Issue #26/27 THE STAUNCH 8/89'er Sep-Dec 1991

THE ADDRESS BUS

PORT TO PORTAL -- Editorial

Preparing this issue has been quite interesting. I think you'll find reading it to be intriguing as well. From my point of view, the fascinating feature was preparing and integrating the graphic elements that appear here. Indeed, I've got more graphics in this one issue than have appeared all year long! Not all, of course, are like the paste-ins for David Shaw's first installment of his HDOS multitasking series or the seasonal labels done by Paul Flexman with Tiny Pascal that you'll see later. Some are simply computer-printed line drawings or tables. But for me, deciding what and **how** to prepare the material furnished by this issue's writers was the challenge.

However, one writer you won't find on this issue's pages--at least in a formal sense--is me. The compressed schedule required to bring this issue to you **before** the end of the year meant that I had to set aside preparation of the second installment in the hardware troubleshoot series. In fact, my schedule looks just as busy through about the end of January. So, even though it'll be slightly out of sequence, you'll find Dan Jerome's traversal of mods and repair of the '89's power supply here instead of my Part two.

Another thing I must bring to your attention is one item in the Software Listing. Trio Company, mentioned in the last issue (p. 11) as a distributor of WordStar 4.0, had a wholesale, stock-reduction sale on that product aimed specifically at CP/M dealers, user groups, and (amazingly) newsletters. Hence, I picked up seven copies on the cheap. And I'm passing the savings along to you. Moreover, I'm enhancing the package with a set of installable patches to bring up the H-19/89's function keys and cursor keypad. These packages **won't** last and I will take orders **only** on a first-come, first-served basis. So if you were thinking about getting WS 4.0 but didn't like the price, now's the time to do it. If there's a continuing interest, I'll stock WS 4.0, but the price will be higher!

Finally, you probably don't need it, but I should remind you that this issue may well be your **last**! Compare the issue number above with the one on the envelope. If both equal "27", it's time to mail a check. And don't forget that the rate goes up to $15/yr. as of New Years Day. So get your check to me **before** you get "socked" with the rate rise!

Kirk L. Thompson

THE EIGHT-BIT R/W -- Letters

Commentary on the Last Issue. [From Lee Hart, 323 West 19th St., Holland, MI 49423; more from this letter appears in the THE LINKAGE LOADER] "Congratulations on obtaining a permanent position. Being a temporary is a very insecure feeling. Like comedian Steven Wright says, 'You know how it feels to lean back in a chair until you almost fall, but then you catch yourself, almost? Well, it feels like that all the time'...

"TO TOM SLAVIK CMOS chips are LESS sensitive to heat than -LS. They normally work from -40 to +85 degrees C instead of 0 to 70 degrees C. Your video fade-out problem is something else. I'd look for an IC that's making a bad connection in its socket.

"On internal heating; the hotter the part, the higher the failure rate. Any reliability text will tell you that every 10 degrees C cuts the life of a semiconductor in half. Therefore the goal of pointing the fan downward is to reduce the temperature of the hottest parts, which are those on the power supply heatsink. If this is not done, those parts will run over 70 degrees C, and are the ones that will fail first. ICs on the logic board do run a little hotter with the fan blowing down, but still don't even reach 40 degrees C (at which their life expectancy is over 50,000 hours).

"To put this in perspective, how long would YOU last at 70 degrees C (158 degrees F) versus 40 degrees C (104 degrees F)?

"ON A PORTABLE H-89 I've received a number of positive comments on a portable H-89. The basic goal seems to be a machine that can run existing H-89 software (CP/M and HDOS) with no modification. Here is a summary of the features people seem to want.

1. LCD screen
   - 80x25 characters
   - 640 x 200 dot graphics (8x8 dots per character)
   - optional backlight
   - same character set as H-19
   - 25th line like H-19
   - optional dot-addressable graphics
   - optional downloadable fonts
2. CPU
   - CMOS Z80, 4 MHz
3. Memory
   - static CMOS, battery backed up
   - 128K minimum, expandable to 1 meg (at $15 per 128K)
   - memory beyond 64K is configured as RAM disk
4. I/O
   - 2 serial ports, same as H-89

[Continued on p. 4]

SOFTWARE LISTING

>---<

General Software Catalog

A catalog of **Staunch** software is available. Initially prepared by Ralph Money, the disk files include listings for **both** HDOS and CP/M. The files are "squeezed" to conserve disk space and an unsqueezer is provided to recover them. This catalog requires only one (1) disk in any format. See the "Placing an Order" section, below, for information on formats.

Source Code in this Issue

If this issue includes any source code, be it BASIC, Pascal, C, assembler, or whatever, you may obtain it **at no charge**! Merely send a **formatted** disk with a **postage-prepaid** mailer and I'll transfer it for you. Please clearly indicate the format you are supplying. See below for supported formats.

FOR BOTH CP/M AND HDOS

Dual-Format Disks

(Source disk provided by Charles Horn)

This offering includes **both** standard hard- and single-sided, single-density soft-sector dual-format (CP/M and HDOS) disks. The hard-sector source disk was prepared by Charles Horn and discussed in the last issue (p. 1). The soft-sector conversion was performed with ZUG's MAN37 utility on its 885-1217 utility disk. The original hard-sector disk was prepared with HDOS 1.6. This approach **maximizes** usable storage for **both** CP/M and HDOS because this old version of Heath's proprietary system placed the directory files at the center of the disk. Capacity for **either** CP/M or HDOS on both disks is 45K. Charles Horn has "tweeked" the HDOS directory files to cut their size to a minimum. At Pete Shkabara's suggestion in his letter in this issue, I have patched the CP/M directory to put the CP/M files protecting the HDOS area in User 31. The hard-sector version is read/write-compatible with HDOS 2.0/3.0x and CP/M 2.2.03/04; it should even be compatible with CP/M 2.2.02, though I have not tested that. The soft-sector version is, with three exceptions, read/write-compatible with HDOS (2.0/3.0x) and CP/M (2.2.03/04). It is **not** compatible with 2.2.02 since that version never supported soft-sector; the same applies to HDOS 1.6. It is also not compatible with Extended Technology's SUPER37 soft-sector driver for HDOS 2.0; see my cautionary note at the end of this issue. When ordering, please indicate whether you wish the hard- or soft-sector version or both.

FOR CP/M ONLY

WordStar 4.0

(Copyright WordStar International)
(Obtained from Trio Company)

I've obtained **seven** (7) copies of WordStar 4.0, the last version for CP/M-80, at stock-reduction, wholesale prices. This latest edition of the classic word processor includes everything you've learned to love (or hate) in earlier editions. However, WordStar (formerly MicroPro) International has enhanced it with some **very** nice features. These include: printing multiple columns per page, boilerplating without the need for a separate "mailmerge" package, The WORD Plus spelling checker with automatic correction and hyphenation, macros, a built-in calculator with the standard (+-*/) operators, indexing, and extensive printer and terminal support. The list of supported printers numbers over 100 and includes all types from TTYs through lasers; you may also customize your own driver. The terminals supported include the H-19/89, but that option does **not** bring up the function keys or keypad. WS 4.0 supports ZCPR's named directories and user numbers of 0 through 31, but directory names are not displayed; the word processor may also not work if your ZCPR system requires a lot of memory.

As enhancements, I will move the files from the original Osborne 1 disks to your choice of media and include a set of patches to implement function keys and cursor keypad. If you already have an earlier version with installed patches, this package **includes** a utility to move most of those patches to 4.0. However, to use this word processor you need 64K of RAM with maximum possible TPA and high-capacity media. A minimal, no-frills system requires 89K of disk space, leaving no room for even essential CP/M system files like PIP! You will need either a harddrive or a minimum of two high-capacity floppy drives. The latter may be any soft-sector configuration or double-sided and/or 96-tpi hard-sector.

The documentation that comes with the package includes WordStar's 400-page spiral-bound manual, a list of supported printers and their features, a quick-reference card, a booklet describing changes from earlier versions, warranty registration card, and on-disk files describing my add-on patches and how to install them. The (usual) limited warranty for the core package is provided by Trio Company (Cheektowaga, NY).

The cost of this package is $60, shipped U.P.S. in the contiguous 48 states, parcel post elsewhere. Shipping is included in the former price; Hawaii, Alaska, Puerto Rico, and Canada, please add $5 for postage; overseas, please add $15. I will ship on a first-come, first-served basis. If there is sufficient interest, I will continue providing this package beyond the seven copies I presently have. However, the cost will be higher (roughly $100)!

Public Domain Utilities

- Primarily for the Programmer
(Selected by Peter Shkabara)
(Provided by Terry Hall)

Here's a collection of programs intended mainly for programmers. Some of this material has appeared elsewhere in **Staunch**'s library, some is new. The files included are:

COMPARE	- compares two files.
CP	- a fast copy program, good for long files or long lists of files; both CP/M- and MSDOS-style command-lines acceptable, but PIP's switches are not supported.
CREATE	- creates a library or archive file for use with LU or LBRDSK.
D	- directory display program, listing files

	in alphabetical order with file sizes on an ASCII terminal.
DU	- CP/M User's Group Disk Utility for displaying and modifying disk sectors.
FIND	- search for ASCII string in a file; filenames may be wildcards.
FINDBAD	- CP/M User's Group program that locates bad data blocks on the disk and creates a file with those blocks in it.
LBRDSK	- allows access to library files as if they were a disk drive.
LDIR	- shows contents of a library.
NULU	- the ubiquitous library utility which allows saving many files under one directory entry.
NULUTERM	- an assembler patch overlay for NULU; requires MLOAD.COM.
MAKE	- utility for directory manipulation including unerasing and changing user areas; has built-in help instructions.
NEAT	- assembler source code formatter.
SD	- another sorted directory program; requires the H-19/89.
SETDRU	- a very useful program, particularly on a harddrive, that allows use of the ZCPR default drive and USET function with programs that have overlays.
UNSETDRU	- removes SETDRU patches.
SWEEP	- this is NSWP207 which allows file transfers, squeezing and many other operations.
SZAP	- a file and disk dump utility, similar to Software Toolworks SUPERZAP, but with different features.
T	- strips control characters and parity bits from a document, such as WordStar files.
UNLOAD	- reverse of LOAD; makes a HEX file from a COM.
ZZSOURCS	- a program disassembler.

This package occupies 176K of disk-space.

FOR HDOS ONLY

TINY PASCAL Version 4.1
(Enhanced by Mark Kroska)
(From OMAHUG's Library)
(With supplemental materials)

This package is constructed around an enhancement of Tiny Pascal that is also offered by ZUG as its 885-1086. However, a large assortment of supplemental material from OMAHUG's library and other sources is provided that go beyond ZUG's release; this additional material is described below. Tiny Pascal is a subset of the Pascal language created by Niklaus Wirth. Like the parent language, programs are compiled from your source code. That means that they are much, much faster executing than those written for interpreted BASIC and may even be faster than other compiled languages, such as FORTRAN. Pascal emphasizes the writing of programs in a series of small modules and was originally conceived by Wirth as a first computer language to teach good programming practice.

Tiny Pascal can be an excellent low-cost introduction to the Pascal language. The compiler incorporates most of the commands from the parent used in the flow-decision process and includes extensions to allow disk read and write of both sequential and random files (the latter not standard for Pascal) in addition to direct line-printer output. Tiny Pascal uses split-octal and decimal notation. In many applications the additional speed obtained by running the compiled program directly instead of interpreted p-code (as with Lucidata or UCSD Pascal) is worth the effort to learn the language. For some applications, such as word processing utilities, the advantages of direct .ABS execution, minimal program development time, and low run-time overhead outweigh the limitations of the subset. The language uses a two-pass compile process, directly producing an .ABS file that will run from the HDOS command prompt.

Limitations include primitive mathematical operations, storage as integer values only, limited file handling, and very limited string operations. Like most Pascals, the compiler tends to appear unforgiving of the slightest syntax error and usually does not provide precise indications as to where the error may be. Even successful compilation does not necessarily guarantee a bug-free program and the provision for tracing the program flow or providing break-points does not exist.

This package includes the two-module compiler, a configuring utility with both object (.ABS) and source (.TPS) code, Mark Kroska's original documentation, Mark's discussion of handling multi-dimensional arrays in Tiny Pascal, a huge (323 sector) file by Frank Christel and Frank Adams describing in greater detail the elements of the language, and a large selection of utilities (most written in the language) and example Tiny Pascal source code gleaned from OMAHUG's library. Among the executable utilities are: DIRSIZE (to reduce the size of DIRECT.SYS on hard-sector disks), STATX and STATS (for displaying system statistics), DUMP (a disk dump utility), PTSET (to set codes on a Paper Tiger printer), and DIRMAP (displays the linkages of files in DIRECT.SYS and GRT.SYS). Another program package in **Staunch**'s collection, UTILITY ONE (by Frank Adams and announced in issue #9), was written entirely in Tiny Pascal. This compiler package requires 1168 sectors on-disk.

Placing an Order

With the **exception** of WordStar 4.0, your cost for this software depends on what you supply:

Formatted disk(s) and self-addressed, stamped return mailer	$2.00 per disk
Formatted disk(s) without mailer	$4.00 per disk
No disk(s) or mailer	$6.00 per disk

Disk formats available are standard (SS/SD) or double-sided (DS/SD), 48-tpi hard-sector and single- or double-sided, 48- or 96-tpi soft-sector for both HDOS and CP/M. (**Staunch** now supports 96-tpi soft-sector if you provide a formatted disk; if this is a problem, let me know.) Please clearly indicate the format you are supplying or require. If you desire DS hard- or any soft-sector format, I will pack **multiple** items onto **one** disk. I will **not** subdivide a disk. Send mailorders to:

Kirk L Thompson / **The Staunch 8/89'er** /
P.O. Box 548 / West Branch, IA 52358

>--<

THE EIGHT-BIT R/W [Continued from p. 1]

- 1 parallel port, same as Z89-11
- real time clock/calendar

5. Floppy disks
   - one H-89 I/O expansion slot
   - can plug in H-17 or H-37 controller board and disk drive
6. SCSI port
   - NCR 83C50 chip, talks to hard disks, etc.
   - similar, but not identical to H-47/H-67 I/O board
7. Console
   - separate Z80 to handle LCD screen and keyboard (like H-89)
   - PC clone keyboard
8. Physical
   - one PC board, approximately 9" x 11"
   - LCD screen and keyboard on separate boards
   - battery life, approx 18 hours with 5 nicad C cells (less disk drives)
   - weight approximately 4 lbs
   - cost would be about $250 for a complete kit
     PC board about $1000 to lay out and tool, then $25 apiece
     Parts cost (128K RAM) about $100.
     Supertwist LCD displays are $30 w/o backlight, $60 with
     PC keyboards are $30 and up

Basically, this is the same basic circuit as an H-89, updated with CMOS logic and static RAM in place of dynamic RAM. Some unnecessary features like single-step, on-board serial port, and all but one expansion slot would be left out. The Z89-11 3-port board and H-47/67 SCSI interface would be built-in.

