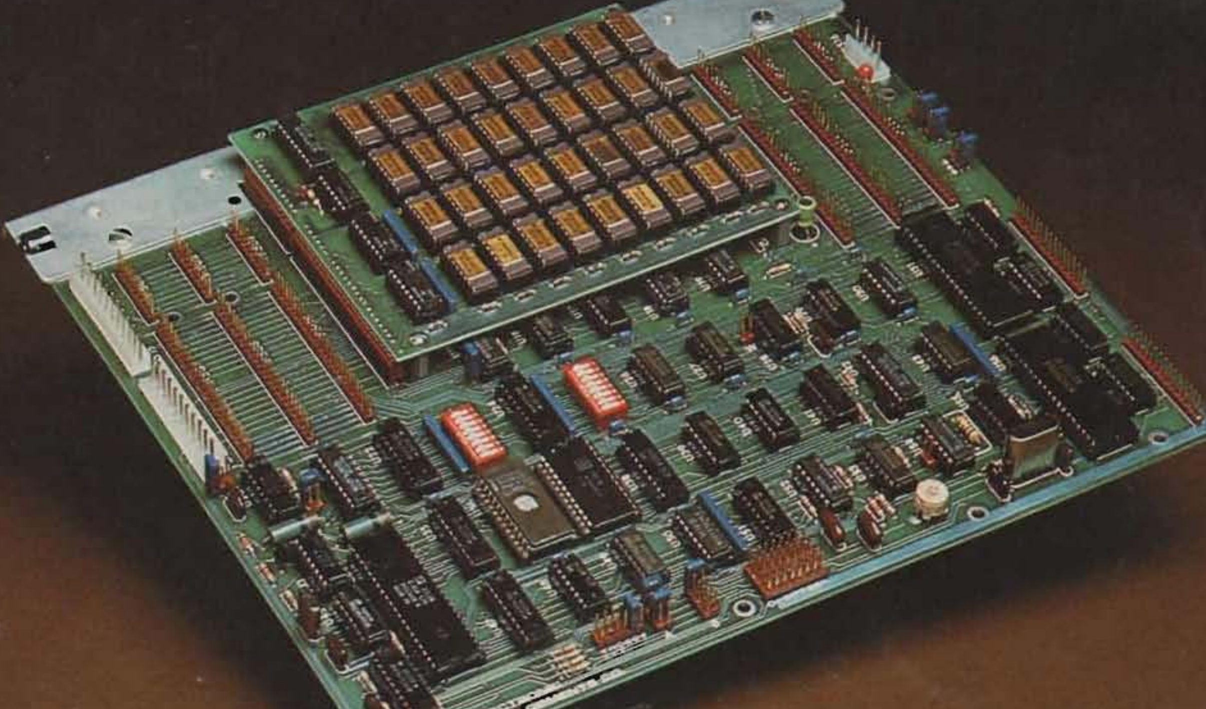
IN THIS ISSUE READ ABOUT
Conversion of IBM-PC BASIC to ZBASIC
by John Stetson
see page 9

$2.50

REMark®

Issue 44 • September 1983



Official magazine for users of HEATH ZENITH computer equipment.

PrintMate GOES GRAPHIC WITH H/Z100!

Picture this—your logos, graphs and fancy print fonts on paper with a PrintMate. Our exclusive AP-PAK lets you print what you picture. In just one keystroke, you can send what's on the screen on to your printer. With an AP-PAK, you can intermix text and graphics on the same line and print in characters up to ⅝-inch high. AP-PAK runs with the Z-DOS™ (MS-DOS) BIOS release 1.00 (Winchester) version 1.10 and above.

With friction and tractor feed standard on the 100 cps PrintMate 99, virtually any form goes through without a hitch...from labels to multi-part. The PrintMate 99's optional single-sheet feeder lets you run single sheets through the front of the machine, thus simplifying all your personal printing And its 1K (optional 2K) buffer provides additional printer performance.

Our premier performer for home or office is the PrintMate 150, featuring 150 cps print speed, three paper paths and 15-inch carriage. Expandable buffers (up to 68K!) dramatically increase flexibility. And with SoftSwitch™ you get an impressive range of control at your fingertips right on the front panel of the printer.

No matter how you compute, you get more print versatility for the price with PrintMate: correspondence-quality printing, superb graphics capabilities and versatile paper handling. At MPI, we want to see you in print. See your dealer today for a demonstration.

