it which is almost as big as a 4P. Interesting stuff.

Fm BP: Roy, I wouldn't say I was "shocked" at your comment about reduced emphasis on the Model I/III/IV software, with future support limited to Model 4 operating systems and a few utilities, but I was saddened. That surely will drive me, and many others, to the MS-DOS world. And I still really like these old machines.

Say, I'll bet there is a silver lining to this one! Maybe MISOSYS will have a special on software! I could upgrade **all** my stuff to the "latest", and final, release. That's a plus. Now to take that overdue inventory!

Any plans for one of those "Buy the Drive, Get the Software" Deals?

Fm MISOSYS, Inc: The closeout prices are listed in TMQ VII.i; these are special prices being offered to TMQ subscribers first. The closeout pricing has been advertised to others starting March 1.

Fm MP: Roy, can we hope that when you drop the old TRS-80 stuff, that you'll at least donate it to the public domain? It is a real drag when old software disappears forever when it could have been placed into the public domain so the users will at least benefit from it.

Fm MISOSYS, Inc: If no one has bought an item in 1-2 years, why should I expect that folks will want it now? Give me a good answer to that!

Fm MP: OK, I'll try my best. First, the most obvious is that the software would be **free**. Second, there will always be "new" people coming into the scene, even if it's only 1 a year. I tend to look towards the future. What happens in 50 years when the Antique Computer Club (or whatever) has a swap meet, and the talk turns to, say, Bounceoids. No-one there has a copy, or knows where to get one because the company last responsible for it simply disappeared. Or, on the other hand, maybe everyone has a copy because the same company donated it to the public domain before abandoning it. I have watched this happen with CP/M software as the years have gone by. Companies have either dropped CP/M software, or gone out of business, and then 10 years later, I get messages on my BBS from people looking for Supercalc 2 or whatever. If Sorcim had donated Supercalc 2 to the public domain before dropping CP/M support, I'd have a copy to give these folks. So I say: IF you are going to drop the TRS-80 anyways, any little potential revenue from the products is lost anyway. So **why not** donate this stuff to the public domain so that future generations can benefit from it? It won't cost you a cent.

Fm MISOSYS, Inc: Sorry, but I don't necessarily have the rights to place all the software I sell into the public domain. Second, if folks would take it for free, but are not willing to pay a reasonable price, then it shows you what its worth. I don't buy that argument. Third, folks will still bug me over support, regardless. Fourth, code within my programs is used in more than one program - I am not discontinuing everything. I am only stopping products which no one seems to be buying at a price needed to make it worth my while.

Fm MF: Roy: If you don't want to donate to public domain (which is pretty risky anyway), please consider transferring your unsold stock and possibly source code as well to the folks at Computer News 80, rather than throwing it away outright. For one thing, *Computer News 80* offers a service of "freshening" original master disks which can no longer be read. For another, they have a library of older software and seem interested in keeping on with TRS-80s.

Thank you for your long support. I use MS-DOS machines, too, so keep me posted on your new endeavors.

Fm MISOSYS, Inc: To all those who have an interest, my recomendation for anyone interested in maintaining a collection of my discontinued software is to consider setting up the way remainder book sellers operate. Publishers are always closing out their old stock. There are companies which remainder discontinued products.

Right now, with the remaining stock I have left of user manuals, someone or some group could buy up all the stock and have sufficient products to last for years. There will not be a need to manufacture new copies. All it takes is for folks to put their money where their mouth is. I am getting out of the TRS-80 market for all but a handfull of products. If some one or group of folks want to buy out the remaining stock, make me an offer for quantities. You have my price list for closeout pricing. I am certainly willing to sell whatever quantity I still have on hand. But after June 21st, anything left will probably go to the recycler. I am tired of paying to rent warehouse space to store such items that do not sell.

As an example, I took over BASIC/S when I acquired the Powersoft products in November of 1989. BASIC/S was then added - along with all the other Powersoft products - to my catalog at a price of $29.95. For three years in a row, I spent money printing the data on BASIC/S in my catalog; I also advertised all the Powersoft products in CN80 for a short spell. I **never** sold a single copy of BASIC/S - that's in three years. Now all of a sudden I say it is being closed out for $5 and someone wants to buy it. I suggest that the product was not overpriced. If no one

wanted to buy BASIC/S for three years, it tells you what the product was worth - at least in the minds of the user community. After three years of 'advertising' a product that wasn't selling, I junked it. No company in their right mind would continue to offer a product which hasn't sold for three years - let alone one year.

So folks, make me an offer. I will not transfer duplication rights for items being closed out. Remaindering is the only option at this point. When LSI decided it was no longer profitable for them to continue in the TRS-80 market, I purchased their software and rights for well over $100,000; royalties are still owed under certain conditions for Model 4 LS-DOS 6.3 sales. When Teletrends wanted out, I purchased their supply of parts for the TT512 modem for over $5,000. I paid a royalty advance up front of $1,000 to The Cornsoft Group for their rights to the five Cornsoft games I now sell. When Powersoft wanted out, I purchased the rights to PowerSoft software for nearly $25,000. When Aerocomp closed down, I acquired their remaining supply of FDC boards, RS232 boards, Model I Doubler boads (DDC), and hard drive parts along with the CP/M hard disk drivers). For this I paid a few thousand up front and owe a royalty on each unit sold. In order to continue to support the TRS-80 market, I have put my money up front to ensure that products remained in availability. I think it only fair that for anyone - or any group - who wants to purchase MISOSYS software should pay up front. There is no other option.

I have plenty of software stock left right now to satisfy any demand for the next ten years. Folks need to put up the money now! I've made some money in the past, so I'm not griping about the costs - only reflecting that in the business world, you better be prepared to invest in order to reap.

File conversion

Fm DH: Any suggestions on how to transfer to a PS/2 a file that was created in Profile Plus on a TRS-80 Model 4? Are there any DOS databases that would handle this?

Fm MISOSYS, Inc: MISOSYS sells a product called TRSCROSS which should run on the PS/2 and read the TRS-80 disks. You will need a 5.25" drive - which may not be available for the PS/2.

We also have a data base package called LB86 which has a data base conversion utility called LBCONV which can directly read your Profile data base file se and create an LB file set. LBCONV can

also convert LB format to dBASE format. This requires the Profile file set to be transferred to the PC as binary images. If you can TRSCROSS the files or modem the files, you can then directly handle the Profile data with LBCONV. LB86 currently sells for $99 + S&H; you can take 50% off by trading in your Profile.

MLIB, Double Duty, LB

Fm AM, Tecamachalco, MEX: Roy, I was reading my TMQ VI and I found it's time to renew my subscription. In addition, I want to buy some software, But first, I have some comments and questions about other programs I have:

MLIB 4.0 and UNREL 1.0b: I was trying to take a look at some of the routines in the library of the high resolution graphics for FORTRAN (GRPLIB/REL) and the fortran library using UNREL, so I tried to separate them using MLIB but got the error "invalid file format". The program spent some time loading the libraries before the error was displayed, so I figured that was some problem with the way the /rel files terminated and MLIB expected to. I moved the /rel files to CP/M and used the LIB program and successfully separated any modules I wanted. I moved back the separated modules to LSDOS. Then I tried to use UNREL but got a similar message. Then, another idea ocurred to me. Using SPLITLIB I created another library equal to the first by specifying a large size so the result was a single file. Then I used MLIB successfully to separate the modules of this new file, and then UNREL to obtain a source listing. My question is: why L80, SPLITLIB, DECODREL and the LIB (from CP/M) can read the files but MLIB and UNREL can't, unless I "pass" them through SPLITLIB?

DDUTY 02.06.00, running under LSDOS 6.3.1: I have a special board called EVM which is a microcomputer without a keyboard or display. It has an RS232 port so it can be used with a computer. The software inside the board has an assembler, a debuger, and a text editor. I'm making a program for the TMS7000 microprocessor. The text editor is very limited, so I edit the source code in the TRS-80 model 4, then download the program to the EVM then assemble it for debugging. That's when DDUTY comes into play: In one partition I run COMM to download the program and to operate the onboard debugger. In the other partition I use the SAID editor, so I can go step by step checking my program using COMM and, when I find an error, just switch to SAID and correct the source code, then back to COMM to debug some other section. It is **very** convenient to do it this way. I found that when DDUTY is switching from one partition to an other and in that moment characters are received from the RS232 then a system crash occurs. Due to the way the EVM behaves, it's very difficult to prevent that from happening from time to time. Of course, that night happen any time COMM is running on one partition and there is no control on the other side of the connection. Is there any solution to this?.

LB 2.2.0: I have been using LB to maintain a database of electronic components of a controller I designed. In each record are the component ID, (for example R203 for a resistor), the component value (100 ohms for example) plus other fields.

I need to generate reports, for example, of the total of the 100 ohm resistors in the database, so when I order material just multiply that total by the number of controllers to assemble. There are about 150 different types (values) of components of a total of about 350. Selecting them one by one will require going to the select menu 150 times (!), and also know all the values beforehand. What I'm trying to say is that beside sorting and selecting options, the capability of making totals of the values of a field can be very useful. For example, a report as follows:

```
100 ohm      2
2.2k         1
100k         2
74C245       1
```

Can be generated from the following data: 100 ohm, 74C245, 100 ohm, 2.2k, 100k, 100k, 100 ohm, where all correspond to the same field. (the value field).

Regarding LB also, I have a question: How can I select records that have a field with no information entered?. I need to do it because at the time the record was typed in some info. was not available and now, I need a report of those incomplete records to update them.

Where can I get a BIOS for my CP/M 3+? The one I have is the once sold by Radio Shack and is full of bugs and badly implemented.

Fm MISOSYS, Inc: Here's some answers which you may find useful:

Both MLIB and UNREL require a library file to have an "end-of-file" indicator as part of the relocatable module bit stream. If you examine the technical information, such a code will start on a byte boundary and have the sequence, 1001111. This implies a trailing 0 to fill out the byte; thus, the end-of-file code is a 9EH. At one time when MLIB was sold as a separate product, the disk had a file called "ENDREL/REL" which contained the single 9EH byte. A note indicated that you could APPEND the ENDREL/REL file to any *foreign* module which would not load properly. That's the problem you are having. I see that ENREL/REL may not have made it to the MRAS or UNREL disks. Simply create such a file (you could use the DOS BUILD command with the HEX parameter), then append the file to any library you are having a problem with.

DoubleDuty should be able to work properly with the COM/DVR provided by the DOS if the driver is installed before loading DoubleDuty. That's because the driver uses the serial port's received character

interrupt for receive data. If the COM driver was installed into a partition which was swapped out, a received character would cause an interrupt which would be vectored to a driver handler no longer resident. The computer would most likely crash. If you are using a communications program which accesses the serial port directly and which uses the interrupt, you can not switch that program out of its running partition. Now since you are using COMM, that implies use of the COM/DVR. So make sure the driver is installed before DoubleDuty is installed.

Your need of a capability in LB for making totals of the values of a field is called a *cross-tabulation*. LB does not currently provide that capability. But I'll consider that as an item for a future release. Cross tabulation could also be done via an external utility. The data base structure is thoroughly documented in the Reference Manual. One utility which operates on the data file was published in The MISOSYS Quarterly, issue III.i (summer 1988). Cross tabulation could be done simply by scanning through the data records - perhaps according to an index file - accumulating totals on those fields you wish to cross tabulate. Obviously for each field, you would need to construct a two-dimensional array (one string element and one integer element) of the differing field contents so the diffeent values could be grouped and totaled. A simple insertion sort could be used to keep the various data entries of a field; or a btree could be used. Perhaps some budding programmer could whip together such a tool before I have a chance to - I'm tied up with a big development effort right now.

As far as selecting records which have no value, that is an easy one. Let's consider a literal field first. Since an "L" field can contain any ASCII value, that represents a range of from 20H (a space) through 7FH. Therefore, if you select on the field with a match string of "!*" and a select criteria of "LT", that will pull in any record that does not have a non-space character. For alphabetic fields, use a select string of "A*". Numeric fields allow digits 0-9, minus signs and periods. Since the minus sign is the lowest-valued ASCII character, the match string would be "-*". Another thing you could do is to turn on the ADD index when you are adding records. You can then go back to those specific records by using the ADD index during an editing session.

As far as Tandy's CP/M 3+ which they sold for the Model 4, it was always rated by users as just about worthless. Your better bet is to obtain a copy of Montezuma Micro CP/M 2.2 which should still be available from Computer News 80 in Casper WY.

LBCONV Revision (MS-DOS version)

Fm MISOSYS, Inc: I had one LB86 user who converted from a Profile 4 database which used about 80 fields. Since LB/LB86 support a maximum of 64 fields, LBCONV likewise was coded to handle a maximum of only 64 fields. It turns out that LBCONV, although only supporting a conversion of 64 fields, did not safeguard against a source data set with more than 64 fields.

I have therefore upgraded the LBCONV utility to properly accept a Profile database file of up to 99 fields - which is the maximum supported by Profile. LBCONV will display the field information for all fields in the Profile database. However, LBCONV will only convert the first active fields it finds up to a maximum of 64 - the LB limit.

Fields are "deactivated" by using the EDIT command of LBCONV to change the field type to "X". To accomplish a conversion of all the fields, convert the Profile file set and LB will take the first 64 fields. Then edit the field list to "X" out at least as many fields as exceed 64 and convert to a new LB file name. This results in two database file sets.

As the TRS-80 version had insufficent available memory to expand the data areas to 99 fields, only the MS-DOS version has been changed.

LB86 and extended characters

Fm DMS, Queensland, AUSTRALIA: I am encountering a problem with the LB86 Data Base Manager when attempting to print Multiple Across Labels. The problem appears to be the use of a Control Code \156\ to print the POUND Sign. Example 1 shows that on lines 2 and 3 the characters move one space to the right on the second label and two spaces on the third label. Presumably this would continue on subsequent labels.

However Example 2 shows that using the # instead of the Control Code there is no problem.

I would be obliged if you could offer any advise to overcome this situation. My printer is a PANASONIC KX-P1180.

Fm MISOSYS, Inc: The use of control codes in LB86 (and equivalently, LB) is for the control of printer functions, although you have found that any character value may be generated with a control code. The reason why LB permits any value to be a control code is that printers may use any character or sequence of characters for control purposes; they do not limit themselves to ASCII control codes.

But the LB User Manual is quite specific

in stating its treatment of control codes. On page 98, the following statement reveals your problem: "In printing control characters, LBDM will not count the characters used to define the control code." Therefore, not only are the sequence of characters within and including the backslashes ignored for the purposes of column counting, neither are any of the character values generated from the control code. Thus, your use of a control code with a value of 156 to print the British POUND sign will print as such on your printer, but LB86 does not increment its column counter. So when you are printing multiple across labels, each succeeding label across will shift that line by one column each time the POUND is printed.

Your solution is to code convert a printable character external to LB. That's pretty easy on the TRS-80 version using a printer filter; the FORMS filter's XLATE parameter could be used for this purpose. Under MS-DOS, it would take a TSR to handle the character conversion. Since LB86 supports only 7-bit *printable* characters, you cannot simply enter and character value greater than 127; thus, either a printable character needs to be converted externally, or the control character facility must be used (which does not align columns).

It sounds like a character conversion table may be a useful addition to a future release of LB. But I'm not making any promises.

Converting PowerMAIL data to MS-DOS & LB

Fm MISOSYS, Inc: We recently had a query on how best to migrate a PowerMAIL data base to MS-DOS. That problem is easily solved using two of our existing products: TRSCROSS and LB86. Here's the solution:

The TRSCROSS program can read the TRS-80 Model 4 disk files directly on an MS-DOS machine. LB86 has a database conversion utility, LBCONV, which can directly generate an LB86 data file set from the PowerMAIL ADDER file. Here's what you need to do:

1. Copy all of your data file records into an adder file; you may need more than one adder file to accomplish this depending on the number and size of your data files. The technique for doing this is covered in the PowerMAIL Plus Operator's Manual under the topic, *Creating subfiles from a PMAIL/DAT file.*

2. The record fielding of the ADDER file is discussed in the *Technical Information* section of the manual. As you will see, each PowerMAIL Plus record is contained in 128 bytes (the record size). Note that the manual has an incorrect value for the "Data 2" field - the size is "12", not "2". The ADDER file also contains an additional sector of data; however, since this is exactly 256 bytes, it can be thought of as two fictitious data records.

