Issue #5 THE STAUNCH 8/89'er Oct-Nov-Dec 1987

PORT TO PORTAL —Editorial

There's one point that I've put into various flyers and other notices, but neglected to mention in all the previous pages of The STAUNCH 8/89'er. Reader mail has reminded me of it, so perhaps it's not quite so obvious; let's bring it up now: Have you ever wondered when your subscription will expire? With STAUNCH nothing could be easier — when the year ends, so does your subscription. In other words, **this** is your last issue, unless you renew for 1988 (or unless you've already done so). That's because all STAUNCH subscriptions are always aligned with the calendar years. For example, anyone subscribing anytime in 1987 receives the 4 issues of 1987 (actually 5 in '87 because Issue #1 from '86 was an extra "bonus"). A new subscriber who joins us **late** in the year still gets all previous issues for the year (plus any upcoming ones of course). Then when he renews, he is put on the list for all the 1988 issues. And **new** subscribers **anytime** in '88 will get the 4 issues for that year. It's easy for quarterly periodicals that don't appear on newsstands to work this way. Individuals who don't approve can always specify which issues they want. But, thus far, the few readers who **did** trouble to "specify", all asked us to start with Issue #1 anyway. There have been no complaints about this setup, and no one on our rolls has missed any back issues.

Now, on to a different subject. Last time in this column, some "new plans" were mentioned. As I promised then, here comes the unveiling: For one thing, The STAUNCH 8/89'er will now be **paying** for submitted articles! We'll pay **cash,** and **upon acceptance** — no delaying of payment until publication. Also, STAUNCH will now accept paid advertising. **Additional pages** will be provided for any ads placed, so **the ads will NOT reduce the amount of newsletter content** you are already accustomed to. Those are the headline items. For **details and discussion** of all of this, read Kirk Thompson's "8-BIT IOWAN" column in this issue!

From a reader's standpoint, other changes will seem more casual: Kirk Thompson will be assuming the functions of STAUNCH editor for the 1988 season — not that I'm going anywhere, mind you. I'll be around, to contribute the occasional comments and articles. However, starting now, all RENEWALS and new subscriptions should be **addressed to Kirk.** For Kirk's address see the back page. And all **checks** should be made payable to "Kirk Thompson". (If you have **already** sent yours my way, that will be okay.) All **article submissions,** too, should be sent to Kirk's address.

We're trying to get this out in time to remind you that, after December 1, renewal rates increase to $8/year. There may still be time to beat that!

—Hank Lotz

*** BE A STAUNCH SUPPORTER! ***
Subscription Renewals for 1988 now being accepted!
RENEW by DEC. 1 at the same $5.00 rate.
(Rate will be $8/year thereafter.)
Make checks payable to "Kirk Thompson".

THE EIGHT-BIT R/W —Letters

Pitch-changing on the H-25

[From James M. Frank, 1761 King George Dr., Kissimmee, FL 32743]... I have heard of a CP/M device driver for the H-25 [printer] but have lost or misplaced the information. I want to be able to change the characters per inch on the H-25. Any help?

More on the Z-89-11 Interface

[From Bob McClure, Jr., Cumby, TX]... Regarding the first of the "Questions and Answers" in Issue #4 [p.8] about the Z-89-11 interface. I believe that board uses a 2661 in the serial section. It is functionally similar to the 8250, being baud-rate-programmable, but is not software-compatible with the 8250. The Z-89-11 was evidence of Heath/Zenith's movement towards replacement of the 8250 with the 2661 which is the only type of UART in the H/Z-100 (not PC) series. The 8250 came back when they went into the clone business. Not having worked with the 2661, I don't know what virtues (if any) it has over the 8250. They did such a good job with the interrupt-driven BIOS interface in the Z-100, I've never needed to program any deeper than that.

The 8251 saw service only in the H-8, namely the H8-5 serial I/O-cassette card and the H8-2 parallel card. Yes, I said parallel. The 8251 drove a 6402 UART to convert the serial bit stream back to parallel. It's a little klugey but it allowed them to drive any device with the same code, changing only the port number. I sure wish they had continued supporting the 8251. I had to write my own HDOS driver for my parallel printer, and modify three CP/M BIOS's. [Thanks much for this info, Bob. -Ed.]

Patch for SD.COM? Just Ask!

[From William S. Derby, P.O. Box 2041, Livermore, CA 94550]... Thank you for your kind remarks about my programs in STAUNCH #4 [p.6]... Regarding your preference for left-adjusted names, I came up with a 3-byte patch you may want to make in your copy of SD. The first two bytes of the patch cause the name to be left-adjusted, and the third eliminates the period if this is desired. This still leaves the extension right-adjusted, as this is more firmly embedded in the design (to allow for ordering by extension). The patch can be made with DDT/SAVE as follows:
[User types the bold-faced portion.]

```
A>DDT SD.COM
-S0429
0429 20 00
042A C2 . (period)
-S0444
0444 20 00
0445 CA .
-S059B
059B 2E 20
059C 3A ↑C  (CTRL C)
A>SAVE 8 SD.COM
```

I do not plan to advertise this patch myself, but I have no objections if you would like to pass it on to your readers. A similar patch to byte 0122 in CMP.COM (normally 05) can be made to cause it to display less than five differences per line.

