$2.50

REMark®

Issue 43 • August 1983

Official magazine for users of HEATH ZENITH computer equipment.

NEW FROM FLOPPY DISK SERVICES, INC.

the H-89 TWOET SYSTEMS

We've got a great idea for your H-88, 89 or 90. It's a dual *internal* half height drive system. Two of our half height 5¼" drives can replace your built-in disk drive, doubling your information storage capacity.

Floppy Disk Services provides you with everything you need. That's two double-sided, double or single density, half height drives in either 48 or 96 tpi format, ail hardware, cables and power connector adaptors. And most important, you get easy, step-by-step instructions, in the Heath/Zenith tradition of good, clear documentation.

We've thoroughly tested the TWOET/Heath set-up. Remember that a double sided 48 tpi will work perfectly as a single sided drive right out of the box! Hard or soft sectored—so you can even use this system with your H-17 controller. And of course we have the software drivers (additional cost) to run 48 or 96 tpi double sided, single density drives on the H-17.

Model TWOET 455
2 Shugart SA-455 half height
48 tpi double sided
All hardware
Metal, shielded mounting plates
Data cable with chassis connector
Power 'Y' connector
Complete documentation
Price .... $595.00 complete

Model TWOET 465
2 Shugart SA-465 half height
96 tpi double sided
All hardware
Metal, shielded mounting plates
Data cable with chassis connector
Power 'Y' connector
Complete documentation
Price .... $695.00 complete

Wondering what to do with your internal drive if you buy this system? Here's the solution. If you purchase a dual half height system for your Heath computer from Floppy Disk Services, just include an extra $50.00 plus shipping and receive a single 5¼ case with power supply *and* data cable ready to receive your SIEMENS internal drive! The case with data cable is normally a $70.00 item. And the cable that comes with your TWOET system includes the external chassis disk I/O connector.

Due to production deadlines, prices in this ad are 2 months old, so we encourage you to call us for current prices and new product info. Prices and specs subject to change without notice.

Dealer inquiries invited.

PAYMENT POLICY — We accept MasterCard, VISA, personal checks and Money Orders. We reserve the right to wait 10 working days for personal checks to clear your bank before we ship. All shipping standard UPS rates plus shipping & handling. NJ residents must add 6% tax.

Toll Free Order Line:
(800) 223-0306
Tech Help or Info:
(609) 799-4440

H-88/89/90 are registered trademarks of Heath Corp.

FLOPPY DISK SERVICES™ INC.

741 Alexander Rd. Princeton, NJ 08540

Heath Users' Group

HUG Manager	Bob Ellerton
Software Engineer	Pat Swayne
HUG Bulletin Board and Software Developer	Terry Jensen
Software Coordinator	Nancy Strunk
HUG Secretary	Margaret Bacon
REMark Editor	Walt Gillespie
Assistant Editor	Donna Melland
Printers	Imperial Printing St. Joseph, MI

REMark is a HUG membership magazine published 12 times yearly. A subscription cannot be purchased sparately without membership. The following rates apply.

	U.S. Domestic	Canada & Mexico	International
Initial	$18	$20*	$28*
Renewal	$15	$17*	$22*

*U.S. Funds.

Membership in England, France, Germany, Belgium, Holland, Sweden and Switzerland is acquired through the local distributor at the prevailing rate.

Limited back issues are availble at $2.50 plus 10% handling and shipping. Check HUG Product List for availability of bound volumes of past issues. Requests for magazines mailed to foreign countries should specify mailing method and appropriate added cost.

Send Payment to: Heath Users' Group
Hilltop Road
St. Joseph, MI 49085

616-982-3463

REMark®

Issue 43 • August 1983

on the stack

ON THE COVER: Pictured is a new offering from Heath Co., the ET-100 educational 16-bit computer. Watch for details in the upcoming Heathkit catalog.

ZLYNK/II

ZLYNK/II ISN'T a modem utility, it's a complete SYSTEM. Modem utilities simply allow you to communicate with another computer system — but ZLYNK/II actually can be PROGRAMMED to operate without user intervention. Commands can be entered from the console, from a program, or even from a login file.

Even more importantly, ZLYNK/II can make DECISIONS based on events! Almost like having someone else do the work — while you have all the fun (without the guilt).

For example, let's say that every day at 10 PM you want to access a timeshare system and pull off the latest stock reports for YOUR stocks. ZLYNK/II will AUTOMATICALLY dial the system at 10 PM, log you in, locate the necessary data, save it to disk (or printer), and get back off. And you can be snoring soundly the whole time!!!

Virtually nothing is impossible for ZLYNK/II. Want a friend to dial in (while you have coffee)? NO problem! Change baud rate and word length on-the-fly? NO problem! Protocols? We supply eight of them, but ZLYNK/II is extensible to an unlimited number by adding new overlays.

Are you the forgetful sort? With ZLYNK/II you'll never lose data because you swapped disks without rebooting. It even remembers the most recent data received — just in case you REALLY MEANT to save it.

Would you believe it even has a BUILT-IN editor?

Could go on, but space here is limited — and ZLYNK/II isn't!! Ask for our FREE brochure. You'll LOVE the difference!!

PALETTE

SURE, you've seen graphics routines before. Maybe you've even seen light pen support and color printer routines as well. PALETTE has all this AND an awf lot more besides!

PALETTE is a complete, integrated graphics design package for general-purpose use. Useful to anyone from artist to engineer. With PALETTE you can draw an image on the screen (with light pen or cursor controls), save it to disk (in compressed, program, or screen form), edit it, or dump it to your printer (monochrome or color).

Before you start thinking of PALETTE as being similar to graphics editors, consider what else it is (and they AREN'T). PALETTE is built upon highspeed 'C' language routines, in compiled form. Screens can be compressed to save disk space, or saved to disk in a form which allows them to be incorporated into your own program.

This is a fully-integrated system, from light pen to hardcopy. PALETTE can be used by itself, or with light pen and/or printer (from us or elsewhere). PALETTE will work with what you have!

Just a few other features we want you to know abo· You can set up reserveu screen areas, define shapes to be moved and scaled, change the footprint of the cursor (including transparent modes), and over 30 different colors are supported. You'll have to ask about the rest!!

PALETTE is certainly different than the competition — it's what they wish THEY were!

IT'S NOT WHAT WE ARE THAT COUNTS...

...IT'S WHAT WE AREN'T!

Both ZLYNK/II and PALETTE cost much less than you think (no, not over $100, FAR from it!!). We WOULD give you the prices here — but frankly, we've a FREE catalog of over 400 different items we'd like you to see as well. Once you see us in operation, we KNOW you'll agree that we're not just another product developer, and not just another computer dealer. While they sleep, we're helping to build the FUTURE of the Z100! Call or write, we LOVE to hear from you — and you'll LOVE the way we solve your toughest problems!!

ZENITH data systems
AUTHORIZED SALES & SERVICE

SOFTWARE WIZARDRY, INC.
122 YANKEE DRIVE
ST. CHARLES, MO. 63301
(314) 946-1968

VISA
MasterCard

One Down ..... How Many to Go?

Recently, I had the pleasure of speaking with Dean Gibson of Ultimeth Corporation. Dean, of course, is the author of the HUG SY: Device Driver under HDOS. What I thought was going to be a pleasing discussion, turned to a tragedy that will send ripples through the entire Heath/Zenith user community.

Why? As most of you know, Dean has supported the Heath/Zenith users with some of the finest software to be produced. Further, he has supported these products via phone consultation and personal contact with various publications including REMark, H-SCOOP, and BUSS/Sextant. Dean sells the products he develops through HUG, H-SCOOP, and direct. When updates and improvements are incorporated, he offers the new products to users at a modest fee. No more folks!

Dean kept very good records of his sales (something that may be a future necessity for most software houses). He indicated that many of his recent phone contacts on the products he offers came from users in cities where the product had not been sold. In checking with the users that contacted him, he found that many of these users obtained his products through copies at local club meetings, through friends who had the product, or through other means.

What does this amount to? Friends, we have lost future development of software for Heath/Zenith products from a very reputable source. This vendor of useful software products can no longer afford to continue to produce software because some individuals continue to illegally copy software.

Software Piracy has been beat to death in several publications including REMark. It is unnecessary to repeat what has been warned over and over. This time the Heath/Zenith user community will pay the price directly via the loss of Ultimeth/Dean Gibson as a source for solid software products.

I am sure this is not the first software house to withdraw the support of a particular computer product line. However, this is the first with such devastating consequences to our user community. One down ..... How many to go?!

Bob Ellerton
HUG Manager

Win up to $500 Worth of Prizes!!!

Now is your chance to cash in on your robotics programming skill and creativity. Enter the first Microcomputing/Heath Company HERO 1 programming contest and win up to $500 worth of prizes. Microcomputing magazine, in conjunction with the Heath Company, manufacturers of the HERO 1, invites all HERO 1 programmers to submit their best applications to this contest. Entries will be judged in the following categories:

1. Standard HERO 1 with arm.

2. Modified HERO 1, including additional RAM or ROM, as well as any mechanical or electrical modifications.

Prizes will be awarded to the top three entrants in *each* category. Two $500 gift certificates (one from each category) will be awarded. Each first place winner will select the prizes of his choice, worth up to $500, from the latest Heath Company catalog. A $100 gift certificate, good toward any purchase from the Heath catalog, will be awarded to both second place winners. Third place winners from each category will receive a copy of Microcomputing columnist Mark Robillard's new book, "HERO 1 Advanced Programming and Interfacing," plus a one-year paid subscription to Microcomputing magazine.

CONTEST RULES

1. All programs must be submitted both on cassette tape and in hard copy form. A brief, written description of the application must accompany each entry.
2. Entries in the modified category must include a complete description of the alterations performed on the robot.
3. The contest is open to all HERO 1 owners, except employees of Wayne Green Inc. (publisher of Microcomputing), and the Heath Company and and their immediate families.
4. All entries, including programs, become the property of Microcomputing.
5. All entries must be received by Microcomputing by September 1, 1983.

Send submissions to:

Robotics Contest
Microcomputing
80 Pine Street
Peterborough, N.H. 03458

7. Contestants may submit more than one entry in one or both categories.

Entries will be judged on originality and technical feasibility. The more practical and easily adaptable the application, the better. Winners will be announced in the December 1983 issue of Microcomputing. So rev up your robot, and let's put the Heath's HERO through its paces!

MICROCOMPUTING®

80 PINE STREET PETERBOROUGH, NH 03458

BUGGIN' HUG

Correction to Grocery Shopping Article

Dear HUG,

See Grocery Shopping article in the June Issue of REMark, Page 9, bottom of left hand column.

The formula has an additional ")", which needs to be removed before it will work. Incidentally, I like the article!

Correctly, it should read, "IF (F8-E8< 0,0,F8-E8)".

Matt Cutter
Heath Company

Info Needed on UCSD-Pascal

Dear HUG,

Regarding UCSD-Pascal, isn't there an easier way to perform programs printing hardcopies than the command: UNITWRITE(PRINTER, STRING,LENGTH); ...? Is there a small neat command that will toggle output to either CONSOLE or PRINTER (1 or 6)? As you see, this will be dealing with printing text and results while executing a program.

Bjorn Petersen
G1 Landevej 18
DK-4000 Roskilde
Denmark

Info On The EC-1120 CP/M Course

Dear HUG,

Being relatively new to computing and just recently having joined HUG, I find REMark a tremendously good source of information. Just couldn't resist the temptation to see all the goodies in those back issues, especially after reading the first lead-in letter to Buggin' HUG in the March '83 issue. I've ordered the set. I do have an item of info that may be helpful to those HUGgies who have taken the Heath CP/M Course EC-1120 and have experienced difficulty with the "XSUB" command on one-drive units. First off, the "XSUB" must be the "first" command in the "filename.SUB" file you create. (This bit of wisdom was found in Doc Campbell's "Getting Started with CP/M and MBASIC".) Since you will be using symbolic parameter to designate the imaginary drive B for one-drive units, the problem is which elements receive the symbolic parameter. These parameters are not used with the XSUB command nor with some of the other functions. Here is a sample program I used which is similar to that used in the CP/M Course (Chap. 10). I labeled my file TRAIN.SUB.

```
1.  XSUB
2.  DIR $1:
3.  ERA $1:*.BAK
4.  ED $1:PRACTICE.4 (PRACTICE.4 is on
drive B)
5.  #A
6.  #T
7.  E
8.  ERA $1:$$*.SUB
9.  DIR $1:
10. STAT $1:
```

When this file is submitted, "A>SUBMIT TRAIN B", some interesting action takes place. There's quite a bit of disk changing but it does work. Also, ED.COM must be on drive A. If you put the symbolic parameter ($1) in front of ED in the above program, it will go back to drive A and edit PRACTICE.4, or open a new file on drive A by that name. This was one of the problems I experienced.

John P. Gallagher
1699 Hoohulu St.
Pearl City, HI 96782

Help Needed From Other HUGgies

Dear HUG,

I would appreciate reading in REMark or hearing from other HUG members about the successful temperature sensing projects with the H89. I would like to use my H89 to monitor and record outside air temperatures. Specific information on A.D converters, sensors, and parallel port interfacing would be appreciated.

Rick A. Martin
8494 East Jamison Circle North
Englewood, CO 80112

More on Cooling the '89

Dear HUG,

Something seemed amiss when I read Larry Fina's letter (June REMark) about the '89 cooling fan, and after some cogitation, I believe his figures prove that blocking the top vents is NOT a good idea.

My vague memory of thermodynamics tells me that if everything else is equal*, then cooler exhaust air means only that it is carrying LESS heat away from the computer, not that the inside is necessarily cooler. The heat that is not carried off must be left behind, inside the computer, making it hotter in there. (*"Everything equal" means temperature of intake air, thermal coefficient of the air, and rate of air flow, plus of course an equal amount of heat being generated inside the computer.)

A better empirical method of evaluating this approach is to measure the actual temperature of the components under operating conditions.

I talked to some engineers and technicians, and they don't favor blocking the vents. This could change the entire pattern of air flow inside the computer, meaning that some hot components might actually get less air for cooling.

I do agree with the idea of blocking 2 or 3 vents beside the fan, to avoid sucking hot exhaust air right back into the computer. And it makes sense to me to remove the ribs that obstruct the flow from the fan since that should allow easier (and therefore greater) airflow. But I wouldn't block the whole set of top vents.

Rob Thorne
Technical Writer
Heath Company

Dear HUG,

Along with Rob Thorne's response to Larry Fina's letter (June REMark), I would like to suggest an alternative method that I use to cool my '89.

When I built my '89, I reversed the air flow by mounting the fan upside down. Now the fan pulls air through the vents onto the power supply. This results in a noticeable movement of warm air out the top vents and the bottom of the computer. The flow is evenly distributed and is doing as good a job, if not better, than when the fan is mounted normally.

However, a potentially serious heat-related problem could be caused by the dust build up in the power supply area. When I swapped out controller boards for soft sectored drives, I also cleaned out an excessive amount of dust and dirt. The concentrated air flow into the computer caused the build up and would normally be avoided if the fan were mounted correctly. This build up could reduce the effectiveness of the heat sinks on the power regulators and transistors to a degree where they could fail (through thermal run-away) and cause some real system problems.

But as long as I clean that area on a regular basis (twice a year is sufficient), I enjoy the benefits of a system that is receiving adequate ventilation.

Tom Huber
St. Joseph, MI 49085

Vectored to 8 ☞

PACMAN for the H/Z-89

Here is a program written in Benton Harbor BASIC for the H/Z-89 which simulates "PACMAN". It makes full use of the graphics capabilities of the H/Z-89. If you have any questions, I may be reached at the address below.

Richard H. Barrett
309 Westbury Dr.
Coraopolis, PA 15108

```
PACMAN for the H/Z-89

00010 REM THIS PROGRAM SIMULATES PACMAN
00020 REM WRITTEN BY R.H. BARRETT FOR THE H89 IN BENTON HARBOR BASIC
00030 REM REQUIRES FILE "HUGMAN.DAT" (OPTIONALLY ENTERED USING DATA STATEMENTS)
00040 DIM D(31),S(31)
00050 E$=CHR$(27)+"E": F$=CHR$(27)+"F":
      G$=CHR$(27)+"G": Y$=CHR$(27)+"Y"
00060 P$=CHR$(27)+"p": Q$=CHR$(27)+"q":
      X$=CHR$(27)+"x5": Z$=CHR$(27)+"y5"
00070 FOR I=1 TO 31: READ D(I): NEXT I
00080 DATA 2,4,1,3,2,4,1,4,2,1,3,1,3,2,4,1,3,2,4,1,3,2,3,1,2,4,2,4,1,3,2,
00090 FOR I=1 TO 31: READ S(I): NEXT I
00100 DATA 35,5,15,18,15,5,5,4,3,3,4,43,5,25,5,48,5,15,18,15,5,5,4,3,3,4,43,
      5,25,5,13
00110 PRINT F$;E$;X$
00120 OPEN "HUGMAN.DAT" FOR READ AS FILE #1
00130 FOR I=1 TO 23: LINE INPUT #1,;A$: PRINT A$: NEXT I
00140 PRINT P$;Y$;"$#p";Y$;"$jp";Y$;"1#p";Y$;"1jp"
00150 PRINT P$
00160 PRINT Y$;"*9q";Y$;"*?q";Y$;"*Lq";Y$;"*Rq";Y$;"*Xq"
00170 PRINT Y$;"+9q";Y$;"+?q";Y$;"+Lq";Y$;"+Rq";Y$;"+Xq"
00180 PRINT Y$;",9q";Y$;",?q";Y$;",Eq";Y$;",Lq";Y$;",Rq";Y$;",Xq"
00190 PRINT Q$
00200 C=69
00210 R=46
00220 G=2
00230 FOR I=1 TO 30: GOSUB 310: C=C-1: NEXT I
00240 FOR I=1 TO 31: G=D(I): L=S(I)
00250 FOR J=1 TO L: GOSUB 310: NEXT J
00260 NEXT I
00270 FOR I=1 TO 41: GOSUB 310: C=C-1: NEXT I
00280 PRINT G$;Z$
00290 PRINT Y$;"6 "
00300 END
00310 ON G GOTO 320,390,460,540
00320 C=C-1
00330 A$=Y$+CHR$(R)+CHR$(C)+P$+"r^  _"+Q$+" "
00340 IF M=0 THEN B$=Y$+CHR$(R+1)+CHR$(C)+"  "+P$+"   "+Q$+" "
00350 IF M=1 THEN B$=Y$+CHR$(R+1)+CHR$(C)+P$+"     "+Q$+" "
00360 C$=Y$+CHR$(R+2)+CHR$(C)+"_"+P$+"   "+Q$+"r "
00370 PRINT A$;B$;C$
00380 GOTO 610
00390 A$=Y$+CHR$(R)+CHR$(C)+Q$+" "+P$+"r  ^_"
00400 IF M=0 THEN B$=Y$+CHR$(R+1)+CHR$(C)+Q$+" "+P$+"   "+Q$+"  "
00410 IF M=1 THEN B$=Y$+CHR$(R+1)+CHR$(C)+Q$+" "+P$+"     "
00420 C$=Y$+CHR$(R+2)+CHR$(C)+Q$+" _"+P$+"   "+Q$+"r"
00430 PRINT A$;B$;C$
00440 C=C+1
00450 GOTO 610
00460 R=R-1
00470 IF M=0 THEN A$=Y$+CHR$(R)+CHR$(C)+P$+"r_"+Q$+" "+P$+"r_"
00480 IF M=1 THEN A$=Y$+CHR$(R)+CHR$(C)+P$+"r   _"
00490 B$=Y$+CHR$(R+1)+CHR$(C)+"^    "
00500 C$=Y$+CHR$(R+2)+CHR$(C)+Q$+"_"+P$+"   "+Q$+"r"
00510 D$=Y$+CHR$(R+3)+CHR$(C)+Q$+"
00520 PRINT A$;B$;C$;D$
00530 GOTO 610
00540 A$=Y$+CHR$(R)+CHR$(C)+Q$+"     "
00550 B$=Y$+CHR$(R+1)+CHR$(C)+P$+"r   _"
00560 C$=Y$+CHR$(R+2)+CHR$(C)+"    ^"
00570 IF M=0 THEN D$=Y$+CHR$(R+3)+CHR$(C)+Q$+"_r _r"
00580 IF M=1 THEN D$=Y$+CHR$(R+3)+CHR$(C)+Q$+"_"+P$+"   "+Q$+"r"
00590 PRINT A$;B$;C$;D$
00600 R=R+1
00610 M=M+1
00620 IF M=2 THEN M=0
00630 RETURN
```

Screen layout for Pacman on page 51.

Comments on "Morse Code Practice in MBASIC"

Dear HUG,

I was pleased to see the program in REMark issue 41 on Morse Code Practice in MBASIC by Bob Horn and was challenged to try to get it running on my H-8/H-19 using HDOS MBASIC.

After chasing a few bugs down, it was determined that my new H-19 would require modification as indicated in the article. Careful study of the terminal logic board and schematic showed that the H-19 does not use the same component numbering system as the H- 89. Changing C-417 (2.2 uf) to 1 uf did the trick. I further modified the H-19 by adding a miniature phone jack with a 500 ohm potentiometer in series connected to the speaker. This enables code practice with headphones as I use my computer in a noisy environment. Caution is necessary when connecting a phone jack in this manner if it is to be mounted on the terminal backplate as one side of the speaker is connected to Vcc. The only solution is to use an insulated jack to avoid shorting either +5 volts or IC U416 pin 8 to ground.

I have not tried to modify the program to use the H-8 "HORN", but suspect that more work would be involved.

I am enjoying REMark much more than I did a year ago as it has better format and covers a larger variety of applications. Keep up the good work!!

Wayne Linschied
USS SAMPLE (FF-1048)
FPO San Francisco, CA 96678

[ED: Two errors crept into the above mentioned article. First, was the indication that the program was written for CP/M when in fact it is an HDOS program. Secondly, the @ symbols were inadvertently dropped at the end of continuing lines in the conversion of the program.]

Switching Output From Terminal to Printer Under CP/M MBASIC

Dear HUG,

There was a question in "Questions & Answers" in the May 1983 issue concerning switching output from terminal to printer under CP/M MBASIC. A better answer was given in an old REMark (I don't know which one anymore), and here it is:

```
10 LET PR=PEEK(3)
20 LINE INPUT "Do you want a print-out?
(Y/N) :;P$
30 IF P$="Y" THEN POKE3,PR+1
40 PRINT "THIS IS A TEST"
```

```
50 POKE3,PR
60 GOTO 20
```

Turn off your printer, and run this program. It is handy, because you only need enter text to be printed once (rather than duplicate PRINT and LPRINT statements). Be sure to include line 50 whenever you use this, or the computer will go off into the Ozone, never to respond again (until the next hard reset)!

Maureen Hackley
PO Box 579
Portland, OR 97207

New Device Driver for the H-25

Dear HUG,

For all owners of the H/Z-25 line printer using HDOS and Microsoft BASIC. If you try to print out data on full width 14" paper and are using the (patched) LPH24 device driver that was supplied in Appendix 1 of the H/Z-25 operations manual, it won't work (along with several other features). Call Heath Company and order part number HOS-5-UP. This is the new device driver for the H/Z-25. It should be sent to you at no charge.

When I received my update, I was pleased to find I also got the complete HDOS 2.0 update (even though I'm still using HDOS 1.6). I was under the impression that any updates would be sent to me automatically by Heath (they do have computer records, don't they?). But I was wrong, you have to ask for these updates. I apologize for all the bad words I said before I discovered this.

John Czarnecki
815 Valencia Street
Walla Walla, WA 99362

New HUG Club Forming in Arkansas

Dear HUG,

I would like to explore the establishment of a local user group in Arkansas. All persons in Arkansas, southwest Missouri, southeast Kansas, and northeast Oklahoma who are interested in forming a club, please contact me.

Gil Hoellerich
2617 Country Way
Fayetteville, AR 72701
501-521-4818

Another Patch To CP/M BASIC V 5.21

Dear HUG,

Your June 1983 REMark contained on page 42 an article titled Get Rid Of "Echo On Delete" in CP/M-85 and MBASIC (...And Other

Vectored to 48

QUESTIONS & ANSWERS

(EDITOR'S NOTE: If you need answers to specific questions on software or hardware problems that would be beneficial to other users, please drop us a note to: Questions & Answers, Heath Users' Group, Hilltop Road, St. Joseph, MI 49085. Please keep your questions brief and to the point. We will do our best to answer your question here in this column in future issues. Some Questions & Answers are contributed by Zenith Data Systems Software Consultation.)

Q. I have a mailing list program on a non-SYSGENed HDOS disk to give more space for data and thus I boot-up with a SYSGENed disk containing MBASIC, after which I RESET SY0:. How can I delete files and CAT on this non-SYSGENed disk?

A. You are able to delete files and do a directory of the non- SYSGENed disk from Microsoft BASIC. The command to delete a file can be entered from the command mode ("OK" prompt) or from program mode (after a line number) as follows:

```
KILL "filename.ext"
```

The entire filename and extension must be included to delete the file. The disk directory can be entered only from the command mode as follows:

```
FILES "SY0:"
```

The drive name must be in quotes the same as any other command from MBASIC.

These same functions can be done from HDOS by using the program called PIP. PIP must be loaded from the SYSGENed disk. When you DELETE a file, do a DIRectory, or RESET a disk, the program PIP is actually what is called from HDOS to do the job. By entering PIP yourself, you will not only be able to do any of the file handling instructions but it will save you time also if you have a number of commands to issue.

PIP uses "flags" to issue commands to the system. Refer to your HDOS manual for details of the flags and how to use PIP. For example, to: 1) RESET SY0:, 2) do a DIRectory of all files on SY0:, 3) TYPE an ASCII file on the screen, 4) DELETE the file, and 5) RESET back to a SYSGENed disk, the following commands could be entered:

```
>PIP
:P:SY0:/RES
:P:/L/S
:P:filename.ext
:P:filename.ext/DEL
:P:SY0:/RES
:P: ↑ D          (enter a CTRL-D to exit)
```

These are the commands to be entered at the PIP command prompt, :P:. Messages and information would appear on the screen with each command entry.

Note:

1) To reset any disk in HDOS, you will need to enter only once at the HDOS prompt:

```
>SET HDOS STAND-ALONE
```

2) If you exit (CTRL-D) from PIP while mounted on a non-SYSGENed disk, HDOS will dismount the disk and ask you to reboot.

Q. I made a copy of my Z-100 demonstration disk, but after I use it for about 20 minutes, the computer locks up and I have to reboot. What is wrong?

A. When you made a copy, you may not have formatted a blank disk. The Format Utility in ZDOS puts the system tracks on the disk using your Z-DOS system software. It should also be noted that the version of Z-BASIC on the Z-100 Demo disk is not compatible with the current version of Z-DOS on the Z-DOS distribution disk. This is the cause of the lockup.

Vectored to 49

MAGNOLIA
MICROSYSTEMS

CP/M Plus™ Released!

Under development since late last year, CP/M-Plus^{tm} support is finally available for Z89 and Z90 computers with our 128K RAM board and either Zenith's Z89-37 (Z90) or our own 77316 Double Density disk controller.

With the banked RAM available on our 128K board, disk performance is dramatically enhanced through the use of Hashed Directory tables, Directory Buffers, and LRU Data Buffers.

New utilities and features include a 'Help' command; optional Password protection and Time and Date Stamping of files; Console Redirection to or from disk files; and many others.

Digital Research's list price for CP/M-Plus is $350, but as an introductory special, it will be included AT NO EXTRA CHARGE with our 77318 128K RAM board if purchased before September 1, 1983!

Our implementation of CP/M-Plus REQUIRES the use of our 128K RAM board. We have no plans to implement the non-banked memory version because most of the advantages of CP/M-Plus are not available (or practical) in the non-banked version.

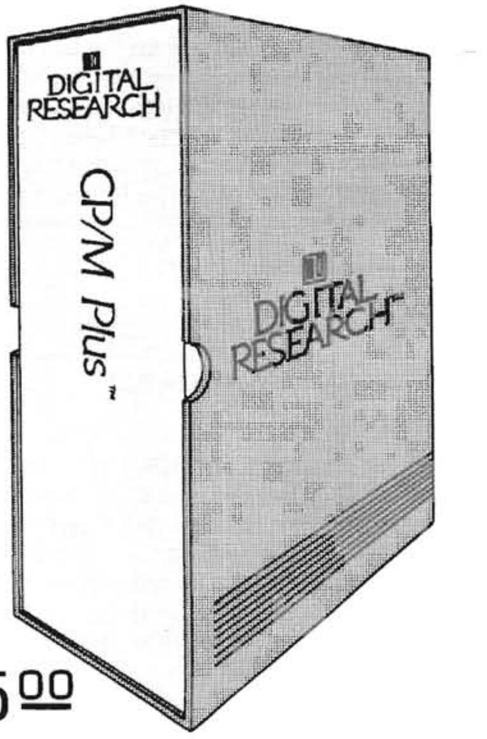
We are including the SOURCE code for our BIOS, together with Digital Research's MAC, RMAC, LINK, and SID software development tools, so you may make whatever modifications necessary for your specific application.

When ordering, be sure to specify which double density controller and size drive you boot from: Zenith's Z89-37 (Z90) or Magnolia's 77316 (5-inch or 8-inch).

Customers who purchased the 128K RAM board after our announcment of CP/M-Plus at CP/M-83 in San Francisco (who provide satisfactory proof-of-purchase) can obtain CP/M-Plus for the nominal $50 cost of the CP/M-Plus documentation (plus shipping and handling).

Earlier purchasers of the 128K RAM board who are also registered owners of the Magnolia Microsystems release of CP/M 2 (who provide satisfactory proof-of-purchase) can update to CP/M-Plus for $100 (plus shipping and handling).

CP/M is a registered trademark, and CP/M-Plus, MAC, RMAC, SID and the DR logo are trademarks of Digital Research



$ 595<u>00</u>

128K RAM Board with CP/M-Plus^{tm}

Double Density Controller reduced $100

now only $495.

