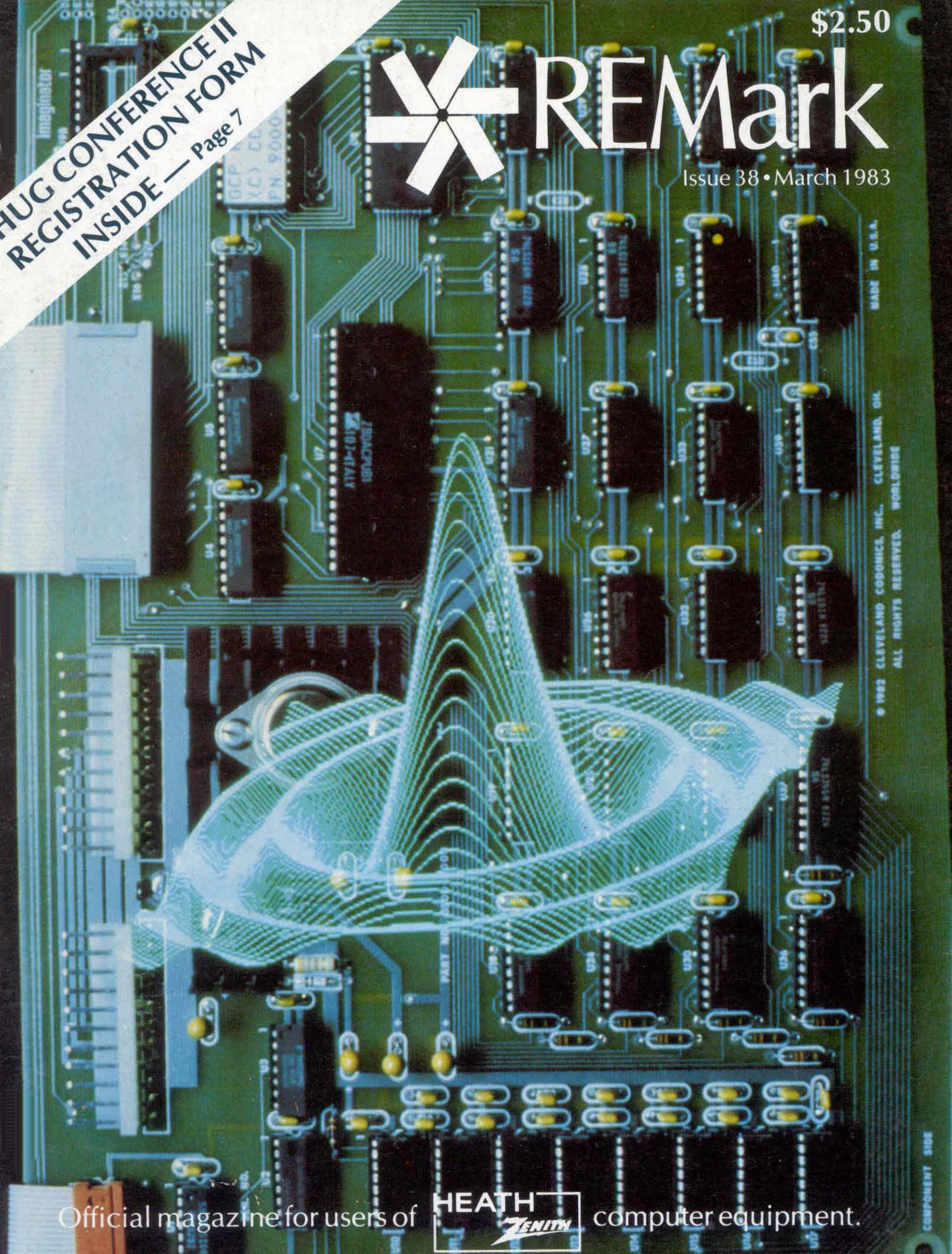
$2.50

REMark

Issue 38 • March 1983

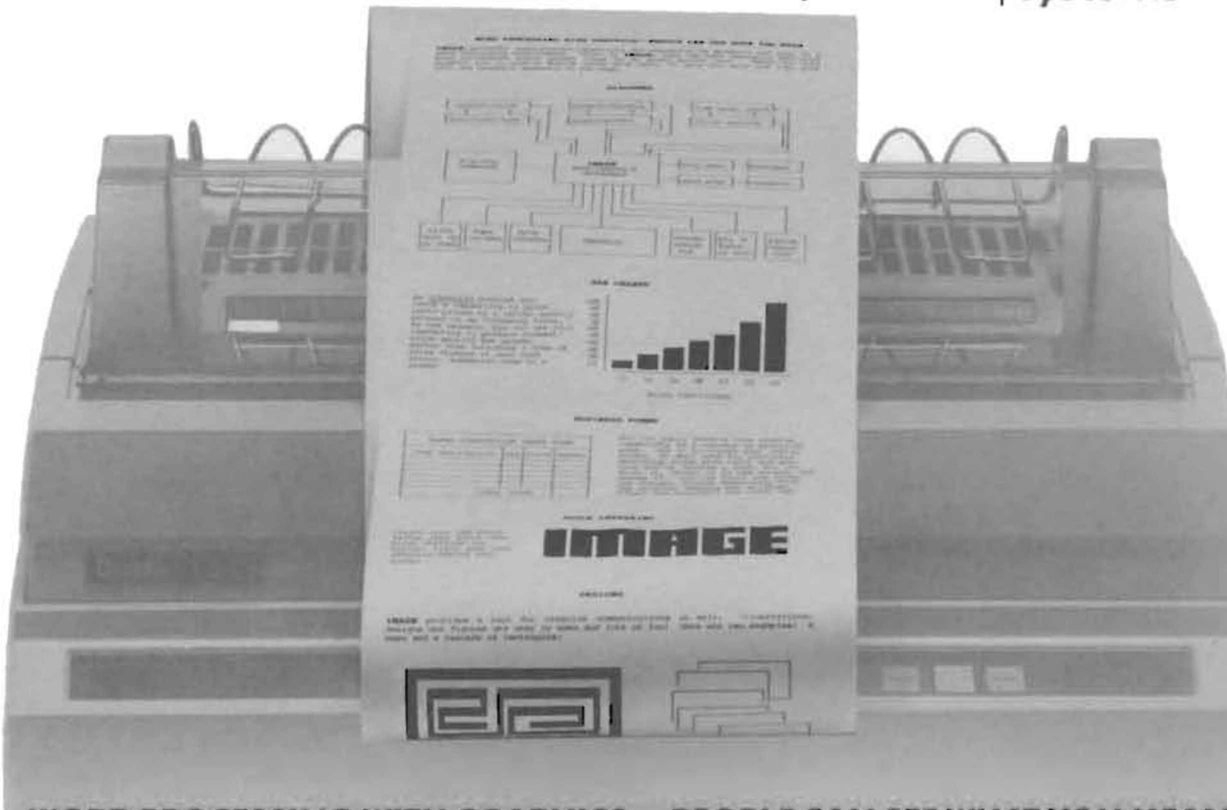
HUG CONFERENCE II
REGISTRATION FORM
INSIDE — Page 7



Official magazine for users of HEATH ZENITH computer equipment.

WORD PROCESSING IS NOW MORE THAN PROCESSING WORDS.

ANNOUNCING **IMAGE**™ FOR ZENITH | data systems



WORD PROCESSING WITH GRAPHICS . . . PEOPLE CAN SEE WHAT YOU MEAN.

IMAGE adds a new dimension to word processing . . . graphics! Now, you can combine text and linear graphics to create *BAR CHARTS, FORMS, ORGANIZATIONAL CHARTS, FLOW CHARTS, BLOCK LETTERS,* and much more. The result? More powerful and more effective communication. People can *SEE* what you mean.

TOP MARKS FROM INFOWORLD

IMAGE won top marks from *INFOWORLD* for its innovation, quality, reliability, and ease of use:

"*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."

IMAGE is a trademark of MicroArt Corporation.

InfoWorld
Software Report Card

Image

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

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

EXTRAORDINARY VALUE . . .
"The modest $295 Price Tag is a Bargain."

IMAGE runs on any Zenith Z/89 or Z/90 computer, on any Heath H/89 or H/90 computer, or on any Z-80-based CP/M system linked to a Z/19 or H/19 terminal.

TO ORDER CALL TOLL FREE:
1-800-MICROART (1-800-642-7627)
in Oregon, call: **1-692-3950**

OR WRITE:

MICRO ART

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

Mastercard, Visa and COD orders accepted.

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

REMark

Issue 38 • March 1983

on the stack

ON THE COVER: Pictured is the IMAGINATOR graphics board for the H/Z-89 & H19, by Cleveland Codonics, which is reviewed on page 19. Superimposed over the board is a sample of its screen output. Slide composition is by E.C. Smith and Associates Inc., St. Joseph, Michigan. Photos by Jim Buszkiewicz.

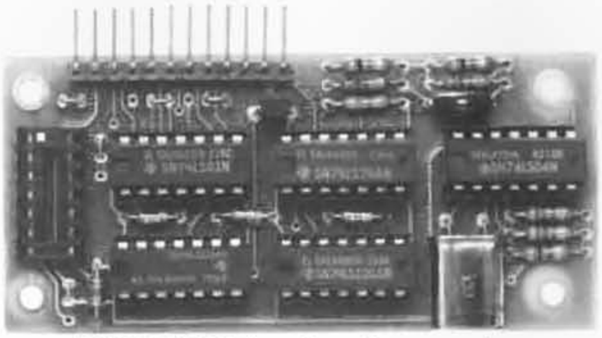
MORE SOLUTIONS

from KRES

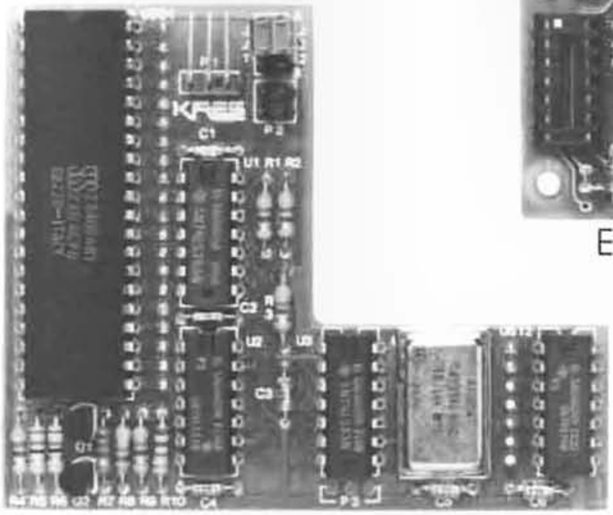
the DUAL SPEED MODULE

a faster solution

Now Available in Two Versions



ESM-240 for the Expansion



DSM-240 for the H/Z 89-90

Why run half speed when you can run full speed? Install one of these KRES Modules in your H/Z 89-90 and cut computation time in half. The supplied HDOS or CP/M software will allow you to operate at regular or double speed (2 or 4 MHZ).

Either version with software **$79.95**

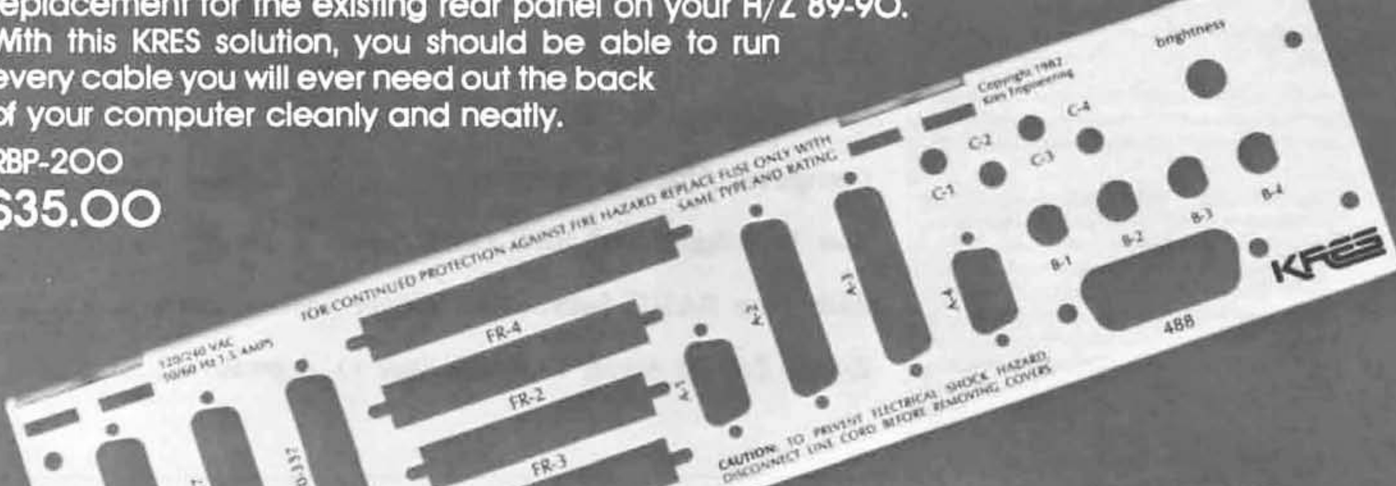
the BACKPLATE

a neater solution

Are you tired of having wires and cables stuffed out the holes of your computer when they should be mounted? The KRES Backplate is an upgrade replacement for the existing rear panel on your H/Z 89-90. With this KRES solution, you should be able to run every cable you will ever need out the back of your computer cleanly and neatly.

RBP-200

$35.00



brightness
Copyright 1982 Kres Engineering
C-4
C-2
C-3
C-1
B-4
B-3
B-2
B-1
FOR CONTINUED PROTECTION AGAINST FIRE HAZARD REPLACE FUSE ONLY WITH SAME TYPE AND RATING
A-2
A-3
A-4
A-1
488
KRES
FR-4
FR-2
FR-3
FR-1
120/240 VAC 50/60 Hz 1.5 AMPS
CAUTION: TO PREVENT ELECTRICAL SHOCK HAZARD DISCONNECT LINE CORD BEFORE REMOVING COVERS
340-347
320-327
330-337
DCE
DCE
DTE

All prices FOB Irvine, CA

California Residents add 6% tax

H/Z 89 is a Registered Trademark of the Heath Company
CP/M is a registered trademark of Digital Research

KRES
ENGINEERING
P.O. Box 17328, Irvine, CA 92713
(714) 559-1047
or (213) 957-6322
Bulletin Board: (714) 559-8579

Don't Miss An Issue!

As you can tell, the quality and quantity of articles appearing in REMark have been steadily increasing over the past year. This improvement is in tune with the steady progress made by our membership in their understanding of micro-computers and computer programming. Many HUGgies have progressed rapidly along this path gaining vast knowledge in hardware, applications and programming while others have progressed at a slower but steady pace and have at least gained an understanding of these matters. Still others are at the threshold of education in the world of micro-computers. With this in mind we have increased the size of REMark to 52 pages. Hopefully we will also be able to have "Super" issues during the year thus presenting you with even more to enjoy.

Because we would like to continue expanding the horizons of REMark we are in need of "Contributing Editors" to provide us with monthly columns devoted to specific subjects. If you have expertise in a given area and feel that you can present this to our membership in a tutorial fashion so they might also gain knowledge, please contact me here at HUG. A form of compensation will be worked out with interested individuals.

The expanded REMark is published monthly. Should you miss a copy please contact the HUG secretary immediately as back issues will become limited. Individual issues of REMark, from #35 back, are no longer available. These issues are however included in volume form: Volume 1 (885-4001) issues 1-13, Volume 2 (885-4002) issues 14-23 and Volume 3 (885-4003) issues 24-35. Selling price of these volumes is $20.00 each. Also, should you move, send us your new address as soon as possible as REMark is NOT forwarded.

In this issue you will find a registration form for the "Second HUG National Conference". Be sure, if you plan on attending, to fill this out and mail it in early. There will be interesting and informative talks, plus special gatherings for you to further your computer knowledge along with the enlarged vendor area, Non- User Hospitality room, plus much more. Don't miss this one.

Walt Gillespie
REMark Editor

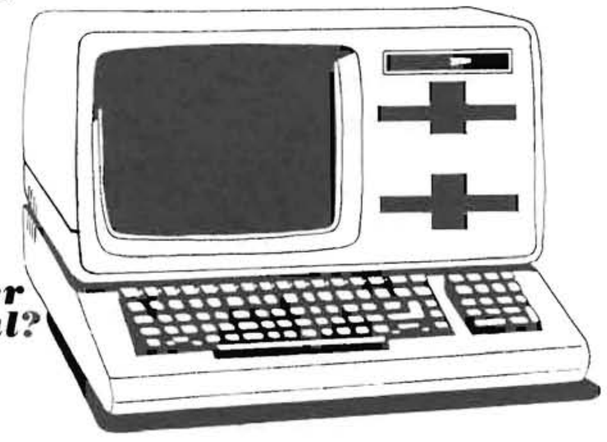
Photos by John L. DePasquale

Can your Z-100 Computer pass these eight tests?

1. Is there COLOR in your life?
If not, you may be lacking either video RAM or a color display. Wizardry handles several top-quality displays as well as having the best price around on video RAM.

2. Does your Z-100 socialize?
Even computers need friends, our ZLYNK modem utility will allow your system to talk with others — and even exchange files! (Wait'll you see our LOW modem prices!!)

3. How's your Z-100's memory?
It takes 192K of CPU RAM to assemble your BIOS. We carry the best (fully-guaranteed) RAM chips for the Z-100 at probably the best price anywhere.



4. Can you read your writing?
Nothing is more sad than a Z-100 whose penmanship is poor. If you can't read your letters, neither can anyone else. Adding a letter or correspondence-quality printer can change all that.

5. Does your computer have growth potential?
We carry several types of compatible S-100 boards to expand your horizons, but NOT deflate your wallet. Ask about the Widget-100.

6. Is your Z-100 disk-hungry?
Our prices on disks will satisfy its appetite no matter how hungry it may be. We carry the disks (the best) that we use on our own systems: Memorex.

7. Can your Z-100 tell time?
If not, we can teach it. We produce a program that uses the Hayes Chronograph to get the time at bootup, or it will use the last date and time if missing.

8. Have you control over your Z-100's character?
Our ALTCHR program allows creation of new characters as well as editing those you already have. Use it to create graphic symbols or whatever you desire.

Call or write for our FREE price list. Software Wizardry, Inc. is both an authorized Zenith Data System dealer (we carry all Heath/Zenith assembled computer products and software) and a computer product developer. Unlike computer salesmen, we can answer your toughest questions about the Heath/Zenith computer series. Come to us for your next computer purchase and we will meet or beat your best price on any product we carry (if humanly possible).

VISA MasterCard

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

SOFTWARE WIZARDRY, INC.

BUGGIN' HUG

Dear HUG,

As a newcomer to the world of personal computing I can easily understand the frustration of the inexperienced user. The need for more new user programs and articles to accomodate the growing number of new HUGGIES out there in the real world is great. We all know that though REMark and all the original 'oldtimers' have matured since issue #1, the current increase in new HUG users justifies channeling some material towards the new user.

When I first joined the HEATH USERS' GROUP my impressions of REMark where disappointing. Here was a personal users' group in which I had just become a member of that was way over my head. These users were all accomplished programmers (and writers of feature articles). To top it off most of the articles and letters were loaded with jargon and information that I just could not understand.

Well after a few issues it became apparent that unless I ordered REMark VOL 1 & 2 and read all those back issues I would never catch up to all those 'old timers'. That by the way turned out to be the second best move of my personal computing career (the first was purchasing the HEATHKIT H89 computer). I found the amount of new user information in REMark Volumes 1 & 2 almost overwhelming. All this from a users group?, I was impressed.

Well that was about six months ago and since then I've been haphazardly experimenting with B.H.BASIC, EDIT, ASM, PATCH, and a miscellany of REMark articles and programs. As a result I've managed to put together a small library of games and user utilities. Even though these programs may not be as sophisticated as some of those available from the HUG library they are useful, cheap, and from a not so experienced programmers standpoint, writing and debugging them was a very educational experience. Most of my ideas for these programs, and indeed some of the programs themselves, came from REMark articles and ideas submitted by users.

In the process of copying, rewriting and combining these ideas to assemble usable programs I noticed that REMark had published at one time (back in issue #7 BASIC IDEAS) a renumbering program for HT-11

Vectored to 8 ☞

Heath Users' Group

NATIONAL HUG CONFERENCE II

Official Conference Registration Form
O'Hare Hyatt Regency Hotel, Chicago, Illinois
August 19, 20 and 21

Name(s) ______________________________

Address ______________________________

Company ______________________________

City ____________________ State ________ Zip ________

Enclosed is $20.00 per individual to attend The Second National HUG Conference to be held the weekend of August 19, 20 and 21 1983. Please send ticket(s) and information regarding hotel reservations.

AMOUNT ENCLOSED__________ NUMBER ATTENDING______

For our information:

Which Heath/Zenith computer do you now operate? ______________

Are you a Non-User-Attendee? ☐ YES ☐ NO

Are you a Heath/Zenith related vendor? ☐ YES ☐ NO

For your information:

Space limitations for the dinner to be held Saturday August 20, 1983, will restrict the number of attendees for that dinner to 1000. Therefore, it is important that you register as soon as possible. Visitor tickets for those of you simply attending and not planning to stay for the dinner and prize drawings will be available at the registration booth for $10.00. Send your registration form or a suitable copy to:

Heath Users' Group
Attention: National HUG Conference Registration
Hilltop Road
Saint Joseph, Michigan 49085

Special Note to Vendors:

Vendor Information Packages will be made available to Heath/Zenith Related Vendors who are planning to exhibit their products while at the conference. Three times more space is available this year for the purpose of showing those products of interest to owners of Heath/Zenith computer products.

Vectored from 7

BASIC. Now here was a program I could really use for all those B.H. BASIC programs with butchered program lines that I had been writing. Therefore with RENUM.BAS from issue #7 as my guide I attempted to rewrite that program for use on BENTON HARBOR BASIC. The result is the enclosed program REN.BAS. Though it is not very fast nor does it incorporate any extraordinary programming techniques I'm sure that some of the new HUGgies out there could use it. If nothing else maybe someone can at least use the basic idea of the program to create some very exciting new program. Nonetheless I humbly present to you REN.BAS in hopes that it will be well received by some of the new users out there who, like myself, have limited software and funds available.

Wally Lizotte
3352 So. 138th E. Ave.
Tulsa, Ok 74134

P.S. Some of us 'new users' could use a KISS article on using DBUG.ABS from HEATH's operating system. How about it?