"The H-89's 2-CPU design would be retained. LCD screens require that you plot every character dot-by-dot, which is very slow. To get the equivalent of 9600 baud performance, we need an expensive LCD controller, a very fast main CPU, or a separate CPU dedicated to the LCD. The latter is the approach chosen here. It's easier to model the H-19 terminal while remaining compatible with existing H-89 software. The TLB is not a separate board, but just a few more chips on the main board, and communicates with the main CPU via a parallel port.

"The design is optimized for low power, low cost, easy construction, and to use standard parts. Note that it is half the weight of most DOS portables, and the batteries last 5-10 times longer.

"It's certainly possible to use a Z180 CPU, 8 MHz clock speeds, etc. But it will become a power hog and get expensive fast. You might as well buy a PC clone. Also, this forces you into surface mount ICs, which can't be used by any normal hobbyist.

"Remember, this is a 4 MHz machine with a big RAM disk. Most 8-bit software is disk-bound, not CPU-bound. The nonvolatile RAM disk will make this machine much faster than any normal H-89, even without fire-breathing clock speeds.

"The design of such a project is too large for any one individual to tackle. If enough people are interested, we could divide up the tasks as the available talent allows. I encourage your readers to contact me with comments and suggestions. Obviously, if I don't hear from anyone, the idea dies!

"PUBLIC DOMAIN CP/M Getting CP/M for an H-89 is no problem. I have them in stock, brand new, for $25 for the complete Heath 2.204 package (all disks and manuals). Specify disk format.

"I don't quite understand Mr. Gilmore's response. He says that Heath can't release CP/M because it was licensed from DRI, and because they've lost the source code anyway.

"CP/M is a whole collection of programs, with many authors. Obviously, Heath cannot release the programs that DRI wrote; the CCP, BDOS, MOVCPMxx, PIP, STAT, DDT, etc. But Heath CAN release the programs that THEY wrote; FORMAT, CONFIGUR, SETLP, BIOS, MAKEBIOS, and SETUP.

"I've found that Heath responds with 'no' to any question. So try the negative option. Ask Mr. Gilmore if Heath would object to distributing the FORMAT, CONFIGUR, SETLP, BIOS, MAKEBIOS, SETUP, and boot loader programs (without BDOS and CCP) in the public domain, exactly as they appear on a Heath CP/M distribution disk. They are all pure Heath property. We don't need source code for any of these except the BIOS, and that's already provided.

"These programs let you buy CP/M for any computer, and reconfigure it to run on an H-89. CP/M is available for obsolete computers like the Osborne and Kaypro for $10 or less. It came free with these machines, so sellers perceive it to have zero value.

"P.S. Mac Heath [a custom machine Lee is building for Mark Hunt that runs CP/M, HDOS, PC, and Macintosh software from an H-89 case] is ... the name of the ghoulish murderer in the Three Penny Opera, made famous in the song 'Mack the Knife'. The realization that I had created a monster prompted the following bit of whimsey.

'The Monster Mac'
mutilated by Lee Hart
(sung to the tune of 'The Monster Mash')

I was working in the lab, late one night,
when my eyes beheld an awesome sight.
I flipped on my Heath to process a byte,
but when it warmed up, to my delight...

(Refrain:) (It's now a Mac)
It's now a monster Mac.
(A monster Mac)
Must be the ultimate hack.
(The greatest hack)
Apple's sure to attack,
(The monster Mac)
Because the saving's a fact!

Once it was a Heathkit, square and gray,
but my trusty tools took its mind away.
In its place I installed a Mac motherboard
from a flea market bargain that Igor scored.

(refrain)

Instead of that "beep" that I've heard so long
It now powers up with a Macintosh 'bong'.
Its great glowing eye is twelve inches, not nine,
And that cavernous case makes expansion divine.

(refrain)

Now the graphics and fonts have gone to my head.

I put Mickey Mouse pictures in my letterhead.
And its voice synthesis is the joy of my life;
My computer talks back, like my kids and my wife.

(refrain)

Now the mouse and the games are all such fun,
But at last comes the time to get real work done.
Then I have to go back to my old CP/M,
which I run with the Mac's emulation program!

(refrain)

More on WS Customizing. [From A.E. Thornton, Kirkland, WA] "...In regard to patching WordStar in general I can highly recommend the following book which I obtained in one of our local 'Half Price Books' stores. (Half Price Books sells second-hand or 'pre-owned' books, and has been a never-ending source of information on CP/M, dBASE II, MBASIC and you name it!)

"The recommended book:

The New WordStar Customizing Guide by Stuart E. Bonney, published in 1988 by Wordware Publishing, Inc.

The subtitle to the book is: "A Complete Guide to Customizing WordStar for the IBM and Compatibles, For All Versions Including Release 4.0." Don't be put off by that subtitle, it contains Appendix C which is the supplement for CP/M-80. The [earlier] versions of WS covered are 3.0 & 3.3 and in any event a large portion of the information given throughout is applicable to **all** versions...." [Thanks for the reference. This appears to be an update to the book Joe Mendez recommends in his WS patch article in issue #24. Used bookstores are, indeed, a good source for out-of-print titles. Others are many of the mailorder booksellers. If you're not sure exactly what to look for, check the "CP/M" subject listing in **Books in Print** at your favorite bookstore. Many of those titles are now actually out-of-print, but are still ones to keep your eye out for. -Ed.]

Request for Direction. [From Peter Shkabara, P.O. Box 1987, Blythe, CA 92226] "As you might have noticed, my response time is delayed and sporadic. Seems that there is so much to fill in the apparent 'free time' here...

"CP/M information **is** scarce. Although CP/M is still in fairly wide use worldwide, its popularity in the US has faded severely. If your publishing enterprise is economically justified, you may indeed seek to expand your horizons to provide some support to CP/M users in general. I may suggest providing reviews of generic public domain software. There is so much of it out there, that without a reduced list of recommended titles, a newcomer may be thoroughly overwhelmed. For that matter, even experienced users may find it difficult to go through. There is some info to that end that I will be sending to you...

"At this point, I am a bit perplexed as to which direction to take in further submissions to you. Could you analyze your goals and directions and help guide me towards what you wish to see? There are ideas on articles relating to the MS-DOS transition to a CP/M environment. All CP/M users will have to face this step at some point as their machines reach the point of unprofitable repair state. Opinions on the state of CP/M or views back on its history are some other options. Or, I could continue with the theme of how to program and tutorials of what goes into the soul of a CP/M machine!

"In reading your issue #25, I found it curious that 'the replacement of a furnace...; that article will appear next time.' I was not aware that you have expanded the 89'er to include furnace repair articles! On a serious note, the dual-format disks info from Charles Horn states that dummy CP/M files are hidden away in User 15. User 15 is **not** a hidden area from CP/M users. It is just not a common area for non-Z-System users. CP/M does include the ability to specify user areas up to 31. CP/M has no access to such areas, but such files would not be bothered by CP/M either. Z-System **can** access User 31, but not conveniently without utilities. This would seem to be a better place to stick the dummy files." [Thanks for your comments, Pete. I think **Staunch** readers would find parts of my response to your letter of interest, so I excerpt it here:

"...[T]hanks for your comments about broadening **Staunch**'s support for CP/M. Reviews of p.d. software is a good idea and one I'll present to the readership by excerpting your letter in the coming issue...

"Thanks[, too,] for your interest in continuing to submit material to **Staunch.** I am certainly more than interested in publishing same. You ask for directions to guide future articles by suggesting the following as potential topics:

Transition from CP/M to MSDOS environments
The current state of CP/M
Retrospectives
How to program under CP/M
Tutorials on the 'soul' of the CP/M machine

Actually, I like **all** of your suggestions! And the first, without intending any disloyalty toward the '89, HDOS, or CP/M, is something which I specifically encourage you to discuss. A number of (former) subscribers have already made the transition, sometimes (based on their correspondence to me) with a great deal of trepidation, prompted by catastrophic failure of their 8-bit hardware, or because the tasks they needed done couldn't be handled by the dear, old '89. Hence, I think some discussion of the transition to current technology is worth pursuing simply because many subscribers will, indeed, face it at some point as you say. As for the rest of your suggestions, I've been pleased with your past contributions and would actually like to see a **mix** of topics. In that regard, I'm giving you something like 'carte blanche'. In a way, I realize that this makes selecting subject matter somewhat more difficult for you!

"As for 'mini-'series on specific topics, that would also be at your discretion. Two- or three-parters on certain subjects, or even longer ones (on assembly language, for instance), are no problem as far as I'm concerned...

"Finally, thanks for your cautionary note, regarding Charles Horn's dual-format disk, on user areas for dummy CP/M files....."

I encourage you readers to go through the p.d. software you have, select the ones most useful to you, and write reviews of them. Remember that I **pay** for articles in excess of 1,000 words, so you can even make some money this way! (In today's economy, that's not a bad idea!) And I'd also like you to see my "author's guide," so call or write for one. It describes how I'd like your contribution to be formatted and the constraints on article size I generally apply. Further, write to Pete or me and tell us what **you'd** like to see in Pete's column. His expertise is an important resource both to you and to this newsletter. -Ed.]

SCAN Patch. [From William S. Derby, P.O. Box 2041, Livermore, CA 94550; readers should know that I had to send Bill his copy of #25 **twice**--the first time, the Postal Service returned it to me for "Addressee Unknown"!] "Thanks for sending recent **Staunch** #25 (twice). My box is still paid for; I will call their mistake to the attention of my local post office. You can use my home address for any **Staunch** correspondence to avoid confusion in case the post office continues to mess up. I prefer to use the P.O. box address [given above] for orders and any correspondence from users of my CP/M programs.

"As always I enjoyed reading **Staunch** #25, especially Hank's 'This 'N' That' discussing his experiences with CLE and KEYMAP. I belong to the ranks of those applauding you for keeping the Heath 8-bit spirit alive...

"I want to give you a simple patch to correct a minor problem in the SCAN program from my Enhanced Utilities. Without the patch the program fails to find an ASCII symbol (with the S command) when it occurs at the end of a line. The S command finds ASCII symbols delimited by spaces or other non-alphanumeric characters. The patch can be made as follows[; enter the boldfaced material **and** periods]:

```
A>DDT SCAN.COM
DDT VERS 2.2
NEXT PC
1100 0100
-SCB5
0CB5 0B FC
0CB6 0D 10
0CB7 FE .
.
.
-SCBA
0CBA 0B FC
0CBB 0D 10
0CBC FE .
.
.
-S10FC
10FC 00 12
10FD 00 C3
10FE 00 0B
10FF 00 0D
1100 xx .
-^C
A>SAVE 16 SCAN.COM
```

This changes the SCAN.COM checksum from 36E5/F965 to 73B8/87CC.

"It will soon be three years since I last updated my Enhanced CP/M Utility Programs. Since their new-user base has fallen to zero, I am inclined to follow Pete's example and release them to your **Staunch** Software Listings. I would need to collect the programs and documentation, probably in a LBR file with a (very) modest suggestion of a shareware contribution. Let me know if you are interested in offering the programs to **Staunch** readers." [Thanks for the patch, Bill. And, of course I would be interested in distributing your utilities. -Ed.]

Sun and Snow!. [A letter from Corky Kirk, Hilo, HI, to Hank Lotz, Pittsburgh, PA] "Believe it or not, I was halfway thru your article 'Square One for Computerphiles, Part 3 - H-89 Terminal Configuration' when I realized it looked familiar and your correspondent was me!...

"Anyway, finished the article & will now go after the H-89 and [set] TLB, S402 #3 from the left **down.** (Bet it's up!) Thanks again for all the good dope.

"Have another problem w/H-90. Was Down Under for 3 wks & when I got back to this high humidity and warm climate of Hilo, Hawaii (65-75% ave./75-85 degrees ave.), I found my terminal CRT sez 'H:' on the top line, and then the bottom 11-1/2 lines were full of exclamation points "!". (Let's see, 66 wide by 11-1/2 lines--egad, that's about 729 of those little 'animals', right?) Anyway, haven't tackled this problem yet--suspect a 2114 on the TLB as I had a similar problem with an ADM-3A dumb terminal a couple of years ago, & that was the problem. Went looking for my repair manual & no find! Musta took 'em up to the cabin where the H-89 is. Will be going up this Sunday, so will get the wrap-around problem & retrieve the manual & schematics to bring back to Hilo. (The cabin is up at Volcano, HI, about a mile from Kilauea Caldera, the active volcano--no danger though.)

"Again Hank, thanks for taking the time to try to keep us guys (who still love our H-89/90's) learning new things (to us). I for one, do appreciate it."

[From Dan Jerome, Burnsville, MN] "...Having read your last newsletter [#25] from cover to cover, I am alarmed to see so many people abandoning their H-89's. This leads me to think that you may be the captain of the Titanic, standing on the bridge and watching the ship go down. I suppose eventually it may happen, but I was not prepared for it to begin happening quite yet. I think there is still a nice mother-lode of gold in them thar hills. In other words, I still like the H-89 quite well.

"If you want, in the next issue you can include a note from me addressed to those who are planning to abandon their H-89s. It should read to the effect that instead of giving them to the junk man, they could do a **good deed** and find some single-parent family who would enjoy using them for years to come. One easy place to find a single-parent family would be to call a local church and talk to the church secretary. Or even some old folks home. It would give the old people something to keep their mind occupied and thus enhance their last days.

Well, here it is 1-Nov-91 and I have been watching it blizzarding outside since before sunup. By now we have over 20 inches of snow, and travel in the Twin Cities has virtually come to a screeching halt. There are hundreds of accidents and many 18-wheelers are in the ditch--some of them plugging up the freeway exit ramps. Same for the cars. The police have called up the reserves, but there aren't enough squad cars to go around. The wreckers are having a heyday.

"P.S. 2-Nov-91: We just got whacked with [another] 30 inches of snow! Most of the traffic stayed home. Those who ventured out either hit the ditch or became involved in a fender bender. I stayed home and watched Channel 5." [Thanks for your comments and the weather reports, Corky and Dan. (My thanks, too, to Hank for sending a copy of Corky's letter.) As for going down with a "sinking ship," I **knew** when I took over this rag from Hank that I was engaged in a "losing" battle insofar as potential and actual readership would decline as time went by. There **are** some good reasons for switching to higher-powered, modern technology as I observed above in my reply to Pete Shkabara's letter. There are things the '8 and '89 simply **can't** do, no matter how much we hold-outs may balk at the idea. Examples include instances where high-resolution graphics and color are required or in monsterous database applications. Many of these are business applications, where the PC and Macintosh were originally targetted anyway. But I'm not particularly impressed with the new where the old and the new share common ground.