In addition to my utility programs, I have nearly finished the development of a replacement Command Line Editor for CP/M on the H-89. This increases the size of the BIOS by 1/2K to 1K; it is incorporated into the BIOS (and the BDOS). The program allows instant recall of the most recent command line, and of one chosen by the user; and full-line editing of every line is allowed before it is sent to the system. It is written in ASM and it does not usurp any characters except those already claimed by CP/M. It will take some time to complete the documentation, as all this has to be done in my "spare" time; but I hope to finish before the end of the year... [On page 6 of the last issue I said I wished the output was left-justified. That's all I did, and this patch was in my mailbox. It took no time to install and works just like Bill said. Talk about <u>support</u>! -Ed.]

Microflash M-89 Flashback
[From Ron Pannatoni, 120 Mark Dowdle Road, Franklin, NC 28734, (704) 524-7623]... I recently acquired an expansion box for the H-89 secondhand. It is called an M-89 and was made by a defunct company called Microflash. I would like to get in touch with the proprietor, Mr. Kan Cheng, to get a little help in bringing the unit up. His former telephone number (in Skokie, IL) has been disconnected. Can any of your readers help me contact him?

The unit that I bought may not be complete. The assembly manual refers briefly to an "H-89-1 decoder interface" which is not included with this unit. For this reason, among others, I would be very grateful to hear from any reader who has used an M-89 with his or her system.

THE 8-BIT IOWAN
by Kirk L. Thompson

As you've gleaned from Hank's editorial, this is a time of change for **STAUNCH.** He and I are swapping places for the coming year and I have a number of things I would like to try, to improve the quality of information we deliver to you. But I'm going to put off describing those till later because there are five short courses I wish to savor before turning to the meat of this issue's column.

4 MHz Mod Update. As an update to my remarks, last time, on one of the speedups for the '89, I was at HUGCON this August and talked with Darrell Pelan of Micronics. Motivated by William Clarkson's **Sextant** article, he has made **significant** *improvements* to his software. Our chat was fruitful because it cleared up my lingering reservations about Micronics's product. I have greater confidence, now, in recommending it to you as the cheapest way to speed up your machine's productivity.

OMAHUG Library Access. I also have an update to the information I included last issue on access to the Omaha Heath/Zenith Users' Group disk libraries. Things have changed more than I anticipated, however! The club's executive board decided to **require** membership. This costs $18 per year. Although library access is free to members, if you must obtain library disks by mail, the club will charge you the cost of disks (unless you supply your own) and a sliding fee for postage. For more details, write to:

Treasurer / OMAHUG / Box 777 / Bellevue, NE 68005

And I most humbly apologize to those of you who wrote to OMAHUG based on the erroneous information I supplied! I hope you weren't inconvenienced too much!

Saving HDOS. While at HUGCON, one of the contacts I made was Robert Todd, distribution coordinator for the **SIG/M** public-domain CP/M library. One of the things he discussed in his seminar on public-domain software, and which we reviewed privately, was the sorry state of HDOS. He proposed that **SIG/M could include the HDOS system** and those utilities and applications which were developed **in-house** by Zenith Data Systems, Heath, and HUG, in its library. Observe that this does **not** include material which is already in the public domain or is licensed!

But to do this, we have to persuade ZDS, Heath, and HUG. I wrote them with that in mind in September. And I urge you to help "save" HDOS by writing to all three of the following gentlemen:

Mr. John Frank / President, Zenith Data Systems / 1000 Milwaukee Ave. / Glenview, IL 60025
Mr. William E. Johnson / President, Heath Co. / Benton Harbor, MI 49022
Mr. Jim Buszkiewicz / Manager, Heath/Zenith Users' Group / Box 217 / Benton Harbor, MI 49022-0217

Tell them, first, that since the demand for HDOS has all but dried up, there is no longer any reason for them to cling to their (obvious) rights to it. Second, mention that HDOS is also of historical interest; it was the most advanced microcomputer operating system of its time and some of those advanced features have only been **added** to MSDOS in the last few years (since J. Gordon Letwin joined Microsoft)! If an H-8 is worth preserving at the Smithsonian (donated by Heath several years ago), certainly HDOS is worth saving. Third, tell them that preserving HDOS will demonstrate to the user community that ZDS, Heath, and HUG won't completely abandon older users as computer technology improves (as many of us presently feel). The goodwill of the user community is certainly a factor to consider in their business.

Finally, ask them to coordinate efforts to collect the remaining source and object code, place all of it in the public domain, and give it to SIG/M. In your letter, mention me if you like, but be sure to include Todd's name and address as follows:

Robert Todd, Distribution Coordinator / SIG/M / Box 97 / Iselin, NJ 08830

Twenty years ago, mail campaigns prolonged the life of television's STAR TREK; perhaps it can save HDOS.

You might include a note of thanks when you write to Buszkiewicz, too. HUG is one of the few vendors which has **not** put its HDOS library in mothballs. It is certainly due some plaudits for that.