3. You would use TRSCROSS to copy the PMAIL/ADD file(s) as binary images. That's just a mode you will specify in the TRSCROSS program.

4. Once the ADDER file is moved to an MS-DOS disk, you can then build an LB86 database using the ADDER data file. LBCONV needs to know the length of a record (128), and the type, starting position, and length of each field. All fields with data usable by LB86 are ASCII information. The length and starting position of each field would be as follows:

Last name	1,15
First name	16,10
Company	26,20
Address1	46,20
Address2	66,10
City	76,15
State	91,8
Zip code	99,10
Data 1	109,5
Data2	114,12

With this set of specifications, LBCONV will generate an LB database file set. You would then be able to access your data using LB86. You can automatically generate a view screen using LB's *Define screen* module. LB's Define print module has a facility for automatically generating mailing label formats as well as form listings you can use.

LB Selecting & Sorting

Fm WEH, Humboldt, TN: On June 4, 1988 you shipped me item no. L-50-510 Model 4 - LB Data Manager and item no. L-50-515 Model 4 - LB Mtce Utility.

At about that time in my retired life, I became extremely occupied with volunteer work for Lions Clubs International and had no time to devote to any new computer projects. I never took the above mentioned items out of their package until a few a weeks ago. I am presently maintaining a mailing list of Lions members names and addresses - about 1800 data sets - using the label printing technique of Allwrite. I cannot take all the data into memory of a Model 4-D at once and do not have sorting capability. My data is maintained in several files and insertions and deletions have to be done manually at the proper places.

Recently I decided to try to set up another data base which I maintain, using Little Brother. I have poured over the manual, reading, re-reading and reading several times again, trying to understand the procedure for setting up a data base. I have concluded that LB is an extremely versatile program and that the author of the manual undoubtedly knows a great deal about it. Unfortunately it has proved ex-

tremely difficult for the manual to impart much help to me in setting up a data base. I have tried to set up an experimental file using data from two amateur radio clubs for which I also maintain mailing lists. I have managed to set up passable field definitions, screen definitions and print definitions. Print- outs of some of the are enclosed. I am stuck, however, at selecting and sorting.

The single letter identification of the two clubs are "A" and "H". There is much overlapping of their membership. I intended to use Field No. 1- STATUS - to be able to select the names and addresses for either "A" or "H", "A" and "H" combined and to add to either or both groups former members "F" and prospective members "P". The resulting grouping needs to be sorted in ascending ZIP order (Field #10). Can you suggest how this can be done?

Recently I received from you an offer to obtain the latest 2.3 version of LB at a 60% savings. Can I assume that the new manual is even just a little bit easier to understand by one who doesn't know all about the subject than the original one? I have no need for calculated fields. I just want to maintain simple mailing lists from which I can include or exclude certain groups of names. If you are willing to ship it to me with the conditions that I may return it for a refund if I just can't comprehend the new manual, I am willing to try it. My origina index page is enclosed.

Fm MISOSYS, Inc: From the description of your existing database and your desire to extract member records based on the STATUS field, your solution is really quite simple. Here's what you can do...

If you want to send a mailing to members who belong to either "A" or "B" (i.e. their status field is either an A, B, AB, or BA), then use the sort/select command:

1. First specify field 10 sorted in ascending order with equal to anything (i.e. just ENTER); this is to cause a sort on the ZIP code. Then specify "AND" for the next selection criteria.

2. Next, select on field one with a match string of "*A*"; do NOT add it to the sort string; this will select any record if the status field has an "A" in it. The reason for the both the [preceding and succeeding asterisks are to ensure the proper selection if the status field has the membership status in any order. Then specify "OR" for the next selection criteria.

3. Next, select on field one with a match string of "*H*"; do NOT add it to the sort string; this will select any record if the status field has an "H" in it.

4. Lastly, there is no more criteria, so designate an index file to use.

If you wish to send a mailing only to members of both the "A" and "B" groups, then use an "AND" connective in step 3 instead of "OR". This will tag the record only if the status field has both membership states.

If you wish to add in the former or prospective members to either of the above selections, then you should simply request an "OR" connective in step 4 with a match string based on where you have the "F" or "P" status stored. If this is also kept in field 1, then you are selecting field one a third or fourth time - that is not a problem so long as you do not add a field to the SORT STRING more than once.

Once the index file is created, you use that file with the PRINT command to print only those selected records. I also recommend creating a macro (JOB) file for the sort/select set of commands so you can automate the index file generation for the next time you need to re-select.

The LB version 2 manual contains an extensive section which takes you through the motions step-by-step of creating a database. It illustrates this activity with a mailing list application included with the package as a template. When we released 2.0, we included that database template named MAILFILE. This template is discussed in the *Getting Started* section of the *User Manual*. With releases 2.0 through 2.2, the template files fit onto the Installation disk; however, with the added modules of 2.3, MAILFILE no longer fit. We have various database templates available (see current list herein). MAILFILE appears on the 2nd template disk. Because a third disk is now needed to provide LB for the TRS-80, we provide the complete TEMPLATE 2 disk for the MAILFILE template.

LB86 and Tandy 2000

Fm MISOSYS, Inc: Just a brief note to those throngs still pecking away at a Tandy Model 2000. I recently decided to put my color 2000 up for sale (see last TMQ issue). In order to do that, I had to put it back together and test it out - I had previously dismantled it to use the 80 track drives over on a Model 4. Once back together, it still worked. I then decided to try out version LB86 2.3.

I discovered without great surprise, that the LB86 install program crashed due to the different video addressing of the Tandy 2000. However, when I copied over a decompressed set of LB86 files, the database package worked without a hitch. Even the color support up to snuff. So if you still are playing around with a Tandy 2000 and want to acquire a database for it, LB86 will work. All you need do is to specify that you need a custom decompressed set of LB86 files.

Inside TMQ

4P40M
A 40 Megabyte 4P

by Roy Soltoff

When MISOSYS first designed its TRS-80 Model III/4 hard disk package, the most common drive available was the MFM drive (ST506/412 type of interface). MFM was the original drive type used with the TRS-80 Model I and III by Tandy. In fact, it was the only type available. Tandy used a custom hard disk controller provided by Western Digital; however, that controller was still quite expensive and had limited availability.

Other common controllers used with MFM drives were the Xebec S1410 (virtually a standard) and the Adaptec 4000. Both of these controllers required a SASI (Shugart Associates System Interface) port on the host (Computer side) and presented an MFM interface on the drive side. SASI grew into SCSI (Small Computer Systems Interface) which is essentially electrically compatible with additional software commands. As an aside, due to the terseness of the command set with regard to supporting more than just hard drives, SCSI grew into SCSI-2, and is on the verge of growing into SCSI-III.

As had been discussed in previous issues of the *Quarterly*, I had originally wanted to use a new PC-compatible controller; however, there never was a direct way to control the sector size (available PC -XT controllers were fixed at 512-byte sectors). Thus, I had turned to the surplus availability of S1410-type controllers. An adequate supply of Adaptec 4010 and Xebec 1421 controllers was acquired (approximately 400). These could be easily associated with the MFM drive type. All that was needed was a host adaptor to couple the TRS-80 50-pin external bus to the SASI interface side of the controller. Thus, the MISOSYS T80 to SCSI host adaptor was born.

Subsequent to the use of the existing controllers and MFM drives, drives with embedded controllers started to make their way onto the scene. Two types of embedded controller protocols were ESDI and SCSI. Since my host adaptor would be port-compatible with the SCSI drive, I decided to explore writing a driver to directly support SCSI drives. The benefit of this would be to lower the cost, do away with the controller and subsequent power consumption, and take up less space for an internal installation. Because of a requirement left undocumented in the Seagate SCSI Interface Manual, the completion of the driver was delayed for well over 18 months; Seagate proved quite elusive in the area of providing technical assistance. As was noted in the VII.i issue of *The MISOSYS Quarterly*, I had finally extracted from Seagate, the information needed to complete my SCSI hard disk driver. Unfortunately, the ST157N is no longer manufactured by Seagate; however, re-furbished drives are available from **jb, TECHNOLOGIES**, 5105 Maureen Lane, Moorpark, CA 93021 [800-688-0908, FAX 529-6057] for $190. You may also need a 5.25" adaptor housing. I also have a small quantity of 20MB ST325N drives available for $150.

My first thought of installation of the Seagate ST157N drive was a configuration internal to the Model 4P computer. That's a relatively compact unit, prized by most folks owning one. An un-modified 4P has two half-height 5.25" drives in a drive bay to the right of the video tube. Considering power consumption, each

original floppy drive draws 0.8 amps average at 12 volts (9.6 watts) and 0.8 amps (4 watts) from the 5 volt supply. Peak current draw from the 12 volt supply increases to 1.3 amps (15.6 watts) during motor turn-on. The Seagate ST157N draws 12 watts which is less than the total average power of the Tandon TM50-1 floppy drive; however, the information at my disposal does not isolate the power consumption between the two supply voltages. My host adaptor consumes about 1.5 watts from a 5 volt supply; thus, the ST157N with host adaptor should easily replace a single floppy on a total power consumption basis.

The 4P in which I installed the 40 megabyte drive was equipped with Tandon TM50-2 360K drives and an XLR8er board. According to the Technical Manual, the TM50-2 should consume the same power as the original TM50-1 installed. However, after installing the hard drive, I made the observation that the video screen bloomed on access of the 5.25" floppy as the motor turned on. Therefore, I decided to reduce the power draw on the supply by replacing the remaining floppy with a 3.5" drive as modern 3.5" floppy drives consume very little power - in the neighborhood of 1.5 watts and only from the 5 volt supply. Besides, the 720K 3.5" drive would work better as a device to back up files from the hard drive.

The following material covers the methods I used to physically install the host adaptor and the hard drive into the 4P. Other methods could be used to mount in a desktop installation (i.e. a 4 or 4D).

What you will need

To install a drive and host adaptor, you will need (* indicates part available from MISOSYS - we can make the cables):

1. two nylon standoffs, 2" in length (* $1)
2. two nylon nuts (see text)
3. one 12" long Y power cable (* $5)
4. three 50-pin female header connectors
5. one 50-pin edgecard connector **or**
5. a fourth header connector and a 50-pin header pin strip
6. three feet of 50-conductor ribbon cable (* $20 for both cables)

Optional parts

1. one switch: Radio Shack 275-636
2. 20" red .22 gauge hookup wire
3. 10" yellow .22 gauge hookup wire
4. 20" 9-conductor ribbon cable (for joystick)
5. one 9-pin male D-sub-miniature IDC connector (for joystick)
6. one 10-pin header connector (for joystick)
   (* $10 for completed cable)

Before you begin

Before I began the installation, I decided to also modify the 4P motherboard to add the port for external floppy drives. This modification was covered in Tsun Tam's article, Upgrade your 4P with external floppy drives, originally appearing in *80 Microcomputing* and reprinted with their permission in *The MISOSYS Quarterly*, issue IV.iii (reprints available). I would likewise recommend that anyone taking the time to install a hard drive into a 4P should perform the external floppy port modification at the same time if it has not already been done. As the 4P I converted was a gate array model, the external floppy modification was exceedingly simple; all it took was the installation of four jumper wires and the fabrication of the new cables. The addition of the external floppy port also provided me an opportunity to re-use the two internal 360K floppies removed to allow room for the hard drive and 3.5" 720K floppy. I conveniently installed the two 360K drives in one of my external floppy drive cases.

Because the 4P upgrade article detailed the procedure on opening up the 4P, I will not discuss that in this article. Please refer to your 4P *Service Manual* or the previous article.

When Tandy was designing the Model 4, they originally wanted to allow for a hard drive internal to the case. Because of that, the motherboard was fabricated with a provision for a header pin strip in common with the 50-pin expansion port edgecard fingers. If you have ever looked at a Model 4 mainboard, you will observe the dual row of holes (25x2) just to the inside of the edgecard fingers. These holes were designed for the installation of a header strip. But Tandy never utilized the external port internally - and consequently never installed the header. The through holes (that's the term applied to the holes going through the printed circuit board) therefore become collected with solder during the wave soldering of the boards.

I wanted my 4P to be able to continue the availability of the external edgecard port, so I decided to populate the header-holes with a header pin strip. This is useful to allow you to add some other non-competing external device needing the port. I use the term non-competing from the standpoint of the CPU ports used by the device. My host adaptor uses the same ports for the hard drive as does the Tandy hard drive; thus, both could not be connected at the same time unless one wasn't electrically *seen*. Using an internally added header for the connection gave me the opportunity to connect, for instance, a different hard drive (Radio Shack) to the machine without having to switch cables simply by disabling the internal host adaptor with a switch. Besides, trying to plug in an edgecard connector to the external port from a cable snaking from inside the machine is rather clumsy. On the other hand, unless you have access to a vacuum desoldering machine, you will not be able to remove the solder from the through holes to enable the addition of a header pin strip. But if you do have access to such a tool, I would recommend that you add the strip.

Installing the hard drive

Physically mounting the hard drive is relatively easy - it takes no more effort than changing out the floppy. With the 3.5" ST157N drive mounted in a 5.25" half height housing, the drive mounts vertically just like a floppy. Vertically mounting the 3.5" hard drive on either edge (not either end) is an acceptable mounting

position according to the specification sheet. This will most likely require some fine tuning of the mounting holes in the drive cage to allow for proper positioning of the drive bezel.

The mounting hole positions for 5.25" drives are a standard - all drives have screw holes in the same location. But these positions are relative to the body of the drive case. The problem with the 4P is that the drive cage mounts with the front bezel of the drive fitting flush against the case front. Different drives have different thicknesses of front bezels - there's no standard for that. If the drives fit through the front case, there would have never been a need to modify the hole positions of the cage; however, because the holes must be positioned according to their displacement from the front of the case, drives with different thicknesses of front bezels will require different locations for the mounting holes. You can find the exact locations for the holes somewhat easily as follows:

1. After removing the drive cage from the 4P, remove the floppy drive which is on the closed side of the cage. This is the drive :1 floppy and it will be removed to provide a position for the hard drive.

2. Now hold the hard drive on the outside of the cage so that you can see its mounting holes; align the front bezel of the hard drive to coincide with the bezel of the remaining floppy. Mark the front-to-back position of the holes on the cage.

3. Now slide the hard drive into the cage and check the displacement of the holes from the closed portion of the cage (the drive electronics board will be adjacent to the remaining floppy. If the holes do not line up exactly, mark their displacement.

Typically, the needed holes will be relatively close to the existing holes; a small round file will allow you to expand the existing holes to fit the mark. Please remove the existing drives from the cage before using the file.

The same procedure can be used to ensure alignment of the holes for the 3.5" drive - if you are proceeding with that change as well (recommended). This time, re-install one of the drives (or install the hard drive) into the closed side of the card cage and check alignment of the holes for the floppy - aligning the front bezel with the front bezel of the existing drive.

Installing the Host Adaptor

The host adaptor is a small circuit board approximately 3.625" by 6.25" in size. The only available position is in the area containing the CRT, power supply, and video board. I prefer to have a solid mounting; therefore, I decided to mount the host adaptor above the power supply. I chose to use 2" nylon standoffs connected to the power supply circuit board to provide a physical mount for the host adaptor. I used the same standoffs to mount the host adaptor above the power supply in my original external hard drive design. The mounting holes in the host adaptor circuit board are in the positions utilized by the 60 watt supply of the Leadman case - which is also the same mounting locations for the 38 watt Astec power supply used in the Model III. The 4P, of course, uses the 68 watt supply, which is larger, and therefore does not provide existing holes in the same positions. Here's where some ingenuity - and daring - comes into play.

I found that using two standoff mounts provided sufficient stability to the host adaptor. In order to provide two mounting holes on both the power supply printed circuit board and the host adaptor board, it is necessary to find corresponding hole locations which do not interfere with the circuit traces nor the components. You must remove the power supply from its mounting panel to drill the necessary holes.