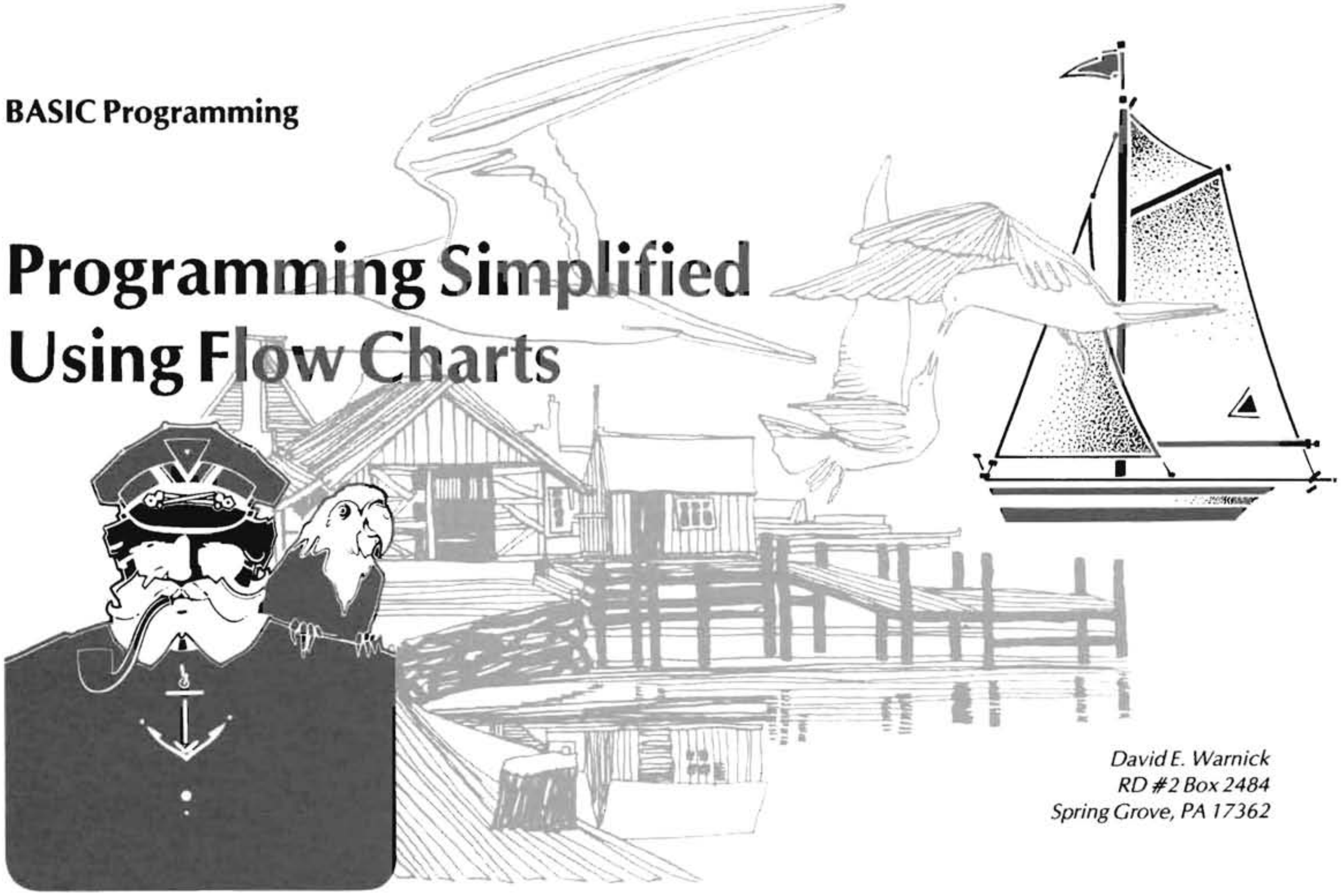
Save $100 on the most versatile controller available for the Z89 or Z90! Supports four 5-inch and four 8-inch single or double-sided Shugart compatible drives. Read and Write over 27 different (including all current Zenith Z89/Z90) media formats.

Everything is the same as before except the price! Includes a copy of CP/M^{tm} 2.2, cables for both 5- and 8-inch disk drives, and a new Monitor (Boot) EPROM.

Subsystems reduced also -- take $100 off of all the double-density floppy disk subsystems shown in our price list dated 3/11/83. For example, our popular Dual 8-inch Double-Sided Subsystem was $2695, now only $2595!

MAGNOLIA MICROSYSTEMS, INC.
2264 - 15th Ave West • Seattle, WA 98119
(206) 285-7266 • (800) 426-2841

Programming Simplified Using Flow Charts



David E. Warnick
RD #2 Box 2484
Spring Grove, PA 17362

Last month I promised that we'd begin to write a game, and that's just where we'll start. We want to play something that's fun, that any age can enjoy, and that the whole family and friends can join in. However, let's not forget the purpose of this column. I really want to show you how to take an idea for a computer process, be it a game, a business application, or anything else, and put that idea into a working program. One that provides all the features you want, one that can be run by anybody, and one that will forgive you for pressing the wrong key. I've chosen this game because it can be made to work in a very simple form and then improved, improved again, and improved further as we learn new techniques.

The idea behind the game is very simple. Someone thinks of a word and types it on the keyboard. When they type return, the computer erases the word and presents the letters in a jumbled fashion. Now the second player or group of players tries to unjumble it. The game has caught on with everyone I've shown it to, and has become habit forming to some. It also presents a lot of programming challenges which we'll solve, hopefully showing you a good approach to programming and processing. We'll try to explain everything thoroughly, but if we do leave you with a question, write us at the address above and include a stamped self-addressed envelope for the answer.

This article is copyrighted by the author and all programs herein are copyrighted by APPLIED COMPUTING so they may not be reproduced except for your personal use. Have fun with them.

Obviously, our finished program will be very complex, but taken a small piece at a time, you'll find it easy. The problem arises when we try to put all the pieces together. We'll have to put them in the right order, make references from one to another, and be able to follow the program as we jump about through it. This is the part of writing your own software that can drive even the best of programmers up the wall. So, what can we do? Programming is an integral part of our hobby. Without it, we're stuck running canned routines and we miss half the fun of computer ownership.

Well, it's not as hopeless as it seems. In fact, not only is the solution to our problem simple, but it will open up a new avenue of understanding. The answer is flow charts. I know you've seen those things before and they seem complicated. Many of you also think that you'll never learn to draw one and use it. Nothing could be further from the truth. Before you finish this article, you will know about and be able to use flow charts, and you'll do it well. Once you've learned this skill, you'll wonder how you ever got along without it, and you'll use it regularly in your programming.

What makes this so easy? The fact that there are very few things a computer can do, and each of them is represented by a symbol. What are these things, and what are the symbols? Take a look at Figure 1.

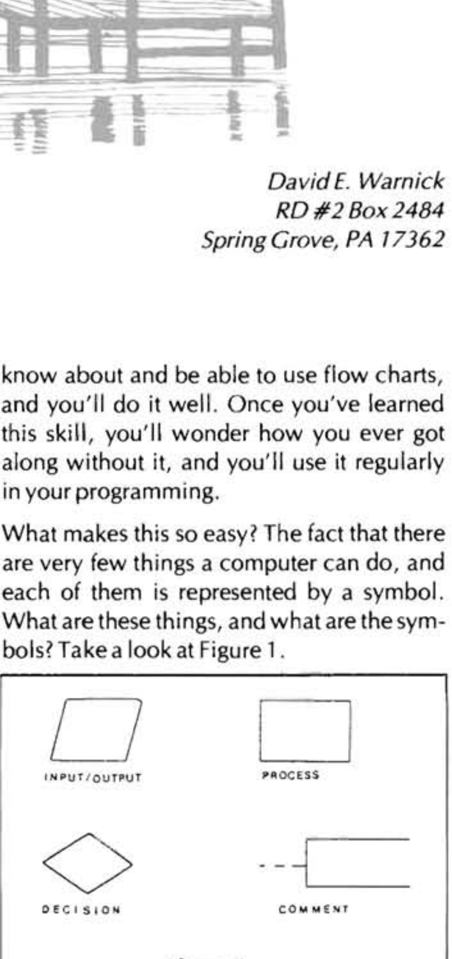


Figure 1

Here you see, in very simple form, symbols for everything your computer can do. All symbols used in this and future articles have been adopted as standard by the American National Standards Institute (ANSI). They'll be understood by other programmers and apply to any computer from the Timex/Sinclair to the IBM 370 monsters. The first symbol represents any input or output function, whether you type on the keyboard, print on the printer, or read from or write to the disk. If you see this symbol in a flow chart, it is calling for an input or output function.

The second symbol is used when processing is performed. It could be anything from adding numbers to sorting a list of names and addresses. The third symbol represents a decision. This is used for things like IF.....GOTO, or any time a decision is made. When you see this symbol, you can expect program operation to be sent to a new subroutine if certain conditions are met.

It seems as though these symbols are almost too general. If a rectangle can be any process, how do you know exactly what the program is doing? You write the function inside the block. If more detail is needed, you use the fourth symbol for notations. Let's see how it works. Figure 2 shows the flow chart for a program which writes your name ten times. First it inputs your name, then writes it. The third symbol shows a decision, "Has the name been written ten times?". If the answer is no, the program is directed to write again. When the answer is yes, program execution can continue toward the end.

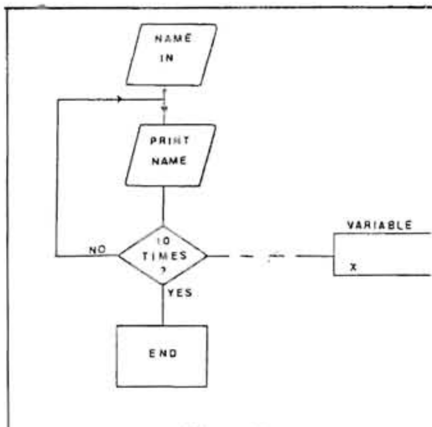


Figure 2

We have used only three symbols, and have shown how a simple program works. Now we'll write a program to do what we wanted, and we'll show line numbers on the symbols so we can refer to them for debugging or later changes. Figure 3 is our newly completed flow chart. I used it as a guide to write our program, and can refer to it to check program execution steps and make debugging easier. Go ahead, type the program, and run it. Follow the flow chart to see what each step of the program is doing. Here it is.

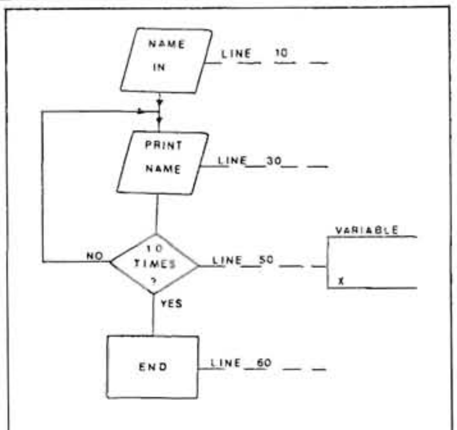


Figure 3

It would be nice if more detail could be drawn into the chart. The four symbols of Figure 1 were easy, so we'll add the special symbols of Figure 4. I've added eight more symbols. Some will get little use, and there are others I haven't shown, such as Paper Tape, Punched Card, etc. These new symbols can be used as follows:

Document - This symbol indicates that a document is used. We'll use it mostly to show output from our printer, but it could be any form with input or output data.

Manual Input - This symbol shows a required keyboard entry.

Connector - Use this one to show common points in a program not easily connected by a line. I'll show this in the game chart as we progress.

Display - This symbol indicates an output to the CRT or a plotter.

Communications Link - When we get into MODEMs, this will be a useful symbol. At this point, we won't use it much.

Off-Line Storage - This shows storage of data not readily accessible to the computer. Cassette storage will be our biggest use.

On-Line Storage - We'll use this one when we want to write info to a file on a disk.

Terminal - Refers to a terminal point or entry point in a program or routine. Not to be confused with our CRT and keyboard.

Don't try to memorize these symbols. As you use them, you'll remember the common ones. I'd also suggest buying a programmers template. It's a thin sheet of plastic with these and other symbols punched out so you can draw them neatly and easily. Each symbol is identified on the template so you won't have to remember what they're for. These templates are available at your Heathkit Center, local computer store, Radio Shack Store, and some drafting supply houses. They'll set you back about five bucks, but are well worth it. There's no one brand to look for. Any of them will do the job.

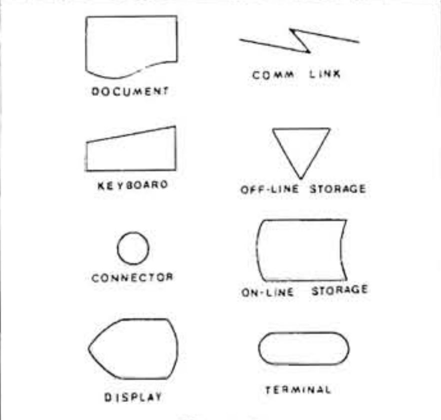


Figure 4

We're finally through all the formalities. We've seen the symbols we'll use, and have used them to describe what we want a simple program to do. Then we wrote the program, making sure we completed every step of the chart. With the small program we didn't really need the aid, but, with more complex problems the flow chart symbols will reduce the whole process to smaller steps, and these steps will be easier to follow and write. It's time now to get into a larger program and put what we've learned to use. Figure 5 is the Program Flow Chart for the

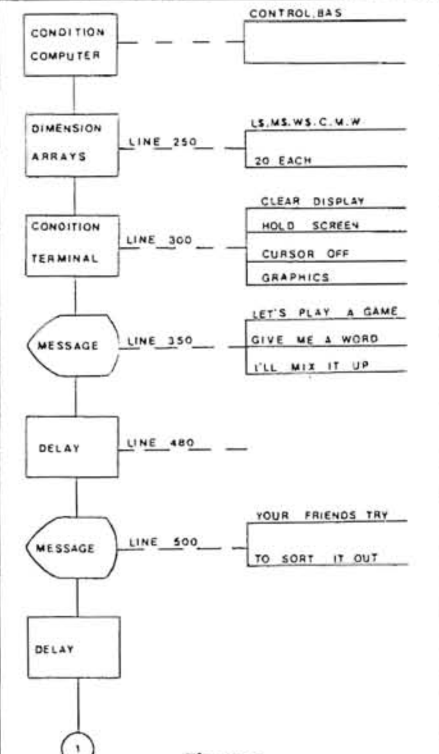


Figure 5

```
10 INPUT "WHAT IS YOUR NAME";N$        'INPUT THE NAME
20 X=0                                 'SET A CONTROL VARIABLE
30 PRINT N$                            'PRINT NAME 1 TIME
40 X=X+1                               'INCREMENT CONTROL VARIABLE
50 IF X<10 GOTO 30                     'HAS IT BEEN PRINTED 10 TIMES
60 END
```

first portion of our game of "WORDS". It shows each step we must perform to get the screen ready and introduce the players to the game. There are refinements to be added later as we work with the finished program and discover "nice" features we'd like, but this will work as is. Let's look at it. There are line numbers for most of the symbols. These are the lines where each step begins. You would normally add them as you write the program. I've put them in now to reduce the number of figures required in this lesson.

The first block tells us to use our library program CONTROL.BAS, which we saved from last month's article. We'll add that later using the MBASIC command MERGE. The second block calls for us to dimension all arrays. We can do this as we go through the program. As we discover a need for an array, we'll come back to line 250 and add it to the DIMension instruction. Why start at line 250? Don't we usually start at 10? Yes, we often do, but we can start anywhere we want, and we must leave room for CONTROL.BAS to fit in at the beginning. We also break our programs into smaller pieces by using a different series of numbers for each section. That's part of our concept of modular programming.

At line 300, we'll condition our terminal. We see by our notation that we want it to:

1) Clear the screen
2) Enter the Hold Screen mode
3) Turn the cursor off
4) Enter the Graphics mode

To do this, we'll use functions provided by CONTROL.BAS. Our program line becomes:

```
300 PRINT E2$;E5$;E3$;E7$
```

With that done, we'll begin to display messages to let the players know what's going on. The first message will have three lines as shown on our flow chart, and we'll begin programming at line 350. To print large letters, we'll use GOSUB routines just as we did in the last article on large letters. We'll refer to lines in our library program LETTERS.BAS. Before we run this routine, we'll MERGE that sub-program too. Where do we start on the screen? With three lines of text each four characters high, we'll use twelve screen lines. That leaves another twelve lines for spaces at the top, 2 in the middle, and another at the bottom. Let's use 2 lines for the top space, 2 for the bottom, and 4 between each line of the message. That means we must start our letters on lines 3, 11, and 19.

Horizontal spacing must also be worked out. We want our message to be neatly centered on the screen. Therefore we must look at LETTERS.BAS to see how wide each letter is. On our first line, each is 4 characters wide. We'll use 4 characters for a space too, but what about the apostrophe? We don't even have one of those. We're all programmers. We'll write an apostrophe routine just like the letters. We can put it after the letters at line 5500 like this.

```
5500 PRINT E1$;CHR$(Y);CHR$(X);"r"
5510 X=X+2
5520 RETURN
```

Our first line now has 13 letters and 4 spaces at 4 characters each, and 1 apostrophe 2 characters wide for a total of 70 characters of printing in the message. On our 80 character screen, that leaves 10 characters of space. We divide that evenly as 5 characters before and 5 after the message. We must therefore start printing at character 6.

Now do the same for the next two lines. Note that the letter "I" is 2 characters wide, and the letters "M" and "W" are each 6 characters wide. Line 2 of the message starts on character 12 and line 3 starts on 13. The program will read like this:

Now turn on your computer and call up MBASIC. Type in lines 5500 to 5520, line 300, and lines 350 to 590. Use the AUTO instruction for lines 350 to 590. If you don't know how to do this, refer to your MBASIC manual. When more than one physical line is required to complete a program line (line 460 is a good example), use the "LINE FEED" key to continue on the next physical line. When you've completed line 590 and the AUTO function prints 600, type CONTROL C. To do this, hold down the CTRL key and type a "C". Now type:

```
SAVE "WORDS",A
```

This will save what you've done on the disk in a file called WORDS.BAS. It's a good idea to use the SAVE command often. Then if you make a mistake or have a problem, the work you did prior to the save will not be lost. **ALWAYS SAVE ANY PROGRAM BEFORE RUNNING IT!!!** If there's an error in your work, you won't have to do a lot of retyping.

Now refer to your MBASIC manual and look up the MERGE command and read it. Then

```
350 X=37:Y=34              'CHARACTER 6, LINE 3
360 GOSUB 5110:GOSUB 5040:GOSUB 5190:GOSUB 5500:GOSUB 5180:X=X+4
'LET'S (SPACE)
370 GOSUB 5150:GOSUB 5110:GOSUB 5000:GOSUB 5240:X=X+4  'PLAY (SPACE)
380 GOSUB 5000:X=X+4:GOSUB 5060:GOSUB 5000:GOSUB 5120:GOSUB 5040 'A GAME
390 FOR X=1 TO 300:NEXT X  'PAUSE BEFORE PRINTING NEXT LINE
400 X=43:Y=42              'CHARACTER 12, LINE 11
410 GOSUB 5060:GOSUB 5080:GOSUB 5210:GOSUB 5040:X=X+4  'GIVE (SPACE)
420 GOSUB 5120:GOSUB 5040:X=X+4:GOSUB 5000:X=X+4   'ME A (SPACE)
430 GOSUB 5220:GOSUB 5140:GOSUB 5170:GOSUB 5030    'WORD
440 X=44:Y=50              'CHARACTER 13, LINE 19
450 GOSUB 5080:GOSUB 5500:GOSUB 5110:GOSUB 5110:X=X+4  'I'LL (SPACE)
460 GOSUB 5120:GOSUB 5080:GOSUB 5230:X=X+4:GOSUB 5080:GOSUB 5190:
X=X+4   'MIX IT (SPACE)
470 GOSUB 5200:GOSUB 5150         'UP
```

That completes our message. We'll use a loop to cause a delay giving the players time to read the screen. Then we'll erase the screen and begin the second part of our message. Use the same methods as before to center each line and provide vertical spacing of this two-line display. Follow the message with another delay before erasing the screen. This will get us ready to enter the input sub-routine next month. Here goes:

type:

```
MERGE "CONTROL"
```

When you get the prompt "OK" type:

```
MERGE "LETTERS"
```

Again type:

```
SAVE "WORDS",A
```

We're almost ready to test the first part of our

```
480 FOR X=1 TO 900:NEXT X       'DELAY
490 PRINT E2$                   'ERASE THE SCREEN
500 X=40:Y=38                   'CHARACTER 9, LINE 7
510 GOSUB 5240:GOSUB 5140:GOSUB 5200:GOSUB 5170:X=X+4  'YOUR (SPACE)
520 GOSUB 5050:GOSUB 5170:GOSUB 5080:GOSUB 5040:GOSUB 5130:
GOSUB 5030:GOSUB 5180:X=X+4   'FRIENDS (SPACE)
530 GOSUB 5190:GOSUB 5170:GOSUB 5240   'TRY
540 X=45:Y=46                   'CHARACTER 14, LINE 15
550 GOSUB 5190:GOSUB 5140:X=X+4   'TO (SPACE)
560 GOSUB 5180:GOSUB 5140:GOSUB 5170:GOSUB 5190:X=X+4  'SORT (SPACE)
570 GOSUB 5080:GOSUB 5190:X=X+4:GOSUB 5140:GOSUB 5200:GOSUB 5190
'IT OUT
580 FOR X=1 TO 900:NEXT X       'DELAY
590 PRINT E2$                   'ERASE THE SCREEN
```

program. We've completed a "MODULE" of the whole game. If we debug this section now, it will be a lot easier than when the whole program is running with 3 or 4 times as many lines. However, if we run the program now, it will print our messages and run into the "LETTERS" sub-program. It will also leave our terminal in the Hold Screen and Graphics modes with the cursor turned off. To test our work, we've got to add a temporary line 600 to take care of this. The line will have to:

1) Exit Graphics Mode
2) Exit Hold Screen Mode
3) Turn The Cursor On
4) End Processing

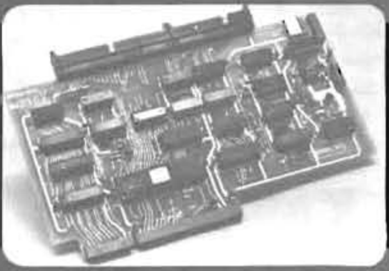
Our new temporary line will read like this:

```
600 PRINT E8$;E6$;E4$:END
```

Add line 600 to your program and type RUN. If you did everything correctly, you got a 3-line message followed by a 2-line message. The second message was erased and the "OK" prompt appeared in the upper left-hand corner of the screen with the cursor below it. Type SYSTEM to exit MBASIC and return to your operating system. That's all we'll do this month. Let's review what we've done and learned.

First we looked at FLOW CHARTS, then we used one to develop part of a much larger program. We typed several lines of BASIC statements and used the MERGE command to add over 170 lines of programming without having to type them again. We SAVEd our work with the ",A" option so we can MERGE it again next month. Finally, we tested our "MODULE" by supplying additional information (line 600) which the complete program will provide later. Welcome to MODULAR PROGRAMMING. Like it or not, you've come a long way and are becoming quite a programmer. See you next month.

1 CONTROLLER



FOR 8" & 5.25" DRIVES

Now be able to run standard 8" Shugart compatible drives and 5.25" drives (including the H37 type) in double and single density, automatically with one controller.

Your hard sectored 5.25" disks can be reformatted and used as soft sectored double density disks. The FDC-880H operates with or without the Heath hard sectored controller.

NEW PRICE $495

Includes controller board CP/M boot prom, I/O decoder prom, hardware/software manuals BIOS source listing.
HDOS driver now available for $40.00.

5-20 day delivery–pay by check, C.O.D., Visa, or M/C.

Contact:
C. D. R. Systems Inc.
7210 Clairemont Mesa Blvd.
San Diego, CA 92111
Tel. (619) 560-1272

Newline Software

P.O. Box 402 • Littleton, MA 01460
617-486-8535 (evenings and weekends)

Software For Heath/Zenith Computers

HDOS 2.0 upgrade for the Z-100 now shipping!

Z-100 HDOS 2.0 upgrade for the Z-100 computer **$79.95**
Installation requires:
Standard HDOS distribution diskettes.
H8 or H89 computer system, Z-100 with ZDOS.
Disk drive compatible with Z-100 (48 TPI SSDD, DSDD).
Disk drive compatible with your standard HDOS disks.
Please order same format as your standard HDOS disks.

Z-100 HDOS 2.0 OPTIONAL SOFTWARE (indicate Z-100 on order)

* Super Sysmod 2	$29.95
* UD.DVD Universal Printer Device	$29.95
* MX.DVD MX-80 Device Driver	$29.95
* H25.DVD Device Driver	$29.95
* UD.DVD/SPOOLER (with built-in spooler)	$39.95
* MX-80/SPOOLER (with built-in spooler)	$39.95
* H25.DVD/SPOOLER (with built-in spooler)	$39.95
* Text Processor 4.1	$59.95

HERO-1 ROBOT SOFTWARE

HERO-1 CROSS ASSEMBLER .... **for H8/H89 computers** **$59.95**
Supports Motorola 6800 Assembly Language and Special HERO-1 OP Codes. List control and assembly directives.

HERO-1 KEYPAD UTILITIES in HERO CASSETTE FORMAT **$24.95**
1. SPEECH UTILITY for creating/manipulating phonemes.
2. MANUAL ENTRY UTILITY eases manual program entry.
3. VOICE DUMP UTILITY same functions as MANUAL ENTRY UTILITY plus robot speech used for reading memory.

* Also available in formats for the H8 and H89 computers.
* Copyright 1983 by SoftShop, Produced by Newline Software

FREE CATALOGUE – Send for yours today!

Using Advanced Features of the H/Z-100 Computer

Pat Swayne
Software Engineer

If you own one of the new H/Z-100 series computers, you are probably aware of some of the marvelous things it does using the ZBASIC graphic language. But you may not be aware of the marvelous things it can do outside of the world of ZBASIC. In this article, I will present some things that it can do in any high level language or in assembly language, under either ZDOS or CP/M on the 8-bit side of the machine.

Color Graphics

As you probably know, the H/Z-100 supports H19 escape codes, and graphic characters. It also supports additional escape sequences that let you access some of its own capabilities. One sequence lets you set the foreground and background color of any character on the screen, or of the whole screen. The sequence is ESC mFB, where F and B are numbers (0 through 7) that represent the Foreground and Background colors. The colors produced for each number are:

0 Black	4 Green
1 Blue	5 Cyan (blue-green)
2 Red	6 Yellow
3 Magenta (violet-red)	7 White

For example, if in BASIC you have:

```
PRINT CHR$(27)"m61";
```

It sets the foreground to yellow and the background to blue. Here is a BASIC program that illustrates this feature. Try running it with MBASIC under CP/M (it will also work with ZBASIC).

```
10 DIM A$(7)
20 E$=CHR$(27):M$=E$+"m":N$=M$+"70"
30 FOR I=0 TO 7:READ A$(I):NEXT I
40 PRINT E$;"E":PRINT
50 PRINT "HERE ARE THE H100 COLORS:":PRINT:PRINT
60 FOR I=0 TO 7
70 B$="0":IF I=0 THEN B$="7":REM  BACKGROUND COLOR
80 PRINT "THIS IS ";M$;CHR$(I+48);B$;A$(I);N$;
90 IF I/2<>INT(I/2) THEN PRINT :GOTO 110
100 PRINT TAB(40);
110 NEXT I:PRINT
120 DATA BLACK,BLUE,RED,MAGENTA,GREEN,CYAN,YELLOW,WHITE
```

The program prints the name of each color in its color. The background color is switched to white for the word BLACK so that it will show up. If you clear the screen with ESC E after setting up colors, they will affect the whole screen. To illustrate this, change line 20 of the above program to read

```
20 E$=CHR$(27):M$=E$+"m":N$=M$+"71"
```

and add line 25:

```
25 PRINT N$;
```

Now, run the program again. The background is now blue all over the whole screen, except where the words BLUE through WHITE are printed, where it is still black. You can switch things back to the default setting of white on black with ESC z (PRINT CHR$(27)"z";).

You can produce colors other than the 8 available by using the graphic symbol produced by the letter "i" to mix foreground and back ground colors. Try this little program.

```
10 E$=CHR$(27)
20 PRINT "HERE IS A BLOCK OF ORANGE: ";
30 PRINT E$"F"E$"m62iiiiiii"E$"m70"E$"G"
```

If you have the low profile model and you are using a color monitor for your only screen, you may find yellow characters easier on the eyes than white characters. The following assemble program will switch to yellow under CP/M.

```
        ORG     100H
START   LXI     D,MSG           ;POINT TO MESSAGE
        MVI     C,9             ;BDOS PRINT FUNCTION
        JMP     5               ;PRINT MSG AND RETURN TO CP/M
MSG     DB      27,'m60',27,'E$'
        END
```

Call the resulting COM file YELLOW.COM, and use CONFIGUR to set up your system so that YELLOW is run on cold boot. Presto! You have an "amber" CRT.

Note: Color on H/Z-100 computers is available only if you have color memory installed. If you only have a monochrome monitor, the colors will show up as different shades. To show the greatest range of shades, adjust the "black level" on your monitor to produce a faintly visible background with the screen blank or nearly blank. Then adjust the brightness of the characters to as bright as you can without "blooming" (characters going out of focus). The black level control on H/Z-120 models is R139 (BRITE) on the Video Sweep board.