REN.BAS is a B.H.BASIC program for renumbering B.H.BASIC programs. It will renumber all Goto Gosub and Then statements that refer to a line number. It will also renumber all 'ON x GOTO xx,xx,xxx,xxxx' statements and any combination of Goto's, Gosub's, Then, and On x Goto statements included on the same line. This extensive search feature slows the renumbering time somewhat but offers complete renumbering. This program also offers the option of renumbering any part of the program and will automatically renumber Goto's, Gosub's, etc., in the program lines not renumbered to correspond to the new program lines renumbered. CAUTION: Do not renumber a portion of the program and designate new line numbers already used in another part of the program. Complete instructions are included in the program run.

```
00010 REM 'REN.BAS'  A RENUMBERING PROGRAM IN BENTON HARBOR BASIC
00020 PRINT CHR$(27)+"E"
00030 PRINT TAB(20);"*************** RENUMBER ***************"
00040 PRINT
00050 PRINT "     THIS PROGRAM WILL RENUMBER B.H.HARBOR BASIC PROGRAM LINES IN A"
00060 PRINT "NEW SEQUENCE.  IN ORDER TO USE 'REN.BAS' YOUR BASIC PROGRAM MUST BE"
00070 PRINT "ON DISK.  THIS PROGRAM IS USED ONLY WITH ONE DRIVE THEREFORE YOU"
00080 PRINT "MUST HAVE 'BASIC.ABS','REN.BAS' AND WHATEVER PROGRAM YOU WISH TO RE-"
00090 PRINT "NUMBER ALL ON ONE DISK."
00100 PRINT "     WHEN RUN THE PROGRAM FIRST CLEARS THE SCREEN THEN ASKS YOU TO"
00110 PRINT "INPUT 'OLD FILE NAME:', WHICH IS THE FILE YOU WISH TO RENUMBER."
00120 PRINT "THEN IT WILL ASK FOR 'NEW FILE NAME:', A NEW FILE NAME MUST BE "
00130 PRINT "SPECIFIED.  IF YOU DO NOT WISH A NEW OUTPUT FILE JUST DELETE THE"
00140 PRINT "THE OLD FILE AND RENAME THE NEW FILE AFTERWARDS.  THE REASON YOU"
00150 PRINT "MUST SPECIFY A NEW FILE NAME IS THAT BOTH OLD AND NEW FILES ARE"
00160 PRINT "OPEN AT THE SAME TIME.  (DO NOT INCLUDE '.BAS' EXTENSION, THIS IS"
00170 PRINT "DONE WITHIN THE PROGRAM)."
00180 PRINT "     NEXT THE PROGRAM WILL ASK FOR 'FIRST INPUT LINE:'.  IF YOU ENTER"
00190 PRINT "A-'0', IT WILL ASSUME THE BEGINNING OF THE PROGRAM.  THEN THE PROGRAM"
00200 PRINT "WILL ASK FOR 'LAST INPUT LINE:'.  HERE A '0' WILL CAUSE THE PROGRAM"
00210 PRINT "TO RENUMBER THE ENTIRE PROGRAM.  FOR 'FIRST OUTPUT LINE:' A '0' WILL"
00220 PRINT "CAUSE THE RENUMBERED PROGRAM LIST TO BEGIN AT LINE # '00010'.  FINALLY"
00230 PRINT "WHEN THE PROGRAM ASKS FOR 'INTERVAL SIZE:' A '0' WILL SPACE LINE #'S"
00240 PRINT "10 LINES APART(ie. 00010,00020,00030,00040,...etc.)."
00250 PRINT TAB(23);"*** To Continue Press 'RETURN' ***":PAUSE
00260 DIM L(500),M(500),K$(2)
00270 READ D
00280 DATA 500
00290 READ K$(0),K$(1),K$(2)
00300 DATA "GOTO","THEN","GOSUB"
00310 PRINT CHR$(27)+"E"
00320 PRINT TAB(35);"RESEQUENCE"
00330 PRINT :INPUT "OLD FILE NAME: ";P$
00340 PRINT :INPUT "NEW FILE NAME: ";Q$
00350 PRINT :INPUT "FIRST INPUT LINE: ";L0
00360 PRINT :INPUT "LAST INPUT LINE: ";L3
00370 PRINT :INPUT "FIRST OUTPUT LINE: ";L1
00380 PRINT :INPUT "INTERVAL SIZE: ";I1
00390 PRINT CHR$(27)+"E"
```

```
00400 IF L3=0 THEN L3=65532
00410 P$=P$+".BAS"
00420 Q$=Q$+".BAS"
00430 IF L0=0 THEN L0=1
00440 IF L1=0 THEN L1=10
00450 IF I1=0 THEN I1=10
00460 C=-1
00470 OPEN P$ FOR READ AS FILE #1
00480 R=CIN(1):IF R=0 THEN 600
00490 LINE INPUT #1,;L$:L$=CHR$(R)+L$
00500 L2=L2+1
00510 T=MATCH(L$," ",1)
00520 S$=LEFT$(L$,T)
00530 S=VAL(S$)
00540 IF S<L0 THEN 480
00550 IF S>L3 THEN 480
00560 C=C+1
00570 IF C>D THEN 1280
00580 L(C)=S
00590 GOTO 480
00600 S=INT(L1)
00610 FOR I=0 TO C
00620 M(I)=S
00630 IF S>65530 THEN 1290
00640 S=S+I1
00650 NEXT I
00660 CLOSE #1
00670 OPEN Q$ FOR WRITE AS FILE #2
00680 OPEN P$ FOR READ AS FILE #1
00690 FOR I=1 TO L2
00700 LINE INPUT #1,;L$
00710 C2=MATCH(L$," ",1)-1
00720 C1=1
00730 GOSUB 1140
00740 FOR J=0 TO 2
00750 C1=1
00760 C1=MATCH(L$,K$(J),C1)
00770 IF C1=0 THEN 970
00780 C1=C1+LEN(K$(J))+1
00790 E=MATCH(L$,":",C1)
00800 IF E=0 THEN E=255
00810 E1=E-C1
00820 M$=MID$(L$,C1,E1)
00830 C2=MATCH(M$,",",1)
00840 IF C2>0 THEN 890
00850 E=MATCH(L$,":",C1)
00860 IF E=0 THEN E=255
00870 C2=E-1
00880 GOTO 950
00890 C2=MATCH(L$,",",C1)
00900 C2=C2-1
00910 GOSUB 1140
00920 C2=MATCH(L$,",",C1)
00930 C1=C2+1
00940 GOTO 810
00950 GOSUB 1140
00960 GOTO 760
00970 NEXT J
00980 C2=MATCH(L$," ",1)-1
00990 C1=1
01000 S$=MID$(L$,C1,C2)
01010 S=VAL(S$)
01020 L2$=STR$(S+100000)
01030 L2$=MID$(L2$,3,5)
01040 L1$=MID$(L$,C2+1,255)
01050 L$=L2$+L1$
01060 PRINT #2,;L$
01070 PRINT L$
01080 NEXT I
01090 CLOSE #1
01100 CLOSE #2
01110 PRINT TAB(10);"----->DONE<-----"
01120 FOR I=1 TO 5:PRINT CHR$(7):FOR J=1 TO 10:NEXT J:NEXT I
01130 END
01140 C0=(C2-C1)+1
01150 S$=MID$(L$,C1,C0)
01160 S=VAL(S$)
01170 IF S<L0 THEN RETURN
01180 FOR K=0 TO C
01190 IF L(K)=S THEN 1220
01200 NEXT K
01210 RETURN
01220 L1$=LEFT$(L$,C1-1)
01230 L3$=MID$(L$,C2+1,255)
01240 L2$=STR$(M(K))
01250 L2$=MID$(L2$,2,LEN(L2$)-2)
01260 L$=L1$+L2$+L3$
01270 RETURN
01280 PRINT :PRINT TAB(10);"***TOO MANY LINES***":STOP
01290 PRINT :PRINT TAB(10);"***LINE # TOO BIG***":STOP
01300 END
```

Dear Walt,

I have received several letters about the Christmas graphics program. Most seemed quite pleased but a few people had trouble with the program running it with CP/M. So I tried it, having to add a lot of spaces, and also got screen garbage and beeps. With swift precision (2 hours) and hints from correspondents, I decided that the higher MBASIC version under CP/M was running a little faster, and thus was getting confused between the codes. There is some similarity:

PRINT CHR$(7)
    and you get a beep
PRINT CHR$(27)CHR$(76)
    insert line (IL$ in program)
PRINT CHR$(27)CHR$(77)
    delete line (DL$)

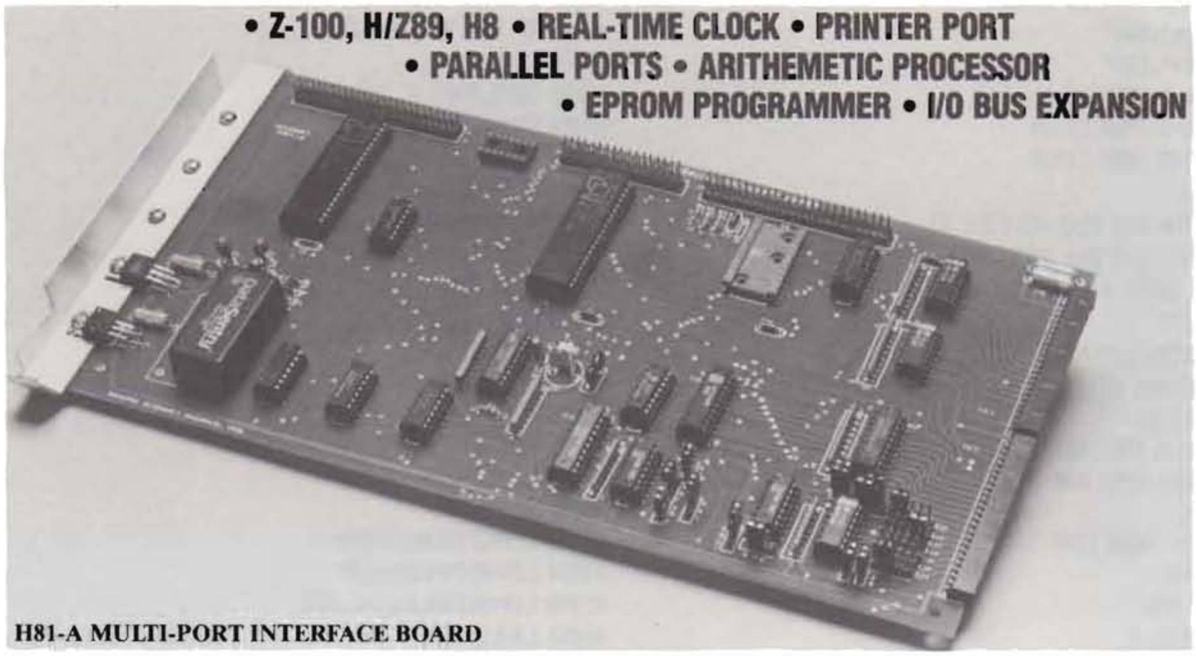
So my solution was to add a line:

```
52 Z$=STRING$(50,CHR$(0)):
                    X$=LEFT$(Z$,10)
```

which gives two strings of nulls, which take BASIC some time to interpret, but nothing gets printed. Then in Line 95 I added X$ after IL$, and in lines 110 and 120 I added Z$ after DL$. The program now runs fine under CP/M. (I did it this way because I have since changed the angel portion, using a lot of insert line commands. They now slowly come down on the screen and flap their wings.) I have also discovered a couple of lines that can be deleted: #155 and #1055.

I wrote this program about 2 years ago. Now, of course, I would do it differently but

Vectored to 28 ☞


• Z-100, H/Z89, H8 • REAL-TIME CLOCK • PRINTER PORT
• PARALLEL PORTS • ARITHEMETIC PROCESSOR
• EPROM PROGRAMMER • I/O BUS EXPANSION

H81-A MULTI-PORT INTERFACE BOARD

EXPAND YOUR HEATH/ZENITH SYSTEM'S UTILITY CAPABILITIES AT MINIMUM COST

MULTI-FUNCTION CAPABILITIES

A single board allows addition of six new features to your system!

- Quartz crystal controlled Real-Time Clock with re-chargeable NI-CAD battery, charging circuit and four user programmable interrupt frequencies.
- A Centronics/Epson compatible parallel printer port.
- Three user-configurable parallel ports.
- AMD9511ADC/12ADC Arithmetic Processor*
- EPROM Programmer support.
- Buffered I/O bus expansion.

AVAILABLE FOR ALL MODELS

- Zenith Z-100 interface, (MICRO WIDGET Z1001-A)
- Heath/Zenith H89/90 interface (MICRO WIDGET H891-A)
- Heath/Zenith H8 interface (MICRO WIDGET H81-A)

SOFTWARE SUPPORT

Software support is available for HDOS®, CP/M®, ZDOS®, and CP/M-85®.

PRICED AND PACKAGED TO SUIT YOUR NEEDS

All MICRO WIDGET utility expansion boards are available as assembled and tested boards, kits and as bare boards. The MICRO WIDGET H81-A can be added to your Heath/Zenith H8 for the incredibly low cost of:

- Assembled full size H81-A board with all options*: $198.95**
- H81-A in kit form with all options* $150.00**
- H81-A bare board only: $95.00
- AMD9511ADC/12ADC Advanced Math Processing Chip: $135.00

Contact MICRO WIDGET WORKS for pricing information for Z1001-A and H891-A boards.

IMMEDIATE DELIVERY

All three models are ready for delivery now! Order by mail or call for **C.O.D., VISA** or **MASTERCARD** orders.

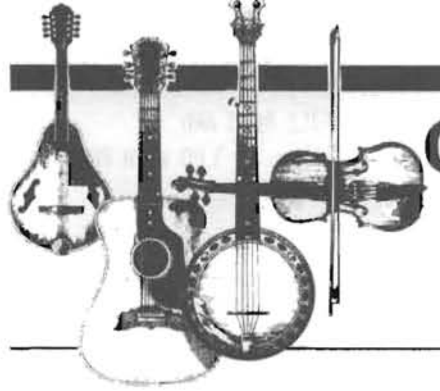
MICRO WIDGET WORKS

P.O. Box 15185
Santa Ana, Calif. 92705
(714) 544-8252

DEALER INQUIRIES INVITED

CP/M and CP/M-85 are registered trademarks of Digital Research, Inc.
HDOS and ZDOS are registered trademarks of Heath/Zenith Corporation.

* AMD9511ADC/12ADC Advanced Math Processing Chip available for additional cost.
** Shipping and Handling charges extra. All California Residents must add 6% sales tax. Charge Card and C.O.D. orders add 4% Surcharge. All prices are subject to change without notice.



Changing Your Tone!

A Music Generation Program

Frank Clark
402 West Ferry
Berrien Spings, MI 49103

Have you ever wanted to generate sounds and music with your computer but were not quite ready to buy extra parts or modify the circuit board. With MUSIC you can get sound from your computer without even opening the cover. All you need is one of the ports already in your computer, some wires and a little speaker.

The MUSIC program generates tones using one of the asynchronous ports of the computer. Binary data streams from pin 2 or 3 (depending on the port) and pin 1 (ground) can be connected to headphones or into an amplifier producing square wave tones. The pitch of these tones is controlled by varying the baud rate of the output.

When you are using the standard port 320 (easily modified to any port on the serial interface) installation is accomplished by merely pushing one bare wire crimped double into pin 1 and another into pin 3. The other ends are attached to a speaker or wrapped around the plug of a pair of headphones. If you want to get fancy you can get special plugs from your local electronics store. On the computer end you will need the appropriate DB25 plug and on the other end a plug to match whatever you wish to plug into. The output can be fed into most amplifier inputs for higher volume levels.

The MUSIC program is run by entering the command:

MUSIC [file]

where 'file' is the name of a file that contains music commands. This file can be written by any standard ASCII editor and contains one letter commands which control the generation of the tones. These are first summarized as follows with each command described in detail afterward.

'c;	comments	T	thirty-second
@n;	time signature	.	dotted note (*1.5)
&n;	key signature	~	triplet (*2/3)
!n;	tone pattern	0-9	octave
^n;	breath value	#	sharp
W	whole	b	flat
H	half	%	natural
Q	quarter	A-G	note
I	eighth	R	rest
S	sixteenth		

The commands can be broken into several distinct groups based on how they work. The first group is those commands which individually set parameters which remain unchanged until a later command changes the parameter. The second group is composed of those commands which may appear together and are summed and multiplied to produce a final value. The third group is composed of those commands which permanently modify the following commands. The fourth group is composed of those commands which perform a temporary modification on the following command. The last group is composed of those commands which actually cause sound to be generated.

Parameter commands

Each of these commands set parameters which remain unchanged until a later command changes them. These are usually only set once and not changed though that is not a restriction.

comments

The comment allows the insertion of a title or whatever comment you desire into the song and instructs the program to ignore it. The comment begins with a single quote and continues for as long as desired, even multiple lines, until the terminating semicolon is encountered. For example:

'title;

could appear in the file indicating the title of the song.

time signature

The time signature is a number indicating how many two millisecond counts per whole note. For example:

@500;

indicates the default speed of 60 quarter notes per minute until a new speed is entered.

key signature

The key signature is a set of seven numbers indicating which notes make up the scale for the key. For example the default key of 'C' the scale contains the notes a, b, c, d, e, f and g:

&0,2,3,5,7,8,10;

until a new scale is entered. A complete list of the values is: A=0, A#-Bb=1, B=2, C=3, C#-Db=4, D=5, D#-Eb=6, E=7, F=8, F#-Gb=9, G=10, G#-Ab=11. As a further example the command for the key of 'G' where 'F' is sharp is '&0,2,3,5,7,9,10;'.

tone pattern

The tone pattern is a number indicating the wave pattern for each cycle. The default pattern of

!7;

causes a simple square wave to be generated. Other values will produce harmonic effects. The maximum useful value is 63 which sets 6 bits of the output tone. With the standard 1 start bit and 0 stop bit there are eight bits per cycle. As a further example the value '!25;' would effectively double the frequency raising the perceived tone one octave and the value '!42;' would raise the tone another octave.

breath value

The breath value indicate what portion of the note is silent. The default value of

^15;

indicates that none of the note is silent. This produces a slurred effect. Other values are 0=1/2, 1=1/4, 2=1/8 ... portion is silent.

Summed commands

The values of each of these commands that appear together are added or multiplied together until the next sounding note.

W	whole	S	sixteenth
H	half	T	thirty-second
Q	quarter	.	dotted note (*3/2)
I	eighth	~	triplet (*2/3)

Permanent commands

The permanent commands select which octave the sounding note is to be played in and remains unchanged until the next command. The octaves are indicated by the numbers 0-9. Note that not all octaves sound equally good. Particularly the lowest octave and the top octaves do not sound very good.

Temporary commands

The temporary commands provide a modification only to the following note.

#	sharp
b	flat
%	natural

Sound commands

The sound commands are the seven letters of the alphabet A-G used to indicate notes plus the 'R' which indicates the sound of silence or a rest.

Conclusion

Any other character other than those indicated above is ignored. It may be useful to enter extra commas or whatever you desire to make the music easier to read. Within limits extra characters will not affect the operation of the program.

Advanced users of the CP/M operating system may be interested to know that special things can be done within the framework of normal though unusual CP/M procedures. Particularly this program is designed to be re-entrant. After the program has been run the input file is permanently included in the memory copy of the .COM file. When the program terminates it prints the total number of memory pages that it uses. The CP/M SAVE command may then be used to create a new file which can be loaded in one step to play music without specifying a separate file. The memory image can be reentered with a zero length .COM file for a quick replay without a disk load.

```
port    equ     0320q
FIRST   ORG     0100h
BEGIN
        LXI     H,0         ;SAVE THE CCP STACK LOCATION
        DAD     SP          ;FOR POSSIBLE DIRECT RETURN
        LXI     SP,NEWSTK+2
        PUSH    H
        IN      PORT+3      ;SAVE OLD PORT VALUE
        STA     PORT3
        MVI     A,080H
        OUT     PORT+3
        IN      PORT
        STA     PORT0
        IN      PORT+1
        STA     PORT1
        LHLD    MAX
        XCHG                ;POSITION VALUE IN CASE WE JUMP
        LDA     05dh
        CPI     ' '
        JZ      READY       ;SKIP FILE READ AND
        LXI     D,065h                        GO WITH PREWRITTEN
        LXI     H,DEF$TYP
        LXI     B,3
        CALL    MBS
        LXI     D,05ch
        CALL    OPENIN
        INR     A
        JZ      NO$FILE
        LXI     D,BUFFER
READ$LP PUSH    D
        LXI     D,05ch
        MVI     C,014h
        CALL    05h
        POP     D
        ORA     A
        JNZ     READY
        LXI     B,128
        LXI     H,080h
        CALL    MBS
        LDA     07h         ;LAST PAGE AVAILABLE
        ADI     0f8h        ;SPARE THE 2K CCP
        CMP     D
        JNC     READ$LP
READY
        XCHG
        SHLD    MAX
        MVI     A,';'       ;END ANY UNFINISHED PARAMETERS
        MOV     M,A
        INX     H
        MVI     A,26        ;CTRL Z FLAG
        MOV     M,A
        LHLD    11
        SHLD    CLOCK
        LXI     H,BUFFER
        PUSH    H
;
; MAIN LUPUS TO PICK UP CHARACTERS
;
LOOP    POP     H
        MOV     A,M
        INX     H
        CPI     26
        JZ      QUIT
        PUSH    H
;
;VARIOUS COMMANDS CHECK
;
        LXI     H,CMD$CHR
        MVI     C,CMD$ADD-CMD$CHR-1
CMD$LP  CMP     M
        JZ      CMD$GOT
        INX     H
        DCR     C
        JP      CMD$LP
        JMP     OCT$CHK
CMD$GOT MVI     B,0
        LXI     H,CMD$ADD
        DAD     B
        DAD     B
        MOV     E,M
        INX     H
        MOV     D,M
        XCHG
        PCHL
CMD$CHR DB      '^@&!R#%b,~''XTSIQHW'
CMD$ADD
        DW      TIME,TIME,TIME,TIME,TIME,TIME,TIME
        DW      COMMENT,TRIP,DOT
        DW      FLT$SET,NAT$SET,SHP$SET
        DW      REST,WAVE,SCALE,SPEED,GAP
;
;COMMAND TABLE ROUTINES
;
SHP$SET LXI     H,MOD$FLG
        INR     M
        JMP     LOOP
FLT$SET LXI     H,MOD$FLG
        DCR     M
        JMP     LOOP
NAT$SET LXI     H,NAT$TBL
        SHLD    NAT$CUR+1
        JMP     LOOP
;
; ADD THE LENGTH OF THE NOTE
;
TIME
        LHLD    COUNT
TIM$LP  DCR     C
        JM      TIM$RDY
        MOV     A,H
        ORA     A
        RAR
        MOV     H,A
        MOV     A,L
        RAR
        MOV     L,A
        JMP     TIM$LP
TIM$RDY
        XCHG
        LHLD    LENGTH
        DAD     D
STORE   SHLD    LENGTH
        SHLD    LENGTHT
        JMP     LOOP
;
; DOTTED NOTE
;
DOT
        LHLD    LENGTH
        MOV     A,H
        ORA     A
        RAR
        MOV     D,A
        MOV     A,L
        RAR
        MOV     E,A
        DAD     D
        JMP     STORE
;
; REST FOR COUNT
;
REST
        LHLD    LENGTHT
        XCHG
        LHLD    CLOCK
        DAD     D
        SHLD    CLOCK
        XCHG
        JMP     WAIT
;
; GAP BETWEEN NOTES
;
GAP
        CALL    GETDEC
        MOV     A,L
        STA     BREATH
        JMP     COMMENT
;
;LENGTH OF A WHOLE NOTE
;
SPEED
        CALL    GETDEC
        SHLD    COUNT
        JMP     COMMENT
;WAVE FORM
;
WAVE    CALL    GETDEC
        MOV     A,L
        STA     NOT$CHR
        JMP     COMMENT
;
; GET A DECIMAL NUMBER TO HL
;
GETDEC
        LXI     D,0
        POP     H
        XTHL
GET$LP  MOV     A,M
        ADI     0d0h
        JM      GET$DN
        CPI     10
        JP      GET$DN
        MVI     B,10
        PUSH    H
        LXI     H,0
GET$SH  DAD     D
        DCR     B
        JNZ     GET$SH
        MOV     C,A
        DAD     B
        XCHG
        POP     H
```

DUMBURNER

16K & 32K EPROM PROGRAMMER

YOU'VE ALREADY GOT THE SMARTS IN YOUR COMPUTER, SO WHY BUY A MICROPROCESSOR BASED PROM PROGRAMMER?

The DumBurner communicates with your H8/H89 computer through an RS-232 serial port, and operates in conjunction with the supplied software.

- **Programs and Verifies 2716, 2732, 2516, and 2532 EPROMs (including high speed and "A" series)**
- **EPROM pulse timing independent of CPU**
- **Transfer Disk files to EPROM**
- **Transfer EPROM to Disk File**

GREAT FOR PROGRAMMING THE WATZMAN MONITOR ROM (HUG 885-1221)

DumBurner Programmer and Software (source included-Basic and/or ASM)	$149
RS-232 Interconnect Cable	$ 9
Bare PC board, Documentation, and Software	$ 25

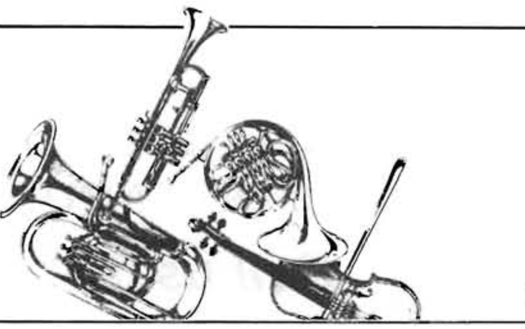
Specify HDOS, CP/M or hardcopy.
Please add $2 postage and handling.
Foreign orders add required postage.

