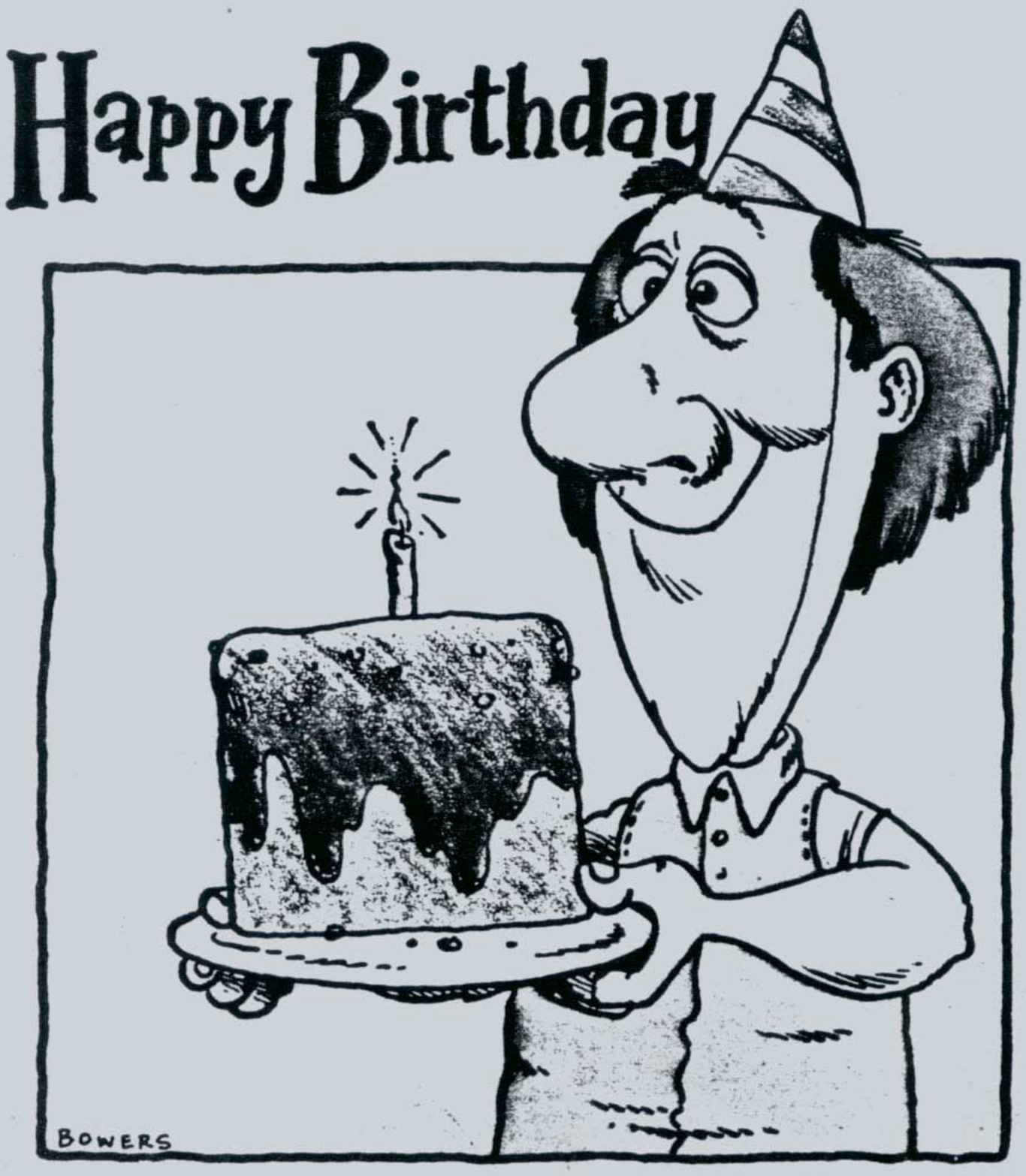
TRS TIMES

covering Model I, III & 4

Volume 2. No. 1. - Jan/Feb 1989 - $4.00



Expanded Birthday Issue

36 pages of pure TRS-80

LITTLE ORPHAN EIGHTY

Well, the clock struck midnight and here we are in 1989. Happy New Year everybody. Much happiness and prosperity to all of you in the year ahead.

To the people who were with us last year, "Thank you for staying with us." To our new subscribers we say: "Welcome aboard."

The start of a new year might be a good time to clarify some concerns many new subscribers have voiced. They ask why TRSTimes only accepts 'calendar year' subscriptions rather than 'running' subscriptions; also, many have said that publishing 'one year at a time' gives the impression of instability.

To answer, let me first say that one of my many philosophies about life in general is that NOTHING IS FOREVER. Then, let me take you back a year or so in history and point out that this philosophy was generously proven by 80 Micro when they cut off support for the TRS-80 at the end of 1987, and again when they, a few months later, ceased publication altogether.

A few years earlier it was COMPUTER USER, BASIC COMPUTING and SOFTSIDE, all very fine sources of TRS-80 information, who just could not manage survival.

OK, so they quit publishing their individual magazines - I have no problem with that. What I do have a beef with, however, is that all four magazines were giving the impression of longevity by asking you to subscribe for 1 to 3 years in their LAST ISSUE. You just know that some poor soul forked over some hard-earned money for a subscription - and never received an issue. I find that distasteful.

Also, in the case of SOFTSIDE, their last 8-10 issues contained nothing but TRS-80 programs they had published in earlier issues - heavy indication of a severe case of BURNOUT.

I REFUSE to involve TRSTimes in these practices. If I find myself burning out, no longer having fun, the year will be completed with material written BEFORE the burnout, and publication will then cease. Now, since we only accept 'calendar year' subscriptions, all subscribers will get each and every issue coming to them. It is the fairest (and the only) way to protect subscribers, as well as myself, from the above-mentioned perils of computer magazine publication.

So, you see, it is not instability. Au contraire, it is STABILITY. TRSTimes will NOT take your money for issues going years into the future that might never be produced; instead, we only promise that TRSTimes will produce six timely issues in 1989 - and that we WILL.

TRSTimes in 1990 - will we continue?

If WE still have fun and YOU are still interested, we may just do it again in 1990. The decision will be announced in the September issue (2.5).

A TRSMAS STORY

Santa Claus, a devoted TRS-80 enthusiast, called about a week before Christmas to tell me that he would be dropping off a present on his way back to the North Pole from Palm Springs.

Sure enough, on December 15th he climbed down the chimney and left a large box under the TRSmas tree. Now, being a night person, I was still awake, so in no time at all I was busily tearing off the pretty paper.

Soon the wrapping was demolished and I opened the big carboard box. Lo and behold, inside was a brand-new printer.

No, not just ANY printer, but the answer to my many prayers, a dream come true, the sure cure to the 'dot-matrix blues', a LASER PRINTER, an HP LaserJet series II. WOW!!

This little goodie had been on my list since TRSTimes began - and now good ol' St. Nick had come through - what a guy!!

I carefully removed the HP from the box and immediately felt intimidated. This thing came with 4 manuals, and it didn't look like any printer I'd ever owned before. It looked more like a medium sized copying machine. I had this nagging feeling that, heaven forbid, I would have to read the manuals.

Luckily, I found the one that said "Getting started with the LaserJet series II Printer" and I followed the directions step by step. Somewhere about 4 A.M. I had it set up and plugged in, ready for action.

To make a long story (and night) short, let me just say that, the minute I saw the first printout, I was sold. This was the way I wanted TRSTimes to look.

At this point the January issue was finished, except for three pages (this page, the Index, and the back cover), ready to go to the printer for mass copying. BUT the masters were done on a dot-matrix printer - and I had just seen a glimpse of heaven!!

Needless to say, I spent the remaining night hours reading the manuals, slept for a while, then got back up and started redoing the masters on the LaserJet, learning as I went along.

Obviously, I have much more to learn. Not only are there many new things to try, but somewhere, somehow, I lost the page numbers. I have no idea where they went; they just left! Yes, this issue does have page numbers, but they were pasted up with scotch tape; so if they are a little crooked, or maybe even completely out of sequence, BLAME MY WIFE - SHE DID IT!!

And now...

Welome to the 2nd year of TRSTimes

TRSTimes - Volume 2. No. 1. - Jan/Feb 1989

CONTENTS:

TRSTimes is published bi-monthly by TRSTimes publications.
20311 Sherman Way #221. Canoga Park, CA. 91306. U.S.A.

1989 subscription rates (6 issues):
United States and Canada: $18.00 (U.S.)
All other countries: $23.00 (U.S.)

• Model I, III & 4

Run IBM software on the TRS-80

HACKER'S HEAVEN

CONVERTING INFOCOM ADVENTURES

By John Fowler

Infocom announced in 1986 that it would no longer support the TRS-80 Models I, III and 4. While this came as somewhat less than a shock (I guess TRS-80 users must be getting used to it), it was a major inconvenience to Infocom players. Now, to play Infocom adventures, one would either have to visit a friend or buy a computer that is supported.

I don't accept either of these solutions, so after some research and a couple of lucky coincidences, I found a way to transfer 'Leather Goddesses of Phobos' from an IBM PCjr to my TRS-80 Model III. The technique I used should work on all normal Infocom adventures (but are there REALLY any NORMAL Infocom adventures?) at least until Infocom decides to change its data file format, which hopefully won't be for a while. While this file is written to explain how to transfer to a TRS-80, it may also work with other orphaned computers for which Infocom had previously manufactured. It all depends on whether you have a visible driver program. At any rate, read on, and if you wish to modify this file for your own computer, feel free to do so.

STRUCTURE

Before I get on to the method of transfer, let me talk a little about the structure of Infocom adventures. In other words, how does Infocom make its games so quickly for so many different computers at once?

Infocom authors always begin by writing their adventures for a "virtual machine," that is, a machine that does not exist in reality. I will refer to the code generated for the virtual machine as the data file. This data file is then transfered to the disk formats of the computers being supported.

Of course, the computers can't understand a word of the data file in itself. So the authors make for each machine a "driver program," designed to interpret the data file and translate it to the machine on which the adventure is to be used. The authors only have to make this driver file once, as long as they don't want to change virtual machines. So, to make a new adventure, all they have to do is transfer the data file to a disk format and the driver program to the same disk.

The method presented here is based on the fact that the data file, with a few exceptions, is always the same file, no matter which computer format it is on. Only the driver programs vary from computer to computer. So, if one has the driver program for a now-abandoned computer, and one finds a way to copy the data file over, one then has a version for his or her computer!

TRANSFERING TO MODEL III/4

Model I users please be patient. I'll get to you soon enough. Read this section, however, because some of it applies to you.

The first thing one needs to run an adventure on the Model III or 4 (III mode only) is the driver program. Because the driver for these computers is the same on all TRS-80 Infocom adventures, any of them will do. The driver program is recognized by a /CMD extension on the name. For example, the driver for Hitchhiker's Guide to the Galaxy is HITCHHIK/CMD (and the data file is HITCHHIK/DAT, but we'll worry about that later). Transfer this driver to a blank, newly-formatted diskette, which may have a bare DOS on it as well.

This driver program knows which data file it is looking for by checking its own name, then sticking /DAT on it. So you will need to rename the driver to whichever game you are planning to transfer. For example, on Leather Goddesses of Phobos, you should rename your /CMD file to LEATHER/CMD.

Now comes the hard part. You need to find a disk format where the data file is easy to get at and transfer. Fortunately, Infocom is starting to take the copy protection off its games, so that the data files are visible from the directory and accessible by normal methods. The IBM version of LGOP is an example of a non-copy-protected disk with a visible, accessible data file (named LEATHER.DAT).

After you buy (yes: **BUY**; see A NOTE ON PIRATING) this version, you will need to transfer it from the foreign disk format to your own. Hypercross or SuperCross/XT may do the trick, or even TRSCROSS if you happen to have an IBM handy. However, if you don't like buying software-translation programs and don't know how to do it yourself, you can use an RS-232 link between a friend's computer and your TRS-80. Name the file the same as your driver program except a /DAT extension. At 1200 baud, it usually takes 20-30 minutes to transfer the data file, meaning 80-120 minutes at 300 baud or 10-15 minutes at 2400. If you have a null modem and the computers right next to each other, you should use 9600, which means the file will be transfered in less than five minutes. 19,200 baud rarely works out.

Once you have the /CMD and /DAT files, you're almost set. But wait, there's one more thing. As a form of primitive copy protection, the /DAT file must have a password on it or the program will bomb out. This password is "SMC." An example command under TRSDOS 1.3 to assign this password to LGOP would be:

ATTRIB LEATHER/DAT (ACC=SMC,UPD=SMC,
PROT=EXEC)

Do NOT put the password on the driver program.

Now type the filename of the driver program to execute. Tah dah! A program in TRS-80 Model III/4 format!

One note: to the best of my knowledge this will not work on Infocom's "Interactive Fiction Plus" stories, even if you have 128K! Infocom's decision to drop the TRS-80 series came before A Mind Forever Voyaging, so a TRS-80 Plus driver was never made. You can write a nasty letter to Infocom if you like, but I doubt it will do any good. Or, if you want a challenge (and I am talking CHALLENGE), you can try to make a Plus driver yourself. Don't get your hopes up on getting any help from Infocom, though.

TRANSFERING TO MODEL I

To play the Infocom games not specifically for the Model I, you MUST have either a double-density modification or a hard drive. The reason for this is that Infocom data files are greater in length than 125K, while SSSD 35-track disk drives hold less than 90K (and no, you can't split the data file into two portions and stick them on different drives, unless you also want a CHALLENGE).

Because Model I Infocom data files are different from other kinds of data files, you will need a new driver. You should copy one from a Model III version, and then apply a few patches. Not having a Model I, I cannot test these patches myself. They are only certified to work under NEWDOS/80 version 2.0, however, and I make no guarantees or offers of further support.

First, get the driver program from a Model III version. If you don't have a double-density adapter, you will need access to a Model III or 4. Again, you can either use an RS-232 link or Super Utility Plus to get the file to one of your own floppies (this one is less than 90K, so it's okay). Now apply the following patches, taken by combining Northern Bytes Volume 6 Number 8 and Volume 7 Number 3. Using a zap program (like Super Utility Plus or SUPERZAP), change all occurences of

(hex) '**21 25 42**' to '**21 18 43**'.

Also, in the first sector,
find '**11 44**' and change to '**49 40**'.

Now you need to get the data file and put the SMC password on it as described before. If you don't have double-density or 80-track drives, you'll need to put this data file on a hard disk, which I know has been done, but I can't offer any specifics. If you can, make the hard drive the master drive and transfer the driver program there as well. I'm afraid that without any experience on Model I's or hard drives myself, I cannot offer any more help than this.

A NOTE ON PIRATING

Infocom has a copyright on its data file as well as its driver program. This means that it is illegal to copy it even to another disk format on another computer, unless you own the supported copy to begin with. You might think Infocom deserves to have its copies pirated for dropping the TRS-80 in the first place, but before you pirate, remember this: Infocom has the right to change its data file format at any time. If you buy the supported copy, that makes Infocom happy because they make money without the expenses of supporting your computer system, so they are likely to keep the data file format around. However, if they suspect that anybody is taking advantage of their data file format by simply grabbing a friend's data file, they could easily change the format permanently, meaning that the TRS-80 would be almost hopelessly abandoned once again. If you like Infocom games, then Infocom should deserve your financial support. You might even want to write a thank-you note on your registration card for keeping the standard data file format.

Go ahead, buy the thing!

• **Model I, III & 4**
• **Basic**

LISTER

by Lance Wolstrup

Every so often I run across a public domain program that just, somehow, captures my fancy. LISTER/BAS is one such program. It was sent to me by a reader who needed help in correcting some minor, but very irritating errors. Since LISTER/BAS is short and uncomplicated, fixing the code took no time at all. In the process I fell in love with it.

LISTER/BAS is a simple 'pretty printer' program that will print out a Basic program saved in ASCII format. It features a 'header' which gives the name of the program being printed, as well as the date of the print-out (the date is shown on page 1 only). It also has a 'footer' which prints the page number. Both tractor-feed and single-sheet feed are supported. Another nice feature is that you can control the indentation of the left margin, thus assuring a nice print-out that can be hole-punched without losing portions of the code.

This program should should serve as an example to ALL programmers, including myself. Code does NOT have to be long, intricate or fancy. Let's face it, the bottom line of all programs is: DOES IT WORK - does it do the job it was intended to do? If the answer is YES, then the program is GOOD.