[One example is word processing. Even speed typists can only move their fingers so fast over the keyboard. And the current generation of commercial word processors for the PC strike me as over-endowed dinosaurs. They would be consigned to California's Le Brea tar pits if it weren't for the expensive, high-speed machines needed to get any reasonable performance out of them.

[Another example is database systems. For most home and hobbyist applications, the power of a dBASE IV+ or Paradox simply isn't needed. In fact, in simpler situations, an '89 will consistently outperform a PC any day of the week!

[Further, we don't have to worry about such things as viruses and Trojan horses that currently infest the PC and Mac environments. True, you can obtain anti-viral software, but you also have to be extremely wary of software you obtain from p.d. sources. (You know about that, Dan!) It actually doesn't take much to spread the most virulent strains of these things, either. They're like something the kids bring home from school! And actually, the continuing emphasis on standard hardware and software platforms, such as that recently agreed upon by Apple and IBM, will only aggravate this situation. In my own humble opinion, the one sure way to eliminate the threat of viruses is to **diversify** the platforms. Viruses for the PC are no threat to the Mac and those for the Mac are no threat to the PC. **Neither** of these will run on the Z80! See my report on PC viruses in the next issue.

[But I should step off of my soapbox. Most of this you already know! The H-8 and H/Z-89/90 are about as "dead" as CP/M is, which means, not much. But to keep things going, we have to support each other. **That** is the bottom line to survival of **any** out-of-production machine. -Ed.]

=====

Multitasking for Real-Time Response... Under HDOS? Part 1 of 4

By David A. Shaw

Introduction. Over the last ten years or so, I've been called on to develop several data communications systems on 8080- and Z80-based microcomputers, including H-8's and H-89's. All delivered good real-time performance, and were built around a simple but very effective multitasking system of my own design. The purpose of this article is to describe the set of tools that I've developed over the years in the hope that they will inspire new applications for Heath 8-bit equipment.

Assembler programming experience is really needed to get the full impact of this article. Also, in the interest of space, I'll try to be brief. There is example code available on disk from Kirk, called EXAMPLE.ASM. EXAMPLE is a trivial and otherwise useless program that demonstrates the use of many of the tools I'll describe in this article. To get the full benefit of this article, you should order the disk. Many of the code examples in this article came from EXAMPLE. [Though written for HDOS, this material could be adapted to CP/M. If you would like this material, merely send me a preformatted diskette and a self-addressed, postage prepaid mailer; I'll transfer it for you. The files only require 68K, so will fit on standard hard-sector. -Ed.]

First, two quick definitions. By "real-time," I am referring to systems that must react to events in the real world, such as the arrival of characters at an I/O port, and take some action in time to have some effect on those real world events. This is programming to meet externally-imposed deadlines. If, for example, you don't read a character at an I/O port before the next one arrives, you will lose a character.

The term "multitasking" has been used and abused enough that almost no one knows what it means anymore. I am not referring to the ability of a computer to run several different programs at one time, such as a spreadsheet and a word processor. This is really "multiprogramming," although multitasking is usually also present. Multitasking as I am using the term is the ability to break a program into two or more subprograms, each of which is executed independently and concurrently, and work together cooperatively to accomplish the job at hand.

The processors we are using, the 8080 and Z80, do not have the ability to do more than one thing at a time. In a real-time multitasking system, you give each task periodic access to the processor, switching from task to task quickly enough and often enough that it appears from the viewpoint of the outside world that the computer is doing several things at once.

I should point out that while interrupts are critical to real-time programs, the coding of interrupt service routines is beyond the scope of this article.

Multitasking. Multitasking is often an inherent function of the operating system. OS/2, for example, offers multitasking along with a rich set of tools to support the interaction of the various tasks. Obviously, with the exception of hardware interrupt support, neither HDOS nor CP/M offer any form of multitasking support. We have to develop our own tools.

There are any number of approaches to multitasking that have been used to good effect. The most common is the "polling loop" or one of the standard variations on this technique. You code each independent task or process as a subroutine. Each subroutine is called in turn by the main polling loop, which looks like this:

```
LOOP    EQU     *       Top of polling loop
        CALL    TASK.1
        CALL    TASK.2
        ...
        CALL    TASK.n
        JMP     LOOP    Loop around and do it again
```

If the tasks are simple and don't take much time to complete, this can work well. The problem is that all the tasks share one common system stack. They cannot keep track of any state information from call to call by pushing it on the stack, and they can't hold data in the registers. This seriously complicates program logic.

Let's assume that TASK.1 is responsible for reading keystrokes from a terminal and sending them to the modem. This simple task can be handled by the polling loop approach. But let's say TASK.1 now has to recognize PF (programmable function) keys by looking for the leading ESC character and the following character and take some action on the key. Let's say PF1 commands the program to accept a file name, open the file, and send the contents to the modem one character at a time. Now TASK.1 has three different modes: pass characters to the modem, save characters in a file name, or read a file and pass the characters to the modem. Imagine having to code this as a subroutine that has to return to the caller quite often, keeping track of what it is supposed to do on the next call, with no stack and lacking the simple ability to save data in the registers!

TASK.1 needs to allow the other independent routines to run so that we maintain the appearance of doing several things at once. It might be easier if TASK.1 could simply CALL each of the other tasks from time to time, perhaps from one of it's own subroutines, getting control back after they run so that it can continue to do whatever it was doing.

Actually, if the other tasks were very simple, we could do just that. Then TASK.1 becomes the main task and the others become subroutines. But this is rarely the case; usually, the other tasks have their own complications and could use the ability to have their own stack, keep data in registers, and call TASK.1 from time to time.

Independently-written routines that call each other like this are called coroutines. TASK.1 calls TASK.2 to allow it to do some job, and when TASK.2 is done, it calls TASK.1, which resumes at the point immediately following it's call to TASK.2. When TASK.1 again calls TASK.2, TASK.2 resumes immediately following it's call to TASK.1!

Obviously, this can't work on a stack-based machine. We need to assign a separate program stack to each coroutine, then call the other tasks through an intermediary subroutine that I call SWAP.

SWAP, shown in Figure 1 [on the facing page], does the following when called:

1. Push all the registers.
2. Save the current stack pointer in a save area dedicated to the current task.
3. Load the stack pointer with the value previously saved for the next task to be run, changing stacks.
4. Pop all the registers and RETurn to the new current task.

SWAP changes from stack to stack, invoking each task in order in round-robin fashion. After a task calls SWAP, SWAP will eventually return to the task immediately after the call, leaving all the calling task's register values and it's stack pointer intact, almost as if nothing happened. In fact, a lot has happened: all the other tasks in the system have executed! The only effect on the calling task is the amount of time it takes to run the other tasks.

Each task in the system treats the set of other tasks in the system as one trivial subroutine that can be called at any time from anywhere. Each task can be as complex as it needs to be. It can use any of the features of the underlying microprocessor, including the stack. It's only responsibility is to call SWAP often enough that all tasks can meet their various deadlines.

SWAP requires that you define several fields:

```
*       Stacks for three of the four tasks
*
STKSIZE EQU     ???     Stack size (up to programmer)
        DS      STKSIZE First stack
STK.TO  EQU     *       Terminal output stack
        DS      STKSIZE
STK.PT  EQU     *       Printer stack
        DS      STKSIZE
STK.ST  EQU     *       Status task stack
```

(Remember that all stacks grow "down" toward low memory. Also, one of your tasks can use the system stack at 42.200A.)

```
*       Stack management table
*
STAKTAB EQU     *
STB.TI  DS      2       Term input task SP
STB.TO  DS      2       Term output task SP
STB.ST  DS      2       Status task SP
STB.PT  DS      2       Printer task SP
*
STAKADR DW      STB.TI  Current stack is TI
STAKNDX DB      0       Current index is TI
TASK.CT EQU     4       There are four tasks
```

You need to initialize each stack before starting the multitasking system:

```
        LXI  SP,STK.TO Init terminal output task
        LXI  D,TT.OUT Task starting address
        PUSH D        Return address
        PUSH D        Dummy PSW
        PUSH D        Dummy BC
        PUSH D        Dummy DE
        PUSH D        Dummy HL
*
        LXI  H,0
        DAD  SP       (HL) = (SP)
        SHLD STB.TO   Save in table
```

This is repeated for each stack. Then, to start the multitasking system, you simply "return" to the first task:

```
        POP  H        Clean the stack...
        POP  D
        POP  B
        POP  PSW
        RET           ...and "return" to the first task
```

To exit to HDOS, have the task that decides to quit do any required clean up, reload the system's SP value, and exit as usual.

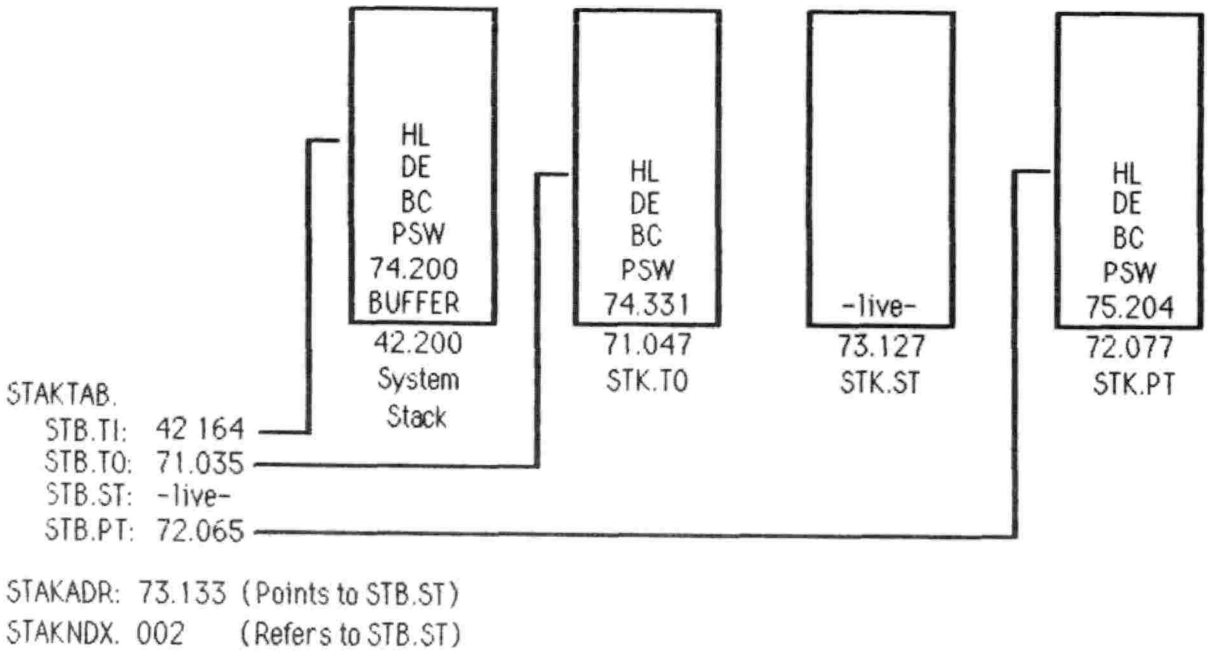
Figure 2 [below] shows what the stacks look like when EXAMPLE is running. Using DBUG, I stopped the program while it was in the status task. The STAKTAB entries for the terminal input, terminal output, and printer tasks point to the top of their respective stacks. STB.ST will hold an old, invalid value. Notice that STAKADR holds the address of the current STAKTAB entry, STB.ST, so that SWAP can easily save SP when it is next called, and that the index, STAKNDX, also refers to the STB.ST.

SWAP can handle any reasonable number of tasks, as long as you can get to each quickly enough to meet their various response time requirements. The real trick in designing a multitasking system is figuring out what tasks you need and what goes into each task. How do you know when to make a task

Figure 1: SWAP

```
*       SWAP    - Change to next task.
*                 Saves all registers on stack.
*
*       ENTRY   - (SP) = Current task stack pointer
*       EXIT    - (SP) = Next task stack pointer
*       USES    - SP
*
SWAP    EQU     *
        PUSH    PSW         Save all registers
        PUSH    B
        PUSH    D
        PUSH    H
        LXI     H,0
        DAD     SP
        XCHG                (DE) = Current (SP)
        LHLD    STAKADR
        MOV     M,E
        INX     H
        MOV     M,D         Put SP in the table
        LXI     H,STAKNDX
        MOV     A,M
        INR     A
        CPI     TASK.CT     Too big?
        JC      SWAP.1      No
        XRA     A           Else, wrap to zero
SWAP.1  EQU     *
        MOV     M,A         Update STAKNDX
        ADD     A           * 2
        LXI     H,STAKTAB
        CALL    $DADA.      (HL) = Stack table address
        SHLD    STAKADR     Update STAKADR
        MOV     E,M
        INX     H
        MOV     D,M
        XCHG                 (HL) = New (SP)
        SPHL                 Update (SP)
        POP     H            Restore all registers
        POP     D
        POP     B
        POP     PSW
        RET                  Start next task
```

Figure 2: TASK STACKS



versus when to code a new function in an existing task? Since independence is the goal, assign a different task to each independent activity or data flow. I usually start out assigning a task to handle every major input, and a task to handle every major device, and especially any device (beside the disk system) that is shared by more than one task.

In the case of a simple communications system, you might have the following tasks:

- one task to handle all character input from the terminal, decode and process PF keys and any other commands, and send data to the modem;
- one task to handle all characters received from the modem, sending them to the terminal, writing them to a file if desired, and queuing them for printer output if desired;
- one task to send output to the printer;
- perhaps a task to send status to terminal line 25 periodically;
- and, depending on how you decide to share the terminal, another task just to output to the terminal.

The need for the last three may not be obvious at first glance. I generally use a separate task to write to the printer because printers frequently run slower than the typical modem. My Diablo 630 prints at around 45 characters per second (cps), while my modem runs at 120 cps. It's easier to keep everything running smoothly if you put a buffer between the character receiver task and the printer task, then let both of them run at their own pace. More on this later.

It's very helpful in a communications system to maintain user-visible status, such as modem signals and free buffer space, on terminal line 25. However, you don't want to send status so often that there is no time left to send data to the terminal! The handling of timed tasks is covered in the next installment.

Finally, there are any number of ways to handle output to a shared device, such as the terminal being used to display both live data and periodic status received from two or more tasks. A couple of approaches are discussed in future installments.

=====

CONTACTS

(A Wanted/For Sale/Swap Column)

Mike Byrd (11 Japonica Lane, Shalimar, FL 32579, 904/651-9771) "I've since moved on to the DOS world (just too much good software out there), but still have lots of info on CP/M. I would appreciate it if you could put the following ad in the next **Staunch 8/89'er**. I think there is excellent reference material in the magazines and hate to see them thrown away.