There are six screws (three on either side) which mount the power supply panel to the case. Remove the screws and tilt the panel with the supply away from the case. Remove the three power cables from the supply so that you can remove the panel. Four screws mount the power supply to the panel. Remove them and you can remove the power supply. Place it on a work surface with the power connectors positioned towards the right. Note the mylar insulator between the power supply and the panel. Make sure it stays in place when you re-mount the power supply to the panel.

If your 4P has a Tandy supply, "TANDY CORP." is stenciled on the PCB to the right of the large 220 µF capacitor. Note the hole which is above the right hand side of the fuse, just beneath resistor R38 to the left of the AC power connector. This is an unused hole which becomes one of the holes for mounting a standoff. Another location for drilling a hole can be found 1-1/8" from the bottom and 5-3/16" from the right. That position is between transformer T1 and inductor L5. If you look at the trace side of the supply, the location is just southwest of the part number 8709365. Hold the board up to the light and you should be able to see the area vacant of both traces and components by looking through the PCB. Mark the location and carefully drill the hole. A 1/8" size is sufficient; however, it is best to start with a much smaller drill bit first, then re-drill with the 1/8" bit. I use a Black & Decker high speed rotary tool for this job.

The host adaptor will be positioned above the power supply with the H/A power connector above the power supply's power connectors. Thus, the H/A mounting hole adjacent to the joystick port is used with the standoff mounted in the hole near R38. A hole corresponding to the drilled hole needs to be drilled in the H/A. The new hole is approximately where the "C" of the "SCSI INTE..." is stenciled on the H/A. Note that there are two traces on the component side of the board; you do not want to drill through them! Mark a position exactly 1-1/16" from the left edge of the H/A and 1/4" from the bottom. Drill the hole from the component side to minimize any flaking of the surface which could unseat the trace. Again, start with a small drill bit (I use a #49 which is .0730" in diameter) then enlarge it to 1/8".

If your 4P has an Astec power supply, two holes have to be drilled as the available

unused hole near D5, R17, and R18 is too far out of position. Incidentally, if you ever wondered why these holes are on circuit boards, sometimes the mechanical apparatus used to manufacture printed circuit boards needs a hole for positioning the board. Turn the Astec supply to see its trace side with the power connector to the left. One hole is marked 1-1/4" from the bottom and 5/8" from the left. From the top, this is approximately where the ground designation is stenciled to the right of C24. The other hole is marked 1-1/16" from the bottom and 5-1/4" from the left. Looking from the top, this hole is positioned above J2 (a jumper), to the right of C7, and to the left of R2. It is just below a toroidal transformer or coil whose designation I cannot read because its covered up with glue.

In both the Tandy supply and the Astec supply, the holes are very close to traces. By using a nylon standoff, there is no chance of shorting the H/A to the power supply. However, you must use either nylon nuts on the standoff or create some insulating washers. The nylon standoffs I use are two inches in length, have a 4-40 threaded shaft on one end, and are tapped for a 4-40 screw on the other end. The threaded shaft is placed through the holes from the component side of the power supply, and tightened down with either nylon nuts to avoid shorting together any of the traces, or insulating washers and regular nuts. I did not have any nylon nuts on hand, so I made some washers. I used a double thickness of the static insulating bag which hard drives come in. This is a very thick mylar. If you do not have that available, raid your kitchen for a freezer bag. A few thicknesses of that should be sufficient. I cut out a small square of about 0.5" then poked a hole through the center. This is easily done with a probe or a standard sharp hardware awl. Place the *washer* over the thread then tighten down a nut. The H/A can then be mounted on the standoffs with two 4-40 screws. I later found nylon washers at Home Depot.

Getting power to the parts

Since you will be replacing one of the floppy drives and adding two components needing power, another power connector must be added. The cable powering the floppies does not leave sufficient room to reach the hard drive, so the easiest solution overall is to use a "Y" power adaptor cable; I used a 12" adaptor cable. Recollect that I wanted to be able to selectively power down the internal drive so that I could use the external port with another drive. To isolate the hard drive from a signal standpoint, it is only necessary to disable the host adaptor - that's the only thing seen by the computer. Since the H/A uses only the 5V DC supply, the H/A can be isolated by switching off the 5V going to it. However, it is just as easy to add a double pole single throw switch to disconnect both 12V and 5V going to the hard drive and host adaptor. I used a Radio Shack Mini Flat Lever Switch, part number 275-636 which retails for $3.39. Using a switch gives you the ability of powering down the drive.

The hard drive has about a half inch of free space above and below it since it is a 3.5" drive in a 5.25" mounting. One of the locations can be used to mount a power switch on the front panel; the other position can be used to mount the 9-pin D-Sub miniature connector used for the optional joystick. All you need to do is to provide some careful drilling and filing of the plastic faceplate for the needed holes. Alternatively, you can use the original 4" faceplate and add some brackets made with sheet metal to mount the switch and joystick connector - that's what I did.

If you do not want to add a switch, all you need do is use the Y cable intact. Connector genders are referenced according to the coupling of the pins; thus, the Y has one male and two female connectors. One leg of the Y is for the H/A which needs only the 5V supply. Therefore, cut both the yellow and black closest to the yellow from one of the female connectors; also cut the opposite ends of that leg at the male connector side. You will use the small cutout to the right of the fan to provide a means of getting that leg of the Y back to the power supply compartment. Add a small piece of electrical tape to the edges of the metal cage to minimize any chaffing of the cable leads. You now have one power connector for the H/A and another for the hard drive. Connect the male connector of the Y cable to one of the floppy power connectors.

If you are going to install a power switch, follow this procedure. Using the Y cable, cut the red and yellow leads one and a half inches from the male connector; red is the 5V line and yellow is the 12V line. There is no need to sever the black leads which are ground leads. Then extend the red/yellow leads using two pieces of wire ten inches in length, from the male connector to the center poles of the switch. The female connector going to the hard drive can be wired to the other side of the switch. Wire the yellow lead to the other side of the switch using the same pole as the yellow coming from the male connector. The red lead from the other female connector which will be used to connect to the H/A needs to be wired to the red side of the switch with the HD connector's red lead. Before you wire that cable piece, snip both the yellow lead and the black lead which is adjacent to the yellow, as they will not be needed (remember the H/A uses only the 5V supply); snip them close to the body of the connector. Now use a ten inch extension piece of wire (you could get by with using the unused yellow lead previously snipped) to extend the red lead of this cable to the 5V switch pole along with the red lead from the HD power cable. I used heat shrink tubing to protect all the soldered connections; electrical tape could be used as well. The power cable going to the host adaptor can be routed as noted previously.

Connecting the signal cables

To connect the H/A to the hard drive, prepare a 50-pin ribbon cable 18 inches in length with two header connectors: one at each end. The headers are positioned on the same side of the ribbon cable. One end will be plugged into the hard drive; the other into the HRD DSK DRV CONTROLLER port (P4) on the host adaptor. As the cable must pass over the back of the power supply/CRT cage, it cannot interfere with the bracket which supports the

carrying handle. Since the position where it wraps over the lip is exactly in that spot, simply fold the portion of the cable over on itself to position it as close to the bottom of the case as you can. There is sufficient room for that positioning. To minimize the pressure on the cable when the rear panel is screwed into place, tape a spacer onto the lip of the cage on either side of the cable; I used a double thickness of ribbon cable.

If you have installed a header pin strip on the mainboard, prepare a 12" length. Using the edgecard, it probably needs to be about an inch or two longer. The connectors are on opposite sides of the cable. As the orientation of pin 1 on the H/A is opposite to the orientation of pin 1 on the mainboard, the ribbon cable will need to undergo a 180 degree twist. Plug in the cables.

There is one screw holding the modem cover plate (or blank panel) onto the rear panel which is directly overhead the ribbon cable coming from the H/A. I recommend you cut the length of this screw or find a shorter one so that the screw does not press into the cable - or leave the screw out.

Want a joystick port?

If you intend on using the joystick port of the host adaptor, you need to prepare a cable to extend from the host adaptor to the mounting point. That cable has a 9-pin D-sub-miniature male connector on one end and a 10-pin header connector on the other. I mounted mine on a small bracket just beneath the hard drive; it needed a length of 20". The cable was routed into the power supply compartment using the same cutout as the H/A power cable. Note that since the cable is 9-conductor, make sure you orient it properly (towards pin 1) in the 10-pin header connector when you are crimping the IDC connectors.

With everything closed up, re-test the drive. You now have a 4P40M.

How to Make a Mod-4 FILTOMAT

by Christopher Fara

MICRODEX
1212 N. Sawtelle
Tucson, AZ 85716
(602) 326-3502

```
;SIMPLE/FLT Mod-4 filter example

      ORG 3000H

HEADER   JR FILTER ;jump over header
LAST  DW $   ;space for end address
NICK  DB HOOK-NAME ;name length
NAME  DEFM   'SIMPLE'
HOOK  DW $   ;space for DCB address
DOSS  DW $   ;reserved for DOS

FILTER   JR NZ,CHAIN  ;skip if not PUT request
      LD A,C ;A=character to be printed
      CP 126 ;maybe tilde?
      JR NZ,CHAIN  ;no, just print it
      LD C,222  ;else replace with "cents"
CHAIN LD IX,(HOOK) ;IX=> next device in chain
FIX   EQU $-2 ;address to be fixed
      SVC 20     ;@CHNIO: chain to next device
      RET    ;done

LOADER   PUSH   DE  ;DE=>DCB if filter SET
      POP IX  ;IX=>DCB, will be used later
      LD  (HOOK),DE ;store DCB address in header
      Svc 101       ;@FLAG$: get IY=>flag table
      BIT 3,(IY+2)  ;flag C: installed via SET?
      JP  Z,NOSET   ;jump if not
      LD  HL,0   ;find current high$
      LD  B,0
      SVC 100       ;@HIGH$: get high$ address
      LD  (LAST),HL ;HL=old high$, store in header
      LD  DE,LOADER-1   ;DE=>last executable byte
      OR  A   ;clear carry for SBC
      SBC HL,DE  ;HL-DE=relocation shift
      PUSH   HL
      POP BC  ;now BC=shift
      LD  HL,FIX ;HL=>address to be fixed
      LD  E,(HL) ;get that address
      INC HL
      LD  D,(HL)
      EX  DE,HL  ;HL=address to be fixed
      ADD HL,BC  ;add the shift distance
      EX  DE,HL  ;put it back
      LD  (HL),D
      DEC HL
      LD  (HL),E
      LD  DE,(LAST) ;relocate the module
      LD  HL,LOADER-1
      LD  BC,LOADER-HEADER
      LDDR
      EX  DE,HL  ;HL=>byte below header
      SVC 100       ;@HIGH$: set new high$
```

Having had my "filter consciousness" raised on Model III LDOS, I was often puzzled by the fuss of installing filters in Model 4 LS-DOS. In Model III it's relatively simple. For instance to install a printer filter SIMPLE/FLT we would enter:

```
FILTER *PR SIMPLE
```

That's all. To accomplish the same thing in Model 4 we must first dream up a name for the "filter device", make sure it's not a duplicate of an existing device name, and then issue two separate commands such as:

```
SET *AA SIMPLE
FILTER *PR *AA
```

Somehow this always reminds me of the famous "expletive deleted" Watergate tapes. Maybe it's the asterisks. Yes, yes, I know, there are good reasons for all that and it's a more powerful method altogether. Be it as it may, many long-distance chats about asterisks with customers who were not always quite comfortable with DOS procedures, led to "filtomats" or self-installing filters, apparently (?) a little Microdex special. To understand it we must first visualize a "normal" FLT-file. About the simplest possible example might be a printer filter that replaces one particular character with another. Perhaps the rarely used "tilde", CLEAR SHIFT PLUS on Model 4 keyboard (ASCII 126), could print the "cents" symbol, code 222 on DWP-210/220 printers (Listing 1).

The SVC "macro" in the listing will be automatically recognized by most assemblers, but if not then replace each "SVC number" with:

```
LD A,number
RST 40
```

The labels and the sequence of the program's segments are admittedly rather

```
      INC HL ;HL=>relocated header
      LD (IX+2),H  ;stuff DCB vector to filter
      LD (IX+1),L
      LD (IX+0),47H   ;stuff DCB type: filter, I/O/C
      LD HL,DONE$
      SVC 10    ;@DSPLY: display message
      LD HL,0   ;exit code: okay
      RET
DONE$ DEFM   'SIMPLE filter installed'
      DEFB   13

NOSET LD HL,NOSET$
      SVC 10    ;@DSPLY: show error message
      LD HL,-1  ;exit code: error
      RET
NOSET$  DEFM   'Must install via SET!'
      DEFB   13

      END LOADER
```

personal: for example I like to have the body of the module at the beginning of the listing, so I can immediately see what it does. But readers familiar with filters will easily recognize the standard components described in Roy Soltoff's bible "*The Programmer's Guide to LSDOS/TRSDOS Version 6*". Filters are just like any "memory module" files: the LOADER relocates the HEADER plus FILTER into high memory (adjusting any fixed addresses by the distance of the relocation) and lowers HIGH$ to protect the module. However, one peculiarity of filter loaders (and drivers for that matter) is a test of the C-flag and the jump to an error message if the program was not invoked via SET command (lines 3-5 of the LOADER). We'll get back to it later.

The first idea in our effort to make the installation simpler for the user was this: after the loader has installed the routine, why couldn't it also process a FILTER command? This way the user would need to enter only one command from DOS, such as:

```
SET *AA SIMPLE
```

We know that the SET command established a Device Control Block (DCB) for the filter and stored its name (such as AA in our example) in bytes 6 and 7 of that DCB. Since register IX already points to DCB, we can easily fetch that name, stuff it into a FILTER command string, and execute that command. So let's rearrange the end of the loader (Listing 2).

```
;LISTING 2
      LD (IX+0),47H   ;stuff DCB type: filter, I/O/C
      LD L,(IX+6)  ;get device name
      LD H,(IX+7)  ;and stuff it into string
      LD (FILTER$+12),HL
      LD HL,DONE$
      SVC 10    ;@DSPLY: display message
      LD HL,FILTER$   ;point to command
      SVC 24    ;@CMNDI: execute and exit
DONE$ DEFM   'Installing SIMPLE '
FILTER$  DEFM   'filter *PR *$$'
      DEFB   13


; LISTING 3
NOSET LD HL,NOSET$ ;point to command
      SVC 24    ;@CMNDI: execute and exit
NOSET$  DEFM   'SET *AA SIMPLE'
      DEFB   13
```

The loader will replace the "dummy" characters $$ with the device name specified in the SET command. Since a filter normally is intended for one particular device (printer in this example), the name such as *PR can be pretty safely hard-coded into the loader. The "done" message is combined with the "filter" command string and displays both the name of the installed routine and the names of the involved devices.

So far so good. But once we were on the simplification binge, the next question was: why bother the user with the SET syntax anyway? Could the installation of the filter be done by simply entering its name just like any DOS command or program? Well, maybe. Remember that the loader checks the CÄflag and jumps to NOSET when bit 3 is zero. This may happen when a creative user enters SIMPLE/FLT. The DOS will execute it like any other program, but the filter wouldn't work without a "device" established by SET. Most likely it would bomb the system, because we'd "stuff" all kinds of funny bytes into wrong places. So it's a good safety valve.

But instead of complaining "Install via SET", the NOSET code could perhaps make itself a bit more helpful and just do it without talking so much. So let's revise this code snippet (Listing 3).