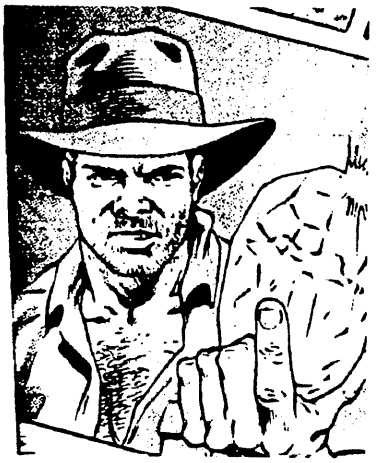
The code to LISTER/BAS is not fancy; it is not intricate and it is certainly not long. It does, however, do the job. So, to the unknown author, out there somewhere, here's a tip of the hat from TRSTimes: "Well done".

```
10 'LISTER/BAS

20 'PUBLIC DOMAIN PROGRAM

30 'FOR MODEL I, III & 4

40 'ORIGINAL AUTHOR UNKNOWN

110 CLEAR 3000:CLS:O=1:DEFINT I-L

120 PRINT"BASIC PROGRAM LISTER WITH
HEADER AND FOOTER PAGE NUMBERS":
PRINT

130 PRINT"ENTER LEFT MARGIN INDENT
(ENTER=0)":PRINT:INPUT ID:F=80:PRINT

140 PRINT"TRACTOR FEED PAGES, OR SINGLE
SHEET FEED (T/S) (ENTER=T)":INPUT H$:PRINT

150 F=F-ID:P=79

160 PRINT"Enter Program Filename (ASCII
only)": PRINT:INPUT FI$

170 OPEN"I",1,FI$:LPRINT:LPRINT

180 IF O)1 THEN LPRINT TAB(ID)"Program:
"FI$ ELSE LPRINT TAB(ID)"Program: "FI$;
TAB(P-LEN(FI$));LEFT$(TIME$,8)

190 LPRINT TAB(ID) STRING$(F,"="):LPRINT

210 IF I$="" THEN LINE INPUT#1,I$

220 IF LEN(I$)<=F THEN J=LEN(I$) ELSE J=F

230 K=0:O=2

240 FOR I=1 TO J

250 IF ASC(MID$(I$,I,1))=10 THEN J=I:K=1:
GOTO 270 ELSE NEXT

270 J$=LEFT$(I$,J-K):I$=RIGHT$(I$,LEN(I$)-J):
L=L+1

300 LPRINT TAB(ID)J$

320 IF EOF(1) THEN FOR I=L+1 TO 55:
LPRINT:NEXT:GOTO 340

330 IF L<55 THEN 210

340 FOR X=1 TO 4:LPRINT:NEXT

350 LPRINT TAB(ID)STRING$(F,"="):PG=PG+1

370 LPRINT TAB(ID) TAB(36)"PAGE- "USING"
##"; PG

390 IF H$="" OR H$="T" THEN 410 ELSE
PRINT"Press ENTER when ready ";

400 I$=INKEY$:IF I$="" THEN 400

410 FOR X=0 TO 1:LPRINT:NEXT

420 L=0:IF NOT EOF(1) THEN 180

440 PRINT"Press <ENTER> to run another
listing, <BREAK> to quit."

450 INPUT Q$:IF Q$="" THEN RUN
```

HUNTING FOR BURIED TREASURE

Peeking & Poking Model 4

by Lance Wolstrup



By now it must be apparent that my favorite DOS is TRSDOS 6.2. Unfortunately there are two versions: the original 6.2.0. and the upgrade 6.2.1. I never took the time to transfer my 6.2.0. files over to the upgrade, so I am often working with the old version, which has a problem with the FORMAT utility. FORMAT/CMD in 6.2.0 does not prompt for the number of sides to format. It simply performs only single sided formatting. Not until 6.2.1. was the user given the choice of the number of sides to format. Very inconvenient, if you use the earlier version of the DOS and have upgraded to double sided drives.

Of course, the best solution is to change to TRSDOS 6.2.1., but if that is not possible (or in my case, too much trouble), let's modiify the FORMAT utility to do what we want.

First, understand that FORMAT/CMD in TRSDOS 6.2.0. was **always** capable of formatting double sided diskettes. The all-wise author certainly included the routine to handle double sided formatting, but excluded the '# of sides' prompt and, as we all know, since the default is 1, the result is a single sided format.

This exclusion takes place in the FLAG$ table, more specifically in LFLAG$ at memory location &H75 (117 decimal).

LFLAG$ holds the following pieces of information:

Bit 0:	1 = inhibit step rate question in FORMAT 0 = display prompt
Bit 1:	Not known at this time
Bit 2:	Not known at this time
Bit 3:	Not known at this time
Bit 4:	1 = inhibit 8" query in FLOPPY/DCT 0 = display prompt

Bit 5:	1 = inhibit # sides question in FORMAT
	0 = display prompt

Bit 6:	Reserved for IM 2 hardware
Bit 7:	Reserved for IM 2 hardware

The initial setting of the bits is: 00110001. From this we can see that bits 0,4 and 5 are 'on' so as to inhibit the step rate question in FORMAT, inhibit the 8" query in FLOPPY/DCT, and inhibit the # sides question in FORMAT.

Logic tells us that if we turn off bit 5, FORMAT/CMD should then ask us how many sides we want to format. There are many ways we can do this, but since this is a column dealing with PEEKing and POKEing, let's fire up BASIC and use it as the vehicle to modify DOS.

From BASIC type: **PRINT PEEK(&H75)** (ENTER)

Unless you have some IM 2 hardware (whatever that is) attached to your Model 4, BASIC will respond by displaying the value **49** on the screen. In other words, memory location &H75 holds 49. This value corresponds to bits 0,4 and 5 'on', while bits 1,2,3,6 and 7 are 'off'. What we need to do now is turn off bit 5.

If you remember the mini-tutorial on AND, OR and XOR from issue #1.1., you know that we can turn off bits by ANDing the PEEK value with a number where ALL the bits are on EXCEPT the bit (or bits) we want to turn off, and then POKEing this new number back into the original memory location.

Type this:
POKE &H75,PEEK(&H75) AND 223 (ENTER)

Now type: **SYSTEM** (ENTER) to get back to DOS so we can try out the double sided formatting capability.

Assuming you have a double sided drive :1, insert a blank diskette there and type:

FORMAT :1 (ENTER)

After you have answered the prompts for 'Diskette name', 'Mater password' and 'Single or Double density (S,D)', you will then be greeted with: 'Enter number of sides (1,2)'.

Type: **2** (ENTER)

Answer the 'Number of cylinders' prompt and you are then formatting a double sided diskette from TRSDOS 6.2.0.

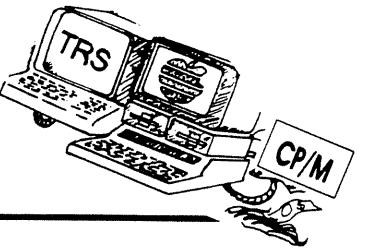
Should you wish to also be prompted for the Boot strap stepping rate, the AND 223 from above should be changed to **AND 222**.

Be sure to **SYSGEN** your new FORMAT configuration before you turn the computer off.

See you in March.....

Disk Duplication & Conversion

(TRS-80, IBM, CP/M)



by Fred Blechman

I've been spoiled! After working for four years with my TRS-80 Model III and three years with my Sanyo MBC 555, my recently purchased IBM PC/XT Clone has been a disappointment in several ways - especially in the area of disk duplication. S-L-O-W! Gads, with all that memory (640K RAM), making disk copies should be speedy - but not with MS-DOS 3.2. The DISKCOPY A: B: command takes almost two minutes to copy a double-sided disk, and that's after loading in the DISKCOPY file to start with.

Also, converting data and text files from CP/M or TRS-80 disks to the IBM takes some special doing. This article, without getting overly technical, will discuss programs and techniques to speed up disk duplication. I'll also cover disk conversion - making disks that can be read by different operating systems.

Disk Duplication

Disk duplication involves making an exact copy of a disk. This can be to make a working copy of a purchased or "master" disk, or to make an archival copy to store, or to make a copy for sale or swapping. Let me make it perfectly clear that I'm not talking about pirating programs. The disk duplication programs I'll discuss will NOT copy protected disks! That's a whole other subject - and as a software originator myself, I'm not about to promote piracy.

Duplication in large numbers is required by software manufacturers, public-domain sources, and computer club librarians. Here the difference in duplication times is multiplied by the number of copies, and can translate to a very significant time saving. For example, a program to be described later for the TRS-80 saves almost three minutes on a copy - or almost an hour saved making only 20 copies!

Duplication is usually provided by the Disk Operating System (DOS). In the case of the TRS-80 Models I/III/4, the command BACKUP is used by most of the available DOS'es. For MS-DOS, the command is DISKCOPY. The awkward combination of FORMAT, SYSGEN and PIP used together provide the normal means for making duplicate disks in the CP/M operating system. I've forgotten the command used by the Apple II +/e/c; my brief experience with an Apple IIc was so frustrating (18 disk swaps to make a single copy !) I've tried to put it out of my mind. Special duplication programs, discussed later in this article, are available for most micros to speed the process.

Disk Conversion

Disk conversion is another matter entirely. Suppose, for example, you have a bunch of text and/ or data files you've accumulated using a TRS-80 Model III, and now you have an IBM PC compatible. Rather than retyping everything, you COULD use two modems, two telephones, two communication programs (and probably two computer operators) to transfer the programs via the telephone lines. Or, you COULD run a cable from the TRS-80 to the IBM PC if they are reasonably close to each other, and with an appropriate terminal program at each end perform the transfer. Both these methods are clumsy, time-consuming and error-prone compared to disk conversion.

Various techniques, programs and equipment exist for making a new disk from your original disk. The new disk is directly readable on the target computer. You just stick it in the disk drive and operate the computer as if that new disk were made on the target computer in the first place.

The basic problem is that different operating systems, and frequently different manufacturers within the same operating system, use disks with different sizes and formats. Most use 5-1/4" double-sided double-density formats these days, but older machines used single-sided or 8" disks, and some newer ones use 3-1/2" diskettes. The "format" defines the number of tracks, sectors and bytes-per-sector, plus various other parameters. In the CP/M world alone there are dozens of different, incompatible formats! ··

Fortunately, the IBM PC 40-track, 9-sector-per-track, 2-sectors-per-cluster, 512-bytes-per-sector, double-sided format of 360K (actually 368,640) bytes per disk has become a de facto standard. But even this is now being challenged by 720K formats used by the IBM PC AT and most 3-1/2" diskettes. So, disk duplication is copying a disk to be used on the same (or 100% disk-compatible) computer, and disk conversion is copying the INFORMATION from a disk to another disk in another disk format.

TRS-80 Model III and RAPIDOS

My first experience with disks was with my TRS-80 Model III. Compared to cassettes, the disks were so fast and convenient I was delighted - until I used the BACKUP command. Even with two disk drives (so no disk swapping was required), and even though the disks were only single-sided (about 184K), it took 3-1/2 minutes to make a diskcopy of a reasonably-full disk. The total time depended on how many tracks were used on the source disk, and whether the target disk was blank or pre-formatted.

This was very slow. I frequently had to make dozens of copies at a time of my AMBIZ-PAK of computer programs for Amway Product Distributors, and updates when Amway changed product prices. The time consumed making BACKUPs of my master disks could have been much better used.

Because Amway's product prices change frequently, and are used within my programs, I could not use a duplication service, since that typically involved a cost for a master disk and a minimum of 50 copies. More on duplication services later.

Then I discovered Rapidynamic Software's "RAPIDOS" for the TRS-80 Model I/III/4. (I don't know if Rapidynamic is still offering this program, but it should still be available from Recreational Mathemagical Software, 129 Carol Drive, Clarks Summit, PA 18411 for $30. Specify TRS-80 Model I, III or 4.)

RAPIDOS is actually a mini-operating system for the TRS-80, with FORMAT, BACKUP, DIRectory, KILL, LOAD and other commands. A special DUPlication section features a simple menu that allows you to FORMAT, BACKUP (onto an already-formatted disk), FORMAT and BACKUP, check display speed, or change disk parameters (number of tracks, diskette name, password or date). With RAPIDOS you can make a BACKUP to a pre-formatted disk in only 26 seconds!

I tested RAPIDOS by making a duplicate of a disk with 30 of its 40 sectors used. The BACKUP command in TRSDOS 1.3 formatted and copied the files on the disk in 3 minutes and 28 seconds, or 208 seconds. Using RAPIDOS, the same job, including formatting, was done in 38 seconds. That's a saving of 170 seconds (almost three minutes) per copy! WOW!

Another really useful feature of RAPIDOS is the "Display Speed" command. It runs the selected drive and displays its rotational speed (around 300 RPM) continually until you hit the BREAK key (which returns to the menu.) This allows you to adjust drive speed with a small screwdriver to get it right-on - and that's especially important with RAPIDOS, since it will not function properly with a drive running at over 302 RPM.

I no longer dread the time to make copies of my AMBIZ-PAK programs and updates, thanks to RAPIDOS.

TRS-80 Model III and Supercross

When I first started writing with my computer, most magazine editors required only printed text copy, which I provided with my dot-matrix printer.

No problem. My favorite word processor, to this day - even after exposure to many other available word processing programs - is ZORLOF.

Things changed. As more editors got computers and word processors, they began requesting the text on disk as well as hard-copy. In the early days, most were using WordStar on Kaypros or CP/M machines, so they asked for the text on a Kaypro disk (which most CP/M machines can be configured to read.) Hmmmm. Problem. I didn't use WordStar, and I didn't have a Kaypro or other CP/M machine. I needed disk conversion.

About this time PowerSoft began selling SuperCross/XT, a disk conversion program for TRS-80, MS-DOS, and CP/M, for use on a TRS-80 Model I/III/4. It was a blessing.

Using SuperCross I would make a Kaypro formatted disk on the upper drive of my TRS-80 Model III, and then transfer text files saved in ASCII by ZORLOF. A real breeze! I never had an editor call with a problem reading the disk on their CP/M machine.

These days most editors want text files on an IBM PC compatible disk. I still use SuperCross regularly to supply editors with an ASCII file on an IBM PC-compatible disk - made on my TRS-80 Model III. I can't even estimate how much this simple-to-use disk conversion program has saved me in time and effort. I don't even want to think of the alternative of sending the text long-distance by telephone and modem.....

I've also had the occasion, using SuperCross, to create a Kaypro disk, with ASCII versions of my TRS-80 AMBIZ-PAK BASIC programs. This allowed an MBASIC programmer to convert my programs to CP/M for use on various machines.

IBM PC Clone and TRSCROSS

Somewhat like SuperCross, this PowerSoft program provides another way of "marrying" an IBM PC to a TRS-80 Model I (double-density), III or 4. In this case, the IBM is the "native" machine, and the TRS-80 disk is the "alien". The purpose is to allow you to read and write TRS-80 disks, so that you can use TRS-80 data, text and BASIC programs on an IBM, or transport data, text or BASIC programs from an IBM to a TRS-80. (BASIC programs need additional translation, but at least you can get the program between machines easily.)

TRSCROSS does not make CP/M disks, but SuperCross does. Therefore, if you have both an IBM using TRSCROSS and a TRS-80 Model III or 4 using SuperCross, you can manage to make just about any format!

Duplication Services

As special equipment has become available to make multiple high-speed disk copies, several companies have been formed specifically to provide mass disk duplication. The equipment is too expensive for the casual user. These disk duplication services can make copies of disks for IBM PC, CP/M, TRS-80, Apple and probably any other micro you'll run across. Some also provide conversion services.

However, the you must need 50 or more copies of a disk to make the cost reasonable. And all the copies will be exactly the same, so you must be sure your programs or data are exactly the way you want them.

Twice I've had the occasion to use duplication services, and I was entirely satisfied with the service both times. However, the first time I had a master and fifty copies made of my Programs/ Utilities Disk - and a user discovered an error I had made in one of the 56 programs. I had to personally change one line of one program on every one of those disks!

Prices change as competition, equipment and blank disk prices change, so I can't give you prices, but here are the two firms I had make duplicates:

(1) ALF Products, Inc., 1315F Nelson St., Denver, CO 80215. Phone: 303-234-0871. Small one-time mastering fee. Copy protection available.

(2) Cycle Software Copying, P.O. Box 444005, Eden Prairie, MN 55344. Phone: 1-800-642-7633. No mastering fee.

Conversion Services

Although SuperCross and TRSCROSS have the capability to make various disk conversions, you may not have the equipment or patience to go through this sometimes-confusing process. Several firms around the country specialize in making disks from one format to another, including differences in disk sizes.

Word processor conversions retain most formatting codes. Data base conversions and extractions are also available. Dealers use these services for customers upgrading from an older system, dedicated word processor or minicomputer. Software publishers don't have to stock every imaginable format, but can order "custom" disks for those customers with less popular micros.