Saving the Screen

ZBASIC has the capability to save a picture on the screen in a file, but suppose you want to do the same thing in MBASIC under CP/M. Thanks to the addition of two new transmit escape sequences, saving the screen under MBASIC is possible. In addition to the H19's Transmit Screen (ESC #) and Transmit 25th Line (ESC]) sequences, the H/Z-100 adds Transmit Current Line (ESC ↑), and Transmit Character at Cursor (ESC —). When you use the Transmit Current Line feature, the computer transmits all of the characters on the line where the cursor is as if you were typing them, including any escape sequences required to reproduce the line just as it is. We can use this capability to generate an MBASIC subroutine for saving the screen.

```
50000 OPEN "O",1,"PICTURE"
50010 E$=CHR$(27):REM  REMOVE THIS LINE IF E$ DEFINED
50020 PRINT E$"j";:REM  SAVE CURSOR
50030 PRINT E$"H";:REM  HOME CURSOR
50040 FOR I=1 TO 24:L$="":PRINT E$"^";:REM  TRANSMIT LINE
50050 L$=INPUT$(80):REM  INPUT TRANSMITTED LINE
50060 FOR J=1 TO 100:NEXT J:REM  WAIT A LITTLE BIT
50070 IF I<24 THEN PRINT #1,L$:PRINT:ELSE PRINT #1,L$;
50080 NEXT I:PRINT E$"k";:REM  RESTORE CURSOR
50090 PRINT #1,E$"A";:CLOSE #1
50100 REM RETURN:REM  REMOVE FIRST REM IF SUBROUTINE
```

If you MERGE this program to the first one in this article, it will produce a file called PICTURE, which, when TYPEd at the CP/M prompt, will produce a display like the one produced by the original program. The disadvantage of this routine is that lines containing a lot of escape sequences will have considerably more than 80 characters, and thus will be truncated. A more sophisticated method would be to input each line, count the escape sequences, and input the lines again with the number of characters input (by INPUT$) adjusted. You cannot use LINE INPUT to input the lines because MBASIC does not accept escape characters with LINE INPUT. In faster languages, you can input the characters one at a time, and test for the RETURN character that the H/Z-100 sends after each line.

You can also use this technique to send screen information to a printer, by removing lines 50000, deleting all but the CLOSE statement from line 50090, and replacing line 50070 with

50070 LPRINT L$:IF I<24 THEN PRINT

However, the printer (even an H/Z-25) will not know what to do with the escape codes, so they should probably be filtered out.

Special Keyboard Controls

There are several special keyboard controls available on H/Z-100 computers. Some of them are accessible through escape sequences, some by writing a special code to port 0F5H (245 decimal), and some are accessible either way. Here is a chart of these features.

Feature	Port 0F5H Code	Escape sequence
Reset keyboard	00	none (ESC z)
Auto repeat on	01	ESC y<
Auto repeat off	02	ESC x<
Key click on	03	ESC y2
Key clock off	04	ESC x2
Clear key buffer	05	none
Click	06	none
Beep	07	none (CHR$(7))
Enable keyboard	08	ESC (
Disable keyboard	09	ESC)
Key up/down mode	10	ESC x@
Normal key mode	11	ESC y@
Enable key interrupts	12	none
Disable key interrupts	13	none
Non-blinking cursor	none	ESC x;
Blinking cursor	none	ESC y;
Expand keyboard	none	ESC x?
Disable keyboard exp.	none	ESC y?

For example, in BASIC you can disable auto repeat with

```
PRINT CHR$(27)"x<";
```

or with

```
OUT 245,2
```

The Reset Keyboard function performs only those portions of an ESC z reset that affect the keyboard. The screen is not affected. The click produced by outputting a 6 to port 245 is a very short beep, as produced by each key when key click is enabled. It can be used to add a few more sound effects to games in addition to the beep you may be getting tired of. Here is a sample sound producing program.

```
10 FOR I=1 TO 80:OUT 245,6:FOR J=1 TO 10:NEXT J:NEXT I
15 FOR I=1 TO 50:NEXT
20 FOR I=1 TO 60:OUT 245,6:FOR J=1 TO 15:NEXT·J:NEXT I
```

It's not too thrilling, but at least you can add a "raspberry" sound to games for when the player looses or makes a mistake. You can also combine the clicks more closely to produce beeps of any duration, for morse code practice programs, etc.

In the key up/down mode (event mode), each key produces a code when pressed and a different one when released. If keyboard expansion is not also enabled, the codes are in the form ESC hNN, where NN is a two digit number. The number is the ASCII equivalent of a hex number, and the number produced when a key is released will be the same number when it is pressed plus 80H. For example, a key might produce ESC h45 when pressed, and ESC hC5 when released. When keyboard expansion is enabled, each key (including control keys) produces a single 8-bit code. When both keyboard expansion and the event mode are enabled, each key produces a code with the high bit reset (zero) when pressed, and the same code with the high bit set (one) when released. The exact codes produced by each key in the various modes are listed in the H/Z-100 Technical Manual. Here is a sample MBASIC program to illustrate the event mode.

```
10 E$=CHR$(27)
20 PRINT E$"x@";:REM  ENABLE EVENT MODE
30 A$=INPUT$(4):REM  INPUT A KEY.  USE INPUT$(2) FOR ZBASIC
40 IF RIGHT$(A$,2)="45" THEN PRINT "SPACE BAR PRESSED"
50 IF RIGHT$(A$,2)="C5" THEN PRINT "SPACE BAR RELEASED"
60 IF RIGHT$(A$,2)="96" THEN 80:REM  EXIT ON "X"
70 GOTO 30
80 PRINT E$"y@";:REM  DISABLE EVENT MODE
```

This program will tell you when you press the space bar, and when you release it. It exits if you press and release the X key.

In ZBASIC, the first two characters of the escape sequence are "dumped", so you must use INPUT$(2) instead of INPUT$(4) to read the keys.

This article only samples some of the special things that can be done on the H/Z-100 computer. I hope many of you reading this will begin to write programs that fully exploit the power of the H/Z-100.

HRUN Update

Pat Swayne
Software Engineer

Now that HUG is releasing HDOS software in soft sector format, I thought it would be a good idea to write an update article on HRUN (*For those of you who are newcomers to HUG, HRUN is a CP/M program that emulates the Heath Disk Operating System. Many programs in the HUG Library are written for HDOS. For more information on HRUN, see the February 1983 issue of REMark.*) for those of you who will be using it to run HDOS programs on H/Z-100 computers, etc. Response to HRUN has been extremely good, with very few users having any problems. However, I found a "bug" in it myself, in the .NAME SCALL processor, which returns the name of a file when you give it the channel number on which the file is opened. (By coincidence, real HDOS also has a bug in the .NAME processor, though not the same bug.) The problem is that if the name uses a full eight characters, HRUN does not put a trailing zero on the end. An easy fix is to shorten the name of the file or program affected to 7 or less characters (not counting any extension). I know of only one program affected by this bug, EZITRANS (HUG p/n 885-8007), which will not run in the Configuration mode unless you shorten the name to EZITRAN or something. EZITRANS has another problem with HRUN that I have not figured out yet. It will not copy files to or from SY0:. On other drives it works OK.

There are a number of programs (from non-HUG sources) that will not work on the H/Z-100 version of HRUN because they use RST 6, which is reserved for debugging in the H/Z-100 version. Among these programs are the PIE editor and others from the Software Toolworks. HRUN can easily be patched to accommodate those programs with this patch.

```
A>DDT HRUN.COM
NEXT  PC
2000 0100
-A201
0201 LXI H,202E
0204 .              (Type a period)
-^C                 (Control-C)
A>SAVE 31 HRUN.COM
```

This patch should only be applied to the H/Z-100 versions, which are called HRUN100.COM and HRUN100T.COM on the distribution disk.

Compatibility of HUG Programs

Here are some general guidelines to help you determine which HUG programs will run under HRUN. If the program is listed as "for hard sector disks only", it probably will not work with HRUN. If it uses the HDOS type-ahead buffer (SUBMIT programs do this), it will not run (which is why HRUN has its own SUBMIT program). And, of course, if the program tries to do anything with the H8 LED display, it will not work.

Here is a rundown of the disks that have been released so far in soft sector format, with the programs that will run under HRUN listed for each disk.

885-1029 - All of the programs work.

885-1060 - On this disk, only the FDUMP program will work with HRUN.

885-1062 - The MEMTEST and DSM programs will work properly. The DUMP program will work only in the file mode, and the track and sector displayed will not be accurate.

885-1067 - All of the programs will work.

885-1071 - This is the HUG Small Business Package, and the HDOS version of Microsoft BASIC is required to run it.

885-1086 - Now CP/M users can experience the power and speed of Tiny Pascal.

885-1089 - MACRO, IHEX, IABS, and TAB2SPC will work.

885-1090 - AH will work. HPLINK will work on H8 and H/Z-89 computers only (not H/Z-100). MBSORT and MBSORTV will work if you remove the lines CODE PIC and DW EXIT from the .ASM files and replace them with a specific ORG nnnn, where nnnn is an address in high memory where you wish the program to reside. Do not combine the programs with RELOC.

885-1097 - These programs will work, providing you have HDOS Microsoft BASIC.

885-1108 - HDOS Microsoft BASIC required.

NOW 12 MEGABYTE

(CDR-10M) $2995

WINCHESTER SYSTEM

For the Heath/Zenith Computer

Systems complete with software, case, power supply, signal cable and interface.
Runs with CP/M, on the H/Z89 & H8 (with Z80 card).

- Switching power supply
- Expansion for backup installations
- Auto attach BIOS
- Hard disk utilities
- Formatting program
- 1 year parts & workmanship warranty

CP/M is a trademark of Digital Research. Heath, H8, H89 are trademarks of Heath Corporation. Zenith, Z89, Z90 are trademarks of Zenith Data Systems.

5-20 day delivery.
Pay by check, C.O.D., Visa, or M/C.

Contact:
C. D. R. Systems Inc.
7210 Clairemont Mesa Blvd.
San Diego, CA 92111
Tel. (619) 560-1272

THE H-100 SERIES: THE WORLD'S FIRST 16-BIT COMPUTER KITS.

with authorized sales and service *only* through the Heath Company and Heathkit® Electronic Centers.*

The world-famous H-100 Series Heath/Zenith computers are the first to give you advanced 16-bit computing at a kit price. Many who have never considered kitbuilding are now interested because most circuit boards are prewired, making the H-100 Series our simplest computer kits. Dual microprocessors deliver 16-bit speed and 8-bit compatibility. The industry standard S-100 card slots allow a host of peripherals and memory expansion to 768K RAM. And our stores and catalogs have the software to help you take full advantage of the faster 16-bit operating speed.

When you're ready to move up to H-100 Series, remember: these computers are available *only* through the companies that give you solid service and a trustworthy warranty... Heath and Heathkit® Electronic Centers.* The companies that stand by their promise of support to kitbuilders. The companies with the pledge: *"We won't let you fail."*

Many companies would like to sell our product but no other dealer or vendor can resell the H-100 Series without voiding the original factory warranty. When buying the world's first 16-bit computer kit, buy only from the world leader in electronic kits.

Authorized sales and service available only through the Heath Company and your...

Heathkit®
ELECTRONIC CENTER*

*Heathkit Electronic Centers are units of Veritechnology Electronics Corporation.

VEC-770

SuperCalc As A Diet Manager

Janet M. Johnson
1413 Hillcrest Drive
Blacksburg, VA 24060

"Watch the calories, eat less sugar, eat less cholesterol and saturated fat", are reminders many people think about every time they sit down for a meal or a snack. Those who must restrict the amount of fat or calories they eat or who want to be sure they are eating a good diet spend a great amount of time keeping records.

The computer is a helpful and efficient tool for people who carefully monitor what they eat. There are many programs on the market designed to evaluate and plan diets. Generally, the diet programs contain a data bank of food compositions used to calculate the nutritional value of foods eaten in a day and compare with a target amount. Dietary management programs are very useful for clinical dietitians who plan many different diets each day and teach dietary management. However, a spreadsheet may be adapted to do the same calculations as a dietary management program.

I use a SuperCalc template to teach dietary problem solving to students of college level food and nutrition courses. One major advantage of using SuperCalc rather than a dietary management program is that I do not have the expense of another piece of software for my Z-89 computer, or the expense of purchasing more software for the students to use in the micro lab at the university. The use of SuperCalc is also a practical exercise for the students who are training to do nutritional counseling in homes and public schools and will not have access to the dietary management programs generally found in only large hospitals and universities.

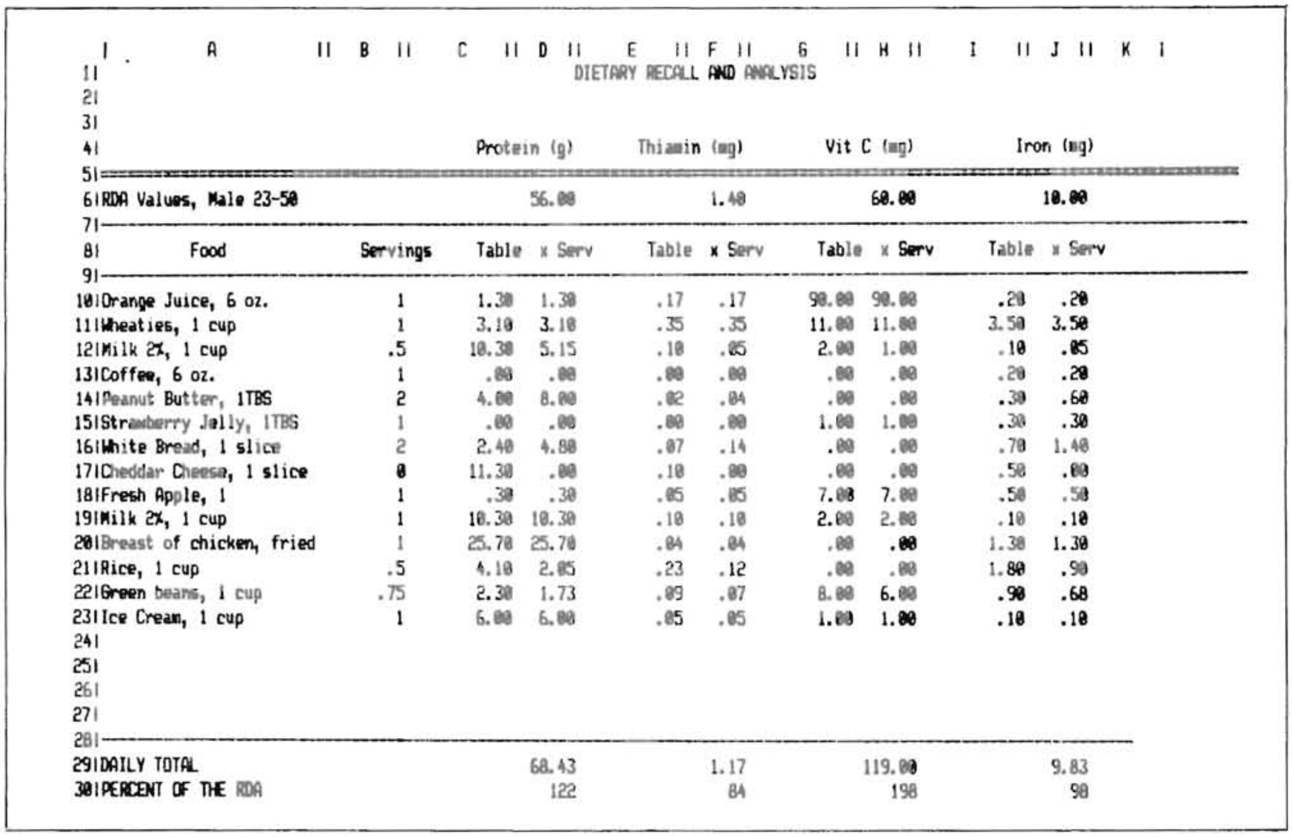
Another advantage of using the SuperCalc template for dietary evaluation, is that any food composition table may be used as the basis of the data. Nutritionists who have studied food composition data banks find some inconsistencies with standard references and have designed models for evaluation of the data bank. The user of the SuperCalc dietary template must look up each food item in a food composition table. However, the user selects the source of data such as a table from a physician or a government publication. There is a listing of some of the available food composition tables at the end of this article.

Preparation of the Format

I adjusted column A to a width of 25 spaces by the command /F,C,25 to allow space to list the food and the reference size serving from the food composition table. I did not change the width of column B since the 9 space width was adequate for listing the number of servings.

I paired columns C through J to list the table value of a specific nutrient and the amount of the nutrient actually consumed. The format of a column of 12 spaces followed by a column of 7 spaces places the columns in pairs and shows the relationship of the two columns. For example, I expanded column C which lists the grams of protein found in the food composition table by the command /F,C,C,12, and decreased Column D which lists the grams of protein actually eaten by the command /F,C,D,7. I continued this pattern to column J to list four nutrients.

The first row contains the title of the spreadsheet beginning at column E. For easier reading, I centered the titles above the pairs of columns on row 4 by starting in the first column with a text command and a 4 or more space indentation, and allowed the title to overlap into the second column. For example, in column C, I entered the command " PROTEIN (G) which overlapped into column D. The lines were drawn with an equal sign at row 5 and with a hyphen at row 7 and 9 to separate the headings. I listed the target amount which was the Recommended Dietary Allowance (Recommended Dietary Allowances, 1980, National Academy of Sciences, Wachington, D.C.). I entered the values for each nutrient in row 6 in the second column of the appropriate pair. I centered the title



```
  | .       A          || B ||    C    || D ||    E    || F ||    G    || H ||    I    || J ||  K  |
 1|                                           DIETARY RECALL AND ANALYSIS
 2|
 3|
 4|                                   Protein (g)     Thiamin (mg)       Vit C (mg)         Iron (mg)
 5|==================================================================================================
 6|RDA Values, Male 23-50                   56.00             1.40             60.00              10.00
 7|--------------------------------------------------------------------------------------------------
 8|        Food             Servings    Table  x Serv    Table  x Serv    Table  x Serv     Table  x Serv
 9|--------------------------------------------------------------------------------------------------
10|Orange Juice, 6 oz.          1       1.30   1.30      .17    .17     90.00  90.00       .20    .20
11|Wheaties, 1 cup              1       3.10   3.10      .35    .35     11.00  11.00      3.50   3.50
12|Milk 2%, 1 cup              .5      10.30   5.15      .10    .05      2.00   1.00       .10    .05
13|Coffee, 6 oz.                1        .00    .00      .00    .00       .00    .00       .20    .20
14|Peanut Butter, 1TBS          2       4.00   8.00      .02    .04       .00    .00       .30    .60
15|Strawberry Jelly, 1TBS       1        .00    .00      .00    .00      1.00   1.00       .30    .30
16|White Bread, 1 slice         2       2.40   4.80      .07    .14       .00    .00       .70   1.40
17|Cheddar Cheese, 1 slice      0      11.30    .00      .10    .00       .00    .00       .50    .00
18|Fresh Apple, 1               1        .30    .30      .05    .05      7.00   7.00       .50    .50
19|Milk 2%, 1 cup               1      10.30  10.30      .10    .10      2.00   2.00       .10    .10
20|Breast of chicken, fried     1      25.70  25.70      .04    .04       .00    .00      1.30   1.30
21|Rice, 1 cup                 .5       4.10   2.05      .23    .12       .00    .00      1.80    .90
22|Green beans, 1 cup         .75       2.30   1.73      .09    .07      8.00   6.00       .90    .68
23|Ice Cream, 1 cup             1       6.00   6.00      .05    .05      1.00   1.00       .10    .10
24|
25|
26|
27|
28|-----------------------------------------------------------------------------------------------
29|DAILY TOTAL                                  68.43             1.17            119.00               9.83
30|PERCENT OF THE RDA                             122               84               198                 98
```

FOOD in cell A8.

The titles above the pairs of columns B8 through J8 were adjusted in an unconventional manner so that the word TABLE and X SERV would appear centered above the correct column. If TABLE was centered above C8, it overlapped into D8 blocking the title X SERV which I wanted to appear above D8. To solve the problem, I typed the last characters of the preceeding title after the text command and the first characters of the next title at the right hand side of the column. For example, cell B8 was typed " (4 spaces) SERVI. Then the next cell C8 was typed "NGS (5 spaces) TABL. The pattern continued in this manner:

D8 = "E (2 spaces) X (1 space) SE
E8 = "RV (6 spaces) TABL
F8 = "E (2 spaces) X (1 space) SE
G8 = "RV (6 spaces) TABL
H8 = "E (2 spaces) X (1 space) SE
I8 = "RV (6 spaces) TABL
J8 = "E (2 spaces) X (1 space) SERV

The values of the nutrients range from whole numbers greater than one to a fraction of one. To set the decimal to two places, I used the $ format in columns C through J. The command for column C was /F,C,C,$.

Preparation of the Formulas

The first step of formula preparation for the spreadsheet was to decide how many rows would be needed to list the foods consumed in a day. I arbitrarily selected row 30 as the bottom of the table. For longer lists of daily food intake, I would select a row farther down the sheet.

The first column of each pair of columns listed the amount of the nutrient found in the food composition table. The second column of each pair of columns listed the adjusted amount of the nutrient when multiplied by the number of servings eaten. For cell D10, I used the command C10*B10. The formula was replicated in column D by /R,D10,D11:D27. This two step procedure was repeated for columns F, H, and J.

The formula in cell D29 was written by the command SUM(D10:D27). I replicated the formula for row 29 by the command /R,D29,F29:J29. This also totaled columns G and I which are meaningless numbers and only confuse the student, so I blanked the cells G29 and I29 with the commands /B,G29 and /B,I29.

The numbers in row 30 show the percent of the target amount consumed. I listed the appropriate Recommended Dietary Allowance at the top of the table which is row 6. To calculate the percentage of the Recommended Dietary Allowance of protein in cell D30, I entered the command D29/D6*100. I replicated the formula for row 30 by the command /R,D30,F30:J30. Since there are no numbers in cells G6 and I6, ERROR messages appear in cells G30 and I30. I erased the ERROR by the command /B,G30 and /B,I30. I used the integer format for row 30 by the command /F,R,30,I. The integer format rounded the percentage numbers to the nearest whole numbers.

Entering Data and Printing

With the format complete, I entered the foods consumed and the nutrient values of the foods from the USDA Agricultural Handbook No. 456. I filled in the table very quickly and I had the assurance that there were no mathematical errors in adjusting for serving sizes and calculating the percent consumed of the Recommended Dietary Allowance. The same format may be used for other food values the user wants to monitor such as calories, sodium, cholesterol or fat, and compare with a daily goal.

To save the time of entering a list of new foods eaten each day, I listed all the foods often consumed by the person. If the item was not on the diet on that particular day, I entered a 0 in the servings column. The nutritional value of the food was conveniently listed but not included in any of the calculations. An example of a food placed in the list but not eaten that particular day is cheddar cheese.

To print the spreadsheet, I set the printer at maximum compressed print to fit more than 80 columns on 8 1/2 inch wide paper. The compressed print displayed four nutrients on the page.

Use of the Spreadsheet

The diet spreadsheet has the advantage of adapting commonly available software for monitoring diets. Another advantage is that any food composition list can be used for the computing rather than a data bank supplied with the program.

A diet spreadsheet is helpful for anyone who must keep a food record to carefully control the intake of calories, sodium, fat or any other nutrient.

Food Composition Tables

1. Food Values of Portions Commonly Used, 1975. C. F. Church and H. N. Church, J. B. Lippincott Company, Philadelphia.

2. Nutritive Value of American Foods, 1975. Agricultural Handbook No. 456, Agricultural Research Service, United States Department of Agriculture, Washington, D. C. Order from Superintendent of Documents, U.S. Government Printing Office, $5.15.

Z-100

ZDS Offers 16-bit, Software Directory

(NOTE: The following material was taken from the Zenith Data Systems Newsdate, May-June 1983 Issue.)

"How much 16-bit software is available for the Z-100?" is probably one of the questions Zenith Data Systems dealers hear most often. The right answer could make or break many a sale.

To help answer that question, and to serve as an important resource aiding dealers in meeting their customers' needs, Zenith Data Systems has compiled a Z-100 Software Directory.

The Directory lists more than 230 16-bit programs that run under the Z-DOS operating system. It is available from distributors and has a suggested retail price of $25.

The new Directory contains a complete description of each software program listed that includes:

- vendor name, address telephone;
- minimum memory;
- programming language;
- source code, if provided;
- disk size and format; and
- a complete summary of product features.

"This new Directory was created to help dealers and users locate Z-DOS based software for the Z-100 that will meet their specific needs," says Robert K. Reid, vice president of marketing for ZDS. "Its size clearly indicates how large the library of such products has become since the Z-100's introduction last September."

The Directory comes in a convenient three-ring binder and is divided into several application-oriented sections representing both horizontal and vertical business uses.

Plans are underway to update the directory as early as October.

AT ZENITH DATA SYSTEMS

WE'RE TAKING QUALITY TO NEW DIMENSIONS.

Our Engineers, Committed To Excellence, Insist On It!

Zenith's commitment to quality has become an American tradition. And a good look at our new desktop computer will tell you why.

Our engineers have designed the new Z100 series utilizing the latest in 16-bit technology, high resolution graphics, networking and ergonomic design. That kind of project requires "hands-on" control from the drawing board to the loading dock ... and beyond.

If you're looking for a sophisticated technical environment with high visibility and a tradition of engineering excellence look to Zenith Data Systems. We still put the quality in first.

HARDWARE DESIGN ENGINEERS

Candidates should have a BSEE and at least 3 years experience in hardware design of 8-bit and 16-bit (8085, 8086, 6800) based computer and terminal systems. Programming experience in Assembly language is desired. Responsibilities will include digital and/or electronic circuit design, design evaluation and system test.

SOFTWARE ENGINEERS

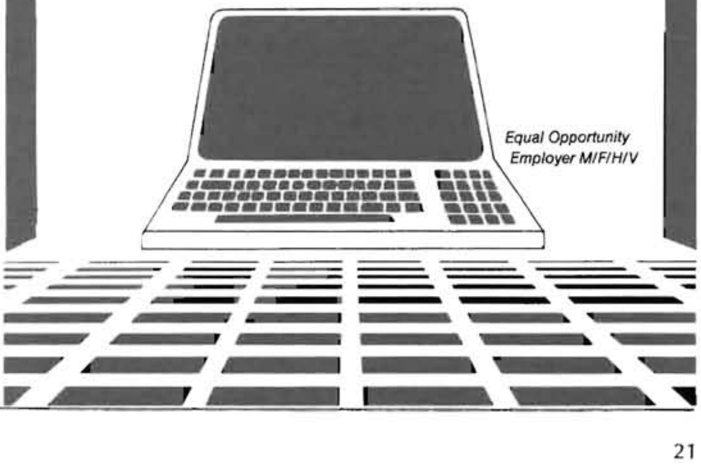
Engineers needed for enhancement of operating systems for microcomputer products. Responsibilities include utility design, adaptation of existing products to microcomputer systems, and reviewing/specifying hardware interfaces. BSCS (or equivalent degree), demonstrated knowledge of assembly level language (8080/Z80, 68000/Z8000/8086) and experience with operating system development (preferably CP/M, HDOS, and UNIX) is required.

While experiencing a satisfying professional career, you will also enjoy the affordable lifestyles of St. Joseph, Michigan. Only 90 minutes from Chicago, St. Joseph is a picturesque community on the shores of Lake Michigan. In addition we offer competitive salaries and excellent benefits including relocation assistance and the opportunity to work on your Master's Degree through an on-site University extension program.

For immediate consideration, call Duane Trombly, 616/982-3504, or send your resume to Zenith Data Systems, Dept. REM, Hilltop Road, St. Joseph, Michigan 49085.

ZENITH | data systems

A Wholly Owned Subsidiary of Zenith Radio Corporation



Equal Opportunity Employer M/F/H/V

What's a Buffer?

Jennifer T. McGraw
12741 SW 68th Terrace
Miami, FL 33183

When I first bumped into home computers, I kept hearing strange words, or familiar words used in strange ways. The first time 'bawd rate' descended on my ears, I got this strange picture in my mind. I mean really, all these serious type people using an old-fashioned word like that, and what did bawds have to do with microprocessors anyhow? I soon learned how to spell it, and have a general idea of it's meaning, and a lot of practical experience when the baud rate of the printer is not what the system is expecting. Anyway, there were a lot of words like that.

The word that confused me the most was 'buffer'.

Now, I knew what a buffer was. The Maginot Line was a buffer. It's a place that's supposed to absorb the impact of a blow. It's a neutral zone between two disagreeable countries, etc. People tried to explain it to me in connection with computers, but it was all very hazy, until I ran head on into the Random File Buffer in Microsoft BASIC. Six months later, I finally understood.

A buffer is any place in memory where data is held until the program or hardware is ready to do something with it, and is especially used with reference to data being sent to or received from a peripheral device, like the disk, printer, or terminal. Buffers are used chiefly because the CPU and allied circuitry are generally much faster than any of the peripherals and the use of the buffer prevents lost data.

Almost every program has a buffer of some sort, whether you are aware of it or not. Operating Systems, which are programs, don't forget, use them fairly frequently. HDOS uses 256 byte buffers while CP/M uses 128 bytes. The H/Z-19 terminal has a buffer. When that buffer gets filled to a certain point, the terminal sends a busy signal to the computer and if it overflows, the terminal complains by 'beeping' and gets even by losing characters. (This is why the Christmas Graphics program had to be changed slightly for CP/M. See the March REMark.)

The HDOS buffers can sometimes make you feel that nothing is happening. Try sending less than 256 bytes to your disk or printer. The device doesn't do anything until you close the channel; then HDOS fills in with nulls (zeroes), and sends them all at once. CP/M sends as soon as you hit a RETURN.

Try the following:

```
HDOS   >PIP SY0:TEST.DAT=TT:
       NOW IS THE TIME FOR ALL GOOD MEN
       TO COME TO THE AID OF THE PARTY
       ^D

CP/M   A>PIP A:TEST.DAT=CRT:
       NOW IS THE TIME FOR ALL GOOD MEN
       TO COME TO THE AID OF THE PARTY
       ^Z
```

With HDOS, nothing happens until the CTRL D is entered. With CP/M, each RETURN causes action. You can, of course, leave out the drive designator if sending to the default or boot drive. You will end up with a file on the disk named TEST.DAT.

If you have a printer, send the above like so, assuming that your printer driver under HDOS is named LP: or that CP/M is configured as LST:=LPT:.

```
HDOS   >PIP LP:=TT:
CP/M   A>PIP LST:=CRT:
```

In case you haven't read your manuals, the above command lines run the program PIP which translates the whole thing as 'Make the line printer (or a disk file) equal what is typed on the terminal'. In HDOS, you can use the command 'COPY' instead of 'PIP'. HDOS looks up COPY in SYSCMD.SYS and finds out that it means 'RUN PIP', which is very helpful to people who can remember the word COPY, but slows things down just a little.