Almost home. The user needs to enter only:

```
SIMPLE/FLT
```

and the rest happens automatically. In effect the program gets executed twice: first to find that the SET command was missing, and then again to do it right. But since filter files are typically very short, the time is of minor concern compared with the convenience. Convenience? Then why bother with the FLT extension? Consider: the SET command assumes "default" extensions FLT or DRV, but any extension can be used if it's spelled out. So if we rename our file SIMPLE/CMD, and add the CMD extension to the "noset"

string:

```
'SET *AA SIMPLE/CMD'
```

then the user needs to enter only:

```
SIMPLE
```

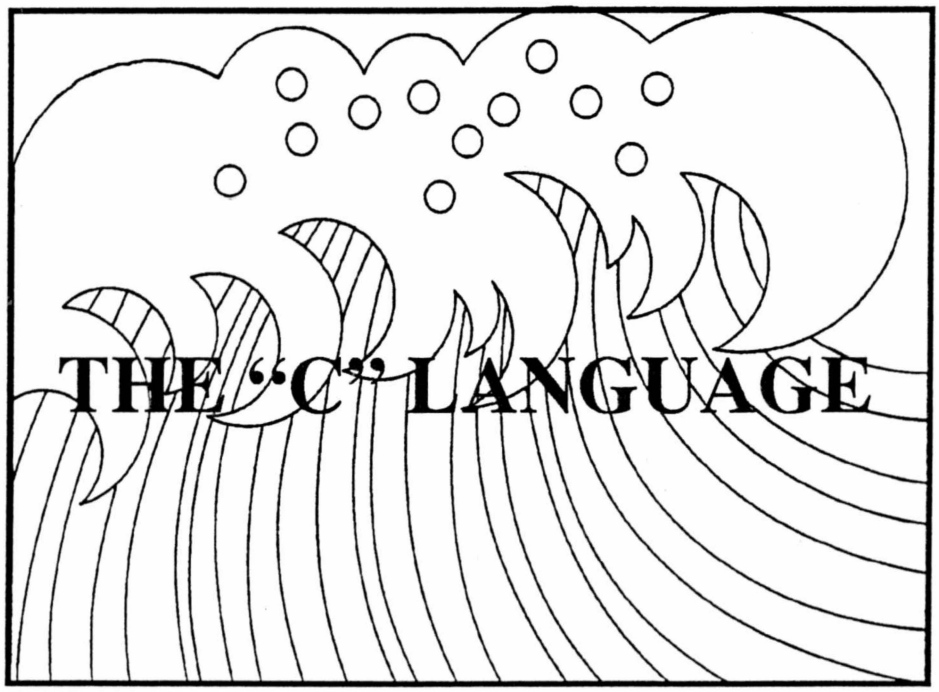
Simple indeed. No asterisks, no syntax, no fuss.

There is a problem, though, if the name which we have hard-coded (such as our *AA) is already used by some device. The installation fails with another cryptic complaint "Device in use". Of course we can always do a "manual" SET, using any device name (and now remembering to spell out the CMD extension). But that brings back the expletives and kills the whole idea. A smarter deal would be to prevent the "in use" error from happening in the first place. So let's revise the NOSET code one more time (Listing 4).

Now, before executing the SET command, we cycle through names AA, AB, and so on, and use SVC 82 @GTDCB to find a name which is not assigned yet to any device. We could just as well start with PA or VA or whichever initial letter seems suitable for "mnemonic" purposes. But what if we run up to Z and still find no free name? Give me a break! If you managed to have 26 devices in the system, all starting with the same letter, you deserve the trouble.

In a full-blown "filtomat" for public distribution we'd have a few other embellishments (making sure that the same filter is not installed twice, high memory test, copyright notice, and such). But otherwise that's all there is to the basic scheme of self-installing filters, invented under the practical pressure to supply customers with simpler software.

```
;LISTING 4
NOSET LD DE,(NOSET$+5)
INUSE INC D   ;try AA, AB, AC, etc
      SVC 82    ;@GTDCB: check if exists
      JR Z,INUSE   ;yes, try next
      LD (NOSET$+5),DE
      LD HL,NOSET$ ;point to command
      SVC 24    ;@CMNDI: execute and exit
NOSET$  DEFM   'SET *A@ SIMPLE'
      DEFB  13
```



The following was originally written by Earl 'C' Terwilliger, Jr. and published as a 6-part series in The LDOS Quarterly and subsequently the LSI Journal. It has been re-edited and brought up to date by Roy Soltoff for publication in The MISOSYS Quarterly.

A brief history of "C"

"C" was developed by Brian W. Kernighan and Dennis M. Ritchie of BELL LABORATORIES. *"THE C PROGRAMMING LANGUAGE"* authored by Kernighan and Ritchie is the ultimate reference document for the "C" language. A wide range of computers now have "C" compilers. (TRS80, IBM PC, PDP11, IBM/370's, etc. "C" runs on various operating systems.) "C" has been called a systems programmers language since it is useful in writing operating systems. In fact, the UNIX operating system (a trademark of BELL LABS) was written in "C". Frequently, any given implementation of the "C" language references a particular UNIX version. (You don't, however, have to write an operating system to take advantage of "C".)

"C" lends itself well to *structured programming*. Structured programming does not mean the absence of the GOTO statement. In fact, the "C" language does implement the goto statement. However as Kernighan and Ritchie (K&R) state in their book, the GOTO is "infinitely-abusable". When I discuss the control and flow of a "C" program I'll talk more on the topic of structure. (Not to be confused with the C concept of structures, i.e., records.)

The 'Basics' of C

"C" was designed to be a portable language. Its not very heavy at all! Many "C" programmers will say, it is light in some features. Most programmers are amazed when they see how relatively few identifiers (statements or keywords) that "C" has. Here is all there is:

auto	break	case	char
const	continue	default	do
double	else	enum	extern
float	for	goto	if
int	long	register	return
short	signed	sizeof	static
struct	switch	typedef	union
unsigned	void	volatile	while

As you can see, many of the "C" statements are the same as those found in other languages. (For examples: if, for, goto, etc.) Did someone say they can't find the keyword enum in the back of K&R's book? I'll pretend I didn't hear that for now. New things come later!

"C" hereinafter referred to as C (I got tried of typing ""), is characterized as a "low level" language. A C source program is compiled (assembled) into an executable machine language program. This is not, however, why it is called a low level language. It is called that because it deals with data items much the same way as machine instructions do. Machine instructions deal with characters (bytes), numbers and addresses. These objects are what C deals with also. To illustrate this, we'll take a look at C's data types.

There are only four data types:

- **char** a single byte (one character)
- **int** an integer
- **float** floating point (single precision)
- **double** floating point (double precision)

The int data type can be further qualified to be short int, long int or unsigned int. The length of these number data types is specific to the machine for which the C compiler was designed.

You might be wondering, at this point, about how "high level" tasks are performed. Or for that matter, how is any task accomplished which other languages perform but for which you don't see a C vocabulary word? You say you don't see any read, write (I/O), string manipulation, or array processing statements in C's above listed vocabulary? I expect not! Such things are specific to a particular machine. You can of course invent functions to do these "high level" tasks and explicitly call them in C. Usually you don't have to invent

them, however, since they are provided as functions in the C run time or the installation (computer specific) library. The installation specific functions, run time functions and functions of your own design aid in creating a more structured program.

One might expect (and rightly so) that the main body of the structured program is also a function! It is in fact and is called main(). (The parentheses () are used to denote main as a function and can optionally enclose variable names representing parameters passed to the function.) There must always be a main() function in your C program as it is always the entry point.

Braces "{}" enclose all of the statements that make up a function. C statements are expressions, such as x = 2 followed by a semicolon. The semicolon is used as a statement terminator. Statements may be, if you wish, grouped together (compounded) into blocks. A block is a statement or many statements which are enclosed in braces. This block or compound statement is treated as a single statement.

Braces surrounding the statements of a function, such as main() also form a block or compound statement.

Functions are invoked by their name followed by an optional argument list in parentheses. Taking a look at a simple C program, the use of the all these symbols "() {}" can be illustrated:

```
main()
{
/* C sample program */
/*to print HELLO!   */
/* and return back to */
/* the DOS prompt */
   printf("HELLO!");
   exit(0);
}
```

Note that comments in a C program are delimited by a "/*" and an "*/". In this sample program three functions are referenced; main(), printf(), and exit(). The exit function passes back to the operating system the return code of 0. This function is unnecessary in this sample program since the program would "fall through" and end normally at the last brace without it. (It is usually a better practice to return a value upon ending a function.)

The printf() function prints the values passed to it in the format specified. (Its counterparts in the FORTRAN language are the WRITE and FORMAT statements; a BASIC counterpart would be PRINT.)

The arguments of a function are passed as copies of the values of the arguments. This is "call by value" versus "call by reference". The function gets its own copy of the variable and can't change the original passed value held by the caller. "Call by reference", that is, passing the addresses of the variables can be achieved in C if pointers (addresses) of the variable arguments are passed. Having the address of the variable, the called function can then change the value. This "call by reference" is actually how array names are passed. When an array name is passed as a variable, it is actually the address of the first element of the array. (It would be impractical to copy the entire contents of an array and pass these separate copies as is done with other variables.)

Any of the four data types (int, char, float, double) can be represented in C by a variable. Numeric and character constants are also used in C as data values. A variable is symbolically represented by a name. This name is composed of letters and digits. The first character of the name must be a letter and although large names are allowed, only the first 8 characters of the name are significant. A C convention is to use lower case for variable names and upper case for symbolic constants. NOTE: C keywords (reserved words) must be in lower case.

Variables must be declared before they are used. In declaring a variable, the data type the variable is to represent is stated. For examples:

```
char c, d, e;
int x;
short b;
string[100];
```

c, d, and e are declared to be each a single character. x is declared as an integer. b is declared as a short precision integer, and string is a character array of 100 characters. (C numbers the elements starting at 0. So this array has elements 0 through 99, each one character in length.)

Variables not only have a type associated with them but also a storage class. The scope ("lifetime") of a variable is the part of, or range of the program in which the variable is defined. There are basically four types of storage classes for variables: extern, auto, static and register. An important concept to introduce before explaining these four types is the difference between the declaration and the definition of a variable. When a variable is declared, properties of the variable are assigned (type, size, etc.). When a variable is defined (done only once) storage is assigned. Except for use with external variables these terms (definition and declaration) are almost synonymous.

Automatic (auto) variables are variables "local" or "internal" to a single function only. This is the default for variables declared within a function. Auto variables have a scope or "lifetime" only within the braces or block in which they are defined. Variables declared as or defaulted to auto, appear as a function is called and disappear when the function ends. Since they have this dynamic "local" nature they are said to be "automatic".

External (extern) variables are "global" in nature. They are permanent and are accessible throughout the entire range of a C program. They can be shared between functions of a single C program and between many C source programs. Only one of the C programs would define the variable, the others would have to only declare it as extern. An external variable defini-

tion appears outside of any function. This actually defines the variable as external (extern). External variables are defined outside of any function and declared in the function which uses them.

Static (static) variables are stored in a fixed memory space. They can be external or internal in nature. When declared to be internal in nature they are like auto variables except they remain in existence in their fixed space. External static variables are global in nature but are only accessible within a single C source program. Static variables represent private permanent storage.

Register (register) variables are stored in machine registers. They are used to store heavily used variables in order to improve performance of the C program.

Here is a sample of some variable definitions and declarations:

```
auto int c;
extern char b;
register r1;
static  x1;
```

As you can see, the storage class and data type can be used to specifically determine the properties of a variable.

Expressions, functions, and operators

Functions in C are analogous to subroutines in other programming languages. They conveniently group commonly used expressions together. Frequently used logic instructions, instead of being typed in multiple times throughout the body of a program, need only be typed in once as a function. The function can then be called whenever its logic process is needed. It is thus that functions can hide confusing details in the main body of the program. Following the main logic flow is then much easier. The programmer can see immediately what is going on and yet need not be concerned as to how things are being done (C lends itself well to structured programming techniques).

In C, a function, like a variable name, has associated with it a storage class and type. A C function with no explicit declaration is by default external (extern). External functions can be called from multiple source files. A function may also be declared as static. Remember, if declared as static, a function can only be called from within the source file where it is typed in (defined). The type of a function can also be specified. Are you thinking that functions are a collection of expressions, and wondering how can they be assigned a storage type? Well, actually, the type associated to a function refers to the value, if any, it returns. Functions, unlike variables, need not but can be declared before they are used. The C compiler knows the difference between an undeclared variable and a function by the left parenthesis '(' immediately following the function name. Here are some examples of declaring a function:

```
static char lnth(a,c);
void compute(len, wid);
int test();
```

The function lnth is declared to be only known within a single source module and returns a character value. The compute() function returns no value whatsoever. The function test is known among multiple source modules and returns an integer value. It could have also been declared as:

```
extern int test();
```

The optional return statement is how the function returns a value back to its caller. It is optional; i.e., the function does not have to pass back a value. Even if the function does not return a value back to its caller, it is good programming practice to include the return statement. If no return is found, control of the function "falls thru" to the end of the function by reaching the ending right brace. Any expression (value) can follow the return statement. Examples of the return statement:

```
return;
return (0);
return ('x');
```

The first example of the return statement returns without passing back a value. The return (0); which could also be written as return 0; returns back the value zero. The last example returns the character 'x'. Although we haven't learned about the expression below, I'll let you ponder over it just to show how an expression can be used in the return statement:

```
return (a ? b : c);
```

Functions can also, optionally, be passed parameters. Remember that these parameters are passed by value rather than by reference. Except for arrays, and other parameters representing addresses of variables, each function gets its own private copy of the variable. These parameters are enclosed by the mandatory parentheses denoting a function. It is the parentheses immediately following a variable name that denote it as a function rather than a variable (data type). The optional parameters passed to a function can be declared by default but it is a better programming practice to explicitly declare them.

The left and right brace "{}", which enclose the expressions or logic performed by the function, are the next ingredients to completing a function. The expression(s) enclosed by the braces is/are called a block. (Blocks are not necessarily limited to use in functions. As we will see later, they are also used in the main body of a program. Enclosing braces are merely used to associate an expression or groups of expressions as a single entity.)

A complete example of a function is:

```
int prtval(c)
   int c;
{
   printf("The parameter
value passed was %d",c);
   return;
}
```

The function prtval is passed a parameter "c" which is declared to be an integer. Within the braces of the prtval function, another function printf is called. The printf

function is part of the standard C function library. It is passed two parameters in this example. The address of the string of characters enclosed in double quotes is the first parameter. The second parameter passed to printf is the value of c. (More will be discussed later about printf and standard C functions.) If necessary, functions can call themselves. This process is called recursion.

As previously mentioned, the main body of a C program is itself a function called main(). It can call functions as needed to perform designated tasks. This main function (the program itself) can have parameters passed to it. If parameters are passed to the C program, the following syntax is used to declare them and the main() function itself:

```
main(argc,argv)
   int argc;
   char *argv[];
{
   program statements
}
```

You are probably wondering, why only two parameters inside the "()" after main? Do I hear you asking what happens when I type in a command to tell the operating system to execute my program and I pass it more than two parameters? You should be asking! For example:

```
myprog p1 p2 p3 p4
```

Since this is a command line given to the operating system, there are actually five parameters or arguments. The program name itself is the first parameter and the four others are: p1, p2, p3, and p4. The main() function for the C program, myprog, would be coded as is done in the example shown above. The declarations for the variables argc and *argv suffice for all parameters passed to the C program in the command line. The variable argc contains the number of parameters passed (including one for the program name itself). The variable argv is actually a pointer to an array of pointers. Each element of the array (one for the program name and one for each successive parameter) is actually a pointer to the parameter. The "*" before the variable argv is a unary operator. It says that the variable it precedes contains an address. It uses this address to fetch the contents sto!)

Variables, Constants, and Expressions

Let's move on to the next topic, that of variables, constants, expressions and operators. Variables in C, and in any language, are used to manipulate data in storage. A variable name is composed of letters and characters and optionally the "_" character to improve readability of the name, as you can see:

```
char byte_of_storage;
```

Don't forget to define (declare) variables before you use them. Also, don't choose a variable name that is the same as a C reserved word (keyword or statement). If you do, don't worry, the compiler will tell you!

Constants are used in C, for the same reasons they are used in other languages. C allows for several types of constants, i.e., number, character and string constants.

It is usually good programming practice to use a special feature of the C compiler to define constants. As you'll discover, "equating" a constant to a name will make it easier to change later on. Just update it where it is "defined" and every occurrence of it will also be updated automatically by the C compiler at compile time. Here are some examples of the C compiler directive #define:

```
#define MAXIMUM 1000
#define TRUE 1
#define CR 015  /* Octal
value for a carriage
return */
```

Note that constants, if given a name, are generally by convention represented by upper-case names. The #define compiler directive functions to the C compiler as an EQUATE directive functions to an assembler. (One line macros!) In the above examples, wherever MAXIMUM is found in the C program it is replaced with 1000. Hex number constants are preceded with a "0x" or "0X". Octal number constants are preceded by just a 0. Another compiler directive can be used to include a file containing multiple #define statements. For example:

```
#include "file.ext"
```

This tells the compiler to include the contents (statements) in the file "file.ext". Usually there are many "standard" constants, declarations and/or expressions which you will include in most all of your C programs. This being the case, included with a C compiler is usually a "standard header" file of the most common constants. This file is what you will #include in most of your C programs.