I'm sure saving HDOS will take more than my initial shove! To keep the pressure on, you will have to do something, too! **So write right now!**

Eight-bit Sales. A number of vendors are clearing their shelves of 8-bit software. Four I've bought from are:

Ms. Pat Diehl / Kres Engineering / Box 1268 / La Canada, CA 91011
Brad Gjerding / Magnolia Microsystems, Inc. / 2818 Thorndyke Ave. W. / Seattle, WA 98199
Lee Schumacher / Schumacher Associates, Inc. / 12619 Valleywood Drive / Woodbridge, VA 22192
Ray Massa / Studio Computers, Inc / 999 S. Adams / Birmingham, MI 48011

Write to the first three for lists of what they have; include a self-addressed, stamped envelope (SASE) in your query to Schumacher Associates. In the case of the last (Ray Massa), send me an SASE (my address is at the end of this issue) and I'll mail you a two-page list of HDOS and CP/M materials. (The story about **that** is too long to go into, here, and all I'm providing is the list!) But when you order from any of these vendors, be sure to add $2 or $3 to cover shipping.

Public-Domain Software. If you are using Hoyle and Hoyle's **QUERY!3** database package, you may be interested in knowing that there is another public-domain utility for it besides my **TXT2Q3** (described by Hank, Issue #3, p.1). Hank recently finished one himself, called **QDELETE**, which flags all records in a database as "DELETED".

He has sent it to me for general distribution. If you would like a copy, send me a **formatted** 48-TPI diskette, either hard- or soft-sector, an SAS return mailer, and $2.00. If you don't want to hassle with the disk and mailer, the cost is $6; just let me know the format you need. The CP/M disk includes a compiled version of the program, with documentation and BASIC source code, **and** a compiled version of **TXT2Q3** for hard-sector. The HDOS disk contains only the MBASIC-interpreted program and documentation.

Fellow **STAUNCH** subscriber Terry Hall is also sending me disks from the Frazer Users' Group HDOS library. So far, I've received 11 of the approximately 30 available. If you would like copies, send me a box of 11 48-TPI, formatted disks and $22. Again, if you don't want the hassle, I'll ship you the set, postpaid, for $66 for hard-sector, $33 for soft-sector. Just be sure you specify the format you need. (Hopefully, this will put me back in your good graces after my muff with OMAHUG!) And I'll keep you posted on further acquisitions.

Next Year. Finally, I arrive at what you all have been waiting for, my plans for 1988 as editor. These fall into four areas. But I would like to comment, first, on **STAUNCH**'s more distant future to give you a picture of where I think this newsletter could go. Then I'll return to my immediate plans.

WHAT DREAMS ARE MADE OF. I think all of us would like to see **STAUNCH** grow. I can easily picture it as a bimonthly, running to some 24 pages. For example, a subscriber base of 3,000 (a not-unreasonable figure) at a subscription rate of, say, $12/year, would make it self-sufficient. But a publication operation that size **also** requires a full-time staff just to get it out, and they would have to be paid wages!

At the moment, **STAUNCH** is surviving only because Hank Lotz and I believe in providing a communications medium for you. It's hardly a money-making proposition. And the staff certainly isn't getting any pay!

Another factor to consider when growing, is maintaining quality. Hank and I have labored hard to keep these pages literate and technically accurate. We feel you should expect (and indeed would accept) nothing less! But expanding either the size of each issue or the frequency of publication requires more **writers** to provide that all-important editorial content. We do **not** want to become the **BYTE** of 8-bit newsletters — a few articles scattered amongst a lot of ads!

We do plan to grow, however. But the rate of that growth is directly tied to the number of subscribers we have. I presently feel that when we have reached 500 subscribers, we can then increase the number of pages in each issue over what I already have planned for the coming year. Increasing the frequency of publication from quarterly to bimonthly will require at least 1,000. These are long-term goals which I have set for myself and which I believe we can achieve.

But **STAUNCH** also needs your assistance to grow. Indeed, you are our best asset! You know the quality that goes into this newsletter. If you want to get the most out of your micro, you also have an interest in seeing us grow. So tell your friends (even enemies!) about us!

FORMAT AND FREQUENCY. What these two factors (subscriber base and editorial quality) mean for 1988 is that **STAUNCH**'s basic 8-page format (with one exception I'll discuss, below) and quarterly period of publication won't change. In fact, you will see few **internal** changes either, in the new year. The reason for that is because Hank's original design, a year ago, was so good, there is little I could improve on! And Hank should be thanked for his foresight.

PAID COMMERCIAL ADVERTISING. In one area of growth, though, I will aim for a two- or four-page advertising insert to **supplement** each issue. One thing **STAUNCH** could use is an alternate way of telling you what new hardware and commercial software are available to you, a task usually performed by ads. **SEBHC Journal** is already doing this by supplying free ad space.

But in **STAUNCH**, commercial advertisers will be charged a small fee, $50 per 8-1/2 x 11 page. Half-pages will also be available for half that price. Advertising should be submitted to me as camera-ready copy in the actual-sized format (full- or half-page) desired. The reason for the charge is **not** to make money for the editor, however!