Micro Peripherals, Inc., 4426 South Century Drive, Salt Lake City, Utah 84107 **1-800-821-8848**

The Printer People
mpi

PrintMate 99

PrintMate 150

H/Z 100 is a registered trademark of the Heath Company.
Z-DOS is a trademark of the Zenith Data Systems Corp.

Heath Users' Group

HUG Manager .... Bob Ellerton
Software Engineer .... Pat Swayne
HUG Bulletin Board
and Software Developer .... Terry Jensen
Software Coordinator .... Nancy Strunk
HUG Secretary .... Margaret Bacon

REMark Editor .... Walt Gillespie
Assistant Editor .... Donna Melland

Printers .... Imperial Printing
St. Joseph, MI

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

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

*U.S. Funds.

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

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

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

616-982-3463

REMark®

Issue 44 • September 1983

on the stack

ON THE COVER: Pictured is the SUPER 89 replacement CPU board from D-G Electronic Developments Co. See page 17 for a review.

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

PRICE BREAKTHROUGH

Super Sale on New Disk Drives

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

Tandon — Siemens — Remex — MPI — Teac — Shugart
single sided 40 track — dual sided 40 track
single sided 80 track — dual sided 80 track
for

RADIO SHACK[1] — HEATH/ZENITH[2] — APPLE[3]/FRANKLIN[4]
IBM/PC-TEXAS INSTRUMENTS[6] & MOST OTHER COMPUTERS

Drive a Hard Bargain!!™

8 M.B. - 12M.B. with Power Supply Case, Cables & Software
Complete Systems . . . . from $1,295.00

Amazing Special!-dual sided 40TK or 80TK drive with Power Supply and Case
and Our Special Warranty - Starting at $Call

TOLL FREE ORDERING GENERAL AND TECHNICAL

1-800-343-8841 **1-617-872-9090**

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

One Year Warranty Available on all Drives!

SOFTWARE SUPPORT, INC.

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

Dealer inquiries invited.

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

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

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

"You've Come A Long Way Baby!......"

This could be said of a number of things in today's society, but most of all the micro-computer. After all, a granddaddy in this field may be only six to ten years old. The proliferation of generations of micros could well put the lemming to shame. Picking up a computer magazine these days gives one some of the feelings that might have been felt by the Mormons during the locust plagues. The battle lines that seem to be being drawn remind me of the recent "War of the Burgers" between the giant bun stuffers.

We, those of us who have chosen Heath/Zenith equipment, question loudly: "Why isn't our chosen champion in the middle of this fracas?". But the smart money might be on the idea that it is best to let these self-inflated giants beat each others brains (and pocketbooks) out. Then, as the pieces settle, our champion could step forward as a true leader in the micro-computer world. There will never be any one sole winner, just a lot of losers. The losers will be those that gambled too much on the premise that if they can infect you with hype, you won't look close at what your really buying. It would be nice when your friends ask, "You bought a what computer?", to pick up any computer magazine and turn to a large, slick advertisement expounding its virtues. I find it interesting that the "three letter" company chose a clown for its spokesman, but he really never has had anything to say.

With the combination of Heath and Zenith Data Systems in this marketplace, we might answer that friend's question with the statement, "Zenith is the last word in computers!".

Yes, you've come a long way micro-baby.

Walt Gillespie
Editor

BUGGIN' HUG

HELP Needed!!

Dear HUG,

Could you assist me in obtaining Fortran library routines for the COMPLEX data type, including matrix algebra for the COMPLEX data type?

I have Microsoft Fortran with an H-8, two disk drives, and CP/M. The H-8 is 8080A based.

Dean Dameworth
600 Cornell
West Memphis, AK 72301

Correction to July Buggin'HUG

Dear HUG,

Having read my letter in the July issue of REMark, I noticed a line was omitted. I'm not sure if this was a typo or if I just forgot to include it in my letter to you. The missing line is inserted as follows:

OLD LINE → The TRS-80 screen is di-
NEW LINE → The TRS-80 screen is divided into 1024 bytes of information, whereas the H/Z-89 screen is di-

I hope you print this correction. It makes a difference in understanding the formulas I submitted. Thank You.

Kenneth B. Blois
PSC #1 Box 2039
APO SF 96366

Another Solution to BDOS ERROR

Dear HUG,

In the July 1983 issue of REMark (Issue #42), I read with some amusement the article by Bill Simpson concerning the problem of "BDOS ERROR" when writing programs in MBASIC. You will recall that in a rather lengthy 3 pages of explanations and examples, Mr. Simpson showed us all how to recover a lost BASIC program when the user receives the infamous "BDOS ERROR" if the "RESET" command was not given.