Enough on system buffers. Now to the buffer that gave me all the problems, but also made it clear to me what a buffer is, in computer parlance.

For each Random file which you open, MBASIC establishes a buffer, a storage place in RAM. Under HDOS MBASIC (Version 4.82), these buffers are 256 bytes each. CP/M MBASIC (Version 5.21), has a default buffer length of 128 bytes which can be changed. See your manual. All action to and from the random files goes through this buffer. This is very important to remember. You are dealing with the buffer. MBASIC takes care of the file. Try this immediately after loading MBASIC:

```
10 REM  TEST RANDOM FILES'       TESTRAN.BAS
20 OPEN "R",1,"TEST.DAT"
30 FIELD #1, 128 AS R$
40 PRINT R$
50 CLOSE
```

And there you are with some of the opening message from MBASIC. What happened? When the file was opened, MBASIC assigned a series of addresses as the buffer for the file. We gave that bunch of bytes the variable name R$, and when the interpreter was told to print R$, it grabbed the information in that piece of RAM and printed it. Apparently, when signing on, MBASIC used that space for information it was going to use only once. Since we made no changes to the buffer, R$ equals parts of the sign-on message.

Now change the program slightly.

```
40 R$="ONCE UPON A TIME"
50 PRINT R$
60 CLOSE
```

Now what's going on? We still have some of the message, because (pause for great light), we didn't change the whole buffer. That, friends, is where LSET and RSET come in. Those commands tell MBASIC to take a fixed-length variable as defined by the FIELD statement, give it the assigned characters, and fill in the rest of it with spaces. Change line 40 and run it.

```
40 LSET R$="ONCE UPON A TIME ETC.
```

We're getting somewhere. Let's create a file of some length and at the same time, try to show why anyone would want to use random files. After all, those spaces take up room on the disk.

```
40 FOR J=1 TO 5 : READ N$(I) : NEXT
50 DATA ONE,TWO,THREE,FOUR,FIVE
60 FOR J=1 TO 5
70 LSET R$=STR$(I)+". THIS IS LINE "+N$(I)+"."
80 PUT #1,J
90 NEXT
100 PRINT R$
110 GET #1,3
120 PRINT R$
130 CLOSE
```

What you should now have printed on your screen is line five and then line three. Notice: You didn't change R$. You told MBASIC to fill the buffer with record #3, (or sector #3), and that changed R$. This is the beauty of a random file. You can get at a particular record (sector) at once. No need to run through the whole file as you have to do with serial files, using up all the memory searching for one record. The only memory used is the little bit for the buffer. It's a tradeoff. Serial files use less disk space, random files use less memory.

Change the program again.

```
100 FIELD #1, 5 AS R1$, 5 AS R2$, 5 AS R3$
110 GET #1,4
120 PRINT R1$ : PRINT R2$ : PRINT R3$
130 PRINT R$
140 CLOSE
```

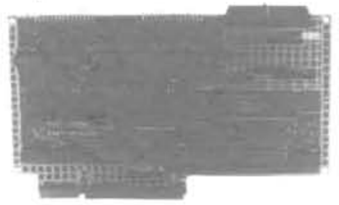
And furthermore, you can split one buffer apart as many different ways as you want to, so long as you don't exceed the size of the buffer. Just remember, each FIELD statement starts counting from position one in the buffer. I have a program that has five different field statements for one file. The variables used refer only to particular sectors. That, folks, is a REAL buffer.

For a deeper look at MBASIC files, read Doc Campbell's article in REMark Issue #10. I have also heard that a book put out by Wiley, "Data File Programming in BASIC", is very good.

H/Z89 PERIPHERALS from SECURED COMPUTER SYSTEMS

PORT SERIAL
2/3 —CARD— I/O
PORT PARALLEL

" . . . not your typical vanilla-flavored serial and parallel interface . . . "



Features:

Chip independent design • Reduces computer data buss loading from 3 to 1 • Choice of Centronics or Epson parallel drivers for HDOS or CP/M • Complete documentation and installation instruction.

• 2 Serial Ports
• Supports: Ring Input, External Clock, Auto Dialer
• 3 Port Parallel with 2 Level Interrupt Control
• Fully compatible with all models of H/Z 88, 89, 90 using CP/M or HDOS.
• Fully tested, 90 day warranty, two serial cables and a parallel cable (internal to computer) and software driver.

PRICE $199.00
Shipping & Handling $10.00

16K RAM EXPANSION CARD

Expands your H/Z89 RAM Memory capacity to a FULL 64K!

Fully compatible with:
H/Z 89 • H/Z 88 • Magnolia Microsystems CP/M and disk drive I/O interface cards

NOW INCLUDING SUPPORT MOUNTING BRACKET

Featuring:
Complete installation instruction • 90 day Warranty
Field reliability record now entering its 21st month

Now Only $65.00

HDOS is a registered trademark of Heath Company
CP/M is a registered trademark of Digital Research

Shipping & Handling $5.00

PRICES ARE LESS SHIPPING AND TAX IF RESIDENT OF CALIFORNIA
MAIL ORDER: 12011 ACLARE, CERRITOS, CA 90701 (213) 924-6741
TECHNICAL INFO/HELP:
8575 KNOTT AVE., SUITE D, BUENA PARK, CA 90620 (714) 952-3930
Terms and specifications subject to change without notice.

ZENITH data systems
SERVICE CENTER

LPRINT in CP/M MBASIC

Terry L. Jensen
Software Developer

(NOTE: This discussion applies only to Heath/Zenith CP/M.)

Introduction

How many times have you wished you could output the same information to the terminal and printer without duplicating the PRINT and LPRINT statements of a CP/M BASIC-80 program? How many of you have wanted to print to two printer devices from the same program but BASIC-80 won't let you? Solutions to these problems will be presented in this article.

In order to accomplish these special LPRINT commands, there are three areas that we will cover. We need to look at two parts of CP/M: 1) the IOBYTE, and 2) the PORT address for the list device. We also will need to run CONFIGUR to prepare CP/M for two printer devices.

The sample programs below show routines that may be built on and added to any of your MBASIC programs. You will need to have some familiarity with the logic of the BASIC language, as well as understand the commands of CP/M MBASIC. I also must assume that you have a reasonable concept of DECIMAL, HEXADECIMAL, OCTAL, and BINARY number bases.

These programs will work on the H8 with the four port serial I/O board, and the H/Z89 or Z90. A solution for LPRINTing to the CRT: and printer device with the H/Z-100 series under CP/M-85 will be presented also. CP/M must be CONFIGURed for your printer device. In order to output to two printers, your computer will need to have two serial DCE ports. (This article will not discuss using parallel ports.)

IOBYTE

Let's look first at setting up the LPRINT statement to print to the screen and then back to the printer device. To accomplish this, we need to understand a little about the IOBYTE of CP/M.

The IOBYTE is a term which describes the function of controlling the reassignment of physical and logical devices. The IOBYTE actually maps (points) the logical (defined) device to the physical (real) device. That means if we reassign the LIST device of the IOBYTE to the screen (CRT:) instead of the printer (LPT: or UL1:), the output of the LPRINT statement will be sent to the screen.

Here is the sample program for using the LPRINT command of MBASIC to output to the screen and then switch back to a Diablo printer device:

```
10 POKE 3, PEEK(3) AND 63 OR 64
20 LPRINT "THIS GOES TO THE CONSOLE"
30 POKE 3, PEEK(3) AND 63 OR 192
40 LPRINT "THIS GOES TO THE DIABLO"
50 END
```

For those of you not familiar with the IOBYTE, lets see if we can understand what we have done in this sample program.

Take out your CP/M binder, you know the thing only "long haired" programmers read. We will be looking at the "Alteration Guide" in "The BIOS Entry Points". Under the "characteristics of each device" (page 15 in my Guide), we see the CONSOLE, LIST, PUNCH, and READER devices described briefly. Down just below that, a paragraph begins "For added flexibility, the user can optionally implement the 'IOBYTE' function ...".

Here we find that the location of the IOBYTE is at 0003H in memory. The definition of the eight bits of the IOBYTE byte are shown, i.e. bits 0 and 1 are reserved for the CONSOLE, bits 2 and 3 for the READER, bits 4 and 5 for the PUNCH, and lastly, bits 6 and 7 for the LIST device. On the next page, we find a table which explains that these eight bits, when assigned a value, define a particular device. We, of course, are concerned with assigning the LIST device.

Note that in the LIST field, a "1" (01B) is defined as the CRT: or screen. A "2" (10B) is the LPT: (or line printer), and a "3" (11B) is the user defined list device, the Diablo printer. Please note, we must change the 6th and 7th bits only. We do NOT want to change any of the other bits of this byte. Therefore, the actual values we need are 64 (01000000B), 128 (10000000B), and 192 (11000000B) for the CRT:, LPT: and UL1:, respectively. Lines 10 and 30 of our sample program do just that.

In line 10, we PEEK the value in location 3 ANDed with 63 (00111111B), OR that with 64 (01000000B), and POKE that value back into location 3. The "AND 63" sets the 6th and 7th bits to 0, while the "OR 64" sets the 6th and 7th bits to 1 and 0, respectively. The CRT: is now defined as the LST: device.

To reset back to the printer device, we need to POKE the IOBYTE to the value of the printer device, which in this case is the Diablo printer or UL1: device. Thus we AND 63 (00111111B) and OR 192 (11000000B).

What would we need to POKE if we were using an H/Z-25? We would AND 63 (00111111B) and OR 128 (10000000B) to set the IOBYTE to the LPT: device, which is used for the H/Z-25.

That concludes the sending of output to the SCREEN and then back to the printer with the LPRINT command. What if we want to output to a Diablo and H/Z-25 from the same program? Can we just change the IOBYTE from the LPT: to UL1: and back again? That is part of it but, in addition, there are two other steps we must take.

PORT Address

If we have two printers as part of our system, then each printer device is connected to a different PORT address. Changing the IOBYTE swaps only the LST: logical device. It does not change the PORT assignment of the device. We must do that by POKEing the appropriate PORT value into memory.

Lets look at the next program:

```
10 PORT=PEEK(2)*256+PEEK(1)+58
20 POKE PORT,&O320
30 POKE 3,PEEK(3) AND 63 OR 128
```

```
40 LPRINT "THIS GOES TO THE H/Z-25"
50 POKE PORT,&O340
60 POKE 3,PEEK(3) AND 63 OR 192
70 LPRINT "THIS GOES TO THE DIABLO"
```

To find where the address of the PORT assignment is located, we need to take out the BIOS Listing for CP/M 2.2.03 and turn to page #020. The PORT assignment for the printer device is located at 003DH and defaults to 0E0H or 340 OCTAL. (Have you ever noticed when you run CONFIGUR that the LST: device has a default of 340 OCTAL?)

The 003DH (61 DECIMAL) is the address located in the BIOS listing. But we don't know where the BIOS is located in memory. However, we do know that the entry point to the BIOS is located at 001H and 002H in memory.

Line 10 of our program will set the MBASIC variable "PORT" to the address of the port assignment. The PEEK(2) is the higher byte and PEEK(1) is the lower. DECIMAL 58, rather than 61, was added to the lower byte to take into account the three byte jump instruction to COLD BOOT.

Lines 20 and 50 POKE the appropriate PORT value (i.e. 320 and 340 OCTAL) into this location. Thus we have the ability to toggle the printer PORT.

Lines 30 and 60 should look familiar to us. For an explanation, refer to our first program above.

CONFIGUR

Something may not seem right to you. By changing the IOBYTE and PORT assignment, we selected a new printer but our printers operate at different speeds, 1200 and 2400 baud for the Diablo and H/Z-25, respectively. The last step we must take is to initialize the ports for the proper BAUD rate for the printer that is connected there.

To initialize the ports, run CONFIGUR and select "Set Terminal and Printer Characteristics". Set the following:

B TTY: Baud rate: 1200 Port: 0E0H = 340Q
C LST: Baud rate: 2400 Port: 0D0H = 320Q

Exit with a "Y" and select "Change the Default I/O Configuration" next. Set the following:

D LST: = LPT: Available TTY: CRT: LPT: UL1:

Exit with a "Y" and then another "Y". I chose the LST: device as port 320 and baud 2400 because the H/Z-25 requires special handshaking. The UL1: device is defined as the DIABLO and includes its own handshaking routines. (Note: I am assuming you know to configure the "Printer Ready Signal" and Baud rate for your H/Z-25.)

We have learned how to use the LPRINT statement to print to two printers. With these two examples, we can build any number of combinations of LPRINTIng to the CRT:, LPT:, and UL1:.

Here is another example program. This program will LPRINT to like printers, in this case two H/Z-25's.

```
10 PORT=PEEK(2)*256+PEEK(1)+58
20 POKE PORT,&O320
30 LPRINT "THIS GOES TO ONE H/Z-25"
40 POKE PORT,&O340
50 LPRINT "THIS GOES TO THE OTHER H/Z-25"
```

In this program, we do not have to change the IOBYTE because the printers use the same LST: device. We do have to change the PORT value though. The port baud rates must be initialized by running CONFIGUR as in the last example. (I assume you have configured CP/M for your H/Z-25 printer.)

(The examples above apply only to serial I/O ports. Parallel boards for the H8 and H/Z-89s exist. The standard H/Z-100s have one serial I/O port and one parallel port. It is beyond the scope of this article to talk about using the LPRINT command between serial and parallel ports.)

The last example is for H/Z-100 users for sending LPRINT output to the CRT: and printer:

```
10 PRINT "OUTPUT TO SCREEN OR PRINTER (S or P)";:A$=INPUT$(1)
20 IF A$="S" THEN 100
30 IF A$="P" THEN 200
40 GOTO 300
100 POKE 3,PEEK(3) AND 63 OR 64
110 LPRINT "THIS GOES TO THE CONSOLE"
120 REM set IOBYTE to printer status
130 POKE 3,PEEK(3) AND 63 OR 0
140 GOTO 300
200 LPRINT "THIS GOES TO THE PRINTER"
300 PRINT "DO YOU WANT TO QUIT (Y/N)";:A$=INPUT$(1)
310 IF A$="Y" THEN 400
320 IF A$="N" THEN 10
400 END
```

Note the difference in line 130. We ORed with "0" rather than 128 or 192 as done above. The reason is CP/M-85 automatically configures the printer as a TTY: device.

This program gives a practical application of how to use the IOBYTE within a program. You can apply this to the previous programs or think of your own application for using the LPRINT statement.

Conclusion

The concept here may not be understandable by everyone. With a little common sense, almost anyone can use these routines to send output to two devices using the BASIC-80 LPRINT command.

QUALITY FIRST

HEATH/ZENITH

HEATH/ZENITH OWNERS We now offer you 27 well documented easy to use software products developed over 3 years.

GAMES Choose arcade style graphics, adventure type, or intellectual (like new WORD WIGGLE). Many play on any system.

SPREADSHEET Full featured - now only $49.95

WORD PROCESSING 50,000 word proofreader, file difference detector and more.

LANGUAGES & UTILITIES Top-rated "C" Compiler (fastest in BYTE benchmark), Z80 macro assembler. Plus five utilities and more.

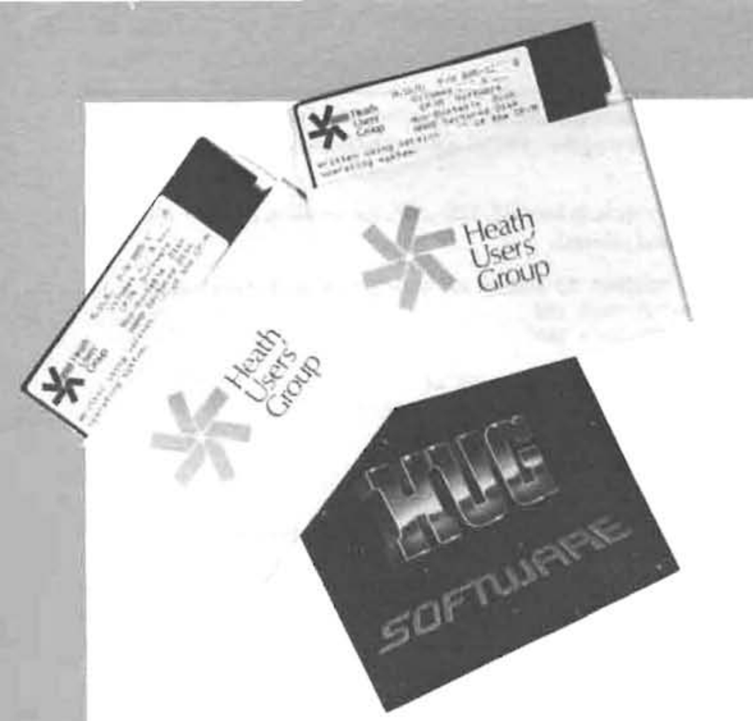
AVAILABLE through Heathkit stores and other Heath/Zenith dealers near you.

FREE CATALOG Call or write today for detailed 16 page catalog.

The Software Toolworks

(213) 986-4885
15233 Ventura Blvd.
Suite 1118, Sherman Oaks, CA 91403

Dealer inquiries invited.



HUG NEW PRODUCTS

885-1231[-37] CP/M Cross Reference Utilities for MBASIC $20.00

Introduction: This disk contains two programs which will provide the Microsoft BASIC programmer with tools for aiding in the development of their programs. The first is a VARIABLE cross reference utility, and the second is a BRANCH statement cross reference utility.

Requirements: The programs require the CP/M operating system version 2.2 or later on an H8/H19 or H/Z-89 or Z90 with a minimum of 32K of memory. A printer is not necessary but is recommended for a hardcopy listing of the output. Only one disk drive is required.

The programs will also run under CP/M-85 on the H/Z-100 unit with one drive.

The BASIC source file must be saved on the disk in ASCII form for XREF or BRNCHREF to process. Both utilities utilize the amount of memory available.

NOTE: The source code is included. The programs have only been tested on MBASIC versions 5.2 or greater.

The following programs are contained on the HUG P/N 885-1231[-37] CP/M Cross Reference Utilities for MBASIC:

```
README   .DOC
XREF     .COM
XREF     .ASM
BRNCHREF .COM
BRNCHREF .ASM
```

Author: Rudi Daniel
Additions and modifications by Pat Swayne
Output modified by Terry Jensen

Program Contents: The following descriptions will provide an explanation of the program options:

XREF - This program will scan a BASIC program for variables used and will print all the line numbers the variable is referenced in. The listing is in alphabetical order with the line numbers referenced in sequential order.

Variables to a length of 14 characters may be processed to allow for the lastest version of BASIC-80. The maximum number of variables is dependent on the amount of memory.

The BASIC file must be stored in ASCII form before XREF or BRNCHREF can process it.

BRNCHREF - This program will scan a BASIC program for branch statements and will print all the line numbers the branch references. Branch statements included are GOTO, GOSUB, THEN, ELSE, and RESUME.

The source file must be in ASCII form.

Comments: This program is a must for CP/M BASIC programmers. The output listings contain paging for long source files that contain a large numbers of variables and branch statements.

885-8018[-37] CP/M FAST EDDY Text Editor and BIG EDDY $20.00

Introduction: FAST EDDY is a text file screen editor that was written for everybody. It was written using the basic commands and keypad keys, so that anyone, even with no experience with an editor, can learn to use it while reading the instructions.

For those files that are too large for your computers memory, BIG EDDY will handle the breaking up of the text for editing with FAST EDDY.

Requirements: This disk requires the CP/M operating system version 2.2 or later, on a Z80 based H8/H/Z-19 or H/Z-89 or Z90 with 32K of memory. A printer is not required, but both FAST EDDY and BIG EDDY have printer options. Only one disk drive is required.

BIG EDDY can be used with large files. A second drive (or high density drives) may be required to break up large files which cannot fit into memory. The original file is not changed or deleted.

NOTE: This program is written in Z80 Assembly Language and can be used only on Z80 based H8 systems. It will NOT work on the H/Z-100 CP/M-85.

The following files are included on the HUG P/N 885-8018[-37] CP/M FAST EDDY Text Editor and BIG EDDY File Handling Utility:

```
EDITOR .COM
BIGED  .COM
TUTOR1.DOC
TUTOR2.DOC
TUTOR3.DOC
```

Author: Hubert L. Reeder

FAST EDDY - This text file screen editor and its documentation have been designed for anyone not familiar with using an editor. The program uses commands and keys that are easy to remember and use.

The editor contains a limited number of commands. However, the commands are designed to provide a useful, easy to use editor. It does not have complex options that require time and effort to use.

The editor contains a command mode and edit mode. The following is a brief list of the options:

COMMAND MODE

Typed Commands:

LOAD filename.ext	(load file)
SAVE filename.ext	(save file)
SAVE XX filename.ext	(save XX number of lines)
MERGE filename.ext	(merge two files)
PRINT NN	(print enter file, NN lines per page)
PRINTD NN	(print double spaced)
PRINT NN XX	(print XX lines, NN lines per page)
FIND anyword	(find the first occurence of a word)
MARGIN nn xx	(set left margin, nn, right margin, xx)
BYE	(exit to CP/M)

Key Commands:

Up arrow - enter EDIT mode at first line of text
Down arrow - enter EDIT mode at last line of text
HOME - enter EDIT mode at pointer (last cursor location)

EDIT MODE

Key Commands:

Up arrow - move cursor up one line
Down arrow - move cursor down one line
Right arrow - move cursor to the right one character
Left arrow - move cursor to the left one character
HOME - return to COMMAND mode
IL - insert line
DL - delete line
IC - insert character
DC - delete character
f1 - align paragraph within left and right margins
f2 - right justify paragraph text
f3 - margin on
f4 - margin off
f5 - MBASIC line split (insert MBASIC logical line extension)
BLUE - find next occurence of word, after FIND of COMMAND mode
RED - page backward
GRAY - page forward
LINEFEED - split line
ERASE - block erase

These are most of the basic commands of FAST EDDY. Please note that it has the ability to align paragraphs to new margin settings and then the option of right justifying the paragraph text.

Details of how to use these options are contained in the documentation. The TUTOR1, TUTOR2, and TUTOR3 documentation files are included with the disk to give the user experience in using FAST EDDY while reading the doc files.

BIG EDDY - This program is a utility to work with text files which are too large to be edited by FAST EDDY directly because of memory limitations. BIG EDDY can be used to browse a file of any size of which the user can break the large file into smaller parts for editing with FAST EDDY.

BIG EDDY asks for the input filename and an output filename. It keeps track of the subfiles and names them accordingly.

BIG EDDY has some useful options to aid the user in preparing the text for smaller files. The BROWSE mode is similiar to the EDIT mode of FAST EDDY, except that no editing can be done to the file.

The following are a list of the commands of BIG EDDY:

SAVEALL - save the entire text in memory to the disk
SAVEPART - save part of the text in memory to the disk
NOSAVE - discard part of text
PRINT - same as FAST EDDY's print commands
BYE - exit to CP/M

With the SAVEPART command, the user can save the text by subject or modules of his choice. Using the CP/M PIP program, the subfiles can be assembled into any order.

Comments: FAST EDDY, with the align paragraph and right justify, allow simple formatting, e.g. doing letters or other papers that do not require specific formatting capabilities.

885-8019[-37] CP/M DOCUMAT Formatter and DOCULIST $20.00

Introduction: DOCUMAT is a word processor which formats text to specific parameters. The numerous options make this formatter one of the most powerful of its kind. DOCULIST will preview and print files that have been processed with DOCUMAT.

Requirements: These programs require the CP/M operating system version 2.2 or later on an H8/H/Z-19 or H/Z-89 or Z90 with a minimum of 32K of memory. A printer is not required to execute the programs, however, hardcopy printouts of the text will require having a printer. Only one disk is required, however, DOCUMAT will access any size file including multiple disk files.

This product will run on the H/Z-100s under CP/M-85.

The following programs are included on the HUG P/N 885-8019[-37] CP/M DOCUMAT Formatter and DOCULIST disk:

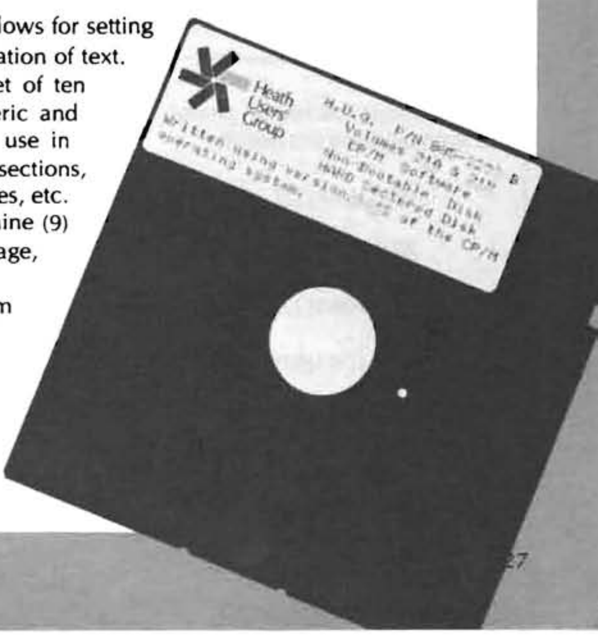
DOCUMAT .COM
DOCULIST .COM
MACLIB .DOC
TESTFILE

Author: Neal A. and Susan Van Eck

DOCUMAT - This program is a wordprocessing system which will produce neatly formatted documents. DOCUMAT takes a file of text (which is created with an editor), including DOCUMAT commands, and formats it for final printout according to your specifications.

The program allows for a number of options including great flexibility of page layout and text control. The commands allow for margin control, paragraph indentation, soft-hyphenation, page size, use of tabs, underline, and bold facing. Sentences are automatically capitalized and separated by two spaces. New pages are started automatically or whenever you wish, and can be set for front and back paging, while maintaining a running header for each page.

The program allows for setting for right justification of text. It provides a set of ten counters (numeric and alphabetic) for use in number pages, sections, chapters, prefixes, etc. It allows up to nine (9) footnotes per page, automatically numbering them as they ap-



pear. It automatically generates a Table of Contents of the text. It has the facility to INCLUDE (or merge) a separate file anywhere in the main text. It provides for interactive sessions while processing. The most useful tool is the powerful macro facility.

DOCUMAT has Global Commands, which apply to the entire text, and Control Commands, which control the text immediately following the command. The following is a list of some of these commands available for the DOCUMAT formatter:

GLOBAL Commands:

CTRk - set numeric counter k
CTRn - set alphabetic counter n
CYCLE - cycle page numbers for facing pages
HEAD - print the running head
JUSTIFY - set for right justification
LINES - set maximum number of lines per page
LMARGIN - set left margin
NUMBER - print page numbers
PAGE - format paging parameters
PARAGRAPH - indent paragraphs
PSPACING - line spacing between paragraphs
SHIFT - indent all text for right & left pages
SPACING - spaces between lines
TMARGIN - set top margin for first line per page
WIDTH - set the number of characters per line

CONTROL Commands:

A - all lines of text following will be copied without change
AR - align right (move the text to the right margin)
B - begin boldface text
C - center current line
Dn - set dot leaders (for ..... example)
Hn - indent each successive line after current line
In - indent n spaces each line including current line
Jn - jump n lines
Ln - start a new line leaving n-1 blank lines
N - start a new page immediately
P - start a new paragraph
R - reset, start new line and turn off temporary options
S - justify current line and start new line
Tn - set tab, space to column n
U - begin underlining
Wn - skip to new page

The ability to define and use macros is the most powerful facility in DOCUMAT. This feature is not found in most other wordprocessors. A macro is a predefined set of text or a procedure, which can be used in the text file wherever and whenever called.

With macros, you can automatically generate numbered chapter and section headings, ensure consistency, and include predefined text and control commands - even macros of macros. You can create printer control macros to send control codes for your printer. DOCUMAT also provides for "structured" writing. (MACLIB.DOC is a file of predefined macros of general use.)

Details of all commands are contained in the instructions. The disk contains a file and the manual contains a tutorial to give the user hands-on experience using many of the commands.

DOCULIST - This program allows you to preview or type the formatted output text of DOCUMAT. The PREVIEW mode will pause after every 24 lines until you are ready to go on. You also can shift the display for lines that are longer than 80 columns.

DOCULIST can be used to display or print any text file. The PREVIEW and TYPE options can be started at any desired page number.

Comments: DOCUMAT will require some study to master its many commands. Once you have learned the commands, you will know how to use one of the most powerful formatters.

The following five HDOS products are availale in soft-sectored format beginning this month:

885-1044[37]	Utilities Disk VI
885-1079[37]	Page Editor PAGED
885-1083[37]	Disk XVI Misc. Utilities
885-1093[37]	Dungeons and Dragons
885-1112[37]	Graphic Games Disk

HUG Price List

The following HUG Price List contains a list of all products not included in the HUG Software Catalog. For a detailed abstract of these products refer to the issue of REMark specified.