Users can salvage programs from their older computers by converting the format used by their new one. Disk conversion can provide continued life for your favorite programs as you upgrade your equipment.

Other Sources

Now that you are more aware of disk duplication and disk conversion, you'll notice advertisements in the computer magazines (usually in the classified sections, or small display ads) offering these services. You can also check in the Yellow Pages of your local phone book under "Computers - Software & Services."

You don't have to be an expert to make duplicate copies or simple conversions. You do need to have the right software to make the job easy. However, if you need to do either of these functions in any quantity, or for odd-ball machines, go to someone in the business.

TRSDOS 1.3 CORNER

COPY THOSE TRSDOS 6, LDOS & DOSPLUS FILES BACK TO TRSDOS 1.3. WITH

CPY/CMD

by Gary E. Campbell

Requirements: Model 3, 4, 4p, 4d

TRSDOS 1.3/1.4/1.5 - EDTASM

"CPY/CMD" is a TRSDOS 1.3 copy aid program that, without leaving the native 1.3 mode, can copy LDOS, DOSPLUS and TRSDOS 6 files over to a 1.3 formatted diskette. If you are lucky enough to have double sided drives, even those double volume files can be copied too! Yes, double sided disk I/O using TRSDOS 1.3!

"CPY/CMD" also has the ability to list other DOSes files to the video, printer, or both, in numerous manners.

"CPY/CMD" supports 6 different parameters. The first 3 (a-c) are very similar to the TRSDOS 1.3 library command "COPY".

a) CPY SOURCE/EXT:D TO DESTIN/EXT.pass:D
b) CPY SOURCE/EXT:D DESTIN/EXT.pass:D
c) CPY SOURCE/EXT:D :D
d) CPY :D
e) CPY SOURCE/EXT:D (V,P,A,H,S)
f) CPY

The source file must be a TRSDOS 6.XX, LDOS or DOSPLUS disk.

The destin file drive number (:d) must be a TRSDOS 1.3 disk.

Both the source and destination drive numbers are required.

The destination password is optional.

All source files, even protected system files, may be copied.

Examples 'a' and 'b' differ only by the optional 'TO' parameter. Note the source and destination filenames may differ.

Example 'c' will give the destination file the same name. Example 'd' displays all active LDOS, TRSDOS 6 or DOSPLUS filenames on the source diskette.

Example 'e' lists the source file, to the P=Printer, V=video, or P,V=both. The listing defaults to the video.

The A option converts the byte to a 7 bit ascii byte. It should be used when output is sent to the printer, so your printer does not receive bytes larger than 7Fh.

The H option converts the byte to 16 bit hex ascii notation. If the S or A options are not used, 8 bit bytes will be sent to the output device. This may be useful if you are echoing the output to your RS232. The S option slows the output. The ',' is optional. Note that when using example 'e', the @ key pauses, and the Break aborts output.

Example 'f' is a help function that displays all parameters supported.

Because of the length of this program, I must limit my comments about the program itself. This program does not work on NEWDOS, due to its non-conforming directory. CPY/CMD may work on other DOSes, and as no writing occurs on the source disk, it is safe to try it on other DOSes. Testing has been limited to single and double sided files on LDOS, TRSDOS 6.XX and DOSPLUS disks.

The first part of the program displays all source disk files, The main processing area, labeled "COPY" gives you an idea of what steps were required to copy a program.

First, the source and destination filenames, along with their respective drive numbers, are checked for syntax, and stored in the programs buffers for future reference.

Now, the program searches for the source directory (GETDIR). It does so by searching through tracks 15-25, sector 1, looking for bytes A2C4, the hash codes for BOOT/SYS and DIR/SYS. It also checks the DIR/SYS directory entry for a name match. It determines if the disk is single or double sided by the record size of the directory, and patches the DOS to be able to read the "second" side of the source drive if required.

After this the program converts the source filename to a 1 byte hash code.

The EXMHIT area examines the hit table (sector 1 of the source directory) for a hash code that matches the source filespec. If a matching hash code is found, a calculation is performed to find its relative position in the HIT table. This position is called the files DEC. (Directory Entry Code).

So far, the program has only examined the HIT table. Now, it has to find the source files directory entry. Using the DEC, the directory sector that contains the source file information can be calculated. As each directory sector may store up to 8 filenames, a further calculation on the DEC will result in knowing where the source file directory data is contained within the directory sector.

Next, the program reads the calculated sector into its buffer, and by adding the offset to the beginning buffer address, we can examine and process the source file directory information.

First, bit 7 of byte 0 of the files directory entry is examined. If this bit is reset, this flags that the entry is a primary entry. If bit 7 is set, this warns that this entry is an extended (a secondary) entry. Bit 4 of byte 0 is then checked. If the file is active bit 4 is set. If the file has been killed, this bit is reset. (Newer DOSes only reset this byte when killing a file, TRSDOS 1.3 zeros the entire entry).

As many different filenames may hash to the same "hit" code, the filename must be also checked for an absolute match. No checking occurs on the protection level of the file.

Next, the source files extents are examined and processed. Calculations on the extent bytes will reveal where the program is located on the disk, and how many consecutive sectors are stored together in one 'bunch'.

Now, the RDWRT area now takes over, processing the extent information, reading appropriate sectors, and then saving each sector to the destination file.

The program then loops through all 4 extents if necessary. If an extent contains FFFF, this marks the end of the file, and control is passed back to the main loop, closing the destination file. If all of the extents are processed without finding the FFFF bytes, the last 2 bytes are analyzed. These last two bytes mark if an extended directory record exists. An extended directory entry is another similar directory entry used to store information on further source file extents. If the first byte of these last 2 bytes are not FE, return is passed to close the destination file. If an FE is found, the last byte of the directory entry is saved. This byte represents the DEC of the source files extended directory entry.

The program then loops back to the DEC processing, to read and save more file sectors.

The source file search, examine and locate procedures are the same as how most DOSes search for, locate, and read a file.

Stay tuned next issue for more 1.3 fun! Your comments, ideas and questions regarding the TRSDOS 1.3 corner are welcome.

Gary Campbell is a self taught programmer, qualified chef, baker, and restaurant consultant.

He can be reached at Suite #209, 1051 KLO Road, Kelowna, British Columbia, Canada V1Y-4X6, or through this publication.

```
001 ;"CPY/CMD" ASSEMBLES WITH
002 ;1 FIELD OVERFLOW ERROR
003 ;
004 ;BY GARY E. CAMPBELL
005 ;DONATED TO THE PUBLIC
006 ;DOMAIN 11/25/88
007 ;
008 ;'CPY <CR>' FOR HELP
009 ;
010         ORG     5200H
011 START   LD      A,(HL)
012         CP      0DH
013         JR      NZ,NXTTST
014         CALL    01C9H
015         LD      HL,HLPMSG
016         CALL    021BH
017         RET
018 NXTTST  CP      ':'
019         JP      NZ,COPY
020         INC     HL
021         LD      A,(HL)
022         SUB     30H
023         LD      B,A
024         LD      A,(4413H)
025         CP      B
026         JP      C,DRVERR
027         LD      A,B
028         LD      (SRCDRV),A
029         CALL    GETDIR
030         CALL    01C9H
031         LD      HL,3C00H
032         LD      (CURSOR),HL
033 BOTH    LD      B,10H
034 COUNT   EQU     $-1
035         LD      E,02H
036 SECT    EQU     $-1
037 DISP    PUSH    BC
038         PUSH    DE
039         CALL    CAT
040         POP     DE
041         POP     BC
042         INC     E
043         DJNZ    DISP
044         LD      A,(SIDFLG)
045         OR      A
046         RET     Z
047         LD      A,(COUNT)
048         CP      10H
049         JP      NZ,RETRN
050         LD      A,12H
051         LD      (COUNT),A
052         XOR     A
053         LD      (SECT),A
054         LD      A,(SRCDRV)
055         OR      04H
056         LD      (SRCDRV),A
057         JR      BOTH
058 CAT     LD      A,(DIRTRK)
059         LD      D,A
060         LD      HL,DIRBUF
061         LD      A,(SRCDRV)
062         LD      C,A
063         CALL    4675H
064         JP      NZ,ERROR
065         LD      B,08H
066 DISP1   PUSH    BC
067         PUSH    HL
068         LD      A,(HL)
069         BIT     4,A
070         JR      Z,DISP5
071         BIT     7,A
072         JR      NZ,DISP5
073         LD      B,08H
074         LD      DE,05H
075         ADD     HL,DE
076 DISP2   LD      A,(HL)
077         CP      ' '
078         JR      Z,DISP3
079         CALL    0033H
080 DISP3   INC     HL
081         DJNZ    DISP2
082         LD      A,'/'
083         CALL    0033H
084         LD      B,03H
085 DISP4   LD      A,(HL)
086         CALL    0033H
087         INC     HL
088         DJNZ    DISP4
089         PUSH    HL
090         LD      HL,(CURSOR)
091         LD      DE,10H
092         ADD     HL,DE
093         LD      DE,4000H
094         RST     18H
095         JR      NZ,CURSOK
096         LD      A,0DH
097         CALL    0033H
098         LD      HL,(4020H)
099 CURSOK  LD      (4020H),HL
100         LD      (CURSOR),HL
101         POP     HL
102 DISP5   POP     HL
103         LD      DE,20H
104         ADD     HL,DE
105         POP     BC
106         DJNZ    DISP1
107         RET
108 COPY    CALL    GETSRC
109         CALL    LIST
110         CALL    GETDST
111         CALL    GETDIR
112         CALL    HASHIT
113         CALL    EXMHIT
114         CALL    CREATE
115         CALL    RDWRT
116         LD      A,(FLAG1)
117         OR      A
118         JR      NZ,RETRN
119         CALL    EXMHIT
120         LD      HL,DIRBUF
121         LD      A,(OFFSET)
122         LD      E,A
123         LD      D,00H
124         ADD     HL,DE
125         LD      DE,03H
126         ADD     HL,DE
127         LD      A,(HL)
128         LD      DE,DSTNAM+08H
129         LD      (DE),A
130         LD      B,A
131         INC     HL
132         INC     DE
133         LD      A,(HL)
134         LD      (DE),A
135         LD      DE,10H
136         ADD     HL,DE
137         LD      DE,DSTNAM+0CH
138         LD      A,B
139         OR      A
140         LD      A,(HL)
141         JR      Z,NODEC
142         DEC     A
143 NODEC   LD      (DE),A
144         CALL    CLOSE
145 RETRN   LD      A,(SIDFLG)
146         OR      A
147         RET     Z
148         LD      DE,44D6H
149         LD      HL,TEMP1
150         LD      BC,02H
151         LDIR
152         LD      DE,44ECH
153         LD      C,02H
154         LDIR
155         LD      A,(HL)
156         LD      (4413H),A
157 CLOSE   LD      A,(FLAG1)
158         OR      A
159         RET     NZ
160         LD      DE,DSTNAM
161         CALL    4428H
162         RET
163 LIST    INC     HL
164         LD      A,(HL)
165         DEC     HL
166         CP      '('
167         RET     NZ
168         INC     HL
169 LSTLP   INC     HL
170         LD      A,(HL)
171         LD      C,A
172         CP      ','
173         JR      Z,LSTLP
174         CP      0DH
```