ROSS CUSTOM ELECTRONICS
1307 Darlene Way - Suite A12
Boulder City, Nevada 89005
PHONE (702) 293-7426

```
        INX     H
        JMP     GET$LP
GET$DN  XTHL
        PUSH    H
        XCHG
        RET
;
;SCALE
;
SCALE
        LXI     D,NOT$TBL        ;PLACE FOR NEW SCALE
        MVI     B,7
SCL$LP  POP     H
        PUSH    D
        PUSH    B
        PUSH    H
        CALL    GETDEC
        MOV     A,L
        POP     H
        POP     B
        POP     D
        STAX    D
        INX     D
        MOV     A,M              ;WHAT DID IT STOP ON?
        INX     H
        PUSH    H
        CPI     ';'
        JZ      LOOP             ;BAIL OUT EARLY
        DCR     B
        JNZ     SCL$LP
        JMP     COMMENT
;
;TRIPLETS
;
TRIP
        LHLD    LENGTH
        DAD     H       ;MULTIPLY BY 2
        LXI     D,0-3
        LXI     B,0
TRP$LP
        INX     B
        DAD     D       ;SUBTRACT 3
        JC      TRP$LP
        MOV     H,B
        MOV     L,C
        JMP     STORE
;
; CHECK FOR COMMENTS
;
COMMENT POP     H
SKP$LP  MOV     A,M
        CPI     ';'
        INX     H
        JNZ     SKP$LP
        PUSH    H
        JMP     LOOP
;
;IS IT AN OCTAVE NUMBER?
```

```
OCT$CHK CPI     '0'
        JC      NOT$CHK
        CPI     '9'+1
        JNC     NOT$CHK
        ANI     0FH
        STA     OCTAVE
        JMP     LOOP
;
; IS IT A NOTE?
;
NOT$CHK
        ADI     0bfh
        JNC     LOOP    ;FORGET IT
        CPI     7
        JNC     LOOP    ;FORGET IT
;
; CHANGE LETTER TO NOTE VALUE IN KEY
;
        MVI     D,0
        MOV     E,A
        LDA     OCTAVE
        STA     OCTAVET          ;SET CURRENT OCTAVE
NAT$CUR LXI     H,NOT$TBL        ;TRANSLATE THE NOTE
        DAD     D
        MOV     A,M
;
;MODIFY THE NOTE SHARP OR FLAT
;
NOTE$SET
        LXI     H,MOD$FLG        ;EXTRA SHARP OR FLAT?
        ADD     M
NOT$SML JP      NOT$BIG          ;NOT NEGATIVE
        LXI     H,OCTAVET
        DCR     M
        ADI     12
        JMP     NOT$SML
NOT$BIG CPI     12
        JM      NOT$RDY
        INR     M
        ADI     0f4h
        JMP     NOT$BIG
NOT$RDY MOV     E,A
;
;CHANGE NOTE VALUE TO FREQUENCY
;
        LXI     H,OCT$TBL         ;GET THE DIVISOR
        DAD     D
        DAD     D
        MOV     E,M
        INX     H
        MOV     D,M
        LDA     OCTAVET          ;CURRENT OCTAVE
        MOV     B,A
OCT$SET
        DCR     B
        JM      TON$RDY
        ORA     A       ;CLEAR THE CARRY
        MOV     A,D
        RAR
        MOV     D,A
        MOV     A,E
        RAR
        MOV     E,A
        JMP     OCT$SET
;
;PROGRAM ACE FOR THE BAUD RATE
;
TON$RDY
        MVI     A,80H   ;SET DLAB
        OUT     PORT+3
        MOV     A,E
        OUT     PORT+0
        MOV     A,D
        OUT     PORT+1
        MVI     A,1     ;SET 6 DATA BITS
        OUT     PORT+3
;
; CALCULATE TIME FOR END OF NOTE AND REST
        LHLD    LENGTHT
        XCHG
        LHLD    CLOCK
        DAD     D
        PUSH    H
        SHLD    CLOCK
        LDA     BREATH
        MOV     B,A
BRTH$LP ORA     A       ;CLEAR THE CARRY
        MOV     A,D
        RAR
        MOV     D,A
        MOV     A,E
        RAR
        MOV     E,A
        DCR     B
        JP      BRTH$LP
        MOV     A,E
        CMA
        ADI     1
        MOV     E,A
        MOV     A,D
        CMA
        ACI     0
        MOV     D,A
        DAD     D
        XCHG
;
;OUT CHARACTERS UNTIL TIME
;
LOOP2
        IN      PORT+5
        ANI     020H
        JZ      LOOP1
        LDA     NOT$CHR
        OUT     PORT
LOOP1   CALL    TIM$CHK
        JNZ     LOOP2
; WAIT FOR REST TO END
        POP     D
WAIT    CALL    TIM$CHK
        JNZ     WAIT
;DONE WITH NOTE RESET VALUES
        XRA     A
        STA     MOD$FLG
        LXI     H,NOT$TBL       ;SET BACK TO CURRENT KEY
        SHLD    NAT$CUR+1
        LXI     H,0
        SHLD    LENGTH          ;NEXT COUNT STARTS AT 0
        JMP     LOOP
;ROUTINE TO COMPARE TICCNT
;
TIM$CHK LHLD    0bh
        MOV     A,D
        CMP     H
        RNZ
        MOV     A,E
        CMP     L
        RET
;DATA VALUES
;
CLOCK   DW      0       ;START TIME
LENGTHT DW      500     ;DURATION OF NOTE FINAL VALUE
LENGTH  DW      0       ;DURATION OF NOTE ADDED VALUE
COUNT   DW      500     ;COUNTS PER WHOLE NOTE
PORT0   DB      0
PORT1   DB      0
PORT3   DB      0
BREATH  DB      15      ;SHIFT VALUE FOR BREATH PORTION
MOD$FLG DB      0       ;SHARPS OR FLATS?
OCTAVE  DB      0       ;OCTAVE IN THE RANGE 0 TO 9
OCTAVET DB      0       ;TEMPORARILY MODIFIED OCTAVE NUMBER
NOT$CHR DB      7       ;NOTE CHARACTER
DEF$TYP DB      'SCR'   ;REQUIRED FILE TYPE
NOT$TBL                 ;TABLE OF CURRENT KEY
        DB      0       ;A
        DB      2       ;B
        DB      3       ;C
        DB      5       ;D
        DB      7       ;E
        DB      8       ;F
        DB      10      ;G
NAT$TBL                 ;TABLE OF KEY C AND FOR NATURAL NOTES
        DB      0       ;A
        DB      2       ;B
        DB      3       ;C
        DB      5       ;D
        DB      7       ;E
        DB      8       ;F
        DB      10      ;G
;TABLE OF NOTES FOR A GIVEN OCTAVE
OCT$TBL
        DW      1047    ;A
        DW      988     ;A#
        DW      933     ;B
        DW      881     ;C
        DW      831     ;C#
        DW      784     ;D
        DW      741     ;D#
        DW      699     ;E
        DW      660     ;F
        DW      623     ;F#
        DW      588     ;G
        DW      555     ;G#
MAX     DW      LAST
no$file CALL    CONPRT
        DB      'File does not exist.',0
QUIT    MVI     A,080H
        OUT     PORT+3
        LDA     PORT0
        OUT     PORT
        LDA     PORT1
        OUT     PORT+1
        LDA     PORT3
        STA     PORT+3  ;RESTORE ALL PORTS
        LHLD    MAX
        CALL    OUT$HX4
        LHLD    MAX
        XCHG
        LHLD    NEWSTK
        SPHL
        MOV     A,H
        CMP     D    ;BAD STACK VALUE PROBABLY MEANING DDT?
        JC      DDT$END
        MOV     A,D
        LXI     H,07h       ;POINTER TO THE MEMORY TOP
        SUB     M
        CPI     8       ;DID THE 2K CCP SURVIVE?
        JC      CCP$END
        RET             ;RETURN TO THE CCP
CCP$END RST     0       ;WARM BOOT
DDT$END RST     7       ; RETURN TO DDT
        DS      64
NEWSTK  DS      2
OPENIN  EQU     $
        CALL    CLR$FCB
        PUSH    D
        MVI     C,023h
        CALL    05h
        POP     D    ;FCB CLOBBERED UNDER SOME CIRCUMSTANCES
        LXI     H,021h
        DAD     D
        MOV     C,M
        INX     H
        MOV     B,M
        PUSH    B
        CALL    CLR$FCB
        MVI     C,0fh
        CALL    05h
        POP     H
        RET
CLR$FCB EQU     $
        PUSH    D
        MVI     A,0
        LXI     B,36-12 ;ZERO REST OF FCB
        LXI     H,0ch
        DAD     D
        XCHG
        CALL    MBSX
        POP     D
        RET
MBSX    MOV     H,D
        MOV     L,E
        JMP     110014
MBS     EQU     $
        MOV     A,M
        INX     H
110014  STAX    D
        INX     D
        DCX     B
        MOV     A,B
        ORA     C
        JNZ     MBS
        RET
CON$PRT EQU     $
        POP     H
        MOV     A,M
        INX     H
        PUSH    H
        ORA     A
        RZ
        CALL    CHR$OUT
        JMP     CON$PRT
OUT$HX4 EQU     $
        MOV     A,H
        PUSH    H
        CALL    OUT$HX2
        POP     H
        MOV     A,L
OUT$HX2 EQU     $
        PUSH    PSW
        RLC
        RLC
        RLC
        RLC
        CALL    OUT$HX1
        POP     PSW
OUT$HX1 EQU     $
        ANI     0FH
        ADI     '0'
        CPI     ':'
        JM      CHR$OUT
        ADI     07
CHR$OUT EQU     $
        MOV     E,A
        MVI     C,02h
        JMP     05h
BUFFER  DB      '@40;W'
        DB      '0ABCDEFG1ABCDEFG2ABCDEFG3ABCDEFG4ABCDEFG'
        DB      '5ABCDEFG6ABCDEFG7ABCDEFG8ABCDEFG9ABCDEFG'
LAST    END     BEGIN
```

```
'Yesterday;&0,1,3,5,7,8,10;^5;@1500;!7;
4IG,F,HF,!0;Q3FE!7;,5IA,%B,#C,D,E,F,E,D,HD,!0;4QCB!7;
5IDDCBA4G5BA,HA
Q4G,F,5A,4G,D,F,5A,HA

4IG,F,HF,!0;Q3FE!7;,5IA,%B,#C,D,E,F,E,D,HD,!0;4QCB!7;
5IDDCBA4G5BA,HA
Q4G,F,5A,4G,D,F,5A,HA

A,A,QD,E,F,IE,D,QE,D,C,D,Q.A,I4!0;FQFF!7;

5HA,A,QD,E,F,IE,D,QE,D,C,E,F,!0;C,B,A!7;

4IG,F,HF,!0;Q3FE!7;,5IA,%B,#C,D,E,F,E,D,HD,!0;4QCB!7;
5IDDCBA4G5BA,HA
Q4G,F,5A,4G,D,F,5A,HA

!0;@2000;QF6A5GDF6!7;AH.A
```



*

PRODUCTIVITY TOOLS
for Heath CP/M users

ZSpool-Plus **$59.95**

eliminates your printer bottleneck. It's the only Mainframe-quality queue manager and spooler for Z80-based Heath/Zenith systems. It will keep your printer busy while doing almost any other task. Extends your CP/M to improve type-ahead, reduce warm-boot time, improve keyboard operation, stop a runaway SUBMIT and much more.

ZSD-89 .... **$19.95**

Z80 screen-dump program to print your H/Z89 display upon demand. Assign keys to print the screen, advance one line or a page. Its features are also available as CP/M calls.

Super-19 from Accusonics **$49.95**

The H/Z19 ROM upgrade everybody is raving about. It makes ZSD-89 work like lightning. A must for operation at 19.2 KB. Includes time-of-day clock, more keyboard functions, light-pen support and most VT100 features. Permits use of extended character ROMs too.

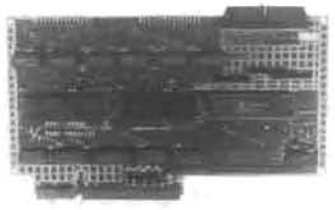
Free shipping anywhere in the USA.
VISA and MasterCard orders call: (800) 343-7775

ATG SYSTEMS, INC.
8 Crescent Street
Wellesley Hills, MA 02181
(617) 431-7870

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.

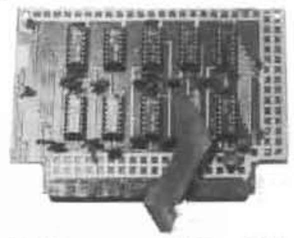
• Fully tested, 90 day warranty, two serial cables and a parallel cable (internal to computer) and software driver.

• We are giving a trade-in allowance for your current Serial interface.

PRICE $275

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 14th month

Now Only $65.00

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

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

Zenith data systems
SERVICE CENTER

DISK DRIVES DISK DRIVES DISK DRIVES DISK DRIVES DISK DRIVES DISK DRIVES DISK DRIVES DISK DRIV

PRICE BREAKTHROUGH

Super Sale on New Disk Drives

Starting at $ **199.95*** complete!!
with Power Supply and Case.
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]
IBM/PC[4] - TEXAS INSTRUMENTS[5] & MOST OTHER COMPUTERS

New!! Special - Special - Call us !!

Now Featuring Siemens Drives.

TOLL FREE ORDERING
1-800-343-8841

GENERAL and TECHNICAL
1-617-872-9090

Diskettes of all sizes (Box of 10) .... starting at	**$15.95**
Dot Matrix Printers................................	**$Call**
Word Processing Printers.......... starting at	**$999.95**
Printer Buffers 8K to 64K........... starting at	**$133.00**
Disk Drive Cases and Power Supplies . starting at	**$49.95**

Dealer inquiries invited.

SOFTWARE SUPPORT, INC.

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

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.

1 ™TANDY CORPORATION
2 ™ZENITH DATA SYSTEMS
3 ™APPLE COMPUTER CORP.
4 ™IBM CORPORATION
5 ™TEXAS INSTRUMENTS

Imagination Gone Wild!

Review of Imaginator Graphics Board

Jim Buszkiewicz
Technical Consultant
Heath Company

Being particularly interested in vector and raster scan graphics, I couldn't pass up the opportunity to evaluate a new product called the IMAGINATOR which is manufactured by Cleveland Codonics Inc. in Cleveland Ohio. This product is a single circuit board that adds moderately high resolution (504h x 247v) dot graphics to a standard H/Z19 or H/Z89/90 without sacrificing any of the original functions or graphics the terminal was capable of.

The kit arrived very well packaged. All loose parts (screws, nuts, etc.) were sealed in plastic bags and all IC's were packed in static resistant tubes. The bare circuit board itself was also sealed in plastic. The assembly and operation manuals came in a standard 1" three ring binder. My first instinct upon opening the box was to page through the manual... "impressive".

The assembly manual was written in a style very much like Heaths'. Each instruction is clearly written and has a check off box when it's completed. There are plenty of pictures and diagrams to supplement these instructions. The only thing I found to be lacking, was a component layout sheet for the entire board. Since each part on the actual circuit board is identified by screening, the manufacturer may have felt that a component layout sheet is not necessary. This may be true in constructing the board, but not true for trouble shooting.

Construction and installation was quite simple and took a total of about three hours. No destructive modifications were necessary to the terminal logic board to install the kit. All that was necessary was the removal of two IC's and the insertion of two preassembled cables. In an H/Z19, the IMAGINATOR fits directly in front of the terminal logic board. In an H/Z89/90, this board fits between the CPU board heat sink and the top of the CRT bezel. Special hardware is required for this mounting task and was included with my kit for evaluation. Normally, when ordering this board, it is necessary that you specify whether it will be installed in an H/Z89/90 or an H/Z19 so the proper hardware is included. The builder is then instructed to make several resistance and voltage measurements on the unit (with IC's removed) before actual power is applied under operating conditions. Before taking the board home and installing it in a H19A, I put it in an older H19 here in my office (instructions are included for doing both). Upon power up, the board appeared to work perfectly according to checkout procedure in the assembly manual.

Programming considerations for the IMAGINATOR are quite flexible. You can use ANY, yes ANY language you like.. even BASIC. All of the programming examples listed in the operations manual are written in Microsoft BASIC. This was done for clarity. The first two demo programs perform some simple graphics operations on the screen and also give the user more confidence that his unit is working. The third demo also accomplishes two tasks. First, it draws the complex picture found on this months' cover. Second, since it takes an hour and a half to complete, it gives the user time to read the rest of the operations manual.

The IMAGINATOR can be programmed in two different modes, ASCII and BINARY. The BINARY mode is the more efficient of the two. Several high level commands are available for plotting. Some of these include: PointAt(x,y), LineTo(x,y), and AreaTo(x,y). In the ASCII mode, plotting a point in the center of the screen would require the following characters be sent to the terminal: P252123. Binary mode, being a bit more complex, would require the following three bytes be transmitted to the terminal: 34h, 5fh, and 3dh. These three hex bytes could of course be converted to ASCII characters if need be. As one can see, it would take less time to accomplish the same task in BINARY mode than it would in ASCII because less characters need be sent to the terminal. Other high level commands are also available which define the type of bit pattern used in drawing a line and what pixel action may be taken when plotting a point. Finally, two instructions exist to affect the display processor hardware itself. One command loads in hex data to a scratchpad ram area (which you must supply). The other command causes execution of the graphics processor to begin elsewhere in memory. Each command form is fully explained in the users manual along with an example, in BASIC, of how that command would be used.

In order to work properly, the terminal must handshake in some manner with the computer to which it's connected. This includes the H/Z89/Z90 since it is essentially a terminal connected to a computer in a single box. The problem occurs when data is transmitted too quickly to the terminal, like from an assembly language program. Once the internal buffer is filled with data, the IMAGINATOR must have a way to tell the computer to stop transmitting temporarily until it is ready to receive more data. This is done in two ways: First, pin four, RTS, of the RS-232 connector is used as a hardware type busy signal, and second, ctrl-S and ctrl-Q are sent as a software type busy signal. The host computer should be capable of recognizing one of these types of handshake signals in order to work at full speed. If it is not, some sort of software timing loop must be inserted into the software so as not to over-run the terminal input buffer. Users of Heath computers should note that a patch is available from Cleveland Codonics for both CP/M and HDOS which will allow proper handshaking to take place.

The IMAGINATOR will have no problem finding its' way into the worlds of business, science, education and that of the individual user. The possibilities for the IMAGINATOR are almost endless. Bar graphs, line graphs, and pie charts are just a few things that can be easily created with simple BASIC programs. Personally I can see myself using the graphics for tactical simulations, maps, and games.

The IMAGINATOR presently sells for $395 in kit form and $445 assembled and tested. The bare board, EPROM, and manuals are also available separately for $215. For an additional $75, an enhanced EPROM can be obtained which is the same as the standard one, plus it directly emulates the Tektronix 4010/4014 series graphics terminal. More information can be obtained directly from the factory by writing to: Cleveland Codonics, Inc., P.O. Box 45259, Cleveland, Ohio 44145. If you're really excited about this new product, you can call them at (216) 327-6405.

A Parallel port for CP/M

Jim Helgesen
6803 Washington Ave. S.
Minneapolis, MN 55435

When Heath came out with the Z-89-11 Multi-function I/O card for the H/Z 89 computer, I rushed out and bought it. It seemed the perfect solution for a budget priced printer. The cost is about the same as an optional serial interface for an Epson printer yet provides a parallel and 2 serial ports. Unfortunately, neither HDOS 2.0 nor CP/M 2.2.03 has parallel I/O capability. The modifications to BIOS turned out to be quite simple and it will support both parallel and serial I/O.

There are 3 basic changes to be made. The 8255 programmable peripheral interface is initialized differently than the serial devices. We will insert a trap in the IN8250 routine to jump to a new initialization routine if the parallel port is specified. The second change is to check handshaking. The final change is the actual output of data.

The Z-89-11 card has a serial port at address 340 octal, another at address 330Q and the parallel port at address 320Q. Any output to address 320Q will be intercepted by our parallel handling routines. This enables us to use both serial and parallel I/O by simply defining which port the printer occupies.

First, format and sysgen 3 disks. Label them ORGBIOS, MODBIOS, and ASMBIOS. PIP the following files from the CP/M distribution disks.

```
ORGBIOS - ED.COM, BIOS.ASM

MODBIOS - PREL.COM, MAKEBIOS.COM

ASMBIOS - MAKEBIOS.SUB, SUBMIT.COM, STAT.COM,
ASM.COM, CONFIGUR.COM
```

With ORGBIOS on drive A: and MODBIOS on drive B: type:

```
ED BIOS.ASM B:
```

On page 114 of the BIOS listing is label LPTOS:. (This is line 3437 in the editor.) Immediately after that line insert:

```
        MVI     A,320Q          ;parallel port address
        CMP     M
        MVI     A,2             ;assume parallel port
        JZ      LPTOSA          ;jump if it is parallel
```

Then on the 6th line, change:

```
        CALL    PINX
```

to:

```
LPTOSA: CALL    PINX
```

That patch takes care of checking if the printer is ready to accept data.

Next locate the label POUT1: (page 117, line 3521). Immediately after that line insert:

```
        MVI     A,320Q
        CMP     M
        JNZ     POUT1A          ;jump if not parallel port
        XRA     A               ;See note 1
        OUT     322Q            ;turn strobe on
        STA     DCLPOS          ;force check of LP status
        INR     A
        OUT     322Q            ;turn strobe off
POUT1A: MOV     A,C
```

That patch will toggle the strobe on and off. The final patch will initialize the 8255. Locate the label IN8250: (page 138, line 4156). Delete the label so it reads:

```
        MOV     B,A
```

Just before that line insert:

```
IN8250: CPI     320Q
        JZ      IN8255
```

Now locate the label CLEN (page 138, line 4199) and insert before that line:

```
IN8255: MVI     A,0AAH          ;set up mode and direction
        OUT     323Q
        MVI     A,1             ;See note 1
        OUT     322Q            ;turn strobe off
        RET
```

Note 1: The patches as shown are for a printer which uses an active low strobe. If your printer uses an active high strobe, change:

```
        MVI     A,1
```

to:

```
        XRA     A
```

in the IN8255 routine and replace the routine at POUT1 with:

```
        MVI     A,320Q
        CMP     M
        JNZ     POUT1A
        MVI     A,1
        OUT     322Q
        DCR     A
        STA     DCLPOS
        OUT     322Q
POUT1A: MOV     A,C
```

That completes the changes. Exit the editor using the 'E' command. BIOS.ASM on the ORGBIOS disk has been renamed BIOS.BAK and MODBIOS now has BIOS.ASM on it. Remove ORGBIOS and put the ASMBIOS disk in drive A:. Do a warm boot, change the designated drive back to A: and type:

```
SUBMIT MAKEBIOS A: B:
```

Respond with the appropriate letter when prompted for system configuration. The MAKEBIOS program takes quite a long time to run. If there are any errors you will have to go back to the editor and correct them. When the message 'MAKEBIOS FUNCTION COMPLETE' is displayed, type:

```
STAT BIOS.SYS
```

The system will respond:

```
Recs   Bytes   Ext Acc
  36     5K     1 R/O  A:(BIOS.SYS)
Bytes Remaining on A: 54K
```

We're almost there. Type:

```
CONFIGUR
```

Type 'N' in response to Standard system. Select option A and on the sub menu select B. Set the TTY: port address to 340Q. Select option C and hit return for baud rate and set the LST: port address to 320Q. Set the Printer Ready Signal Polarity to the correct level. Then type 'Y' to get back to the main menu. Now type 'D' and set the 'Run automatic command line on cold boot' to false. Type 'Y' to return to main menu and 'Y' to have it saved on disk.

You should be all set. Reboot the system and type a control P then type 'DIR'. The directory should now be printed. If it doesn't, try turning the power to the computer off and on and rebooting. If it still doesn't work and the system won't accept anything beyond the first letter you type, then the Ready signal polarity is probably wrong. Change that with the configur program.

The BIOS.SYS can be transferred to other disks by typing:

```
PIP B:=A:BIOS.SYS[R]
```

If you are having difficulties using the editor and would like more detailed instructions for performing the modifications, send a self addressed stamped envelope to:

Instrumentation and Control Electronics
6803 Washington Ave. S.
Edina, MN 55435

If you want ICE to modify your BIOS. Send a disk (5 1/4" only) with BIOS.ASM on it along with $5.00. We also have a device driver for a parallel port that runs under HDOS 2.0.

4 MHz Update

Pat Swayne
Software Engineer

In REMark issue #34 (Nov. 82), page 25, I described a modification for the H89 to upgrade it to 4 MHz operation. In this article, I will present corrections to that article, and hints to help you achieve 4MHz operation if you are having trouble.

Corrections

The following corrections should be made to the original article. On page 25, in the last column near the bottom of the page, change "U1 pin 7 to U1 pin 8" to "U1 pin 7 to U2 pin 8". On page 26, the last few lines of the RAM test (beside Figure 2) should be:

```
040111 000 362
040112 000 303
040113 000 122
040114 000 7        (hit RETURN)
H:
```

On page 27, near the top of the first column, change "(replaces U550)" to "(replaces U516)". Also on page 27, the line that reads

```
ANI     -1-4            RESET 4 MHZ BIT (SET 2 MHZ BIT)
```

should read

```
ANI     -1-4 AND 0FFH   ;RESET 4 MHZ BIT (SET 2 MHZ BIT)
```

and every comment line should have a semicolon (;) before it, as shown above.

The Case of the Missing Resistor

The Z80 requires at its clock input a signal whose voltage level exceeds what is normally produced by TTL devices (most of the IC's in the H89 are TTL devices). To compensate for this, the designers used a 330 ohm "pull up" resistor at the Z80's clock input. Unfortunately, because of the way the board is laid out, my modification takes that resistor out of the circuit. It is only because most IC's are "forgiving" and can work outside their specifications that the circuit worked for me and many others who built it. If yours does not work, it may be because of the missing pull up resistor. To correct the problem, connect a 330 ohm resistor from pin 6 to pin 14 of the 74S132 in the modification circuit. This can be done by soldering the resistor to the back of the PC board, at the unused socket where the modification is installed.

4 MHz with MTR88

If the ROM monitor in your H89 is MTR88, you may have noticed that 4 MHz operation works when you run the RAM test, but there is no noticeable change in speed when you switch speeds in HDOS. That is because MTR88 does not make use of the Control Byte at 40066A when it writes to port 362Q. (If you are not sure whether you have MTR88 or not, reset your computer to get the H: prompt and type L. If the word "Load" appears, you have MTR88.) There are two solutions to the problem. The first is to upgrade to a newer monitor. The original 4 MHz article gives the part numbers and instructions for upgrading to MTR90 (see corrections above). The disadvantage to this fix is that you will no longer be able to run Cassette software. The second fix is to switch speeds only in BASIC using OUT 242,4. This turns on 4 MHz but turns off clock interrupts, so that the PAUSE command in BH BASIC no longer works, and your disk motors will not stop if they are spinning at the time you issue the OUT. Be sure to OUT 242,2 before you return to HDOS.

Everything Works Except the Printer

If your 4 MHz modification seems to work fine, but your printer will not work while you are running at 4 MHz, remove your serial interface card (H88-3), and see if it has an IC labeled U610. If it does, remove the IC and connect a jumper wire from pin 2 to pin 5 on the U610 socket. Also, connect a jumper from U601 pin 5 to U610 pin 8. You can make the connections on the back of the PC board. If your printer still does not work, or your interface does not have U610, try swapping the 8250 (443-952) in your printer channel (usually the 340/347 channel, so it would be U602) with the 8250 in another channel on the board. If that fixes it, you will need to get another 8250 to replace the one that will not work at 4 MHz (unless you can spare a channel).

The H89A

If you have an H89A and would like to try the 4 MHz modification, there are additional modifications needed, and you have to decide where to put the modification circuit (there is no unused IC position). It should be possible to just mount the board at any convenient location near U501 and run wires from it to the various connecting points on the board, passing them through feed-through holes in the board if necessary. I am hoping that a user who comes up with a clever way of doing the mod will write it up and send it to Walt, our REMark Editor.

The additional modification needed for the H89A is to remove the IC at U563 and connect a jumper wire from pin 2 to pin 5 on the U563 socket. Also connect a jumper from U564 pin 11 to U563 pin

8. You can make the connections on the back of the PC board.

Which Pin is Which

Early schematics of the H89A showed a new connection (not on the H89) running from U552 pin 9 to P509 pin 17. Pin 6 on U552 was still unused, so that is why I used it in my 4 MHz mod. However, later revisions of the H89A schematic show pin 6 going to P509, and pin 9 unused, and I am told that is the way it is. There is currently no board that Heath sells that uses that line, so the mod can still be built using pin 6 of U552, but you may want to use pin 9 instead, to be prepared for any future use of the line from pin 6 (possibly a bank switchable memory board). If you use pin 9, the bit to switch to 4 MHz is 00001000B instead of 00000100B, so you will have to change the software mods in the REMark #34 article accordingly. On page 27, change the POKE and OUT values from 38 to 42. In the next column, the HDOS POKE and OUT values change from 6 to 10. Change MVI A,6 to MVI A,10. Change the number 4 in the CP/M ASM example to 8 at the ORI and the ANI instructions. On page 28, change MVI A,6 to MVI A,10, both in the first column and in the second column, near the bottom. On page 29, second column near the bottom, change ORI 4 to ORI 8.

If All Else Fails, Take 3

If you have tried everything, and cannot get your H89 to run at 4 MHz, you may want to try 3 MHz. It is almost certain that any H89 will run at 3 MHz, and, while that isn't quite as good as 4, it is still better than 2 MHz operation. Modification to 3 MHz operation is very easy once you have installed the 4 MHz mod, because divide-by-three circuits are really divide-by-four circuits with feedback. In this case, the feedback goes through the XOR gate at U3 in my schematic. Just remove the IC at U3 (74LS86), and connect a jumper from pin 1 to pin 3 on the socket. You can make the connection by inserting the ends of a 1-inch (2.5 cm) piece of number 24 single strand wire into the socket holes.