Part Number	Description of Product	Selling Price	REMark Issue
HDOS			
885-1029 [-37]	Disk II Games 1 H8/H89	$ 18.00	40
885-1038 [-37]	Wise on Disk H8/H89	$ 18.00	42
885-1042 [-37]	PILOT on Disk H8/H89	$ 19.00	42
885-1060 [-37]	Disk VII H8/H89	$ 18.00	40
885-1062 [-37]	Disk VIII H8/H89 (2 Disks)	$ 25.00	40
885-1064 [-37]	Disk IX H8/H89	$ 18.00	42
885-1067 [-37]	Disk XI H8/H89 Games	$ 18.00	40
885-1071 [-37]	MBASIC SmBusPk H8/H19/H89	$ 75.00	41
885-1078 [-37]	HDOS Z80 Assembler	$ 25.00	42
885-1086 [-37]	Tiny HDOS Pascal H8/H89	$ 20.00	40
885-1089 [-37]	Disk XVIII Misc H8/H89	$ 20.00	41
885-1090 [-37]	Disk XIX Utilities H8/H89	$ 20.00	41
885-1097 [-37]	MBASIC Quiz Disk H8/H89	$ 20.00	41
885-1107 [-37]	HDOS Data Base System H8/H89 ..	$ 30.00	42
885-1108 [-37]	HDOS MBASIC Data Base System ..	$ 30.00	41
885-1121	Hard Sectored Support Package	$ 30.00	37
885-1122	MicroNET Connection	$ 16.00	37
885-1123	XMET Robot/Cross Assembler	$ 20.00	40
885-1124	HUGMAN & Movie Animation Pkg ..	$ 20.00	41
885-1125	MAZEMADNESS	$ 20.00	41
885-1126	HDOS UTILITIES by PS:	$ 20.00	42
885-8015	TEXTSET Formatter	$ 30.00	42
885-8016	Morse Code Transceiver Ver 2.0	$ 20.00	41
885-8017	HDOS Programmers Helper	$ 16.00	42
CP/M			
885-1211 [-37]	Sea Battle	$ 20.00	36
885-1222 [-37]	Adventure	$ 10.00	36
885-1223 [-37]	HRUN HDOS Emulator	$ 40.00	37
885-1224 [-37]	MicroNET Connection	$ 16.00	37
885-1225 [-37]	Disk Dump and Edit Utility (DDEU) ..	$ 30.00	38
885-1226 [-37]	CP/M Utilities by PS:	$ 20.00	38
885-1227 [-37]	CP/M Cassino Graphic Games	$ 20.00	38
885-1228 [-37]	CP/M Fast Action Games	$ 20.00	39
885-1229	XMET Robot/Cross Assembler	$ 20.00	40
885-1229 [-37]	XMET Robot/Cross Assembler	$ 20.00	40
885-1230	CP/M Function Key Mapper	$ 20.00	42
885-1230 [-37]	CP/M Function Key Mapper	$ 20.00	42
885-3003 [-37]	ZTERM Modem Package	$ 20.00	36
885-8012 [-37]	Modem Appl. Effector (MAPLE)	$ 35.00	36

ZDOS

885-3004-37	ZBASIC Graphic Games Disk	$ 20.00	37
885-3005-37	ZDOS ETCHDUMP	$ 20.00	39

MISCELLANEOUS

885-0004	HUG 3-Ring Binder	$ 5.75	
885-4001	REMark VOLUME 1, Issues 1-13 ...	$ 20.00	
885-4002	REMark VOLUME 2, Issues 14-23 ..	$ 20.00	
885-4003	REMark VOLUME 3, Issues 24-35 ..	$ 20.00	
885-4600	Watzman/HUG ROM	$ 45.00	41

NOTE: The [-37] means the product is available in hard-sectored or soft-sectored. Remember, when ordering the soft-sectored format, you must include the "-37" after the part number; e.g. 885-1223-37.

If you are reading a borrowed copy of REMark...

maybe now is the time to join the National Heath Users' Group.

● *You will receive a copy of **REMark** filled with new and exciting articles and programs each month.*

● *You will also receive access to the HUG library filled with a large variety of programs.*

And remember, your local HUG is an excellent source of information, support, and comradery. A membership package from the National Heath Users' Group contains a list of current local HUG clubs as well as other interesting information

PILOT/80 Revisited

Kurt Albrecht
1061 Boot Rd.
Downingtown, PA 19335

I would like to thank those readers who sent me letters detailing their experiences with PILOT/80. With their help, I have been able to correct several bugs and make a couple of enhancements.

```
190 IF C$=CHR$(13) THEN PRINT:D$=A$:GOTO 270
```

This ensures the proper operation of the Match command.

```
330 IF B$="" OR C$=":" THEN RETURN ELSE IF C$="*" THEN@
    A$=B$: RETURN
```

This prevents a line preceded by colon to be converted to uppercase.

```
420 IF LEFT$(B$,1)="-" THEN BL%=1 ELSE BL%=VAL(B$):LN%=BL%:@
    GOSUB 450:IF EF%=1 THEN RETURN
```

This allows you to LIST and DELETE from line one to the desired line number. The format is; LIST -[Line number] or DELETE - [Line number]

```
1080 V$=MID$(P$,I%+1,10):FOR J%=1 TO LEN(V$):C1$=@
     MID$(V$,J%,1)

1090 IF C1$=>"a" AND C1$<="z" THEN MID$(V$,J%,1)=@
     CHR$(ASC(C1$)-32)

1100 NEXT J%:FOR K%=1 TO VP%:IF INSTR(V$,V$(K%,1))=1 THEN@
     PRINT V$(K%,2); :I%=I%+LEN(V$(K%,1)):GOTO 1070 ELSE@
     NEXT K%:GOTO 1060
```

This allows variables to consist of a dollar sign followed by one to ten characters. Therefore, $NAME1 is now a legal variable.

```
1140 IF C%=2 THEN GOSUB 170 ELSE A$=INPUT$(1):D$=A$:IF@
     D$=>"a" AND D$<="z" THEN D$=CHR$(ASC(C$)-32)
```

This just corrects an error in the operation of the Input command.

```
1150 V$(0,2)=D$:IF LEFT$(P$,1)<>"$" OR LEN(P$)<2 THEN 1020@
     ELSE P$=MID$(P$,2,10):IF VP%=0 THEN 1180
```

This ensures the proper operation of the Accept input command.

```
1300 A$="*"+LEFT$(P$,10):FOR I%=1 TO LL%:IF@
     LEFT$(P$(I%),11)=A$ THEN SP%=SP%+1: S%(SP%)=LN%:LN%=@
     I%:FOR J%=1 TO 1:NEXT:GOTO 1020 ELSE NEXT: PRINT ER$@
     "Line"LN%"jumps to a label that does not exist.":@
     GOTO 100
```

This just corrects an error in the User subroutine command.

You may have noticed that the documentation that accompanied PILOT/80 doesn't quite match the program. The reason is that REMark published the documentation for Version One of PILOT/80. I suppose for space and deadline reasons, they were unable to publish the documentation for Version Two, the program that was listed. The specifics concerning the program commands were also not published. This documentation is quite lengthy.

Due to several requests, I am now providing an improved version of PILOT/80 on a single sided, single density, 5.25" hard-sectored diskette for ten dollars. This contains the complete documentation (almost 90 sectors worth!) and several PILOT demo programs. At the present time, this is only available for HDOS and requires a minimum of 48K of memory.

Part I
An Introduction To 'C'

Brian Polk
86-02 Little Neck Parkway
Floral Park, NY 11001

This is the first of a series of articles intended to introduce the 'C' programming language from a unique perspective: I will be writing the articles as I am learning the language myself. My purpose is to help anyone who has an interest in the 'C' language by relating my own experiences. As with many other of the introductory articles published in REMark, this will be a guideline to help you get started. It can't teach you everything you need to know about the language, but it can whet your appetite so that you may be motivated to pursue the language on your own. I know you will find the articles informative and educational. Feel free to write to me with comments or suggestions.

What Is 'C'?

'C' is a language originally written in PDP-11 assembler by Dennis Ritchie of Bell Laboratories, and used for the development of the UNIX operating system. Although it is usually associated with the UNIX operating system, several versions have now been produced to run on many different micro-computers. The version I will be using is C/80 Version 1.6. This is a subset of the PDP- 11 'C' language but still contains most of the original features. I will be running under HDOS, but versions are available for CP/M also. 'C' is a two-step compiler language. The 'C' compiler produces a text file of assembler instructions which are passed through an assembler to produce an executable file. The disadvantage of having to go through a two-step procedure is outweighed by the fact that the executable file runs as fast as a program written in assembler. You will see an example of this two-step procedure later in this article. 'C' also allows you to intermix 'C' and assembler statements to get the best of both worlds. This makes it an appealing language for everyone, including those who like to program in assembler.

References

Several reference books are available on the 'C' language. Two of the better books are 'The C Programming Language' by Brian Kernighan and Dennis Ritchie (Prentice-Hall, 1978) and 'Learning To Program In C' by Thomas Plum (Plum Hall, 1983). Also check out REMark Issue 34 for an excellent article by William Bently titled 'The Next Step is C'.

Let's Get Started

The best way to assure success in a new language is to "get started, and don't quit!". Let's begin with an example which you know will work. This has two advantages. First, you get to find out if everything has been set up properly on your system regarding the compiler, assembler, operating system, and disk drives. Second, by following a pattern that works, you can begin with a positive experience to grow upon instead of being frustrated by failure, not knowing if you did something wrong or not.

The program we will run will print a sign-on message on the terminal. First, you will need to create a disk containing the operating system and the following files:

ASM.ABS - your normal Heath-supplied assembler
CC.ABS - the C compiler
CLIBRARY.ASM - the C I/O routines

Notice that I named the compiler CC.ASM instead of C.ASM. This is so that all users of SYSMOD, which abbreviates the HDOS 'CAT' command as the letter 'C', can still execute the compiler. You will probably also want to create a separate disk to hold your programs. I will be using SY2: in my examples.

Next, using your favorite text editor, create the file 'HELLO.C' containing the following statements:

```
main()
      {
      int i;
      i = fopen("tt:","w");
      write(i, "hello, you did great!\n", 22);
      }
```

If you are at all familiar with 'C', you probably know better ways of writing this program. However, for this illustration, this method is adequate. 'C' is free-format. Therefore, it is not necessary to enter the program exactly as I did. And as a matter of fact, you can enter the entire program on one line! For the sake of readability, however, you might as well get into the habit of structuring the layout of your code.

The '{' and '}' symbols you see used delimit every program written in 'C'. We will also use them elsewhere later.

Every 'C' program must have all variables defined according to how they will be used. For C/80, only character (char) and integer (int) variables are allowed. Later on we will get into "pointers", but for now everything we declare will either be an alphanumeric character or a signed numeric integer. In this example, we declare an integer variable 'i'.

Since we want to write a message to the terminal, we need to tell the program to make the console available for output. We do this with the 'fopen' function which takes two arguments: the file name ("tt:") and the operation ("w"=write). The function returns a channel number which we are assigning to the variable 'i'.

Next, we will print our message with the 'write' statement. We must specify a channel number ('i'), the message to be printed, and the number of characters to be printed (22). The '\n' is a 'C' representation for new-line and counts as only one character.

As you enter the program, keep in mind that 'C' programs are normally entered in lower-case, with upper-case only used in special situations. (If you only have an upper-case terminal, consult your C/80 operations manual for special instructions.)

Once you have the program entered, invoke the 'C' compiler by entering:

```
CC SY2:HELLO  -  the default extention is '.C'
```

You will see:

```
C/80 Compiler 1.6 (10/16/81)  -  depending on your version

0 errors in compilation  -  or any error messages
16K bytes free  -  depends on system configuration
```

If you check your disk's directory, you should now find a file named 'SY2:HELLO.ASM' which contains the assembler code which the 'C' compiler generated. Now we need to assemble this code in order to create an executable file.

```
ASM SY2:HELLO=SY2:HELLO
```

Now, when you enter:

```
SY2:HELLO
```

You will see:

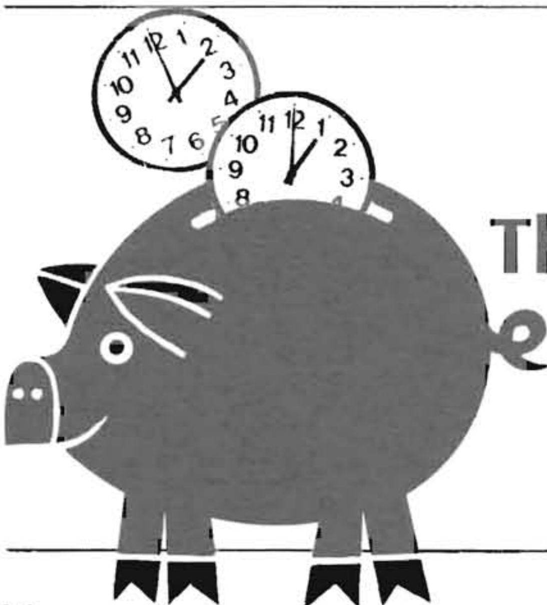
```
hello, you did great!
```

If you were not successful in getting to this point, try going back over the instructions again. If still unsuccessful, check your 'C' operations manual supplied with the compiler. There is usually a program similar to the one here with instructions on how to run it.

Once you have been successful in reaching this point, you might want to try experimenting with the compiler. For example, remove one of the semi-colons (';') just to get a feel for the type of compiler errors which can be generated. Also, try changing 'write' to 'qrite'. In this case, the 'C' compiler will NOT generate any error messages, but the assembler will.

Next time we will talk about program structure and introduce different ways to input and output information.

'C' you next time.



The Time Saver

George W. Stephenson, Jr.
1253 Park Place
Quincy, IL 62301

Michael J. Karas has defined a zero-size utility program which he calls @.COM. As he described it in the October, 1981 issue of LIFELINES, it is intended to reduce the repeated reloading of a transient program in a SUBMIT file. By this time, Mr. Karas (and others who have used it) must know that the suggested use is only one of several things that can be done with the program, all of which save time, and in at least one instance, a great deal of frustration. Quite a bit more needs to be said about Mr. Karas' brainchild.

To review, the program file is created in the CP/M command mode by typing the line 'SAVE 0 @.COM' and a C/R. When called by typing the filename (@), the CCP searches the directory, finds the program, loads it into transient program memory, and transfers control to it. But, since there is no program to load, control is actually given to the last **transient** program that was used. I stress the word transient to emphasize that the use of built-in commands (DIR, ERA, REN, SAVE, or TYPE) doesn't affect what was last read into the transient program area. Nor does CTRL C!

The original application was for the repeated use of PIP. After PIP was once loaded, the use of @ runs PIP again without requiring that PIP be read from the disk into memory again.

Now that you know what it is and what it was created for, here comes the better news!

It can be used with STAT. That can save a lot of time and disk wear when you are changing the attributes of a number of files on a disk. Just substitute @ for each use of STAT after the first.

Now, this won't work with all transient programs. If the program uses overlays, it may not be possible to re-use it with @. Trial and error will tell which ones will allow it.

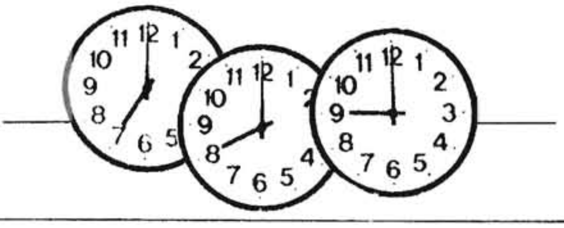
By this time, you should be ready for the best news yet. Suppose you have written an MBASIC program which has a SYSTEM command at the end of it. You've made some changes in it and you run the program without remembering to save it first. If you don't catch it before it exits, you'll have to reload MBASIC, and your program, and make those changes again. Well, maybe not. Let's try @. Whew! You now have BASIC and your program back in memory with two keystrokes!! Saved by a non-existent program.

What's that? You don't have @.COM on this disk? No matter. Create it. The SAVE utility doesn't change things any.

How about one more application? If you have files for use under different user numbers, you know what a hassle it is to copy the files into the user libraries other than 0. The suggested method is to put PIP into memory with DDT and then save it under the required user number. Of course, PIP may now be on the disk more than once. The shortcut is to call PIP, wait for the prompt, and exit with a C/R. Now, change user numbers, create @.COM, and use it to run PIP as if it were in this user number.

The only drawback I've found so far is that my natural laziness balks at having to use the shift key to type '@' and I've not yet decided on a better name. Maybe I'll try the apostrophe. Anyway, there's no hurry.

Surely there are other uses for this gem. Please let us know about them.



THE INTEGRATION OF WORD PROCESSING AND GRAPHICS.

THE **IMAGE WORD/GRAPHICS PROCESSOR** FOR

TeleVideo® AND ZENITH | data® systems

Advanced Word Processing

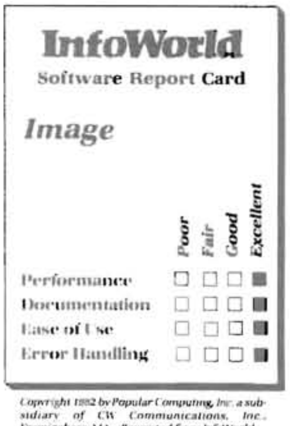
Now you can enjoy a great screen-oriented word processor that provides powerful, new editing features. Besides headers, footers, page numbering, superscripts, subscripts, boldface, underline, and other standard W/P benefits, IMAGE gives you flush and indented word wrap, on-screen left, right, center, and full justification, variable cursor direction, and the simplest, most flexible column move in the industry.

Move and Mix Words and Graphics

Draw lines and blocks as easily as you did with your "Etch-A-Sketch." Then move what you've created or mix it with text anywhere on your terminal screen. Create bar charts, forms, organizational charts, flow charts, block letters, calendars, signs, invoices, purchase orders, diagrams, drawings, and more.

Point-and-Press Commands

Finally, say "goodbye!" to memorizing control codes, dot commands, and embedded printer gobbledygook. IMAGE displays commands for you directly above the top row of function keys. Point to the command you want, then press the key below.



InfoWorld
Software Report Card

Image

	Poor	*Fair*	*Good*	*Excellent*
Performance	□	□	□	■
Documentation	□	□	□	■
Ease of Use	□	□	□	■
Error Handling	□	□	□	■

Copyright 1982 by Popular Computing, Inc. a subsidiary of CW Communications, Inc., Framingham, MA—Reprinted from InfoWorld.

Immaculate printing quality

Previously obtainable from only the most expensive dedicated W/P systems, IMAGE provides true letter proportional printing for an impeccable, typeset look. No other system at any price will mix true proportional printing with graphics on a standard, letter quality printer.

Top marks from InfoWorld

"IMAGE certainly deserves accolades in the performance category... the documentation is simply superb. It is professionally done from cover to cover... without a doubt, this program is easy to use... the program is bombproofed so well that I had trouble finding any errors."

Extraordinary Value

Again from InfoWorld: "the modest $295 price tag is a bargain."

Specifications

Computers:
TeleVideo 802, 802H
TeleVideo 950 connected to Z-80 based CP/M system
Heath/Zenith Z89, Z90
Heath/Zenith Z19 connected to Z-80 based CP/M system

Printers:
Diablo, NEC, CItoh, EPSON, Zenith Z-25 and others.
(No terminal or printer retrofitting required.)

To Order Contact your local TeleVideo or Zenith dealer or call toll-free 1-800-MICROART (1-800-642-7627); in Oregon 1-692-3950.

MICRO ART

MicroArt Corporation
200 Market Bldg., Suite 961
Portland, OR 97201

MasterCard, VISA, COD, and qualifying purchase orders accepted.

All quotes are from InfoWorld's *IMAGE* software review, by Marty Petersen, June 14, 1982.

Easy Cursor Control For The Novice ... And Others

Wallace M. Theodore
P. O. Box 2488
Hammond, IN 46323-0488

If someone had told me two and a half years ago, when I bought my H-89, how simple Direct Cursor Addressing REALLY is, I would not be rewriting programs today to make them look the way I always wanted them to but was afraid to try.

Like many others, I've looked through the operator's manual and bypassed such "mysterious" enhancements as cursor control, use of the twenty-fifth line, and Defined Functions. These were things I would "learn tomorrow" since, for the most part, they appeared both confusing and ominous.

Fortunately, along with the many hours of trial-and-error programming and digging through manuals, I've had the help of Michael Thornton, a programming student at Purdue University Calumet Campus (Hammond, Indiana) and programmer at U.S. Steel's Gary Works. He taught me the simple, direct method cursor control I use today - one which even the most unaccomplished novice should be able to use with no difficulty.

One of the first things I DID learn though, is that the H/Z-89 is capable of solving many of its own programming problems. With a little thinking, programs may usually be devised to instruct the computer to do the work. A program which requires the operator to search, find, input or otherwise continually assist the computer is, in my opinion, defeating its own purpose.

It was with this in mind that I read, several times, the article appearing on page 19, REMark #39, dealing with cursor control and several other functions. While this is surely an acceptable method of screen control, it requires the USER to look through manuals, convert codes to characters, remember to replace the quote sign with a decimal code, and then tell the computer where to put the cursor. This, to me, is a slow, confusing, cumbersome method.

I've found much more efficiency and practicality through use of the Defined Function (DEF FN), described in the MBASIC manual as "... a statement used to define an implicit function".

Since both the H/Z-89 operating manual and the article describe in detail how Direct Cursor Addressing works, I will not again describe it here.

Pat Swayne, also in REMark #39, defines BASIC as a "high level" language "... because the programmer does not have to be concerned with what actually goes on in the computer". When using DEF FN, it is also unnecessary to know how or why this method of cursor control works - only that it does.

Prior to writing the actual program, ESCape and string sequences must be established to call for reverse video, graphics, etc. I have found that "standardizing" these variables in all programs helps eliminate confusion and increase writing speed. In my programs, I use ES$ (ES for ESC) for the ESCape sequence:

```
ES$=CHR$(27)
```

The reason for the two letter designation will become apparent as we progress.

A Defined Function (DEF FN) is used to call for line and column positions and add 31 to each. It is not necessary to know why the 31 is there (your manual will tell you if you really want to delve into it).

In this definition, which is used as a standard listing in all my programs using cursor control:

CC$ - designates (C)ursor (C)ontrol. CC is my personal preference, but any single or double letter substitute which suits you may be used,

Y - is used as the line number,

Y1 - designates the column number.

To enter Direct Cursor Addressing, ESC Y must be entered. This is done using ES$ (as defined above) plus the letter Y, which must be entered in quotes:

```
ES$+"Y"
```

Next, 31 is added to the line number, designated by Y, and the column number, designated by Y1.

```
CHR$(31+Y)
CHR$(31+Y1)
```

DEF FN precedes the entry to let the computer know it is a Defined Function.

It is NOT necessary to know what each of the above listings really does. Descriptions are included only for clarification, if needed, and for those who desire additional information on how this method of cursor control works.

What IS important are the following lines, which include all the information necessary to tell your computer that YOU will be controlling cursor placement directly. All the instructions needed to begin using direct cursor addressing appear in the following boxed area.

```
10 ES$=CHR$(27)
20 DEF FNCC$(Y,Y1)
      =ES$+"Y"+CHR$(31+Y)+CHR$(31+Y1)
```

Sample line 20 need only be entered once, at the beginning of the program. Once the ESC code is designated as ES$ and the above line is entered, cursor control is simple and direct. To place the cursor on line 12, column 40, for example, type:

```
30 PRINT FNCC$(12,40)
```

That's all there is to it!

To then move the cursor to line 5, column 8, type:

```
40 PRINT FNCC$(5,8)
```

Of course, multiple instructions may be placed on one line through use of the semicolon(;).

```
50 PRINT FNCC$(12,40);FNCC$(5,8)
```

This example will place the cursor on line 12, column 40 and immediately move it to line 5, column 8. Some basic functional cursor control examples will be presented later.

Many programs use the designations E$, E1$, E2$, etc. to convert ESC sequences to string variables. I have found when a large number of these designations are used (i.e. E10$, E11$ or more), they become confusing and require a continual review of each variable to remember its purpose.

As with the assignment of CC$ to Direct Cursor Addressing, I also assign more descriptive designations to other variables. For example, when "E$" is equal to CHR$(27), entering and leaving reverse video and graphics would appear as:

E$=CHR$(27)	(ESCape sequence)
E1$=E$+"p"	(Enter reverse video)
E2$=E$+"q"	(Exit reverse video)
E3$=E$+"F"	(Enter graphics)
E4$=E$+"G"	(Exit graphics)

In the following examples, the first two letters indicate the function shown at the right.

RV$=E$+"p"	(R)everse (V)ideo
NV$=E$+"q"	(N)ormal (V)ideo
EG$=E$+"F"	(E)nter (G)raphics
XG$=E$+"G"	e(X)it (G)raphics
KC$=E$+"x5"	(K)ill (C)ursor
DC$=E$+"y5"	(D)isplay (C)ursor
E25$=E$+"x1"	(E)nter line (25)
X25$=E$+"y1"	e(X)it line (25)
CS$=E$+"b"	(C)lear (S)creen

CS$ (Clear Screen) is used in lieu of ES$ (Erase Screen) since ES$ is used for the ESCape sequence. Similar designations may be used for any function you desire. Of course, you can use any one or two letter substitute with which you feel comfortable. The listings above are most convenient for me.

While this method MAY use slightly more memory in some cases, it is much more efficient than the use of E$, E1$, E2$, etc., since it immediately shows you what the function is without further checking.

One more useful tool of BASIC-80 should be pointed out at this time since lines are necessary for many graphics, especially borders: STRING$, which is used to build a string.

For those who are just beginning programming, STRING$ will save both programming time and computer memory. You might, for example, want to print a string of 48 a's. To do this, you would write:

```
10 PRINT "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa"
```

When this instruction, or a longer string, is written several times into a program, memory is used needlessly. One solution during repeated use might be by means of a GOSUB, but the entire string must still be typed out. What's more, the GOSUB itself uses additional memory.

STRING$ offers a more practical solution which allows you to simply type:

```
10 PRINT STRING$(48,"a")
```

The 48 in the above example indicates the number of repeat printings of the character "a". This listing will return an exact duplicate of the previous listing.

The following "no frills" demonstration program presents some idea of how cursor control works. It also contains several reference sequences which may be very helpful when you first incorporate Direct Cursor Addressing in your own programs. It is by no means complete but rather, shows Cursor Control in its most basic form. Included is enough information to draw lines up or down on the screen, right to left or vice versa, and create a square which can be adapted to your own programs as, possibly, a border. Also included is limited use of graphics, reverse video, and the twenty-fifth line.

The program is written in sections with a description of each section given as the listing progresses. Each section is followed by a STOP which will halt execution of the program at that point. After inspecting what the program does, type CONT to display the next example.

To begin, we must first enter the ESC code, Defined Function, and control codes for reverse video, etc., and erase the screen.

```
10 ES$=CHR$(27)
20 DEF FNCC$(Y,Y1)=ES$+"Y"+CHR$(31+Y)+CHR$(31+Y1)
30 EG$=ES$+"F":XG$=ES$+"G":RV$=ES$+"p":NV$=ES$+"q"
40 E25$=ES$+"x1":X25$=ES$+"y1":CS$=ES$+"b":KC$=ES$+"x5"
50 DC$=ES$+"y5":PRINT CS$
60 PRINT FNCC$(10,35);STRING$(12,"a")
70 STOP
```

Line 60 will place the cursor at line 10, column 35, and print a string of 12 a's. Line 70 will STOP program execution until CONT is entered.

```
80 PRINT EG$;FNCC$(11,35);STRING$(12,"a");XG$
90 STOP
```

Line 80 will print a solid line (the graphics equivalent of "a") immediately below the line of a's. The instruction is identical to that of line 60 except EG$ has been used to enter the graphics mode and XG$ to exit. Line 90 stops execution.

```
100 PRINT RV$;EG$;FNCC$(13,35);STRING$(12,"a");XG$;NV$
110 STOP
120 PRINT CS$
```

Line 100 is identical to line 80 except the solid line is printed in reverse video. After CONT is typed, line 120 will clear the screen.

At line 110, all three of the above configurations will be shown on the screen.

FOR/NEXT loops allow the printing of characters or graphics down OR UP the screen since a part of the Defined Function takes on the value of the loop, in this case "X".

Line 130 will print the character * from line 6 to line 14 in column 40. Line 150 will repeat the process using graphics and placing the display in column 41 thus giving a side-by-side display of both the character and its graphic representation.

Since the cursor has a tendency to "flash" as it moves to a new location, it will be turned off by means of KC$ in line 130. This will help produce a clean, neat display.

```
130 PRINT KC$:FOR X=6 TO 14:PRINT FNCC$(X,40);"`":NEXT X
140 STOP
150 PRINT EG$:FOR X=6 TO 14:PRINT FNCC$(X,41);"`":NEXT X:
160 PRINT XG$:STOP
170 PRINT CS$
```

Using the FOR/NEXT loop and a minus step instruction, it is possible to print a character or graphic from the bottom of the screen to the top. Line 190 will show this, using the same character as in the previous demonstration.

```
180 PRINT EG$
190 FOR X=14 TO 6 STEP -1:PRINT:FNCC$(X,40);"`":NEXT X
200 PRINT XG$:STOP
```

When using Direct Cursor Addressing, always remember that it is necessary for the cursor to be AT THE BOTTOM OF OR BELOW the display to erase it from the screen. Line 210 will move the cursor to line 17, column 1, before the clear statement is issued.

```
210 PRINT FNCC$(17,1);CS$
```

For the benefit of the novice, the four graphic characters to be shown in lines 220 and 230 will become the "corners" of a square to be created in subsequent lines. Following STOP, the cursor is moved to line 24, column 1, before erasing the screen.

```
220 PRINT FNCC$(15,40);EG$;"f";" ";"c";
230 PRINT FNCC$(17,40);EG$;"e";" ";"d";XG$
240 STOP:PRINT FNCC$(24,1);CS$
```

Using the techniques outlined above, the following instructions will create a square using Direct Cursor Addressing. Note that graphics will be used and "corners" will be included in the instructions.

```
250 PRINT EG$;FNCC$(10,25);"f";STRING$(30,"a");"c"
260 FOR X=11 TO 20:PRINT FNCC$(X,56);"`":NEXT X
270 PRINT FNCC$(21,56);"d"
```

As in the case of printing a line from the bottom to the top of the screen, line 280 will print a continuous line from RIGHT TO LEFT using a minus step instruction.

```
280 FOR X=55 TO 26 STEP -1:PRINT FNCC$(21,X);"a";:NEXT X
290 PRINT FNCC$(21,25);"e"
```

Line 300 will move graphics UP the screen, from line 20 to 11, and 320 will place text inside the square. Note that XG$ in line 310 exits the graphics mode before text is used.

```
300 FOR X=20 TO 11 STEP -1:PRINT FNCC$(X,25);"`":NEXT X
310 PRINT XG$:STOP
320 PRINT FNCC$(15,36);"I DID IT!"
```

Line 330 will instruct the cursor to enter the twenty-fifth line, turn on reverse video, place the cursor at column 32, and print the statement.

```
330 PRINT E25$;RV$;FNCC$(25,32);" I DID IT AGAIN! "
```

The FOR/NEXT loop in line 340 creates a pause governed by the value of the "TO" portion (in this case, 1500 - the higher the number, the longer the pause). The cursor will then exit the twenty-fifth line to line 22, column 1 (this is done to keep the display from moving up a line if the cursor were returned to the 24th line), normal video is returned, and the cursor is again displayed before the program ends.

```
340 FOR X=1 TO 1500:NEXT X:PRINT X25$;NV$;FNCC$(22,1);DC$
350 END
```

You may want to try changing the values of various cursor instructions to get the feel of using the system. See, for instance, if you can move the words "I DID IT!" to each of the four corners of the square or other positions inside or outside the square. This is done in line 320 by changing the value of (15,36).