A character constant is formed from a single character enclosed in single quotes. Certain special characters are represented with an "escape sequence" to denote it as a special character. Here are some examples:

```
'x'   /* single lower-
case character x */
'A'   /* single upper-
case character A */
'\r'  /* return  */
'\n'  /* newline  */
'\0'  /* null   */
'\t'  /* tab    */
'\015'   /* return  */
'\\'  /* backslash */
```

The last example shows how to generate any character you want. It is the "\" followed by one to three octal digits. A string of characters is represented by characters enclosed in double quotes. In C, a string is always terminated with a NULL or '\0' character. It need not be typed in the string itself since the C compiler adds it on automatically. An example?

```
"Earl C. Terwilliger" /*
My name as a string */
```

Note: the above string has a length of 19 and a size of 20. Don't forget the terminating NULL added by the C compiler!

Operators

Operators are the next topic. They specify what is to be done to variables and constants. Operators when combined with variables and constants are called expressions. As an expression is evaluated, there is a precedence or order of evaluation. It is important to know the order of evaluation when the different types of operators are combined in an expression, especially if you want a correct result! Below is a chart of the operators available in C. It shows the relative order (level) of precedence, and the associativity of operators of equal precedence. (The associativity is given to show how expressions are evaluated if operators of an equal level of precedence are found side by side.) The actual workings of each operator (with some sample expressions) will be covered later. The Operator Precedence chart is presented below.

Let's take a look at a sample C program which is a multiple file LIST utility. The syntax for running it is:

```
LISTEM file1/ext file2/
ext file3/ext
```

Each file name passed in the LISTEM command will be passed to DOS to execute for each file name parameter. The commands built and passed to DOS for the above example would be:

```
LIST file1/ext
LIST file2/ext
LIST file3/ext
```

As you look at the sample program below, review the concepts just learned. Not all of the programs statements will be clear, but see how much you can understand. Looking at the comments may prove beneficial.

```
/* Multiple File LIST Utility                    */
/* Author - Earl C. Terwilliger Jr.              */
/* Written for LDOS and the LC C compiler        */
#include stdio.h      /* MC standard header file  */
#option INLIB        /* Special MC compiler option */
#define CLEAR "\x1c\x1f"   /* Codes to clear screen */
main(argc, argv)
   int argc;               /* Declare parameters */
   char *argv[];
{
   int c, rc;         /* declare AUTOMATIC variables */
   char buf[100];
   puts(CLEAR);       /* Call to CLEAR function   */
   puts("Multiple File LIST Utility\n\n");
   if (argc < 2)      /* Must have at least 2 parms */
      {
      puts("Syntax: LISTEM file1 file2 ...   \n");
      exit();         /* Exit back to LDOS       */
      }
   while (argc > 1)  /* More parameters still?   */
      {
      buf[0] = 0x00; /* Set string to NULL ""     */
      ++argv;        /* Go to next parameter     */
      strcat(buf,"LIST "); /* Connect the 2 strings */
      strcat(buf,*argv); /* Connect "list" and parm */
      puts(buf);     /* Display the string on video */
      rc = cmd(buf); /* Execute DOS cmd and return */
      printf("\nReturn Code was %d\n",rc);
      --argc;        /* One less to deal with now! */
      }
}
```

Operator Precedence Table

Operators	Level	Type	Associativity
() [] -> .	15	Primary	left to right
! ~ ++ - -	14	monadic	right to left
(type) * & sizeof	14	monadic	right to left
* / %	13	arithmetic	left to right
+ -	12	arithmetic	left to right
<< >>	11	shift	left to right
< <= > >=	10	relational	left to right
== !=	9	relational	left to right
&	8	bitwise logical	left to right
^	7	bitwise logical	left to right
|	6	bitwise logical	left to right
&&	5	logical	left to right
||	4	logical	left to right
?:	3	conditional	right to left
+= -= *= /= %=	2	assignment	right to left
|= ~= &= >>= <<=	2	assignment	right to left
,	1	comma	left to right

Operators

The C language is very rich in its capabilities of data manipulation. Operators perform this data manipulation. In the C language, operators are usually grouped into categories. These categories (or types) are shown in the operator chart.

Operators specify what is to be done to variables and constants. Operators, when combined with variables and constants, are called expressions. When an expression is followed by a semicolon it becomes a statement. In the next article we will see more on expressions, statements and the logic control and flow in the C language.

Before I go on, I would like to clarify a few things you will encounter in the C programming language in dealing with expressions and operators. Basically, these points involve the concepts of lvalue, rvalue, "side effects" and type conversions.

An *lvalue* (left hand value) is an expression which references storage in the computer. In C, an object is a manipulatable area of storage. An lvalue is an expression which refers to that object. An lvalue is the only value which can be on the left side of an assignment statement. An lvalue or an rvalue can be found on the right side of an assignment statement. An expression which is not an lvalue is sometimes called an rvalue. An rvalue doesn't refer to an object (area of storage). An example? Consider the following C statements:

```
int c;
c = 0;
```

For Z80 based machines, if the HL register pair was used to obtain the value of c from the stack, you might see the following instruction (among others) generated as a result of compiling the above C statements:

```
LD HL,0  ;Load HL pair
with 0
```

The 0 is contained in the instruction and doesn't take up any data storage.

Side effects come into play when there are no explicit C rules as to how an expression is to be evaluated. In this article, a chart is used to summarize the rules of precedence and associativity for the various operators.

If no rule of evaluation applies, the C compiler is free to evaluate an expression or sub-expressions as it sees fit. The results (expected or not) are called side-effects. To overcome side-effects, break up larger expressions or statements and use explicit "temporary" lvalues.

When expressions involve arithmetic with different data types, a conversion (or promotion) to a common data type occurs. Since both data types are of a common type, the result is also of that type. Since "lower" data types are converted to "higher" types, the process is called promotion. Of course there is a set of rules for this conversion or promotion. The "ladder of promotion" (if I can be allowed to invent a phrase) is shown to illustrate this promotion:

double
float
unsigned long
long
unsigned int, unsigned short
int, short
unsigned char
char

Besides arithmetic expressions, type conversions can take place when a function is called. After all, a function argument is an expression. Conversions can also take place across an assignment statement. When this happens the value on the right side is converted to the type on the left side. The above promotions can occur or the reverse process can also occur. (Demotion!) A word of caution! The C language guarantees any character in the machine's standard character set will never be negative when used in arithmetic expressions. However, C does not specify whether variables of type char are signed or unsigned. So, be careful when using various bit patterns (other than the standard character set) because you may be dealing with negative values. Later, we will see how to force a type conversion using the cast operator.

Now that these points have been mentioned, let's get back to more on expressions. As an expression is evaluated, there is a precedence or order of evaluation. It is important to know the order of evaluation when the different types of operators are combined in an expression, especially if you want a correct result! Remember the chart of the operators available in C presented previously. It shows the relative order (level) of precedence, and the associativity of operators of equal precedence. The associativity is given to show how expressions are evaluated if operators of an equal level of precedence are found side by side in the same expression.

Since variables must be declared before they are used, I will declare a few so they can be used in our sample expressions.

```
int a, b, c, d;
```

Variables can be initialized in their declaration statement. As you might guess, an operator (the equals sign) is needed to accomplish this. For example:

```
int a = 10;
int b = 5;
int c = 2;
```

An easier way to initialize the above variables is:

```
int a=10, b=5, c=2;
```

(Well... anyway it takes less lines in your program.) On the other hand, you may want to use a separate line for each definition in order to document the use of the variable by means of a comment string.

To accomplish the initialization, the variable name is followed by an equals sign and a constant. The constant's value serves as an initializer. Multiple initializations can be performed in a single statement (as is done above) when separated by a comma. The comma, used in this way as a separator, is not an operator and does not man-

date the rules of evaluation as does the comma operator. Some variable types, external and static, are automatically initialized to zero by default of the C compiler. However, as stated by K&R, it is better programming practice to state their initialization anyway. If automatic variables are not initialized, their values will be of an undefined value. (Garbage!) You should note that for external or static variables, the initialization is done only once.

For automatic variables which are initialized in a function, they are re-initialized each time the function is called.

I may need to declare a few more variables to illustrate all of the operators. However, I will do so when each individual operator is introduced. Let's begin!

Operators () [] -> .

```
printf("C is a very
useful language to
know!\n");
```

The "()" is called the function call operator. The left and right parentheses separate any optional arguments passed to the called function.

Square bracket "[]" operator(s) is used for arrays. In this example, an array of characters, whose name is buffer, is declared to have 100 elements. The "[]" is used to reference (index or subscript) elements of the array. In this example, the elements are buffer[0] through buffer[99]. In C the index starts at 0. Be careful about your use of these subscripts, since the C compiler does not check to see if the subscript you use is within the bounds of the array.

```
struct date { int month,
day, year;} birth;
struct date *pdate;

pdate = &birth;
birth.month = 10;
pdate->day = 3;
```

The "." and the "->" (period and arrow) operators are used when dealing with structures. The "." operator is called the structure member operator because it connects the structure name and a member name. Pointers to structures are very frequently used, due to the restrictions imposed by the C language. (These limitations imply that structures are not to be dealt with as a single unit. This limitation does not hold for pointers to structures. Therefore a shorthand operator, ->, is used to make pointers to structures and structure members easier to use. In the examples above, birth is a structure of type date, month and day are members of the structure and pdate is a pointer to a structure of type date. More later on structures! (As you can see, we haven't yet discussed the "*" and "&" operators!)

Operators ! ~ ++ -- - (type) * & sizeof

All of these operators are unary or monadic operators. This means they need only one operand.

```
!a
```

The truth value of "a" is reversed. If a was TRUE (non-zero) it is now FALSE (zero). If a was FALSE it is now TRUE. Since I declared a to be 10 earlier, after the !a, the result returned will be FALSE (zero).

```
~a
```

The one's complement of the operand "a" is returned. Just invert each bit of the operand. If a bit was a one switch it to a zero and if the bit was a zero switch it to a one.

```
++a --a a++ a--
```

These are the increment ++ or decrement -- operators. They may prefix or postfix the operand. If the operator prefixes the operand, its value is taken after the increment or decrement. If the operator postfixes the operand, its value is taken and returned by the expression before it is incremented or decremented. These operators do not necessarily mean to add one or subtract one from the operand. The increment or decrement value is based upon the storage TYPE of the operand!

```
-a
```

This is the unary minus operator. It reverses the sign of a. After -a, since a was declared to be 10, the result is now a -10. (There is no unary + operator.)

```
myfunc((double) c);
```

The variable c was defined as an integer above. If the function myfunc() needs to be passed a double precision floating-point variable, c can be coerced into that type. This construct of forcing a variable into another type is called a cast. In this example, c is cast into type double.

```
char name[] = "Earl C.
Terwilliger Jr.";
char *ptr;
ptr = &name[0];
```

The "&" operator obtains the address of an object. (The & operator applies only to variables and array elements.) The "*" operator precedes an operand containing an address. This address is then used to fetch the data stored there. A lot more will be said about these operators when pointers and arrays are discussed in greater detail!

```
sizeof(a)
```

This operator returns the size of any object. The value returned is determined by the number of units contained in the object. Each "unit" or "byte" has the same relative size as type char.

Operators * / % + -

```
d = a * b;
d = a / b;
d = b % c;
d = a + b;
d = a - b;
```

The *, /, +, and - operators perform the arithmetic functions of multiplication, division, addition and subtraction respec-

tively. Since integer division in C truncates the fractional part, use the & or modulus operator to obtain the remainder. (% can not be applied to float or double operands.) (Note that parentheses are also used around expressions to specify order of evaluation. Watch out! The C compiler can sometimes rearrange a parenthesized computation. For some associative and commutative operators (remember your arithmetic for * and + ?) the C language does not specify the order of evaluation. Use explicit temporary variables to overcome this, if in fact this re-arrangement makes any difference.

Operators << >>

```
a >> 2    /* a shifted
right by 2 bits */
b << 4    /* b shifted
left by 4 bits */
```

These shift operators perform left and right shifting of the bits in their operand. The number of bits shifted is the right operand. When an operand is shifted left, vacated bits are filled with a 0. Right shifting an unsigned operand causes the vacated bits to be filled with zeros. Right shifting a signed operand may be an "arithmetic" or "logical" shift depending on the machine and the C compiler.

Operators < <= > >= == !=

```
ret = a < c
ret = a <= c
ret = a > c
ret = a >= c
ret = a == c
```

Each of these operators takes two operands and returns the result of the comparison: a less than c, a less than or equal to c, a greater than c, a greater than or equal to c, a equal to c; the result is TRUE or FALSE (1 or 0). The result is based upon the relationship of the two operands. (Note: one of the more frequent errors by new C programmers is the failure to differentiate the == operator from the = operator. The == operator tests for equality while the = always sets equality.)

Operators & ^ |

```
res = a | 0x01
res = a & 0x0f
res = a ^ 0xf0
```

These operators provide C the capabilities of bit manipulations. Bits of the left operand can be ORed ("|"), ANDed ("&"), or exclusively ORed ("^") by bits in the right operand.

Operators && ||

```
&& /* and */
|| /* or */
```

These operators logically connect their operands. (They should not be confused with the & and | bitwise operators.) These operators can be used to connect expressions. For example:

```
(a == 2) || (a == 3)
(a != 0) && (c == 2)
```

Operator ? :

```
(a==b) ? (c=1) : (c=2)
```

This "? :" construct is a ternary (three) operator. It provides an alternative to the IF ... THEN ... ELSE... type logic. If the expression to the left of the "?" is TRUE then the expression immediately following the "?" is evaluated. If the expression to the left of the "?" is false then the expression after the ":" is evaluated.

Operators = += -= *= /= %= != ~= &= >>= <<=

```
a += 1
a -= 3
a *= 5
c /= 2
c %= 2
c |= 0x80
c ~= 0x02
c &= 0xf0
c >>= 2
c <<= 4
```

These operators represent the assignment operators. They were seen before without the equals sign. In the assignment expressions, `a op= b` is equivalent to the expression `a = a op b`. They make expressions such as: `a = a * 5` much easier when written as: `a *= 5`. Usually the resultant assembler (machine) code produced as a result of these operators is more efficient too. They are especially helpful if a long complicated expression is on the left of the = and needs to be used to the right of the =.

Operator ,

The comma is used in C as punctuation. It is seen separating expressions in declarations and it is seen separating arguments passed to functions. The comma separator is different from the comma operator. The comma operator is usually found used in a for statement. An example of a 'for' expression? OK!

```
for (i=1; i != 50; ++i)
```

Logic, control, and flow

The specific C language vocabulary words that will be used in this discussion are:

for	while	if	else
switch	break	continue	do
goto			

In conjunction with the above logic, control and flow vocabulary words, statements and blocks of statements accomplish the tasks designed into a C program. Let's take a look at these C vocabulary words and their use in a C program.

But first, a quick reminder about (expressions) statements and blocks! Remember, a C statement is an expression followed by a semicolon. For examples:

```
a = 24;
c = getchar();
printf("%d \n",e-18);
```

These are all examples of C statements. Each expression is ended with a semico-

lon. It is used in C as a statement terminator rather than a separator. (You might also note in the above example with the printf() function a general rule in C. Wherever it is permitted to use the value of some type of variable, it is also permitted to use an expression of that type. Hence the e-18 expression is used instead of having to assign it to some intermediate variable. You can save a lot of coding using this rule, but be careful! You can also make your program confusing!)

Whenever it is necessary to group statements (declarations, etc.) and treat them as one, they can be enclosed in braces "{}". This creates a "block" or "compound statement". This block enclosed by the braces is not followed by a semicolon even though the enclosed statements are treated as one. The need for blocks or compound statements will be seen as the C logic, control and flow vocabulary words are explained. Shall we begin as K&R does with the if statement?