Instead, for example, you should see more advertising **by STAUNCH** in the media. Thus far, we've relied mostly on word of mouth. Unfortunately, paid advertising in **REMark** and **Sextant** isn't cheap and part of **STAUNCH**'s advertising revenues would be used for that purpose.

PAID AUTHORS. The rest of our income from any paid advertising will be used to pay authors! I would like to bring more viewpoints and ideas into these pages, so I'll pay you $50 for every 1,000- to 2,000-word effort. There are a lot of things Hank and I know nothing about, but you do. Now you can tell others your experiences and even make some money doing it.

Moreover, you will be paid when an article is accepted. One of my (personal) aggravations with **REMark** and **Sextant** is that they pay only **after** your article has appeared. None of that with **STAUNCH**! You will get your check when I accept your work, *even if publication is delayed several issues.* So if you are interested in writing for us, send me an SASE and I will mail you an author's guide with details.

And I have a topic on my immediate want list, too. That's a **series** of articles on the ins and outs of HDOS 3.0. Regrettably, Bill Parrott, the new system's designer, is so tied up with other things that he doesn't have the time to write for **STAUNCH**. (I asked him at HUGCON!) So I can almost guarantee you space if you want to write, in depth, about it.

THIS COLUMN. So what happens to "THE 8-BIT IOWAN" in the new year? It will still be here, but in a reduced format. Its size will depend on how much other material I have to shoehorn into **STAUNCH**'s pages. And I will still emphasize ways of improving your system, although I will look more at software in the coming year.

So there you have it! **The STAUNCH 8/89'er** will continue as you've come to know and love her. There will be some changes, specifically in the advertising and paid author areas, but I am also trying to keep growth to a tightly controlled rate. I've heard of too many newsletters and magazines which bit off more than they could chew and folded, leaving their subscribers with a worthless cancelled check. **STAUNCH** is not going to do that.

But to grow any further than I have projected for 1988 will require a significant increase in the number of subscribers. And to meet my goals I also need your valuable assistance. So "be a **STAUNCH** supporter" and broadcast the word!

In the Queue. Next time, to conclude my major series on hardware, I'll look at RAMdrives. There are several makes on the market for the '89, but my emphasis will be on how they improve system performance. See you then!

ZERO-LENGTH CP/M FILES: Creating and Using Them

by Hank Lotz

This is beginner-level but **no one** should skip the last little paragraph! The reference there **may** have slipped past **a few** old timers, and should be consulted.

Today's article is a reworking of a piece I had in the Nov 1984 PITTSBURGH HUG NEWSLETTER. It deals with a *kind of disk file, a file that doesn't* **actually** exist, except as a **name** entered in the disk's **directory**. You might call it a "null file" but that has added connotations. I call it a zero-length file. What's it for? Well for one thing it can be an identifier "label" on a CP/M diskette, so you can tell from the screen what floppy is in a drive without removing the floppy. Along the same line, the Feb 1987 **SEBHC Journal** (p.17) gave a short assembly language disk-labeler program, with corrections in the March issue (p.3). I tried that and it's a valid way to write a nice disk label to the screen. My quick-and-dirty method performs a similar function, but instead of typing LABEL you use the DIR (or STAT) command. Also, my way takes **no program space** on the disk. You can display the "label" from the system prompt at any time. Your label will **head** a DIR (or STAT) list even when no *qualifiers are given with the DIR (or STAT) command.* To see **label only**, type: DIR *.LBL (or STAT *.LBL). That works with the auto-command line, too, to show the label at bootup. Finally, **programs** you write can fetch the "disk labels" to verify that the proper disk is installed before attempting I/O operations! All in all, simple but useful. Here's how it's done.

CP/M's SAVE command can create a directory entry (that is, a file name) that can serve as a visible disk label (pseudo-label if you like) because it will appear in directory listings. The CP/M SAVE command is: SAVE n filename,
where "n" 256-byte pages of memory (starting at address 0100H) get saved to the default disk under the name "filename". ("Filename" can contain a drive spec if default drive is not desired.) The **trick** is, we're going to let n=0: SAVE 0 filename,
so **nothing** gets saved! But the filename gets entered into the directory! The file thus named won't own any data tracks; it's a "zero-length" file. Only one entry (or "extent") in the directory is used. If the day comes when you need that directory space you can sacrifice the filename.

We aren't done yet. We want our label name to be **first** *in the DIRectory listing so it'll show up* top-left on the screen —a **consistent**, conspicuous position. To get it there, we must place the name **physically first** in the directory because DIR doesn't alphabetize. This is easy to do since the first name SAVEd onto a newly formatted disk will go into the directory's first slot by itself. (Get into the habit of adding labels to new disks first thing.) What if the disk already has files on it? A newly SAVEd label will **still** fall into the first directory position **if** that position holds the name of an ERAsed file. Check this by trying it and watching where it goes. Just type:

```
SAVE 0 LETRS/01.LBL    (to coin an example).
```

Then type DIR. If your label appears at the top of the leftmost column of file names your task is complete. If not, ERAse the misplaced label and do the following to "label" a disk: (All files should be backed up first.)