Remember that, in this system, the LINE number is always first, followed by the column number. Unless the twenty-fifth line is used, the highest line number is 24 and the highest column number is 80.

You'll find this method of Direct Cursor Addressing a fast, easy, efficient means of discovering new and exciting programming capabilities. The examples in the demonstration program will only get you started but with a little practice, you'll be well on your way to finding how valuable and useful this tool is: not only for better program appearance, but for more EFFICIENT programs as well.

If you have questions or comments, I'll be happy to assist you in any way I can. Please include a self-addressed stamped envelope to PO Box 2488, Hammond, Indiana, 46323-0488.

About the Author:

Wallace Theodore is a 49 year old president of Telesound Specialties, a broadcast advertising agency and also "mother company" of Bio-Specialties, the creator of the Bio-Guide/Bio-Analysis.

A self-taught programmer, Wallace has been using his H-89 (with H-37 and TI-810 printer) with CP/M and BASIC-80 for about two and a half years. Unlike many users', his computer is almost totally used for business functions, and for programming of the biorhythm report. Most of his programs are "home brew" since many commercial programs fall short of his specific needs.

Software for HEATH/ZENITH Systems

UTILITIES

UD.COM - A printer driver/spooler similar to UD.DVD for CP/M 2.2
SUPER SYSMOD2—An HDOS enhancement program that abbreviates HDOS commands and adds many new ones.
UD.DVD—A truly universal driver for printers. Provides seventeen options including multiple copies and page formatting.
H25.DVD—A special driver for the H/Z-25 printer.
MX.DVD—A special driver for the MX-80 family of printers.
SPOOLER DRIVERS—Above drivers with built-in spooling.
BASEDIT—Adds a built-in editor to HDOS BASIC
EDGE—A powerful character oriented text editor
DISAS—A fast dissembler (includes TINY PASCAL)

LANGUAGES

HFOFTH79—fig FORTH plus the FORTH-79 standards
HFORTH—fig FORTH 1.1 with many other features

EDUCATIONAL

SUPERMATH1—Provides educational math instruction for grades 1 to 6
SPANISH, FRENCH, GERMAN, and ITALIAN HANGMAN
Each version contains over 2000 words presented as flashcards or in the popular game of HANGMAN.

Call or write for FREE catalog.
Available at your local HEATH dealer or from:

35 Shadow Oak Dr.
Sudbury, MA 01776
(617) 443-9693

Soft Shop
Jim Teixeira

DISK DRIVES DISK DRIVES DISK DRIVES DISK DRIVES DISK DRIVES DISK DRIVES DISK DRIVES DISK DRIVES

PRICE BREAKTHROUGH

Super Sale on New Disk Drives

Starting at **$199.95 complete!!**
with Power Supply and Case.

Tandon — Siemens — Remex — MPI — Teac — Shugart
single sided 40 track — dual sided 40 track
single sided 80 track — dual sided 80 track
for
RADIO SHACK[1] — HEATH/ZENITH[2] — APPLE[3]/FRANKLIN[4]
IBM/PC-TEXAS INSTRUMENTS[6] & MOST OTHER COMPUTERS
Drive a Hard Bargain!!™

8 M.B. - 12M.B. with Power Supply Case, Cables & Software
Complete Systems . . . . from $1,295.00
Amazing Special!-dual sided 40TK or 80TK drive with Power Supply and Case
and Our Special Warranty - Starting at $Call

TOLL FREE ORDERING **GENERAL AND TECHNICAL**
1-800-343-8841 **1-617-872-9090**

Model I/III/IV **Drives** (0 1 2 3)	*starting at* **$175.00**
Color Computer Drives (0 1 2 3)	*starting at* **$Call**
Apple/Franklin Compatible Add-On Drives	$229.95
Apple/Franklin Compatible Drives with Controller	***$299.95***
Model I/III/IV Memory Upgrade	*starting at $Call*
Printers — Daisywheel/Dot Matrix	$Call
Diskettes	**Quantity** Priced from $17.00
Cases and Power Supplies — (Single-Dual-**1/2 Height**)	starting at $49.95
Printer **Buffers** 8K to 512K	starting at $143.95
Percom Double Density Controller (Model I)	$Call
Holmes Model I/III Speed-up Mod-VID/80	*starting at* $90.00
Gold Fingered **Edge Card Extenders**	*starting at* $13.00
Cables — Printer/Disk Drive	*starting at* $25.00
DOSPLUS	***$Special prices***
Repair Services Now Offered for Disk Drives— *Fast Turn-a-Round*	$Call

One Year Warranty Available on all Drives!

SOFTWARE SUPPORT, INC.

One Edgell Road, Framingham, MA 01701 (617) 872-9090
Hours: Mon. thru Fri 10 am to 6 pm (E.S.T.) Sat. 10 am to 5 pm

Dealer inquiries invited.

TERMS:
M.C./Visa/Amex and personal checks accepted at no extra charge.
C.O.D., please add $3.00.
Shipping: Please call for amount.

Canada
MICRO R.G.S. INC.
751, CARRE VICTORIA, SUITE 405
MONTREAL, QUEBEC, CANADA, H2Y 2J3
Regular **Tel. (514) 845-1534**
Canadian Toll Free 800-361-5155

1 ™TANDY CORPORATION
2 ™ZENITH DATA SYSTEMS
3 ™APPLE COMPUTER CORP.
4 ™FRANKLIN COMPUTER, INC.
5 ™IBM CORPORATION
6 ™TEXAS INSTRUMENTS
© Copyright 1983

Processing H/Z-19 Special Functions Using Heath ASseMbly Language

William R. Rousseau, M.D.
1388 Highland Ave.
Salem, OH 44460

The intent of this article is to show one method of implementing the special function keys of the H/Z-19 terminal. Many programs are available that make extensive use of the special function keys of the H/Z-19 terminal. HUG's UDUMP.ABS (Part number 885-8004) and Soft Stuff's Full Screen Editor (a.k.a. PIE.ABS, Heath part number SF-9200) are prime examples of such programs.

A few articles are available on the use of these special function keys; however, most of them have been written in specific reference to BASIC language. While BASIC can conveniently utilize these keys, it does so in a limited manner compared to the potential of assembly language that has an inherent ease and speed for dealing with individual keyboard inputs. Further, in assembly language, the special functions can be integrated with ASseMbly's other advantages of speed, optimum RAM utilization, minimum program disk space, and access to numerous system ROM subroutines. There is no intent here to advocate one language over another; perhaps after reading and grasping the basic ideas involved, the reader will develop some ideas that will help him to employ the special function keys in a number of different available languages.

The special function keys commonly lend themselves, but are not limited, to two programming situations. The first is menu subroutines. While the menu subroutine can be called by any alphanumeric key, special function keys add a distinctive, professional touch to programs. The second situation is considerably more complex. Assume that you are composing a program that must input or process or manipulate (or all three functions) potentially any or all of the alphanumeric characters of the keyboard. Programs that edit text are good examples. Further, assume that you desire your program to be capable of executing any one of a number of side functions or subroutines, such as recording the contents of the screen onto disk, converting all lower case letters of your text to upper case, printing all or part of your text in reverse video, or simply "tell-

```
        TITLE   'DEMO.ASM'

* Demo program for Heath H19/89 special function and
* alternate keypad keys by W. R. Rousseau, M.D.
*
*HDOS EQUATES AND DEFINITIONS
BELL    EQU     07Q              1 VALUE FOR 'BELL' CHARACTER
ESC     EQU     33Q              2 VALUE FOR 'ESCAPE' CHARACTER
TICCNT  EQU     40033A           3 ADDRESS CONTAINING TIC COUNTER VALUE
$TYPTX  EQU     31136A           4 HEATH ROM SUBROUTINE
NL      EQU     12Q              9 NEW LINE CHARACTER ('RETURN' KEY)
USERFWA EQU     42200A           10 STARTING ADDRESS OF PROGRAM
**********************************************************************
        ORG     USERFWA          11 PROGRAM STARTING ADDRESS
GO      LXI     H,EXIT           12 SET UP CONTROL-C HANDLING
        MVI     A,3              13       WITH
        SCALL   .CTLC            14 HEATH '.CTLC' SCALL
        LXI     H,SHIFT          15 SET UP CONTROL-A HANDLING
        MVI     A,1              16 (TO ENTER THE ALTERNATE KEYPAD
        SCALL   .CTLC            17 MODE) WITH HEATH SCALL
        LXI     H,UNSHFT         18 SET UP CONTROL-B HANDLING
        MVI     A,2              19 (TO EXIT THE ALTERNATE KEYPAD
        SCALL   .CTLC            20 MODE)
GO1     CALL    CMNE             21 WANT CHARACTER MODE, NO ECHO
        CALL    $TYPTX           22 TYPE PROMPT
        DB      NL,'Hit key: ',200Q
GO1.5   SCALL   .SCIN            24 GET USER-INPUT KEY
        JC      GO1.5            25 NOT THERE, GET AGAIN
        CPI     ESC              26 KEY STRUCK, IS IT AN 'ESCAPE' CHARACTER ?
        CZ      PCC              27 YES, NOW PROCESS IT!
        JMP     GO1.5            28 NO,TRY AGAIN
*******************************************************************
*'PCC' This subroutine waits until the next character is received
*       or until time (20mS) has run out.
PCC     EQU     *
        LDA     TICCNT           29 GET VALUE IN THE TIC COUNTER. . THEN
        ADI     10               30 ADD 10 FOR COMPARISON STANDARD .AND
        MOV     C,A              31 SAVE THAT VALUE IN 'C' REGISTER.
PC1     SCALL   .SCIN            32 HAS H19 SENT ANOTHER CHARACTER ?
        JNC     OUTCHR           33 IF SO, PROCESS IT, OTHERWISE. .
        LDA     TICCNT           34 GET CURRENT TIC VALUE
        CMP     C                35 COMPARE WITH OUR STANDARD . .
        RP                       36 TIME UP. . . RETURN
        JMP     PC1              37 STILL TIME. .CHECK INPUT AGAIN
*******************************************************************
*'OUTCHR' subroutine processes the second character in the sequence
OUTCHR  CPI     '?'              38 ALTERNATE KEYPAD CHARACTER?
        JZ      AKP              39 YES, PROCESS IT . . OR
        CPI     'Q'              40 NO, BUT IF H19 SENDS A 'Q' THEN THE
        JZ      REDOUT           41 RED KEY WAS STRUCK
        CPI     'P'              42 ..AND SO ON FOR THE
        JZ      BLUOUT           43 BLUE KEY
        CPI     'R'              44
        JZ      GROUT            45 GRAY KEY. . .
        CPI     'S'              46 WAS IT 'F1' KEY?
        JZ      F1               47 IF SO, GOTO 'F1' SUBROUTINE
        CPI     'T'              48 'F2'?
        JZ      F2               49      AND
```

ing" the program that you are now finished and wish to terminate activities, and - the hooker - be able to call any one of these subroutines during the working segment of the program.

Because the program is dealing with potentially all alphanumeric characters of the keyboard, a unique method must be devised to call a desired subroutine. This job can be handled in a number of ways:

(1) The program can be designed to respond to characters that rarely appear in any text, such as the character ' | ' (174Q), or '~' (176Q). The former is called a 'vertical bar (broken)', and the latter, a 'tilde'. Use of the "rarely occurring character" has the major drawback of being limited in quantity. Thus, if one has six, eight or more subroutines available, he will quickly run out these characters with which to call them.

(2) Another convenient method for calling subroutines during runtime is to use a control character (CTRL-C, D, etc.). The documentation for implementing these is available in the 'HDOS Programmers' Guide'. While this is an improvement over the use of the "rarely occurring character", one still has problems in dealing with more than just a few available subroutines.

(3) Use the special function keys!

The special function keys of the H/Z-19 terminal consist specifically of the keys 'f1' through 'f5' as well as the 'blue', 'red', and 'gray' keys. They differ from the alphanumeric keys of the keyboard in that they generate a sequence of ASCII characters, an escape sequence. In addition, if the H/Z-19 is set to the Alternate Keypad Mode, the 12 numeric keys of the small keypad generate escape sequences and can also be utilized in the same way as the special function keys. The use of these keys will also be discussed. Section 12 of the H19 Operation Manual should be consulted for additional information on the subject of Heath escape sequences. Unfortunately, this documentation leaves software implementation largely to the user's imagination.

The special function keys differ from alphanumeric keys of the H/Z-19 terminal in that they cause more than one ASCII character to be sent by the terminal. The first character in this sequence is always the ASCII ESCape character (octal 33). Next, in immediate succession, a second ASCII character then serves to identify the special function key struck. For example, if the 'f1' key were struck, the two character ASCII sequence, 'ESC' (033Q) and 'S' (123Q) would be sent. As mentioned, the terminal can be set to the Alternate Keypad Mode (see lines

```
        CPI     'U'                     50          SO
        JZ      F3                      51            ON
        CPI     'V'                     52           FOR THE
        JZ      F4                      53               OTHER
        CPI     'W'                     54                  SPECIAL FUNCTION
        JZ      F5                      55                       KEYS
        JMP     NOKE                    56 WAS 'ESC' BUT NOT ONE OF OURS
*
*PROCESSING SUBROUTINES, SPECIAL FUNCTION KEYS RED, BLUE, GRAY & F1-F5
*
REDOUT  CALL    $TYPTX
        DB      NL,NL,'Red Key struck!',NL,200Q
        JMP     GO1
BLUOUT  CALL    $TYPTX
        DB      NL,NL,'Blue Key struck!',NL,200Q
        JMP     GO1
GROUT   CALL    $TYPTX
        DB      NL,NL,'Gray Key struck!',NL,200Q
        JMP     GO1
F1      CALL    $TYPTX
        DB      NL,NL,'Function Key #1 struck',NL,200Q
        JMP     GO1
F2      CALL    $TYPTX
        DB      NL,NL,'Function Key #2 struck',NL,200Q
        JMP     GO1
F3      CALL    $TYPTX
        DB      NL,NL,'Function Key #3 struck',NL,200Q
        JMP     GO1
F4      CALL    $TYPTX
        DB      NL,NL,'Function Key #4 struck',NL,200Q
        JMP     GO1
F5      CALL    $TYPTX
        DB      NL,NL,'Function Key #5 struck',NL,200Q
        JMP     GO1
NOKE    CALL    $TYPTX
        DB      NL,NL,'No Special key struck',NL,200Q
        JMP     GO1
*
*'AKP' This subroutine processes the third byte for alternate keypad
*       mode activities.
*
AKP     SCALL   .SCIN               GET THIRD BYTE
        JC      AKP
        CPI     'r'                 DOWN ARROW KEY ?
        JZ      DOWN
        CPI     't'
        JZ      LEFT                          ANOTHER
        CPI     'u'                             THAT
        JZ      HOME                              WE
        CPI     'v'                                SET UP
        JZ      RIGHT                                FOR
        CPI     'x'                                   . ? . .
        JZ      UP                                    . .  .   .
        CALL    $TYPTX              NO, IT'S AN UNDEFINED ALTERNATE PAD KEY.
        DB      NL,'Shifted key pad, undefined key',NL,200Q
        JMP     GO1                 DO IT ALL OVER AGAIN !
***************************************************************************
*
DOWN    CALL    $TYPTX
        DB      NL,'Down Arrow key struck.',NL,200Q
        JMP     GO1
LEFT    CALL    $TYPTX
        DB      NL,'Left arrow key struck.',NL,200Q
        JMP     GO1
HOME    CALL    $TYPTX
        DB      NL,'Home key struck.',NL,200Q
        JMP     GO1
RIGHT   CALL    $TYPTX
        DB      NL,'Right arrow key struck.',NL,200Q
        JMP     GO1
UP      CALL    $TYPTX
        DB      NL,'Up arrow key struck.',NL,200Q
        JMP     GO1
*
```

☞

```
CMNE    XRA     A               'CMNE' SUBROUTINE SETS UP THE
        MVI     B,201Q          HDOS .CONSL SYSTEM CALL. .
        MVI     C,201Q          . .SEE YOUR HDOS MANUAL FOR
        SCALL   .CONSL          DETAILS.
        RET
SHIFT   CALL    $TYPTX          TERMINAL ENTERS 'ALTERNATE KEYPAD
        DB      ESC,'='         MODE'
        DB      NL,NL,NL,BELL,'Alternate Keypad Mode!',NL,NL,200Q
        JMP     GO
UNSHFT  CALL    $TYPTX          TERMINAL EXITS 'ALTERNATE KEYPAD
        DB      ESC,'>'         MODE'
        DB      NL,NL,NL,BELL,'Exit Alternate Keypad Mode!',NL,NL,200Q
        JMP     GO
EXIT    XRA     A               'EXIT' SUBROUTINE CAUSES PROGRAM TO
        SCALL   .EXIT           EXIT TO THE HDOS COMMAND MODE.

        END     GO
```

15 through 20 of the accompanying DEMO program). Then the keys of the small keypad cause the ASCII ESCape character plus TWO additional characters to be sent. The second character is ASCII '?' (077Q); the third character identifies the specific alternate keypad key that was struck.

The programmer's task is simply to set up a program that will, in the midst of processing the standard alphanumeric characters of the keyboard, detect the receipt of the ASCII 'ESC' (33Q) character. The receipt of that character causes a subroutine to be called that would identify the second character and, thus, the special function key struck. If one intends to use the Alternate Keypad Mode, his subroutine should also determine whether the second character is ASCII '?' (077Q). If that condition evaluates positive, a second subroutine would be called to identify the specific alternate keypad key that was struck.

One last consideration! The subroutine that identifies the specific special function key struck should provide for receipt of the second character within a very short period of time - short enough for the program to determine that the second character received was originated from the internal workings of the H/Z-19, and was not the result of a series of erroneous key strikes. To better understand this particular scenario, assume that instead of striking the 'TAB', 'Q', or '1' key, the 'ESC' key is accidentally struck. The program will call its 'identifier' subroutine and at that point, await the next character. If the next key in the sequence were 'R', the program would make the erroneous interpretation that the GRAY key had been struck. A far more likely probability, however, is that the second key would not be a valid second character of a special function sequence. This could be a source of program bugs. In assembly language, the incorporation in a '.SCIN' loop of a timing segment of sufficiently short duration (20 milliseconds, nominal) allows for the receipt of the second character of a special function sequence, while forcing a RETurn from the identifier subroutine before a user-generated character could occur, no matter how fast the keys are struck.

The accompanying example program demonstrates one method of implementing this information. First, a Control-C handler is set up to allow the user to exit when finished (lines 12-14). Control-A and Control-B handlers then establish the means to enter and exit the Alternate Keypad Mode, respectively (lines 15-20). Line 21 sets the terminal to the character mode, without echo, and eliminates the need to deal with the ASCII line feed character. Lines 22 and 23 set up the program prompt and lines 24 through 28 create an HDOS '.SCIN' loop to detect the receipt of the ASCII ESCape character.

If an ESCape character is encountered, the program processes it in a subroutine called at line 27, 'PCC'. This subroutine makes use of the HDOS TIC COUNTER. The TIC COUNTER increments every 2mS, thus, by adding to the initial tic count a value of 10 and testing for the counter's incrementing beyond that value, a timing loop is established. Within this loop the program tests for the receipt of a second character. The PCC loop uses the HDOS '.SCIN' system call to detect receipt of that character.

If the PCC subroutine detects a second ASCII character within 20mS, the second character is used to identify the special function key (lines 38-56) struck. If no second character is received within the 20 mS loop, the program returns to its first '.SCIN' loop and nothing occurs on the screen.

As previously stated, in the Alternate Keypad Mode, three characters are sent. In that mode, when an alternate keypad key is struck, the first character is again the ASCII 'ESC' followed by the ASCII '?' character. Thus, line 38 tests for '?', and if detected, this causes a jump to the 'AKP' subroutine, where a third '.SCIN' system call detects the last and identifying character. In the example program, only the 'HOME' and 'arrow' keys of the alternate keypad are identified by the program. For the sake of program brevity, the seven other alternate keypad keys are identified as such, but not specifically. It can be safely assumed that if the program receives an 'ESC' and a '?' within 20mS, only an alternate keypad key could have been struck, and for that reason, subroutine 'AKP' does not require a timing loop.

The need for incorporating a timing loop in the input subroutine, 'PCC', becomes obvious if lines 29, 30, 31, 34 and 35 are disabled (place a '*' at the beginning of these lines before assembling) and reassembling the program. Now, while the program still responds as advertised, note that the user can "fool" the program by entering the individual character components of the special function sequences. For example, by striking the 'ESC' key followed by the 'R' key, the program "thinks" that the GRAY KEY has been struck and will so announce. In the shifted keypad mode, the individual components of those keys can be entered with similar results. For example if one strikes the 'ESC' key followed by the '?' key and then the 'r' key, the program will identify the 'down arrow' key as having been struck.

In summary, the special function keys generate two or three character sequences. These sequences always begin with the ASCII 'ESC' character, and a second character, the "identifier". In the alternate keypad mode, the three character sequence is generated; first the 'ESC' character, followed by the '?' character, followed by the identifying character. They allow the programmer to design programs capable of calling as many as 20 separate subroutines during runtime. This can be particularly useful in calling subroutines in programs receiving alphanumeric character inputs.

About The Author:

*Dr. **William Rousseau** is an anesthesiologist practicing in Salem, Ohio. His interests in electronics go back at least 30 years when he built his first Heathkit. He is keenly interested in digital electronics as well as computer hardware and software.*

CONC.BAS File CONCentrator

Roger Evans
11657 64A Ave.
Delta B.C.
Canada V4E 2C8

I own one of the earliest H-89 systems purchased back in 1979, when not even the software was available to run it. Due to previous kit building experience having made me familiar with Heath products and the local store, I had faith in them and as expected, the store came through with the software about a week after I had finished the kit. This was just as well as I was getting fed up just writing characters on the screen in off-line mode, not even being able to boot-up.

Since those early days, the 89 has accumulated several thousand hours of run time. Mostly programming and small business operations using custom software. It is from the needs of the small business operation that the feature program of this article was conceived. The 89 is still in its original hardware configuration, with 48K of memory and the built in single sided, single density Perkin Elmer drive.

The primary limitation of the system for the small business operator, and many home computer owners as well, is the capacity of the disk drive. As most single disk drive owners are aware, by the time you get the operating system, a language, and the program all on the same disk, there is precious little room for data.

CONC.BAS is a program CONCentrator which deletes spaces, tabs, and remarks from BASIC programs, enabling them to be stored in considerably less space and making them run more efficiently. Written in Microsoft BASIC, the program is readily adaptable to other BASIC dialects, and if the file handling is changed, will run on CP/M. The listed version is for HDOS.

The program is best copied onto a new floppy which contains BASIC, and whichever device drivers are required for your system. This disk will then become your concentrator disk. Single drive systems should make the disk bootable, but delete non-essential system files to give more working space on the disk. I usually delete ERRORMSG.SYS, SET.ABS, SYSHELP.DOC, HELP., and FLAGS.ABS. The sw flags must be unset first, using the FLAGS program. This saves about 32 sectors. A utility disk doesn't normally need the date, so to make boot-up easier, I usually >SET HDOS NODATE. Do this before you delete the SET program.

Having copied CONC.BAS onto the disk, you should Debug using a short test routine. This one works fine:

```
10 REM File this as "TEST.BAS",A
20 ' These first 2 lines will be deleted
30 FOR I=1 TO 10:   'Test loop
40    PRINT"This is a  test": 'Tab deletion,  spaces remain when in quotes.
50 NEXT I
60  A=0: B=0:  C=0:  D=0:'  Colons  remain in multi  statement  lines.
70 PRINT"Exit program.": END
```

File TEST.BAS is an ASCII file. This is the ,A extension for Microsoft BASIC. You can, of course, run TEST.BAS and TEST.CNC to prove the action of the concentrator. Both versions should produce exactly the same results. After correcting any typos, run CONC.BAS and type CONC to create a concentrated version named CONC.CNC. This should store in about 10 sectors. Copy the full version over to a backup disk, and delete CONC.BAS from your concentrator disk. Then rename the concentrated version using:

>RENAME CONC.BAS=CONC.CNC

From now on you can load and run by calling:

>MBASIC CONC

Load the new version and stop execution with a ↑C, then SAVE it. This will store on disk in Microsoft compressed format and in the future, will load faster. This completes the preparation of the concentrator disk.

Concentrator Procedure

1. Boot-up on the concentrator disk and call >CAT. Determine that there is enough free space to hold both the BASIC file you wish to condense, and its concentrated version when they are both saved in ASCII format. The version generated by CONC.BAS will be about 70% of the size of the source program. Delete any old files.

2. Use ONECOPY to copy the source program onto the concentrator disk. Load and list it to check completeness, then file it as an ASCII file.

3. Load and run CONC.BAS and type the name of your program. The BAS extension is not required. A large program will run for about 15 minutes (48 sectors). You can watch it being processed on the screen. The bell will sound completion and the number of deletions is given.

4. Load the concentrated version, xxxxx.CNC, and call RENUM. Renumbering the lines will indicate if any GOTO or GOSUB statements were directed to REM lines which have now been deleted. This is one way to cure a bad habit. Fix up any errors, and test. Save the concentrated version in the normal manner, and then copy it back to its working disk.

5. DELETE your program and its .CNC version from the concentrator disk to make it ready for next time.

The total size reduction from ASCII to .CNC compressed format can be as high as 40%, depending on the number of multi-statement lines, and how heavily loaded with remarks the program is. There is often a noticeable increase in the speed of operation of the program.

A major advantage which is not so obvious, is for programs which use large amounts of string space. These are usually editors, word processors, and spread sheet programs. The concentrated version will use less memory to contain it, so if the .BAS version has (say) CLEAR 5000, then this can be upgraded to about 8000 bytes. This speeds response time and is especially noticeable when handling large files.

Listing

```
10 REM %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%. CONC %%%%%%%%%%%%%%.
20 REM Z$=characters in process. X$=Input string. V$=Char buffer. L=Len string
30 REM K=Chars processed. J=# chars in Z$. I=Increment along Z$. C=Counter.
40 'F=Flag delete line no. G=null count. W$=line no. buffer. Y$=quotes buffer.
50 PRINT CHR$(27)+"E": 'Clear screen
60 PRINT TAB(23)"CONC. File CONCentrator Revision 0.1"
70 CLEAR 10000: PRINT: PRINT"The concentrator removes redundant spaces "+@
    "& remarks from ":PRINT "BASIC programs to conserve memory & disc space.":@
     PRINT
80 PRINT"The input file must be an ASCII file, not Microsoft compressed format."
90 PRINT"Use  SAVE'(filename)',A  to convert to ASCII.": PRINT"Default suffix is .BAS"
100 PRINT: PRINT"CAUTION. The program that is being concentrated MUST NOT":@
    PRINT "have goto & gosub statements directed to remark statements."
110 PRINT"These are removed by the concentrator."
120 PRINT: LINE INPUT"Name of existing file to be translated, as CSORT --- ";A$
130 PRINT: R1$=CHR$(27)+"p": R2$=CHR$(27)+"q": 'Reverse video.
140 D$=".BAS": A$=A$+D$
150 A=LEN(A$): B=A-3: B$=LEFT$(A$,B): C$="CNC": B$=B$+C$
160 PRINT"Input file is  --- ";A$;"    Output file is --- ";B$: PRINT
170 OPEN "I",1,A$: OPEN "O",2,B$
180 REM ----------------------------------------------------------------- BUFFER
190 LINE INPUT #1, X$
200 L=LEN(X$): G=0: F=1
210 FOR H=1 TO 6: 'Locate Ist space & remember line no.
220      J=1:I=I+1: Z$=MID$(X$,I,J): 'Get char.
230      IF Z$=CHR$(34) THEN GOSUB 600: 'Quotes
240      IF Z$=CHR$(9) THEN PRINT W$;" ";R1$;"    ";R2$;: F=0: C=C+1: @
         PRINT #2, W$;" ";: GOTO 290: 'Tab
250      W$=W$+Z$: 'BUFFER
260      IF Z$=" "THEN 290
270 NEXT H
280 '----------------------------------------------------------------- SORTER
290 FOR K=1 TO L
300      J=1: I=I+1: V$=""
310      Z$=MID$(X$,I,J): 'Get next J char(s).
320      IF Z$=" " THEN PRINT R1$;Z$;R2$;: C=C+1: GOTO 440: 'KILL SPACES.
330      IF Z$="R" THEN V$=Z$: I=I+1: J=2: GOTO 310
340      IF Z$=":" THEN V$=Z$: I=I+1: J=4: GOTO 310
350      IF Z$="'" THEN F=0: GOSUB 580: GOTO 440: 'Remove REM line.
360      IF V$<>"" THEN 480
370      IF Z$=CHR$(34) THEN GOSUB 600: GOTO 430: 'Quotes
380      IF Z$=CHR$(9) THEN Z$="": PRINT R1$;"    ";R2$;: C=C+1: 'Kill tab.
390      IF LEN(Z$)>1 THEN 480
400      IF Z$="" THEN G=G+1
410      IF G>3 THEN K=L: GOTO 440: 'NULL counter for E.O line.
420      IF F=I THEN PRINT W$;: PRINT #2, W$;: F=0: 'Flag, print line number.
430      PRINT Z$;: PRINT #2, Z$;: 'Print normal
440 NEXT K
450 PRINT: PRINT #2,: I=0: IF EOF(1) THEN CLOSE: GOTO 660
460 Z$="": X$="": W$="": Y$="": V$="": GOTO 190
470 ' -------------------------------------------------------------- CONVERTER
480 IF V$=":" AND Z$=" REM" THEN I=I+5: Z$="": GOSUB 580: GOTO 440
490 IF V$=":" AND Z$="REM " THEN I=I+4: Z$="": GOSUB 580: GOTO 440
500 IF V$=":" AND LEFT$(Z$,2)=" '" THEN Z$="": GOSUB 580: GOTO 440
510 IF V$=":" AND LEFT$(Z$,1)="'"  THEN Z$="": GOSUB 580: GOTO 440
515 IF V$=":" AND LEFT$(Z$,3)="  '" THEN Z$="": GOSUB 580: GOTO 440
520 IF V$=":" THEN Z$=":": I=I-1: GOTO 430
540 IF V$="R" AND Z$="EM" THEN F=0: GOSUB 580:  GOTO 440: 'Remove REM line.
550 IF V$="R" THEN Z$=V$: I=I-1: GOTO 420
560 STOP
570 REM -------------------------------------------------------------- PRINT REMS
580 PRINT R1$;" ";R2$;: K=L: C=C+1: RETURN
590 '  ------------------------------------------------------------ "STRING SUBR."
600 J=1:PRINT Z$;: PRINT #2, Z$;: Y$=""
610 FOR M=1 TO L
620      I=I+1: Z$=MID$(X$,I,J): Y$=Y$+Z$
630      IF Z$=CHR$(34) THEN Z$=Y$: RETURN
640 NEXT M: RETURN
660 PRINT: PRINT A$;" concentrated into ";B$;"      ";R1$;C;R2$;"DELETIONS."
670 PRINT CHR$(7);"Exit program.": END
```

Console Status Program

Robert G. Traub
C.P.O. Technical Services Inc.
9731 - 154th Street
Edmonton, Alberta
CANADA T5P 2G4

Having owned a Heathkit H8 microcomputer since their introduction, I have had an opportunity to write a number of utility programs for the unit. Recently, after upgrading the system to incorporate the Z-80 microprocessor, I have started converting the source code for these utilities to Z80 mnemonics. The original programs, written in 8080 code, will of course operate on the new Z80 board as they are. However, they can be condensed, requiring fewer bytes in memory, and operate faster if converted to utilize the features inherent in the Z80 instruction set. One of the shorter utility programs written was designed to provide the user with a print out of the current console settings as last performed by the HDOS 'SET TT:' command. This program is called 'CONSTAT.ABS' for 'console status' and presents an opportunity to use instructions not found in the 8080 microprocessor to great advantage.