FOR SALE (plus shipping) to CP/M User: The following magazines (the wife said they or I must go):

REMark (all issues from # 1 to 1989)
User's Guide (The Magazine for CP/M and MS-DOS Computer Users) (all issues)
Sextant (all issues)
BIOS listing for CP/M 2.2.04
SEBHC Journal (assorted issues)

All of the above are in two boxes (40 and 44 lbs). Free to the first person who comes and gets them or sends me money for shipping...

"I think this is a good deal for someone still in the 8-bit world and the **PRICE IS RIGHT**!"

Terry Hall (516 East Wakeman Ave., Wheaton, IL 60187-3670, 708/665-4594) "I finally got around to sorting and organizing all my 8-bit stuff I'm now willing to part with. I'm not abandoning the H-89/90 world altogether, as I'll keep one system, and I have many MB of data accumulated on 8" disks. But I've had to get heavily into the PC and Mac worlds for my work of creating hundreds of professional crossword puzzles. I have [a] Z-386/33E and a high-end Macintosh networked together on my desk. But right beside them on the same desk is my trusty H-89, which was my sole computer system from 1980 till last year.

"All those who responded to me earlier have received copies of all this. But I'd be grateful if you'd mention in your next issue that I have all this stuff and anyone interested call or send me a #10 SASE for a complete list...." [Terry sent me a massive, four-page, two-column inventory, most of it for the '89/90. It includes a wide selection of hardware (circuit boards, third-party add-ons, floppy drives, cabling, a Z-19 terminal, and spare '89s), HDOS software (device drivers; diagnostic disks; games; system distribution for 1.5, 2.0, 3.0, 3.02, and OMDOS/SMALLDOS; and software from Hoyle & Hoyle, Quikdata, ZUG, Interactive Micro, Keyboard Studio, Microsoft, Newline, Softshop, Softstuff, Software Toolworks, Software Wizardry, Sunflower, public domain, and miscellaneous), CP/M software (various compilers, dBASE II, WordStar, DESPOOL, Anapro's EMULATE, Magic Wand, Multiplan, QUERY!2, various spelling checkers, utilities, and public domain materials), books and manuals, Heath's continuing ed. courses for assembler and MBASIC, magazines, and a small selection of software, books, and hardware for PC's. I've only been able to scratch the surface in my description above! -Ed.]

=====

Connecting the Dots

By Paul Flexman

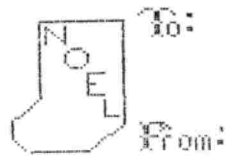
At one time or another, we all have probably tried the dot-addressable graphics on our dot matrix printers. What an experience!! I can remember thinking this is something to do when you are really bored. Counting those dots and trying to type them into a program without mistakes is enough to make a grown man cry. Well, I finally got bored enough and adventurous enough to try to develop a system to create graphics with little stress. I have used this method extensively and effectively on my Epson MX-80 and it should work equally well on similar printers that are equipped with dot addressable graphics.

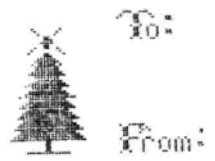
The following items will be needed to create your printer graphic:

- Pattern (see pattern discussion below)

- o Graph paper (I use graph paper that has 10 sq to the inch)
- o 3 x 5 card
- o Scratch paper
- o Computer program (optional - discussed later)

Pattern. Many items can be used for patterns. If you can find a design you want in a cross-stitch pattern, these work well. Those of you that have artistic talent (I'm jealous) can draw your own. All of my designs are first put on graph paper to judge size. Forty-nine lines will produce a graphic that is 11/16 of an inch high.





Learn by Doing. I find I can understand easier if I try someone else's way myself instead of reading about it. The following paragraphs will walk you through a simple graphic figure.

MAKING A PATTERN. Starting on the lower left, go up 49 squares and draw a line horizontally 3 to 4 inches in length. Place an empty soda can upside down in the lower left hand corner so that when you draw a line around it the line falls in the leftmost column and bottommost squares. Draw that circle. Move the can up to where the next circle will fall in the squares below the line drawn at 49. Draw that circle (neatness doesn't count). You should have ended up with two circles touching or overlapping at the center (a number 8).

CHANGING LINES INTO DOTS. When creating a graphic to be printed on a printer, you need some idea as to the result you want to achieve. A single row of dots will make a line finer than the line from a #2 pencil. For this example, every square that a line passes through will have a dot placed into it. Any side that does not have two dots together horizontally--use your best guess--add one.

On most printer graphics I have done, the rule that I use is that if 50% or more of the square (looking inside out) is on the inside, the square gets a dot. Viewing your work from 8-10 ft. will give you some idea how the finished work will appear when printed. Counting up from the last row of squares, draw a horizontal line every seven rows. Now, you should have seven rows of seven squares. Number the rows from the top down. Row #1, row#2,... row #7.

CHANGING THE ROWS INTO PIN NUMBERS FOR THE PRINTER. Place your 3x5 card to the right of Row#1 Column #1. In very small numbers (to line up with the squares) write the pin numbers along the edge of the card. Starting from the bottom, the numbers are 1, 2, 4, 8, 16, 32, 64. These numbers (or a combination) are the numbers that we will send to the printer to generate the dot pattern. On a piece of scratch paper, write line #1.

Starting with the left most column of row #1, we total the squares that have a dot in them and write them down in order. The 1st column will probably be 0 (no dots in any squares in that column). Note: I have found that if I have several columns in a row with the same number, it is easier for me when I type if I write the number of consecutive columns the number appears in followed by a dash and then the number, i.e., 3-0 (three columns of zeros), 4-127 (four columns with all seven dots used). Using the number to the right (on your 3x5 card) and adding the numbers together that are next to a dot, produces the number for that column. When you get to the last column with a dot in it, you can stop. It is not necessary to right justify all of the columns. Now check for any columns we might have missed. Count the numbers you have written down (3-60 counts as 3, 5-80 counts as 5, etc). Write this number over the top of Line #1. Counting the columns from left to right should give you the same number. I know it's boring, but for the next six rows do the same things:

1. Write down line #
2. Write down pin totals
3. Write down total number of numbers
4. Check against column used

Program. I have found Tiny Pascal for HDOS (ZUG pn 885-1086 [and in this issue's Software List -Ed.]) to be the easiest way to print my graphics. I created a base file that consists of these four lines duplicated seven times in order

```
PRINT (27,75,??,0);
PRINT (
PRINT (
PRINT (10);
```

To use this program, the line "PRINT (27,75,??,0);" should have the "??" replaced with the column count (the number written over the line #). The lines "Print(" should have the numbers after the line # typed in them in order with a comma separating them. Do this seven times (one set for each line). Precede the entire program with BEGIN and, of course, finish with END.

TROUBLESHOOTING. The program can be run now but the results will not be the finished results. The reason to run it now is to check for missing or incorrect pin codes. It is fairly easy to correct mistakes in this form (extra space between lines). If your printer beeps at you or gives you extra characters, check the number of characters in each line.

CLEANING UP. Once you are satisfied with the results, add the following two lines under BEGIN

```
PRINT(27,85,10);    !unidirectional printing!
PRINT(27,49,10);    !set line spacing to 7/72!
```

If you followed these instructions, you should now have the knowledge to create graphics of your own. Adding a monogram to stationery, creating distinctive labels, and even signing your name, are all possible by using printer graphics.