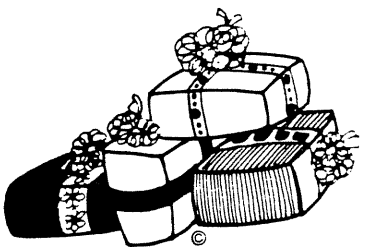
```
175          JR      Z,CHKBYT
176          CP      'A'
177          CALL    Z,CHKASC
178          LD      A,C
179          CP      'P'
180          CALL    Z,CHKPRT
181          LD      A,C
182          CP      'V'
183          CALL    Z,CHKVID
184          LD      A,C
185          CP      'S'
186          CALL    Z,CHKSLW
187          LD      A,C
188          CP      'H'
189          CALL    Z,CHKHEX
190          JR      LSTLP
191 CHKVID   LD      A,(FLAG1)
192          SET     0,A
193 LSTRTN   LD      (FLAG1),A
194          RET
195 CHKHEX   LD      A,(FLAG1)
196          SET     5,A
197          JR      LSTRTN
198 CHKPRT   LD      A,(FLAG1)
199          SET     1,A
200          JR      LSTRTN
201 CHKASC   LD      A,(FLAG1)
202          SET     7,A
203          JR      LSTRTN
204 CHKSLW   LD      A,(FLAG1)
205          SET     6,A
206          JR      LSTRTN
207 CHKBYT   LD      A,(FLAG1)
208          BIT     0,A
209          JR      NZ,BYTOK
210          BIT     1,A
211          JR      NZ,BYTOK
212          SET     0,A
213          LD      (FLAG1),A
214 BYTOK    JP      SETHSH
215 GETSRC   LD      DE,SRCNAM
216          LD      B,18H
217 GETLP    LD      A,(HL)
218          CP      0DH
219          JR      Z,PRMERR
220          CP      ':'
221          JR      Z,SRCPRM
222          CP      ' '
223          JR      Z,NXTCHR
224          LD      (DE),A
225          INC     DE
226 NXTCHR   INC     HL
227          DJNZ    GETLP
228          JR      PRMERR
229 SRCPRM   INC     HL
230          LD      A,(HL)
231          SUB     30H
232          LD      B,A
233          LD      A,(4413H)
234          CP      B
235          JR      C,DRVERR
236          LD      A,B
237          LD      (SRCDRV),A
238          INC     HL
239          LD      A,0DH
240          LD      (DE),A
241          RET
242 PRMERR   LD      A,0BH
243 ERROR    PUSH    AF
244          CALL    RETRN
245          POP     AF
246          POP     HL
247          JP      4409H
248 ERR1     POP     HL
249          JP      RETRN
250 DRVERR   LD      A,20H
251          JR      ERROR
252 BADDSK   LD      A,11H
253          JR      ERROR
254 FILERR   LD      A,1FH
255          JR      ERROR
256 GETDST   LD      A,(FLAG1)
257          OR      A
258          RET     NZ
259          LD      B,18H
260          LD      DE,DSTNAM
261 DSTLP    LD      A,(HL)
262          CP      ' '
263          CALL    Z,CHKIT
264          CP      ' '
265          JR      Z,GETNXT
266          CP      ':'
267          JR      Z,DSTPRM
268          CP      0DH
269          JR      Z,PRMERR
270          LD      (DE),A
271          INC     DE
272 GETNXT   INC     HL
273          DJNZ    DSTLP
274          JR      PRMERR
275 CHKIT    PUSH    HL
276          INC     HL
277          LD      A,(HL)
278          CP      'T'
279          JR      Z,CHECK1
280 CHKRTN   POP     HL
281          LD      A,(HL)
282          RET
283 CHECK1   INC     HL
284          LD      A,(HL)
285          CP      'O'
286          JR      Z,CHECK2
287          JR      CHKRTN
288 CHECK2   INC     HL
289          LD      A,(HL)
290          CP      ' '
291          JR      NZ,CHKRTN
292          INC     HL
293          POP     AF
294          LD      A,(HL)
295          RET
296 DSTPRM   INC     HL
297          LD      A,(HL)
298          SUB     30H
299          LD      B,A
300          LD      A,(4413H)
301          CP      B
302          JR      C,DRVERR
303          LD      A,B
304          LD      (DSTDRV),A
305          LD      A,0DH
306          LD      (DE),A
307          LD      HL,DSTNAM
308          LD      A,(HL)
309          CP      0DH
310          JR      NZ,ADDDRV
311          LD      HL,SRCNAM
312          LD      DE,DSTNAM
313          LD      B,18H
314 DNAMLP   LD      A,(HL)
315          CP      0DH
316          JR      Z,ADDDRV
317          LD      (DE),A
318          INC     HL
319          INC     DE
320          DJNZ    DNAMLP
321 ADDDRV   LD      A,':'
322          LD      (DE),A
323          INC     DE
324          LD      A,(DSTDRV)
325          OR      30H
326          LD      (DE),A
327          INC     DE
328          LD      A,0DH
329          LD      (DE),A
330 SETHSH   LD      HL,SRCNAM
331          LD      DE,HSHNAM
332          LD      B,08H
333 HSHCHR   LD      A,(HL)
334          CP      '/'
335          JR      Z,GETEXT
336          LD      (DE),A
337          INC     HL
338          INC     DE
339          DJNZ    HSHCHR
340 GETEXT   LD      DE,HSHNAM+08H
341          LD      BC,10H
342          LD      A,'/'
343          CPIR
344          RET     NZ
345          LD      BC,03H
346          LDIR
347          RET
348 GETDIR   LD      A,(SRCDRV)
```

```
349          LD      C,A
350          LD      DE,0F01H
351 AGAIN    LD      HL,DIRBUF
352          CALL    4675H
353          JP      NZ,ERROR
354 TSTONE   LD      A,(HL)
355          CP      0A2H
356          JR      Z,TSTTWO
357 NXTTRK   INC     D
358          LD      A,D
359          CP      19H
360          JP      Z,BADDSK
361          JR      AGAIN
362 AGAIN2   LD      E,01H
363          JR      NXTTRK
364 NOPE     POP     DE
365          JR      AGAIN2
366 TSTTWO   INC     HL
367          LD      A,(HL)
368          CP      0C4H
369          JR      NZ,NXTTRK
370          LD      E,03H
371          LD      HL,DIRBUF
372          CALL    4675H
373          JP      NZ,ERROR
374          LD      HL,DIRBUF+05H
375          PUSH    DE
376          LD      DE,DIR
377          LD      B,04H
378 COMP     LD      A,(DE)
379          CP      (HL)
380          JR      NZ,NOPE
381          INC     HL
382          INC     DE
383          DJNZ    COMP
384          POP     DE
385          LD      A,D
386          LD      (DIRTRK),A
387          LD      HL,DIRBUF+17H
388          LD      A,(HL)
389          CP      05H
390          RET     NZ
391 DBLSID   LD      A,01H
392          LD      (SIDFLG),A
393          LD      HL,44D6H
394          LD      DE,TEMP1
395          LD      BC,02H
396          LDIR
397          LD      HL,44ECH
398          LD      C,02H
399          LDIR
400          LD      A,(4413H)
401          LD      (DE),A
402          LD      HL,DBSIDE
403          LD      (44D6H),HL
404          LD      (44ECH),HL
405          LD      A,07H
406          LD      (4413H),A
407          RET
408 DBSIDE   ADD     A,TABLE
409          LD      (LOADIT),A
410          LD      A,(TABLE)
411 LOADIT   EQU     $-2
412          RET
413 TABLE    DEFB    01H
414          DEFB    02H
415          DEFB    04H
416          DEFB    08H
417          DEFB    11H
418          DEFB    12H
419          DEFB    14H
420          DEFB    18H
421 HASHIT   LD      B,0BH
422          LD      DE,HSHNAM
423          LD      C,00H
424 HASHLP   LD      A,(DE)
425          INC     DE
426          XOR     C
427          RLCA
428          LD      C,A
429          DJNZ    HASHLP
430          LD      A,C
431          OR      A
432          JR      NZ,ENDHSH
433          INC     A
434 ENDHSH   LD      (HASH),A
435          RET
436 EXMHIT   LD      HL,DIRBUF
437          LD      (DIRADD),HL
438          LD      HL,0100H
439          LD      (CMPCNT),HL
440          XOR     A
441          LD      (FPDE),A
442 EXAM2    LD      HL,DIRBUF
443          LD      A,(DIRTRK)
444          LD      D,A
445          LD      E,01H
446          LD      A,(SRCDRV)
447          LD      C,A
448          LD      HL,DIRBUF
449          CALL    4675H
450          JP      NZ,ERROR
451          LD      A,(HASH)
452          LD      HL,(DIRADD)
453          LD      BC,(CMPCNT)
454          CPIR
455          JP      NZ,FILERR
456          LD      (DIRADD),HL
457          LD      (CMPCNT),BC
458          DEC     HL
459          LD      DE,DIRBUF
460          XOR     A
461          SBC     HL,DE
462 EXAM3    LD      A,L
463          AND     1FH
464          ADD     A,02H
465          LD      E,A
466          LD      A,(DIRTRK)
467          LD      D,A
468          LD      A,(SRCDRV)
469          LD      C,A
470          LD      A,L
471          AND     0E0H
472          LD      (OFFSET),A
473          LD      HL,DIRBUF
474          CALL    CNVRT2
475          CALL    4675H
476          JP      NZ,ERROR
477          LD      A,(OFFSET)
478          LD      E,A
479          LD      D,00H
480          ADD     HL,DE
481          LD      A,(FPDE)
482          OR      A
483          RET     NZ
484          LD      A,(HL)
485          BIT     7,A
486          JR      NZ,EXAM2
487          PUSH    HL
488          LD      DE,14H
489          ADD     HL,DE
490          LD      A,(HL)
491          LD      (MAXSEC),A
492          POP     HL
493          LD      DE,05H
494          ADD     HL,DE
495          LD      DE,HSHNAM
496          LD      B,0BH
497 CPLOOP    LD      A,(DE)
498          CP      (HL)
499          JP      NZ,EXAM2
500          INC     HL
501          INC     DE
502          DJNZ    CPLOOP
503          RET
504 CREATE   LD      A,(FLAG1)
505          OR      A
506          RET     NZ
507          LD      DE,DSTNAM
508          LD      HL,BUFF
509          LD      B,00H
510          CALL    4420H
511          JP      NZ,ERROR
512          JR      C,COPYMS
513          LD      HL,PROMPT
514          CALL    021BH
515 INPUT    LD      A,0EH
516          CALL    0033H
517          CALL    0049H
518          PUSH    AF
519          LD      A,0FH
520          CALL    0033H
521          POP     AF
522          CP      'Y'
```

```
523          JR      Z,OVWRT1
524          CP      'N'
525          JR      NZ,INPUT
526          CALL    OVWRIT
527-         JP      ERR1
528 OVWRIT   CALL    0033H
529          LD      A,0DH
530          CALL    0033H
531          RET
532 OVWRT1   CALL    OVWRIT
533 COPYMS   LD      HL,CPYMSG
534          CALL    021BH
535          LD      HL,(4020H)
536          LD      (CURSOR),HL
537          RET
538 RDWRT    LD      B,00H
539 EXTAGN   PUSH    BC
540          LD      A,(SRCDRV)
541          LD      C,A
542          LD      HL,DIRBUF
543          LD      A,(OFFSET)
544          LD      E,A
545          LD      D,00H
546          ADD     HL,DE
547          LD      DE,16H
548          ADD     HL,DE
549          LD      A,B
550          ADD     A,A
551          LD      E,A
552          ADD     HL,DE
553          LD      A,B
554          CP      04H
555          JR      NZ,NXT1
556          LD      A,(HL)
557          CP      0FEH
558          JR      NZ,NXT1
559          INC     HL
560          LD      A,01H
561          LD      (FPDE),A
562          LD      A,(HL)
563          LD      L,A
564          POP     BC
565          CALL    EXAM3
566          JR      RDWRT
567 NXT1     LD      A,(HL)
568          CP      0FFH
569          JR      NZ,NXT2
570          INC     HL
571          LD      A,(HL)
572          CP      0FFH
573          JR      NZ,NXT3
574 NXT4     POP     BC
575          RET
576 NXT3     DEC     HL
577 NXT2     LD      D,(HL)
578          INC     HL
579          LD      A,(HL)
580          AND     1FH
581          OR      A
582          INC     A
583          ADD     A,A
584          LD      B,A
585          ADD     A,A
586          ADD     A,B
587          LD      (RECS),A
588          LD      A,(HL)
589          AND     0E0H
590          LD      B,05H
591          OR      A
592 ROTATE   RRA
593          DJNZ    ROTATE
594          OR      A
595          ADD     A,A
596          LD      B,A
597          ADD     A,A
598          ADD     A,B
599          LD      (STSEC),A
600          CP      12H
601          JR      C,NOCNV
602          CALL    CNVRT
603 NOCNV    LD      E,A
604          LD      A,(RECS)
605          LD      B,A
606          LD      HL,BUFF
607 READ     PUSH    BC
608          CALL    4675H
609          POP     BC
610          JP      NZ,ERR2
611          PUSH    BC
612          PUSH    DE
613          LD      A,(MAXSEC)
614          LD      B,A
615          LD      A,(COUNTS)
616          CP      B
617          JR      Z,SKIP
618          INC     A
619          LD      (COUNTS),A
620          LD      A,(FLAG1)
621          OR      A
622          JR      Z,WRITE
623          LD      DE,BUFF
624          LD      B,00H
625          LD      A,(FLAG1)
626          LD      C,A
627 PRMLP    PUSH    DE
628          PUSH    BC
629          LD      A,(DE)
630          LD      B,A
631          BIT     7,C
632          CALL    NZ,ORIT
633          BIT     0,C
634          CALL    NZ,VID
635          LD      A,B
636          BIT     1,C
637          CALL    NZ,PRNT
638          POP     BC
639          POP     DE
640 PRMLP1   LD      A,(3971H)
641          CP      01H
642          JR      Z,PRMLP1
643          CP      04H
644          JP      Z,ERR3
645          INC     DE
646          DJNZ    PRMLP
647 SKIP     XOR     A
648          JR      SKIPIT
649 WRITE    LD      DE,DSTNAM
650          CALL    4439H
651          JR      NZ,SKIPIT
652          PUSH    HL
653          LD      A,(COUNTS)
654          LD      L,A
655          LD      H,00H
656          LD      (4121H),HL
657          LD      BC,0000H
658          LD      HL,(CURSOR)
659          CALL    132FH
660          LD      A,20H
661          LD      (HL),A
662          POP     HL
663          XOR     A
664 SKIPIT   POP     DE
665          POP     BC
666          JP      NZ,ERR2
667          INC     E
668          CALL    CNVRT1
669          DJNZ    READ
670          POP     BC
671          INC     B
672          JP      EXTAGN
673 ORIT     AND     7FH
674          LD      B,A
675          RET
676 VID      PUSH    BC
677          BIT     5,C
678          JR      Z,NORM
679          LD      B,A
680          CALL    CONV
681          CALL    0033H
682          LD      A,B
683          CALL    CONV1
684 NORM     CALL    0033H
685          LD      B,05H
686          BIT     6,C
687          CALL    NZ,0060H
688          LD      B,03H
689          CALL    0060H
690          POP     BC
691          RET
692 PRNT     PUSH    BC
693          BIT     5,C
694          JR      Z,NORM1
695          LD      B,A
696          CALL    CONV
```

```
697          LD      C,A
698          CALL    003BH
699          LD      A,B
700          CALL    CONV1
701 NORM1    LD      C,A
702          CALL    003BH
703          POP     BC
704          RET
705 CONV     OR      A
706          RRA
707          RRA
708          RRA
709          RRA
710          AND     0FH
711          CP      0AH
712          JR      C,NOADD
713          ADD     A,07H
714 NOADD    ADD     A,30H
715          RET
716 CONV1    AND     0FH
717          CP      0AH
718          JR      C,NOADD1
719          ADD     A,07H
720 NOADD1   ADD     A,30H
721          RET
722 ERR2     POP     BC
723          CALL    CLOSE
724          JP      ERROR
725 ERR3     POP     DE
726          POP     BC
727          POP     BC
728          RET
729 CNVRT    LD      A,04H
730          ADD     A,C
731          LD      C,A
732          LD      A,(STSEC)
733          SUB     12H
734          LD      (STSEC),A
735          RET
736 CNVRT1   LD      A,E
737          CP      12H
738          RET     C
739          LD      A,(SIDFLG)
740          OR      A
741          JR      Z,TRACK
742          LD      E,0
743          LD      A,C
744          CP      04H
745          JR      C,SIDE1
746          SUB     04H
747          LD      C,A
748          INC     D
749          RET
750 SIDE1    OR      04H
751          LD      C,A
752          RET
753 TRACK    INC     D
754          LD      E,0
755          RET
756 CNVRT2   LD      A,E
757          CP      12H
758          RET     C
759          SUB     12H
760          LD      E,A
761          LD      A,C
762          ADD     A,04H
763          LD      C,A
764          RET
765 PROMPT   DEFM    'FILE EXITS!!
OVERWRITE'
766          DEFM    '-IT ???  (Y/N):'
767          DEFB    03H
768 CPYMSG   DEFM    'COPYING RECORD:'
769          DEFB    03H
770 HLPMSG   DEFM    '"CPY/CMD" SUPPORTS
THE'
771          DEFM    ' FOLLOWING
PARAMETERS:'
772          DEFB    0AH
773          DEFB    0AH
774          DEFM    'A) CPY
SOURCE/EXT:D TO'
775          DEFM    '
DESTIN/EXT.PASS:D'
776          DEFB    0AH
777          DEFM    'B) CPY
SOURCE/EXT:D'
778          DEFM    '
DESTIN/EXT.PASS:D'
779          DEFB    0AH
780          DEFM    'C) CPY
SOURCE/EXT:D :D'
781          DEFB    0AH
782          DEFM    'D) CPY :D'
783          DEFB    0AH
784          DEFM    'E) CPY SOURCE:D
(V,S,P,A,H)'
785          DEFB    0AH
786          DEFB    0AH
787          DEFM    'OPTION D DISPLAYS
ALL'
788          DEFM    ' SOURCE DISK
FILENAMES ONLY!'
789          DEFM    ' OPTION E'
790          DEFB    0AH
791          DEFM    'LISTS THE SOURCE
FILE'
792          DEFM    ' USING THE
FOLLOWING'
793          DEFM    ' OPTIONAL
PARAMETERS:'
794          DEFB    0AH
795          DEFB    0AH
796          DEFM    'P=PRINTER
S=SLOW OUTPUT'
797          DEFM    '    V=VIDEO
(DEFAULT)'
798          DEFB    0AH
799          DEFM    'H=HEX ASCII  (8
BIT DEFAULT)'
800          DEFM    '  A=7 BIT ASCII'
801          DEFB    0AH
802          DEFM    '@ KEY=PAUSE  BREAK
KEY=ABORT'
803          DEFM    '  (KEYS ACTIVE
OPTION E ONLY)'
804          DEFB    0DH
805 DIR      DEFM    'DIR '
806 STSEC    DEFB    00H
807 RECS     DEFB    00H
808 MAXSEC   DEFB    00H
809 COUNTS   DEFB    00H
810 DIRADD   DEFW    0000H
811 CMPCNT   DEFW    0000H
812 CURSOR   DEFW    0000H
813 SRCDRV   DEFB    00H
814 DSTDRV   DEFB    00H
815 OFFSET   DEFB    00H
816 FPDE     DEFB    00H
817 HSHNAM   DEFS    '
818 SRCNAM   DEFM    '
     '
819 DSTNAM   DEFM    '
     '
820          DEFS    1AH
821 TEMP1    DEFW    0000H
822          DEFW    0000H
823          DEFB    00H
824 SIDFLG   DEFB    00H
825 FLAG1    DEFB    00H
826 DIRTRK   DEFB    00H
827 HASH     DEFB    00H
828 DIRBUF   DEFS    256
829 BUFF     DEFS    256
830          END     START
```

Note: HSHNAM is 11 spaces
SRSNAM is 24 spaces
DSTNAM is 24 spaces



Tim's PD Express

by Timothy Sewell

Congratulations!

We have made it to a second year of TRS-Times!
I thank you for the support and encouragement over the last 6 issues and I hope to meet your expectations and keep up the quality of information you have come to expect and enjoy.

Before we get to the "meat" of this month's column, let's get the plugs out of the way first... The Big news this month is that The File Cabinet's ORCHESTRA-90 music file catalog is now ready to ship. Its more than 75 disks are crammed full of music files that are compiled and ready to play on your computer with the Orchestra-90 music synthesizer module. Be sure to make The File Cabinet your prime source for Orchestra-90 Music files!

There are still some issues of CLOAD magazine that are missing from my reference collection. March 1978 through October 1978 as well as November and December 1983 are missing. If you can provide information to the whereabouts of these elusive CLOAD Cassettes, please contact me.
The SOFTSIDE disk/cassette reference collection is almost complete as well. I am still looking for October 1978 through December 1978, and September 1979 through January 1980. I am also looking for ANY of the SOFTSIDE adventure series disks/cassettes (You wouldn't believe how much of this stuff shows up as Public Domain!).

Now that the plugs are out of the way, let's talk about something that is on the minds of many of the people who write to me... What is the best communications program to use for the Model 4?
There are several available to you but the two that stand out are XT4 by Bill Andrus and FASTTERM by Mel Patrick.