The console status program will use a single byte, which is stored in memory, and test each bit of the byte in order to determine which of the two possible settings the bit represents. The 8080 instruction set does not have an instruction that would allow the programmer to test a single bit of an 8 bit byte. The procedure to test a bit, with the 8080, would involve a number of awkward steps, requiring considerably more bytes in memory. The Z80 microprocessor, on the other hand, has a single instruction that will allow any bit of a byte in memory to be tested.

The byte of interest, stored in memory at address 207DH (040.327) split-byte octal, the CONSOLE COMMAND byte, has the console settings coded in each of its bits. See fig. 1. The bits are numbered from 0 to 7, and are tested by the 'BIT n,(HL)' instruction of the Z80 microprocessor. The order in which the bits are tested, and the two possible settings of each are:

BIT No.	SET To	RESULT
7	0	Console will not process backspace key
7	1	Console will process backspace key
6	0	Console will not process Form-Feed Char.
6	1	Console will process the Form-Feed Char.
5	0	Console will not allow lower case on input
5	1	Console will allow lower case on input
4	0	Console will not allow lower case on output
4	1	Console will allow lower case on output
3	0	Console is set for 2 stop bits
3	1	Console is set for 1 stop bit
2	0	This bit is not used (always 0)
1	0	Console will not map rubout key to backspace
1	1	Console will map rubout key to backspace
0	0	Console will not support tab character
0	1	Console will support tab character

Figure 1

As each bit can represent one of two different possibilities, it is not difficult to write a program to simply check each bit of the byte and print out the correct message depending on how that bit was set - 1 or 0.

The CONSOLE COMMAND byte is loaded into memory at address 207FH each time the system is booted up. Whenever a SET TT: command is performed, the new setting is placed in memory and on the diskette. In this manner, the byte loaded into memory each time the system is booted up will always reflect the last console setting.

There are other memory cells that hold information about the console, such as the console width setting, the date, and the HDOS version number. These will not be used in this program, but the address of the memory cells have been included in the 'EQU' statements, this is done so you may expand the program if desired.

This program is assembled using the HDOS version of MACRO-80 or M80, LINK-80 (L80), CREF.ABS. The source code was prepared with the Heath editor EDIT.ABS.

This program consists of seven repeated occurrences of the same basic routine. The only difference between each routine is the labels and the message that will be displayed. For this reason, the first routine will be explained in detail, and the rest are understood to operate in the exact same manner. First however, we must set up the program's stack area and origin. This will be followed by a small routine to turn up two new lines on the console and display a 'sign-on' message. We can define the physical start of the program with the statement:

```
        ORG     USERA   ;Physical start of program in memory
```

The next thing to do is set up the stack area and turn up two new lines. This is done with:

```
BEGIN:  LD      SP,STACK        ;Load Stack address in
                                            'SP' register pair
        LD      A,NMLW  ;Get new line character in 'A' register
        DW      SCOUT   ;Let HDOS show new line on console
        CW      SCOUT   ;Show another new line
;
```

At this point we can insert a 'SIGN-ON' message with:

```
        LD      HL,MESS20       ;Get sign-on message
                                        address in 'HL' reg.
        DW      PRINT   ;Let HDOS display message on console
```

We are now ready to write the first of seven identical routines. Each of the routines will have a different label and will print different messages, depending on the bit being tested, but they all work in exactly the same way. The first routine will test bit number 7 (MSB) of the command word to see if the console routine will allow the backspace key to be processed or not. This is done as follows:

```
BKS:    LD      HL,SCSLMD       :Get command byte address
                                            in 'HL' register
        BIT     7,(HL)  ;Test bit 7 of byte pointed to by 'HL'
        JR      Z,BKS1  ;If it was 0, exit
        LD      HL,MESS1        ;If 1, Get address of
                                      message 1 in 'HL' reg.
        DW      PRINT   ;Let HDOS display message on console
        JR      FORM    ;Go test next bit now (6)
BKS1:   LD      HL,MESS2        ;Get address of message 2
                                           in 'HL' reg. pair
        DW      PRINT   ;Let HDOS display message on console
```

This routine will test the designated bit of the byte located in memory at the address in the 'HL' register pair. The high order byte of the address is in the 'H' register, and the low order byte of the address is in the 'L' register. In order to test the bit, we first load its memory address into the 'HL' register pair, the address was given the label 'SCSLMD', and the instruction 'LD HL,SCSLMD' places the address into the 'HL' register pair.

This being done, we can now test the desired 'bit'. The next instruction 'BIT 7,(HL)', informs the microprocessor to test bit number 7 of the byte labeled SCSLMD. When this instruction is carried out, the ZERO FLAG in the flag (F) register will be set to the COMPLIMENT of the bit tested. That is, if the bit tested is a zero, the 'Z' flag will be set to one. If the bit tested is a one, the 'Z' flag will be set to zero.

The next instruction, 'JR Z,BKS1', informs the microprocessor to go to address BKS1, if the ZERO FLAG in the flag register was a one. We can summarize this as:

BIT IS	'Z' FLAG	CONDITIONAL JUMP
0	1	Yes, add offset and exit
1	0	No, do next sequential instruction

The conditional jump instruction (JR Z,ADDR) will add the displacement or offset amount to the address in the program counter and 'jump' to that address if the ZERO flag is a one. If the ZERO flag is a zero, the next instruction in the program will be carried out.

Assuming the bit tested was a one, the 'Z' flag will be set to zero and the next sequential instruction will be carried out. This instruction tells the microprocessor to put the address of the message 'MESS1' into the 'HL' register pair. In this way, the 'HL' register pair is pointing to the address of the start of the correct message. A 'call' is then made to the HDOS operating system and the routine located there will print out the message and return to the next sequential instruction. This is done with the instructions:

```
        LD      HL,MESS1        ;If 1, Get address of
                                        message 1 in 'HL'
        DW      PRINT   ;Call HDOS 'Print' routine
        JR      FORM    ;Go test next bit now
```

The last instruction, 'JR FORM', is a non-conditional jump instruction. This tells the microcomputer to exit the program at that point and go to the given address (FORM).

Assuming the bit tested was a zero, the 'Z' flag will be set to one and the displacement will be added to the address in the 'PC' register, and the program will 'jump' to the address given by the label 'BKS1'. At this point, the address of the start of message number two (MESS2) will be loaded into the 'HL' register pair and a call to HDOS will be carried out in the same manner as explained. After this message has been displayed on the console, the program will then 'Fall through' to the next sequential routine.

The complete listing for the program has been included at the end. This program represents only a few of the unique instructions available with the Z80 microprocessor, and offers an insight to the advantages of the chip.

```
; PROGRAM TO DISPLAY SETTINGS OF HDOS 'COMMAND BYTE'
; FOR TT:  BY: ROBERT G. TRAUB  MARCH 1983
; Z80 REVISION
;
        ASEG
SDATE   EQU     20BFH   ;SYSTEM DATE (IN ASCII - 9 BYTES)
SCSLMD  EQU     20D7H   ;CONSOLE COMMAND BYTE ADDRESS
SCONWI  EQU     20D9H   ;CONSOLE WIDTH SETTING ADDRESS
SCIN    EQU     01FFH   ;CONSOLE INPUT ROUTINE ADDRESS
SCOUT   EQU     02FFH   ;CONSOLE OUTPUT ROUTINE (BYTE AT A TIME)
PRINT   EQU     03FFH   ;CONSOLE OUTPUT ROUTINE (STRINGS)
VERS    EQU     16FFH   ;ADDRESS OF DOS VERSION NUMBER
SEXIT   EQU     00FFH   ;SYSTEM EXIT ROUTINE
STACK   EQU     2280H   ;STACK ADDRESS (BUILDS DOWN)
USERA   EQU     2280H   ;PHYSICAL START OF PROGRAM
NWLN    EQU     0AH     ;ASCII NEW LINE CODE
;
        ORG     USERA   ;PHYSICAL START OF PROGRAM IN RAM
;
BEGIN:  LD      SP,STACK        ;SET UP PROGRAM STACK AREA
        LD      A,NWLN  ;GET A NEW LINE CHAR IN 'A'
        DW      SCOUT   ;SHOW A NEW LINE ON CONSOLE
        DW      SCOUT   ;SHOW ONE MORE
        LD      HL,MESS20       ;POINT TO SIGN ON MESSAGE
        DW      PRINT   ;SHOW IT ON THE CONSOLE
;
; ROUTINE TO TEST BIT NUMBER 7 FOR
; BACKSPACE SETTING
;
BKS:    LD      HL,SCSLMD       ;POINT TO ADDRESS OF COMMAND BYTE
        BIT     7,(HL)  ;TEST BIT 7 OF COMMAND BYTE
        JR      Z,BKS1  ;IF BIT WAS A 0 - EXIT
        LD      HL,MESS1        ;GET ADDRESS OF 'NO BKS'
        DW      PRINT   ;SHOW MESSAGE ON CONSOLE
        JR      FORM    ;DO NEXT BIT NOW
;
BKS1:   LD      HL,MESS2        ;POINT TO 'BACKSPACE' MESSAGE
        DW      PRINT   ;SHOW IT ON CONSOLE
;
; ROUTINE TO TEST BIT 6, FOR
; FORM FEED CHARACTER PROCESSING
;
FORM:   LD      HL,SCSLMD       ;POINT TO COMMAND BYTE
        BIT     6,(HL)  ;TEST BIT 6 OF COMMAND BYTE
        JR      Z,FORM1 ;EXIT IF BIT WAS A '0'
        LD      HL,MESS15       ;POINT TO 'WILL NOT PROCESS' MESSAGE
        DW      PRINT   ;SHOW MESSAGE ON CONSOLE
        JR      LCSIN   ;GO DO NEXT BIT NOW
;
FORM1:  LD      HL,MESS16       ;POINT TO 'WILL PROCESS' MESSAGE
        DW      PRINT   ;SHOW MESSAGE ON CONSOLE
;
;
; ROUTINE TO TEST BIT 5 FOR
; LOWER CASE ON INPUT
;
LCSIN:  LD      HL,SCSLMD       ;POINT TO COMMAND BYTE ADDRESS
```

```
        BIT     5,(HL)  ;TEST BIT 5 OF COMMAND WORD
        JR      Z,LCSIN1        ;IF BIT IS A '0' - EXIT
        LD      HL,MESS3        ;POINT TO 'WILL NOT' MESSAGE
        DW      PRINT   ;SHOW MESSAGE ON CONSOLE
        JR      LCOUT   ;TEST NEXT BIT NOW
;
LCSIN1: LD      HL,MESS4        ;POINT TO 'WILL ALLOW' MESSAGE
        DW      PRINT   ;SHOW IT ON THE CONSOLE
;
; ROUTINE TO TEST BIT 4, FOR
; LOWER CASE ON OUTPUT
;
LCOUT:  LD      HL,SCSLMD       ;POINT TO COMMAND WORD
        BIT     4,(HL)  ;TEST BIT 4 OF COMMAND WORD
        JR      Z,LCOUT1        ;IF BIT IS A '0' - EXIT
        LD      HL,MESS5        ;POINT TO 'WILL NOT' MESSAGE
        DW      PRINT   ;SHOW MESSAGE ON CONSOLE
        JR      STBIT   ;TEST NEXT BIT NOW
;
LCOUT1: LD      HL,MESS6        ;POINT TO 'WILL ALLOW' MESSAGE
        DW      PRINT   ;SHOW MESSAGE ON CONSOLE
;
; ROUTINE TO TEST BIT 3 FOR,
; NUMBER OF STOP BITS
;
STBIT:  LD      HL,SCSLMD       ;POINT TO COMMAND BYTE
        BIT     3,(HL)  ;TEST BIT 3 OF COMMAND BYTE
        JR      Z,STBIT1        ;IF BIT IS '0' - EXIT
        LD      HL,MESS7        ;POINT TO '2SB' MESSAGE
        DW      PRINT   ;SHOW MESSAGE ON CONSOLE
        JR      RUB     ;DO NEXT BIT NOW
;
STBIT1: LD      HL,MESS8        ;POINT TO '1SB' MESSAGE
        DW      PRINT   ;SHOW MESSAGE ON CONSOLE
;
; ROUTINE TO TEST BIT 1, FOR
; RUBOUT KEY MAPPING
;
RUB:    LD      HL,SCSLMD       ;POINT TO COMMAND BYTE
        BIT     1,(HL)  ;TEST BIT 1 OF COMMAND BYTE
        JR      Z,RUB1  ;IF BIT WAS A '0' - EXIT
        LD      HL,MESS9        ;POINT TO 'DOES NOT' MESSAGE
        DW      PRINT   ;SHOW MESSAGE ON CONSOLE
        JR      TAB     ;DO NEXT BIT NOW
;
RUB1:   LD      HL,MESS10       ;POINT TO 'DOES' MESSAGE
        DW      PRINT   ;SHOW MESSAGE ON CONSOLE
;
; ROUTINE TO TEST BIT '0', FOR
; TAB PROCESSING
;
TAB:    LD      HL,SCSLMD       ;POINT TO COMMAND BYTE
        BIT     0,(HL)  ;TEST BIT '0' OF COMMAND BYTE
        JR      Z,TAB1  ;IF BIT WAS A 'O' - EXIT
        LD      HL,MESS11       ;POINT TO 'DOES NOT' MESSAGE
        DW      PRINT   ;SHOW MESSAGE ON CONSOLE
        JR      EXIT    ;ALL DONE RETURN TO SYSTEM
;
TAB1:   LD      HL,MESS12       ;POINT TO 'DOES' MESSAGE
        DW      PRINT   ;SHOW MESSAGE ON CONSOLE
;
EXIT:   LD      A,0
        DW      SEXIT   ;RETURN TO SYSTEM
;
MESS1:  DB      'TT: WILL PROCESS BACKSPACE',8AH
MESS2:  DB      'TT: WILL NOT PROCESS BACKSPACE',8AH
MESS3:  DB      'TT: WILL ALLOW LOWER CASE ON INPUT',8AH
MESS4:  DB      'TT: WILL NOT ALLOW LOWER CASE ON INPUT',8AH
MESS5:  DB      'TT: WILL ALLOW LOWER CASE ON OUTPUT',8AH
MESS6:  DB      'TT: WILL NOT ALLOW LOWER CASE ON OUTPUT',8AH
MESS7:  DB      'TT: SET FOR 2 STOP BITS',8AH
MESS8:  DB      'TT: SET FOR 1 STOP BIT',8AH
MESS9:  DB      'TT: MAPS BACKSPACE TO RUBOUT',8AH
MESS10: DB      'TT: DOES NOT MAP BACKSPACE TO RUBOUT',8AH
MESS11: DB      'TT: SUPPORTS TABS',8AH
MESS12: DB      'TT: DOES NOT SUPPORT TABS',8AH
MESS15: DB      'TT: WILL PROCESS FORM-FEED CHARATER',8AH
MESS16: DB      'TT: WILL NOT PROCESS FORM-FEED CHARACTER',8AH
MESS20: DB      'YOUR CURRENT CONSOLE SETTINGS ARE:',0AH,8AH
;
        END     BEGIN
```

ATTENTION Z-100 USERS!

We'd like to introduce you to ADS' complete line of S-100 products:

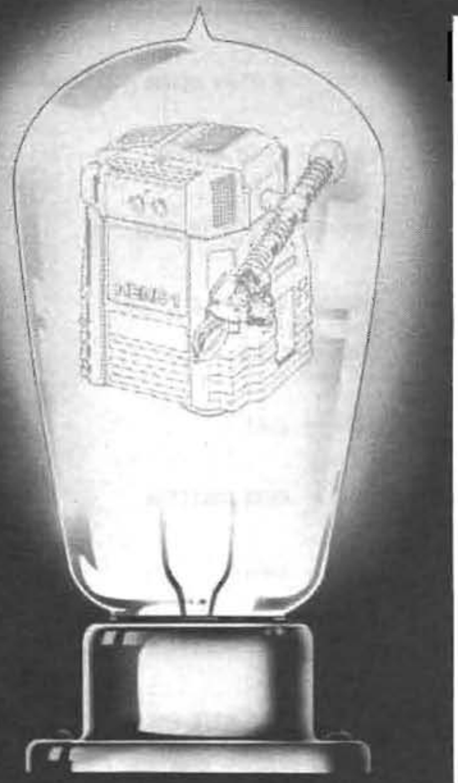
Octafloppy — Disk controller card. Handles eight drives, 5¼", 8" SS/DS & SD/DD.

Synthetalker — Speech synthesis card, uses Votrax SC-01.

Promblaster — programs 19 different eproms from software.

And much more! Call or write today for more information.

Ackerman Digital Systems, Inc.
ads
110 No. York, Suite 208
Elmhurst, IL 60126
(312) 530-8992

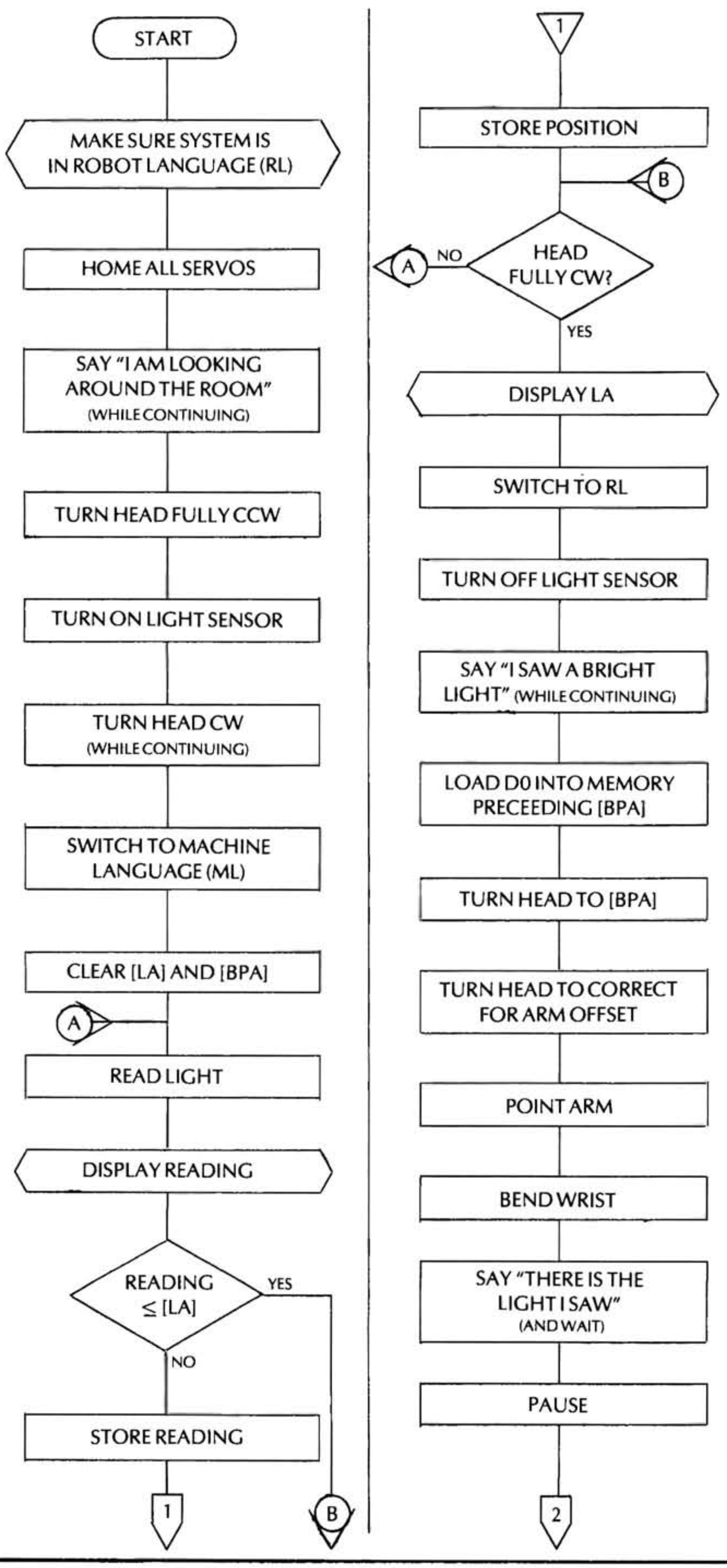


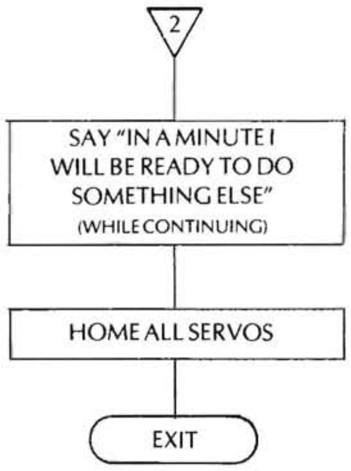
HERO - In Search of the Light

Frank J. Lazorishak
ATES Technical Institute
2076 Youngstown-Warren Road
Niles, OH 44446

Ed: The following program was contributed by Frank J. Lazorishak, Instructor, ATES - Technical Institute. This program was written by Frank as a demonstration of some of HERO's capabilities. As the program executes, HERO scans his environment, finds, and points to the brightest light source. While performing this task, HERO makes various speeches explaining what he is doing.

Frank has indicated to HUG that ATES has integrated HERO into their Associate Degree course in Electronics Engineering as an aid in teaching the principles of robotics.


START
MAKE SURE SYSTEM IS IN ROBOT LANGUAGE (RL)
HOME ALL SERVOS
SAY "I AM LOOKING AROUND THE ROOM" (WHILE CONTINUING)
TURN HEAD FULLY CCW
TURN ON LIGHT SENSOR
TURN HEAD CW (WHILE CONTINUING)
SWITCH TO MACHINE LANGUAGE (ML)
CLEAR [LA] AND [BPA]
A
READ LIGHT
DISPLAY READING
READING ≤ [LA]
YES
NO
STORE READING
1
B
1
STORE POSITION
B
A
NO
HEAD FULLY CW?
YES
DISPLAY LA
SWITCH TO RL
TURN OFF LIGHT SENSOR
SAY "I SAW A BRIGHT LIGHT" (WHILE CONTINUING)
LOAD D0 INTO MEMORY PRECEEDING [BPA]
TURN HEAD TO [BPA]
TURN HEAD TO CORRECT FOR ARM OFFSET
POINT ARM
BEND WRIST
SAY "THERE IS THE LIGHT I SAW" (AND WAIT)
PAUSE
2


2
SAY "IN A MINUTE I WILL BE READY TO DO SOMETHING ELSE" (WHILE CONTINUING)
HOME ALL SERVOS
EXIT

FIND THE LIGHT - VERSION 2.0

BY: FRANK J. LAZORISHAK

LOCATION	CODE	DESCRIPTION
0B00	B6	CHECK TO BE SURE SYSTEM IS IN
1	0E	ROBOT LANGUAGE
2	E1	
3	81	
4	00	
5	26	
6	01	
7	3F	
8	FD	HOME ALL SERVOS
9	00	
A	00	
B	4D	
C	00	
D	00	
E	62	
F	49	
0B10	71	TALK WHILE CONTINUING
1	0B	
2	90	
3	D3	RUN RELATIVE AND WAIT
4	CC	HEAD CCW SLOW
5	03	3 UNITS
6	D3	RUN RELATIVE AND WAIT
7	D4	HEAD CCW MEDIUM
8	03	3 UNITS
9	C3	RUN ABSOLUTE AND WAIT
A	D8	HEAD CCW FAST
B	00	TO STOP
0B1C	41	TURN ON LIGHT SENSOR
0B1D	D3	RUN RELATIVE AND WAIT
E	C8	HEAD CW SLOW
F	03	3 UNITS
0B20	D3	RUN RELATIVE AND WAIT
1	D0	HEAD CW MEDIUM
2	03	3 UNITS
3	CC	RUN ABSOLUTE AND CONTINUE
4	D8	HEAD CW FAST (MAY BE DC)
5	C2	TO STOP (MAY WANT SMALLER NUMBER)
6	01	
7	7F	CLEAR LIGHT ACCUMULATOR [LA]
8	00	
9	40	
A	7F	CLEAR BRIGHTEST POSITION
B	00	ACCUMULATOR [BPA]
C	50	
Ⓐ 0B2D	BD	JUMP TO CLEAR DISPLAY SUBROUTINE
E	F6	
F	4E	
0B30	B6	LOAD ← LIGHT SENSOR
1	C2	
2	40	
3	BD	JUMP TO DISPLAY ACCUM A SUBROUTINE
4	F7	
5	AD	
6	B6	LDAA ← LIGHT SENSOR
7	C2	
8	40	
9	B1	COMPARE ACCA WITH [LA]
A	00	
0B3B	40	
0B3C	23	BRANCH TO Ⓑ IF LS ≤ [LA]
D	0C	Ⓑ RELATIVE
E	B6	LDAA ← [LS]
F	C2	
0B40	40	
1	B7	ACCA→ [LA]
2	00	
3	40	
4	B6	LDAA ← HEAD POSITION
5	00	
6	05	
7	B7	ACCA→ [BPA]
8	00	
9	50	
Ⓑ 0B4A	B6	LDAA ← HEAD POSITION
B	00	
C	05	
D	81	COMPARE ACCA WITH END POINT
E	BE	
F	23	BRANCH TO Ⓐ IF POSITION ≤ C
0B50	DC	Ⓐ RELATIVE
1	BD	JUMP TO CLEAR DISPLAY SUBROUTINE
2	F6	
3	4E	
4	B6	LDAA ← [LA]
5	00	
6	40	
7	BD	JUMP TO DISPLAY ACCUM A SUBROUTINE
8	F7	
9	AD	
0B5A	01	
0B5B	51	TURN OFF LIGHT SENSOR
C	71	TALK WHILE CONTINUING
D	0B	
E	B6	
F	86	LDAA ← D0
0B60	D0	
1	B7	STAA → 004F
2	00	
3	4F	
4	F3	RUN MOTOR TO [BPA] (AS SPECIFIED
5	00	AT 004F AND 0050)
6	4F	
7	D3	RUN RELATIVE AND WAIT
8	D0	HEAD CCW MEDIUM
9	10	OFFSET
A	D3	RUN RELATIVE AND WAIT
B	48	SHOULDER MEDIUM
C	03	3 UNITS
D	C3	RUN ABSOLUTE AND WAIT
E	50	SHOULDER FAST
F	50	TO LEVEL
0B70	C3	RUN ABSOLUTE AND WAIT
1	98	WRIST MEDIUM
2	60	TO LEVEL
3	72	TALK AND WAIT
4	0B	
5	D4	
6	8F	PAUSE
7	00	
0B78	10	
0B79	71	TALK AND CONTINUE
A	0B	
B	F5	

```
    C    FD    HOME ALL SERVOS
    D    00
    E    00
    F    4D
 0B80    00
    1    00
    2    62
    3    49
    4    3A    RETURN TO MONITOR
    5    01
    6    01
    7    01
    8    01
    9    01
    A    01
    B    01
    C    01
    D    01
    E    01
 0B8F    01
 0B90    15    I (AM LOOKING AROUND THE ROOM)
    1    00
    2    09
    3    29
    4    03
    5    2F    AM
    6    00
    7    0C
    8    03
    9    18    LOOKING
    A    16
    B    16
    C    19
    D    0B
    E    14
    F    03
 0BA0    32    AROUND
    1    2B
    2    15
    3    23
    4    2D
    5    0D
    6    1E
    7    03
    8    38    THE
    9    32
    A    23
 0BAB    03
 0BAC    2B    ROOM
    D    37
    E    37
    F    0C
 0BB0    03    (END OF STRING)
    1    FF
    2    01
    3    01
    4    01
    5    01
 0BB6    15    I (SAW A BRIGHT LIGHT)
    7    00
    8    09
    9    29
    A    03
    B    1F    SAW
    C    3D
    D    03
    E    06    A
    F    21
 0BC0    29
    1    03
    2    0E    BRIGHT
    3    2B
    4    23
    5    08
    6    29
    7    2A
 0BC8    03
 0BC9    18    LIGHT
    A    23
    B    08
    C    29
    D    2A
    E    03    (END OF STRING)
    F    FF
 0BD0    01
    1    01
    2    01
    3    01
 0BD4    38    THERE (IS THE LIGHT I SAW)
    5    00
    6    05
    7    00
    8    2B
    9    03
    A    0B    IS
    B    09
    C    12
    D    03
    E    38    THE
    F    32
 0BE0    23
    1    03
    2    18    LIGHT
    3    23
    4    08
    5    29
    6    2A
    7    03
 0BE8    15    I
    9    00
    A    09
    B    29
    C    03
    D    1F    SAW
    E    3D
    F    03    (END OF STRING)
 0BF0    FF
    1    01
    2    01
    3    01
    4    01
 0BF5    0B    IN (A MINUTE I WILL BE READY TO)
    6    09     DO SOMETHING ELSE)
    7    0D
    8    03
    9    06    A
    A    21
    B    29
    C    03
    D    0C    MINUTE
    E    0B
    F    0D
 0C00    00
    1    2A
    2    03
    3    15    I
    4    00
    5    09
    6    29
    7    03
 0C08    2D    WILL
    9    0B
    A    09
    B    18
    C    03
    D    0E    BE
    E    3C
    F    29
 0C10    03
    1    2B    READY
    2    02
    3    00
    4        1E
    5        29
    6        03
    7        2A      TO
    8        37
    9        37
    A        03
    B        1E      DO
    C        36
    D        37
    E        37
 0C1F        03
 0C20        1F      SOMETHING
    1        32
    2        23
    3        0C
    4        39
    5        0B
    6        09
    7        14
    8        03
    9        02      ELSE
    A        18
    B        1F
    C        03      (END OF STRING)
 0C2D        FF
```

Vectored from 9

Patches). I had already figured out how to patch CP/M MBASIC Version 5.21 to cause the delete key to do a (backspace-space-backspace) action which causes the last character to be blanked and the cursor backed up one. My patch is shorter than the one published and it also erases the character instead of just backing up the cursor as the published patch appears to do.