LISTING

(This listing has been edited for column width)

```
!PROGRAM CHRISTMAS LABELS BY PAUL FLEXMAN!

   CONST CR=10;
   VAR   X,T:INTEGER;

PROCEDURE WREATH;
BEGIN
   PRINT(27,75,77,0); !SEND 0 LINE GRAPHIC INFO!
   PRINT(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0);
   PRINT(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0);
   PRINT(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0);
   PRINT(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0);
   PRINT(2,4,8,8,8,7,7,8,8,8,4,2,0,0,0,3,3,CR);
   PRINT(27,75,77,0);  !SEND 1ST LINE GRAPHIC INFO!
   PRINT(0,0,0,0,0,0,0,1,3,3,7,15);
   PRINT(15,15,31,31,63,63,63,63,63,127,127,127);
   PRINT(127,127,127,63,63,63,63,63);
   PRINT(31,31,15,15,7,7,3,1);
   PRINT(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0);
   PRINT(0,0,0,0,33,94,94,33,0);
   PRINT(30,33,33,30,0,0,12,12,CR);
   PRINT(27,75,47,0);  !SEND 2ND LINE GRAPHIC INFO!
   PRINT(0,0,1,7,15,63,127,127,127,127);
   PRINT(127,127,127,127,127,126,126,124,124);
   PRINT(124,124,120,120,120,120,120,120);
   PRINT(124,124,124,126,126,127,127,127);
   PRINT(127,127,127,127,127,127,127,63);
   PRINT(31,15,3,1,CR);
   PRINT(27,75,49,0);  !SEND 3RD LINE GRAPHIC INFO!
   PRINT(7,63,127,127,127,127,127,127,127);
   PRINT(124,112,96,64,0,0,0,0,0,0,0,0);
   PRINT(0,0,0,0,0,0,0,0,0,0,0,64,96,112);
   PRINT(124,127,127,127,127,127,127,127);
   PRINT(127,127,127,63,7,CR);
   PRINT(27,75,48,0);  !SEND 4TH LINE GRAPHIC INFO!
   PRINT(127,127,127,127,127,127,127,127,127);
   PRINT(0,0,0,0,0,0,0,0,0,0,0,0,0);
   PRINT(0,0,0,0,0,0,0,0,0,0,0,0,0);
   PRINT(127,127,127,127,127,127,127,127,127);
   PRINT(127,127,CR);
   PRINT(27,75,48,0);  !SEND 5TH LINE GRAPHIC INFO!
   PRINT(96,124,127,127,127,127,127,124,124,124);
   PRINT(14,2,0,0,0,0,0,0,0,0,0,0);
   PRINT(0,0,0,0,0,0,0,0,0,0,0,1,2,14,60);
   PRINT(125,124,127,127,127,127,127,127);
   PRINT(127,126,112,CR);
   PRINT(27,75,86,0);  !SEND 6TH LINE GRAPHIC INFO!
   PRINT(0,0,0,96,112,124,126,0,127,127,127);
   PRINT(63,63,31,31,15,15,7,0,31,31);
   PRINT(31,31,31,31,31,31,0,15,31,31,31,63,63);
   PRINT(127,127,127,127,0,127,127);
   PRINT(126,124,120,96,64,0,0,0,0);
   PRINT(0,0,0,0,0,0,0,0,0,0);
   PRINT(8,7,7,8,8,8,10,4,2,0,0,0,0);
   PRINT(0,0,0,0,0,0,0,0,0,0,0,3,3,CR);
   PRINT(27,75,87,0);  !SEND 7TH LINE GRAPHIC INFO!
   PRINT(0,0,0,0,0,0,0,0,127,126,126,124);
   PRINT(124,124,124,120,120,113,113,3);
   PRINT(127,127,126,124,120,124,126,127);
   PRINT(7,115,113,113,120,120,124,124);
   PRINT(126,126,127,0,0,0,0,0,0,0,0,0,0);
   PRINT(0,0,0,0,0,0,0,0,0,0,0,0);
   PRINT(33,94,94,33,32,32,0,31,32,32,16);
   PRINT(0,30,33,33,30,0,31,32,31);
   PRINT(32,31,0,0,12,12,CR);
END;

PROCEDURE STOCKING;
BEGIN
   PRINT(27,75,57,0);  !SEND 0 LINE GRAPHIC INFO!
   PRINT(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0);
   PRINT(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0);
   PRINT(0,0,0,0,0,0,0,0,0,0,2,4,8,8,8);
   PRINT(7,7,8,8,8,4,2,0,0,0,3,3,CR);
   PRINT(27,75,57,0);  !SEND 1ST LINE!
   PRINT(0,0,0,0,0,0,0,0,0,127,64,95,72);
   PRINT(68,66,65,64,95,64,64,64,64,64,64);
   PRINT(64,64,64,64,64,64,64,64,64,64);
   PRINT(64,127,0,0,0,0);
   PRINT(0,0,0,0,33,94,94,33,0,30,33);
   PRINT(33,30,0,0,12,12,CR);
   PRINT(27,75,36,0);  !SEND 2ND LINE!
   PRINT(0,0,0,0,0,0,0,0,0,127,0,96,0,0,0,0);
   PRINT(64,99,4,8,8,8,8,4,3,0,0,0,0,0);
   PRINT(0,0,0,0,0,127,CR);
   PRINT(27,75,36,0);  !SEND 3TH LINE!
   PRINT(0,0,0,0,0,0,0,0,0,127,0,0,0,0,0,0,0);
   PRINT(96,16,8,8,8,8,16,99,2,2,2,2);
   PRINT(2,0,0,0,0,0,127,CR);
   PRINT(27,75,36,0);  !SEND 4TH LINE!
   PRINT(0,0,0,0,0,0,0,0,0,127,0,0,0,0,0,0);
   PRINT(0,0,0,0,0,0,0,0,127,17,17,1,1);
   PRINT(1,0,0,0,0,0,127,CR);
   PRINT(27,75,36,0);  !SEND 5TH LINE!
   PRINT(0,0,0,1,2,4,8,16,32,64,0,0,0,0,0);
   PRINT(0,0,0,0,0,0,0,0,0,0,0,0,0,0,63);
   PRINT(0,0,0,0,0,127,CR);
   PRINT(27,75,66,0);  !SENT 6TH LINE!
   PRINT(63,64,64,0,0,0,0,0,0,0,0,0,0,0,0);
   PRINT(0,0,0,0,0,0,0,0,0,0,0,0,0,0,64);
   PRINT(64,64,65,66,4,120,0,0,0,0);
   PRINT(8,7,7,8,8,8,10,4,2,0,0,0,0,0,0);
   PRINT(0,0,0,0,0,0,0,0,0,3,3,CR);
   PRINT(27,75,66,0);  !SENT 7TH LINE!
   PRINT(96,16,8,4,2,1,1,1,1,1,1,1,1,1,1);
   PRINT(1,1,1,1,1,1,1,1,1,1,2,4,8,16,32);
   PRINT(64,64,0,0,0,0,0,0,0,0);
   PRINT(33,94,94,33,32,32,0,31,32,32,16);
   PRINT(0,30,33,33,30,0,31,32,31);
   PRINT(32,31,0,0,12,12,CR);
END;

PROCEDURE CANDLE;
BEGIN
   PRINT(27,75,61,0);  !SEND 0 LINE GRAPHIC INFO!
   PRINT(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0);
   PRINT(0,0,0,0,0,0,0,0,0,0,0,0,0,0);
   PRINT(0,0,0,0,0,0,0,0,0,0,0,0,2,4,8,8,8,7);
   PRINT(7,8,8,8,4,2,0,0,0,3,3,CR);
   PRINT(27,75,61,0);  !SEND 1ST LINE GRAPHIC INFO!
   PRINT(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0);
   PRINT(0,0,0,6,31,127,31,6);
   PRINT(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0);
   PRINT(0,0,0,0,33,94,94,33,0,30);
   PRINT(33,33,30,0,0,12,12,CR);
   PRINT(27,75,23,0);  !SEND 2ND LINE GRAPHIC INFO!
   PRINT(0,0,0,0,0,0,0,0,0,0,0,0);
   PRINT(0,0,0,0,0,15,79,127,79,15,CR);
   PRINT(27,75,23,0);  !SEND 3RD LINE GRAPHIC INFO!
   PRINT(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0);
   PRINT(127,127,127,127,127,CR);
   PRINT(27,75,23,0);  !SEND 4TH LINE GRAPHIC INFO!
   PRINT(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0);
   PRINT(127,127,127,127,127,CR);
   PRINT(27,75,26,0);  !SEND 5TH LINE GRAPHIC INFO!
   PRINT(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1);
   PRINT(127,127,127,127,127,1,1,1,CR);
   PRINT(27,75,70,0);  !SEND 6TH LINE GRAPHIC INFO!
   PRINT(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,64);
   PRINT(96,127,127,127,127,127,96,64);
   PRINT(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0);
   PRINT(8,7,7,8,8,8,10,4,2,0,0,0,0,0);
   PRINT(0,0,0,0,0,0,0,0,0,0,3,3,CR);
   PRINT(27,75,70,0);  !SEND 7TH LINE GRAPHIC INFO!
   PRINT(12,30,63,63,63,63,63,63,63,63,63);
   PRINT(63,63,63,63,63,63,63,127,127);
   PRINT(127,127,127,63,63,63,63,63,63,63);
   PRINT(63,63,63,63,63,63,63,63,63,30,12);
   PRINT(0,0,0);
   PRINT(33,94,94,33,32,32,0,31,32,32,16);
   PRINT(0,30,33,33,30,0,31,32,31);
   PRINT(32,31,0,0,12,12,CR);
END;

PROCEDURE TREE;
BEGIN
   PRINT(27,75,51,0);  !SEND 0 LINE GRAPHIC INFO!
   PRINT(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0);
   PRINT(0,0,0,0,0,0,0,0,0,0,0,0,0,0);
   PRINT(0,0,0,0,2,4,8,8,8,7,7);
   PRINT(8,8,8,4,2,0,0,0,3,3,CR);
   PRINT(27,75,51,0);  !SEND 1ST LINE GRAPHIC INFO!
   PRINT(0,0,0,0,0,0,0,16,8,4,1,3,3,1,4,8);
   PRINT(16,0,0,0,0,0,0,0,0,0,0,0,0,0);
   PRINT(0,0,0,0,0,0,0,0,33,94,94,33,0);
   PRINT(30,33,33,30,0,0,12,12,CR);
   PRINT(27,75,18,0);  !SEND 2ND LINE GRAPHIC INFO!
   PRINT(0,0,0,0,0,0,4,8,16,1);
   PRINT(67,119,119,67,1,16,8,4,CR);
   PRINT(27,75,18,0);  !SEND 3RD LINE GRAPHIC INFO!
   PRINT(0,0,0,0,0,0,2,38,95,127,127);
   PRINT(127,127,127,127,95,38,2,CR);
   PRINT(27,75,20,0);  !SEND 4TH LINE GRAPHIC INFO!
   PRINT(0,0,0,0,1,19,55,127,127,127,127);
   PRINT(127,127,127,127,127,127,55,19,1,CR);
   PRINT(27,75,21,0);  !SEND 5TH LINE GRAPHIC INFO!
   PRINT(0,0,0,9,27,63,127,127,127,127,127);
   PRINT(127,127,127,127,127,127,127,63);
   PRINT(27,9,CR);
   PRINT(27,75,60,0);  !SEND 6TH LINE GRAPHIC INFO!
   PRINT(0,4,77,95,127,127,127,127,127,127);
   PRINT(127,127,127,127,127,127,127,127);
   PRINT(127,127,95,77,4,0,0,0,0,0,0,0,0,0,0,0);
   PRINT(8,7,7,8,8,8,10,4,2,0,0,0,0,0);
   PRINT(0,0,0,0,0,0,0,0,0,0,3,3,CR);
   PRINT(27,75,60,0);  !SEND 7TH LINE GRAPHIC INFO!
   PRINT(32,96,96,96,96,97,97,97,97,97,97);
   PRINT(127,127,97,97,97,97,97,97,96,96);
   PRINT(96,96,32,0,0,0,0,0,0,0,0,0,0);
   PRINT(33,94,94,33,32,32,0,31,32,32,16);
   PRINT(0,30,33,33,30,0,31,32,31);
   PRINT(32,31,0,0,12,12,CR);
END;

PROCEDURE BOXES;
BEGIN
   PRINT(27,75,47,0);  !SEND 0 LINE GRAPHIC INFO!
   PRINT(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0);
   PRINT(0,0,0,0,0,0,0,0,0,0,0,0,0,0);
   PRINT(2,4,8,8,8,7,7,8,8,8,4,2,0,0,0,3,3,CR);
   PRINT(27,75,47,0);  !SEND 1ST LINE GRAPHIC INFO!
   PRINT(0,0,0,0,0,0,0,0,0,0,12,19,16);
   PRINT(8,4,3,0,0,0,0,0,0,0,0,0,0,0,0,0,0);
   PRINT(0,0,0,0,33,94,94,33,0);
   PRINT(30,33,33,30,0,0,12,12,CR);
   PRINT(27,75,28,0);  !SEND 2ND LINE GRAPHIC INFO!
   PRINT(0,0,63,48,107,68,66,65,65,65,33);
   PRINT(97,51,31,15,13,124,36,102,69);
   PRINT(69,69,69,37,57,5,3,1,CR);
   PRINT(27,75,28,0);  !SEND 3RD LINE GRAPHIC INFO!
   PRINT(0,0,127,0,0,0,0,127,0,0,0);
   PRINT(0,0,0,0,64,63,31,15,8,72,54);
   PRINT(0,0,0,0,0,127,CR);
   PRINT(27,75,28,0);  !SEND 4TH LINE GRAPHIC INFO!
   PRINT(63,24,116,19,18,26,30);
   PRINT(116,19,18,10,6,3,0,0,127);
   PRINT(127,127,0,0,0,0,0,0,0,127,127,CR);
   PRINT(27,75,30,0);  !SEND 5TH LINE GRAPHIC INFO!
   PRINT(127,0,0,127,0,0,0,127,127,0,0,0);
   PRINT(63,31,24,20,114,113,113,25,29,23);
   PRINT(19,17,17,17,9,125,3,1,CR);
   PRINT(27,75,56,0);  !SEND 6TH LINE GRAPHIC INFO!
   PRINT(112,8,4,126,2,2,2,126,2,2,2,126);
   PRINT(127,0,0,0,127,0,0,0,0,127,127);
   PRINT(0,0,0,0,0,127);
   PRINT(8,7,7,8,8,8,10,4,2,0,0,0,0,0,0);
   PRINT(0,0,0,0,0,0,0,0,0,3,3,CR);
   PRINT(27,75,56,0);  !SEND 7TH LINE GRAPHIC INFO!
   PRINT(0,0,0,0,0,0,0,0,0,0,0,0,96,8);
   PRINT(4,2,127,1,1,1,1,127,127,1,1,1);
   PRINT(1,1,127,0);
   PRINT(33,94,94,33,32,32,0,31,32,32);
   PRINT(16,0,30,33,33,30,0,31,32,31);
   PRINT(32,31,0,0,12,12,CR);
END;

BEGIN
   PRINT(27,49,CR);     !SET LINE SPACING TO 7/72!
   PRINT(27,85,CR);     !UNIDIRECTIONAL PRINTING!
   WRITE(27,69,CR);
   WRITE(CR,CR,'CHRISTMAS LABELS');
   WRITE(CR,'By Paul E. Flexman');
   WRITE(CR,CR,'This program prints Christmas ');
   WRITE('labels of five different designs');
   WRITE(CR,CR,'First label printed should by ');
   WRITE('lined up with top of bail bar.');
   WRITE(CR,CR,'RETURN WHEN READY');
   READ(T);
   PRINT(CR,CR);
   WREATH;
   PRINT(CR,CR);
   STOCKING;
   PRINT(CR,CR);
   CANDLE;
   PRINT(CR,CR);
   TREE;
   PRINT(CR,CR);
   BOXES;
   PRINT(CR,CR);
   WRITE(27,'@',CR);
END.
```

=====

The Linkage Loader
(A column of reader-furnished routines)

Writing Memory-Resident Programs under WHM. [From Lee Hart, 323 W 19th St., Holland, MI 49423; continued from the "Letters" column] "To Mark Hunt and Peter Shkabara on memory resident programs: You can write your own memory resident CP/M programs quite easily with **Write-Hand-Man.** The various applications supplied (notepad, calculator, etc.) are really just ordinary CP/M programs that WHM loads into a protected space in high memory. These programs can then be called any time by console, printer, or disk accesses. Source code is supplied for the Notepad to show how this is done.

"Loading WHM does three things. First, WHM is attached to the BDOS, so any following programs you run will think you have 5K less memory. Second, WHM creates a 'hole' in high memory to load user-defined programs. Third, the BDOS and BIOS entry points are patches so WHM can monitor console, printer, and disk I/O calls.

"When WHM is triggered, it saves the state of the CP/M environment, and creates a complete duplicate, with its own TPA. Memory resident programs load into this private TPA and are executed. Though all BDOS calls are available, you should avoid using #0 SYSTEM RESET and #13 RESET DISK. They make major changes in the disk system, and cause trouble for the program that was interrupted (closing its files prematurely, etc.).

"Your memory resident program exits back to WHM, which restores the CP/M environment. The original program can then continue, unaware of any interruption.

"WHM loads your memory resident program in high memory instead of at 0100h. So it is kept on disk as a relocatable .REL file instead of a .COM file. If the program is too big, or you only have its .COM file (no source code), WHM includes a SWAP application that saves whatever is in the TPA to disk, then loads your program. When the program exits, SWAP restores the original program in the TPA and resumes executing it exactly where it left off.

"To create your own memory resident application, write and debug it as a normal CP/M .COM program first. Aim for a 1.5K TPA size; if you need more, you can reconfigure WHM to reserve up to 9K.

"Now change all absolute memory addresses to 'BASE+address'. Thus to call the BDOS, don't use 'CALL 5'; use 'CALL BASE+5'. BASE is the address of the reserved memory that WHM will actually load your program into. You need not define BASE; WHM will do it for you at load time.

"WHM closely duplicates the CP/M page 0 memory map. BASE+5Ch has an FCB filled in for the file 'name.DAT' wnere 'name' is the name of the application. The DMA address is set to BASE+80h. Execution of the application begins at BASE+100h. BASE+0 has a JMP instruction to the return point of WHM. BASE+5 is a JMP to the BDOS.

"The user number and logged-on disk are set as defined when the 'WHM ON' command was issued. Your application can change the USER number, logged-on drive, and DMA address; they will be restored when control returns to WHM.

"There are two new BIOS functions available for application programs. BASE+10h has a JMP to a routine to home the cursor; 'CALL BASE+10h' thus homes the cursor. There are also a set of extended BDOS functions, which take less code to call and provide easier access to function keys and screen graphics. BASE+13h has a JMP to a chaining routine, so an application can use overlays or chain control to another application. Put the name of the new application at BASE+5Dh (the file name part of the default FCB, with trailing blanks). Now execute a 'JMP BASE+13h'. The new application is loaded and executed with the environment set up as described above.

"WHM provides a 16-level (32-byte) stack for the application; most applications need not use their own stack area (a space saver).

"The JMP instruction at BASE+0 doesn't go to the standard BIOS vector table; use absolute address 0 as for normal CP/M. Remember to return to BASE+x, or WHM will lose control.

"Once you've replaced the absolute addresses in your application, assemble it with the Microsoft M80 assembler. A typical command line would be:

```
A>M80 yourfile,yourfile=yourfile
```

This makes 'yourfile.REL' and 'yourfile.PRN' from 'yourfile.MAC'. Any other assembler that produces relocatable (.REL) files compatible with M80 can be used, though we've only tested it with M80.

"To load your application, call WHM and hit any key to get the 'enter filename >' prompt. Type your filename, then RETURN. The WHM loader accepts a subset of the LINK-80 loader commands. It cannot search libraries, and recognizes only CSEG addresses (Code SEGment, the default); do not use any ASEG, DSEG or COMMON directives in your application.

"To end your application, exit with a RETURN to go back to the WHM menu, or JMP BASE to return immediately to the interrupted program. To suspend your application (so you can continue it later), write the address to resume at (BASE+*) into the variable CHAIN, then JMP BASE to exit. The next time WHM is triggered, it will immediately resume execution at address CHAIN, without displaying a menu or prompting for a new application.

"The prologue of Notepad is included here as an example of how to code your application.

```
;
; NOTEPAD by Lee Hart, TMSI
;
        org  0
base    equ  $
bdos    equu base+5    ; WHM's BDOS entry
home    equ  base+10h  ; WHM's home cursor subroutine
chain   equ  base+13h  ; WHM's chaining subroutine
functab equ  base+18h  ; base address of Clipboard
bdosi   equ  base+1Ah  ; extended BDOS functions
fcb     equ  base+5Ch  ; WHM's file control block
fcbcr   equ  fcb+32
fcbr0   equ  fcb+33
fcbr2   equ  fcb+35
fcbs2   equ  fcb+14
buf     equ  base+80h  ; WHM's DMA buffer
;
listout equ  5         ; BDOS functions used
conio   equ  6
open    equ  15
```

```
close   equ  16
make    equ  22
setdma  equ  26
ranread equ  33
ranwrit equ  34
fsize   equ  35
;
        org  100h      ; actual program starts here
        call home      ; home cursor subroutine
        mvi  b,16
topline:lxi  h,top     ; display top line of notepad
        call pstring
        dcr  b
        jnz  topline   ... and so on
```

"WHM owners can order a disk with full source code and instructions for all WHM applications for $10 from TMSI c/o Lee Hart, 323 West 19th, Holland MI 49423. Be sure to identify the desired disk format."

=====

Pete on CP/M

By Peter Shkabara

Hey, I got to writing a lot sooner this time - started July 6th. I was even going to tease Kirk about going to press quickly. Alas, other things got in the way and here it is six weeks later and I finally get it done.

Hank Lotz sent me a note to thank me for finally answering some of his queries in my last installment. He did however, have some comments on my comments. The first comment had to do with my expression of "UGH" in reference to BASIC programming. No, I do not hate BASIC. I can even be caught using the language on occasion. In fact, Quick Basic which comes with version 5 of MS-DOS is kind of neat! The problem is that BASIC does not lend itself to good programming practice and large programs are usually a real terror to work on. Pascal or C do a much better job.