You will have to change the disk timing constants as presented in the original article in order to run at 3 MHz. At the label BOOT4 on page 28, the the constants should be changed to

```
BOOT4   DB      30,10,24,9,20,15,250,5,7,32,32,160
```

Change H17SDL*2 to H17SDL*3/2 and 2048/17 to 2048/24. In the H47 modifications on page 29, there should be only 2 lines of MVI A,0 at the labels WDN1 and WND1.

The CP/M constants (page 29, bottom of first column) should be changed to

```
WHDA    EQU     33          WRITB   EQU     10
WHNA    EQU     33          WRITC   EQU     32
WSCA    EQU     120         READA   EQU     75
WRITA   EQU     32
```

In the second column on page 29, you should have only 2 lines of MVI A,0 at W4D1 and 2 lines of NOP at WBS371. The two lines of MVI A,20 should be changed to MVI A,15.

The last 5 lines of the MOVCPM mod should be changed as follows.

```
607  30  48
613  50  A0
75E  14  1E
76B  14  1E
876  0A  14
```

This completes the changes to run at 3 MHz instead of 4. Run the same tests given in the 4 MHz article to ensure proper operation.

NEWLINE SOFTWARE

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

SOFTWARE FOR HEATH/ZENITH COMPUTERS

TAXCALC-82 — Helps you prepare your 1982 Federal Income Tax. Prepares forms 1040, 2210, schedules A, B, C, W. Questionnaires help you organize the necessary data to complete the tax forms. Provides printed report for transfer to tax forms.
For CP/M 2.x, MBASIC, PRINTER REQUIRED **$59.95**

CHKPRO — Screen oriented checkbook, savings account, and credit card manager. Easy entry of data. On-screen recalculation. Categorized accounts. Printout program included.
For CP/M 2.x, Z-DOS **$29.95**

Domestic Management System — Menu driven program for managing your family finances, budget, bank accounts. Keep track of expenditures. Easily customized menus. Simplified input.
For CP/M 2.x **$39.95**

Text Processor 4.1 — Full function, screen oriented text editor. Very easy to use. No computer expertise required. Great for reports, letters, programs, memos, proposals, documents, etc.
For HDOS, CP/M 2.x, Z-DOS, CP/M-85 ... each only **$59.95**
Special priceorder both Z-DOS & CP/M-85 formats
.... only **$99.95**

HDOS 2.0 for Z-100 — Run HDOS applications on your Z-100. Twice the speed. Write for details and prices.

FREE CATALOG — Send for yours today!

Chronograph Revisited

Dear Walt:

Thank you for publishing my article about using the Hayes Chronograph, I hope it helps other HUGgies. Since I wrote the article, I have added two more functions to the BASIC program for accessing the Chronograph. These functions are (1) set the Chronograph port to 300 baud, and (2) set the Chronograph port to 1200 baud. Here are the changes:

Add the lines:

```
191 PRINT"17. Set the baud rate
        to 300 (INTERPRETER speed)"
192 PRINT"18. Set the baud rate
        to 1200"
```

To the end of line 215 add ",522,524" so that it now reads:

```
215 ON F% GOTO 225,235...500,510,520,522,524
```

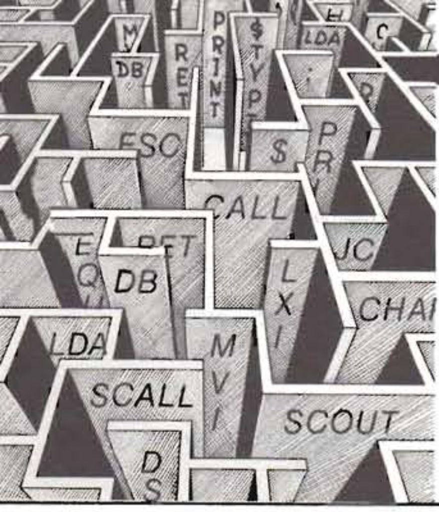
To the end of line 520 and ":GOTO 530" so that it now reads:

```
520 OST$="ATCL":GOTO 530
```

Finally add the lines:

```
521 ' FUNCTION 17
522 OUT &0323,131:OUT &0320,128:
        OUT &0321,1:OUT &0323,3:OK=0:GOTO 540
523 ' FUNCTION 18
524 OUT &0323,131:OUT &0320,96:OUT &0321,0:
        OUT &0323,3:OK=0:GOTO 540
```

Glen E. Hassebrock, Jr.
2195 E. Decatur St.
Decatur, IL 62521



ASM For The Novice

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

My first exposure to assembly language was a fleeting one. I built my H8 about five years ago, and in the process of checking out my H19 terminal a transistor blew out. There I was, a new computer and a terminal disabled by a $3.00 component on backorder from not only Heath.....but from the OEM supplier as well. After a few days of looking at my new H8 I decided to see what I could make it do without a terminal. The little routine in the manual...the one that made the LEDs say "your H8 is up and running".....provided the starting point. I set about trying to figure out how in the world 076,002,062,010,040,etc could possibly make the LEDs behave like that. I discovered the key in the HASL-8 section of my H8 manual...all of those mysterious three-letter codes that corresponded to instructions like "jump" and "call" and "return" and "increment register A". I compared them to the code I was entering into the front panel and EUREKA!....I saw how those little numbers tied together. I spent several weeks developing a crude program to make the front panel act like a clock, and then my errant transistor arrived. I finished my H19, loaded BASIC, and promptly forgot everything about octal code and assembly language. It took the passage of several years and a new H89 with its convenient disk drive to rekindle the interest....and I've been having a great time all over again!

I consider myself a complete novice at assembly language, but I'd like to share what I've learned in novice terms with other novices. I've not progressed into manipulation of the individual "bits", or into precision mathematics, but someday I'll feel the need. Right now I'm enjoying the "thrill of victory" from a few successfully running programs....and it's so easy to do I'd like to encourage others to try.

First, you need to understand that assembly language really isn't a language in the same sense as BASIC. It's just a way to organize and create "object code"....which is the string of numbers that give instructions to the microprocessor. With the text editor EDIT you can create lists of instructions (very similar to BASIC instructions) that define how the program is to run....called "source code". Then, the assembly language is used to convert this source code (.ASM file) to object code (.ABS file) which can be directly executed by the computer.

Object code talks directly to the microprocessor, so we must be aware of where the instructions and data are going. The little "pigeonholes" we can stuff numbers into are far more structured than with BASIC. We need to reserve space for our variables in memory and take care that our mathematical operations don't trample over each other. That sounds more difficult than it really is, since there are some tools already in the microprocessor to help. First, the accumulator (Register A) is used as a scratchpad and to perform arithmetic operations. It seldom holds a number more than a few steps. The H and L registers are generally used to hold the high (H) and low (L) bytes of an address (such as 040 100) for some "indirect addressing" commands that use H and L to point to addresses. Registers D and E can be used as a pair (like H and L) or individually for temporarily storing a number during computation. Memory locations are given names for storage of constants, variables and for locating the program steps. This will become clearer as we develop our program.

It has helped me to think of the assembly language instructions in terms of their counterparts in BASIC. Let's look at a few examples and then build a simple program to demonstrate.

JMP in assembly language is just like GOTO in BASIC. JMP LINE2 will transfer program execution to the instruction at LINE2 just like GOTO 6000 will transfer execution to line 6000.

CALL is just like GOSUB. As above, CALL SUBR will transfer control to the subroutine located at SUBR just like GOSUB 7000 will transfer control to line 7000. The command RET is identical to BASIC's RETURN.

Constants and variables are handled differently in assembly language. Usually a constant is assigned its value with the EQU instruction. The instruction WIDTH EQU 027 will assign the value 27 to the constant named "WIDTH". Notice how nice it is to be able to use more descriptive terms than allowed by BH BASIC (such as WIDTH instead of W, or TIME2 rather than T2).

Variables are usually stored in named memory locations so that we can retrieve them at will, change them, and then replace them. One byte of memory can be reserved with the DS (define space) command. The command SPEED DS 1 reserves one byte of memory and calls it "SPEED". The value of SPEED can be changed at any time by putting the new value in Register A (our trusty little scratchpad) and then transferring it with the STA (store A) command as follows:

```
MVI     A,15
STA     SPEED
```

MVI A,15 means move 15 into A immediately. Then, STA SPEED puts 15 into the memory location called SPEED. We can retrieve that number by loading it into Register A with the LDA (load A) command as follows:

```
LDA     SPEED
```

Two additional commands will allow us to begin some mathematical manipulations. ADI means "add immediately to A" and SUI means "subtract immediately from A". We can add 5 to SPEED as follows:

```
LDA     SPEED   (GET SPEED AND PUT IT IN A)
ADI     5       (ADD 5 TO THE VALUE IN A)
STA     SPEED   (STORE THE NEW VALUE OF A IN SPEED)
```

Now that's a little more complicated than the S=S+5 statement we'd use in BASIC, but it executes much faster, and that can make a real difference in a game or graphics program.

There are two more crucial math commands. ADD and SUB allow you to add and subtract other registers from the A register. This allows us to add two variables, as follows:

```
    LDA     DIST1   (GET DIST1 AND PUT IT IN A)
    ADD     B       (ADD B REGISTER TO A)
    STA     DIST2   (PUT SUM OF DIST1+(B) IN DIST2)
```

All that's left now is to find a way for our program to communicate with the keyboard and CRT screen. Fortunately some subroutines have been included in HDOS that we can use for that purpose. ".SCOUT" (system console out) will send the character in Register A to the CRT screen. ".SCIN" (system console in) will get a character from the keyboard and put it in Register A. The ".CLRCO" (clear console) routine will clear the keyboard buffer. We can call these with the SCALL command (system call), which is just like CALL except for its special application to system subroutines.

Last, although it's not an assembly language instruction, there is a very useful subroutine called $TYPTX contained in the H-17 ROM that you can call on to write sentences on the console. This subroutine can be utilized as follows:

```
    CALL    $TYPTX
    DB      'This line was typed by $TYPTX',200Q
```

where the 200Q acts as an end-of-line signal.

That's enough commands for now. Let's write a program to perform the following simple demonstration routine:

1. Clear the screen
2. Input a letter from the keyboard
3. Print the corresponding graphic character on the H19 screen
4. Exit to HDOS

First we should define a few constants for use in our subroutines:

```
    ESC     EQU     027     (DEFINE THE ESCAPE CHARACTER)
    .EXIT   EQU     000Q    (POINTER TO .EXIT SCALL)
    .SCOUT  EQU     002Q    (POINTER TO .SCOUT SCALL)
    .SCIN   EQU     001Q    (POINTER TO .SCIN SCALL)
    .CLRCO  EQU     007Q    (POINTER TO .CLRCO SCALL)
    $TYPTX  EQU     031136A (LOCATION OF $TYPTX ROUTINE)
```

Now, let's write a subroutine to clear the screen:

```
    CLEAR   MVI     A,ESC   (PUT ESC IN A)
    SCALL   .SCOUT  (SEND IT)
    MVI     A,'E'   (PUT 'E' IN A)
    SCALL   .SCOUT  (SEND IT TO COMPLETE ESC E)
    RET
```

To input the character:

```
    INPUT   CALL    $TYPTX
            DB      'Input a character',200Q
    LOOP1   SCALL   .SCIN   (GET A CHARACTER)
    JC      LOOP1   (GO BACK IF NO CHARACTER YET)
    STA     CHAR    (STORE DECIMAL INPUT IN MEMORY)
    RET
    CHAR    DS      1
```

To print the graphic character:

```
PRINT CALL    $TYPTX
      DB      'The corresponding graphic character is',200Q
      MVI     A,ESC
      SCALL   .SCOUT
      MVI     A,'F'   (ESC F ENTERS GRAPHIC MODE)
      SCALL   .SCOUT
      LDA     CHAR
      SCALL   .SCOUT  (PRINT THE CHARACTER)
      MVI     A,ESC
      SCALL   .SCOUT
      MVI     A,'G'   (ESC G EXITS THE GRAPHIC MODE)
      SCALL   .SCOUT
      RET
```

To exit to HDOS we'll use the .CLRCO SCALL to clean our escape codes out of the type-ahead buffer and then .EXIT:

```
DONE    SCALL   .CLRCO  (CLEAR THE BUFFER)
        MVI     A,0     (ZERO IN A MEANS NORMAL EXIT)
        SCALL   .EXIT
```

Now we need a little program at the front to tie all of our subroutines together:

```
START   CALL    CLEAR
        CALL    INPUT
        CALL    PRINT
        CALL    DONE
```

A final bit of housekeeping requires that we tell the assembly language where to put our program in memory with the ORG (origin) statement. The H89 puts octal code (.ABS) programs at a location called USERFWA (user free working area). Also, we need to give it a title. The final program looks as follows when complete:

```
        TITLE   'Our first ASM program'
USERFWA EQU     042200A LOCATION OF USERFWA
        ORG     USERFWA
*
*
ESC     EQU     027     DEFINE THE ESCAPE CHARACTER
.EXIT   EQU     000Q    POINTER TO THE .EXIT SCALL
.SCOUT  EQU     002Q    POINTER TO THE .SCOUT SCALL
.SCIN   EQU     001Q    POINTER TO THE .SCIN SCALL
.CLRCO  EQU     007Q    POINTER TO THE .CLRCO SCALL
$TYPTX  EQU     031136A LOCATION OF $TYPTX ROUTINE
*
*
START   CALL    CLEAR
        CALL    INPUT
        CALL    PRINT
        CALL    DONE
*
*
CLEAR   MVI     A,ESC   PUT ESCAPE IN REGISTER A
        SCALL   .SCOUT  SEND IT
        MVI     A,'E'   PUT LETTER E IN REGISTER A
        SCALL   .SCOUT  SENT IT FOR ESCAPE E
        RET             RETURN TO MAIN PROGRAM
*
*
INPUT   CALL    $TYPTX  PRINT THE NEXT SENTENCE
        DB      'Input a character',200Q
```

```
LOOP1  SCALL  .SCIN   GET THE NEXT CHARACTER
       JC     LOOP1   GO BACK IF NO CHARACTER YET
       STA    CHAR    STORE THE CHARACTER IN MEMORY
       RET            RETURN TO THE MAIN PROGRAM
CHAR   DS     1       RESERVE A SPACE FOR THE CHARACTER
*
*
PRINT  CALL   $TYPTX  PRINT THE NEXT SENTENCE
       DB     'The corresponding graphic character is',200Q
       MVI    A,ESC   PUT ESCAPE IN REGISTER A
       SCALL  .SCOUT  SEND IT
       MVI    A,'F'   PUT LETTER 'F' IN REGISTER A
       SCALL  .SCOUT  SEND IT FOR ESCAPE F
       LDA    CHAR    PUT THE CHARACTER IN REGISTER A
       SCALL  .SCOUT  SEND IT
       MVI    A,ESC   PUT ESCAPE IN REGISTER A
       SCALL  .SCOUT  SEND IT
       MVI    A,'G'   PUT THE LETTER 'G' IN REGISTER A
       SCALL  .SCOUT  SEND IT FOR ESCAPE G
       RET            RETURN TO THE MAIN PROGRAM
*
*
DONE   SCALL  .CLRCO  CLEAR BUFFER OF UNWANTED CHARACTERS
       MVI    A,0     ZERO IN REGISTER A MEANS NORMAL EXIT
       SCALL  .EXIT   RETURN TO HDOS
       END    START   TELLS ASM WHERE TO START AND STOP
```

Now we need a disk containing EDIT.ABS and ASM.ABS. Use your text editor EDIT to create a file identical to the one above. Be sure to "left-index" all of the columns. You can use the tab feature to make the columns ("fields"). When done, NEWOUT the file to the disk with the name PROGRAM.ASM. Call up the assembler with the command >ASM, and you will get an asterisk as a prompt. Type PROGRAM=PROGRAM and hit return. The assembler will develop an object code program from your source code and put it on the disk with the name PROGRAM.ABS. Now you can run your program with the command >PROGRAM.

I've only begun to scratch the surface of possibilities with Assembly Language. I'll leave the rest up to you. You'll want to explore indirect addressing with MOV and LXI. Soon the conditional CALLs and JMPs as well as IF and ENDIF will be directing the order of your program execution. Register commands such as LHLD, SHLD, XCHG and DAD will open the world of sixteen- bit manipulation. POP and PUSH (fascinating names aren't they?) will let you use the stack for storage of data. Soon you'll be using all of the commands.....so maybe someone can explain what to do with RAR, XRA and RAL?!

Join with other HUGgies at the 1983 NATIONAL HUG CONFERENCE II

Generic Software's FINANCIAL $ERIE$ works hard for your $$...

DEPRECIATION-MA$TER... New!...$95.00
Database system for assets and depreciation schedules. Includes all IRS Pub. 534 methods. Standard ACRS schedules supplied on disk. Generates Tax Form 4562 output! **(MBASIC)**

LOAN-MA$TER.................$29.95
Comprehensive loan analysis program. Provides amortization schedules and loan summaries. Can handle "balloon" payment schedules. Very useful for evaluating auto and home loans.

INVESTMENT-MA$TER...........$29.95
A program to handle lump sum and annuity investments. Extremely useful in obtaining answers about periodic saving plans, mutual funds, and IRA's.

ENERGY-MI$ER................$24.95
A spreadsheet program for maintaining and displaying energy usage and cost data. Provides for hardcopy reports and plots for trend analysis.

Relax after hours with Generic EDUCATION/ENTERTAINMENT programs...

WORLD-MAP......(MBASIC)......$19.95
An introduction to world geography which provides a graphic picture of the earth's surface on the H19 screen!

GAME-PAC-1......(MBASIC)......$19.95
Includes GUNS-OF-THE-ALPS, a projectile shooting game, and NUMBER-ROTATION, a challenging number memory game.

Generic Software

QUALITY SOFTWARE PRODUCTS

P.O. BOX 790 MARQUETTE, MI 49855
Phone: (906) 475-7151 10 am - 5 pm EST

Call or write for more information or our **FREE** Spring 1983 catalog.

All products are available in HDOS and CP/M format, except where noted. Payment may be via check or money order. Add $2 for shipping. Michigan residents please add 4% for sales tax. PAYMENT IN U.S. FUNDS PLEASE.

Keep track of information easily with Generic's DATABASE Programs...

SEQ-BASE.......(MBASIC).......$29.95
A database editor for any type of user data. Powerful features, yet simple to use. Extremely fast, since all record searching, updating, and sorting is done in memory.

MAIL-IT II.... NEW FEATURES! $39.95
A full-featured mail list database editor which uses random-access disk I/O and index files for fast and efficient data storage and retrieval. Provides for name/address reports as well as user-defined labels. **(MBASIC)**

CARD-IT.......(MBASIC).......$24.95
A credit card management program which maintains a data-base of your "plastic money". Tells you what your current credit card balance is, and how close you are to the limit!

Generic's DATA ENTRY Utilities make getting information into your computer easier!

SFDVD.......(HDOS only!).......$19.95
Screen form driver for the H19. Makes data entry and editing easy and minimizes input errors! Supports numeric only, alpha only, upper case, and no-echo data input fields.

FORM-IT (MBASIC)..............$19.75
FORM-REL (FORTRAN)...........$19.75
Data entry kits for your H19. Each kit includes sophisticated utilities for screen form editing and generation. A set of library routines are provided which allow you to easily utilize screen forms within your program!

...And Save Time! with these UTILITIES from Generic...

CATALOG-MASTER.....NEW!.....$29.95
The most extensive disk cataloging program available for your H/Z computer! Provides selection criteria and file descriptions for master directories. Completely menu-driven.

REMREF II......UPDATED!......$24.95
An ASCII text file containing Author, Title, and Keyword information for REMARK Issues 1-36. Now, available in **ZDOS** format also!

ATTENTION SOFTWARE AUTHORS:
GENERIC SOFTWARE is interested in high quality and well documented software for HEATH/ZENITH computers. GENERIC offers professional packaging, and high royalties. If you are interested in making money from the software you develop, then request our **FREE** booklet, **"SOFTWARE AUTHOR'S KIT"!**

NEW HUG PRODUCTS

P/N 885-1225 [-37] CP/M Disk Dump and Edit Utility (DDEU) $30.00

Introduction: DDEU is a versatile disk dump and edit utility for CP/M that is completely compatible with all Heath/Zenith disk hardware except the Z-67 Winchester.

Requirements: DDEU requires the CP/M operating system version 2.2.02 or later on an H/Z19/H8 or H/Z89 with 32K of memory. Only one disk drive is required.

DDEU is compatible with the H/Z-100 computers under CP/M 85. (Order soft-sectored; i.e. 885-1225-37.)

The source code for DDEU.COM consists of five separate modules as listed below.

NOTE: *DDEU requires the H/Z-19 terminal or the H/Z-100 screen display.*

The following files are contained on the HUG P/N 885-1225 [-37] Disk Dump and Edit Utility disk:

README	.DOC	SKIPLIB	.MAC
DDEU	.COM	SYSCALL	.MAC
DDEU	.MAC	EQUATE	.MAC
DSUB	.MAC		

DDEU.MAC — contains the main program module and all data associated with it.

DSUB.MAC — contains the functional subroutines for DDEU.

SKIPLIB.MAC — contains all of the general subroutines for DDEU. (These are terminal ESCape sequences for the most part.)

SYSCALL.MAC — contains MACROS used in DDEU.

EQUATE.MAC — contains all of the EQUate statements relating to the other modules.

DDEU Program Content: This disk dump and edit utility allows disk access in three different ways:

1) direct track and sector access,
2) sector by sector display or a filename, and
3) sector by sector display of a selected group number.

In all modes of operation, sectors and tracks are numbered starting at zero. Sectors are logical CP/M sectors. In the edit mode, both the cursor and bytes that have been changed are highlighted in reverse video. The sector display consists of both a hexadecimal table and an ASCII table, with editing done in either table. Any change made in either table will appear in the other, after the changes are written to the disk.

DDEU may be run from any drive and after a return to CP/M via a warm boot, it will return to the logged drive. DDEU is completely menu driven. Cursor movement in both HEX and ASCII tables is facilitated by arrow keys and the HOME key.

The main menu of DDEU is as follows:

f1	File Dump
f2	Track/Sector Dump
f3	Group Dump
WHITE	Exit to CP/M

f1 - File Dump — This option allows the user to access any legal filename on a disk. Once a file has been selected, the screen displays the first logical sector of the file. The entire filename, group number, logical file sector number, track number, sector number and last file sector number will also be displayed.

The commands available to the user in the display mode are displayed on the 25th line in reverse video and are as follows:

f1 — advance one logical sector into the file,
f2 — move backward one sector,
f3 — GOTO any other legitimate file sector,
f4 — hexadecimal edit mode,
f5 — ASCII edit mode,
White — return to the main menu.

f2 - Track/Sector Dump — This option allows the user to access any sector on any track on a disk. The program prompts for the track number and then the sector number, while displaying the largest track and largest sector which can be accessed.

The following commands are available for the track/sector dump option:

f1 — advance one logical sector,
f2 — move backward one logical sector,
f3 — GOTO any track and sector,
f4 — hexadecimal edit mode,
f5 — ASCII edit mode,
White — return to main menu.

f3 - Group Dump — This routing allows the user to access the sectors which comprise a selected group number. The prompt asks for the group number.

The following options are commands for the group dump routine:

f1 — advance one sector into the group,
f2 — move backward one sector in the group,
f4 — hexadecimal edit mode,
f5 — ASCII edit mode,
White — return to main menu.

WHITE - Exit to CP/M — When exiting to CP/M, the console is reset to the power up configuration, the screen is blanked, and CP/M does a warm boot and returns control to the CCP.

Comments: This is a screen oriented, disk dump and edit routine for CP/M.

TABLE C Rating: (0),(1),(3),(10)

P/N 885-1226[-37]
CP/M UTILITIES by PS: $20.00

Introduction: This disk contains a collection of CP/M utility programs for listing directory files, controlling printers, and testing disk drives.

Requirements: These programs require the CP/M operating system version 2.0 or higher on 32K of memory on an H19/H8/H17 or H89. The soft-sectored format [-37] will run on the H/Z100 computers. Only one disk drive is required.

NOTE: *The H19 graphics capability (H19, H/Z89,90, H/Z100) is required for DIR19.*

The following files are included on the HUG P/N 885-1226 CP/M utilities by PS.

README	.DOC	PSETMX	.COM
DIR19	.COM	PSETMX	.ASM
DIR19	.ASM	TYPER	.COM
PDIR	.COM	TYPER	.ASM
PDIR25	.COM	PWHEEL	.COM
PDIR	.ASM	PWHEEL	.ASM
DISKID	.COM	WSCON	.COM
PSET25	.COM	WSCON	.ASM
PSET25	.ASM	ROTATE	.COM
		ROTATE	.ASM

Authors:

All programs are by Patrick Swayne, HUG, except DISKID.COM, which is by Marvin Fichter.

DIR19 — This is a disk directory program that takes advantage of Heath/Zenith terminal features to list as many disk files as possible on the screen in an easy-to-read format. In addition to file names, the size in K and the attributes of each file are shown. DIR19 displays up to 80 files on the screen, and if there are more than 80 to show, it prompts the user to hit RETURN to show another "page" of files, and will continue this process for up to 255 files. Files can be listed alphabetically or in their actual directory order. DIR19 takes "wild card" arguments to allow the user to show specific groups of files or individual files. A user number can be specified after the drive and/or file specifications, to show files in different user areas.

At the top of the screen, DIR19 shows the drive name. It also shows the disk volume number and label, if they have been created with DISKID (see below). At the bottom of the screen, the number of files found, the total disk space used by files, and the free space on the disk are shown.

PDIR — This program works like DIR19, but its output goes to a printer. PDIR25 uses H/Z25 graphics, while PDIR is for any printer.