The general format (syntax) of the if statement is:

```
if (expression)
   statement_1
else
   statement_2
```

(You will note the importance of differentiating between a statement and an expression!) The if-else statement is used to make decisions. The expression is evaluated. If it is true (i.e., has a non-zero value) then statement_1 is executed. The else is optional. If it is present and the expression is false (i.e., has a zero value) then statement_2 is executed. Since the else is optional and can be omitted, you could be confused by the following:

```
if (a == 2)
   if (c == 2)
      d=2;
else
   d=4;
```

The rule in C is that the else is associated with the closest previous else-less if. Because of the way the above compound if statement is indented, you may be led to falsely believe that the else should be paired with the if it is aligned with. Another important point to mention here deals with indentation. It is generally practiced to have the else aligned with the if to which it belongs. Thus the following is more readable:

```
if (a == 2)
   if (c == 2)
      d=2;
   else
      d=4;
```

If the else was in actuality to be paired with the first if, then the {} can be used to force the proper association as follows:

```
if (a == 2)
   {
   if (c == 2)
      d=2;
   }
else
   d=4;
```

The else is thus paired with the first if. The second if is contained in a "block" and is the statement_1 referenced in the general format of the if statement. Of some note also is the placement or "style" of placing the braces and their alignment in the above if else statement. Each C programmer develops a way of placing and or aligning if-else, else-if and the braces. Consider the following two examples:

```
/* EXAMPLE 1 */
if (expression) state-
ment
else if (expression)
statement
else if (expression)
statement
else  statement

/* EXAMPLE 2   */
if (expression)
   statement
else if (expression)
   statement
else if (expression)
   statement
else
   statement
```

Both examples work the same but are of different styles. Perhaps the most popular or common style (used in the K&R book) is represented via the second example. Example 1 may look nice too, but consider how long the actual expressions and statements may be. If they are quite long, the style of example 2 may appear nicer. Whichever style (method) you choose, it is a good rule to be consistent.

If you noticed, the above two examples demonstrate a generalized way of writing a multi-way decision. If any expression is true, its associate statement is executed and the whole else-if chain is ended. If none of the expressions are true then the statement after the last else is executed. This represents the "default case". Any of the statements can be a block of statements in the braces. The last else could also be missing and there would be no default statement executed.

Another way of making a multi-decision in C is with the switch statement. The syntax for the switch statement is:

```
switch (expression)
   {
   case constant:
      statement;
      break;
   case constant:
   case constant:
      statement;
      break;
   case constant:
      statement;
      break;
   default:
      statement;
      break;
   }
```

The switch statement is followed by an integer expression and a block enclosed in braces. The logic of the switch statement is to evaluate the integer expression and compare its value to the constant case

values. Each case is "labeled" by a constant expression (usually an integer or character constant). If a case matches the value of the expression, that case begins the execution. Statements after that case are then executed. If a break statement is encountered the switch statement (block within braces) is exited. If no cases match the expression then the default case begins the execution. The default case is optional. The cases and default can occur in any order, but the cases must all be different. If no cases match and no default case is present, nothing happens at all. (Nothing happening at all has been described as "the sound of one hand clapping"). It is good programming practice to put the break statement at the end of a case. If a break is not present, execution "falls through" to the statements which follow. This may not be the desired action! An example of the switch statement follows:

```
switch (answer)
   {
   case 'y':
   case 'Y':
      printf("The answer
was YES!");
      break;
   case 'n':
   case 'N':
      printf("The answer
was NO! ");
      break;
   default:
      printf("Enter only
Y or N!\n");
      break;
   }
```

The above switch statement could possibly be used to test for a Y<es> or N<o> reply. Note that it uses a case for the upper or lower case possible responses. You are no doubt asking what happens if the default case is executed and you want to allow another response until Y or N is entered? Well, you could use the C statements which allow looping! Looping (executing a statement or groups of statements a given number of times) can be accomplished in C via four basic ways: for, while, do - while and goto.

The syntax of the while statement is:

```
while (expression)
   {
   statement
   }
```

If the expression after evaluation is true, the statement is executed. The expression is then re-evaluated and if true statement is executed again. This process is repeated until expression is false (zero).

The syntax of the for statement is:

```
for (expression 1; ex-
pression 2; expression
3)
   {
   statement;
   }
```

Expression 1 and expression 3 are typically assignments or function calls and expression 2 is an expression to be evaluated as true or false (a relational expression).

Another way to write the logic of the for statement using while is shown as follows:

```
expression 1;
while (expression 2)
   {
   statement;
   expression 3;
   }
```

From the explanation of the while, you can see how the for statement works. In the for statement the expressions could be multiple expressions separated by commas. For example:

```
for (i=0,j=0; s[i]; ++i)
   {
   if (s[i] == 'a')
      ++j;
   }
```

Whether you use the while or the for statement is just a matter of choice. Typically the for is used for simple initialization and re-initialization. It is analogous to the FORTRAN DO loop or BASIC for-next statements.

The syntax of the do-while is :

```
do
   statement;
while (expression);
```

The difference between the do - while and while is a subtle one. With the do - while, the statement is always executed at least once. The expression is evaluated at the bottom of the loop instead of at the top.

Remember the break statement from the switch? It can also be used in the for, while or do-while to exit. Another statement, the continue statement is related to break. It does not exit from a for, while or do-while statement but causes the next iteration of the enclosing loop to happen. An illustration for you to ponder:

```
for (i=0,j=0; s[i]; ++i)
   {
   if (s[i] == 'a')
      continue;   /*
Skip this character */
   if (s[i] == '\n')
      break;   /* Exit
for if new line */
   ++j;
   }
```

In the above for statement, the only ways for it to end are if `s[i]` equals 0 or the newline character. Note that the relational expression is `(s[i] != 0)`, but it can be and is shortened in this example to `(s[i])`. That's because any zero value is interpreted as FALSE and any non-zero value is interpreted as TRUE; good programmers take advantage of this fact!

With the above new C language commands, you can perform various logic patterns, and control the flow of a C program. Another flow control C statement is the goto. The object of the goto is a label. A label has the same form as a variable name but is followed by a full colon. The goto and the label to go to must be in the same function. The use of the goto is not

recommended, except for possibly branching out of some heavily nested logic.

Initialization, Blocks, Pointers, and Arrays

Several computer languages are block-structured in the sense that they allow functions to be defined within other functions. C does not allow this. In C, functions are always "external" since they are not inside of other functions. I am alluding to the fact that functions are blocks of C code and a block is enclosed via braces "{}". These braces enclose functions and other blocks. After the "{" comes variable definitions, if any. Variables in C, are thus defined in a block-structured manner. Variables can be declared following the brace that begins any compound statement. Also after the brace that begins a function, variables can be declared (defined).

If more variables need to be declared, later in the function, they can be, by declaring them after the left brace which begins a block. These variables can even have the same name as other variables. Their declarations "supersede" the identically named variables in outer blocks. They exist only within the block in which they are declared. Don't forget or confuse what you have learned previously about variable storage class and what you are learning now. The above comments on variables declared within blocks hold true for external variables too. Now can we look at how variables can be initialized? (No freedom of choice, is there?)

Initializers

If you would like to assign an initial value to a variable when it is defined, C will allow it. As an interesting point, C does initialize certain variable classes for you. If you do not specifically assign an initial value to an external or static variable, C will initialize them to zero for you. However, automatic and register variables are not initialized automatically for you by C. So, don't count on them containing anything worthwhile unless you specifically initialize or assign a value to them. An equals sign and a constant expression are used to initialize simple variables. (Arrays and structures are initialized differently, as we shall C later.) Here are some examples of simple initialization:

```
int a = 5;
int b = c = d = e = 0;
char g='x', h, i='y';
char f = 'f';
int d = 45 * 67;
```

As you can imply, this initialization saves "extra", sometimes unnecessary, assignment statements which assign a value to a variable. K&R call this shorthand for assignment statements. Remember what was just learned about blocks and how variables can be declared within them? Well, variables declared within these blocks (or functions) can also be initialized. This initialization takes place each time the function or block is "entered". External and static variables are initialized only once. (Are you wondering why this is? External and static variables are of different storage class and scope than automatic and register variables. Think about how and when these variables come into existence and when they go out of existence (if they do)?) Also, for automatic and register variables, the initialization can be done via any valid expression. This initializer is not limited to a constant expression.

Before I discuss how arrays can be initialized, shouldn't I discuss what they are and how they are declared (defined)? For example:

```
int number[10];
```

This declares an array of size 10. In essence, this is a "block" of 10 integers together. Likewise:

```
char name[12];
```

declares a block (an array) of 12 characters. Each member of the array is called an element. Each element is numbered or indexed. In C the index starts at zero. For example, in the number array above, the elements can be referred to individually via number[0], number[1], ..., thru number[9]. C also supports multi-dimensional arrays. For example:

```
int a[10][20];
```

This declares a two dimensional (rectangular) array. Elements of a multi-dimensional array, in C, are stored by rows. Viewing storage as linear, elements of the array are seen in storage order if the right most index varies the fastest. Now, how can arrays be initialized?

Arrays are initialized differently than other variables. Only external and static arrays can be initialized, automatic arrays can not be initialized. External and static arrays are initialized as shown in this example:

```
static int numbers[10] =
{ 0,1,2,3,4,5,6,7,8,9 };
```

Remember, in the absence of explicit initialization, all elements of external and static arrays are initialized automatically to zero.

In initializing external and static arrays, fewer initializers can be used than there are elements. In this case, the remaining elements will be zero. C also disallows more initializers than elements. Wouldn't it be nice to be able to repeat an initializer or just to initialize specific elements and ignore others? Well, sorry, C does not provide a means to do that.

Here is an example of a character array and its initialization:

```
/*
 ....5...10...15...20..
*/
static char me[] =
"E.C. Terwilliger Jr.";
```

Quick! How many elements does the array *me* have? (Use the comments ruler line to help you count.) Did you guess correctly with 21? Each character between the quotes is an element plus the \0 which is added by

the C compiler to terminate the string. Did you notice that the size of the array, i.e., the number within the [] was omitted? If you do not include it, C will compute the size of the array for you based on the number of initializers. Another way to initialize a character array is as follows:

```
char name[] = { 'E',
'A', 'R', 'L', '\0' };
```

Notice that it is so much easier to use:

```
char name[] = "EARL";
```

Are you thinking that the initialization of a character array is like a "string copy"? If so, be careful in your evaluation of the following statements:

```
static char msg[5];
msg = "TEST";
```

This is not a string copy! C does not provide any operator for string copying or dealing with an entire string of characters as a single unit. Also, msg is the name of an array, it is a constant. It is not an lvalue and the above expression using it as such is ILLEGAL! How then can elements of an array be assigned values? The answer is by individually assigning values to each element. To "blank out" a character array, examine the C code which follows:

```
char message[20];
...
for (i=0, i<20, ++i)
   {
   message[i] = ' ';
   }
```

Also, note that the message array does not necessarily have to be external or static. It could be an automatic array!

Pointers

Next, onward to pointers! A pointer is a C variable which contains the address of another variable. I can hear you thinking! You are no doubt asking, how does the pointer get the address? The unary operator & mentioned in an earlier part gives the address of its object. The & operator applies only to array elements and variables. Consider the following:

```
char a;
char *ptr;
...
a = 25;
ptr = &a;
```

In the expression: `ptr = &a`, ptr is assigned the address of a. By the way, there is no such thing as just a pointer. In C, pointers are always pointers to a particular data type. As shown above ptr is a pointer to type character. The "*" operator denotes indirection, it treats its operand as an address. It accesses this address to obtain the contents stored there. For example:

```
char *ptr, a, b;
b = 'x';
ptr = &b;
a = *ptr;
```

In the above examples, b is assigned the value 'x'. ptr is assigned the address of b. a is assigned the value of the character pointed to by ptr, which is 'x'. *ptr is a C mnemonic declared in this example to be a character. The combination of the * and ptr denote a character just like the above variable b does. When a pointer is declared, the type of data it points to is stated. The pointer is limited to point to data of that type. Also, pointers and pointer references are lvalues and can appear on the left side of assignment statements. Above, the pointer ptr is seen appearing on the left of an assignment statement. Below, *ptr is shown on the left of an assignment:

```
char *ptr, a, b;
b = 'x';
ptr = &a;
*ptr = b;
```

After the above statements are executed, a will contain the same value as b! *ptr is a pointer reference. In the case above it actually references a. ptr contains the address of a and *ptr references the character stored at the address in ptr.

Having the address of a variable is very useful. Remember from a previous part that C passes copies of variables as arguments to a called function. This is "call by value". The called function can not alter a variable in the calling function. (Actually, it could if the variable used in both functions was an "external" variable.) Now that you have learned about the & operand, you can use it to pass, as parameters to a function, addresses of (pointers to) variables. The called function can declare the arguments passed as pointers and alter the referenced data!

Looking back over the above discussion on arrays, do you remember the problem of assigning values to an array? Consider this, now that you are familiar with arrays and pointers:

```
char *myname;
myname = "Earl C.
Terwilliger Jr.";
```

This also is not a string copy! But it is a valid expression. myname is a pointer and it is assigned the address of the string! Comparing these two C statements with the ones shown to illustrate arrays, you should be wondering about the relationship between an array and a pointer. Actually an array name is a pointer expression. However, keep in mind that a pointer is a variable but an array name is a constant. If an array name is passed as an argument to a function, what is actually passed is the location (address) of the beginning of the array. (Using the & operator on just an array name is invalid. C does however, allow the & operator to take the address of an array element, for example &myname[4]. The & operator applies only to variables and array elements!)

A called function, when passed an array name as an argument, can declare the argument as a pointer and reference thru the elements of the array. Would you like an example?

```
main()
{
    static char myname[] = "Earl C. Terwilliger Jr.";
    char a;

    a = 'l';
    printf("%d\n",scount(myname,a));
}
scount (ptr,ch)
    char *ptr, ch;
{
    int c = 0;
    while (*ptr)
        {
        if (*ptr++ == ch)
            ++c;
        }
    return (c);
}
```

The function scount will return the number of occurrences of a given character in a given character string (array). The two parameters passed to it are the address of the string to search and the character to search for. If you follow the logic, pay particular interest to the *ptr++ expression. The value printed after the above code is executed should be 3!

(What? You don't believe me? Type in the code and try it out on your favorite C compiler.)

Pointers, Arrays, Structures, and Common Errors

Could you use some POINTERS on how to STRUCTURE better C programs? Oh? You thought when you read the word POINTERS and the word STRUCTURE that this part would really be discussing techniques for improving your C code? Ha! Well... OK, not to disappoint you, included in this part is a discussion of the most common errors or "things" not to do in a C program. Will that help?

Pointers and Arrays

You just saw an expression `*ptr++`. Were you puzzled? Remember back when the ++ and – operators were introduced? It was stated that ++ added one to its operand and – subtracted one from its operand. Be careful applying this to pointers. The "one" referred to which is added to or subtracted from a pointer is actually a scale factor. This scale factor is dependent on the type the pointer points to. That means it is scaled by a size equal to the data type length. This holds true for all "pointer arithmetic". (For example, in a Z80 based machine the scale factors are 1 for char, 2 for int.)

There are some rules to follow when doing arithmetic in C using pointers. It is legal to:

- add an integer to or subtract an integer from a pointer;
- subtract a pointer from a pointer;
- compare a pointer to another pointer.

All other conceivable arithmetic, including shifting or masking is illegal. (Note, a pointer containing NULL or 0 is a special case. The C language guarantees that if a pointer points to valid data, it will not contain 0. The 0 value is usually used to indicate an error condition. An example of this would be when a storage allocate function is called. This function may have been designed to return a non zero pointer to the beginning of the allocated storage. If storage can not be allocated it could return a NULL (zero) value indicating an error of some type occurred.)