1) PIP the first-listed file given by DIR to a new location. If that new location is on the same diskette, use a new, temporary file name (e.g., KEEPTHIS.COM).
2) ERAse the **original** file that you just used PIP to copy.
3) Do the SAVE 0 labelname.
4) REName the PIPped file back (e.g., REN original=KEEPTHIS.COM). Or if in step 1

you moved the file to **another disk** PIP it back to the original disk now.

But we **also** want the label to appear first if a **STAT** command is used. The STAT program (in the form STAT *.*) lists filenames **alphabetically,** so choose the **first character** of your label from early in the ASCII table so STAT alphabetizes it to the top of the list. The exclamation point, (!), is the first visible ASCII character, but it can be confused with a 1, and just generally gets in the way of readability, so I use the double quotation mark, ("). This is the next-earliest ASCII character. I don't anticipate any (!)'s in my directory to preempt the ("). So a label on a BACKUP disk, say, might become "BACK/01.LBL where the quote (") is part of the name. Placed **first** it puts the label on top in utility sorts. BACK tells me it's a backup disk and /01 allows expansion to 99 backup disks. The consistent file type, LBL, says the file is just a label, and also lets me use DIR *.LBL to list only the label to the CRT. Consistent use of the .LBL extent also lets programs check for I/O-accessibility of a specific disk by verifying that the entry in the disk's "label" position is **indeed a label** before checking whether it is the **right** label.

Your own labels depend on your needs and imagination. I was once under the misconception that CP/M file names **had** to start with a letter (not a numeral) like a variable name in some languages. Not true; you can call a file 12345678.123 if you like!

A zero-length file, created using "SAVE 0 filename", has at least one other use besides "labels". I refer you to REMark, Issue 43, August 1983, p.31. In that article, George W. Stephenson, Jr. (a STAUNCH subscriber) describes a powerful device I use constantly. I advise naming the file /.COM. The slash is near RETURN so one hand suffices, but it's not so close as to be struck accidentally. Believe me the idea is **extremely valuable.** If any CP/M user hasn't heard of it, you'll want to start using /.COM.

FLIPPIES, BACKUPS, and TWEEZERS!

[Today's first "flippy" story was entered by L.W. "Bud" Cooke:]

They say it shouldn't be done, but I've been using both sides of my 5-1/4" hard-sectored disks for a few years now and so far have not experienced any problems. I make these "flippies" myself, and thus get twice the disk space on each disk. However, I do back up **all** my disks for safety, which means I have two copies of everything. So paradoxically I'm back to square one spacewise. But at least it's like getting free backup space.

The tools I use are: 1) A cardboard template that I fashioned myself; 2) A one-hole, hand-held, squeeze-type paper punch; 3) Two clothespins to hold the template; 4) A matchbook cover (less matches) that fits in at the center hole perfectly and protects the disk surface from the punch; and 5) A pair of tweezers (you'll find out why).

[Editor's Note: Steer clear of magnetized tweezers, etc. In fact, never keep any kind of magnet around your computer work area where you store disks. Bud proffers his address: 343 Fletcher St., Orange, CA 92665, PH (714) 637-2633. Next, we have (in a slightly abridged version) Anthony P. Musnick's method of preparing flippies:]

a) Carefully cut away as little of the trailing edge of the sleeve as will permit removal of the disk. I use a small photographic paper cutter. Remove the disk and set aside on a clean, flat surface. Don't touch with fingers -- use a soft tissue. b) Make a paper template of the location of the detection hole and alignment notch. c) Move the template to the opposite (upper) side of the sleeve and position it so that the holes are in precisely the same position. Insert the punch through the hole in the sleeve and punch a hole. d) Punch a semi-circular hole for the alignment notch. This works just as well as using an expensive notching punch. e) Reinsert the disk. f) Seal the edge with tape. g) Format the new side of the disk.

[Editor's note: Tony recommends using one side for executable files, the other for data, etc. Now I must brave the wrath of both these fine gentlemen by telling you why I **don't** use flippies. As Bud noted, the practice isn't recommended, but I feel that **even if it works,** it introduces another variable into the troubleshooting process if you DO experience hardware trouble, whatever the cause actually is! Also, if you insist on removing disks from sleeves, be careful what tape you seal them shut with. Sticky adhesives are extremely hazardous to the well-being of diskettes. I've even deleted a recommendation of a specific type of tape from the above for fear of ambiguity.]

Q minus A

You've heard of a **Q & A** column. Where do you think we get answers to all those questions? We don't! Some are unanswered. This, then, is an open-forum "Q-only" column! No A's here, just Q's. If you can answer these posers, send in your assistance!

Q — There is a system directive used by MBASIC ver. 4.8 which tells HDOS to load both system overlays before the boot drive (SY0:) is reset from within MBASIC. What is the value of this directive in hexadecimal or octal so I can find it with a dump program?

Q — It is my understanding that, in general, to prolong the life of electronic components, it is wise to provide a means of heat dissipation. Therefore, wouldn't it be advisable to put a cooling fan on my H-8?

Q — My D-G Super 89 board has a clock. How can I access the clock from a **program**?