DISKID — This program lets you create a volume number and label on a CP/M disk (for filing purposes). The DIR19 and PDIR programs display the volume number and label when you use them to look at the files on the disk.

DISKID was formerly released on disk 885-1213.

PSET25 — This program allows you to set options on your H/Z25 printer before use. You can set the character spacing to 10, 12, 13.2, or 16.5 characters per inch; or the line spacing to 6 or 8 lines per inch. The program is very easy to use. For example, to set 12 characters per inch, you enter PSET25 12.

PSETMX — This program allows you to set options on your MX80 printer before use. You can set compressed, double strike, emphasized, or normal printing; and you can set line spacing to 6 or 8 lines per inch, or to 7/72 inch spacing for graphics. Like PSET25, this program is easy to use.

NOTE for HRUN users: *The programs DIR19, PDIR, and the two PSET programs can be re-assembled for use with HRUN. Instructions are included in README.DOC and the .ASM files for the programs.*

TYPER — This program lets you type information at your keyboard and send it to your printer. You can include any escape sequences or control codes required by the printer to set up its options.

PWHEEL — This program prints all of the characters on a daisy wheel printer's printwheel in two columns on a single page (including the two "hidden" characters on 96 character Diablo printwheels). An optional heading describing the printwheel can be entered, to be placed at the top of the page.

WSCON — With this program, you can convert ordinary text files created on any editor into the format used by Wordstar for document files, so that the file can be justified or processed by Wordstar.

ROTATE — This program lets you test and/or adjust the rotation speed of any 5.25 inch drive on any Heath/Zenith computer (H8, H/Z89,90, H/Z100) using a Heath/Zenith controller (H17, H88-1, Z89-37, or H207 [H/Z100]) and Heath/Zenith CP/M. If you are having trouble with a disk drive, it could be that your rotation speed is off, and a simple adjustment will restore it to normal operation. The ability to adjust drive speed under software control was formerly available only to HDOS users.

Comments: These utilities cover a wide range of uses. The ROTATE program can be utilized by the majority of Heath/Zenith users.

Table C Rating: (0),(4),(10)
(See HUG Software Catalog page ix for Table C Ratings.)

P/N 885-1227 [-37]
CP/M Cassino Games $20.00

Introduction: This disk contains three Cassino type gambling games that use the graphic features of the H/Z-19 terminal; KENO, POKER, and VEGAS.

Requirements: This disk requires the CP/M operating system version 2.2 or later on an H19/H8 or H89 with 48K of memory. Only one disk is required.

The soft-sectored format will run on the H/Z-100 computer series. KENO and POKER use the function keys which are defined for the H/Z-19 terminal. The function keys of the H/Z-100 will work but are defined differently from the H/Z-19 terminal.

The programs are compiled versions of BASIC-80. Hardcopy source code listings are included as part of the package.

NOTE: *The H19 terminal is required. BASIC-80 is **NOT** required for executing these programs.*

The following files are included on the HUG P/N 885-1227 [-37] Cassino Graphic Games disk:

README	.DOC	POKER	.COM
KENO	.COM	VEGAS	.COM

Authors:

KENO and POKER — Tom Dornback
VEGAS — Larry Wakeford

KENO — This game is a simulation of a Las Vegas KENO Slot machine. A "bowl" of 80 numbers are displayed on the screen. The player selects from 1 to 10 of the numbers. The computer will then randomly chose 20 unique numbers. If enough of the numbers that the player selects are picked by the computer, the player wins.

The player can start the game with any amount from $5.00 to $1,000.00, betting either Quarters or Silver Dollars. The computer keeps track of winnings (and losses). The amount the player can win (or lose), depends on how much is bet and on good luck.

The game uses the special function keys, the 25th line, and other features of the H19 terminal.

POKER — This game is a simulation of a Las Vegas POKER game. Five cards are dealt face down. After the player makes a bet the cards are shown. The player can STAND (or HOLD) or draw any number of new cards up to the five cards. The player can start the game with any amount from $5.00 to $1,000.00. The computer keeps track of winnings and losses.

The game uses the special function keys and other features of the H19 terminal.

VEGAS — This program is a simulated BLACKJACK card game. Seven players can place their bets against the dealer (computer). The computer keeps track of winnings and losses.

Comments: No comments.

Additional Programs on P/N 885-3004-37

In addition to the games mentioned in the February, Issue 37 of REMark for the HUG P/N 885-3004-37, ZDOS ZBASIC games disk, the following games are also included on the disk:

SLOTS	.BAS
DORNBACK	.BAS
QUBIC	.BAS
BATTLE	.BAS

For authors and abstracts to the above programs, refer to P/N 885-1068 in the new HUG Software Catalog (page 22 & 23). All modifications to ZBASIC have been done by Gerry Kabelman.

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			
885-1121	Hard Sectored Support Package	$ 30.00	37
885-1122	MicroNET Connection	$ 16.00	37
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
885-3003 [-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
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	

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.

Vectored from 9

why change something that works? I'd rather work on another program. And also, by going around Murphy's barn to cross the yard, I did manage to slow some things down. One hint I haven't seen in REMark. If the keyboard is turned off, you can turn it back on by holding down the following four keys at the same time. Two hands are required.

CTRL T Y BACKSPACE

It's easier to delete line 45.

If possible, please pass along my apologies to a member in Tupelo for being unable to provide the help he wanted.

REMark really looks great now, much easier to read. Keep up the good work.

In case anyone is interested here is a REMarks file for the program.

15 . . .when omitted puts a curse on your computer. Works almost as well as copyrights.

20-55 . escape codes to terminal. See your operation manual. EX$ is for editing purposes. By PRINTing it after a CTRL C, it puts the terminal back in normal operating modes.

40-45 . Blank the screen, set star flag (A1) and counter (A) to 0, turnoff the keyboard.

60 . .Print Merry Christmas, set flag for one star, print star

65 . .Print row of asterisks at bottom of screen.

70 . . P is the variable for a pause loop; there are a lot of them thrown in for variety. I have discovered that printing long strings of nulls (see Z$ above) causes a pause, and think that might be the best way to do it from now on.

75 . . Gosub print Bethlehem and which star to print (A1=0 here)

85 . .Delete 15 columns of 23 characters (DC$) to make room on right

90 . .Print shepherd and pause

95 . .Insert 24 lines, or, move Bethlehem down

100-105 . Print angels, print moving mouth loop of 5 with two pause loops

110 . . Delete 24 lines, or, move angels up and off screen

115 . . Change star flag

120 . . Print Bethlehem and medium star, move picture up to top of screen to make room for camels

125 . . Go to camel subroutine

130 . . Print Bethlehem and medium star,

overlay with Manger

140 . . Turn on keyboard, overlay with Merry Christmas, print two blinking stars.

145 . . Go do it all again SUBROUTINES OTHER THAN GRAPHICS

160 . . print Bethlehem with star field

165 . . if A1 flag then print medium star and return

170-210 . print growing star, first a period, then a circle, then a diamond, add rays of bigger and bigger sizes.

215-235 . print big blinking star (blinking big star?)

975-1050 . routine for controlling movement of camels. L1 and L2 are the line numbers for the camels, X is column number for camel #1, Y for #2, Z for #3. CY is the total cycle. If line 980 is changed to 5 instead of 10, this routine will end with the 3 camels on the screen, but if you don't want #1 to end up with it's head cut off, add line 987:

987 IF CY=5 AND I>6 THEN 1030

990-1000 . bring partial camels on the screen. If I ever do this again, HA!, they are going to go the other way, and I won't have to mess with partial camels, I'll just delete character them.

1005-1020 . print one of four positions of whole camel, for each, depending on where you are in the total cycle.

In all the camel graphics is a variable called K1$. This is a combination of "go to previously saved cursor position, move down 1 line, and save the new position" or, in terminal parlance, ESCAPE "k", ESCAPE "B", ESCAPE "j"

Have a Happy New Year,

Jen McGraw

Dear Pat;

I just received my issue #33 of REMark and found that someone else did not like the block cursor in their "MAGIC WAND", just as I did not in my "SPELLBINDER". This was all mentioned on page 7 of the Q & A section. I phoned Lexisoft in Davis, California and spoke to a Perry to see if he could help with the modification of my Spellbinder program to also change the cursor from a 'block' to the standard underline that I am used to working with. He looked up the code and I am now really pleased with my new custom version of Spellbinder.

Pat, I wrote you on July 23 of this year with a problem regarding the "Fix Point Package" program. You took the time to make the corrections on my disk and with the utmost speed. Just maybe, by passing on the this Spellbinder information I can say thank you for your kind help.

Well....here is the information that I used from Lexisoft:

Spellbinder version 5.12 — dated 4/1/82

Use only after running the "Configuration" program.

```
DDT SB.COM

S5FBD

5FBD 1B Change to 00
5FBE 78 Change to 00
5FBF 34 Change to 00
5FC0 1B space bar C/R
CTRL C
SAVE 130 SBB.COM (length of configured SB.COM w/user guides)
RENAME To SB.COM
```

Pat, I'm no programmer, but I know you will be able to make sense of my hieroglyphics.

Sincerely,

W.A. (Pat) Call Jr.
2159 Westmoreland Drive
San Jose, CA 95124

Dear HUG;

I recently renewed my subscription (a few months after expiration) and would like to say I like very much the changes I have seen (REMark) in recent months. I like the density increase, the enlarged Q&A section and even the ads.

Keep up the good work!

Sincerely,
Jeffrey S. Close
30 Peachtree Road
Penfield, NY 14526

Dear Walt,

My many thanks to you and the others who labor together to put REMark into the hands of us "HUGgies".

It is true that no one will receive more from something than they put into it. In my case, I received more credit than I deserve in the Pat Swayne write-up on "Recovering Delete Files", page 15 of issue 33.

Please let REMark readers know that Pat did all the work and I only suggested an alternative.

Thanks

Larry T. Wier

Dear HUG,

Since I got my H-25 printer I've been converting many of my programs to print results in hard copy form. In some program modifications I found I was going through many pages of printout before I finished debugging them. To save paper and time I found a little trick that I haven't seen mentioned in REMark so I'd like to pass it on to other members.

Toward the beginning of my programs I insert these lines:

```
PRINT" SCREEN = 1 PRINTER = 2"
PRINT" SCREEN OR PRINTER? (1 or 2)":INPUT Q
IF Q=1 THEN OPEN "O",1,"TT:"
IF Q=2 THEN OPEN "O",2,"LP:"
```

Then anywhere in my program where I have a PRINT statement I simply type:

PRINT #Q,

Following this I enter my string characters or variables as usual.

The beauty of this is that I can debug programs on my screen without wasting paper and I also have a program where I can choose to list the results on the screen or on my printer.

John Czarnecki
815 Valencia St.
Walla Walla, WA 99362

Dear Walt:

I would like to clear up a possible point of confusion some of your readers may have concerning replacements of MTR-88 components in the H-89 computer. I refer specifically to the third question in the Questions & Answers section of REMark issue #37.

It is possible that some of the Heathkit versions of the H-89 computer purchased since October 21, 1981 may have MTR-88 components installed in them by mistake. In any such instances that we are aware of, these parts (444-40 and 444-43) will be replaced by MTR-89 parts (444-62 and 444-61) along with the instruction sheet for their installation (595-2547) free of charge. (Proof of purchase date will be required)

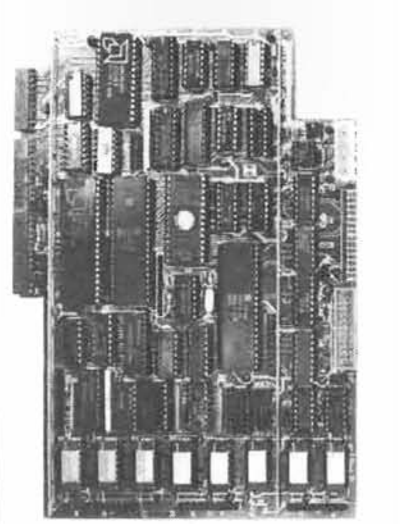
If the computer was purchased prior to October 21, 1981, the above parts must be purchased by the customer. All H-88 and HKS-89 computers prior to that date were produced with the MTR-88 parts as the standard configuration.

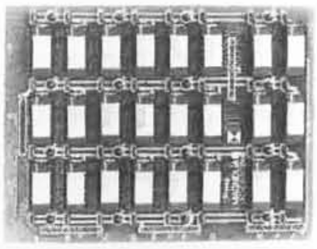
Customers who desire information regarding the purchase of these parts may contact Heath Marketing Services at (616) 982-3285.

Respectfully,

Skip Gwyer
Software Consultation Group

INTRODUCING -- MAGNOLIA's MOST POWERFUL BOARD YET!





The Z89/90 is a good solid computer. But it does have certain limits.

We've had many requests to increase its CPU speed to 4MHz, but nothing met our standards for both field installation and reliability. And who needs anything less.

We increased its RAM to 176K with our 128K board, allowing use of MP/M™. However, without DMA, multi-user operations under MP/M are slow.

Networking '89s together has been a long-term goal of ours. Through the Corvus Constellation multiplexer we provided the first stage -- allowing computers to share Winchester disk storage. But that technique leaves much to be desired -- since each computer independently manages its disk allocation there are significant operational limitations on sharing disk files.

We first announced our impending 'RS422 Network Interface' at the 1982 West Coast Computer Faire. Our design goals included:

- RS422 Network Communication at 800K baud
- On-board Network Control and Data Buffering

As it progressed, we realized that we had designed an 'I/O Board' which was FAR MORE POWERFUL THAN THE '89 ITSELF!

After agonizing delays, the board is now in production as the '77422 Network Controller'. Compare these specifications with the Z89 itself [or the new 'upgrade' CPU boards]:

- 4MHz Z80™ CPU
- 64K RAM, 256K [bank selectable] optional
- 8 K EPROM
- 4 CHANNEL DMA controller
- 2 Serial Ports: RS422 at 500KBaud
  RS232 at up to 19.2 KBaud

We also changed its name from '77321 Network Interface' to '77422 Network Controller'. Our 773xx products must be used in a Z89/90 computer. Although this product CAN be an I/O card in an '89 or 90, it can also be used independently as a stand-alone station on the network:

- permitting a terminal [Z19 or other] as a network station [with local disk storage], or
- permitting a printer to spool data directly from the network.

Network applications software is currently being developed and tested.

Software for an exciting non-network application is included. The workload is split between the Z89/90 CPU and the CPU on the '422 board:

- The Z89's 2MHz CPU continues to run Magnolia CP/M™, handling all physical I/O [console, printer, floppy and Winchester disks]
- The '422 board's 4MHz CPU runs an interface program which 'looks like' CP/M to an application program -- but passes CP/M system calls to the Z89's CPU for execution, allowing a 63K TPA on the '422 board!

With this division, the following Z89 limitations are overcome:

- The application runs on a 4MHz CPU
- The application has 63K of RAM available, independent of BIOS size
- The DMA controller [making a slave I/O processor of the '89 board] frees the application CPU from detailed I/O functions.

This last advantage is especially significant when executing MP/M on the '422 board, using the optional bank-selectable memory.

Hardware-dependent programs [including utilities like 'format'] which perform actual physical I/O, must run on the 'slow' Z89 CPU. A special utility is provided to 'mark' those application programs which can execute more efficiently on the '422 board CPU.

The board is presently available in 64K and 256K versions which mount in an I/O slot [P504/P510] on the '89 CPU board. To limit power dissipation within the '89, no disk drive may be used internally.

64K Network Controller Board
order 77422-064-xx **$695**

256K Network Controller Set
order 77422-256-xx **$1295**

CP/M Operating System by MMS if needed
order CPM-xx add **$100**

MP/M II Operating System
requires 77422 order MPM-xx **$595**

CP/NET™ Network Software
requires 2 or more 77422 boards, one with MPM
order CP/NET-xx **$450**

Use 'xx' to specify preferred media format

MAGNOLIA MICROSYSTEMS
2264 - 15th W • Seattle, WA 98119
(206) 285-7266 (800) 426-2841

CP/M, MP/M II, and CP/NET are trademarks of Digital Research, Pacific Grove, CA.
Z80 is a trademark of Zilog, Cupertino, CA.

Introducing The Heath Educational Robotics Organization

Yesterday's Dreams ... Today's Reality

Bob Ellerton
HUG Manager

How many times have you found yourself thinking of our modern technology and the advances made over the past century. For most of us, we read the history books and ponder this awesome growth. Some of you have even experienced these advances, many of which are considered common place today. The electric light, the automobile, the airplane, the radio and the rocket are just a few of the items that come to mind.

Just think, the rocket engine was a scientists and hobbyist toy in the thirties, a device that seemed to have very little practical value, the same device that eventually enabled man to set foot on the moon in July, 1969. At one time or another, we have all read science-fiction comics or novels describing such incredible machines as a nuclear-powered submarine, spaceships, lazer beams, etc. even before these devices had "official" names.

Probably one of the most popular subjects of fiction and the subject of countless movies, comics and novels is a device known as a **robot**. The subject of robots usually leads one to think of a device that performs tasks and is human-like with appendages designed to simulate human characteristics. Beyond the robot itself, we each think of a device with some intelligence, a device capable of independent actions once "taught".

As with some of the machines previously mentioned, the robot is generally thought of as a fictional device. Well, not any longer! Advances in micro-electronics, specifically the microprocessor, have given us some of necessary tools to create a human-like device that, no doubt, will advance in characteristics as did the micro-computer over the past 10 years.

Heath Company, an electronics leader in the kit electronics business, began the process of investigating the possibility of putting together a robot shortly after the introduction of the micro-computer kit early in 1978. As the plan seemed to become more viable, Heath used its expertise as a leading instructional manual developer and electronics kit company to produce the first educational look at this fascinating subject area. The Heath Educational Robot, now know as HERO I, was introduced to the public in December 1982.

The introduction of HERO has stimulated the imaginations of those people of all ages who tend to find themselves on the leading edge of technology throughout the world. HERO has drawn the attention of national and international news agencies to a degree that caused the local community to offer this unique machine the Key to the City. Indeed, HERO is the incarnation of the fictional robots we have all come in contact with. Beyond that, HERO has a purpose. This device is designed to lead us into a new age ... the age of robotics.

Since HERO is a computer, it seems logical to form a place here in REMark to exchange information, programs, ideas and modifications for this little wonder. In the next few issues interfacing hardware and software will be offered to make HERO compatible with our world of micro-computers. One of the first articles will be devoted to interfacing the H-89 with the HERO I. Also planned are some small programs that will enable the HERO user (or should I say pet-owner) to teach HERO to be more useful.

If you are new to the Heath Users' Group, or if you are an avid HUGgie, stay tuned to see HERO grow. Remember, HERO is the "son" of our current hobby. If you would like to contribute to this very special column, please feel free to contact the REMark Editor with your thoughts or articles. It's just possible we are on the edge of the formation of another user group directed toward the growth and investigation of robotics. Any comments you may have are most welcome.

Introduction To Z-BASIC Part IV

Gerry Kabelman, C.E.T.
Zenith Data Systems

This is the fourth article in a series of articles dealing with the new commands of the Z-100's Z-BASIC over BASIC-80. Previous articles dealt with the CLS, LINE, LOCATE, KEY, PSET and DRAW commands, plus using the 25th line. This month we will look at the PAINT, GET and PUT commands being used to create objects on the screen, placing the object in memory and retrieving it from memory.

Let's take a piece of last month's code and use it for creating an object, in this case the star.

```
10 CLS:'        STAR.BAS        GK:
20 C6=6:PSET(26,3),C6:'         Create Star
30 DRAW"F3R3G3D3H3G3U3H3R3E3"
40 PAINT(26,4),C6
```

THE above four lines are pretty straight forward in that line 10 clears the screen, line 20 sets the variable C6 to the value of six, and turns the dot on at location (26,3) to the value of C6. Variables may be used in place of all numerical values within the PSET command. The main reason to use variables instead of a number is that if all use of a color is to be changed to another only one variable needs to be changed. Line 30 is simply drawing the star that was used last month. Line 40 is using the first of the new commands, which is the PAINT command. The PAINT command's syntax is:

```
PAINT(Xstart,Ystart)[,paint attribute [,border attribute]]
```

That looks like a very complex command, but it is quite simple when normally used. In the above example the Xstart location is 26, the Ystart command is 4 and the paint attribute (color) is the variable C6. The border attribute was not used so it defaulted to the paint attribute. When using the PAINT command it is very important that the area to be painted is completely surrounded by the border color (or the paint color if no border color is being used). If the PAINT command is directed to start the painting function on the border of the same color that is to be painted the function is canceled.

Try using the PAINT command inside an object such as a square and then try painting outside the square and see what happens.

```
10 LINE(0,0)-(100,50),7,B:PAINT(50,25),7:'  Inside
```

or

```
10 LINE(0,0)-(100,50),7,B:PAINT(200,50),7:' Outside
```

Now try the program at the beginning of this article and try changing the X and Y starting locations for the PAINT command and note the results.

The next new command is the GET command. Before using the GET command another command must be used to prepare for the GET command. That command is the DIM command for dimensioning the size of the object that we wish to GET. To try to keep this as simple as possible we will ONLY deal with the double precision numbers (#).

First we must decide how large our object is by determining the outer most X and Y points of all four sides of the object. Sounds like a real problem, but let's simply imagine that we are to draw a rectangle all the way around the object with the rectangle covering the points of the star. Add this line to the four lines at the beginning of this article and see if the star is now enclosed within the rectangle.

```
50 LINE(20,3)-(32,12),7,B
```

Try changing the numbers in the above line and note how the tips of the stars will start to show either inside the rectangle or outside. Also try using the zero (0) for the color of the box and note how the tips appear to be gone when using the coordinates listed above.

Now that the exact size of the star has been determined, let's calculate the width and height of the box by using a simple formula.

```
Width  = (Xend - Xstart) + 1
        or
    13 = (32   - 20    ) + 1
Height = (Yend - Ystart) + 1
        or
    10 = (12   - 3     ) + 1
```

That means that we will use an area that is 13 dots (pixels) wide and 10 dots high. Or an area of 130 pixels (13*10=130).

We now need a formula to calculate the total dimension that is required for using the GET command. The formula will set aside the memory for the object and must represent size of the object. The formula below will do that job for us.

```
Dimension = (4 + ((Width + 7)\8) * 3 * Height)\8
        or
        8 = (4 + ((13    + 7)\8) * 3 * 10    )\8
```

Believe it or not I did not dream up that formula, it is actually in the Z-BASIC manual, just written slightly different, but I think the above formula is a little easier to understand. The back slashes (\) used in the formula mean to divide using only the integer of the result when finished.

The formula may actually be broken down into three parts, first we calculate the width as an integer value of the actual width.

```
Width=(Actual Width+7)\8 or 2=(13+7)\8
```

Once the integer width has been calculated then it is multiplied by the three color planes and the actual height, give us a square area be covered by the object.

```
Square Area = Width * 3 * Actual Height or 60=2*3*10
```

The four is then added to the square area and the result is divided by eight (8 is for double precision numbers), using only the integer for the final results.

```
Dimension = (4 + Square Area)\8 or 8=(4+60)\8
```

The dimension comes out to a value of eight (8) and should be written into the program by typing DIM A#(8) where A# is the variable name to be used for object (star), in double precision.

Let's take line 50 and modify it by adding the DIMension command and changing the LINE command to the GET command. The syntax of the GET command is:

```
GET(Xstart,Ystart)-(Xend,Yend),Variable.
```

This is the way line 50 should now look.

```
50 DIM A#(8):GET(20,3)-(32,12),A#
```

Now, that the star has been saved, into memory, we may place the star anywhere on the screen. The command to put something on the screen is the PUT command, which has the syntax of:

```
PUT(Xstart,Ystart),Variable
```

Try using the PUT command by picking the location on the screen and adding another line to the program such as:

```
60 PUT(100,100),A#
```

Then add another line that is the same as 60 at 70.

```
70 PUT(100,100),A#
```

When running the program with both lines 10 to 70, make sure lines 60 and 70 are the same. The program will first create the star in the upper left corner, then PUT it at location 100,100, and then erase it from that location.

Let's try putting all this information on GET's and PUT's to a useful application.

```
10 CLS:'        STARS.BAS Version 01.21.83    GK:
:
20 C6=6:PSET(26,3),C6:'        Create Star
30 DRAW"F3R3G3D3H3G3U3H3R3E3
40 PAINT(26,4),C6
50 DIM A#(100):GET(20,3)-(32,12),A#
:
60 CLS:LOCATE 25,16:PRINT"Press any key to stop"
:
70 R=35:FOR I=1 TO 13:'              13 Stars
80 C=(I*(360/13))*3.14159/180
90 PUT(135+(INT(COS(C)*R))*2,63+INT(SIN(C)*R)),A#
100 NEXT I
:
110 A$=INKEY$:IF A$="" THEN 70 ELSE CLS:LIST
```

Take a close look at the above program noting line 60 clears the screen and prints a message on the screen. Lines 70 to 100 actually use a FOR-NEXT loop to create thirteen stars.

The PUT statement in line 90 looks very complex, however, if we break the line into three separate lines the PUT command won't look so complex.

```
90 X=135+(INT(COS(C)*R))*2
91 Y=63+INT(SIN(C)*R))
92 PUT(X,Y),A#
```

The formulas in 90 and 91 will calculate the X and Y coordinates or we could have combined the lines as done in the original line 90.

Line 110 is used to check for a keyboard input and if one has been received then clear the screen and LIST the program. Using the LIST command in the program is very useful for program debugging. If no input is received the program will continue on at line 70.

Try the program and you will see how the PAINT, GET and PUT commands work.

Next month we will learn some additional ways to use the PUT command and how to use the CIRCLE command.

HOME FINANCE SYSTEM
VERSION 2

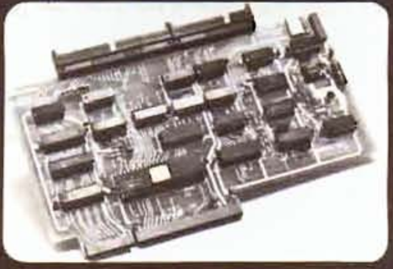
—An extensive Home Finance System that keeps track of checking, asset accounts (cash, savings, IRAs, CDs), and regular bill payments. Let your printer write your checks for you on **any** business-sized check (design your own check format).
—Checks have user defined codes and a separate flag for tax deductible items.
—Many reports, including listing all checks, or checks by codes or tax flag.
—System consists of 100 page users manual with 5 program disks (5-¼") and a sample data disk.