Consider the statements below for the discussion following:

```
char *ptr;
static char a[5]="test";
ptr = a;
++ptr;
```

ptr is a pointer to type character. ptr is initially set to the address of the array a. This is written as `&a[0]` or simply a. Next, ptr is incremented to point to the next element of the array. This is written as `ptr++`. (Other possible ways to code it, in this example, could have been `*ptr++`, `*++ptr`, `*(++ptr)` or `*(ptr++)`. Note that `(*ptr)++` would create a different undesired result than `*ptr++`. The ++ and * operators are of equal precedence and associate right to left.) From the above statements, you can conclude that array subscripting can be done by incrementing a pointer. You can also conclude that the following two expressions are equivalent:

```
ptr = a;
ptr = &a[0];
```

(Note that in effect, an array name is a pointer expression. Note also that using pointers rather than array subscripting usually results in more efficient code.)

As general rules:

- `a[n]` is equivalent to `*(a+n)`;
- `*(&a[n])` is equivalent to `*(a+n)`;
- `&a[n]` is equivalent to `&a[0]+n` is equivalent to `a+n`;

Perhaps if I spelled out how to "pronounce" some of the expressions used in the general rules above, these rules might become more clear?

`&` means the address

`*`	means the data at
`a[n]`	means element n of array a
`&a[n]`	means the address of element n of array a.
`*(&a[n])`	means the data (element) at the address of element n of array a. This says the same thing as element n of array a
`*(a+n)`	means element n of array a
`a + n`	means element n of array a

Did the above help?

If you have been looking at some sample C programs, you may have seen by now that sometimes an array name is written as a[] or *a when used as parameters in a function. Rather nice don't you think? The function, when passed an array name, can treat it as an array, as a pointer or both. If you have some doubt, look at the C code below:

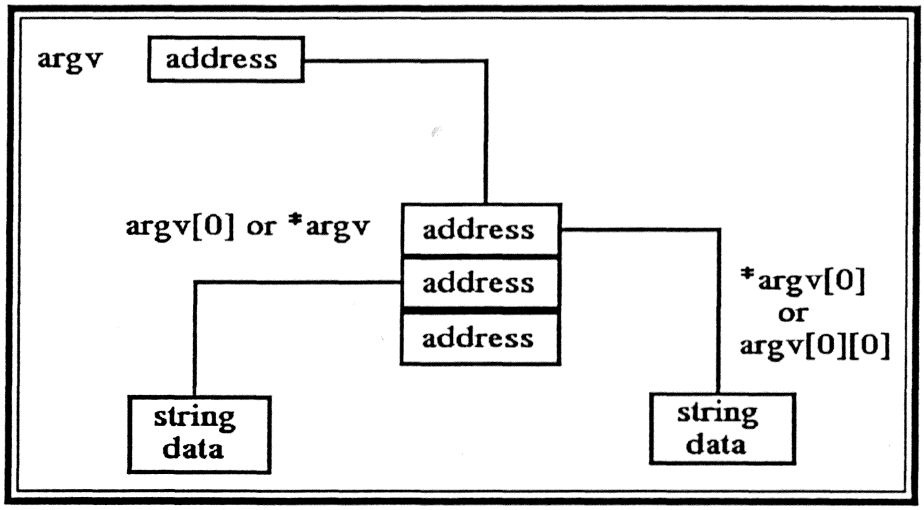
```
main (argc, argv)
   int argc;
   char **argv;
{
   if (argc < 2)
      {
      printf("Error - no
parameter was given!");
      exit(1);
      }
   ++argv;

printf("%c\n",argv[0][0]);
}
```

The arguments argc and argv are not new to you, they were described previously. As you noted, argv is treated as an array and as a pointer in the above program. Are you curious about the argv[0][0] expression used in the printf function? What will print is a single character, the first character of the command line argument after the program name. If the above program was called test, to invoke it and pass it an argument, you might type:

```
test -l myfile/dat
```

If you compile it and try it using the above invocation, you should see the - printed. (Try it with different argument values and different numbers of arguments.) Of what value is this? Well, actually this program might be used as part of a larger program and the argv[0][0] could be used to test for a "switch" such as + or - in front of a parameter. In the example invocation above, I included the myfile/dat parameter to suggest some possibilities for you to ponder!

The argv function parameter, is a pointer to an array of pointers. Here is a list of possible ways or forms in which you might see it used: argv, *argv, argv[n], *argv[0], (*argv)[0], argv[0][0]. Having some trouble "visualizing" what each represents? Look at a possible storage map (chart) of argv:



I hope the above map will be of some aid. Try and fit into the above map all of the ways of using the function argument argv. Enough of this for awhile! Let's switch topics and introduce structures.

A nice feature for a language is the ability to group variables of different types and treat them as one. This grouping of variables, called a structure in C, is called a record in other computer languages. Here is an example of the declaration of a sample C structure:

```
struct payroll
   {
   char name[30];
   int    age;
   char   sex;
   int    pay;
   };
```

The struct keyword is used to declare a structure. An optional name or tag can follow the struct keyword. In this example, I used the tag of payroll. The tag names the structure and can be used as a shorthand method for the complete structure declaration. For example, to declare two more structures of type payroll, it might be done as follows:

```
struct payroll person1,
person2;
```

The variables declared in the structure are referred to as members. Structure members or tags can have the same name as other simple variables. The C compiler can tell them apart due to the way they are used. Members of a structure are referenced as follows:

```
structure-name.member
```

The "." is called the structure operator. It connects the structure name to a member name.

Now, as promised, is a list of many of the most common errors found in a C pro-

gram. Keep these "mistakes" in mind as you code in C. Looking out for these pitfalls will help you design a more "bug free" program.

- Using = instead of == in an if statement
- Thinking arrays start at index 1 instead of 0
- Unclosed braces or brackets
- Forgetting a ;
- Using / instead of \
- "Off by one" errors in looping or array indexing
- declaring function arguments after the {
- Forgetting the precedence of operators
- Thinking C has built in string comparisons
- Using ' instead of "
- Using () instead of []
- function arguments placed in the wrong order
- Not reserving an array element for the terminal \0
- Forgetting about "side effects"

As you read some of the above most frequent C coding errors didn't you say to yourself, YES I have done that before? If you did, you are not alone! Some of these errors are caught by the C compiler, many are not. Another program for detecting possible errors in C code is called LINT. It typically better enforces the rules of C and reports more possible errors than does the C compiler.

DOS Environments

An enhancement to LDOS and LS-DOS

Code by R.N.Deglin
Text by R. Soltoff

Back in the old days of the Model I (remember them?), programmers had to develop tricks in order to pass data from one program to another. Usually, they resorted to stuffing the data into high memory thereby making use of BASIC's HIGH= parameter in order to keep BASIC from using all of memory. The problem with this approach soon became evident when other programs were written to relocate memory modules to high memory hoping to protect them from use by lowering the operating system's HIGH$ pointer. A great deal of consternation prevailed when high memory had to be used to load hard disk drivers and enhanced keyboard drivers.

To counteract some of this difficulty, some DOS vendors provided a small (and I do mean small) amount of space in lower memory for use by non-DOS programs. Unfortunately, a handful of bytes is by no means sufficient for most programs which need inter-program communications space to utilize for that purpose. Wouldn't it be great if there were a standard means of providing such a inter-program communications region? Well, there just could be!

Most low-end C programmers should be familiar with the two arguments passed to a main program: argc and argv. These arguments are the means to access data passed to the program via a command line invocation. But in the world of C, there is usually a third argument, *env*.

MS-DOS users should be somewhat comfortable with the concept of an *environment*. This is an area of memory set aside to store user-defined configuration data such as a PATH string, a PROMPT sequence, or a SHELL specification. Essentially, the environment in MS-DOS is used to store the strings of data associated with these specifications; uniform DOS service calls can then access the data so programs may know, at runtime, how the user has established his or her machine configuration. Programs may also make use of this space by adding additional environment strings dynamically. In fact, in the latest version of LB86, I make use of the environment by storing video configuration data which each LB86 module can access.

In the TRS-80 environment, one can extend this concept of environment to set aside a block of memory whose contents can then be manipulated and accessed by a uniform set of procedures. In the Model 4 environment, it has been long established that programs must honor the DOS high memory pointer; thus, reserving a portion of high memory for environment storage should be relatively safe. Even in the Model III environment, the use of LDOS has fostered an atmosphere where programs honor HIGH$.

The implementation of an LDOS and LS-DOS environment provided with the programs, routines, and functions accompanying this article were designed specifically for use with the MC C-compiler; however, a programmer wishing to use the underlying concept should be able to duplicate the environment access routines.

Not all of the modules making up the environment package will appear in this article; however, all of the modules are included on the DiskNOTES corresponding to this issue of *The MISOSYS Quarterly*.

Argument strings within an environment should take the form:

```
NAME=string
```

The string name, "NAME", should be in upper-case text by convention. The string can be anything consisting of printable ASCII characters (in the range 20H-7EH). To access the environment from a C program, it is necessary to simply supply the third argument to main. For example,

```
#include <stdio.h>
#option  ENVIRON  ON
extern char **environ;
main(argc, argv, envp)
   int argc;
   char **argv, **envp;
```

In order to accomplish this feat, it is necessary to revise the MCx/ASM file to incorporate code which creates a global environment variable, environ, and includes the necessary pointer argument on the stack frame passed to main(). The environment variable can be accessed through the global environment variable or through the environment pointer passed to main. The #option statement is a new one; it allows the inclusion of the environment pointer initialization at the option of the programmer during the compilation of the C program. Without the addition of the #option ENVIRON ON statement, no additional initialization code is included and the environment is not accessible.

Note that the envp argument is typed similarly to argv; it is a pointer to a pointer. That's because the environment is constructed of strings in a manner identical to that of the command line arguments. So if you wish, your program can access the environment string by string. However, most use of the environment is made by accessing specific strings - i.e. access by environment string name. That's where the structure of the string comes into play. So you don't have to provide your own string parsing and searching routines, C provides a few functions to access environment strings by name. These functions and their declarations are:

```
extern char *getenv(char
*varname);

extern int putenv(char *
envstring);
```

The getenv() function will return a pointer to the text string which follows the varname if that name is found in the environment; otherwise, it will return a NULL. The putenv() function returns a NULL if it successfully added the envstring to the environment and -1 if it was not successful.

The environment must first be created. This is done by a separate DOS program called INITENV/CMD. The program can create a new environment space, or it can either shrink or expand an existing environment moving any existing strings accordingly. The latter can only be done if HIGH$ can be altered; i.e. if the environment was the last module added to memory. Under MS-DOS, minimal environment space is automatically created; it is expanded by a parameter of the SHELL command. One would use INITENV as follows:

```
INITENV <size>
```

Size is the amount of memory to reserve for the environment; it must be within the range 128-1024 bytes. If the argument is omitted, a default of 256 is provided. Once the environment is created, any C program compiled with the appropriate option can access the environment strings via the getenv() and putenv() functions. These functions make use of other lower-level functions - all of which are bundled together in a library file, ENVx/REL. One of the changes introduced into MCx/ASM forces the linker to search this library if the ENVIRON option is enabled.

Finally, take a look at the sample TESTENV/CCC program which illustrates the access of the environment once it has been created by INITENV.

Environment Files

The following files associated with the MC environment support are included on Disk NOTES 7.2:

CLEARENV/ASM [CLEARNV5/CMD, CLEARNV6/CMD]

A DOS-level command to clear all environment strings.

ENV/DEF

Definitions needed to re-compile/assemble the environment library.

ENV/H

Header file to be #included with C programs which use the environment table.

ENV5/REL, ENV6/REL

DOS 5 and DOS 6 environment function libraries.

FENVTAB/ASM

An internal C function referenced in the environment functions.

GETENV/ASM

An internal C function referenced in the environment functions.

INITENV/ASM [INITENV5/CMD, INITENV6/CMD]

A DOS-level command to set initialize the environment block.

M805/H, M806/H

Standard MC include file for M80 with modifications for the environment.

MC5/ASM, MC6/ASM

Standard MC include file for MRAS with modifications for the environment.

```
/* TENVIRON/CCC */
#include <stdio.h>
#option ENVIRON   ON

extern char *getenv();
extern char **environ;
extern int putenv();

main(argc, argv, envp)
     int argc; char **argv, **envp;
{
     printf("The entry environment, environ, is:\n");
     penv(environ);

     printf("The entry environment,  envp, is:\n");
     penv(envp);

     testgetenv("AAA");
     testgetenv("BBB");
     testgetenv("CCC");

     testputenv("AAA=");
     testputenv("BBB=123");

     testgetenv("AAA");
     testgetenv("BBB");
     testgetenv("CCC");

     printf("The exit environment, environ, is:\n");
     penv(environ);
}

testgetenv(s)
     char *s;
{
     char *p;

     p = getenv(s);
     if (p)
        printf("getenv(\"%s\") gives \"%s\"\n", s, p);
     else
        printf("getenv(\"%s\") gives NULL\n", s);
}

testputenv(s)
     char *s;
{
     int i;

     i = putenv(s);
     printf("putenv(\"%s\") %s\n", s,
          i ? "failed" :"succeeded");
}

penv(evp)
     char **evp;
{
     if (! evp)
        printf"\tThere is no table\n");
     else while (*evp)
        printf("\t%s\n", *evp++);
}
```

MCMACS/ASM,MCMACS/MAC

MC macro files which define the ENVIRON option.

PRINTENV/ASM [PRINTNV5/CMD, PRINTNV6/CMD]

A DOS-level command to print the environment strings.

PUTENV/CCC

An internal C function referenced in the environment functions.

RMENV/ASM [RMENV5/CMD, RMENV6/CMD]

A DOS-level command to remove the environment block.

SETEN/CCC

An internal C function referenced in the environment functions.

SETENV/ASM [SETENV5/CMD, SETENV6/CMD]

A DOS-level command to set an environment string.

TENVIRON/CCC

The C-level program which illustrates the use of the environment.

UNSETEN/CCC

An internal C function referenced in the environment functions.

UNSETENV/ASM [UNSETNV5/CMD, UNSETNV6/CMD]

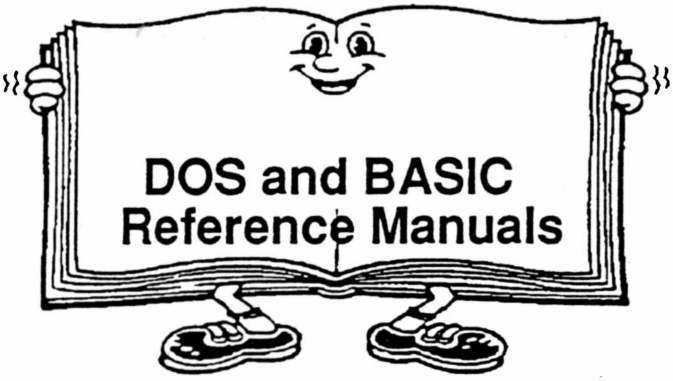
A DOS-level command to remove an environment string.

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. Obtain 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 FDUBL driver eliminates PDUBL and RDUBL and takes less memory; enhanced resident driver eliminates TWOSIDE.
- ☆ Model III version auto-detects Model 4 for installation of KI4 keyboard driver; supports CAPS, CTRL, and function keys.
- ☆ SPOOL command offers Pause, Resume, and Clear parameters. (OFF) attempts to reclaim memory used.

MISOSYS, Inc.
P. O. Box 239
Sterling, VA 20167-0239
703-450-4181

- ☆ 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 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.



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 $5.00 S&H. Since this new manual covers our compiler BASIC, you can purchase the disk version of EnhComp for $23.98.

TRS-80 Software and Hardware from MISOSYS

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. New print module. Remember those appointments.

✔ **CAL**endar 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 auto**DIALER** for Hayes modems.

✔ **CAL**Culator gives you 4-functions at your fingertips. **RPNCALC** gives 7-functions in bin, oct, dec, and hex.

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

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

PRO-WAM Application Pack

Mister ED

Mister ED is loaded with **editor** applications. All are full screen which make your editing jobs easy. Best of all, these are all **PRO-WAM** applications so they can **pop up** even when you are using other programs and applications.

✔ **DED** edits disk sectors; **FED** edits file records; and **MED** edits memory pages (even alternate banks). All use a similar display screen and strikingly similar commands to enable you to edit anything. Get comfortable with one and you will know how to use all three of these editors.

✔ **VED** lets you edit the video screen with CARD-type editing. You get cut & paste; with this, you can easily use it as the *clipboard* facility found on more expensive systems.