Regarding Hank's request for a format program to format a single track, some clarification is in order. My AFORM program (source code available) can easily be modified to do just that. However, AFORM is only for H-37 soft sector disks. I do not have a source listing of the H-17 formatter. Mark Brooks of C.D.R. had disassembled the Heath code and created a new version for the C.D.R. controller software package. It would be possible to hack the existing object code by poking around with DDT or DSD, but it would be so much easier if Mark was to share his source with us. Can you carry the ball on this one, Hank?

Hank clarified his problem with DDT eating up TPA when KEYMAP was installed. Seems that after DDT loaded itself in and marked appropriate low address vectors to protect itself from the program it was debugging, it was unable to restore proper addresses on exit. Must be KEYMAP restoring some vector along the way to protect itself also. A little explanation may be in order for those who are totally lost at this point (which may be most of the readers out there).

DDT is a program which makes possible the debugging and modification of another program. To allow the loading and running of another program in the TPA while debugging it, DDT has to move itself into high TPA area. To prevent the other program from clobbering DDT in high TPA, DDT makes a patch to some CP/M vectors in the area below the 100hex start of TPA. The debugged program then thinks that DDT is actually part of CP/Ms BDOS and leaves it alone. This is essentially the same thing that memory resident programs such as KEYMAP also do. So here we have both KEYMAP and DDT trying to make everything else see them as BDOS. How successful DDT is in restoring the original condition of the system after it is through depends on what patches KEYMAP originally made. Remember that being memory resident is not a standard CP/M procedure and must be considered a hackers dream come true! Thus conflicts can (and apparently do) arise.

All this now leads us to Hank's original request number three of how to write a memory resident program to intercept terminal or disk I/O calls. The memory resident part means that the program has to move itself into high TPA and then patch the CP/M BDOS vector at address 0005 so that applications think that the BDOS is now lower. One problem with doing this is that BDOS is above the CCP (the command processor) while the resident program is below it. Normally the CCP may be overwritten by a transient program if it needs the TPA area. If the memory resident program sits below the CCP and then makes the transient program think that the BDOS is below the actual CCP area, you loose several kilobytes of TPA. For this reason I do not like this approach. There is another way.

Do you remember the MOVCPM program? Do you know what it does? In case your answer is no to either of the questions, I will explain a bit. MOVCPM is a program developed by Digital Research and modified by Heath. Its purpose is to adjust the addresses within CCP, BDOS and BIOS to allow positioning them to run at various locations in memory. If a size is specified as part of the MOVCPM command, the CCP, BDOS and BIOS will be adjusted to fit just within the amount of RAM specified (in kilobytes). Otherwise, an asterisk (*) will force an adjustment to the maximum RAM available. By specifying a RAM size smaller than the true size, it is possible to leave some **holes** at the top of RAM. Remember that this is all done as a memory image. You need to run SYSGEN immediately afterwards in order to save the RAM image to disk.

[You can also SAVE the image to disk as a named **file** for later use by SYSGEN in circumstances where you've particularly customized the system. This is useful, for example, when preparing a bootable system for program development under various Pascal (or other) compilers to ensure that those programs will load and run on computers where the available RAM is significantly **less** than on your own machine. SAVEing custom systems also reduces the hassle of having to rerun MOVCPM and/or MAKEBIOS on those rare occasions when you need them. -Ed]

Let us say that we have 64K of RAM and run MOVCPM 63 followed by SYSGEN. Of course you will actually run MOVCPM17, MOVCPM37 or whatever is actually needed. The variations come from the Heath mods which put appropriate bootup code in the first sectors of the disk. In any case, we will have 1K byte of RAM free above the BIOS portion of CP/M. No program or CP/M activity can bother this area. CP/M and all its application programs think that there is only 63K of RAM in the computer! Enter our desired memory resident modification. Instead of the "normal" below CCP installation, let us install the mod into the very top of RAM. Although CP/M doesn't know there is that space, our program does (we wrote it). The only patches to CP/M that need to be done now will depend on what is desired from the program. Most likely we will need to patch into the BDOS itself, or into the BIOS vector table at the start of BIOS. Both locations are known to our program, and the functions are standard CP/M documented items. From here your assembly language skills are put to use.

An almost indispensable book in writing such programs is **Inside CP/M** by David Cortesi, (1982, Holt, Rinehart and Winston Publishers). I have had a collection of many CP/M references and some were quite useful, but Cortesi's book was the most comprehensive although not easiest to read. It may be out of print now, but can still be found.

For those who have been exposed to the Z-System, there is another choice. Due to running out of time (or motivation), I never implemented the replaceable device driver feature in my Z-System BIOS. It was certainly planned, but sales never picked up enough to justify the effort commercially. My own needs did not require it since I was starting to switch over to an MS-DOS system at the time. Rick Swenton has done it on his system and perhaps he may be convinced to contribute such mods for you readers out there. [I've approached Rick about this question and he is willing. -Ed.] By the way, you readers seem to be very quiet lately. Are you out there?

Perhaps a real software construction article next time, but I better send this in to the press before another six weeks go by.

=====

Troubleshooting the '89, Pt. 3: The H-89 Power Supply Module

By Daniel N. Jerome

Introduction. The scope of this document is to dissect the power supply of the H-89 computer. However, the reader should understand that there are other major modules which make drastic changes to the power provided to them by the H-89 power supply. These modules include the video circuit board with its Horizontal Sweep and High Voltage, the Video Driver Circuit Board, and the Terminal Logic Circuit Board (commonly referred to as the TLB).

The H-89 first hit the market in 1978. At that time the power supply was the best in the business. It was designed to be heavy-duty and to last a long time. When you compare it against its main competitors, such as the Radio Shack TRS-80, the H-89 was far ahead in all features. By rights, one must give credit to the original Heath engineers who designed the H-89 and its power supply. However, with the passage of time and new developments, electronic requirements change. The power supply that was overdesigned for 1978 requires various enhancements in order to handle current demands made upon it by the more advanced accessories of modern times. These improvements will be discussed later on in this document.

For those who do not know, the power supply is a device which converts the AC voltage to DC voltage delivered at various maximum currents. This DC power feeds the various circuits of the computer. There is a difference between voltage and current. As an example for the uninitiated, think of the voltage as a city water tank built on a hill. The water pressure available is the same in that tank. The pressure or electromotive force represents the water pressure. The current can be compared to different diameter pipes that come forth from the water tank. The larger the pipe, the more pressure can be delivered. Various components in the computer demand their own specific amounts of current.

Refer to Figure 1: H-89 Power Supply Module - Exploded View [on the facing page] for detailed layout.

(1) THEORY OF OPERATION. The primary circuit of the power supply consists of slow-blow fuse F1, ON/OFF switch SW3, 115/230-volt switch SW1, NORM/LOW line switch SW2, and the primary windings of transformer T1.

The red secondary windings of transformer T1 supply AC voltage to the discrete diode bridge rectifier network composed of diodes D109 thru D112. The 65-volt rectified output of the bridge is filtered by capacitor C1. This 65-volt supply provides voltage to the Video Board via connector P202.

The yellow secondary winding of transformer T1 supplies AC voltage to the diode bridge rectifier BR1. The rectified output of the bridge (approximately 9 volts DC) is filtered by capacitors C101 and C103, passes through 5-volt regulators U101 and U102 which provide two regulated 5-volt supplies and 12-volt regulator U103, which provides one regulated 12-volt supply. The voltage supply from U101 is used on the CPU Logic Circuit Board. The voltage supplies from U102 and U103 are used to supply the H-17 (and H-37/H-47/H-67 if applicable) floppy drive controllers and the three-port serial board. The 5-volt and 12-volt lines enter the CPU Logic Board at connector P515.

The green secondary windings supply center-tapped 30 volts AC to the discrete diode bridge rectifier network composed of diodes D101 through D104. The rectified outputs of the bridge, + and - 18 volts DC, are filtered by capacitors C102 and C104. These voltage supplies provide voltages and currents to the CPU Logic Board thru connector P154 and to the Terminal Logic Board through connector P515.

The three power supplies: (1) +65 volts, (2) +8.5 volts, and (3) +/- 18 volts are not interconnected on the power supply circuit board. Instead, they pick up their appropriate circuit grounds at the circuit boards they power. The +65 volt video supply connects to + and ground points on the Video Circuit Board. The external conductive coating of the CRT (Cathode Ray Tube) and the CRT socket arc-ring both connect directly to the Video Circuit Board ground.

The +8.5 volt and the + and - 18 volt supplies connect directly to the Logic Circuit Boards with no common grounds until they meet at the Terminal Logic Circuit Board.

This grounding technique produces two independent operating systems that do not interact with

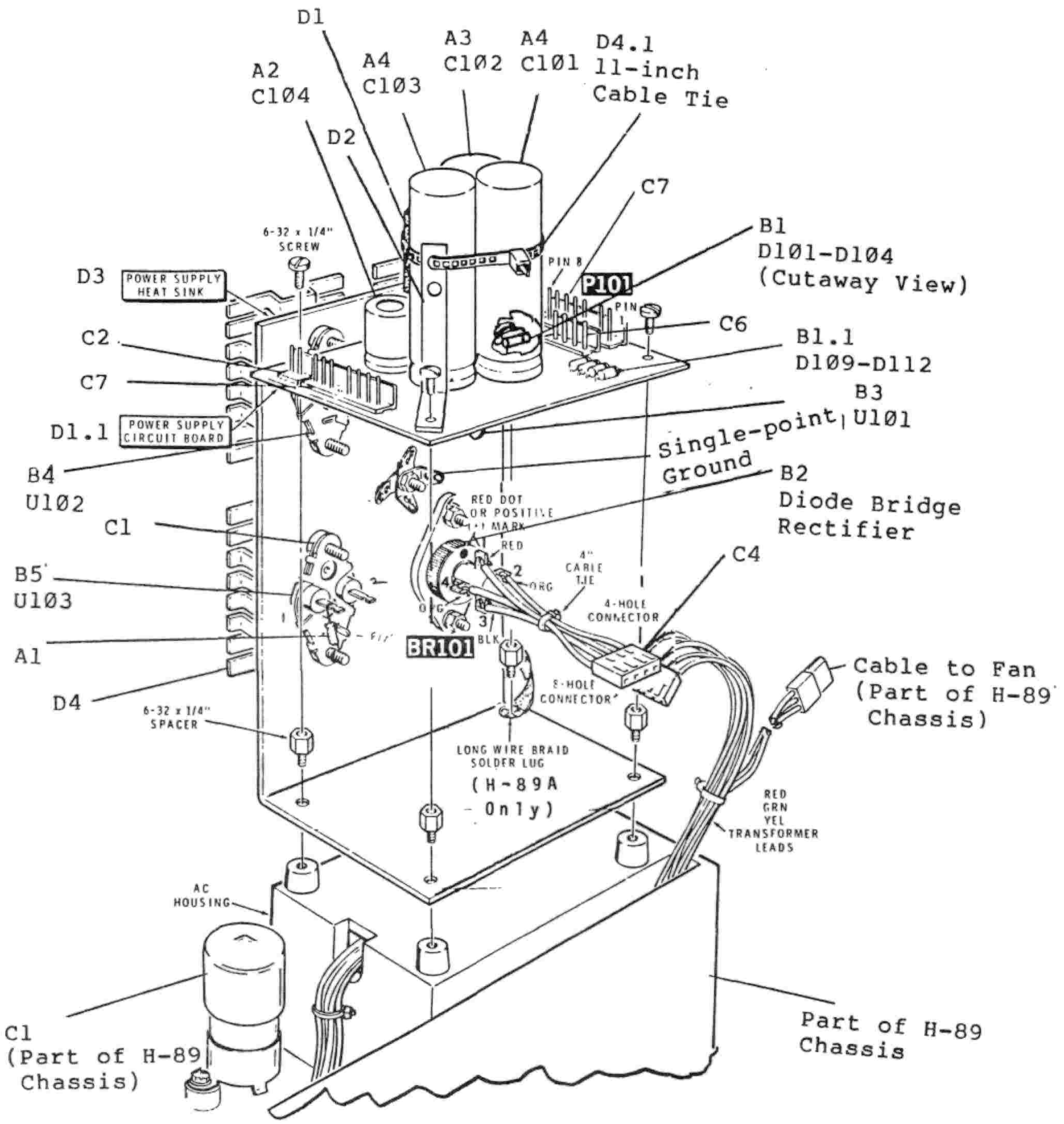


FIGURE 1: H-89 Power Supply Module - Exploded View

each other, except through the signal ground and synch/video inputs. In the event of a CRT **arc**, the arc discharge current is confined to the Video Circuit Board and it does not induce transients into the logic circuits.

The protective ground input (pin 1) of the EIA RS-232 connector connects to the Terminal Logic Board ground.

(2) POWER SUPPLY ENHANCEMENTS: (A) Upgrade the diode bridge rectifier, BR1. After a period of use the standard Heath part, BR1, begins to get flaky. You notice diagonal lines appearing on your screen, especially if you have added new accessories. The best way to solve or prevent these problems is to change the Diode Bridge Rectifier, BR1, for a unit that provides more current. Even if you could purchase a replacement part from Heath, it would be capable of supplying only marginal current.

Purchase the following parts from Radio Shack or an equivalent vendor:

Full-Wave Bridge, 25-Amp, 50PIV, R-S Part No. 76-1185, $2.69
Heat Sink Compound, R-S Part No. 276-1372, $1.59

Refer to Figure 1: The H-89 Power Supply Module- Exploded View for details, and replace the diode bridge rectifier in accordance with the following steps:

1. First insure that AC power is **off** and the computer cover is removed.
2. Unfasten and remove the four #6 screws that hold the power supply circuit card to the chassis, and swing the chassis to the side. To do this, you do not need to disconnect the cables, but you may if you wish.
3. Remove the standard Heath issue BR1 from the power supply heatsink, and clean up the place where it was fastened with a clean cloth rag.
4. Then fit the bridge in the power supply heat sink, and orient the unit so the terminal marked plus (+) faces upward.
5. Drill one hole in the power supply heatsink that will accommodate a #6 screw at a convenient location as close to the center of the bridge as possible. The caution here is that the wiring must be able to reach the appropriate terminal.
6. Heat sink compound is not toxic. Pick up some silicone heatsink compound on your fingers, and smear it liberally on the backside of the new bridge.

CAUTION: Insure that you don't touch this compound to your eyes or your clothing. You cannot clean it from your clothing by any method known to science!

7. Secure the new bridge to the power supply heat sink, using a #6 screw of the appropriate length and a star lock washer.
8. Using a 25-watt soldering iron, or the equivalent, resolder the wires to the new part. The red output wire goes to the terminal that indicates plus (+). The black output wire goes to the terminal that indicates minus (-). Each of the yellow wires go to the two remaining terminals, but it doesn't matter which wire goes to which terminal, since the input voltage seen by the bridge rectifier is AC.