Hardware: H8/H19 or H-Z89/90 with 64K RAM and two disk drives. Printer strongly recommended (any Heath® Zenith® or other printer).
Software: HDOS 2.0 and Microsoft® MBASIC for HDOS.
—Complete system: \$89† (specify hard-soft sector 5-¼", or 8").
Manual alone \$21†.

Order From: *†Prices include shipping*

Jay H. Gold, M.D.
Jay Gold Software,
Box 2024, Des Moines, IA 50310
(515) 279-9821

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. HSOD 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. (714) 560-1272

ZTERM PATCHES

The ZTERM program released as HUG part no. 885-3003 has a problem in the part that allows baud rate selection from the menu (when you type ZTERM B) in the Z100 version does not work. We have corrected the problem and are updating our stock, but if you already have the program, you can patch it with DDT. This patch should be made to the Z100 version (called ZTERM100.COM on the HUG distribution disk). In the patch example below, with ZTERM100.COM and DDT.COM on drive A:, what you type is shown in **bold print**.

```
DDT ZTERM100.COM
NEXT  PC
XXXX 0100            (XXXX is a number)
SDE9
  0DE9 87 0
  0DEA 11 .
-S107F
  107F 02 72
  1080 03 73
  1081 04 74
  1082 05 75
  1083 06 76
  1084 07 77
  1085 08 78
  1086 0B 7B
  1087 0C 7C
  1088 0D 7D
  1089 00 .
-G0              (Letter G, Number 0)
A>SAVE 16 ZTERM100.COM
```

Do not make the patch if the numbers printed by DDT (before the new numbers shown in bold print) do not match what is shown here.

If you prefer, you can make the patch to the assembly source code and re-assemble it. Locate the label BAUD1 near the end of the program, and add the lines shown in **bold print**.

```
BAUD1   MOV     E,A             ;ANSWER IN E
        CALL    DCONSOL         ;PRINT IT
        SUI     30H             ;MAKE 0-9
        ENDIF
        IF      H8H89
        ADD     A               ;DOUBLE IT
        ENDIF
        IF      H8H89 OR Z100
        LXI     D,BAUDS         ;GET BAUD VALUES
```

Change the second line with the lable BAUDS from this:

```
BAUDS   DB      2,3,4,5,6,7,8,11,12,13
```

to this:

```
BAUDS   DB      72H,73H,74H,75H,76H,77H,78H,7BH,7CH,7DH
```

There is another problem in ZTERM that affects all versions, and can only be fixed by changing the assembly source code and re-assembling. It does not allow enough space for a maximum length MicroNET password. This modification increases the space for both the password and ID number (just in case). First, locate the label MSG23, and change the numbers shown in bold print:

```
MSG23   DB      '70898,99',13,'$'
        DS      5
MSG24   DB      'YO-MAMA',13,'$'
        DB      4
```

A few lines below the label RDONE, make the changes shown here:

```
CALL    PMSG                    ;PRINT "ENTER ID"
MVI     A,13                    ;ALLOW 13 CHARACTERS
CALL    CONIN                   ;INPUT MICRONET ID
LXI     H,MSG23+CBUF-256  ;PUT IT HERE
MVI     B,14                    ;MOVE 14 CHARACTERS
CALL    MOVE
LXI     D,PASMSG
CALL    PMSG                    ;PRINT "ENTER PASSWORD"
MVI     A,11                    ;ALLOW 11 CHARACTERS
CALL    CONIN                   ;INPUT PASSWORD
JZ      GSOURCE                 ;NO, GET SOURCE
LXI     H,MSG24+CBUF-256  ;PUT PASSWORD HERE
MVI     B,12                    ;MOVE 12 CHARACTERS
CALL    MOVE
```

This completes the modifications to ZTERM. Be sure you set the assembly parameters at the beginning of the .ASM file to make the version you need before you assemble it, as outlined in the ZTERM documentation.

If you want to become more productive with your computer. . .

DON'T

buy this game.

If you're like a lot of people, you bought your Heath computer to do home finance, word processing, or perhaps to learn how to program. You've got PerfectStar, DataPlan, MergeWriter, and all the latest Pascal compilers. Fine. Just don't buy GRAVITRON.

We're telling you this because we've noticed that people who play GRAVITRON get hooked. As they pass through the warp gates to the first planet, they realize that this is no ordinary game. They are intrigued by the challenge of maneuvering a spacecraft that is subject to the force of gravity. Their skill increases, and they move on to other worlds, each more extraordinary than the last. Soon they become obsessed with the desire to explore all the strange (and lethal) planets of GRAVITRON.

If you want to give it a try, go ahead. But you've been warned. Once you visit the worlds of GRAVITRON, you may not want to come back to Earth.

GRAVITRON, $19.95 plus $1.50 shipping. Requires H/Z89, Z90, or H8-H19, 32k, and HDOS or CP/M. Specify disk format (hard or soft sector 5-¼"). Free catalog of games and utilities. CP/M t.m. Digital Research.

APOGEE SOFTWARE

Box 15124
Savannah, GA 31416
(912) 925-3765

In Defense Of Magic Wand

David B. Garwood
Phillips & Garwood
State Bank Building
Red Coud, NE 68970

Lately, Magic Wand has come under some criticism, notably from Mr. Hugh Kenner in the Fall 1982 issue of Sextant. His comments there stirred me to write this rebuttal, as I believe that Magic Wand is an excellent word processing system. I am an attorney in private practice and have been using Magic Wand almost daily for two and a half years. My experience with Magic Wand began in the spring of 1980 as I was looking for a way to increase productivity in my office.I had previous experience with Heathkit, and so in my search for an affordable word processing system good enough for the law office, decided to include a trip to the Omaha Heathkit Center. I was pleasantly surprised to find a unit of the quality and capability of the H89 and took one home the same day. Approximately 40 hours later I had a functioning HDOS system. I quickly realized that the system editor was not satisfactory for my purposes, which resulted in another trip to the Heathkit Center. There I learned that Magnolia Microsystems had developed a modification which would allow me to run standard CP/M and their version of Magic Wand, a word processing system by Small Business Applications, Inc. (Magic Wand is now owned by Peachtree Software.) I placed my order and in about a week was ready to try Magic Wand.

The designers of Magic Wand apparently felt that print formatting functions should be performed when the document is printed, contrary to the design philosophy of Wordstar and some other editors which do the formatting at the same time as the editing. The system is therefore separated into two major programs, EDIT.COM and PRINT.COM, comparable to PIE and TEXT by Software Toolworks, used by Mr. Kenner (and myself in the HDOS version). PRINT.COM takes a file created with EDIT.COM and processes it, using embedded formatting commands and, optionally, commands given to it from the keyboard. Commands can be on separate lines or embedded at any point in the text using the command marker (default is "\" but can be changed to any character) to delimit the beginning and end of the command string. Thus, the line "Now is the time \CPI10,LM10,RM65\for all good men to come to . . ." would be printed at 10 characters per inch with a left and right margin of 10 spaces, assuming letter size (8.5 in. wide) paper is being used. The argument to the RM command is actually the line length which will result in the desired right margin. Any commands on a line are processed before the text, and are taken in order from left to right. Therefore in this line the LM and RM commands are based on the 10 character per inch spacing command. These commands will also override any pre-existing or default values.

One of the things I most like about Magic Wand is the ease and quickness with which those who are NOT computer literates can begin to do useful work with it. I believe this is due to the design of the system. Each of the commands in both the PRINT and EDIT programs is carefully chosen for a mnemonic value which relates to its function. Thus, LM followed by a number sets the left margin; JUST means justify both margins using blank insertion; and FORMS means single sheet paper as opposed to FORMC which is continuous form paper. Thanks to Doug Miller of Magnolia Microsystems the only control key sequences used are "control T" for moving the cursor to the top of the text and "control B" for moving the cursor to the bottom of the text. The forward and backward line and page scroll functions, page feed, search & replace, repeat search and block marker have all been assigned by Doug to the top row special function keys. I have replaced these keycaps with a set from Arkay Engravers so that I have what appears to be a dedicated word processor. I did not change the keycaps on the numeric keypad, since the mnemonics were already correct - IC for insert character, DC for delete character, IL and DL for insert and delete line respectively, and the home and arrow keys for cursor motion. The IL key is a toggle, the first press opening up the text so that as many lines as desired may be typed in without interference or overtyping, and the next press closing up the text as before. To prevent unintended results, the DL key must be pressed twice to delete from the cursor to the end of the line. Doug also changed the delete key to a word delete key. (NOT the same as the Heath version). Because the edit functions are so obvious, almost anyone can be entering text within five minutes.

One disadvantage of the separate edit and print functions is that you have no way of knowing where the page breaks will occur. This problem is dealt with in two ways. First, lines can be tied together, so that if less than a given number of lines remain on the page, a new page is started and the tied lines printed together on the next page. This is important in a law office, where we like to have at least one line of text on the page with the signature lines. Secondly, Vers. 1.1 (update furnished to me at no charge by Magnolia Microsystems) includes the SCREEN ON command which directs the output from PRINT.COM to the screen instead of the printer, showing exactly where page and line breaks will occur. This also gives you a chance to quickly proof your work, catch logic errors such as missing command markers, and saves a lot of paper. In practice, we have found that most of our work product which requires more than one page is a previously used form, so we know where the page breaks will occur.

I take exception to Mr. Kenner's statement that "the manual is a shambles." There is no way to learn a system as extensive as Magic Wand except to use it, and that is exactly what the manual does. It includes nine lessons, which are used with sample files furnished on the distribution disk, and lead the beginner through the simple editing commands to the more complex print commands. It also includes Notes on Edit and Print, which is an alphabetical listing of every command with a description of its function and uses; a glossary; helpful comments for CP/M users; and some Notes for Programmers, describing in detail the structure of Magic Wand disk files. Since the system does not come with a two week course and instructor as do some of the dedicated word processors, I expect the manual to train my personnel. It does this very well.

When I began lesson one, I learned that the Gettysburg Address reached its immortal form through many, some rather humorous, revisions. The first draft starts out "Its great to be in Pennsylvania. Mrs. Lincoln and I appreciate the hospitality you have shown us during our stay." Following the step by step instructions I typed EDIT SAM-

PLE1 LESSON1 and a carriage return. This instructs the system to load and execute EDIT.COM which will in turn load the input file SAMPLE1 and establish LESSON1 as the output file when the session is ENDed. As the edit program loads, the system stops to allow a change of disks in case you have only one drive and wish to keep text files on separate disks. This is apparently a hold-over from Vers. 1, as EDIT now permits swapping disks at any time. No more Bdos errors or lost files. When the desired disk is inserted another carriage return quickly brings you to the command screen. At this point another carriage return puts you into the text file to be edited, from which ESC will return to the command screen. The command screen display shows the edit system status - the active files, number of lines, words and characters available and in use, the Mode setting, line length, and tab setting. It is from the command screen that the edit commands for block functions, file manipulations, disk directory, edit screen line length and tab settings, text or program mode selection, and the quick-print feature are available. The program mode, which is automatically selected if the output file extension is ASM, MAC, COB, FOR, BAS or PRN, allows editing of Microsoft BASIC files which have been SAVED in ASCII format. I now use the built-in basic editor only when debugging. When you become accustomed to moving the cursor to any point in the text at will, it is very difficult to go back to a line oriented editor.

We have also found the file manipulation commands added in Vers. 1.1 very useful, especially the X (for eXtract) and C (for Change input file) commands. There are many letters to write in a law office, and we have found the following has become the standard operating procedure. EDIT is invoked without specifying either an input file or output file. When at the command screen, the command CLTR is given to Change the input file to LTR and the R command then Reads this letter template file into memory. When editing of the letter is completed the P= command sets the page length and left margin for the P command, which prints the letter without exiting EDIT. PB (print the text within the block markers) then prints the address on the envelope. If it is a first draft or might be needed later, the letter is then saved on disk with the command X=filename. If the file already exists on the disk, you will be asked if you wish to write over it and the block delete function is then used to clear memory for the next letter. I have installed the proportional logic on my Xerox 1750, so that letters printed from EDIT can also be proportionally spaced. This works well but the more complicated printing tasks are still left to PRINT with its own proportional spacing logic, bidirectional printing (BI ON and BI OFF) and many other goodies.

Though the editing commands are quickly learned, I am still learning new applications for the other features of Magic Wand. The commands available make it like a high-level programming language, allowing you to automatically number paragraphs, use variables, create and use data files such as address lists, and many other applications. In my law practice a form used quite often is an affidavit to prove mailing of a copy of published notice to everyone involved in the proceeding. I use the DISK ON command of PRINT to create a disk file containing the names, addresses and other pertinent data such as social security number, family relationship, etc. of the persons involved. This file is then used to supply the values to the name and address variables when the affidavit is printed, so that a list of names and addresses is printed within the body of the affidavit. The list is also used to generate letters, including the proper address, salutation, etc., and envelopes as required. Another example of the programming power of Magic Wand is the ability to number paragraphs. When drafting a contract we use a file containing many possible alternative paragraphs. The contract is composed by including the desired paragraphs in the desired order with the command Ifilename, then I for the first screenful of the file or I@heading for a specific paragraph. Each paragraph begins with the command \#P=#P+1,:P\. This causes 1 to be added to the value of the variable "P" each time it is encountered and the resulting value printed. The system itself keeps track of the page, pass (how many times the file has been processed), record number of the external data file being processed and if it is the last one, the current line number and lines left on the current page, and the current column location on the print line. Use of these variables allows you, for example, to number the page, or process a data file to the last record. There is also an IF command which makes it possible to compare string and numeric variables and values using =, <, >, =>, =< and <>. The GET command allows you to input a value to a variable and SET is used to assign values, similar to the INPUT and LET statements in BASIC.

The print formatting features allow you to do almost anything desired, including horizontal and vertical motion in 1/48 and 1/120 inch increments respectively on capable printers, such as the Diablo. Also supported are centering, boldface (9 intensities), subscripting, superscripting and underlining (solid or broken). We normally print contracts and other documents proportionally spaced (PROP ON), with space and a half line spacing (SP+1) and justified (JUST) using blank insertion as opposed to character spreading (JUSTC). This gives the most satisfactory appearance with the Xerox 1750KSR and Cubic PS or Bold Legal metal printwheels. There is a Configur utility included to customize the system to the printer you happen to be using. I have written a help file to keep track of the locations of the various special characters on each printwheel, such as the degree and section signs. If you cannot remember what the screen must show, and how the printwheel select switches must be set to get the desired character, just type DHELP (Dfilename) to display the help file. A help file is furnished on the distribution disk showing the proper form of address and salutation for various dignitaries. There are several other D commands, among them DS, which will display the current status of the major text shaping and formatting commands.

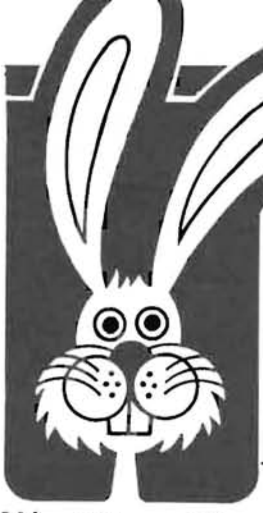
With all its features and strong points, there are some weaknesses or "bugs", if you prefer. When using proportional spacing, the TAB command seems to lose its place on the print line, so that columns of numbers interspersed with text will not line up. The solution is simple, though it does slow down printing somewhat. You first turn off vertical spacing, then print the text, use the NL (new line) command so the column is reset to 0, and then TAB to the desired position. For example the line "\SP0\Text . . . \NL,TAB60,SP1\$10.00" will all print on the same line. The next line is done the same way and the decimals will align. The other problem occurs when processing a multiple page document using the START command. The system loses its position on the print line and the printer will do strange things. This too is easily handled. Just remember to give the command BI OFF before beginning processing. When printing starts at the desired page you may then interrupt printing and resume bidirectional printing.

There are many other features and commands I have not mentioned, but I hope my experiences related here are helpful to others who are using Magic Wand or may be wanting a word processor. I would like to hear from other users who have applications or problems they would like to share. It is apparent to me that my word processing needs and expectations are as different from Mr. Kenner's, as his may be from the next person's. Since the applications for "word processing" are so diverse, it is unfair to make a judgment based on only one application. One man's bane may be another's blessing. The defense rests.

Forgotten DOC File!

The program CHEAPCALC.BAS, presented in the February 1983 issue (37), looks for a file called MBCALC.DOC at line 2790 if you answer '7' at line 2770. Below is that file, type it using a text editor and file it as MBCALC.DOC.

```
MBCALC variables are limited to a single capital letter (A-
P) followed by a numeric index that can vary from 1 to 4Ø
(e.g.  A1,B33,C39 are allowed while AA1 and C41 are
illegal).  Numerical constants can also be used in the
mathematical operations described below.

Movement through the cells displayed on the screen is
accomplished with the arrow keys (shift-keypad) or by the
command mode discussed below which moves the entire screen
display area to another part of the spread sheet.

The displayed cells can contain constants, operations, or
text as selected by the user.

MBCALC operations can consist of combinations of the
following mathematical operations:

+,-,* ,^ & / are all usable operations with the usual
definitions, but parentheses can not be included [12+A1/15-B4
is allowed but 3*(A1+B2) cannot be used].  The mathematical
operation does not have to fit in the cell width since only
the result of the operation is displayed in the output
array.

Editing of the input cells is accomplished on the 2nd line
of the CRT where the input cell parameters are displayed.
The user must enter the first character of the input cell,
but then can copy the characters from the screen by using
CTRL-U to move the cursor over the screen characters.  The
basckspace & delete keys both delete characters.

Certain characters are reserved for commands to MBCALC.
@,:,!,/ and & are used in the first character of an input cell
as commands.

@ is used to clear the input cell

: is used to define the format of the numerical results in
all of the cells of the problem.

   :I displays the operation result as an integer
   :$ is the default format and displays dollars and cents
   :F displays the operation results in floating notation
   :* displays the operation results as ******

! causes recalculation of all operations

/  gives the user the choice of 7 commands

   1. Set column width of output array at current column
   2. Save data file on disk
   3. Load data file from disk
   4. Clear all data in input array to get ready for new
      problem input.
   5. GOTO input array location which will then
      be the upper-left hand cell on the screen
   6. Printout output array to hard copy device
   7. Display this file on screen

& Defines range of summation operation by asking for the
first cell location and the last location.

CONTROL-R is used to replicate cell parameters by first
positioning the cursor cell at the cell to be replicated.
Enter CONTROL-R and move the cursor cell to the desired
location where CONTROL-R is again entered to replicate the
parameters with incremented indices.
```



A Programmable 3 MHz Modification For The H8 (With HA-8-6)

Pat Swayne
Software Engineer

When I did my 4 MHz mod on my H89 at home (see "A Programmable 4 MHz Modification for the H89", REMark #34, page 25), it got to be a bit of a drag to spend all day long in my HUG office contending with a 2 MHz H8. Besides that, I figured that some of my loyal readers might think that I had abandoned the old boy. So I got out my H8 schematics and looked into ways to pep him up. I saw right away that increasing the speed on the original 8080 CPU board would involve a bit too much. I also saw that a speed increase all the way to 4 MHz would be difficult to implement, and settled for a 3 MHz modification on an H8 equipped with the Heath HA-8-6 Z80 CPU board.

If 3 MHz doesn't sound too exciting to you, be advised that a 3 MHz Z80 with no wait states runs about as fast as a 4 MHz one with one wait state (which some folks ask you to put up with), and just a tad slower than the 8-bit side of a Z100 type computer (5-MHz 8085 with one wait state). This modification also offers these advantages.

1. The system comes up at 2 MHz, so no modifications to the boot ROM are required. This also ensures compatibility with obsolete operating systems that you may be using that you cannot modify for 3 MHz operation.

2. The system can be programmed to run at 3 MHz at any level. It can be programmed at the operating system level, or, if you do not wish to modify your operating system, it can be programmed at the user program level — even from BASIC.

3. This modification works with "stock" Heath memory and I/O boards. It has been tested with the 8k RAM board, the 16k board, and the Heath 64k board.

NOTE: This modification uses bus pin 24, normally a ground or not used on a standard system. If you are already using pin 24 for something else, such as hard sector disk side select, you should use another pin (18 is also free) or an external jumper wire for that other purpose to free up pin 24 for this modification. The modification involves changes to the CPU board and minor changes to the H17 disk controller board and one front panel connecter (the front panel does not have to be removed). The old 8k RAM boards may run even hotter at 3 MHz than their usual egg frying temperature, so you may want to provide additional cooling.

The Modification Circuitry

The modification utilizes two unused IC positions on the HA-8-6 board that are labeled U29 and U34. Figure 1 is a schematic of the modification circuitry, which consists of a 7492 divide-by-12 counter (only the divide-by-6 section is used) and a 74S132 quad two-input NAND Schmitt trigger. The counter is used to divide the 18 MHz output of the board's oscillator down to 3 MHz. This is fed to the NAND gate along with the 2 MHz signal and an unused output from the General Purpose port IC, U30, which is used to select between the two frequencies. The 2 MHz signal is also sent to bus pin 24 regardless of the processor speed selected. This is because the front panel requires it to generate the 2 ms clock interrupts, and the H17 disk controller board also requires a 2 MHz signal.

Construction

To build the circuit, unplug your computer and remove the CPU board. If there is any solder in the pad holes at the unused IC locations U29 and U34, remove it with a solder suction tool or solder removal braid. Remove the jumper wires at E1-E2, E6-E7, and E9-E10. Install jumper wires from E1 to E18, from E2 to E7, and from E3 to E37. Two of these jumps are quite long. You should use insulated wire for all three, and you may want to tape the long wires to the top of the IC's they will pass over.

Install 14 pin sockets (434-298) at U29 and U34. Make the connections shown in the schematic on the back of the board, using jumper wires from the pads of U29 and U34 to the pads of the other IC's or jumper pads shown. Set the CPU board aside temporarily.

Remove the lower of the two connecters that connect wires from the front panel to the mother board. Remove the wire from hole 22, which is the uppermost wire on the connecter. To remove the wire without pulling it off of the spring connecter, press in with a small screw driver or scribe into the slot on the side of the connecter, depressing the barb that holds the spring connecter in place. When you have removed the wire (and its connecter), re-install it in hole 24 and replace the connecter on the mother board. If you depressed the barb too far in on the spring connecter, just hold the wire while you replace the connecter on the mother board.

Remove the H17 controller board (if you have one) and cut the trace coming from pin 13 of U23 on the top side of the board. On the back side, connect a jumper wire from the pad by bus connecter 24 to U23 pin 13. Replace the controller board.

Check all of your wiring on the CPU board and install a 7492 at U29 and a 74S132 at U32. Replace the board in the computer and turn it on. It should come up normally with a beep and numbers on the display. If it does not, check your wiring carefully, then check the new IC's.

Testing at 3 MHz

When I modified my system, it worked perfectly at 3 MHz without any memory or other failures, and without having to replace any memory or other IC's. But there is always a chance of failure in something like this, so you should test your system thoroughly.

The area where the system is most likely to fail at higher speed is the system's memory (RAM or ROM). If your system is configured for extended operation (CP/M compatible), the only memory actually used is the RAM (the ROM is copied into RAM at power up).

To test your memory, enter the following code using the front panel. This is the memory test that is listed in your H8 operation manual.

Address	Data	Address	Data	Address	Data
040100	041	040120	000	040140	043
040101	160	040121	052	040141	302
040102	040	040122	101	040142	125
040103	021	040123	040	040143	040
040104	260	040124	004	040144	303
040105	xxx	040125	064	040145	121
040106	066	040126	176	040146	040
040107	000	040127	270	040147	172
040110	315	040130	312	040150	254
040111	147	040131	135	040151	300
040112	040	040132	040	040152	173
040113	043	040133	166	040153	255
040114	302	040134	000	040154	311
040115	106	040135	315		
040116	040	040136	147		
040117	006	040137	040		

Replace the **xxx** shown at address 040105 with a number from the following chart.

Total RAM memory	Extended system	Standard system
32k	177	237
40k	237	277
48k	277	337
56k	337	377
64k	377	---

For example, if you have 3 16k RAM boards (48k total), and your system is not CP/M compatible, then you would use 337 as the number to enter at 040105. After you enter the test, enter REG PC ALTER 040100 ALTER to set up the program counter, and REG BC to display the BC register. Then press GO to start the test. The left three digits on the display will start incrementing (000, 001, etc.). Take note of how fast they increment, then press RST/0 and 0 to stop the test.

Now enter MEM 050362 if you have an extended system, or MEM 010362 if you have a standard system, and press OUT. The system has been switched to 3 MHz operation. If it does not continue to function normally at this point, then you probably have a problem in the modification, or a marginal component that cannot operate at 3 MHz. It is also possible that your Z80 will not work at 3 MHz, which means that you must replace it with a Z80A (Heath part no. 443-953). If everything seems OK, enter REG PC ALTER 040100 ALTER REG BC and press GO. The left three digits should increment as before, but noticeably faster. If they do not increment faster, you have a problem in the modification circuitry.

If the test is running as it should, let it continue until the left 3 digits reach 377. If it stops before that, you have at least one component in your memory that cannot operate at 3 MHz (assuming that your memory is good at 2 MHz). Press REG HL to see the address where the test failed, and use that address to locate the bank of memory IC's that failed. The B register will hold the data that the bad address should have contained, which may enable you to narrow the problem to a single memory IC.

If the memory test passes, reset your computer (RST/0 and 0) and boot up on HDOS or CP/M and run BASIC or MBASIC. Load in a game or other program (Robot NIM is a good one for this), and RUN the program, noting how fast things happen. Then enter

POKE 13,42:OUT 242,42 (for CP/M)

POKE 8246,42:OUT 242,42 (for HDOS, Extended system)

POKE 8246,10:OUT 242,10 (for HDOS, Standard system)

to turn on 3 MHz operation and RUN the program. It should run noticeably faster this time. When you are ready to return to the operating system, enter

POKE 13,34:OUT 242,34 (for CP/M)

POKE 8246,34:OUT 242,34 (for HDOS, Extended system)

POKE 8246,2:OUT 242,2 (for HDOS, Standard system)

to switch back to 2 MHz. Be sure you are in 2 MHz operation before you exit to the operating system.