Q — The H: prompt on my H-89 is slow to come up — the machine has to warm up first. What is wrong? [The editor replies: I "promised" no answers, but this is makeshift. My '89A did that too. I diagnosed a loose contact, if not its exact location, by pinching the 2 main boards (CPU and TLB) together **slightly** and **gently** at the top (don't get near the High Voltage) while hitting SHIFT-RESET. This gave a prompt (and beep) right away. Also, leaving the lid unlatched (closed, but not latched) seems to relieve strain somewhere; now my machine ain't broke enough to **risk** "fixing" it! This is certainly not very scientific, it may not work for you, and I'm not recommending too much flexing of circuit boards.]

Q — My H-89A and Epson FX-80 printer **set for 9600 baud** work fine with HDOS 2. When I use CP/M 2.2.03 (configured for 9600 baud) it will not print. If I configure for 4800 and correspondingly change the printer it works OK. Any idea why it won't work on 9600 with CP/M, while it will with HDOS?

Q — Has anyone ever solved the problem of the H-89 glitching disks left in a drive while powering down?

Q — Is there a modification to the H-14 printer so it can be used for graphics?

SOME INS and OUTS of COMPILED PROGRAMS

by Hank Lotz

Intro and Purpose. As you may have gathered from reading STAUNCH, I like to write my own programs when I can. Some of you H-8'ers and H-89'ers probably do the same. If you do, then rather than have those programs eat up too much memory for the tasks they perform, perhaps you'd feature **packed** code. (If not also feature-packed code.) There's a Z-150 member (not a subscriber to STAUNCH) in Pgh-HUG who, whenever we get on this subject, says he doesn't care about conserving memory because his PC has more than enough, and besides he's retired and can wait if a program is not real fast. That's understandable for him but my programs have to run in 64K. (And I've found 64K to be more than enough in my five years with it.) But "small" memory is not the only reason I aspire to compactness. No, aside from the 64K limitation, I'm fascinated by the sheer aesthetics, **including** the efficiency, of economical programming.

After you've read this you aren't going to be a genius at writing efficient code (unless you already are). My purpose is to point out something about the code generated by two specific compilers, and to pass along a tip on how a compiler's output can **sometimes** be improved "sizewise". This should be informative to some and interesting to the rest.

The Resources. Of course the real way to pack in the code is with assembly language. The trouble is, for most of us it takes too much time and effort to create an entire major program in assembly language. With a **compiler** you can generate a pretty swift program and **still** enjoy the relative ease of high-level programming. Don't throw out your assembler though, because that **compiler** output can be spruced up here and there with a dash of auxiliary assembly language. The compilers I'll talk about are from Microsoft: BASIC, version 5.35; and FORTRAN-80, version 3.4. Mine are both for CP/M. (That FORTRAN-80 version number was neither in the manual, nor on the screen. I browsed through the F80.COM file with a disk dump utility, and there it was floating in a sea of hexadecimal!) During this discussion I'll often use the term "MBASIC" when referring to the Microsoft BASIC **compiler,** and sometimes simply "FORTRAN" when I mean the Microsoft FORTRAN-80 compiler.

Sizing up the Resources. It has been my experience that the Microsoft FORTRAN-80 compiler generates far leaner code than the MBASIC compiler. I suspect it's not so much the compiler per se, as it is the relative largeness of the MBASIC relocatable library modules: Try programming the same task in FORTRAN and in MBASIC (almost **any** task), and then compare the sizes of the two COM files. Let me now offer three such "benchmark" comparisons, albeit very simple ones. These are trivial programs but they're useful **analytically.** In the following illustrations the MBASIC listings will always be on the left, the FORTRAN on the right. At the top of each listing are the sizes of the resulting **COM files** as would be reported by STAT.COM. A closer idea of the size **differences** appears in the commentary before each exhibit. Omitting the END statement in MBASIC meant only about 4 bytes difference, so don't mind my inconsistency there. "PROGRAM <name>" statements were used in all FORTRAN sources but are not shown here.

Listing 1 is our first pair of source files. A bare-bones program in compiled MBASIC (left) to print one single character ("H") to the screen gave a COM file about 2.5k bytes **larger** than the same thing in compiled FORTRAN (right). But in **both,** note the outlandish COM file sizes, for the precious little they do:

Listing 1

```
(COM file = 9k)           (COM file = 7k)
 1 PRINT "H"                WRITE(1,1)
                          1 FORMAT(" H")
                            END
```

Next, in **Listing 2,** is a program to do one simple arithmetic computation (in floating point) **with absolutely no I/O whatsoever.** [For the beginners who I know are out there "I/O" means "input/output".] It required a COM file some **4.5k larger in MBASIC** than in FORTRAN:

Listing 2

```
(COM file = 6k)           (COM file = 2k)
 1 A=3!                     A=3.
 2 B=7!                     B=7.
 3 C=A*B                    C=A*B
 4 END                      END
```

And **Listing 3** below shows a (useless) program to assign a value to a 2-byte integer variable (i.e., I=5), **but to do nothing more.** No I/O and no arithmetic between variables. (There **was** only the one variable.) Yet the COM file in MBASIC took up a costly 4k; the FORTRAN only 1k. The **actual** difference in size was slightly **less** than 3k bytes.