XT4

XT4 is the program that I suggest to people who are just starting out in the communications hobby. It is a simple yet powerful program that is easy to use and can be learned quickly. By depressing the BREAK key anywhere in the program, you get a menu functions available to you so that you can configure the program to your individual needs.
The most powerful feature of this program is its automatic dialing function. With this feature you can store and retrieve up to 99 different phone numbers and have them all pre-configured for dialing. The program will also let you configure automatic log on sequences that a host computer will recognize and you can program macro keys that will issue instructions with a single keystroke.
XT4 also has a built in HELP feature that will explain every function that is available to you within the program. If you get stuck on a feature, use the HELP function. XT4 supports XMODEM file transfer (both CRC and Checksum) as well as ASCII buffer capture. If your Model 4 has 128K, the program will automatically use the extra memory as additional buffer giving you 84K of buffer space available.
XT4 also has the ability to configure to three of the more popular modems used with TRS-80 computers. It will automatically adapt itself via the Configuration function to Radio Shack Modems, HAYES and compatable modems, and the Model 4P internal modem. Many Model 4P users have said that this is a godsend!
The program comes with full documentation and all the files you will need to run the program. At this writing, I understand that Bill Andrus is no longer offering support for XT4. It was released under the SHAREWARE concept, but donations and responses were so poor that Bill sacked his Model 4 and went with the PC crowd (see what happens when you don't support SHAREWARE authors).

XT4 can be found on most Bulletin Board Systems and is available through The File Cabinet on disk MD4COM02.

FASTTERM

FASTTERM by Mel Patrick is for the more advanced user. In addition to many of the features mentioned in XT4, FastTerm adds SEVERAL enhancements that will make your BBS experience much more pleasurable.

FastTerm is window controlled. That is to say that it's various functions are available in windows that pop on to your screen without disturbing the existing data. The program can emulate several different types of terminals (VT100 is very popular), and one of the best features is a split screen mode that allows you to use the top two lines of the screen for typing a message before sending it to the host computer. This comes in very handy for conferences where several computers are talking at the same time. You no longer have to put up with on going messages screwing up your typing.

Another feature of FastTerm is the 1K Xmodem (sometimes known as Ymodem) file transfer protocol. This allows you to transfer files in 1K blocks rather then the standard 128 byte blocks of Xmodem. This can save a bit of time when transferring large files.

Other features that are a part of FastTerm include Area Code Search, logging of long distance charges, online time clock, and many others.

FastTerm is SHAREWARE and a $15 registration fee to Mel Patrick brings a Printed manual and an updated disk with the latest release of FastTerm and well as other programs to try. The Shareware release of FastTerm does NOT contain full documentation though a simple help file is included to get you started.

Like I said, FastTerm is not recommended for the beginning communications hobbyist. If you do not register and obtain the manual, you will be lost when it comes to many of the advanced features.

FastTerm has gone through many modifications and upgrades over the last few months and it gets better with each release. The current version that I use is 14.5 and it does just about everything that I want it to do. The YMODEM transfer protocol saves you a bit of time and money when you download those huge MacPaint picture files. YMODEM is also available on GEnie for faster file transfer.

FastTerm can be found on most TRS-80 BBS systems as well as The File Cabinet's library disk MD4COM13.

I have just received a new communications program called PCPLUS version 1.2 by Michael Bailey. I have not had the chance to try out the program but it's documentation states that it has a 208 number "in memory" BBS directory, unlimited changeable directories, unlimited macro keys, multi-dial up to 208 numbers, optionally linked BATch files, expanded terminal operation, and animated graphics and sound displays.

I have not had the chance to explore this program as I have just received it, but you will hear about it when I am finished running it through the mill. Watch for it!

That just about wraps up this issue. I hope you all had a great holiday season and I wish everybody the best in the coming year.

Make Mine TRS-80!

from HOLLAND

Model 4 - Basic

Cross reference your VISICALC files with

VCXREF4

by Fer Cornet

When a Visicalc worksheet increases in size, a Cross Reference utility is often needed to document the formulas used. Using the 'save to printer' option, <press /SS*PR>, described in the Chapter on the 'Print Command', you can print the worksheet entry position by entry position, from the lower-right entry to the upper-left entry, followed by format information. As each entry position takes one line, much paper is wasted when printing a large worksheet this way. In practice the need for listing is restricted to the entries containing the sometimes very complicated formulas. Labels and numerical constants are already so often documented as you printed an image copy of the worksheet just as it appears on the screen.

In 80-MICRO 5/84 I found a Basic Mod III program that selects from the Visicalc worksheet only the formulas for printing, thus saving a lot of paper. I converted the program to Mod 4 Basic, and discovered that by extending it, the program could be turned into an excellent Cross Reference utility.

Since a Visicalc file is saved to disk as a sequential file, and since sequential files are pure ASCII, the data can be handled with all the powerful techniques of string manipulation available from Basic.

The VCXREF/BAS program does not only give a listing of all formulas, but can now also execute a search for formulas in a specified rectangle. From a worksheet with upper-left field A1 and lower-right field BK200, you can restrict the search to for instance, **C10 - H60**. You can also restrict the search to a single formula, Visicalc command, arithmetic or logical function, operator or target-coordinate. So if you want to know all fields containing the function **@COS(** or all fields in which is refenced to the coordinate **C10** the hunt will only be for such an item.

The report of the reference procedure will appear on your screen along with the number of matches. Specifying option **P** will do the same, but will also send it to the printer. The 'save to printer' option in Visicalc produces a listing starting with the field with the highest coordinate down to the field with the lowest coordinate, which is in my opinion rather illogical. VCXREF/BAS output is in the order from low to high.

A manual is not needed. I tried to keep it user-friendly by clear prompts and entry control routines. Many Remarks are also included to make the program self-explanatory. As to the use of Basic it may be of interest to point to the following items:

line 300. STAMP$ UPDATE SUBROUTINE.

I inserted this SBR which was published in 80-MICRO 5/87 and found it very helpful. It is a nice example of using VARPRT to change the contents of a string-variable.

line 30. TABLE OF CONTENTS.

If you are wondering why the lines are preceded by GOTO statements, the reason is that they change automatically each time that lines are renumbered or removed during development of the program. In fact the program itself jumps over the table of contents, so these GOTO statements are never executed.

line 100. PRINTER CONTROL.

Contrary to the model 3, where a memory address in the ROM is tested, testing of printer availability in model 4 is to be done on the status of the relevant IN/OUT Port.

```
1 CLEAR:GOTO 10
2  STAMP$ = "last update: 880307  09:00:18"
3 GOSUB 300' update subroutine
4 SAVE "VCXREF4/BAS:1": END
9 GOTO 2200
10 CLS 'Cross Reference Utility for Visicalc files'
11 'based on the principles published in 80-
MICRO 5/84 #52 page 66 by David A. Williams.
12 'adapted and extended for Mod 4 Basic by
Fer Cornet, Neerkanne 44, Amsterdam -
Buitenveldert, Holland
13 'Search can be restricted to formulae inside
a specified rectangle or containing specified com
mands, or to coordinates in which is referenced'
14 'to a specified other coordinate.
18 GOTO 97
30 '          TABLE OF CONTENTS
40 GOTO 99  'Subroutines'
42 GOTO 100 'printer operational?
44 GOTO 200 'lowcase characters to uppercase
46 GOTO 300  'update stamp'
50 GOTO 500 'set coordinate to alfabetic a/o
numeric format
52 GOTO 600 'entry organization for filename
a/o parameters
54 GOTO 1000 'initiation
56 GOTO 1200 'restriction of search to
```

```
specified area of sheet
58 GOTO 1400 'restriction of search to
specified keyword,operand,target
60 GOTO 1500 'Z-80 searches selection wanted
62 GOTO 1700 'output to screen
63 GOTO 2000 'output to printer
64 GOTO 2200 'exit protection
65 GOTO 2300 'error handling
66 GOTO 2400 'any suggestions?
67 '    END TABLE OF CONTENTS
97 GOTO 1000
99 'Subroutine: printer operational? (The
FLAG4% values are for STAR SG-10 and may
need replacing by the values for user's own
printer)
100 FLAG4% = INP(248): CLS
105 WHILE FLAG4% < >61
110 IF FLAG4% = 29 THEN PRINT @(2,5),"error!
engaged but not 'on line'";CHR$(30);
115 IF FLAG4% = 93 THEN PRINT @(2,5),"error!
no power or not 'on line'"; CHR$(30);
120 IF FLAG4% = 157 THEN PRINT @(2,5),
"error! not 'on line'";CHR$(30);
125 IF FLAG4% = 221 THEN PRINT @(2,5),
"error! paper out";CHR$(30);
130 SOUND 7,0: FOR DELAY = 1 TO 300:
NEXT DELAY: FLAG4% = INP(248)
135 WEND:CLS
140 RETURN
199 'subroutine: lowcase characters to uppercase
200 UPPERCASE$ = "":
FOR I = 1 TO LEN(LOWCASE$)
205 CHR = ASC(MID$(LOWCASE$,I,1)):
IF CHR = 97 THEN CHR = CHR-32
210 UPPERCASE$ = UPPERCASE$ +
CHR$(CHR):NEXT:RETURN
299 'subroutine: update stamp - invoked by
<BREAK> type GOTO 300
300 C = VARPTR(STAMP$)
305 MSB = PEEK(C + 2)*256:LSB = PEEK(C + 1):
ADDRS = MSB + LSB
310 DATUM$ = RIGHT$(DATE$,2) +
LEFT$(DATE$,2) + MID$(DATE$,4,2)
315 G$ = "last update: " + DATUM$ + " " + TIME$
320 FOR T = 1 TO PEEK(C)
325 H = (ADDRS + T-1):
IF H32767 THEN H = H-65536!
330 POKE H,ASC(MID$(G$,T,1))
335 NEXT T
340 RETURN
499 'subroutine: set coordinate to alfabetic a/o
numeric format
500 COORDINATE$ = LEFT$(COORDINATE$,
POST%-1):COORDINATE$ = MID$(COORDINA
TE$,2)
505 IF ASC(MID$(COORDINATE$,2,1)) = 48
AND ASC(MID$(COORDINATE$,2,1)) < = 57
THEN COORDINATEC$ = "0" + LEFT$(COOR
DINATE$, 1):COORDINATER$ = MID$(COOR
DINATE$,2) ELSE COORDINATEC$ = LEFT$
(COORDINATE$, 2):COORDINATER$ = MID$
(COORDINATE$,3)
510 IF LEN(COORDINATER$) = 2 THEN COOR
DINATER$ = "0" + COORDINATER$
515 IF LEN(COORDINATER$) = 1 THEN COORDI
NATER$ = "00" + COORDINATER$
520 RETURN
599 'subroutine: entry organization of filename
a/o parameters
600 POST% = INSTR(B$,"("): IF POST% = 0 THEN
PAR$ = "" ELSE PAR$ = MID$(B$,POST%):
B$ = LEFT$(B$,POST%-1)
605 IF RIGHT$(B$,2) = " " THEN PRINT"
wrong input format - type again";:GOTO 1050
610 IF RIGHT$(B$,1) = CHR$(32) THEN
B$ = LEFT$(B$,LEN(B$)-1)
615 POST% = INSTR(B$,"/"): IF POST% = 0 THEN
NAM$ = B$:GOTO 625 ELSE NAM$ = LEFT$(B$,
POST%-1)
620 IF LEN(MID$(B$,POST%))4 THEN PRINT
"wrong extension - type again";: GOTO 1050
625 IF LEN(NAM$) > 8 THEN PRINT "wrong
filename - type again";:GOTO 1050
630 IF INSTR(B$,"/") = 0 THEN B$ = B$ + "/VC"
635 IF INSTR(PAR$,"P") < >0 THEN FLAG5% = -1
ELSE FLAG5% = 0
640 IF INSTR(PAR$,"I") < >0 THEN FLAG6% = -1
ELSE FLAG6% = 0
645 RETURN
999 'initiation
1000 GOSUB 100:DIM A$(500)
1005 SYSTEM "reset *pr":POKE &HB94,
PEEK(&HB94) AND 248
1007 'protective reset - protective removal of
scroll protection
1010 PRINT"This Cross Reference Utility is for
VISICALC files only"
1020 PRINT"type <filename/ext (parameters) >
/ext default = /VC"
1030 PRINT TAB(12)" the parameters are: P -
directs output to printer"
1040 PRINT TAB(34)"I - with individual sheet
pausing":PRINT TAB(28)"default = output to
screen"
1050 PRINT TAB(34)"";:LINE INPUT B$:
LOWCASE$ = B$:GOSUB 200:B$ = UPPERCASE$
1060 GOSUB 600 'check input-format'
1070 ON ERROR GOTO 1090
1075 OPEN "I",1,B$
1080 CLOSE:PRINT:GOTO 1200
1090 IF ERR = 53 THEN PRINT"file "B$" not
found.":PRINT TAB(20) "try again";:
RESUME 1050
1095 IF ERR = 55 THEN PRINT"open file -
now being closed.":CLOSE:PRINT TAB(20) "try
again!";:RESUME 1050 ELSE 2300
1097 '
1199 'restricting search to specified area of sheet
1200 PRINT"if you wish to restrict selection to
for mulae INSIDE a particular"
1210 PRINT"area of the VC-sheet only, then
input <UPPER LEFT> and"
1220 PRINT"<LOWEST RIGHT> of that area;
default = <ENTER>."
1230 PRINT TAB(16)"";
1240 LINE INPUT "type upper left column-
```

```
row <COORDINATE> ";UPLEFT$
1245 IF UPLEFT$<>"" THEN IF ASC(LEFT$
(UPLEFT$,1))<65 THEN PRINT"column first! ";:
GOTO 1240
1250 LOWCASE$=UPLEFT$:GOSUB 200:
UPLEFT$=UPPERCASE$
1260 IF UPLEFT$<>"" THEN COORDINATE$
=UPLEFT$:GOSUB 505:ULC$=COOR
DINATEC$:
ULR$=COORDINATER$ ELSE FLAG1%=-1
1270 IF ULC$>"BK" THEN PRINT"maximum =
BK  ";: GOTO 1240
1280 IF ULR$>"254" OR LEN(ULR$)>3 THEN
PRINT "maximum = 254 ";:GOTO 1240
1285 IF ULR$="000" THEN PRINT "0 impos
sible";:GOTO 1240
1290 PRINT TAB(16)"";
1300 LINE INPUT "type lowest right column
-row <COORDINATE ";LOWRIGHT$
1305 IF LOWRIGHT$<>"" THEN IF ASC
(LEFT$(LOWRIGHT$,1))<65 THEN PRINT"column
first!";:GOTO 1300
1310 LOWCASE$=LOWRIGHT$:GOSUB 200:
LOWRIGHT$=UPPERCASE$
1320 IF LOWRIGHT$<>"" THEN COORDINA
TE$=LOWRIGHT$:GOSUB 505:LRC$=COORDIN
ATEC$:LRR$=COORDINATER$ ELSE
FLAG2%=-1
1330 IF LRC$>"BK" THEN PRINT"maximum =
BK  ";: GOTO 1300
1340 IF LRR$>"254" OR LEN(LRR$)>3 THEN
PRINT"maximum = 254 ";:GOTO 1300
1345 IF LRR$="000" THEN PRINT "0 impos
sible";:GOTO 1300
1350 IF LOWRIGHT$<>"" AND (LRC$<ULC$
OR LRR$<ULR$) THEN PRINT"faulty input! ";:
GOTO 1300
1399 'restriction of search to specified comand,
coordinate a/o operand
1400 PRINT:PRINT"if you wish to restrict list
ing to formulae containing a particular"
1410 PRINT"VC-keyword or operand or any tar
get-coordinate, then input keyword, operand or
target-coordinate"
1420 'examples: you wish to list only formulae
containing the command:'
1430 '@LOOKUP( then type: <@LOOKUP(>.
Or you wish to list all lines in which'
1440 'is referenced to target-COORDINATE
C123. then type <C123> and so forth & so on.
1460 LINE INPUT "type <KEYWORD>, <OPERA
ND> or <TARGET>; default = <ENTER>: ";
TARGET$
1470 LOWCASE$=TARGET$:GOSUB 200:
TARGET$=UPPERCASE$
1499 'Z-80 searches selection wanted
1500 PRINT TAB(41)".....searching.....";
1520 OPEN "I",1,B$
1530 I=1
1540 LINE INPUT#1, A$(I)
1550 POST%=INSTR(A$(I),CHR$(58))
1560 IF INSTR(POST%+1,A$(I),TARGET$)=0
THEN 1640
1570 IF LEFT$(A$(I),1)CHR$<>(62) THEN 1640
1580 IF INSTR(A$(I),CHR$(34)>)0 THEN 1640
1590 IF INSTR(A$(I),CHR$(64))+INSTR(A$(I),
CHR$(43))+INSTR(A$(I),CHR$(45))+INSTR(A$(I)
,CHR$(42))+INSTR(POST%+3,A$(I),CHR$(47))
+INSTR(A$(I),CHR$(94))=0 THEN 1640
1600 COORDINATE$=A$(I):GOSUB 500:
AC$=COORDINATEC$:AR$=COORDINATER$
1610 IF FLAG1% THEN 1620 ELSE IF
AR$<ULR$ OR AC$<ULC$ THEN 1640
1620 IF FLAG2% THEN 1630 ELSE IF
AR$>LRR$ OR AC$>LRC$ THEN 1640
1630 I=I+1
1640 IF EOF(1) THEN CLOSE ELSE 1540
1650 NUMBER%=I-1' stack of value number%
1660 '
1699 'output to printer or screen
1700 IF FLAG5% THEN GOTO 1999
1710 SYSTEM "reset *pr"  to be sure about
reset
1720 CLS:PRINT"Formulae and/or references
selected from the Visicalc file named:";CHR$(34);
B$;CHR$(34)
1730 IF TARGET$="" THEN PRINT "the
search is for ALL formulae result: "; ELSE
PRINT"searched for target : ";TARGET$;"
result: ";
1733 IF NUMBER%=0 THEN PRINT "no
matches!" ELSE IF NUMBER%=1 THEN PRINT
NUMBER%"match!" ELSE PRINT NUMBER%;"
matches!"
1740 IF UPLEFT$="" AND LOWRIGHT$=""
THEN PRINT
1750 IF UPLEFT$<>"" THEN PRINT"rectangle's
upper left = ";UPLEFT$;"       ";
1760 IF LOWRIGHT$<>"" THEN PRINT
"rectangle's lowest right$ = ";LOWRIGHT$
1770 'print file in reverse order'
1780 POKE &HB94, PEEK (&HB94) OR 3
' 3 lines scroll protect
1790 N=NUMBER%:L=0
1800 FOR I=N TO 1 STEP -1
1810 PRINT"    ";LEFT$(A$(I),70): L=L+1
1820 IF LEN(A$(I))-70>0 THEN PRINT" ";
RIGHT$(A$(I),LEN(A$(I))-70): L=L+1
1830 IF L>18 THEN PRINT STRING$(19-L,13):
L=0 ELSE 1870
1840 PRINT@(22,17),"<ENTER>  to continue
printing";
1850 J$=INKEY$: IF J$="" THEN 1850
1860 PRINT@(22,17),STRING$(30," ")
1870 NEXT
1880 IF NUMBER%=0 THEN PRINT "no
matches!" ELSE IF NUMBER%=1 THEN PRINT
NUMBER%" match!" ELSE PRINT NUMBER
%;"matches!"
1885 IF L<18 THEN PRINT
1930 GOSUB 2200'       exit protection
1940 POKE &HB94, PEEK (&HB94) AND 248
'remove scroll protection
1950 PRINT"end of run": END
1960 '
1999 'lprint file in reverse order'
```

```
2000 SYSTEM "link *pr *do" data sent to
printer also visible on screen
2010 CLS:LPRINT"Formulae and/or references
selected from the Visicalc file named: ";
CHR$(34); B$;CHR$(34)
2020 IF TARGET$ = "" THEN LPRINT "the
search is for ALL formulae result: "; ELSE
LPRINT "searched for keyword: ";TARGET$;"
      result: ";
2025 IF NUMBER% = 0 THEN LPRINT"no
matches!" ELSE IF NUMBER% = 1 THEN
LPRINT NUMBER%;"match!" ELSE LPRINT
NUMBER%;" matches!"
2030 IF UPLEFT$ < > "" THEN LPRINT
"rectangle's upperleft = ";UPLEFT$;"
      ";
2040 IF LOWRIGHT$ < > "" THEN LPRINT
"rectangle's lowest right$ = ";LOWRIGHT$
2050 N = NUMBER%:L = 0
2070 FOR I = N TO 1 STEP -1
2080 LPRINT"    ";LEFT$(A$(I),70): L = L + 1
2090 IF LEN(A$(I))-70 > 0 THEN LPRINT"
    ";
RIGHT$(A$(I),LEN(A$(I))-70): L = L + 1
2100 IF L > 58 THEN LPRINT STRING$(65-L,
13):L = 0 ELSE 2140
2110 IF FLAG6% THEN PRINT@(22,17),
"<ENTER> to continue printing" ELSE
GOTO 2140
2120 J$ = INKEY$: IF J$ = "" THEN 2120
2130 PRINT@(22,17),STRING$(30," ")
2140 NEXT
2145 IF NUMBER% = 0 THEN LPRINT
"no matches!" ELSE IF NUMBER% = 1 THEN
LPRINT NUMBER%;"match!" ELSE LPRINT
NUMBER%;" matches!"
2150 LPRINT"end of run":LPRINT:LPRINT
2155 GOSUB 2200'      exit protection
2160 SYSTEM "reset *pr":POKE &HB94,
PEEK(&HB94) AND 248
2165 'remove link between printer and screen -
remove scroll protection
2170 END
2199 'exit protection
2200 PRINT@(22,0),"";:LINE INPUT"type <P>
for hard-copy, or <S> for screen repeat,
<ENTER> for exit ";I$
2210 LOWCASE$ = I$:GOSUB 200:
I$ = UPPERCASE$
2220 IF I$ = "P" THEN FLAG5% = -1:
FLAG6% = -1:GOTO 1999
2230 IF I$ = "S" THEN FLAG5% = 0:
FLAG6% = 0:GOTO 1700
2240 PRINT@(22,0),STRING$(70,32):
PRINT@(22,0),"";:LINE INPUT"are you sure you
wish to exit? type <Y> or <N>  ";I$
2250 LOWCASE$ = I$:GOSUB 200:
I$ = UPPERCASE$
2260 IF I$ = "" THEN 2240
2270 IF I$ = "N" THEN GOTO 2200 ELSE
RETURN
2299 'error handling
2300 PRINT"error in line ";ERL;" code:
";ERR;"(manual appendix D, page A- 83)."
:RESUME 2310
2310 PRINT:PRINT"to end this procedure,
press /"
2320 X$ = INKEY$: IF X$ = "" THEN 2320
2330 IF X$ = "/" THEN END ELSE GOTO 1
2399 'any suggestions?
2400 'your suggestions are welcomed by:
TRS&PC club, afd. West, subgroup M3/4,'
2410 '1625 TD 49, HOORN, The Netherlands.
```