The above illustrates how to switch speeds from BASIC in CP/M or HDOS. If you want to switch in Cassette BASIC, use OUT 242,8 to go to 3 MHz, and OUT 242,0 to return to 2 MHz in a standard system. In an extended system, use OUT 242,40 to switch to 3 MHz, and OUT 242,32 to switch back to 2 MHz.

You can also switch speeds in an assembly language program in a similar way. For example, in HDOS on an extended system, this code would do the trick.

```
MVI     A,2AH       RAM 0 BIT + SPEED BIT + CLOCK
STA     40066A      SET CONTROL BYTE
OUT     362Q        SWITCH TO 3 MHZ

MVI     A,22H       RAM 0 BIT + CLOCK BIT
STA     40066A      SET CONTROL BYTE
OUT     362Q        SWITCH TO 2 MHZ
```

In CP/M, you can use the same code, but change the argument to STA from 40066A to 0DH. In HDOS on a standard system, use this code.

```
MVI     A,0AH       SPEED BIT + CLOCK BIT
STA     40066A      SET CONTROL BYTE
OUT     362Q        SWITCH TO 3 MHZ

MVI     A,2         CLOCK BIT ONLY
STA     40066A      SET CONTROL BYTE
OUT     362Q        SWITCH TO 2 MHZ
```

For a fuller discussion of how the above examples work, see the REMark #34 article referred to at the beginning of this article.

Modifying Your Operating System

Once you experience the thrill of running at 3 MHz, you probably will want to do it all of the time. This will require modifying your operating system. Specifically, you will have to modify your HDOS disk device driver(s) so that 3 MHz is set when the driver is loaded, and all processor speed dependent delays are fixed. In CP/M, you will also have to fix the delays in the BIOS and modify the cold boot code to make the speed switch.

Modifying HDOS

In HDOS, the ideal place to make the switch to 3 MHz is in the disk device driver, when it is loaded, since critical delay constants have to be changed anyway. The source code for the 5.25 inch hard sector and 8 inch (H47) disk drivers is supplied with HDOS 2.0, so I will present modifications to them.

To fix the 5.25 inch hard sector driver, you will have to modify the files SYDVD.ASM and SYINIT.ASM and re-assemble the driver. First, modify the file SYDVD.ASM as follows. Locate the label SYLOAD, and a few lines below it, add the lines shown here in bold print.

```
SYLOAD  EQU     *

        XRA     A
        OUT     UP.FC           Set Fill character = 0

*       Set up the original vectors

        LHLD    SYDD+1
        PUSH    H       Save current system device
        LXI     B,BOOTAL
        LXI     D,BOOTA
        LXI     H,D.CON
        CALL    $MOVE   Move in constants and vectors
        LXI     B,12
        LXI     D,BOOT3
        LXI     H,D.CON+2
        CALL    $MOVE   Move in 3 MHz constants
        MVI     A,2AH   Use 0AH for non-Extended system
        STA     40066A  Update control byte
        OUT     362Q    Switch to 3 MHz
        POP     H
        SHLD    SYDD+1  Restore system device
```

Now, locate the label TDT and add one line.

```
TDT     DB      DEF.TDT         Track delay time
BOOT3   DB      30,10,24,9,20,15,250,5,7,32,32,160
        DW      0               Dummy Relocation address
```

Modifications to the HUG SY: device driver are similar, but must be done in the file MFDVD.ACM, and the added lines must be preceded by IF MFBOOT and end with ENDIF.

The file SYINIT.ASM must be modified if you want to initialize disks while operating at 3 MHz. Locate the line

```
        MVI     A,H17SDL
```

and replace it with

```
        MVI     A,H17SDL*3/2
```

If you have the HUG SY:, modify MFINIT.ACM by replacing the line

```
        LXI     DE,15*2048/35
```

with

```
        LXI     DE,15*2048/24
```

After you modify SYDVD.ASM and SYINIT.ASM, you will have to assemble them and combine them to get the device driver. Refer to the 4 MHz article in REMark issue 34 for instructions on assembling the driver, combining the parts, and implementing the driver in your system.

NOTE: When you BYE from HDOS while operating at 3 MHz and wish to re-boot, you will have to reset the computer and boot from the front panel. You cannot just press RETURN to re-boot because the boot code on the disk or in the ROM has not been modified for 3 MHz operation, and the computer still operates at that speed until you reset.

To operate H47 disks at 3 MHz, you will need to modify the files DDDVD.ASM and H47LIB.ACM. At the beginning of DDDVD.ASM, add the following line before the first STL statement.

```
*
PRG3MHZ SET     0               ASSEMBLE FOR 3 MHZ
        STL     'Assembly Constants'
```

Locate the label DDLOAD and add the lines shown in bold print.

```
DDLOAD  CALL    RST
        IF      PRG3MHZ
        MVI     A,2AH   Use 0AH for non-Extended system
        STA     40066A  SET CONTROL BYTE
        OUT     362Q    ENABLE 3 MHZ
        ENDIF
        ANA     A       Ignore any errors
```

That completes the modifications to DDDVD.ASM. Near the beginning of H47LIB.ACM, add a line after IF.SMALL.

```
        SPACE   4,10
        IF      .SMALL
PRG3MHZ SET     1               NO 3 MHZ DURING BOOT
        ELSE
```

Locate the lable DLY and make this change.

```
DLY     MVI     A,60Q
        ANA     A               F = 'NC'
```

Now find the label WDN1 and add these lines.

```
WDN1    DCX     B
        IF      PRG3MHZ
        MVI     A,0             WASTE SOME TIME IF 3 MHZ
        MVI     A,0
        ENDIF
        MOV     A,B
```

Finally, find WND1 and add these lines.

```
WND1    DCX     B
        IF      PRG3MHZ
        MVI     A,0             WASTE SOME TIME IF 3 MHZ
        MVI     A,0
        ENDIF
        MOV     A,B
```

After you have made the changes, assemble DDDVD.ASM and DDINIT.ASM and combine the two .ABS files into a new device driver. Unlike the hard sector driver, this driver will still operate at 2 MHz, so you can freely switch speeds while testing software, etc. In fact, the unmodified driver will work most of the time at 3MHz, but make these modifications just to be safe.

Modifying CP/M

All modifications for going to 3 MHz with CP/M are made to the file BIOS.ASM (supplied with CP/M), which must then be re-assembled.

This discussion will cover Heath/Zenith CP/M version 2.2.03, but modifications to 2.2.02 should be similar. Copy BIOS.ASM to a working disk before you modify it, and if it is write protected, remove the protection with STAT.

The first modifications are for 5.25 inch hard sector disks. Locate the line WHDA EQU 20 and modify six of the constants as follows.

```
WHDA    EQU     33
WHNA    EQU     33
WSCA    EQU     120
WRITA   EQU     32
WRITB   EQU     10
WRITC   EQU     32
READA   EQU     75
```

The next modifications are for H47 disk operation. These modifications may be skipped if you are not using an H47. Locate the label W4D1:, and add these lines.

```
W4D1:   CALL    H47INS
        ANI     DSDONE
        JNZ     W4D2
        DCX     B
        MVI     A,0
        MVI     A,0
        MOV     A,B
```

These modifications are for 5.25 inch soft sector disk operation (H/Z37). You may skip them if you are not using that kind of disks. NOTE: These modifications have not been tested, since I do not have a Z89-37 on my modified H89.

Find the label RESH371:, and change the line BEFORE it.

```
        MVI     A,15
RESH371:
        DCR     A
```

Find RDYH37B:, and change the line before it.

```
        MVI     A,15
RDYH37B:
        DCR     A
```

Locate the lable WBS37:, and add 2 lines below it.

```
WBS37:  MVI     A,150
WBS371: DCR     A
        NOP
        NOP
        JNZ     WBS371
```

The next change must be done regardless of what kind of disks you are using. It causes the computer to switch to 3 MHz operation at cold boot. Locate the label CBOOT:, find the line LXI H,CTLPRT a few lines below it, and add two lines.

```
        LXI     H,CTLPRT
        MOV     A,M
        ORI     8
        MOV     M,A
        OUT     H88CTL
```

This completes the modifications to the BIOS. It must be re-assembled using the MAKEBIOS program supplied with CP/M. If the procedure given in the Heath/Zenith manuals is too difficult to understand, you may want to try the method I outlined in the article "Making Sense of MAKEBIOS" in REMark #26 (March 1982). There are two typographical errors in that article in the listing of MAKE.SUB in the left column on page 14. Where it says $R/W, change it to $$R/W, and where it says $DIR, change it to $$DIR.

After you assemble the new BIOS, you will need to install it on a system disk. You should first prepare a test system disk (with FORMAT and SYSGEN) and copy STAT.COM, MOVCPMnn.COM (that is, MOVCPM17 for hard sector disks, MOVCPM47 for H47, etc.), PIP.COM, CONFIGUR.COM, and STAT.COM to it. Then delete the old BIOS from it with these commands.

```
A>STAT BIOS.SYS $R/W
A>ERA BIOS.SYS
```

Now, copy the new BIOS.SYS to this disk and enter

```
A>MOVCPMnn * A:
```

replacing nn with 17, 47, or 37 as appropriate. When MOVCPM is finished, run SYSGEN and hit RETURN when it asks for a source disk, and type A when it asks for a destination disk. Then reset your computer and re-boot on the test disk. CONFIGUR will run if everything is working, and you can set up things as you wish and make other system disks from the new one.

If you want to format disks under CP/M while running at 3 MHz, you will have to modify the FORMAT program. To do it, use the S command in DDT to make the following changes to FORMAT.COM.

Address	Old data	New data
103	43	CD
104	6F	CF
105	70	06
106	79	C3
107	72	1B
108	69	04
418	CD	C3
419	CF	03
41A	06	01
4DE	01	02
4FB	01	02
5AA	01	02
607	30	48
613	50	A0
75E	14	1E
76B	14	1E
876	0A	0F

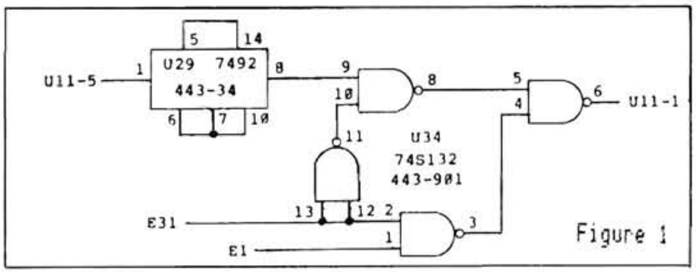


Figure 1

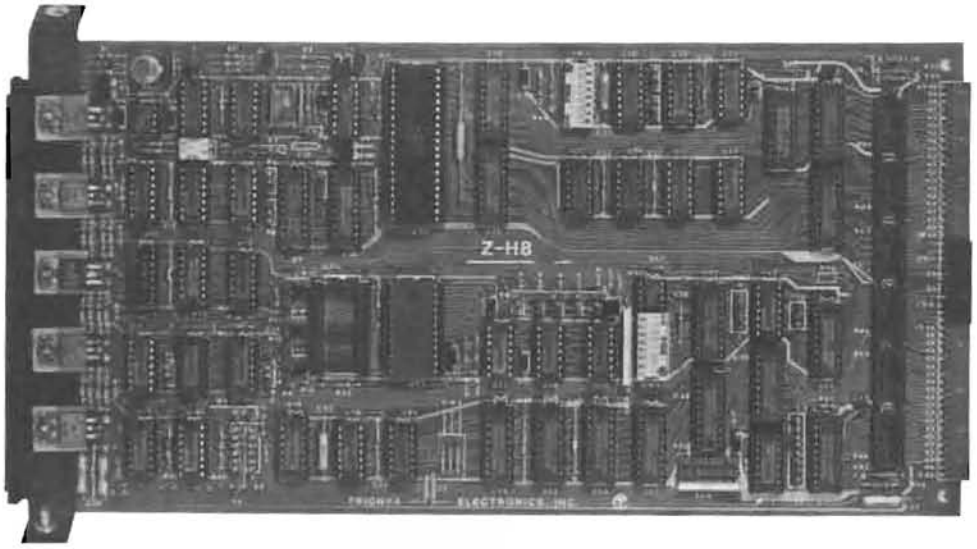
These modifications affect delays for all three disk formats that I have presented BIOS modifications for. The soft sector 5.25 inch part has not been tested.

A Final Note

Once you have modified your operating systems, you should experiment with test disks, running several different programs, until you are satisfied that everything works. You will notice different effects on different programs. For example, an action game that uses the computer's 2 ms clock to time its actions (such as Sea Battle — 885-1103) will run at the same speed, while a game that is processor speed dependent (such as Galactic Warrior — 885-8009) will run faster. Most programs will run much faster, and it's almost like getting a new computer for the price of a few IC's, so enjoy it.

Z80 CPU BOARD for the H8*

* H8 is a Registered Trademark of the Heath Company



Model Z-H8

$ 300.00 Assembled **$ 250.00 Kit**

Programmable Clock Rate — 2 MHz or 4 MHz Operation

Clock Rate under Software Control — May Be Changed at Any Time

Selection of Wait States (None, 1/2, 1 or 2) at 4 MHz

Buss Termination Network • 51 Integrated Circuits • Transistor Power Supply

Includes Heath Extended Configuration (ORG ZERO) Circuits

Exclusive Z80 Front Panel Monitor ROM — Based Upon Heath XCON-8 ROM

Exclusive Software Speed Utilities (For Both HDOS* and CP/M) on Diskette**

Fully Compatable With All Heath H8* Hardware and Software

* H8 and HDOS are Registered Trademarks of the Heath Company.
** CP/M is a Registered Trademark of Digital Research.

Check • Money Order • VISA • MASTERCARD • C.O.D.
Phone Orders Welcome (714) 830-2092 — Send For Free Brochure

TRIONYX ELECTRONICS, INC.
P.O. BOX 5131, SANTA ANA, CA 92704

Computer Aided Instruction, Part 1

Walt Gillespie
REMark Editor

As I indicated in my December 1982 editorial we will begin to address the subject of Computer Aided Instruction (CAI). Over the next few months I plan to layout the framework of how to develop CAI programs. I will try to point out pit-falls and, hopefully, how to enhance your project so it will make the best impact as a teaching aid.

The field of CAI is vast, possibly more complex than any other area concerning computers. Many different things must be brought together to make an effective CAI project. I use the word project, rather than program, because CAI can be more than just programs. CAI can entail the use of not only computer programs but also printed materials or audio visual aids, possibly the combination of two or all of these to best explain and teach. As with any project, a CAI project must be well planned to succeed.

Throughout this series I will use as a reference source a booklet entitled "A Guide To Developing Instructional Software For The APPLE II Microcomputer" published by the Minnesota Educational Computing Consortium (2520 Broadway Drive, St. Paul, MN 55113 — MECC Publication #M(AP)-2). I feel this booklet is very informative and well written. We have been given permission for it's use as a reference source.

As I indicated earlier you must plan ahead. Your first choice, and probably the hardest, will be that of a subject. A working knowledge of the chosen subject is not always necessary if you have a good source of collaboration, possibly informational books, text books or an expert in your chosen field that you can check with to make sure your materials are correct. I stress this check of correctness now because no matter how well you might prepare your programs or other materials they will loose all effectiveness if they contain errors concerning the subject you are trying to teach.

Virtually, anything can be made the subject for a CAI project, just about any standard course taught in our public schools, English, Math, Science, Music, etc. Also specialty areas can be used for CAI projects, from TV repair to airplane piloting, from house construction to gardening. Many of these subjects can be broken down into sub-topics as in the case of, say, computer programming. You have many different languages and each of these can be splintered still further into specific applications and the use of special features of these languages. With gardening you might have vegtables, exotic flowers, landscaping or even the growing of Bonsi trees. The possibilities are limited only by your imagination. Just remember that there might not be a whole lot of people out there who are interested in Growing Mosquitos for Fun and Profit, even if you have prepared the best materials ever on the subject.

Once the decision of subject has been made (I didn't say CAI was easy), you can move into the formal planning stages. Below is a brief planning outline.

1. Analysis
   a. Applicability for a microcomputer
   b. Equipment restrictions
   c. Modes of interaction
   d. Necessary support materials
2. Design Task
   a. Display layout (Screen Layout)
   b. Movement between screens
   c. Requesting input
   d. Menus
   e. Hard Copy
3. Choosing a Language
   a. speed
   b. knowledge
   c. portability
   d. acceptability
4. Keeping a Clean House
   a. processing user input
   b. error trapping
   c. storage of data
   d. output of results
5. Fine Tuning
   a. test running
   b. testing unexpected input
   c. critical user test
   d. naive user test
   e. final checklist

Part 1. Analysis

This is a very critical part of developing any CIA project, because, a major error in judgement here can mean wasted hours and an unusable project later.

After choosing your subject make a list of both pro and con arguments as to its merits as a microcomputer CAI project, be objective as you review this list. Also make a list of what equipment will be needed by the student in use of your materials. Remember, a lot of people do not have the funds to purchase most peripherals available and if you choose a subject that will use a great many, or a very special piece, you will be limiting the market.

"Deciding whether a particular idea you have is appropriate for a microcomputer is very difficult for people to handle. An example of inappropriate use of the computer is to use it as a 'page turner'. Presenting large quantities of material in the form of screen full after screen full of printing might be presented more effectively, if simply put in a printed form. It is probably cheaper this way too," so says the MECC.

The MECC also adds; "A major mistake made by many designers of computer materials is that of expecting much more than their microcomputer is capable of delivering."

Also keep in mind the cost and availability of any support materials that you may be requiring be used with your project. An example would be; if you supplied a set of slides will they fit equipment generally available.

I will address Part 2 The Design Task and Part 3 Choosing a Language in the next issue, there is enough contained here to keep you awake many a night.

You Just Purchased BASCOM?

Using the CP/M Basic Compiler

Thomas E. Wiles
72 Peru Street, Apt. 6
Plattsburgh, NY 12901

You have been programming in BASIC for some years now, during this period you have been impressed by the speed and transportability of several of the compiled languages; so you assumed that the best place to start learning about compiled languages was to start with a BASIC compiler! You purchased this beautifully advertised package thru ZENITH, diligently waded through the documentation (a no small feat), wrote a very simple BASIC program to test this new toy only to find that it won't run. A quick check through the HEATH BASIC programming courses; nothing about compilers. A recheck of the documentation (only about two pages of the entire text actually refers to the compiler itself) doesn't solve the problem, now what?

The Microsoft BASCOM documentation was evidently written with the assumption that the user would already have a very complete knowledge of compilers in general and the Microsoft FORTRAN compiler in particular. The great majority of the documentation concerns "special treatments" of BASCOM and its interface with assembly language programs. If you wanted an assembler you would have purchased an assembler not a compiler, right?

For the programmer who is programming in BASIC only three files from the source disk are needed for his applications: BASCOM.COM, BASLIB.REL, and L80.COM. All other files may be deleted from the working disk. The reason for this is that there are only two types of relocatable files (.REL) that the linker (L80.COM) can deal with. These are called "MAIN" programs and "SUBPROGRAMS". The BASCOM compiler only compiles "MAIN" programs. The linker (L80.COM) can link one "MAIN" program to any number of "SUBPROGRAMS" but since BASCOM can not write a "SUBPROGRAM" (an example of linking a FORTRAN subprogram will be shown later) there is no need for the library manager (LIB.COM). (An assembly language program assembled under M80.COM which meets the "SUBPROGRAM" call protocol can be linked, but if you are skilled enough to do this you have already proceeded to the next article.)

What programs will BASCOM compile? Almost any BASIC program written under Microsoft BASIC 4.7 or 5.1 should compile. I personally recommend running all programs under BASIC 5.1 prior to compilation but this is not absolute. Some of the extensions added by Microsoft to the BASIC language will not compile. These words are listed on page 17 of the MBASIC reference manual and only two really apply. If you are currently using overlays within your program thru the MERGE command the program will not compile. Any program with COMMON variables or which is being CHAINed to another program (with one exception) will not compile and run correctly. BASCOM will CHAIN one program to another but will not pass variables. BASCOM closes all files and completely exits from the first program prior to calling the new program. An example of this type of CHAIN would be a menu which calls five programs (say games). The menu program runs first and you select a game. The menu programs CHAIN's to the game program and when you are finished with the game, the game program CHAIN's back to the menu program.

One note of warning, a program compiled under BASCOM (even though it runs correctly) may not produce the same results as the same program running under the interpreter. If you write correct code, you should not run into this problem but if you are lazy like me and are careless in your use of data types it can result in a little hairpulling. BASCOM is more restrictive in the use of data types for instance:

```
DEFINT I-N
A = 5.4545
I = A
PRINT I
```

Under the interpreter A will be rounded to the nearest integer so the printed result will be 5. The compiler seems to skip the statement entirely and goes merrily along its way leaving 0 in I (if I had not been previously assigned). This can result in some interesting results. I = CINT(A) or A = CINT(A) will execute correctly and should have been used in the first place.

OK enough! Lets get to the compiler!! Before we can compile a program we need one to compile so one has been supplied.

```
10 REM *** THIS PROGRAM DEMONSTRATES
                        CERTAIN FEATURES OF BASCOM. ***
20 REM *** THIS PROGRAM COMPUTES THE AREA OF A CIRCLE
30 INPUT "RADIUS OF THE CIRCLE? ";R
40 A = 3.14159 * R ^ 2
50 PRINT A
60 PRINT
70 PRINT "CONTINUE Y/N ";
80 Y$=INPUT$(1)
85 PRINT
90 IF Y$= "Y" OR Y$ = "y" THEN 30
100 STOP
110 END
```

Line 100 will print an end execution message via BASCOM. If no end execution message is desired replace STOP with SYSTEM prior to compilation.

1. Execute MBASIC and type the above program in.

2. Run the program then test for proper execution.
(As long as you have an interpreter, all debugging should be done

under the interpreter rather than trying to trace thru the compiled program.)

3. Save the program i.e. (SAVE "TRIAL1.BAS",A)

(NOTE: THE COMPILER ONLY UNDERSTANDS ASCII, IT KNOWS NOTHING ABOUT COMPRESSED BINARY MBASIC FILES. IF YOU TRY TO COMPILE THE PROGRAM WITHOUT SAVING IT IN ASCII YOU WILL GET A MESSAGE TO THE EFFECT THAT THE COMPILER GAVE UP AFTER ENCOUNTERING 14 FATAL ERRORS.STRANGE NOISES PLUS A STRING OF "N"'S DOWN THE RIGHT HAND SIDE OF THE SCREEN GENERALLY ACCOMPANY THE ERROR MESSAGE.)

4. PIP the saved program over to the disk which contains BASCOM.COM, BASLIB.REL, and L80.COM.

5. B>BASCOM<CR>
The compiler gives you a * for a prompt.

6. *=TRIAL1<CR>
(That is all you have to do. The compiler will give you a file called TRIAL1.REL and exit.

7. B>L80<CR>
(The linker will give you a * for a prompt.)

8. *TRIAL1,TRIAL1/N/E<CR>
(TRIAL1, tells the linker you want to link TRIAL1 to something: TRIAL1/N tells the linker you want a file written to disk labeled TRIAL1.COM: /E tells the linker you want TRIAL1 linked with BASLIB.REL (very necessary) then tells the linker you are finished and please go away.

9. B>TRIAL1<CR>
(Your program should now run exactly as it did under the interpreter.)

LINKER (L80.COM)

Easy enough? Well what about all the switches? Basically most of them do not apply. /N and /E we have already mentioned and should be used together as shown. /G can be used but I recommend running the program directly from the operating system. /P and /D do not concern the BASIC programmer. They can be used to relocate the program into a particular block of memory and are used for special problems. /U and /S will be demonstrated later. /X creates a file with a .HEX extension. This is a load module which can be loaded into memory and saved under the CP/M operating system. Your CP/M documentation is adequate to explain the use of this option and is of no real concern to the BASIC programmer since the /E switch will get you to the same place without all the hassle. /R starts all over again in the event you made a mistake. /D tells you how you are coming along with the linking of the "MAIN" program to all the "SUBPROGRAMS". If you are only working with BASIC and BASCOM you are not going to have any undefined global calls and therefore should never need to do a /D (if you can't stand the answer don't ask the question).

COMPILER (BASCOM.COM)

The compiler also has some switches that may apply. If you have written error recovery routines into your program, you must include the appropriate switch, either /E or /X. /Z, /N, /S, /D, /C you can forget about. /D has some application but if the program has run correctly under the interpreter, there shouldn't be a problem under the compiler. (Remember, no dynamic array assignments when using the compiler, DIM A(N) will not compile and will give an error message). /4 tells the compiler to use OBASIC (i.e. BASIC 4.83) and I think it is best to rewrite these programs so that they will run under the BASIC 5.1 interpreter.

What does the compiled program look like? Well, basically it consists of two parts; the run-time module (about 10K) and the compiled program. If you take the amount of free memory it takes the interpreter to run the program in kilobytes and add 10K to this value you will have an approximate length. The run-time module which is linked to the "MAIN" program handles some error recovery for you, prints some error messages when run-time errors occur, and does some helpful housekeeping chores to make programming simpler for the programmer. If you ran the above program with the "END" statement in the program, the exit message printed on the screen was provided by the run-time program.

The linker will link a program which has a total length of about 45K on a 64K machine. This means that a basic program which resides in 38K should compile and link. Since under the interpreter only about 34K is available to the program, any program which runs under the interpreter should compile and run. Because the compiler does not dynamically allocate string space it never runs out of string space like the interpreter does. If you are annoyed when you program stops execution for 25-30 seconds while the interpreter reallocates all its string space so it can accept more input, just compile the program.

The remainder of this article discusses the linkage of a FORTRAN "Subprogram" to a MBASIC compiled program. Where the discussion which follows does not apply to the programmer who is writing in BASIC only, it might provide some useful insight into the use of the compiler.