Listing 3

```
(COM file = 4k)           (COM file = 1k)
 1 I%=5                     INTEGER*2 I
                            I=5
                            END
```

Making a Choice. Okay, earlier I talked about improving these ghastly sizes by injecting **assembly language.** First we have to decide whether to do that to the MBASIC or the FORTRAN. What's your philosophy? Both can be improved, but MBASIC is bigger so maybe it **needs** the help more! On the other hand, FORTRAN can take **better advantage** of the help by yielding smaller programs after the fact! (If you're an "MBASIC-only family", the choice is vastly simplified!) I pick the FORTRAN route because we end up with a smaller COM file which was our original goal. However, I'll go through it with both languages.

Our "size troubles" loom largest back in Listing 1 (9k and 7k), so we'll operate on the programs in **Listing 1** only. Other assembly routines can be devised to take care of other needs. This is just an illustration of size-effectiveness. **Listing 4** is an **assembly language subroutine** called "DSPLAY" which we'll use instead of the WRITE statement in our Listing 1 FORTRAN program (or the PRINT in the MBASIC). If we substitute a call to DSPLAY into Listing 1 we come up with Listing 5 which replaces the Listing 1 programs. We can get away with this since there are no **variables** to print out. A different assembly subroutine would be needed for that. I expanded the "H" to be "HELLO" to make it easier to spot; the difference in memory is insignificant. The 13,10 does a CR and LF after the HELLO; the '$' is needed by CP/M's BDOS function 9 to mark the end of the message. Don't be afraid if you've never done this assembly stuff, I'll show the steps. If you like, type the subroutine in and name the disk file DSPLAY.MAC.

Listing 4

```
;DSPLAY.MAC -- SUBROUTINE FOR CRT MESSAGES
;
F9      EQU     9       ;CP/M function #9
BDOS    EQU     5
MESSG:  DB      'HELLO',13,10,'$'
;
        ENTRY   DSPLAY  ;Used by L80 for linking
DSPLAY: PUSH    PSW     ;Put the registers on stack
        PUSH    B
        PUSH    D
        PUSH    H
        LXI     D,MESSG ;Get set to print HELLO
        MVI     C,F9    ;This chooses function 9
        CALL    BDOS    ;This does the output
        POP     H       ;Put back the registers
        POP     D
        POP     B
        POP     PSW
        RET             ;Return to MAIN
        END
```

Combining High-level and Assembly Languages. We can now directly compare the programs in **Listing 5** with those in Listing 1.

Listing 5

```
(COM file = 4k)       (COM file = 1k)
 1 CALL DSPLAY         CALL DSPLAY
 2 END                 END
```

If you try both the FORTRAN and MBASIC examples concurrently, **avoid** naming one source MAIN.FOR and the other MAIN.BAS. Instead, use different names because FORTRAN and MBASIC **both** create <name>.REL, and **only one MAIN.REL can exist on a disk.** With that in mind let's call the sources "MAINF.FOR" and "MAINB.BAS". Again, the **sizes** in Listing 5 are for the COM files **after** the assembly language subroutine has been added! Quite a difference from the 9k and 7k of Listing 1! We replaced, in the FORTRAN, the WRITE and FORMAT statements. Of course if there had been other lines in Listing 1, we would not delete **them,** except that any other WRITE statements must be replaced to eliminate the library module and reap the benefits of reduced size. I'll mention other exceptions later.

Here's how to make the COM files. After you compile a MAIN program, its relocatable (REL) file will be on disk. Name the assembly source disk file DSPLAY.MAC and assemble the subroutine using the Microsoft MACRO-80 Assembler (M80). For example:

```
        A>M80 =DSPLAY
```

You now have REL files on disk for the main program and the subroutine. Link the program's relocatable to its subroutine using the L80 loader. This can be done interactively by typing "L80" <RETURN>, and then typing the commands and module names, one per line. But if all your sources are debugged, and you know what drives everything is on, you can do it all in one command line. In the case of our FORTRAN example:

```
        A>L80 MAINF/N,MAINF,DSPLAY,FORLIB/S/E
```

This yields the disk file MAINF.COM which can be run from disk like any other program. I can't show all variations of the above command line, but if you've compiled and loaded programs before, you can see what is happening. To load an MBASIC module (e.g. MAINB.REL), substitute MAINB for MAINF, and BASCOM for FORLIB. (In older MBASIC versions, BASCOM.REL is called BASLIB.REL.)

Earlier, when you compiled the FORTRAN file, MAINF.FOR, you should have typed:

```
        A>F80 =MAINF
```

This created MAINF.REL. Similarly, with the MBASIC compiler, you obtained MAINB.REL after you typed:

```
        A>BASCOM =MAINB/O
```

The letter "O" here tells the MBASIC compiler you will later want a self-contained COM file and will not want to use the RUNTIME module. The "O" switch is not needed if your MBASIC compiler is an older version w/o the RUNTIME module. (I'm withholding my opinion of RUNTIME modules! Or am I?)