ASSEMBLY 101

Z-80 WITHOUT TEARS

by Lance Wolstrup

Yes, I know, - I did it! I opened my big mouth and promised to teach everybody assembly language. I said something like: "It isn't that difficult." Certainly that is true, but then again "ain't nothing difficult, if you know how!" Realistically speaking, I cannot teach you anything about anything. There is only one person in the entire world who can do that: YOU.

Oh yes, I might be able to present the material in a manner which will shed light on the subject, BUT, it is up to you to do the learning. So, rather than your teacher, think of me as your very elementary tourguide into the world of assembly language.

Before we begin, let me say to anyone with assembly language experience who may be reading these articles that I do not intend to 'go by the book'. The 'book' has served only to confuse most beginners, so I am throwing it away. I will give only partial information when I feel it necessary, and in other instances I may be redundant if that suits the purpose. I will not be a 'byte miser'; that is, the code presented might be written shorter or, most likely, much more elegantly. Lastly, I will use my own terminology as I see fit, rather than the 'standard' words or names, if I think it advantageous.

In other words, this will be done MY WAY.

Now that I've covered my rear, let's cut the formalities and get on with the important stuff:

LEARNING ASSEMBLY LANGUAGE.

We will need the following:

1. Model I with TRSDOS 2.3B
or Model III with TRSDOS 1.3
or Model 4 in Model III mode with TRSDOS 1.3
(note: other DOSes may be acceptable; however, no guarantees)

2. EDTASM from Radio Shack
(note: EDTASM from Apparat may be acceptable, but only the RS version will be discussed)

3. A fair knowledge of BASIC programming.
(You need not be an expert, but a knowledge of the 'basics' of BASIC is essential.)

4. Willingness to learn, an open mind and, at least, 1 bottle of aspirin.

ASSEMBLY LANGUAGE VS. BASIC

Just as a musician already knowing how to play one instrument can easily learn to play another, a BASIC programmer can learn another language. You already know the concept of programming - all you need to learn is the mechanics of the other language.

When you program in BASIC you deal with LINENUMBERS, COMMANDS, DATA and VARIABLES. You put these parts together in a logical manner and, voila, you have a program.

Assembly language does not use LINENUMBERS or VARIABLES; instead it uses LABELS, COMMANDS, DATA and REGISTERS (and physical MEMORY LOCATIONS).

BASIC has english-sounding commands that, for the most part, make sense. Assembly language has 2 or 3 letter commands (called OP-CODES) that, to a novice, will make no sense whatsoever. When explained, however, they will mellow out - somewhat.

Another important difference is that when programming in BASIC, you write AND execute the program from BASIC. Assembly language is written using an editor/assembler.

In our case we will use EDTASM which is actually two programs in one: it is a text editor AND an assembler.

First you use the editor to write your program, then you use the assembler to convert your instructions to machine code.

Your program CANNOT be executed from within EDTASM (or any other editor/assembler). You must return to DOS before your newly written program can be executed.

EDTASM

Like BASIC, EDTASM is a program. BASIC came with your DOS; EDTASM did not. It had to be bought from Radio Shack seperately.

Since, at this point in time, you may have obtained this program from other sources, make a working DOS disk and copy EDTASM/CMD over to it. We are now ready to start.

Turn the computer on, insert the DOS disk with EDTASM on it and get to the 'DOS Ready' prompt.

Type: **EDTASM** (ENTER)

You are now in EDTASM and on the top portion of the screen you should see the copyright message. About a third of the way down the screen is an asterisk *. This is EDTASM's prompt for you to do something.

Quick recap:

BASIC prompts with '**OK**'
EDTASM prompts with '*'.

Many options are now available and I will list the more important ones later, but for now let's take it one step at a time.

Since you haven't done anything yet, you have 4 commands available at this very minute:

Q (ENTER) - quit - back to DOS.

M (ENTER) - memory - display text used and available memory.

L filename/ext:dn (ENTER) - load an EDTASM source code file.

I linenumber,increment (ENTER) - insert programming lines.

The (Q)uit command simply exits EDTASM and brings you back to DOS. Nothing difficult to understand here.

The (M)emory command is an aid to let you know how many bytes of text (program) you have used. It will also display how many bytes of memory is still available. At this point this is not an important command - we may never use it.

The (L)oad command brings in a source file that has been created previously by EDTASM. The syntax is: **L filename/ext:dn**.

If the extension is omitted, EDTASM will assume it to be /SRC.

The (I)nsert command is the most important command in EDTASM. It allows you to insert (write) your program. The syntax is: **I linenumber,increment**. This works much like the AUTO command in BASIC.

Since I mentioned above that assembly language does NOT use line numbers, let me clarify why we all of a sudden are using them now:

EDTASM is part text editor and part assembler. When we invoke the **I** command, we are using the editor portion to type in our program. If we were perfect, we would never need to edit our program; but, alas, we are not, so the editor portion of EDTASM has built in editing commands allowing you to modify the text as needed. When you use these editing commands you must reference the text you wish to edit by its associated line number. This is the exact same concept as when editing a BASIC program. Incidentally, the editing commands themselves are almost identical to those you are so familiar with in BASIC.

Here, however, is where the similarity to BASIC ends.

BASIC also uses line numbers to reference a **branch** in the program flow; that is, you can GOTO or GOSUB to a particular line number.

You CANNOT do that in assembly language; instead the branching commands JP (GOTO) and CALL (GOSUB) must be told to branch to a particular MEMORY LOCATION.

Quick recap:

BASIC uses line numbers for editing purposes AND program execution.

EDTASM uses line numbers for editing purposes.

Assembly language does NOT use line numbers for program execution.

Now let's move on to see how the EDTASM editor expects you to type in a program. While the BASIC editor makes you type a line number followed by an instruction or series of instructions and pressing (ENTER) when you are done, EDTASM is a little more rigid. Here you can only type in one instruction per line, and it must be typed correctly over a span of four pre-defined 'fields'.

Field #1 is the LABEL field.
Field #2 is the COMMAND field.
Field #3 is the DATA field.
Field #4 is the COMMENT field.

An assembly language line will have this form:

LINENUMBER LABEL COMMAND DATA ;REMARK

for example:

```
00240 LOOP    LD    A,32    ;Chr$(32) now in A
```

Quick recap:

The line number (00240) is NOT considered a field.

"LOOP" is the LABEL.
"LD" is the COMMAND.
"A,32" is the DATA.
"Chr$(32) now in A" is the COMMENT.
(note that a comment must always be preceded by a semicolon.)

When you first use the (I)nsert command the cursor will sit one space to the right of the line number. This is the LABEL field.

Pressing the (RIGHT ARROW) key moves the cursor to the next field, which is the COMMAND field.

Pressing (RIGHT ARROW) again moves the cursor to the DATA field.

One more press of the (RIGHT ARROW) key moves the cursor to the COMMENT field.

Pressing (LEFT ARROW) moves the cursor to the previous field.

Quick recap:

(RIGHT ARROW) moves cursor to next field.

(LEFT ARROW) moves cursor to previous field.

OK, now that you know how to move the cursor, let's practice:

Type: **I 100,10** (ENTER)

You have told the editor that you wish to start with line 100, and that each new line number should increment by 10. You should see the cursor sitting one space to the right of the line number 00100.

Now practice moving the cursor between fields right and left. Press (ENTER) a couple of times. Notice that the line number increments and the cursor is placed in the LABEL field of the new line.

When you are done, press the (BREAK) key. This gets you back to command mode and the asterisk prompt.

As you have inserted lines (even though they might be blank), we need to get rid of them.

When you want to get rid of everything you have typed in BASIC, you use NEW.

In EDTASM you use the (D)elete command.

Type: **D#:*** (ENTER)

This is shorthand for: DELETE first line to last line.

"#" means first line
":" is the seperator
"*" means last line.

The EDTASM editor is now clear and ready for you to type in a new program. We will do that in just a minute. First, however, let's learn a little bit about Assembly language.

The TRS-80 models I, III & 4 are all built around the Z-80 chip which contains several REGISTERS used for manipulating data.

The word REGISTER may sound scary at first, but relax; think of them as nothing more than VARIABLES, almost like we use in BASIC. We can store numbers there; we can access the stored numbers there, and we can manipulate the numbers there.

These registers are named: **A B C D E H L**
(there are more, but for now we will concentrate on the these seven.)

Each are 8 bit registers; that is, they are only capable of storing numbers ranging from 0 to 255.

When we have the need to access and manipulate numbers in excess of 255 we can combine registers B and C, D and E, H and L, thus making three 16 bit registers named **BC, DE** and **HL**. These 16 bit registers are capable of handling numbers from 0 to 65535.

The A register (also known as the ACCUMULATOR) is very special. It is the only one of the 8 bit registers capable of performing math.

16 bit math can be done by using the HL register.

Notice that register A could not team up with a partner to form a 16 bit register. Actually, it does have a partner. It is called register F (the FLAG register), but it is used for another special purpose which we will discuss in a later installment.

If we can think of registers as being variables, then the following example will easily demonstrate the similarity between the two languages:

```
BASIC:              A=65
                    or    LET A=65

Assembly language:  LD A,65
```

Hopefully, we all understand the BASIC statement, A=65.

Variable A now has the value 65 stored. You can now use the available BASIC commands to use this value in whatever way you see fit.

The Assembly language version, LD A,65, does the very same thing. It LoaDs the value 65 into register A, at which point you can use the available Assembly language commands to manipulate it any way you like.

At this point let's go back to when you first learned BASIC. What was the first thing you learned to do?

I'll bet it was something like:

```
10 PRINT"HI, I AM YOUR TRS-80 COMPUTER"
```

This was easy to understand because you could **see the result** on the screen.

The program can be modified to look like this:

```
10 A$="HI, I AM YOUR TRS-80 COMPUTER"
20 PRINT A$
```

It can be further modified to:

```
10 A$="HI, I'M YOUR TRS-80 COMPUTER"
20 GOSUB 100
30 END
100 PRINT A$
110 RETURN
```

Now, this may be a silly way to write a simple PRINT statement, BUT - **this is how you write it in Assembly language.**

The concept in BASIC is:
Store the text in a **variable**
Jump to the PRINT subroutine and print the text

The concept in Assembly language is:
Store the text in **memory**
Jump to the ROM PRINT subroutine and print the text
Very, very similar, isn't it?

We briefly discussed some of the editing commands that the editor portion of EDTASM understands (Q,M,L,I). The other half of EDTASM, the assembler portion, also has some special commands. These assembler commands (also known as PSEUDO-OPS) are very important, because they will be included in your code to tell the assembler to do certain things for you.

Here are, for now, the important ones:

ORG memory location - tells the assembler where to store your program in memory.

DEFM 'string' - (DEFINE MESSAGE) this tells the assembler that the characters inside the single quotes must be treated as ASCII text rather than extracting the numeric value of each character and then treating the number as a command or a piece of data.