How do you use a "Subprogram" CALL? What we are going to do is replace line 40 in the above BASIC program with a FORTRAN CALL. The new BASIC program is shown below.

```
10 REM *** THIS PROGRAM DEMONSTRATES
                              CERTAIN FEATURES OF BASCOM. ***
20 REM *** THIS PROGRAM COMPUTES THE AREA OF A CIRCLE
25 REM *** A = AREA  :  R = RADIUS
30 INPUT "RADIUS OF THE CIRCLE? ";R
35 A=0
40 CALL CIRCLE(R,A)
50 PRINT A
60 PRINT
70 PRINT "CONTINUE Y/N ";
80 Y$=INPUT$(1)
85 PRINT
90 IF Y$= "Y" OR Y$ = "y" THEN 30
100 STOP
110 END
```

NOTE: Line 35.It is not necessary to assign a value to A prior to calling a "SUBPROGRAM" but it is a good habit to get into and I always do it. When you call a "SUBPROGRAM" the compiler passes all the variables so I like to know what is being passed. Both R and A are passed to the "SUBPROGRAM" from the "MAIN" program and both R and A will be passed back to the "MAIN" program from the "SUBPROGRAM". One note of caution, the data type of every variable passed must be assigned in both programs and the assignment must be the same. Remember, FORTRAN automatically assigns the data type of INTEGER to variables starting with the letter I thru N. Since the BASIC fault assignment is single-precision for the same variables the programmer must be careful.

O.K. so far? After the alterations have been made to TRIAL1.BAS to get the above program we will save it as TRIAL2.BAS

SAVE "TRIAL2.BAS",A<CR>

Now PIP TRIAL2.BAS over to the disk with BASCOM.COM, BASLIB.REL, and L80.COM.

Compile TRIAL2.BAS

B>BASCOM =TRIAL2.BAS

Note that I did not enter BASCOM prior to typing in the command to compile the program. Like most CP/M programs the instructions to the called program may be included on the same line that calls the program! Either method of executing the compiler can be used and each method works equally well.

If you check your disk you should now find the program TRIAL2.REL on the disk.

The next step is to write and compile the FORTRAN SUBPROGRAM. Below is the source file for this example.

```
      SUBROUTINE CIRCLE(X,Y)
C     ***X= RADIUS OF CIRCLE, Y= AREA OF CIRCLE
      PI = 3.141593
      IF(X) 20, 20, 10
10    Y = PI * X ** 2
      RETURN
20    Y = 0
      RETURN
      END
```

If you have access to the Microsoft FORTRAN compiler and wish to link this module, the text may be typed in on any editor (I use PIE from the SOFTWARE TOOLWORKS). For the readers who don't know FORTRAN the C and the numbers 10 and 20 start at the left margin. All the other letters start in column 7.

To compile the above program you need the FORTRAN compiler F80.COM. I recommend the above source file also be placed on this disk and be named CIRCLE.FOR. The .FOR is mandatory.

C>F80 =CIRCLE

Now we want the following programs on the same disk: L80.COM, FORLIB.COM, BASLIB.COM. For convenience it is nice to have TRIAL2.REL and CIRCLE.REL also on the disk with the linker and the libraries. The entire program can be linked with a one-line command but I am going to go thru a step by step link to demonstrate each function as it is done.

>L80 TRIAL2,TRIAL2/N<CR>
(Loads TRIAL2 into the linker and opens a disk file titled TRIAL2.COM)

Data   0103   01EE   < 235>   (printed by linker)

103 = start of program (HEX) (CP/M uses 0100 to 0102 as the jump to execution instruction: this is the start of program space and will always be the same.)

01EE = last byte currently in program (address)

< 235> = number of bytes currently loaded into linked program.

BASLIB REQUEST (tells you that the linker recognizes a request for global calls in the BASLIB)

```
-$5.0   0134    -$END  01B1    -$EQSA  019C
-$FSFA  0149    -$IN$  017A    -$INOA  013A   *
-$INI   0123    -$IPUA 013E    -$IPUB  0146
(ETCETERA) ---  -CIRCLE 0156
   16 Undefined Globals (s)     (this is the total number)
46025 Bytes Free     (amount of space left for the linker to
                     put the calls)

                              * (these are the global calls)
```

Notice that the last global "CALL" is CIRCLE. This CIRCLE is our FORTRAN subprogram. It is a good idea to link all of our work prior to linking the libraries so that is what we will do next.

```
*B:CIRCLE<CR>   (*CIRCLE if everything is on the same drive)
Data    0103    023F    < 316>

 BASLIB REQUEST         FORLIB REQUEST      (it now wants to
                                            see both libraries)
-$5.0   0134    -$CA    0232    -$EA    021F
-$END   01B1  (ETCETERA)
  20 Undefined Global (s)     (The FORTRAN and BASIC global
45920 Bytes Free              calls are now mixed together,
                              no problem)
```

Now the procedure could be finished by a /E<CR> but for demonstration purposes we will load the libraries individually.

*BASLIB/S<CR> (the /S tells the linker to search the library, NOT load the library. If you omit the /S the program will run but will sit in about 30K instead of about 10K)

To get a readout of the information that the linker previously gave for free you must use a /U.

```
*/U
Data    103     289E    <10139>

 BASLIB REQUEST         FORLIB REQUEST

                              (Even though the linker has
-$EA    021F                  searched BASLIB it still
   1 Undefined Global (s)     wants to see it again)
32496 Bytes Free
```

Notice that the FORTRAN SUBPROGRAM added five undefined global calls and now there is only one left. The two libraries have most calls in common but the linker is still going to want to look at both libraries.

```
*FORLIB/S/U
Data 103        2A25    <10530>

 BASLIB REQUEST         FORLIB REQUEST
32037 Bytes Free
```

Well we are finished and the program is in final form but the linker still wants to process both libraries. This is why it is best to use the auto library search feature by using a /E instead of searching each library individually.

```
*/E

DATA 103     2A25  <10503> (It didn't find anything did it?)

 BASLIB REQUEST          FORLIB REQUEST
32037 Bytes Free       (nice to know if you intend to rewrite
                       or add to the program in the future)
[011F   2A25    42]
```

011F = The entry point (start execution address) of the program. You do not need to know this since it has already been passed to CP/M. i.e. a jump instruction is loaded into 100H thru 102H (now you know why the program load started in 103H

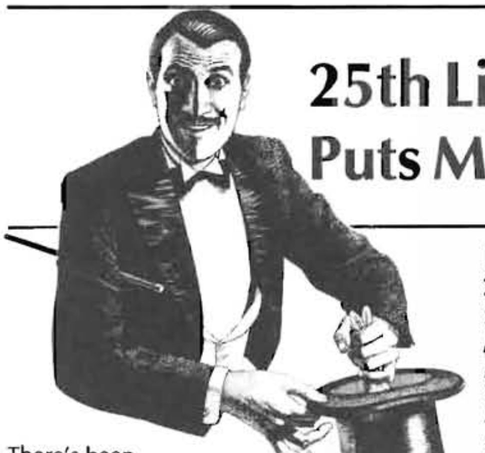
2A25 = last byte used by the program

42 = used when saving the program with the CP/M "SAVE" command. (42 pages) SAVE 42 TRIAL2.COM The linker does this for you so you should not need to use this number but it is nice to know.

>TRIAL2<CR> (The compiled program should run the same as the BASIC program under the interpreter)

Well there is very little more to say. When I first got the Microsoft FORTRAN compiler up and running, I was about six months learning to use it. At many points the frustration level became very high but the compiler is very powerful and not difficult to use once you become familiar with it. Many compilers do all their linking automatically and are much simpler to use. These compilers are (in general) less powerful (fewer linking options: the ability to link BASIC, FORTRAN, and COBOL modules together is unique to my experience) but easier to learn.

I hope I have shed a little light on a dark subject and I hope you will find your compiler a little easier to use.



25th Line Subroutine Writer Puts Magic Into Your H-89

Tommy Towery
2610 Gindy Drive
Omaha, NE 68147

There's been a lot of articles lately concerning the use of the function keys of the H-89. Most programs which use the f-keys use a 25th line label to keep tab of their functions; but, the 25th line is as much a mystery to many as the function keys themselves! More often than not, when we decide to use the function keys, we have to look up escape codes and pull out the H-89 manual to see how to access and write to the 25th line in order to label them as we wish.

Wouldn't it be nice if you didn't have to write all those lines needed to build the 25th line labels in your program? If a computer is so smart, why can't it write its own program?

Well, sit back and relax, now you computer will literally write its own 25th line subroutine, in Microsoft Basic and put it on your program disk. All you do is answer the questions asked and it will write the subroutine for you to merge with your main program.

The 25th Line Subroutine Writer is a Microsoft Basic program designed to allow you to easily build a subroutine to label the f-keys of your H-89. It starts by asking you what label you desire above each of the keys. Labels are limited to four or less symbols and can be in reverse or normal video. If you do not desire a label above any of the keys you

```
1 'PROGRAM TO WRITE 25th LINE SUBROUTINE
2 'CREATED BY TOMMY TOWERY, COPYRIGHT (c) 1982
3 '        THANKS TO MJP
4  CLEAR 500
5  E$=CHR$(27):R$=E$+"p":N$=E$+"q"
10 PRINT E$"E":FOR T=1 TO 50:NEXT T:PRINTTAB(25)"25th LINE SUBROUTINE WRITER"
11 :PRINT:PRINT:PRINT
15 DIM L$(9)
60 FOR X=1 TO 9
61 READ F$:DATA F1,F2,F3,F4,F5,ERASE,BLUE,RED,WHITE
65 IF X>9 THEN GOTO 500
70 PRINT"WHAT LABEL DO YOU WANT ABOVE THE ";F$;" KEY";:INPUT"";L$(X)
75 IF LEN(L$(X))>4 THEN PRINT"   YOU MUST USE FOUR LETTERS OR LESS!":GOTO 70
76 IF LEN(L$(X))<4 THEN L$(X)=L$(X)+" ":GOTO 76
78 A$(X)="N"
80 INPUT"DO YOU WANT IT IN NORMAL OR REVERSE VIDEO (N OR R) <CR FOR N> ";A$(X)
82 IF A$(X)="R"THEN L$(X)=R$+L$(X)+N$:GOTO210
83 IF A$(X)<>"N"THEN 78
90 IF A$(X)="N" THEN 210
210 NEXT X
500 PRINTE$+"x1":PRINTE$+"Y8%":FOR X=0 TO 9:PRINT "   ";L$(X);:NEXT X:PRINTE$+"H"
510 PRINT E$+"E":FOR Y=1 TO 200:NEXT Y
550 PRINT:PRINT:INPUT"IS THIS CORRECT (Y or N)";A$
555 IF A$="Y"THEN 680
560 IF A$<>"N"THEN 550
570 PRINT"HIT THE KEY YOU WISH TO CHANGE":F$=INPUT$(2)
580 FK$=MID$(F$,2,1)
590 X=0
600 IF FK$="S"THEN X=1
601 IF FK$="T"THEN X=2
602 IF FK$="U"THEN X=3
603 IF FK$="V"THEN X=4
605 IF FK$="W"THEN X=5
606 IF FK$="J"THEN X=6
607 IF FK$="P"THEN X=7
```

Vectored to ☞

simply leave those label blank and in normal video. After you answer these questions about the nine function keys, you will be shown a sample of what the 25th Line subroutine will look like. You are then asked if what you see is what you really wanted. If one or more of the labels are not correct, you are allowed to selectively give keys new labels or change video modes. You do this by answering "No" to the question: "Is this correct?". You are then instructed to press the key whose label you wish to change. At that time you can relabel it. You are allowed to change labels until you indicate that the display is correct.

Then the magic begins. You are asked for the drive for the output program (ie. SY0:). You next give the subroutine a starting line number, and the subroutine will be written to that disk as a program titled "LINE25.SUB". Be careful to start the subroutine at a number that will not interfere with the program with which you wish to merge it. Any line numbers in the merged program will replace those same line numbers in the main program. Once that is completed you can simply load your main program, use the MERGE "LINE25.SUB" and appropriate drive number to complete the procedure. Now you add a simple GOSUB statement and there are your 25th line lables. Simple as can be. To disable the 25th line just add an CHR$(27)+"y1" in your program and it will disappear. It can be recalled anytime with the GOSUB statement.

The secret of the program lies in lines 680-800. It opens an output file named "LINE25.SUB". The default subroutine is started at line number 10000 and increased by 10 each line ending at line number 10150. String values for the escape codes are sent to the subroutine. Your labels are also stored as string values. Line 762 checks to see if you wanted reverse video, and if you did then 763 strips the escape codes from your label string, with new string values for the reverse video added at line 765. Normal video lables are built in line 770 and the subroutine line number added to the label. Line 775 prints out each line number and label string to the output file in a PRINT #1, statement. Instructions to HOME the cursor and a RETURN statement is then added and the file is CLOSED.

The possibilities of adapting this type of program to other uses is great, but even if you don't, you'll have to look hard to find an easier way to write the 25th line labels you want for your programs.

```
608 IF FK$="Q"THEN X=8
609 IF FK$="R"THEN X=9
610 L$(X)="    ":PRINT"WHAT DO YOU WANT THE NEW LABEL TO SAY";:INPUT"";L$(X)
620 IF LEN(L$(X))>4 THEN PRINT"    YOU MUST USE FOUR LETTERS OR LESS!":GOTO 610
630 IF LEN(L$(X))<4 THEN L$(X)=L$(X)+" ":GOTO630
638 A$="N"
640 INPUT"DO YOU WANT IT IN NORMAL OR REVERSE VIDEO (N or R) <CR FOR N> ";A$
650 IF A$="R"THEN L$(X)=R$+L$(X)+N$:GOTO500
660 IF A$<>"N"THEN 638
670 GOTO 500
680 D$="SY0:":INPUT"ON WHICH DRIVE DO YOU WISH
                                   THE PROGRAM WRITTEN <CR FOR SY0:>";D$
685 IF D$="SY1:"THEN 700
686 IF D$="SY2:"THEN 700
687 IF D$<>"SY0:" THEN 680
700 OPEN"O",1,D$+"LINE25.SUB"
703 X=0:L=10000
705 INPUT"AT WHAT LINE NUMBER DO YOU WISH
                           THE SUB ROUTINE TO START <CR FOR 10000>";L
715 LIN$=STR$(L)+" '25th Line Subroutine By TOMMY TOWERY"
716 PRINT#1,LIN$
720 L=L+10:LIN$=STR$(L)+" E$=CHR$(27):R$=E$+"+CHR$(34)+"p"+CHR$(34)+":
                                        N$=E$+"+CHR$(34)+"q"+CHR$(34)
725 PRINT#1, LIN$
730 L=L+10
740 LIN$=STR$(L)+" PRINT E$+"+CHR$(34)+"x1"+CHR$(34)
745 PRINT #1, LIN$
750 L=L+10:LIN$=STR$(L)+" PRINT E$+"+CHR$(34)+"Y8%"+CHR$(34)
755 PRINT #1, LIN$
756 L=L+10:LIN$=STR$(L)+" PRINT"+CHR$(34)+"   "+CHR$(34)+";"
757 PRINT#1,LIN$
760 FOR X1=1 TO 9:L=L+10:X=X+1
762 RV$=LEFT$(L$(X),2):IF RV$<>R$ THEN 770
763 L$(X)=MID$(L$(X),3,4)
765 LIN$=STR$(L)+" PRINT "+CHR$(34)+"   "+CHR$(34)+";
                                  R$"+CHR$(34)+L$(X)+CHR$(34)+"N$;"
766 GOTO 775
770 LIN$=STR$(L)+" PRINT "+CHR$(34)+"   "+CHR$(34)+
                                   ";"+CHR$(34)+L$(X)+CHR$(34)+";"
775 PRINT #1, LIN$
780 NEXT X1
782 L=L+10:LIN$=STR$(L)+
     " PRINTE$+"+CHR$(34)+"H"+CHR$(34)
783 PRINT#1,LIN$
790 L=L+10:LIN$=STR$(L)+" RETURN"
795 PRINT #1, LIN$
800 CLOSE#1
810 END
```

```
10000 '25th Line Subroutine
      By TOMMY TOWERY
10010 E$=CHR$(27):R$=E$+"p":N$=E$+"q"
10020 PRINT E$+"x1"
10030 PRINT E$+"Y8%"
10040 PRINT"   ";
10050 PRINT "   ";R$"F1  "N$;
10060 PRINT "   ";"F2  ";
10070 PRINT "   ";R$"F3  "N$;
10080 PRINT "   ";"F4  ";
10090 PRINT "   ";R$"F5  "N$;
10100 PRINT "   ";"ERAS";
10110 PRINT "   ";R$"BLUE"N$;
10120 PRINT "   ";"RED ";
10130 PRINT "   ";R$"WHIT"N$;
10140 PRINTE$+"H"
10150 RETURN
```

Z-100 Z-DOS Swap Program

Marc O. Aagenas
Software Consultant
Zenith Data Systems

If you have a H/Z 89-90 computer and a H/Z-19 terminal you may already be using my CP/M swap program to access your machine from different locations or users. But if you have a H/Z- 100 only CP/M could be used with it. Well, I didn't want the Z- DOS users to feel left out so here is the Z-DOS SWAP program.

The concept behind the Z-DOS version is the same as the CP/M version; the logical CONsole and AUX:(TTY: in CP/M) are swapped. The implementation, however, is quite different.

CP/M ,through the I/O BYTE, allows easy re-direction of the logical devices. Z-DOS, on the other hand, has no like facility. But, because of it's device I/O structure, Z-DOS has much more flexible I/O capabilities.

This program does not use many Z-DOS function calls, it uses BIOS calls instead. Therefore, it is important that the INCLUDE files needed to assemble this program are current for the version of Z-DOS you are using. Otherwise the addresses gleaned from these INCLUDE files might be incorrect.

Further information on the BIOS calls used can be gotten from the Z-DOS Manual Vol. II Appendix I pages 18-24 and pages 38-43.

To assemble this program you need a copy of Z-DOS disk II in drive B: and a disk with MASM, LINK and this program in A:. Assemble it as follows:

A:MASM SWAP,SWAP,SWAP;(RETURN)

Then LINK it as follows:

A:LINK SWAP,SWAP;(RETURN)

Configure your AUX: port for user defined protocol and select the options and baud rate you want.

Hook up your alternate terminal(H/Z-19) and run SWAP.

Make sure your cable has a jumper from pin 5 to pin 6 on the computer's side and you should be all set.

```
        TITLE   Z-100 SWAP PROGRAM
        PAGE ,132

        .XLIST
        INCLUDE B:DEFMS.ASM
        INCLUDE B:DEFCHR.ASM

        .LIST
;
;******************   PROGRAM SETUP   *********************
;
;
STKSEG  SEGMENT STACK
        DB      100H DUP(?)
STKSEG  ENDS
;
DATSEG  SEGMENT
;
RT_ADDR DD      0
;
CON_CHRD        DB 10H DUP(0)           ;CONSOLE TABLE SWAP AREA
AUX_CHRD        DB 10H DUP(0)           ;AUX DEVICE TABLE SWAP AREA
;
MESSAGE1        DB 027,'E CONsole is available !',07,'$'
MESSAGE2        DB 027,'E',027,'Y',43,64,'System busy !',027,'H$'
MESSAGE3        DB 027,'E',07,0DH,0AH,'$'
                DB 'Copyright 1983 by Marc O. Aagenas'
;
DATSEG  ENDS
;
;
PGMSEG  SEGMENT
;
        ASSUME CS:PGMSEG,SS:STKSEG,DS:DATSEG,ES:NOTHING
;
HERE:
        MOV     AX,DATSEG               ;  SET UP DATA SEGMENT
        MOV     DS,AX                   ;  REGISTER
        MOV     WORD PTR RT_ADDR+2,ES   ;  SET UP EXIT ADDRESS
        MOV     ES,AX                   ;  COPY DS TO ES
        JMP     MAIN_LOOP
        PAGE
;
;****************     CALLED PROCEDURES   ******************
```

```
RD_CHRDS        PROC NEAR       ; READ IN DEVICE TABLES (AND SWAP THEM)
;
        MOV     AH,CHR_STATUS           ; THIS IS A STATUS OPERATION
        MOV     AL,CHR_SFGC             ; LOAD READ TABLE COMMAND
        PUSH    AX                      ; SAVE COPY OF COMMAND
        MOV     BX,OFFSET AUX_CHRD      ; LOAD TABLE POINTER
        CALL    BIOS_CONFUNC            ; EXECUTE FUNCTION
        POP     AX                      ; RETRIEVE COMMAND
        MOV     BX,OFFSET CON_CHRD      ; READ OTHER TABLE
        CALL    BIOS_AUXFUNC
        RET
RD_CHRDS        ENDP
;
WR_CHRDS        PROC NEAR       ; WRITE TABLES OUT
;
        MOV     AH,CHR_CONTROL          ; THIS IS A CONTROL OPERATION
        MOV     AL,CHR_CFSU             ; LOAD WRITE TABLE COMMAND
        PUSH    AX                      ; SAVE IT
        MOV     BX,OFFSET CON_CHRD      ; LOAD POINTER
        CALL    BIOS_CONFUNC            ; EXECUTE
        POP     AX                      ; RETRIEVE COMMAND
        MOV     BX,OFFSET AUX_CHRD      ; WRITE OTHER TABLE
        CALL    BIOS_AUXFUNC
        RET
WR_CHRDS        ENDP
;
MESS    PROC NEAR               ; PRINT MESSAGE
;
        MOV     AH,DOSF_OUTSTR
        INT     21H
        RET
MESS    ENDP
;
READ_AUX PROC NEAR              ; TEST AUX DEVICE FOR A CHARACTER
;
        MOV     AH,CHR_READ
        CALL    BIOS_AUXFUNC
        JC      BACK            ; NO CHARACTER THEN RETURN
        CMP     AL,0DH          ; TEST FOR CARRIAGE RETURN
        JNE     BACK
        POP     AX              ; CLEAN UP STACK
        JMP     AUX_USER
BACK:   RET
READ_AUX ENDP
```

```
READ_CON PROC NEAR              ;TEST CON DEVICE FOR A CHARACTER
;
        MOV     AH,CHR_READ
        CALL    BIOS_CONFUNC
        JC      BACK1           ; NO CHARACTER THEN RETURN
        CMP     AL,0DH          ; TEST FOR CARRIAGE RETURN
        JNE     BACK1
        POP     AX              ;CLEAN UP STACK
        JMP     CON_USER
BACK1:  RET
READ_CON ENDP
        PAGE
;
;*****************    MAIN BODY   ***********************
;
;
MAIN_LOOP:
;
        CALL    READ_AUX                ; FIRST SEE IF NON-ACTIVE USER WANTS
                                        ; THE SYSTEM. IF HE DOES THEN HE WILL
                                        ; GET IT OTHER WISE BOTH USERS WILL
                                        ; HAVE EQUAL OPPERTUNITY.
        MOV     DX,OFFSET MESSAGE1      ; GET MESSSAGE POINTER
        PUSH    DX                      ; SAVE IT
        CALL    MESS                    ; SEND TO PRESENT CON:
        CALL    RD_CHRDS                ;
        CALL    WR_CHRDS                ; SWAP USERS
        POP     DX                      ; SEND MESSAGE TO PRESENT AUX:
        CALL    MESS                    ;
        CALL    RD_CHRDS                ; SWAP USERS AGAIN
        CALL    WR_CHRDS
;
WAIT_LOOP:                      ; WAIT UNTIL SOMEBODY WANTS THE CONSOLE !
;
        CALL    READ_AUX
        CALL    READ_CON
        JMP     WAIT_LOOP
;
CON_USER:               ; THE CURRENT CONSOLE WANTS TO RETAIN USE
;
        CALL    RD_CHRDS                ; SEND OTHER USER
        CALL    WR_CHRDS                ; THE NOT AVAILABLE
        MOV     DX,OFFSET MESSAGE2      ; MESSAGE
        CALL    MESS
        CALL    RD_CHRDS
        CALL    WR_CHRDS
        MOV     DX,OFFSET MESSAGE3      ; RETURN CONTROL TO
        CALL    MESS                    ; CURRENT CONSOLE
        JMP     RT_ADDR
;
AUX_USER:               ; THE CURRENT AUX USER WANTS THE SYSTEM
;
        MOV     DX,OFFSET MESSAGE2      ; SEND THE NOT AVAILABLE
        CALL    MESS                    ; MESSAGE TO CURRENT CONSOLE
        CALL    RD_CHRDS                ; SWAP
        CALL    WR_CHRDS                ; CON: & AUX:
        MOV     DX,OFFSET MESSAGE3
        CALL    MESS                    ; RETURN TO Z-DOS WITH NEW CONSOLE
        JMP     RT_ADDR
PGMSEG  ENDS
        END     HERE
```

*

Join with other HUGgies at the 1983 NATIONAL HUG CONFERENCE II

PRIMERS FOR THE BEGINNER

GETTING STARTED WITH CP/M AND MBASIC

WITH PARTICULAR REFERENCE TO RANDOM FILES

Featuring a complete "menu driven" ready-to-run disk mail list program, program explanations, and complete tutorials, this package forms the perfect introduction to MBASIC programming under CP/M, and includes useful information for those new to the CP/M operating system (ZBASIC also supported). Included is a 56 page manual and a disk containing sample programs. Specify Heath/Zenith Computer model number, disk size and format—hard- or soft-sectored, single- or double-density, 5¼" or 8", and CP/M or ZDOS. $25.00

GETTING STARTED WITH HDOS & ASSEMBLY LANGUAGE PROGRAMMING

A 36 page tutorial covering aspects of Assembly Language programming under HDOS. Provides significant information for HDOS which other manuals lack. $15.00

PLEASE SEND CHECK OR MONEY ORDER TO:

WILLIAM N. CAMPBELL, M.D.
855 Smithbridge Road
Glen Mills, PA 19342
(215) 459-3218

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

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

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

Heath
Users'
Group

Hilltop Road

Saint Joseph, Michigan 49085

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

POSTMASTER: If undeliverable, please do not return.

Volume 4, Issue 3

885-2038