My patch works by using the fact (undocumented as far as I could tell) that MBASIC Version 5.21 uses backslashes when the delete key is pressed, but performs the desired (backspace-space- backspace) action when the backspace key is pressed to delete the last character.

My patch overlays a section of code which originally detected typing a key other than delete after typing one or more deletes and in that case, put out the ending backslash before echoing the character typed. This code is not needed now since the patch changes the deleting action to not use back slashes.

Use the following series of commands to patch CP/M MBASIC Version 5.21 for the H-8, H/Z-89 computers (Is this the same MBASIC used with the H/Z-100 series?):

```
A>DDT MBASIC.COM
NEXT PC

6100 0100
-A4BDD
4BDD JNZ 4BF1
4BE0 MVI A,08
4BE2 JMP 4C21
4BE5 (CR ONLY)
-↑C
A>SAVE 96 MBASIC.COM
```

This patch detects the typing of a delete key and replaces it in the accumulator by a backspace character, then jumps to a section of code already in MBASIC set to put out a (backspace-space- backspace) sequence when the backspace key is typed.

I just discovered that the MBASIC Version 5.21 I have allows underscore characters as input and doesn't interpret them as delete characters. Maybe this means that the CP/M MBASIC for the H-8 and H/Z-89 is different from that for the CP/M-85 on the H/Z- 100 machine, as your article seemed to indicate that MBASIC interpreted the underscore as a delete and your article mentioned MBASIC in connection with CP/M-85. If this is true, my patch wouldn't work for MBASIC for CP/M-85.

William H. Yost
3821 Trilbey Dr. Apt. D
Indianapolis, IN 46236

More On Z-BASIC & Flags Program

Dear HUG,

This letter is in response to your article "Introduction To Z- BASIC Part VI" in the May Issue of REMark. I found the article interesting and decided to enter the FLAG.BAS program in my computer (H-100). When I got the program to run, I noticed that the flags displayed were not correct when I compared them with the encyclopedia. The U.S. flag has three white stripes below the blue field. The program displays a flag with only two white stripes below the blue field. I also noticed that the flag with 13 stars had the circle of stars rotated 90 degrees. So I decided to make some revisions.

The attached program listing shows these revisions. In addition to making corrections to the flags as noted above, I decided to try drawing a different star. The result requires more program lines but provides a more proportional star. For my own interest in

```
100 :'          FLAG.BAS        J. SLATTER      5-29-83
110 :
120 CLS
130 PSET(7,1),6 '                                        DRAW STAR
140 LINE(6,2)-(8,2),6
150 LINE(5,3)-(9,3),6
160 LINE(1,4)-(13,4),6
170 LINE(3,5)-(11,5),6
180 LINE(5,6)-(9,6),6
190 LINE(4,7)-(6,7),6
200 LINE(8,7)-(10,7),6
210 LINE(3,8)-(4,8),6
220 LINE(10,8)-(11,8),6
230 DIM STAR#(100)
240 GET(0,0)-(14,8),STAR#:'                                  STORE IMAGE
250 :
260 CLS:LOCATE 25,30: PRINT "PRESS ANY KEY TO STOP"
270 :
280 CIRCLE(25,7),15,6: PAINT(30,9),6:'                       TOP OF POLE
290 :
300 PSET(30,15),7: DRAW"M39,20D155M37,233":'                 ROPE
310 :
320 LINE(20,15)-(30,225),6,BF:'                              POLE
330 :
340 FOR I=1 TO 13: J=I MOD2:'                                RED & WHITE
350 IF J=1 THEN CL=4 ELSE CL=7:'                             STRIPES
360 LINE(40,I*12+8)-(600,I*12+20),CL,BF: NEXT I
370 :
380 LINE(40,20)-(252,103),1,BF:'                             BLUE FIELD
390 :
400 IF R<>0 THEN 470
410 R=33: PI=3.14159: FOR I=1 TO 13:'                        FIRST U.S. FLAG
420 C=(I*(360/13))*PI/180:'                                  WITH 13 STARS
430 PUT(142+(INT(COS(C-PI/2)*R))*2,
       57+INT(SIN(C-PI/2)*R)),STAR#
440 NEXT I: FOR T=1 TO 1000, NEXT T
450 GOTO 530
460 :
470 FOR A=36 TO 220 STEP 18:'                                CURRENT U.S. FLAG
480 FOR B=1 TO 9:'                                           WITH 50 STARS
490 IF (B/2<>B\2 AND A/36<>A\36) OR
       (B/2=B\2 AND A/36=A\36) THEN 510
500 PUT (A+14,B*8+17),STAR#,XOR
510 NEXT B,A:FOR T=1 TO 1000: NEXT T
520 R=0
530 A$=INKEY$:IF A$="" THEN 380 ELSE CLS:LIST
```

doing the same thing in a different way, I used modulo division instead of floating point integer division for generating the stripes. One other addition is a short time delay for viewing each flag.

Jerry Slatter
5032 Lk. Wa. Blvd. NE
Renton, WA 98056

Dear Editor,

HELP! I have no one to HUG (at least not another Heath user) for several thousand miles around me, and have an annoying little printer problem.

I have an H-8, H-19 (installed in a grass-roofed hut, the only such in the world which is computerized) and have recently added a Microline 82 printer-not 82A or B, mind you, just 82, printing 80 CPS.

The little gem has the annoying habit of losing a character at the beginning of a line every little once in awhile. I suspect that it is losing the end or beginning of a buffer. I am using the H-14 DVD, which is satisfactory in all other respects, an H- 8-4 card, and am hooked up as follows:

	PRINTER	H-8-4
pin	1	1
	3	5
	6	11
	7	13
	20	7

I have tried the printer in both SSD and DTR mode, same results.

Perhaps there is a HUGgie who has had the same problem? I have tried setting null characters after CR and LF, and a few others, to no effect.

I enjoy REMark from here in my lonely but lovely outpost, and would like to see more on the H-8, despite its antiquity.

Robert J. Dixon
B.P. 934
Papeete, Tahiti
Polynesie Francaise

CP/M For The H8/H19

Dear HUG,

I have been reading REMark since it began and found some good and useful information in it. Now it is my turn to pass on something that I have learned which I hope will be helpful to others. Just recently I have acquired CP/M for my H8/H19 system. In using the program "List.com" supplied on the disk, I discovered a bug when using it in the prompt "*" mode. For those who don't know what "List.com" is, it is a CP/M program that allows you to print any printable file in the LST: device with pagination, headings, tab expansion, and other optional features.

Everything works well as far as I can tell, except for the default pagination. If you do not specify what page number you want to start with, the program assumes page 1. If you print only one file all is well, but if you want to print a second file without leaving List.com, the default page number starts at 1 space, then page 2 gets numbered 1!, page 3 - 1", and so on.

The problem is very simple to fix, since all that is wrong is the original page default stored in the program as loaded from disk is stored in memory in the wrong order.

Here is how to fix it using DDT. The user types what is in bold print, the other is the computer response.

```
A>DDT LIST.COM
     DDT VERS 2.2
     NEXT      PC
     0700      0100
-S642
0642 31 20
0643 20 31
0644 3C Control D
?
-Control C
A>SAVE 6 LIST.COM
```

You now have a corrected List.com program. Also, if you enter on the first file printed a page number other than 1, then that number becomes the default starting page number for all additional files printed. Be sure to set the default back to page 1 if you indeed want the default to start at page 1 for the remaining files.

I also have a question. When is Heath or somebody from HUG going to update or modify a new EPROM for the H-25 printer? There is so many printers that do dot addressable graphics and downloadable character fonts that I wonder why Heath/Zenith hasn't come up with something to make the H-25 more competitive with these printers. I know there is EPROM space available in the H-25 and the printer is good quality and reliable. So hey, don't abandon it - support it!!

Richard J. Mazur
10 Welsh Road
Pittsburgh, PA 15203

NEXT MONTH
CONVERTING IBM
BASIC TO ZBASIC
WATCH FOR
IT!

Vectored from 9

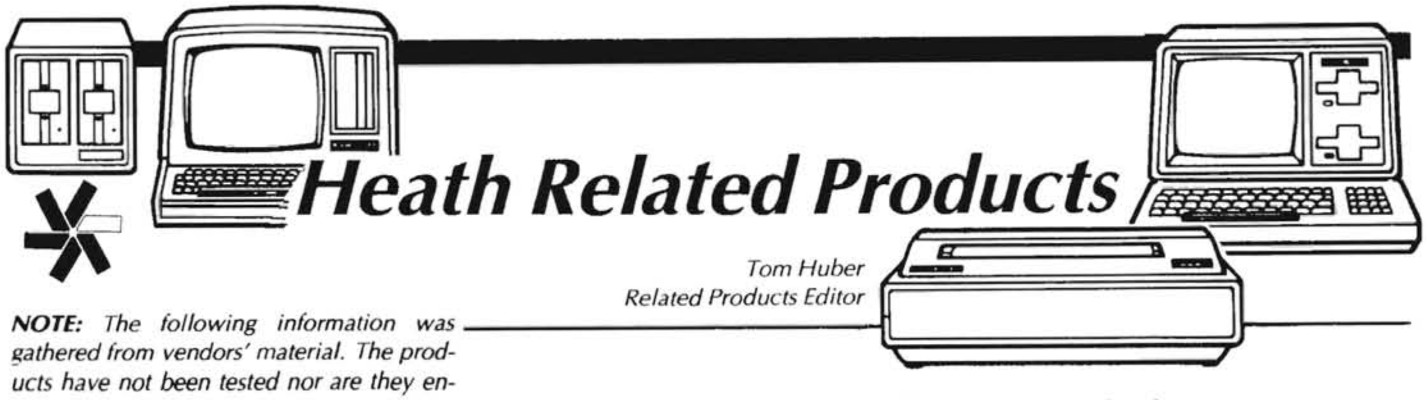
Q. I see a lot of software available for CP/M systems. Now the question is, before I go out and spend $150 on the CP/M operating system for my H-89, how much of this advertised CP/M compatable software will I be able to use? Or will I be locked into Heath/Zenith and HUG software only?

A. Yes, there is much software written under the CP/M operating system. That is one of the primary reasons for the popularity of CP/M. There are many software vendors that support programs specifically for the Heath/Zenith computer equipment. (Refer to advertising in each issue of REMark.) Other vendors write software that is non-hardware dependent (i.e. not written for a particular computer), which will run on your H89.

It is important that your system matches the software specifications. The following is a quick-check list for purchasing non-Heath/Zenith vendor software:

☐ 1) It is CP/M-80 based (runs under 8080 or Z80 processor).
☐ 2) It does not use the hardware of another computer (e.g. screen output).
☐ 3) It is available in Heath/Zenith CP/M format (i.e. all CP/M 5 1/4" disk formats are not the same).
☐ 4) Your computer has enough memory.
☐ 5) The software is written in a language you have.
☐ 6) The source code is available.

There is a tremendous amount of public domain CP/M software. Much of it is useable by the Heath/Zenith CP/M owner however, the majority will require some modification by the user to get the software running properly.



Heath Related Products

Tom Huber
Related Products Editor

NOTE: The following information was gathered from vendors' material. The products have not been tested nor are they endorsed by HUG. We are not responsible for errors in descriptions or prices.

Z-DOS Version of dBase II

dBASE II, Ashton-Tate's popular relational data base management system, is now available for the H/Z-100. Many consider dBASE II the most powerful relational data base system available for microcomputers. It features English commands that allow the user to easily create sophisticated application packages. Sammamish Data Systems, a ZDS and Ashton-Tate dealer, has adapted dBASE II to Z-DOS and is offering it for a limited time at a special introductory price. Contact the vendor for further details.

Vendor: Sammamish Data Systems
1413 177th Ave. NE
Bellevue, WA 98008
(206) 644-2442
Price: $440.00 + $5.00 S&H
Washington State residents add 7.9% Sales Tax.

Type & Print Adapts Olivetti Praxis 30 and 35 to Computer Parallel Printer Port

The Type & Print is a standard Centronics compatible parallel interface that allows connection to the readily available Olivetti Praxis 30 or 35 portable typewriter for a low-cost alternative to dot-matrix printing. The Praxis 35 features a "condensed" print that is perfect for spreadsheets and accounting. Contact the vendor for more information.

Vendor: Applied Creative Technology, Inc.
2723 Avenue E East, Suite 717
Arlington, TX 76011
(817) 261-6905
(800) 433-5373
Price: $179.00 (adapter only, cables & additional interfaces are extra)

AutoCAD, a Microcomputer-aided Drafting and Design Program

AutoCAD is a two-dimensional computer-aided drafting and design program. It is a general-purpose system, suitable for a wide variety of applications, including architectural and landscape drawings; drafting for mechanical, electrical, chemical, structural, and civil engineering; and printed-circuit design. The system supports Summagraphics and Houston Instruments digitizers and all Hewlett Packard and Houston Instruments plotters. Contact the vendor for more details.

Vendor: AutoDesk, Incorporated
16 St. Jude Road
Mill Valley, CA 94941
(415) 381-1819
Prices: $1,000.00
(AutoCAD)
$500.00
(Automatic dimensioning option)

New Boards and Software for Hero I

Perbotics has announced a 44K RAM/communications board, an 8K RAM/communications board, and software (on cassette tape) for loading and dumping Hero's instructions through the RS-232 ports on the new boards. The 44K board includes 48K of RAM while the 8K board provides 20 sockets for user-supplied memory expansion. The software tape includes a memory verification routine for use while expanding the 8K board.

Vendor: Perbotics
211 Costa Mesa Street
Costa Mesa, CA 92627
(714) 645-9294

Prices:	44K RAM board	$795.00
	8K RAM board	$395.00
	Software	$ 49.00

Free Catalog For Computer Interference Control Products

Electronic Specialists, Inc. have announced their latest catalog of computer interference control products. Protective devices for smooth software operation include equipment isolators, AC power line filter/suppressors, line voltage regulators, and AC power interrupters. Request catalog 831.

Vendor: Electronic Specialists, Inc.
171 South Main Street
Natick, MA 01760
Phone: (617) 655-1532

Software Toolworks' Announces Word Wiggle Game

In Word Wiggle, the players try to find as many words as possible in a 4 by 4 square of letters. Competition is against the clock and the computer with its 30,000 word dictionary. Eleven computer skill levels, a 3 by 3 or 5 by 5 square, random or entered letter patterns (especially good for children), and illustrative computer modes are provided. Available for CP/M or HDOS, in 5.25- inch or 8-inch configurations (specify, including computer model - Z-100 users specify Heath/Zenith CP/M and use CP/M-85).

Vendor: The Software Toolworks
15233 Ventura Blvd. Ste. 1118
Sherman Oaks, CA 91403
(213) 986-4885
Price: $29.95

New Report Explains Tax Breaks for Computer Buyers

Research Press, Inc., has announced a new report, *Tax Breaks for Computer Buyers*. The author, Vernon Jacobs, a Prairie Village, Kansas, CPA, explains that most people who buy a computer can probably take a deduction for it on their tax return. He also says that many computer buyers seem to be unaware that they can deduct up to $5,000 of the cost in a single year. In some cases, an employee may be able to get a tax deduction for a personal computer. These and other rules are explained in a concise, 26- page report, available from the vendor.

Vendor: Research Press, Inc.
Box 8137-P
Prairie Village, KS 66208
(913) 362-9667
Price: $9.00

KLEEN LINE Security Provides Protection Against Telephone Line Spikes

Electronic Specialists, long known for their line of power line spike suppression and regulation equipment, has added a telephone line suppression unit (called the KLEEN LINE Security system) to their products. The units provide modem protection for either the standard 4-pin modular connectors (RJ-11) or the wider, 8-pin connectors (RJ-45). The model PDS-11 uses standard 4-pin modular connectors (RJ-11), provides suppression on the red and green lines (pins 3 & 4), with the yellow and black lines feeding straight through. Contact the vendor for more details and pricing on other units.

Vendor:	Electronic Specialists, Inc. 171 South Main Street P.O. Box 389 Natick, Mass 01760 (617) 655-1532
Price:	PDS-11 . . . . $56.95

Communications Software for the '89 and '100

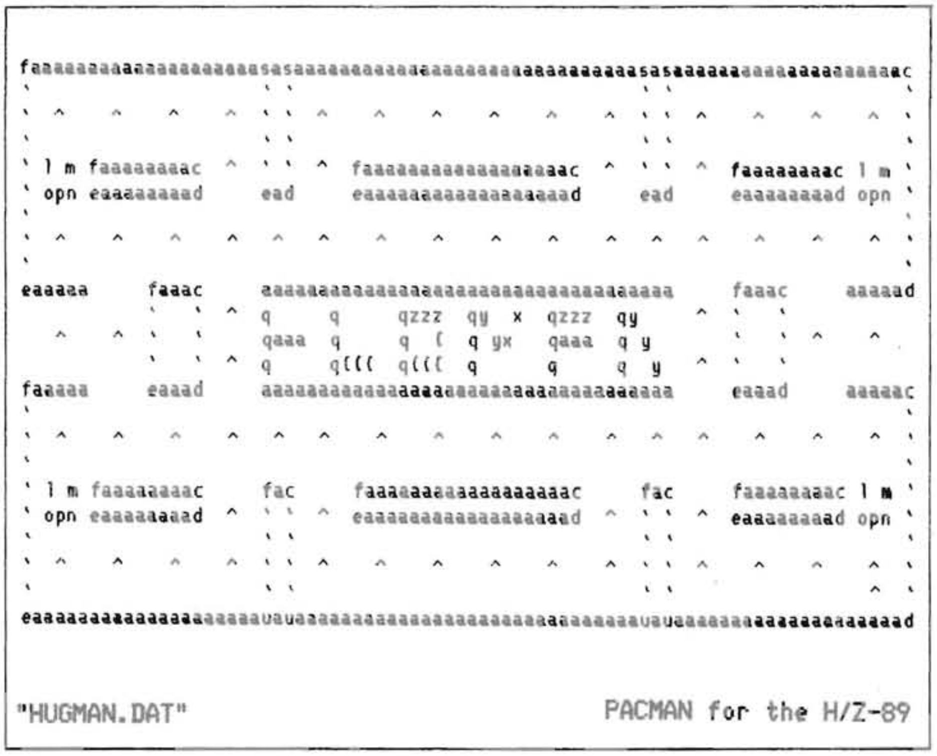
KEA Systems has announced the RCS89 (Remote Communications System) for the H/Z-89 & Z-90, and ZSTEM (Z100 Smart Terminal Emulator) for the H/Z-100. The RCS89 runs under CP/M or HDOS and is written in 8080 assembler code. ZSTEM is written in 8088 assembler code and runs under Z-DOS. Both programs emulate the Z-19 or DEC VT52 control sequences, and include printer and bi-directional disk file transfer support. Contact the vendor for more details.

Vendor:	KEA Systems Dept ZR-1 #311-811 Beach Avenue Vancouver, BC, Canada V6Z 2B5 (604) 687-2744
Prices:	RCS89 . . . . $48.95 ZSTEM $98.95

A High-Resolution Package for the Z-19 Terminal

Northwest Digital Systems has announced Graphics-Plus, a high- resolution graphics package, which is Tektronix ® 4010 compatible, for the H/Z-19 terminal. Features include 512 (horizontal) by 250 (vertical) resolution, selective vector and area erase, any combination of 80/132 column by 24/48 lines of text (both Z-19 and VT™100 character set are included), up to seven pages, menu-driven set-up mode, and sixteen programmable function keys (up to 128 characters each). Contact the vendor for more information.

Vendor:	Northwest Digital Systems P.O. Box 15288 Seattle, WA 98115 (206) 362-6937
Prices:	$849.00 (upgrade - specify H/Z-19 model) $1495.00 (includes Z-19, assembled)



```
faaaaaaaaaaaaaaaaaaasasaaaaaaaaaaaaaaaaaaaaaaaaaaaaaasasaaaaaaaaaaaaaaaaaaaaac
`                   ` `                              ` `                     `
`   ^   ^    ^    ^ ` `   ^    ^    ^    ^    ^    ^ ` `   ^    ^    ^    ^  `
`                   ` `                              ` `                     `
` l m faaaaaaaac  ^ ` `   ^  faaaaaaaaaaaaaaaaaac  ^ ` `   ^  faaaaaaaac l m `
` opn eaaaaaaaad    ead      eaaaaaaaaaaaaaaaaaad    ead      eaaaaaaaad opn `
`                                                                            `
`   ^   ^    ^    ^   ^   ^    ^    ^    ^    ^    ^   ^   ^    ^    ^    ^  `
`                                                                            `
eaaaaa     faaac     aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa     faaac     aaaaad
            `  `  ^  q     q     qzzz  qy  x  qzzz qy      ^  `  `
    ^   ^   `  `     qaaa  q     q   ( q yx   qaaa q y        `  `   ^    ^
            `  `  ^  q     q(((  q(((  q      q    q y     ^  `  `
faaaaa     eaaad     aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa     eaaad     aaaaac
                                                                             `
`   ^   ^    ^    ^   ^   ^    ^    ^    ^    ^    ^   ^   ^    ^    ^    ^  `
`                                                                            `
` l m faaaaaaaac     fac     faaaaaaaaaaaaaaaaaac    fac      faaaaaaaac l m `
` opn eaaaaaaaad  ^ ` `   ^  eaaaaaaaaaaaaaaaaaad  ^ ` `   ^  eaaaaaaaad opn `
`                   ` `                              ` `                     `
`   ^   ^    ^    ^ ` `   ^    ^    ^    ^    ^    ^ ` `   ^    ^    ^    ^  `
`                   ` `                              ` `                  ^  `
eaaaaaaaaaaaaaaaaaaavavaaaaaaaaaaaaaaaaaaaaaaaaaaaaaavavaaaaaaaaaaaaaaaaaaaaad


"HUGMAN.DAT"                                             PACMAN for the H/Z-89
```

Changing your address? Be sure and let us know since the software catalog and REMark are mailed bulk rate and it is not forwarded or returned.

CUT ALONG THIS LINE =

HUG MEMBERSHIP RENEWAL FORM

When was the last time you renewed?

Check your ID card for your expiration date.

IS THE INFORMATION ON THE REVERSE SIDE CORRECT? IF NOT, FILL IN BELOW.

Name ______________________________

Address ______________________________

City-State ______________________________

Zip ______________________________

REMEMBER - ENCLOSE CHECK OR MONEY ORDER

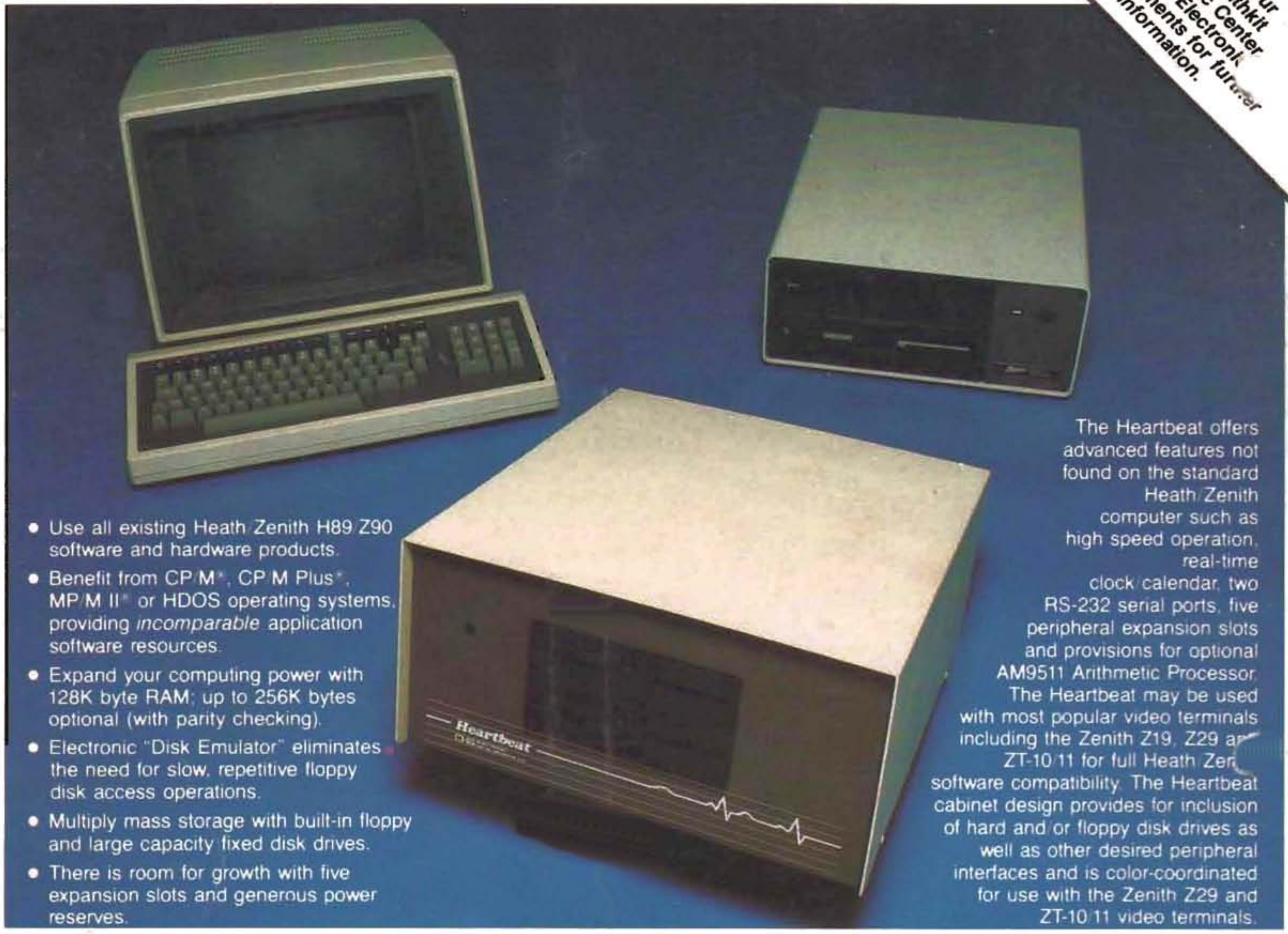
CHECK THE APPROPRIATE BOX AND RETURN TO HUG

	NEW MEMBERSHIP RATES	RENEWAL RATES
US DOMESTIC	$18 ☐	$15 ☐
CANADA	$20 ☐	$17 ☐ US FUNDS
INTERNAT'L*	$28 ☐	$22 ☐ US FUNDS

* Membership in France and Belgium is acquired through the local distributor at the prevailing rate.

In a Heartbeat...

Contact your local Heathkit Electronic Center or DG Electronic Developments for further information.



- Use all existing Heath/Zenith H89/Z90 software and hardware products.
- Benefit from CP/M*, CP/M Plus*, MP/M II* or HDOS operating systems, providing *incomparable* application software resources.
- Expand your computing power with 128K byte RAM; up to 256K bytes optional (with parity checking).
- Electronic "Disk Emulator" eliminates the need for slow, repetitive floppy disk access operations.
- Multiply mass storage with built-in floppy and large capacity fixed disk drives.
- There is room for growth with five expansion slots and generous power reserves.

The Heartbeat offers advanced features not found on the standard Heath/Zenith computer such as high speed operation, real-time clock/calendar, two RS-232 serial ports, five peripheral expansion slots and provisions for optional AM9511 Arithmetic Processor. The Heartbeat may be used with most popular video terminals including the Zenith Z19, Z29 and ZT-10/11 for full Heath/Zenith software compatibility. The Heartbeat cabinet design provides for inclusion of hard and/or floppy disk drives as well as other desired peripheral interfaces and is color-coordinated for use with the Zenith Z29 and ZT-10/11 video terminals.

D·G ELECTRONIC DEVELOPMENTS CO.

**700 SOUTH ARMSTRONG
DENISON, TEXAS 75020
(214) 465-7805**

Heath Users' Group
Hilltop Road
Saint Joseph, Michigan 49085

**BULK RATE
U.S. Postage
PAID
Heath Users' Group**

Volume 4, Issue 8

POSTMASTER: If undeliverable, please do not return.

885-2043