DEFB value - (DEFINE BYTE) tells the assembler to treat this value as numeric data only.

END - indicates the end of the program. While BASIC does not require its use, Assembly language **demands** that the absolute last line of your program is the END statement.

Keep in mind that these commands are ASSEMBLER COMMANDS; that is, they tell EDTASM what to do when you assemble your program into machine code. They are not real Z-80 Assembly language instructions.

Quick recap:

The editing mode uses editing COMMANDS

The assembler mode uses assembler COMMANDS

Assembly language uses Assembly language COMMANDS

At this point may I recommend that you break open your bottle of Aspirin and, maybe, indulge in the beverage of your choice while you relax for a few minutes. Next stop: Writing a program.

GETTING DOWN TO BUSINESS

Let's expand on the Assembly language concept presented at the top of this page:

Tell assembler where to store program in memory
Define your text with the DEFM assembler command
Begin program
Jump to the ROM PRINT subroutine
Make program return to DOS
END program

The program will look like this:

```
00100          ORG  7000H
00110 MSG1     DEFM 'Hi, I am your TRS-80 computer'
00120          DEFB 0DH
00130 BEGIN    LD   HL,MSG1
00140          CALL 021BH
00150          RET
00160          END  BEGIN
```

Note: line 00140 is for **Model III only**.
Model I should substitute: **CALL 4467H**

Type this program into EDTASM, following these step by step instructions:

From the asterisk prompt type: **I 100,10** <ENTER>

The cursor is now immediately next to **00100**
Press <RIGHT ARROW> once and type **ORG**
Press <RIGHT ARROW> once and type **7000H**
Press <ENTER>

Your screen should now display:

```
00100          ORG  7000H
00110
```

Line 00100 tell the assembler to assemble your program into memory starting (ORiGinating) with location 7000H, which is the first available location for user programs. DOS and ROM resides in the locations directly before 7000H.

The cursor is now located next to **00110**
Type **MSG1**
Press <RIGHT ARROW> and type **DEFM**
Press <RIGHT ARROW> and type **'Hi, I am your TRS-80 computer'**
Press <ENTER>

The screen looks this way:

```
00100          ORG  7000H
00110 MSG1     DEFM 'Hi, I am your TRS-80 computer'
00120
```

Line 00110 uses the LABEL **MSG1**. The LABEL could be any alpha-numeric combination of six characters or less, but MSG1 is fine. The LABEL serves as a marker for the assembler. Any time we later refer to MSG1, the assembler will substitute the address of the actual memory the memory location. Next, we **DEF**ined our **M**essage as **Hi, I am your TRS-80 computer.** The LABEL **MSG1** now refers to the memory location containing the first letter of the actual text.

At line **00120** press <RIGHT ARROW> type **DEFB**
press <RIGHT ARROW> and type **0DH**
press <ENTER>

```
00120          DEFB 0DH
```

Line 00120 is a necessary evil. Because the ROM PRINT subroutine is written to quit only when it encounters a **0DH**, we must end our text with that number, otherwise the routine will go off into the dreaded 'never-never land'. You are, in essence, telling the ROM routine that this is the end of the text and, since **0DH** is also the code for a carriage return, to move the cursor to the beginning of the next line (**0DH** is 13 in decimal).

Continuing on line **00130** type the LABEL **BEGIN**
Press <RIGHT ARROW> and type **LD**
Press <RIGHT ARROW> and type **HL,MSG1**
Press <ENTER>

Now the line will appear this way on the screen:
00130 BEGIN LD HL,MSG1

Since line 00130 is the actual beginning of our program, we LABEL it as such. Then we LOAD register HL with the **ADDRESS** of MSG1.

LD is the asembly language command. HL,MSG1 is the data that the LOAD command will work on. It looks at MSG1, realizes that this represents a number, and then proceeds to store that number in register HL.

We can put any number we wish (0 - 65535) into register HL. In this case it suits our pupose for register HL to contain the memory address of where we stored our text. I'll explain why this suits the purpose when I talk about line 00140.

Let's do line 00140 now.
Press <RIGHT ARROW> and type **CALL**
Press <RIGHT ARROW> and if Model III type **021BH** followed by <ENTER>
Model I should type **4467H** followed by <ENTER>

Model III:
```
00140          CALL 021BH
```
or Model I:
```
00140          CALL 4467H
```

Line 00140 is THE important line of the program. It is the one that puts the text on the screen.

The Model III has a ROM routine that performs this task. It is located at memory location O21BH.

Model I uses a DOS routine located at 4467H for this function.

Both routines expect register HL to contain the address of the text to be displayed on the screen, which is why we did just that in line 00130.

CALL is like GOSUB in BASIC. As mentioned earler, CALL does not branch a line number; instead it branches to the specified memory location. There it performs whatever action the subroutine provides and, when done, it returns to our program. The action at Model III's **021BH** and Model I's **4467H** takes the character at the memory address specified by HL at puts it on the screen at the current cursor position. HL is incremented by 1 and the cursor is moved to the next position. HL now contains the address of the second character of our text. This repeats until HL contains the address of the terminating character (0DH). At this point the subroutine RETurns to its caller (our program).

This brings us to our line 00150.

We have accomplished our goal of displaying the text, so we want to return to DOS.

At line 00150 press <RIGHT ARROW> type **RET**
press <ENTER>

```
00150          RET
```

Line 00150 issues the RETurn instruction. If we have done things correctly, and we have, this command will send us back to DOS.

You might think of our program, or any program for that matter, as sort of a subroutine called from DOS. When it sees the RET command it sends us back to the original caller which, of course, is the DOS command line.

There is much more to explain about RET, but we will do that in another installment.

Lastly, we get to line 00160.

This line is mandatory in all Assembly language programs and forgetting to include it will result in a one-way ticket to the dreaded 'never-never land'.

At line 00160 press <RIGHT ARROW> and type **END**
Press <RIGHT ARROW> and type **BEGIN**.
Press <ENTER>

```
00160          END  BEGIN
```

The END statement should always specify a start address. You can use either the LABEL that specified the start of the program, in our case it was BEGIN, or you can specify the actual starting address; in our program it is 7000H.

At line 00170 press the <BREAK> key to get back to the EDTASM * prompt.

We have written the program and we now need to write it to disk. Let's name the program **ASM1/SRC**.

EDTASM's write-to-disk command is **W**.

Type: **W ASM1/SRC** <ENTER>

You have now saved the SOURCE CODE . Next we need to ASSEMBLE the program into an executable CMD file.

EDTASM's assemble command is **A**.

Type: **A ASM1/CMD** <ENTER>

You should see all kinds of weird things scrolling on your screen. In a few seconds, however, you should

see the following toward the bottom of the screen:
00000 Total Errors
BEGIN 701E
MSG1 7000

This tells you that you have made no errors, and that the labels MSG1 and BEGIN is translated into 7000H and 701EH respectively.
If you have made mistakes, type: **D#:*** <ENTER> and start over again.

If everything is correct, you can leave EDTASM by typing: **Q** <ENTER> which will bring you back to DOS. You can now execute you program by typing:
ASM1 <ENTER>

Now that you can display text on the screen, go ahead and practice with ideas of your own. Write a program that displays several lines of text, fill the screen with text, do anything you want, but whatever you do, keep practicing what you just learned.

Next installment will talk quite a bit more about displaying text on the screen. We will learn to POKE the screen from Assembly language, as well as position the cursor for PRINT@ text display, and more.

So until March.....Bye!!

EDTASM editor commands:

A	Assemble source in text buffer
D	Delete specified line or lines
E	Edit mode. Works almost exactly as the BASIC editor.
F	Find a specified string of characters in the text buffer
H	Hardcopy - print specified line or lines on the printer. **H#:*** <ENTER> prints entire listing
I	Insert source line or lines at a specified line with a specified increment
L	Load a source file from disk into text buffer
N	reNumber source lines in text buffer
P	Print to screen (LIST) specified line or lines of source code in text buffer **P#:*** <ENTER> lists all lines
Q	Quit - return to DOS
R	Replace specified line in text buffer Works like the Insert command, only the line is overwritten
T	Same as H, only no line numbers are printed
W	Write text buffer to disk
<RIGHT ARROW>	Cursor to next field
<LEFT ARROW>	Cursor to previous field
<UP ARROW>	Displays previous source line
<DOWN ARROW>	Displays next source line
.	Displays current source line

Copy Limited Files Under TRSDOS - LSDOS 6.X.

by Roy T. Beck

I recently purchased PFS:Files, and noted in the user's manual that it is limited to making 5 backup copies. At first glance this seemed a rather small number, and I was a little concerned in case I ran into trouble. (You never know!)

Being both nosey and cautious, I decided to enter into a little research project to determine as much as possible about operation with a copy-limited program.

The first step was to install a write-protect tab on the original disk. Sort of like AIDS protection. Next, I used one of the aftermarket copying utilities to make a verbatim copy without using up any of the five official copies.

This first copy I identified as disk #0. Disk 0 is a bootable disk, which turned out to have TRSDOS V6.2.0 on it.

I then proceeded to make copy #1, #2, #3, #4, and #5 from disk #0.

When I attempted copy #6, the DOS reported, "Disk contains protected files. Backup reconstruct invoked.", and then went on to make a copy.

I then attempted to boot all 7 disks, #0 through #6.

The first 6 booted properly and AUTO started PFS:File.

The last disk, #6, would bootup OK, and issue the AUTO command. But the DOS then responded with "File not found." A DIR check of the disks showed that #0 through #5 contained all the same files, but #6 was lacking two files; PFS/CMD and FILES/CMD. Obviously these are the protected files.

So far, the "5 permitted copies" seemed valid. BUT!

I next booted up with disk #1 and attempted to back it up to the freshly reformatted #6 disk. No squawks, it backed up just fine. And now #6 boots up PFS just fine!

So I continued with backing up from #1, and ended up with bootable PFS disks #6, #7, #8, and #9. These all booted PFS correctly.

#10 however, lacked the two protected files again.

Next I reformatted #10. I now went to disk #2 and repeated the process. This time I got three working disks, #10, #11, and #12.

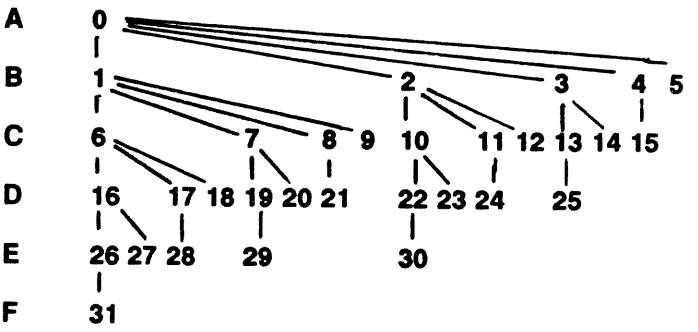
I won't bore you with the repetitions, but instead will refer you to the chart below which reveals what is going on.

There is a counter on the original disk #0, initially set at 5. As each backup is done from this disk, the counter is first checked, and if not zero, decremented and a new backup made. This is a perfectly good, bootable disk with all files on it. The copy's counter matches the decremented value of the disk it is being copied from. Only when the source disk has decremented to zero does it complain and refuse to backup the protected files.

The table is constructed in the following fashion: Row A has only the original disk #0 in it. This disk is initially good for 5 copies, after which the counter is zeroed out. (Actually an FF is placed in the counter at that point, 0 never appears). The next row B has 5 working copies, #1 through #5. #1 is initially good for 4 copies, #2 is good for 3 copies, etc. Row C has good disks #6 through #15. Row D has good disks #16 through #25. Row E has good disks #26 through #30. The last row F has only one good disk, #31, for a grand total of 32 working disks, including the starting disk #0! As you descend down the table, each disk has one less permitted copy than the level above; the same is true as you move from left to right in the table.

It turns out that the number of disks, including the 0 disk is given by the equation Q=2^N, where N is the number of "permitted copies". The limitation of "5" copies is really not so limiting after all, when you understand what is going on. Of course to make maximum use of the limited disk, you must make all of the 31 permitted copies initially before any are lost through inadvertance, bad machines, spilled coffee, etc. You need not be as systematic as the chart implies. Just keep backing up from each disk until the "Disk contains protected files." message appears.

Genealogy of a Copy Limited Disk



To carry the research a bit further, I next identified the counter in the protected disk. It is byte C6 of sector 2 of track 0. Initially this contained 5, but after each backup cycle it decremented once, except after 1 it

changed to FF, which is apparently interpreted as zero copies remaining.

Further testing all of this, I used Super Utility to replace the FF with a 5 and confirmed that the disk was once again willing to make more working backups. Byte C6 is part of BOOT/SYS, so evidently this copy limit scheme is inherent in TRSDOS 6.X, and probably was there from the earliest version 6.0.

This system works similarly to the one in TRSDOS1.3 (see Lance's article in TRSTIMES V1N5P16). This counter is just in a different place, and does not show a zero when the copies are used up. By the way, TRSDOS normally has an FF at this byte, so I guess you could say that all copies of TRSDOS6.X are copy limited, but only for "protected" files, of which there are normally none.

The counter and protection scheme work in conjunction with a flag associated with the protected program(s). Other exploration has revealed that bit 4 of byte 1 of each directory entry serves as a flag to indicate the file is a copy protected file.

Simply resetting this bit to 0 via SuperUtility or equivalent will allow BACKUP to copy by files, including the file which originally had bit 4 set. There is yet more to be discovered, as clearing bit 4 does not suppress the message "disk contains protected files" which shows up during BACKUP. However, even though the message appears, the file WILL be copied, regardless of the counter in BOOT/SYS, and the program WILL execute.

As part of my search for the bit 4 flag, I perused Roy Soltoff's "The Programmer's guide to LDOS/TRSDOS Version 6" and located bit 4 as the likely candidate by carefully reading what RS had to say about the bytes and bits in the directory. Bit 4 stood out because the only explanation was "This bit is used internally by the system." A red flag if ever I saw one!

Some information also turned up in BACKUP/CMD. The copy limit can be turned off by PATCHing a piece of code. There is a relative jump which can be NOP'ed with 00 00, which will cause BACKUP to copy by sector as opposed to copy by file, and this will give you a verbatim copy regardless of the contents of the disk. In addition, the counter byte at C6 in the BOOT sector must also be edited.

The following patches will work for the DOS version as noted:

TRSDOS 6.2.0 and 6.2.1

PATCH BOOT/SYS.LSIDOS (D02,C6=00;F02,C6=FF)

PATCH BACKUP/CMD.UTILITY (D06,C3=0000;F06, C3=2807)

TRSDOS 6.3.0

PATCH BOOT/SYS.LSIDOS (D02,C6=00;F02,C6=FF)

PATCH BACKUP/CMD.UTILITY (D07,7B=0000;F07, 7B = 2807)

As a final bit of irony, the new utility included in TRSDOS 6.3, DISKCOPY/CMD, will do a verbatim copy of a disk, regardless of the presence of protected files, the counter, the bit 4 flag, or anything else I know of. SuperUtility can also properly copy the disk, except for the date/password structure under TRSDOS 6.3. It puts the old 6.2 structure in place, and you will have to use DATECONV to make the new disk kosher for 6.3.

Model I

INSIDE THE NEWDOS v.2.0. SYSTEM COMMAND

by Ben Mesander

I have long been puzzled by the NEWDOS/80 SYSTEM command. It is hard to use, and requires a reboot to make any changes to the system options. This is especially annoying for people like me, who have Aerocomp Double-Density disk controllers, because the BOOT command does not work, and the computer must be reset manually.

On page 2-46 of the NEWDOS manual was the clue that I needed: the system parameters are stored in the 3rd sector of the diskette. What I wanted to do was find where these options were stored in memory so I could write programs that used them, but first, I found out how they were stored on the disk.

I used SUPERZAP to print a copy of diskette sector 2 (remember model I sectors start with zero). Then I changed the system parameters and printed a copy of the sector again. The byte that changed was the byte I wanted.

I found that the options were not stored in a particularly logical order. I then used DISASSEM to disassemble SYS0/SYS, and followed the boot-up code near 4D3EH, and sure enough, it was easy to find where the starting code read in the third sector into a buffer at 4200H.

The program reads in the sector, and stores the options in various places. I have not documented all the relevant locations, but just the ones I have used in writing programs. It's really very easy to find the others.

Please note that when you alter these locations, it is a temporary change that lasts until the next reboot unless you also change the third sector.

If you are wondering why anybody would like to change these parameters, I have included a very simple sample utility, A/MAC, that demonstrates their usefulness. It is written with Microsoft's M80 assembler, but it could easily be assembled with any other assembler, as it uses no fancy features.