NOTE: If you opt to perform the connectors P101 and P103 modification described below, ignore the yellow wires. If you decide not to perform this other modification, then the yellow wires must be snipped off from connector P101 and connected directly to the diode bridge rectifier. This came as a Heath bulletin some time ago.

9. Now, double-check your wiring. Red wire goes to the (+) terminal and black wire goes to the (-) terminal. If there is no terminal marked (-), assume that it is the terminal opposite the (+) marked one.
10. Once you are done, reattach the power supply circuit card. The next time you turn on your H-89, you should notice a clearer and steadier screen.
11. The final step is to insure that the H-89 fan, located on the top cover, is blowing down toward the power supply. This will tend to keep it cooler than if it were blowing upward.

(B) Replace Power Supply Connectors P101 and P103. After a period of use, the pins (male) and sockets (female) at connectors P101 and P103 corrode and resistance, with its consequent heat, increases to the point that it can cause an electrical fire! The problem is that different types of pins and sockets were used in the original design, and corrosion is inevitable. Connectors P101 and P103 may be found at the rear of the power supply circuit card. Connector P101 has 9 pins. Connector P103 has 4 pins. This makes them easy to identify. Before you do this mod, inspect the female connector shells at P101 and P103. If these shells are streaked with brown, you are seeing evidence of overheated pins and burning of the plastic! If this is the case, you **should** perform this modification. If you don't, you **will** have problems later on! Refer to Figure 1: H-89 Power Supply Module - Exploded View for orientation details. The repair technique is inexpensive, fairly simple, and will be permanent and safe. You will need the following parts:

1. **Stranded** hookup wire in several colors: green, yellow, and red are suggested in an attempt to match the type of wire and gauge on the original. NOTE: Colors may vary according to that which is available. DO NOT use solid wire!
2. Molex connectors from Radio Shack, or equivalent source:

4-pin male	RS Part No. 274-224	$1.09
4-pin female	RS Part No. 274-234	1.09
9-pin male	RS Part No. 274-229	1.59
9-pin female	RS Part No. 274-239	1.59

NOTE: The Molex connectors come with their own pins and sockets. If you solder the wires **directly** to the power supply PC board, an alternative solution to the problem, you won't need the Molex parts. You **may** need the wire to slightly extend existing wiring. Read through the entire procedure before you decide which alternative to use.

The original P101 connector is wired as follows:

Pin 1	RED
Pin 2	RED
Pin 3	No Connection
Pin 4	YELLOW
Pin 5	YELLOW
Pin 6	GREEN
Pin 7	GREEN
Pin 8	GREEN/YELLOW
Pin 9	No Connection

The original P103 connector is wired as follows:

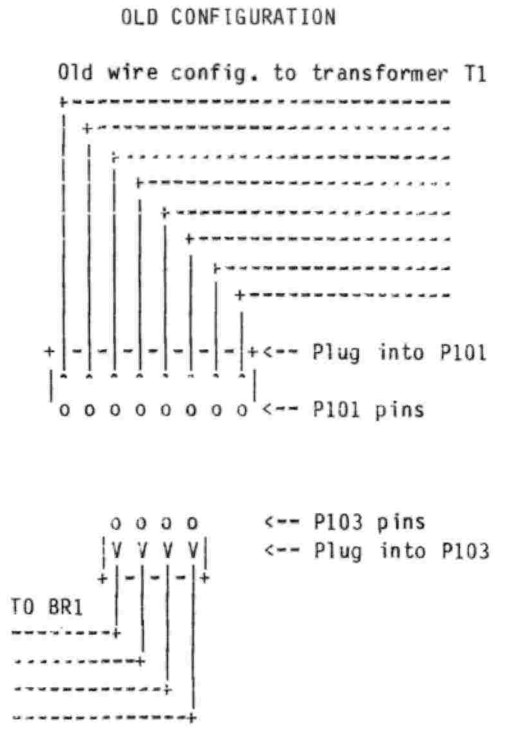
Pin 1	Longer ORANGE
Pin 2	BLACK
Pin 3	Shorter ORANGE
Pin 4	RED

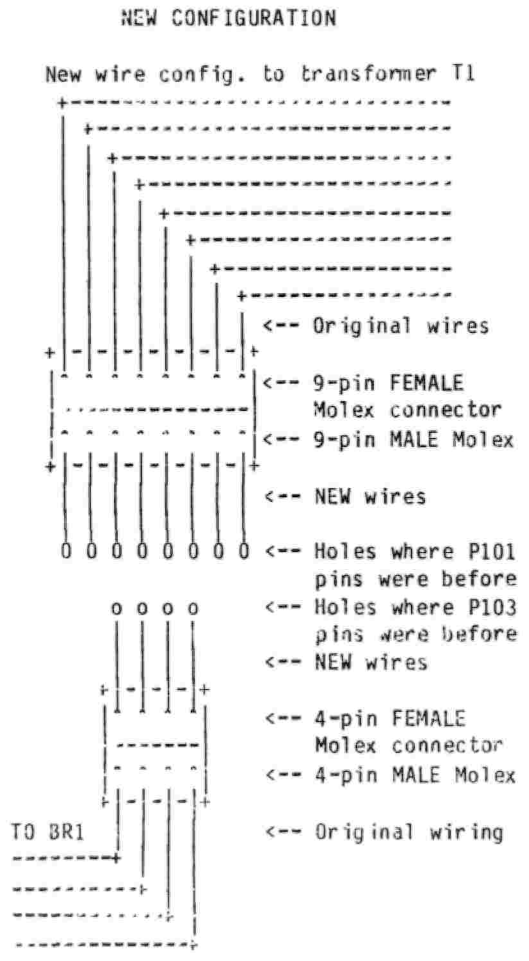
First alternative: Take one connector at a time and clip the wires off as close as possible to the old connector block. Strip and trim the ends of the leads and solder them to the MALE 4-pin Molex connector block to the cable coming from the bridge rectifier, P103. Connect a 4-pin FEMALE plug to 4 new wires into the vacated holes of P103 on the circuit board. Now strip and trim the ends of the leads and solder them to a FEMALE 9-pin Molex connector block to the seven wires coming from the power transformer, T1. Connect 7 wires to the MALE 9-pin Molex connector block, and solder these wires to the empty holes where P101 was formerly located. This order of "sex" connection is used so that there will be the lowest possibility of electrical shorts in the future. All connector blocks carrying "hot" wires have the relatively recessed female connectors (sockets), and the more passive wires have the exposed male connectors (pins).

Second alternative: Take one connector at a time and clip the wires off as close as possible to the old connector block. Strip and trim the ends of the leads. Carefully tilt the power supply PC board and **remove** the pins of the old connector header by applying your soldering iron to each pin underneath the board while pulling the same pin out with a pliers from the top. The plastic block will come away when you remove the last pin. Carefully clean the holes thus exposed with your favorite desolder aid, such as braid. Now individually insert the **original** stripped and trimmed wires you have already prepared into the holes from the **top** of the board and solder them to the foil on the underside. Be sure you insert and solder the proper wire to the correct hole!

Either alternative: It is **vital** to double-check, then triple-check to insure that all wires are properly restored in the same orientation as they were originally. Plug in the connectors if necessary, and **with power off** check the wires **again**! If you have made a wrong connection, you may

Transformer/Power Supply Wiring
First Alternative




NEW CONFIGURATION
New wire config. to transformer T1
<-- Original wires
<-- 9-pin FEMALE Molex connector
<-- 9-pin MALE Molex
<-- NEW wires
<-- Holes where P101 pins were before
<-- Holes where P103 pins were before
<-- NEW wires
<-- 4-pin FEMALE Molex connector
<-- 4-pin MALE Molex
<-- Original wiring
TO BR1

have to "kiss" your H-89 goodbye after the AC power goes on. So be v-e-r-y careful!

After you are ABSOLUTELY SURE that you have everything connected properly, check that the "OFF LINE" key on your keyboard is UP, and turn the AC power on. If you don't IMMEDIATELY hear the two beeps: one from the TLB board and one from the CPU board, you are in BIG TROUBLE! In this case, kill the power immediately, and look over your handiwork to determine what you did wrong. [See the wiring line-drawings on the preceding page.]

(3) Parts List. NOTE: The following list of parts is to be used in conjunction with Figure 1, The H-89 Power Supply. This figure is shown earlier.

KEY NO.	HEATH Part No.	QTY	DESCRIPTION	CIRCUIT COMPONENT # (Reference Design.)
Electrolytic Capacitors				
A1	25-197	3	1uf tantalum	C105,C106,C107
A2	25-891	1	470 uf	C104
A3	25-906	1	4700 uf	C102
A4	25-902	2	10,000 uf	C101, C103
Diodes				
B1	57-42	4	3A1 diode	D101 thru D104
B1.1	57-27	4	1N2071 diode	D109 thru D112
B2	57-67	1	10A20 bridge rect	BR101
B3	442-651	1	78H05 5-v regulator	U101
B4	442-30	1	UA309K 5-v regulator	U102
B5	442-650	1	78H12 12-v regulator	U103
Sockets, Connectors, Pins				
C1	434-117	3	Transistor socket	
C2	432-943	1	2-pin plug	
C3	432-974	1	2-hole connector shell	
C4	434-319	1	4-hole plug	
C5	432-1070	1	Large 4-hole connector shell	
C6	432-1069	1	4-pin plug	
C7	432-876	2	8-pin plug	
C8	432-1002	4	Large female socket pin	
Miscellaneous				
D1	73-80	1	Foam pad	
D1.1	85-2384-1	1	Bare circuit board for power supply	
D2	204-182	1	Capacitor support bracket	
D3	215-637	1	Power supply heat sink panel	
D4	215-658	4	Heat sink for regulators	
D4.1	354-10	1	11-inch nylon tie	

(4) Troubleshooting. The component parts have been assigned reference designations which will help you to determine their location in the computer. For example:

REFERENCE DESIGNATIONS	LOCATION
0-99	Parts mounted on the cabinet base or front panel
100-199	Parts mounted on the power supply module
200-299	Parts mounted on the video circuit board
300-399	Parts mounted on the keyboard circuit board
400-499	Parts mounted on the TLB (terminal logic circuit board)
500-599	Parts mounted on the CPU logic board
600-699	Parts mounted on the serial interface circuit board
700-799	Parts mounted on the cassette interface circuit board
800-899	Parts mounted on the hard-sectored controller board
900-999	Parts mounted on the video driver circuit board

[The troubleshooting guide for this article is on the facing page. Be careful while checking voltages, particularly that for the CRT anode. The latter requires a special, high-voltage voltmeter. Further, careless contact with the CRT anode connector or any bare wire on or from the flyback transformer could be deadly! -Ed.]

=====

This 'n' That

by Hank Lotz / 2024 Sampson St. / Pgh, PA 15221

My column title usually signals a potpourri of topics. This time I have only two items. Maybe we can think of one as "this" and the other as "that!"

How About Quick-Print?: As its fans know, Magic Wand has two sections, EDIT and PRINT. EDIT creates and edits text files for you. PRINT outputs your file to a printer, **or** as a **formatted** disk file.

However, you **can** do limited printing **directly from within EDIT.** That feature is called "**Quick-Print.**" It's nothing fancy, mind you. In fact, after a few tries at it myself, I abandoned it! Quick-Print was too, shall we say, "quirky."

But that was years ago. **Now** I use it, and it's great! What fixed it? Nothing. I just **tried it again** recently, only to discover I'd simply misunderstood Quick-Print those other times. I think I'm to be excused though, because Wand's (otherwise fine) documentation was scanty when it came to Quick-Print (Supplemental Manual to Version 1.1). It didn't detail the problem I experienced. That's one of the things I'll cover here.

What I'm saying today, then, is, "Hey, if you're avoiding Quick-Print because it seemingly 'acts up' on you, chin up!"

But I'm saying something else, too. If you haven't been using Quick-Print, you may have forgotten you even **have** it! If you need a nice quick way to print BASIC listings (or anything) with room to punch holes (for a 3-ring binder), dust it off and put it to work! Q-P is good for **many** print jobs, where you want control over all four margins without a lot of fuss.

Another use is, if you want a permanent record of **how you did something with embedded commands** in Magic Wand, Quick-Print's printout will show any embedded commands you have in the file. That's because Quick-Print does not **execute** the

Troubleshooting Table for "Troubleshooting the '89, Pt. 3"

PROBLEM	POSSIBLE CAUSE
Nothing happens at turn on	1. Unit not plugged in 2. Fuse F1 blown 3. Line cord pulled out 4. Switch S1 5. Fuseholder shorted/open 6. Power transformer T1 shot 7. Capacitor C1 shot
Fuse blows	1. Check primary wiring 2. Short circuit on power supply 3. Short circuited xfmr secondary 4. Diodes D101 thru D112, BR1 5. Capacitors C1, C102 thru C104 6. Microcircuits U401 thru U405 7. Shorted transistor Q204 8. Incorrect fuse 9. Power transformer T1
No output from 5-volt supplies; voltage is too high or too low	1. Microcircuits U401, U402 2. Diodes D105 thru D107 3. Capacitor C102
No +12 volts, or voltage is too high or too low	1. Microcircuit U403 2. Diodes D101 thru D104 3. Capacitor C102
-12 volts, or voltage is too high or too low	1. Microcircuit U404 2. Diodes D101 thru D104 3. Capacitor C104
No -5 volts, or voltage is too high or too low	1. Microcircuit U405 2. Microcircuit U404 (-12 volt source supplies, -5 volt regulator) 3. Diodes D101 thru D104 4. Capacitor C104
No +53 volts, or voltage is too high or too low	1. Transistors Q201, Q202, Q204 2. Diodes D201, D202
No unregulated voltages (+65, +8.5, + or - 16 on power supply board	1. Check appropriate secondary of T1, diode bridges, or filter capacitor
No anode voltage when other voltages are okay	1. No sync pulses coming from TLB 2. Transistors Q213, Q214 3. Deflection yoke 4. Coils L203, L204 5. Capacitors C228, C232 6. Diode D208 7. Microcircuits U201, U202
+500 volt supply is too high or too low	1. Diode D211 2. Capacitor C231
-90 volt supply is too high or too low	1. Diode D207 2. Resistor R259 3. Capacitor C229
-6 volt supply is too high or too low	1. Diode D203 2. Resistor R212

commands, as PRINT does. It just copies the screen as is. (When I didn't know that, I went to a lot of trouble, substituting command markers left and right trying to get PRINT to give a hardcopy that would show command markers. A lot of unnecessary effort.)

You access Quick-Print from EDIT's command screen. There are 5 commands: P, P!, PB, PB!, and P=. You always start with the "P=" command, which takes you to a small "Quick-Print Options" screen. Here you tell Q-P your paper size (in lines), and you set up margins and other parameters. You'll easily figure out that short menu. But notice there is no "Top Margin" spec. You must make your "Bottom Margin" large enough to suffice for both bottom **and** top, and then adjust the paper (Top Of Form) in your printer to wherever you want actual printing to start.