✔ **TED** is just like the editor you get with LS-DOS 6.3; but ours works from PRO-WAM while you are using other pro-grams! It's friendly, fast, and great for writing notes when you are right in the middle of a program you can't interrupt.

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/mod/shift, logical AND/OR/XOR/NOT, binary ops EQ/GE/GT/LE/LT/SHL/SHR; unary HIGH/LOW. Labels may be up to 15 characters long; start with A-Z, "@", or "$"; positions 2-15 may also use "?" and "_".

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.

Z80 RELocatable Assembler

MRAS/PRO-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 command 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.

SAID 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 multiple editing buffers.

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 Disassembler

DSMBLR/PRO-DUCE

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* telling 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.

CRT output is in screen-sized pages. PRINTER output is paged with column headings, page numbers and titles for print-outs that look identical to an assembler listing. Output to DISK produces a file 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 diskette 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.

TRS-80 Software and Hardware from MISOSYS

Communications Terminal

LS-Host/Term

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

ADDS25 is set up to look like a Radio Shack DT-1 emu-lating an ADDS-25 terminal. Full cursor positioning, reverse video, and blinking fields are supported.

TERM6 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 ADDS25. 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 repre-sentation, to/from INTEL Hex format and checksum files.

Full C Compiler

MC/PRO-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 Z80 relocatable macro assembler code (M80 or our MRAS). The libraries are files of relocatable object modules. **MC** is a full-featured compiler for the discriminating programmer!

MC supports command line *I/O redirection* for compiled programs, *wild-card* file specifications, parsing for UNIX "." extensions in file specifications, *overlay* support (requires MRAS), a full pre-processor, lots of options, and is designed for the programmer wishing the ultimate in C compilers. The package is supplied with the compiler, pre-processor, an optimizer, assembler macro files, C libraries, a Job Control Language file, the header files, and a 400+ page user manual. MC requires the use of either M-80 or MRAS, 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 struc-tured 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 irrelevent sections ignored when processing the 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 Model III Microsoft BASIC and supports additional commands and functions. Standard is floating point with both **single and double precision** functions; random file access ("X" mode for reclens to 32767), turtle graphics, pixel graphics, keyed 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 eliminate** contorted string packing and DATA statement high-memory module techniques for your BASIC program to access a machine code module. Z80-MODE accesses BASIC'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

RATFOR Compiler

RATFOR-M4

RATFOR reduces your programming time and effort dramatically over that required when FORTRAN is used, because RATFOR code is **fully structured**, facilitating modification and debugging, and because program flow is apparent from the overall appearance of the program; comments are sim-pler and more versatile than in FORTRAN, simplifying self-documentation. This allows changes without the subsequent debugging tolerated when modifying FORTRAN. RATFOR *compiles* source code to an *object* of FORTRAN; use your existing FORTRAN compiler to convert this to executable.

RATFOR is free-field; blanks are significant as delimiters; numerical statement labels are mostly unnecessary; all 80 columns are available for statements; provides user-defined macros; and RATFOR provides powerful loop constructs.

RATFOR is an excellent language for general purpose use, but it is vastly superior to FORTRAN when working with a large number of modules without documentation, as is necessary when producing very large programs.

Extensions supported include the "arith" macro to perform binary arithmetic operations, read and print macros for short form READ and PRINT, and support of any valid FORTRAN expression for "switch" and "case" operands.

This package includes the language translator, a *batch* file to automate compilation, a language Reference Manual, an Installation Manual, application programs in source code on disk, and our LED text editor for source code preparation.

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, "*midstring*" parameters allow sorting based on a portion or "midstring" 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 develop-ment project. DSM will create and maintain index files for you. Since the sort 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. Specify 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; General 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 sequence 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.

TRS-80 Software and Hardware from MISOSYS

Text Editor

LED

A full screen text editor for almost any type of ASCII file, 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: *overtype*, *insert*, *insert line*, and *delete*. Block mode allows the manipulation of large text areas. Search and Search/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 4/4P/4D. 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 3 in all. 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 Stacey 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 slap 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 elf 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!

TRSDOS 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. *THE SOURCE* is not only informative, but also an excellent learning tool.

These books contain the complete, **commented assembler source** code for TRSDOS 6.2, excluding hard disk support, the Microsoft BASIC and the HELP utility. Each book is softbound, 8-1/2 by 11. The complete set totals over eleven hundred pages of cleanly commented, elegant source code. Volume 1, The System, covers SYS0 to SYS5 and SYS9 to SYS13. Volume 2, The Libraries, covers all of the library commands making up SYS6, SYS7 and SYS8. 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.
- **ARCHIVEx** lets you backup some or all of the files on your hard drive to multiple floppies; 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.

DoubleDuty doubles your Model 4

DoubleDuty

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 64K 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 ScripsitPro 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

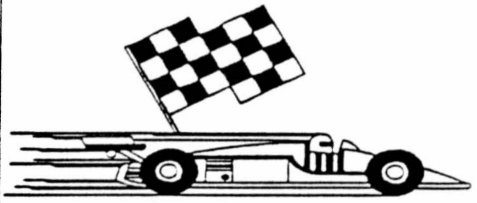
Hard Disk De-fragger

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 our 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 visual display of its de-fragging operation, which in minutes can restore a ten-megabyte directory of files to a mininum number of extents. HDPACK can even work on floppy diskettes, too.

PUTS ZIP IN YOUR DRIVE



Mod 4 features for Mod III

Hardware Interface Kit

This will allow you access to your Model 4 *hardware features* while using LDOS 5.3. Here's what you get: • **KI4** keyboard driver uses <CTRL> key, <CAPS> key, and function keys. • **SET2RAM** switches to Model III RAM mode. **@BANK** provides bank switching capability; **@EXMEM** handler allows for easy programming of memory bank I/O. • **MemDISK** provides a one or two bank RAM disk. • **BANKER** manages bank utilization. All four modules come fully documented and are ready to install into your LDOS 5.3 system using a Model 4 computer. A 128K machine is required for MemDISK/DCT and the memory management facility.

TRS-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-relocating RAM memory test.
✓ **READ40** allows access to a 40T disk in an 80T 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.

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
✓ LISTBAS print formatting for BASIC programs
✓ STRIP7 removes high-order bit off all characters
✓ STRIPCNT replaces output >127 or <32 with a # symbol
✓ MONITOR displays control chars in string form (%xx)
✓ TITLE prints a title after form feed
✓ UPPER converts lower-case character to upper case
✓ LOWER converts upper case character to lower case
✓ SLASH0 translates *zero* to *zero-backspace-slash*
✓ TRAP discards any user-defined character
✓ LINEFEED adds or removes a linefeed after return
✓ PAGEPAWS pauses after formfeed for <ENTER> key
✓ CALC performs hex/dec/bin conversion; hex add or sub
✓ REMOVE removes occurrences of a byte from a disk file
✓ COMM1 tests for modem carrier
✓ DICTATE toggles cassette on/off from the keyboard
✓ DOSPEED regulates output device speed from keyboard
✓ KSMPLUS features key re-definition *on the fly*
✓ LCOUNT adds a line number before each line of output
✓ MARGIN sends a 2-char control before margin spaces
✓ MAXLATE translates one character to a group of chars
✓ SLOSTEP for drives that require additional settling time
✓ VIDSAV saves the current video screen in high memory

Model 4 Utilities

LS-Utility Disk

Filters and *Utilities* for DOS 6.x:

✓ **PRCODES** gives control of **boldface** and underlining
✓ **TRAP** discards any user-defined character.
✓ **MAXLATE** is a translation filter system for I/O devices. Does 1:1 or :many; includes EBCDIC and DVORAK tables.
✓ **KSMPLUS** improves on the DOS; allows key re-definition *on the fly*; defines strings for the function keys.
✓ **READ40** allows access to a 40T disk in an 80T drive.
✓ **TYPEIN** combines the functions of JCL and KSM. Allows programs such as Profile 4 to be totally automated.

FORTH Compiler

HartFORTH

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

HartFORTH is designed to ***run under an operating system*** which is totally transparent to the programmer or user. The virtual Memory that it accesses for storage and retrieval purposes is a normal DOS file that is requested by the FORTH system when it is first entered. HartFORTH supports double length integers, string handling, cursor manipulation, graphics, random numbers, and floating point.

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; **IOMON** traps disk input errors; **MAPPER** checks the granulization of files; **RAMTEST** performs an exhaustive test of RAM; and **UNREMOVE** 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; **DOEDIT** provides command editing; **MEMDIR** gets a memory directory; **PaDS** 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** zaps disk or file sectors on a full-screen basis; **IFC** to interactively copy, move, rename, delete, and invoke files; **ZCAT** catalogs 6.3 disks.

Cornsoft's arcade-type games

MISOSYS has licensed the action games previously published by The Cornsoft Group: Frogger™, Scarfman, Bouncezoids, Crazy Painter, and Space Castle are exceptional action games with great video and Alpha joystick support (even the MISOSYS HD joystick). All games are for Model III/I (or 4 in III mode). All five games are included on a single disk. Requires a DOS disk.

TRS-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 includes the famous games: ANDROID NIM, BEEWARY, 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.

TRS-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 you collectors.

TRS-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 darn useful programs that you might use for real purposes. Also has some educational games for the kids..

TRS-80 Software and Hardware from MISOSYS

Mod 4P internal 1200 bps modem

TT512P Modem

MISOSYS has the **TeleTrends 300/1200 baud** Hayes compatible modem which slips right into the Model 4P's **internal** modem slot. This has a **full "AT" command set** capability. For now, you Model 4P folks can upgrade to a real Hayes-compatible modem operating at 1200 baud. It's just what you needed for your 4P.

Hard Disk Drives and Parts

MISOSYS MSCSI Hard Drives

Genuine MISOSYS MSCSI hard drive kit packages plug into Model 4/4P/4D and Model III computers. The 11" x 10" x 4.75" (LWH) beige drive case has space for a half or full height drive, 115V/230V 30 watt power supply and fan, hard disk controller (HDC), host adaptor (H/A), optional hardware real time clock (RTC), LED status lights, and 50-pin SCSI female connector. Optional Joystick port with joystick. Includes software for one DOS installed (Model III or 4), and 4-foot host cable.

20 Meg drive kit (ST225), complete
40 Meg drive kit (ST251-1), complete

Aerocomp Hard Drives

Genuine FCC-certified (EZU5PL3000) Aerocomp drives are now available from MISOSYS. These are new and complete units ready to run. The external hard drives are FCC Class B certified. They include continuous duty switching power supplies, filtered forced air ventilation, effective EMI filtration, and solid steel construction. Five front panel lights indicate power, ready, read, write, and select. Drivers available for Montezuma Micro CP/M, LDOS, or LS-DOS. Comes complete with 4-foot host cable.

Aerocomp 5 Meg drive Hard drive
Aerocomp 20 Meg drive Hard drive
Aerocomp 40 Meg drive Hard drive

Component Piece Parts:

Seagate ST225 5.25" 20M drive, 65ms
Seagate ST251-1 5.25" 40M drive, 28ms
MISOSYS SCSI H/A with software
Xebec 1421 Hard Disk Controller
Adaptec 4010 Hard Disk Controller
power Y cable
XT drive cable set (20P HDR-EDC.; 34P HDR-EDC)

Model I Double Density Controller

Double Density Controller (DDC)

80% more disk capacity is what you get when you add Aerocomp's DDC to your TRS-80 Model I. This controller has withstood the test of time. All the others are gone, yet the DDC endures. Why? Because it has proven itself as the only way to achieve reliable floppy disk operation on the Model I. Requires the Radio Shack Expansion Interface and software driver. Use our LDOS for top-notch up-to-date DOS performance.

Model III or 4 RS-232 Serial Card

RS232 Serial Card or Kit

MISOSYS acquired the remaining stock of genuine Aerocomp serial cards for your Model III or 4. Replace 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.

Model III/4 Floppy Disk Controller

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"x4.125"x2"; mounting holes are 3.75"x7.25". Direct replacement for Tandy Model 4 power suppy.

XLR8er'd Model 4 in III mode

LDOS 5.3 XLR8er Interface Kit

This is based on our Hardware Interface Kit but is for the XLR8er board operating under LDOS 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 from 2 to 27 banks; however, only 10n are supported through XLR2RAM. The VIDX expands scroll protect from 0 to 15 lines rather than just 3; supports reverse video, and supports the @DSP of character codes from 0-31 and 192-255.

Model I/III Data Base Manager

Auto 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.

AFM 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 AFM database file. The size of an AFM database file can expand to accommodate a full fifteen megabytes.

AFM allows you to set printer parameters through a FORMS command very similar to the FORMS 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 the major key fields through the use of powerful relational operators including AND, 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 do your data base. AFM 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 screens are 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 field your data each with different field lengths.

• FORM-LETTER output of AFM 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>ext option which allows selected field output without fielding 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, 4!

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 Model 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. SU+ does not support Newdos/80 double-sided disks.

Super Utility Plus 4/4P/4D

The Model 4 version of Super Utility has all the features of the Model I/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 builtin 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 to have ready. 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 6 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 EPSON or compatible printers to be fully utilized with SuperScripsit and SCRIPSIT PRO.

This is a replacement driver for the ones you got with SuperScripsit. 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.

Mailing List/Fixed Database

PowerMail Plus

This program was because all the other mailing list/data base 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 floppies or hard disk under all popular TRS-80 operating systems. Allows importing of data from several other once popular mailing systems to avoid re-typing. Written in machine language by the author of Super Utility, 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.

Form Letter Module

Text-Merge

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.

A Major Enhancement for SCRIPSIT 4, III and I

PowerSCRIPT

This modification for Radio Shack's SCRIPSIT program turns it into a POWERHOUSE! Our 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 configureable to just 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). Lastly, PowerScript removes the limited copy "feature" of SCRIPSIT so that you may make as many copies as you need or copy it to your hard disk without hassle.

animated TRS-80 screen graphics!

PowerDraw

PowerDRAW allows you to create graphics (mixed with text if desired) and save them to disk. It allows you to create up to 33 "frames" of animation and "play" them like a movie. It also allows you to save the graphics in several modes, including BASIC listings, CMD file format, etc. These can then be merged into your own 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 type of printers. Lastly, PowerDraw has the ability to load in many types of TRS-80 graphic's and convert them to BASIC listings like a BASIC program generator!

Block Graphics Drawing

PowerDOT 2.0

This program is similar to PowerDraw, but quite different. It allows you to create "hi-res" type screen graphics combined with text, and allows you to create drawings much larger than your screen. The screen is a "window" to a much larger drawing area and you use the arrow keys to move about the drawing. In a way, it is similar to Macpaint for the Macintosh computer. It also allows you to create custom fonts for ads, etc. Many of our early ads were created with PowerDot. It creates the hi-res effect due to each TRS-80 block pixel being printed as a single dot. Please specify if EPSON, Okidata, Prowriter, or Radio Shack printer.

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 data base 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* {±ddddddd.dd}, *floating point* {±ddddddd.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, LBMANAGE, 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 word processor. 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. A 200-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 straight forward, 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 couldn'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 SuperSCRIPSIT 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 SuperSCRIPSIT 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 SUW4/4P™ 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

MISOSYS, Inc.

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.

Prices currently in effect:

Complete MISOSYS Hard Drive:	
20 Megabyte kit:	$395
40 Megabyte kit:	$495
Joystick option	$20
Hardware Clock Option	$20
LDOS software interface	$30
SCSI software interface	$25
Aerocomp Hard Drives:	
20 Meg unit	$350
40 Meg drive	$450
H/A with MFM software	$75
Separate Hard Disk Controllers	
Xebec 1421 HDC	$45
Adaptec 4010 HDC	$45
WD1002S-SHD	$45
Drive power Y cable	$5
XT drive cable set	$5

Note: freight charges are additional.
Prices subject to change without notice.

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.

MISOSYS, Inc.
PO Box 239
Sterling, VA 20167-0239
U.S.A.

Contents: Printed Matter

BULK RATE
U. S. POSTAGE
PAID
Sterling, VA
PERMIT NO. 74

Attention Postmaster: Address Correction Requested
Forwarding and return postage guaranteed