The program allows the user to change the default drive number for the DIR command and the default drive number to create files on when no explicit drive is specified in the filename (system options AN and AO). This is something like the CP/M or MSDOS commands that allow you to "log in" a disk drive. In addition, the program makes sure you do not attach to a drive that does not exist, and it will also display the drive you are currently "logged into".

Once you have assembled the program, you use it as follows:

A # where # is a valid drive number on your system to "Attach" to that drive.

A with no drive number to display the drive number you are currently "Attached" to.

SYSTEM command options and their RAM locations:

AL(1)	439FH	Maximum drive number on system
AL(2)	477AH	These 2 are different location with the same information
AN	43A0H	Default drive for the DIR comand.
AO	43A1H	Default drive to create files on.
AP	4049H	High memory value.
AX	4370H	ASCII code of highest character to send to printer
BI	4501H	Cursor character ASCII value.

Internal NEWDOS data structures:

INBUF	4318H	Location of dos ready kbd input buffer.
ENTRY	4403H	Entry addr of m/l program (after loading).

;A/MAC

```
; NEWDOS/80 V2 ATTACH UTILITY
; BEN MESANDER 7/7/88
;
; A/MAC allows one to change the default drive #
; for the system command options AN & AO.
; they are the default dir drive # and the default
; drive # to create files on - error checking is through
; the system option AL (max drive #)
;
;
;
                .Z80
; CONSTANT DEFINITIONS:
_SYSAL1 EQU     0439FH      ;one place that option
                            AL is stored
_SYSAN  EQU     043A0H      ;where option AN
                            is stored
_SYSAO  EQU     043A1H      ;where option AO
                            ;is stored
        ASEG
        ORG     5200H
;human-proof the input
        LD      A,(HL)      ;get cmd line parameter
```

```
            CP      0DH         ;see if no arguments
            JR      Z,DISPLAY   ;so dsply current drive
            CP      '0'         ;everybody has drive 0
            JR      C,ERROR     ;see if less than 0
            LD      B,A
            LD      A,(_SYSAL1)     ;get max driveno + 1
            ADD     A,'0'-1     ;ascii bias-1 offset
            CP      B           ;check for greater than
            JR      C,ERROR
GETNXT:     INC     HL          ;check for proper
                                cmd termination
            LD      A,(HL)
            CP      ' '
            JR      Z,GETNXT    ;skip whitespace
            CP      0DH         ;check for end-of-line
            JR      NZ,ERROR
;MAIN ROUTINE
            LD      A,B         ;get driveno back
            SUB     '0'         ;remove ascii bias
            LD      (_SYSAN),A  ;set default dir drive
            LD      (_SYSAO),A  ;set default create drive
            JP      402DH       ;And exit
;ERROR HANDLER
ERROR:      LD      A,20H       ;Illegal driveno error
            JP      4409H
;DISPLAY ROUTINE
DISPLAY:    LD      A,(_SYSAN)  ;get default dir drive
            ADD     A,30H       ;add in ascii bias
            LD      (DRIVENO),A ;store in message
            LD      HL,MESSGE   ;display
            CALL    4467H
            JP      402DH
MESSGE:     DEFM    'DRIVE: '
DRIVENO:    DEFS    1
            DEFB    0DH
            END     5200H
```

Note: The following changes must be made in order for A/MAC to work with EDTASM:

Remove the **.Z80** instruction

Change **_SYSAL1** to **SYSAL1**

Change **_SYSAN** to **SYSAN**

Change **_SYSAO** to **SYSAO**

Remove the **ASEG** instruction

Remove **all colons** at the end of labels.

Change **DISPLAY** to **DISPLY**

Change **DRIVENO** to **DRIVNO**

Make sure **all** references to the above labels are also changed in the **operand** field.

ITEMS OF INTEREST

FOR SALE

My computer room has become so overstocked with program packages in my library that I have to clear some out to make room for things like sleeping and such!

These programs are all original and come with the original disks and documentation unless noted.

- They are being offered on a "Make me your best offer basis" (sort of like a mail auction).
- Offers will be taken until February 28th, 1989.
- Only people who have made the best offer will be notified.

Send your offers to:

The File Cabinet's Garage Sale
P.O. Box 322
Van Nuys, Ca. 91408.

Program Packages on DISK for the Model 4:
CP/M Plus Operating System - CBASIC for CP/M - Money Decisions 1-5 (make offer on complete set) - Formation - Deskmate - Accounts Receivable - Query (disk #2 is a backup reference disk) - Designe - Target Planner Calc - Data Ace - Double Duty - Disk Scripsit (disk is backup reference disk) - TRSDOS Training Course - TK! Solver - TRS-80 C - TRS-80 Pascal - Videotex Plus - TRSDOS 6.2 Utilities SuperScripsit - SuperScripsit Dictionary - Scripsit Pro

Program Packages on DISK for the Model III:
The Home Accountant (Continental release) - The Home Accountant (Radio Shack release. Disks are back up reference disks) A.O.S. Utilities #1 - Agri-Calc Feeder Pig Module - Maxi Stat - Maxi Crass - PFS: Report - Blitzkrieg - Frogger - Computer Diplomacy - Planetfall - Spox - The Dean Vaughn Memory Course - Electric Webster Personal Income Tax Records Manager - Portfolio Data Manager - Domes of Kilgari - Trashman - Music Teacher - Kitchen Sink - Eureka - Geography Explorer: Mid East - Quikpro + II - RPM - Temple of the Sun - Voyage of The Valkyrie - Proof Edit - Superkeys - TRS-80 Compiler - Basic Network 4 Operating System - Visicalc - Profile III Plus - Alcor C - Corplan - Desktop Plan 80- Inventory Master - Data Master - Checkwriter 80 - Assembly Language Tutor - COBOL - Laser Blazer - Series-I Editor Assembler - Space Chase - Timetable - Home Budget - The Search For Elsoliado - Earthquake San Francisco 1906 - Spook House/Toxic Dumpsite - System Diagnostic

Program Packages on CASSETTE for the Model III:
TRS-80 Basic course - Geography Explorer: Europe - Math-Pak-3 - Defense Command/Stellar Escort - Meteor Mission 2/Cosmic Fighter - Conflict 2500 - Adventure #0 Special Sampler - Sea Dragon -

Adventure #3 Mission Impossible - Rear Guard -
Adventure #4 Voodoo Castle - Back 40 -
Adventure #6 Strange Odyssey - Planetoids -
Adventure #8 Pyramid of Doom - Project Omega -
Adventure #9 Ghost Town - Escape from Traam -
Adventure #11 Savage Island Part Two -
Adventure #12 Golden Voyage - Showdown -
The Time Dungeon-U.S - The Time Dungeon-World -
Plotting Graphs for Line Printer - Business Package I -
Plotting Graphs for Video Display -
Mostly Basic-Interfacing/ Scientific -
Mostly Basic-Household - Mostly Basic-Educational

This is just a small sample of what I have to offer. I have close to 100 TRS-80 oriented books that I will be offering soon. If you would like more information, feel free to drop me a line and I will let you know what I have to offer.

Color Computer II system for sale:

HARDWARE

Radio Shack Color Computer II - RS Power strip
FD-501 Disk Drive Controller - 2 Floppy Disk Drives -
Amdek Color 300 Monitor - Multi-Pak Interface
Deluxe RS-232 Program Pak - TRS-80 X-Pad -
Modem 1B DC-2212 Modem - Deluxe Joy Sticks -
CCR-81 Data Cassette Recorder
Botek CCP-2 Serial to Parallel

SOFTWARE

Cash Budget Management - Color Disk EDTASM -
OS-9 1.01.00 - OS-9 C Compiler - OS-9 TRS-Copy -
Coco Max - 10 disks of Public Domain Software including MIKEYTERM

ROM PACKS

Androne - Canyon Climber - Dungeons of Daggorath
Galactic Attack - Spidercide - EDTASM +

MISCELLANEOUS

Program Pak File - Coco II Service Manual -
Assembly Language Programming
All cables and covers are included in this package.

THIS SYSTEM IS FOR SALE AS A PACKAGE DEAL ONLY! Items will NOT be sold individually.

Asking price is $400 or best offer. Price does not include shipping.

CONTACT:

Tim Sewell
P.O. Box 322
Van Nuys, Ca. 91401
(818) 766-7982 (voice)
5pm to 9pm Pacific time.

SOFTWARE

PROGRAMS AVAILABLE FROM HINRICHS $OFTWARE:

The Basic Word Processor for Model I, III & 4/III
16K, 32K, 48K - tape or disk
Full featured, with wrap-around, font changes, global search/replace, form letters, etc.
Specify your system.
$10.00 - tape - $12.00 - disk
$15.00 compiled (48K M1 disk)

Multivariable Regression Analysis
Model I, III & 4/III
This is a 10-program disk-based package. It handles up to 9999 observationsof up to 57 variables each. Reverse stepwise operation on up to 47 variables. Unlike some other such programs being sold, it is **accurate**. It will also print any outputs on a printer.
$25.00
(simpler tape version - $10.00)

Fourier Analysis and Synthesis
Determines cyclic variation in data, and will use significant cycles to forecast future data. Save data on tape or disk. Graphic & tabular output, and print on printer. $10.00

FontPrint
Print my WP text files in SCRIPT, or in other fonts on Epson or DMP-type printers. FontPrint also supports block graphics and proportional spacing.
Model I - 48K disk only. $10.00

Year's Data
Plot year's data of several variables on 8.5x11 sheet. Disk, MX-80, other printer with linespace control needed. Generates disk data file useable by regression.
$10.00

Mail List
Prints address labels or listing. Select the output by code, zip, name or number. Alphabetize.
$10.00

Many other programs available.

CONTACT:

Hinrichs $oftware
2116 S. E. 377th Ave.
Washougal, WA. 98671-9732
(206) 835-2983

ATTENTION ALL TRSDOS 1.3. USERS (Models III, 4, 4P & 4D)
DBSIDE ALLOWS TRSDOS 1.3. TO USE DOUBLE SIDED 5-1/4" DRIVES

Upgrades 1.3. to use DS/DD drives, by treating the "other" side as a completely new independent drive.
Up to 8 drives are now supported! it's like adding 4 new SS/DD drives!
SO GO OUT AND UPGRADE TO DOUBLE SIDED DRIVES! IT'S GREAT!

A new library command, "SWAP", allows disk drive numbers to be changed. Eg: After SWAP (1,2) is issued, the command DIR :1 <CR> would actually display drive two's directory! Even the SYSTEM DRIVE can be swapped with any other drive, allowing a data disk, or no disk at all, to reside in Drive :0

Many of the system overlay programs have been improved. A few of the improvements include restoring killed files, re-boot using "BOOT", debug ANY memory area/system files. BOOT/SYS, all system overlays, and DIR/SYS have been added to the directory. You can process/modify /examine any of these files, by using DEBUG, or even BASIC! Newly created files will now use an improved date format, being MM/DD/YY, instead of that old meaningless MM/YY format. And for you basic programmers, you can now execute DOS commands from within BASIC, and be returned to BASIC with all variables intact, even if DOS errors occur!

Send **$29.95** US funds, postage paid to: **GRL SOFTWARE, Suite 209, 1051 KLO Road, Kelowna, British Columbia, Canada, V1Y-4X6, Attn: DBSIDE**

ATTENTION TRSDOS 1.3. USERS!

GRL SOFTWARE PROUDLY ANNOUNCES "SYSTEM 1.5."
THE MOST COMPREHENSIVE 1.3. UPGRADE EVER OFFERED!

MORE POWER!!! MORE PUNCH!!! MORE SPEED!!!

"SYSTEM 1.5." UPRADES TRSDOS 1.3. TO USE THESE COMMANDS **

ALIVE = On/Off	Displays a moving graphic ALIVE monitor
TRACE = On/Off	Monitors the current STACK POINTER address
DATE = Yes/No	Enables/Disables the boot up Date prompt
TIME = Yes/No	Enables/Disables the boot up Time prompt
CURSOR = 'XX'	Defines the boot up cursor character
BLINK = Yes/No	Defines the boot up cursor status
CAPS = On/Off	Defines the boot up condition of CAPS lock
CONFIG = Yes/No	Save current configs/loads file during boot
DEVICE = Pr/Do	Outputs current active driver locations to?
TYPE (parms)	Adds a type ahead buffer with a "key click"
MEMORY = On/Off	Display BASIC free mem at all times
FILTER (parms)	Enables complete keyboard byte translation
FILTER (parms)	Enables complete printer byte translation
MACRO (parms)	Adds user defined MACRO KEY capability
SPOOL (parms)	*Enables a memory and/or disk LP spooler
SYSRES XX/All	*Select and store DOS overlays into memory
ADDLF = Yes/No	Adds a line feed after a 0DH is printed
IGLF = Yes/No	Ignores line feeds in those "OTHER" files
HARD = Yes/No	Sends 0CH to PR direct, instead of LF/LF/LF
PAGE = 'XX'	Starts page numbering on pg XX (DOS/BASIC)
FORCE = Yes/No	Forces new page after XX lines printed
UNKILL (parm)	Recovers/restores previously killed files
LIB/DVR	Enables DOS LIBRARY commands to be used by a BASIC PROGRAM without data/variable loss!

All options are installed using a new "SYSTEM (parm,parm)" command, similar to what LDOS or TRSDOS 6 users have had for years. For example, the type ahead driver is enabled, and the key "click" is defined, by using the command:

SYSTEM (TYPE = ON, TONE = 'xx', LENGTH = 'xx', PORT = 'xx') <CR>

Enter graphics/special chars/comp codes from the keyboard, reconfigure the keyboard, speed up paging, add pg numbers, increase Disk I/O speed, link memory spooling to a disk, etc. So many added benefits, we just can't describe them all. The manual takes beginners step by step through each command, and a 'PRO" chapter documents drivers & tables.

* Model 4/4P/4D, (64/128K), except a 64K 4P, may use bank memory (16-32K for SPOOL/SYSRES options.
** Most options also support additional parameters. Eg: Spool parameters include options to alter or define ON/OFF status, buffer SIZE, select HI or BANK data storage, RESET buffer, DISABLE, ENABLE, DISK link, DISK SIZE, etc.

Send **$39.95** US funds, postage paid to: **GRL SOFTWARE, Suite 209, 1051KLO Road, KELOWNA, BRITISH COLUMBIA, CANADA, V1Y-4X6. Attn: "SYSTEM 1.5."**

CLOSE #1

Whew!!! Just met the deadline. This is the latest TRSTimes has ever gone to the print shop. We made it though, and it sure feels good. Do hope you like the new, sharper, non dot-matrix look.

Again, we could not hold the size to the scheduled 32 pages. Due to Gary Campbell's first installment of the TRSDOS 1.3. column we had to go the extra pages. This material simply could not be left out. Yes, it is a very long listing, but 'minor miracles' usually do take that much code. And, indeed, a minor miracle it is. You were always able to transfer files **FROM** TRSDOS 1.3. **TO** the other DOS'es via various built-in utilities, but never, never, never could you transfer files **FROM** the other DOS'es **TO** TRSDOS 1.3. Now you can. This is a 'first'. A big 'thank you' to Gary for sharing this outstanding utility with us.

(note to TRSDOS 1.3. users: Be sure to read Gary's ad for TRSDOS 1.4 & 1.5. on the previous page - there you'll find additional miracles at a low, low price.)

Another reason was my ASSEMBLY 101. Somehow, it ended up being more than twice the size I had originally anticipated. However, being a tutorial for beginners, I felt it important to take it one simple step at a time - and the length be darned. Future installments will be shorter, and the pace will be stepped up somewhat.

If you have ever played ZORK, or any of the other Infocom adventure games, we know you're hooked. John Fowler tells us how we can convert some of the games that were released for the IBM only, and make them WORK on the TRS-80.

Roy Beck researched and solved the Model 4 'backup limited diskette' problem. You can now make as many copies as you like.

The PD EXPRESS discusses two of the better communications programs. If Tim likes them, they are good.

Fred Blechman, who has written articles for almost every computer magazine ever published, graces us with a look at the world of disk duplication & conversion.

Fer Cornet from Holland gives us a fine cross reference utility for Model 4 Visicalc. This is good stuff. Don't miss it.

Model I is covered by Ben Mesander, a true 'old school' hacker of the finest order. He explores NEWDOS' SYSTEM command and includes a handy little machine language utility.

Hunting for Buried Treasure is my usual ramblings about TRSDOS 6.2. This time we focus on 6.2.0 and the missing double-sided format prompt.

Finally, I did not write LISTER. It is an obscure public domain program that is just too good to be ignored. It should work on all models. Try it, I'm sure you'll like it!

The March issue, hopefully, will have REAL page numbers. I will certainly do my best to find out what happened to them. Until then...STAY TUNED!!