The Options screen queries you for a "Left Margin," but there is no "Right Margin" on the menu. It's important to realize that you already control the right margin **from the command screen** simply by using the L command. (L60 permits lines up to 60 chars long, L75 allows 75, etc.) To this extent, Q-P is a WYSIWYG function.

"Single Sheet" on the menu should be set to "N" if you use continuous paper, as most of us do.

After setting the options (the "P=" command), you return to EDIT's command screen. All of the other four Q-P commands are to send text to the printer.

Now you're ready to print. Let's say you have 66-line paper and you've set Quick-Print for a bottom margin of 10 lines. Just for simplicity, let's say you'll start printing at the very top of the page. Under these conditions you can print 56 lines on each page.

But let's suppose you have a total of only 50 lines of text currently in your EDIT workspace -- just over 2 screens full. (Quick-Print takes each line **on the screen** as 1 line of output, regardless of whether or not it ends with a carriage return.)

You enter the P command and hit RETURN, and your printer springs to life! At the end of the 50th line it stops, because that's all you have. (We're getting to the part where I made my big mistake!)

Now you decide to do additional editing on the text. Having finished that, you want a new printout. You reach over to the printer and hit its formfeed button, ejecting to a new page. Then you type the P command again. Once more your printer springs into action, but this time it prints **only 6 lines** and ejects to the top of the **next** sheet! It then prints the remainder of your file, but you have an awful gap on that previous page, don't you? Well **that's** essentially what happened to me.

But I can **explain** it now: Q-P keeps a line counter as it prints. It won't formfeed to a new page until it has reached 56 lines (in this case). Quick-Print didn't know about our local formfeed (**at the printer**). Its line count was still at 50, from our previous printout. **That's** why it printed only 6 lines on the new page: It did a formfeed at what it thought was 56.

The false impression I was under can be summed up this way: I thought each P command started fresh. But the **fact** is, Q-P remembers the line count from the previous P command. So just watch out if you do more than one printout in the same EDIT session. Or, to be safe, use the P! command.

The P! (P with exclamation point) **resets** the line counter. **But** it always does a formfeed before printing. My best advice, therefore, is go ahead and **let** Q-P do your formfeeds for you. If you don't, it won't reset its line counter at the **times you** want it reset. Use P! and you'll keep things straight; it's that simple.

The P command is fine for your first listing during an EDIT session. P is also very useful when you want additional printing to go to the same page. But whenever you want a fresh page use the P! command, not the panel switch on your printer.

An alternative to P! is to answer "Y" to the "Start New Page" option on the Quick-Print Options menu. The difference here is each printout will **always** start on a new page, even with the P command.

It's the same story for the PB and PB! command pair. Their **behavior** is the same, even though their **purpose** is to print only text that's **between block markers** ("Print Block"), whereas **the P and P! commands always print the entire text.**

Quick-Print is aptly named. It's a no-fuss, ready tool -- easy-to-use yet versatile enough to handle more job types than have probably occurred to us!

A Multicolumn Utility: Please excuse me for now putting in a plug for a program I wrote myself! I've done this kind of plugging before (for LP.COM for the H-14), but I think it's honorable to speak out for **MCOLS**, too, as it "saved my life" more than once. Because, you see, I simply **have no other way**, under CP/M, to create multiple columns out of a single-column file! (Other programs exist, but I don't have them.) I've been given jobs where I badly needed columns to save paper and repro costs. You could **do** it with Magic Wand, but it'd be very impractical --1 line at a time, and many manual steps per line. Too slow and tedious a job, and you'd be working without macros; there aren't many with Wand. Even if you set up Wand macros with KEYMAP (as I alluded to in #13, p.12 and #25, p. 6) their use is hardly recommendable for **this** task.

Kirk has printed short blurbs about MCOLS, including the Inserts of #8 and #9, but let me add a few details here. To do that I'm rehashing a piece I once put in a Pgh-HUG newsletter. (Speaking of Pgh-HUG, you may not know that club dissolved in Jan 1990 when we lost our meeting place.)

The name MCOLS means "Multiple COLumnS" because the program formats a single-column disk file into multiple columns. I wrote it in Microsoft BASIC (CP/M) and it should be easily adaptable to other BASICs. (I've done program conversion between BASICs, and it's no big deal.) The MCOLS listing is documented. It should be possible to put it into a version for **HDOS.**

MCOLS **does not alter your original file,** it just reads it and prints it out in a number of columns. The output device is menu-selectable, and output **can** go to a disk file. The resulting disk file may be edited with a text editor or word processor.

One good application for MCOLS is to list long assembly-language sources that otherwise would go down the left side of the pages, wasting a lot of white space to the right. Doing listings in columns saves paper, and that also means lots less page-turning. And it doesn't have to be an assembly listing. Other lengthy listings of relatively short lines will also be improved. For example, one guy who approached me was thinking about running a word-frequency analysis on a text. This outputs a very long list of **single words** -- a **real** "white space generator"! But with MCOLS he should easily get 6 columns across the page. The program is configured to allow you anything up to 13 columns, but can handle even more if you change the source code. Your paper width is your limitation.

You don't have to start your listing with the first record (line) in your file, either. You can tell MCOLS where to start the output. This feature has obvious advantages; but one is, you can break up a **long** printout into **more than one session**.

The MBASIC program-listing of MCOLS is bulky, but size isn't a drawback because the program works so well. And anyway, there are a lot of REM statements that can be dropped (from the copy you run). Those REMs list and describe every variable in the program --again, helpful when converting to other BASICs. I've written a thorough documentation (6 pages) describing the various features and how to use them.

But this doesn't mean you can't use it without the instructions. The program has on-screen prompting to guide you as you go. And after you tell MCOLS how many columns you want, how many lines per page, how many spaces **between** columns, starting record number, etc., you have the chance to change any of these parameters before actually doing the output.

MCOLS can report to you the **number** of records in your input file, as well as the **lengths** of the **longest** and **shortest** records in that file. (Sometimes I run the program with no output just to get these stats alone!) Also, MCOLS can tell you if your file contains TAB characters. If TABs **are** present it expands them to standard columns. (The record lengths that MCOLS reports are, conveniently, the lengths **after** TAB expansion.) Also, **before** you print, MCOLS can report how many pages of output there will be.

Naturally, the more columns you want, the narrower each one must be. So you will be warned **ahead of time** if you choose **too many** columns to accommodate the longest record in your file.

When one sees a layout of columns on a page, it is customary to read **down** the leftmost col, then read **down** the next col, etc. However, with MCOLS it is **also** possible to arrange the sequence of your records **across** columns, from **left to right**, if you desire.

The MCOLS program is in the public domain, and is being distributed by our editor, Kirk Thompson. There is also a compiled version for CP/M. Kirk is continually building **Staunch**'s "catalog" of software, both public-domain **and** commercial, for our 8-bit machines. (He has even threatened to publish an actual physical catalog some day!) As you know, much of Kirk's software is gratis except for shipping and handling.

BEAT THE NEW YEAR'S DAY DEADLINE

RENEW NOW BEFORE THE RATE GOES UP!
Send $12 if you live in the U.S. or Canada,
Overseas, please send $16 in U.S. funds.
ON NEW YEAR'S DAY, THE RATE BECOMES $15/YR.
U.S. AND CANADA, $19/YR. OVERSEAS.
!So send a check before you forget!

MISCELLANY

The Bookshelf. A number of computer books have crossed my desk over the last several months that you might find of interest. I only have room in this issue to discuss three of them. Two are of these are devoted to the COBOL language and are still available. The third is the one on troubleshooting recommended by Safford Magee in the last issue (p. 4).

COBOL. You should already be aware of my own interest in this, the oldest of high-level languages, from my discussion in issue #20/21 (p. 5). While searching for novice materials to supplement the old tutorial in **REMark** by H.W. Bauman and Heath's disappointing home-study course (EC-1105), I stumbled across two books that are still in print, yet applicable! One of these is Ruth Ashley's **Structured COBOL: A Self-Teaching Guide** (John Wiley & Sons, 1980, softcover).

This is a tutorial in the finest sense of the word. Ashley presents you with some material, then quizzes you on it, and at the end of each chapter has a "self-test" on the contents of the chapter. Space is proved for your answers, even reproductions of the conventional COBOL coding form. And the correct answers are no more than a page away from each quiz. She even begins by emphasizing the importance of "structured" programming techniques before she delves into the elements of the language.

Covered are unit-record files, arithmetic, conditions, sequential and random-access files, tables, qualified data names, REDEFINES, the case structure, and elementary use of the ACCEPT and DISPLAY verbs. Indeed, much is here that neither the **REMark** series nor Heath course even touch on. The version of COBOL discussed is even the one (without the custom enhancements Microsoft added) that was used in the two commercial compilers available for our systems. The only drawback the book has is that compiler use isn't covered until the final chapter. But I recommend it if you're interested in learning the language.

When I discussed resources for learning COBOL in issue #20/21, I mentioned the desirability of finding a key-word reference book to supplement your materials. The other in-print COBOL book I discovered satisfies that recommendation. This is Ruth Ashley and Judi Fernandez' **COBOL Wizard: A Wiley Programmer's Reference** (John Wiley & Sons, 1987, spiral-bound). This language reference uses the 1985 COBOL standard for its discussion, but the differences between that and the late 70's standard aren't that great. Further, the authors specifically note where the '85 standard differs from the earlier one and include an appendix summarizing those differences. And unlike the Sordillo book I suggested in that earlier issue, this one is organized mainly by the DIVISIONs in a COBOL program. Example code, both short and extended, are given throughout.

I might add parenthetically that I have always found any of the computer books by Ashley and/or

Fernandez to be excellent investments. Regrettably, many of them are now out of print, so recourse must be made to the used bookstore.

TROUBLESHOOTING. You undoubtedly recall Safford Magee's letter in #25 recommending Robert Paynter's **Microcomputer Operation, Troubleshooting, and Repair** (Prentice-Hall, 1986). As I mentioned there, I ordered it through a local bookstore and it turned out to be the most expensive book I have ever bought! (Computer books, even soft-cover, are costly anyway.) Current price is $57.00, but it's a large-format (8-1/2 x 11) book and is profusely illustrated with line drawings, photos, and circuit diagrams. When he wrote it, Paynter was a technical school instructor with strong connections to Heath.

The book is divided into four sections: computer fundamentals (including number systems), electronics fundamentals, digital circuitry, and microcomputers and peripherals. Indeed, it merges technical information from books such as Beechhold's and Middleton's (both mentioned in #24, p. 8) and appears to have been the text for a technical school course in computers. Once out of elementary material in the first nine chapters, Paynter divides his time between Motorola's 6800 CPU (exemplified by Heath's ET-3400 microcomputer trainer) and Zilog's Z80 (in the '89A). The former is used to introduce CPU architecture, programming, and hardware interfacing.

After this preliminary material, Paynter turns to the '89A as an example of a production microcomputer. He discusses, in turn, the keyboard and its I/O; the CRT, character generators, and controllers and their troubleshooting; serial I/O with the '89A's board as an example; printers, using the H-14 as representative (he even spends some time covering the workings of the beast!); and a sketchy introduction to floppy drives. Paynter concludes with a chapter on troubleshooting and analysis and another introducing 16-bit chips, using Motorola's 68000 as an example.

One thing I had better mention is that you **cannot** simply turn to the few chapters on the '89A and expect to find yourself in familiar territory. To understand Paynter's discussion of the '89A, you must either have already digested earlier portions of the book or have a preexisting, thorough grounding in digital electronics. Like the Middleton book discussed in the introduction to hardware troubleshooting in issue #24, Paynter's book functions at a level considerably "grittier" than the troubleshooting articles you'll encounter on these pages. However, I **can** recommend it as a replacement for Middleton simply because some of its discussion is specific to the '89A. The only remaining hurdle is the price, so you might first check your used bookstore for a copy.

Dual-Format Soft-Sector Diskettes. Bill Lindley's discussion of dual-format soft-sector in the last issue (p. 1) prompted me to experiment with the hard-sector master furnished by Charles Horn (and **finally** listed in this issue) and ZUG's (formerly, HUG's) old MAN37 utility from its #885-1217. For those of you who are unfamiliar with this ZUG disk, it contains a pair of CP/M utilities for duplicating hard- (H-17) and soft-sector (H-37) disks that will accurately copy **both** CP/M and HDOS media. But, as I noted, it also includes MAN37, a utility for preparing single-density, single-sided **soft**-sector disks from **hard**-sector originals. Price from ZUG (P.O. Box 217 / Benton Harbor, MI 49022-0217) is $20 plus $2 shipping; if ordering on **soft**-sector, append "-37" to the part number. The title of the disk is "HUG Disk Duplication Utilities."

My thought was that this would be a nifty way of preparing dual-format soft-sector disks from Charles Horn's hard-sector original! Alas, it has a limitation. More to the point, the HDOS soft-sector **device driver** you are running effects whether or not you can use the disk so prepared. A soft-sector disk produced by this conversion is **unreadable** under HDOS 2.0 **if** you're running Extended Technology's driver (as I explained I'm doing in issue #3). However, it works fine when used with CP/M 2.2.03 or 2.2.04, HDOS 2.0 (with either Heath's standard driver or ZUG's replacement on #885-1127), and HDOS 3.0x.

Further, a hard-sector disk INITialized under HDOS 2.0 that is subsequently converted to soft- with MAN37 **is** readable by Extended Technology's driver. So whatever "problem" there is with Charles's **HDOS 1.6** original must be that ET's driver either can't find the directory files or can't digest the boot track. Given the character of the access noises from the drive while the system attempts to mount the disk and that HDOS 1.6 **predates** the soft-sector controller, I think the former is likely. Either way, if you order the **soft**-sector version of the dual-format disk listed earlier in this issue, you should be aware of this limitation.

THE STAUNCH 8/89'er, created by Hank Lotz, is a bimonthly newsletter on 8-bit H/Z computers. The editor is Kirk L. Thompson; P.O. Box 548; West Branch, IA 52358; home: 319/643-7136. Subscriptions always start and end with the calendar year. Rate: $12.00/year. (Overseas, add $4.) Single copies: $2. Make checks payable to "Kirk L. Thompson". **Staunch** pays authors for their articles; write for an author's guide. It also accepts commercial ads for a modest fee; contact the editor. Neither this newsletter nor its editor is responsible for damages or losses resulting from use of any information presented herein. Info from **THE STAUNCH 8/89'er** may be reprinted only if this publication's name and address is included. Credit should also be given to authors and other sources of said material, if known. This publication is archived by the University of Iowa Libraries. CP/M is a registered trademark of Digital Research, Inc. **REMark** is a registered trademark of Zenith Users' Group. EOF