Getting Practical. The DSPLAY example just scratches the surface to show how effective this "hybrid" technique is. In **many** applications other assembly routines can be created to do other I/O functions within a program. But some of them will be much more complex. Subroutine DSPLAY is only good for printing screen messages, without any variables. If you do printer output, or READ's, or WRITE's to **disk,** big module(s) (e.g., $W2) from the FORTRAN library will be linked in to service **those** calls, blowing your size advantage. So it won't do any good to use DSPLAY if you end up needing the big library module **elsewhere** in your program anyway. But whenever you **can** use DSPLAY, its output string is certainly not limited to "HELLO". You can substitute big menus, or voluminous instructions, and still shave almost 6k from FORTRAN programs.

But it is not **only** to the programmer whose first love is trimming bulky code that I address the following more general notion: FORTRAN is a lean, fast, powerful programming language. It is sometimes neglected, even by its fans, because it doesn't provide certain features desirable in microcomputer software. — What must not be overlooked is that **assembly language** can easily supply FORTRAN with the features it lacks, **and this particular "team effort" is dynamite.** The FORTRAN/Assembly-Language-Subroutine combination is a real winner!

Acknowledgment. My thanks to Allen Gilchrist, Jr., for his excellent article on p.47 of the June 1984 REMark. Its title is, ironically, "Assembly Language Subroutines With BASIC". But before I read it I simply did not know **how to use** the MACRO-80 Assembler that came with Microsoft FORTRAN-80!

MISCELLANY

Stamina Corner: Do you realize that in 1967 I built a Heathkit Model GR-295 (25") color TV which today still sees probably 4 to 7 hours of daily operation? I replaced the VHF tuner once and have gone through a few CRT's. Now if my H-89A lasts **that** long I should have no complaints! But the TV's most recent electronic component failure I remember was 3 years ago. [Would you believe, since I wrote this the TV conked out and I replaced a tube, two capacitors, and a 3-watt resistor! Not only that -- this morning, I had disk drive trouble on the H-89... Got it back up now though, but I'm not sure how. (Swapped two drives.)]

Computer Surplus Store: In response to Terry Hall's request for 8" disks (last issue), Herbert Mallicoat says he has bought both new and used disks from this store, and claims they are reasonable. The address is Computer Surplus Store / 715 Sycamore Drive / Milpitas, CA 95035 / (408) 434-0168 [What newsletter are you going to mention when you contact them??]

QUESTIONS and ANSWERS

Q — I want to redirect screen (or printer) output to a disk file from within a program. For example, I use ROOTS/M genealogy program (CP/M) which sends output to printer or CRT. I would like to be able to send it to a disk file so I can edit it with a word processor. [And from another reader:]

Q — I would like to be able to dump dBase II reports to disk (a feature of dBase III [sic] which is not available in CP/M).

A — Why not check into CompuMagic's "SCREEN" program. (See Issue #4, page 8, where SCREEN is listed as part of CompuMagic's 20-program Utility Package.) **If** ROOTS/M and SCREEN (or dBase II and SCREEN) can both fit into memory concurrently, you **may** just be home free. Note that SCREEN is not a **redirection** program per se, because it does not "redirect" screen output, it "echoes" it to the disk file. The info still appears on the screen as well. We do not believe SCREEN processes printer output.

Q — What does SASI stand for?

A — "Shugart Associates Standard Interface". It is for interfacing hard disks to computers. SASI can be thought of as a subset of the newer SCSI (pronounced "scuzzy") which stands for "Small Computer Systems Interface". SCSI, which is later and more general than SASI, is an 8-bit parallel interface that can control practically anything: printers, floppy disks, hard disks, etc. There is an article on page 183 of the January 24, 1985 issue of the periodical EDN which should prove to be interesting reading. Its title is "Use SCSI Devices for Multiprocessor, Smart-I/O Systems."

Q — What is an interrupt and what does it do?

A — A CPU executes a program one instruction at a time until it reaches the end of the program. But during that execution time it can be interrupted by (for example) a signal generated by an external device connected to a port. This signal is the **interrupt.** When this happens, the CPU stops what it is doing and saves all the current registers, pointers, and addresses, so that it can continue later where it leaves off. It then executes a service program referenced by the interrupting signal. After completion of this second program, the CPU restores the saved registers and resumes execution of the original program. This entire procedure is often so fast that a user at a terminal may not know it occurred. There are diverse applications of this feature because of its usefulness in satisfying a device which demands immediate service (for example, a measurement or control device in a manufacturing or testing process). There are also, however, **software interrupts** initiated by **programs** rather than hardware, to activate other programs.

THE STAUNCH 8/89'er, created by Hank Lotz, is a quarterly newsletter on 8-bit H/Z computers. The editor is Kirk L. Thompson; #6, West Branch Mob Hom Vil; Route 1; West Branch, IA 52358. Subscriptions always start and end with the calendar years. Rate: $8.00/year. (Overseas, add $2.) Single copies: $2. Make checks payable to "Kirk Thompson". 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. CP/M is a registered trademark of Digital Research, Inc. REMark is a registered trademark of Heath/Zenith Users' Group. -30-