While I am sure this procedure works, it requires the user to have DDT available on a disk and do quite a lot of "fancy steppin'" to get that program back. In addition to having a working knowledge of how BASIC stores the program in RAM, Mr. Simpson's method requires too many calculations and conver-

Vectored to 49 ☞

Questions & Answers

Compiled and edited by Terry Jensen, Software Developer.

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

Q. I own a Z-90 computer. Why can't I copy MBASIC random files with the CP/M PIP utility without losing the last part of the file?

A. CP/M only allocates space to random files for records that are actually written. For example, a random file that has only record number 1 and record number 1000 written to it will not have all the record groups in between, nor will there even be space allocated for them on the disk. Any program reading the file sequentially, as PIP does, will indicate that it has reached the end of the file the first time it reaches one of these empty record groups (even though there may be more records later).

One way to correct this problem is to write all the record numbers of a random file with at least null data so that it may be copied sequentially by PIP. However, this will cause the file to be much larger than it needs to be. Another way to make a copy of the file is to write a special program that copies the file by reading and writing each record, if it exists, in random mode. The CP/M BDOS function "Compute file size", function code 35 (023h), is the best way to determine the last random record number of a file without reading every possible random record number.

Q. I have an H89 with HDOS. I have several times had the misfortune of receiving the following error message:

?02 SYS ERROR #39
Disk Structure is Corrupt. Contact Technical Correspondence for help.

When it happens, I cannot mount the affected disk and the files are lost. What causes the error, how can one take precautions to avoid it from happening, and how can a "nonprofessional" retrieve the files from the bad disk?

A. A "corrupt" disk is the result of the directory tracks becoming messed up. It can happen for any number of reasons, but the two most common causes are from: 1) the drive head "banging" or writing to those tracks due to a power fluctuation, and 2) from the disk warping due to lack of use, or from extreme heat and/or humidity.

The most effective precaution is to make and keep BACKUPS of all important files and programs. System cautions include: 1) remove the disks from the drives before you turn off your computer system, 2) shut down the computer during thunderstorms or electrical utility work in your area, 3) maintain all disks in a safe, cool, dry storage area, and 4) create "fresh" or new disks from any older disks that sit idle for long periods of time (i.e. a few months).

Even by maintaining these precautions, Murphy's Law will prevail sooner or later. There are programs available that will allow you to retrieve files from a corrupt disk. One program is called RECOVER on HUG P/N 885-1083, which will copy sector by sector from the bad disk to a second disk. A second program, FAKEMNT on P/N 885-8004, will "mount" a corrupt disk and copy files from the bad disk to a second disk. (Refer to the HUG Software Catalog for details.)

Q. Is the use of flippy diskettes practical on the H17 drives?

A. In theory, flipping the disk can double the amount of disk storage. However, diskette manufacturers generally do not recommend the use of flippy diskettes. By flipping the diskette, the mylar surface rotates in the opposite direction against the "jacket" or cover. This can dislodge dirt particles that have been picked up by the jacket liner during normal rotation.

Vectored to 19 ☞

NEW FROM FLOPPY DISK SERVICES, INC.

the H-89 TWOET SYSTEMS

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

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

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

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

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

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

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

Dealer inquiries invited.

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

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

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

FLOPPY DISK SERVICES™ INC.

741 Alexander Rd. Princeton, NJ 08540

Get more out of your computer with Heathkit online services.

Your own direct link to Benton Harbor with the Heathkit Online Catalog and the HUG Bulletin Board. Yours through the CompuServe Information Service.

The CompuServe Information Service is a large timeshare database system that allows users from across the country and around the world to access a wide variety of services using a computer over ordinary phone lines. In most cases with a local call.

CompuServe offers many areas of service, interest, information and fun. Heath Company and the Heath Users' Group are pleased to be part of the system.

```
          HEATHKIT ONLINE CATALOG

  1. Product of the Month  11. Instruments
  2. Bargains              12. Marine
  3. Automotive            13. Software
  4. Amateur Radio         14. Satellite TV
  5. Clocks                15. Security
  6. Computers             16. Stereo
  7. Education Courses     17. Television
  8. Energy                18. Tools
  9. Furniture             19. Weather
 10. Home                  20. ORDER
     21. Return To Main Menu

 Enter your selection >
```

Review model numbers, descriptions, prices for over 400 major products. Almost 1000 items in all.

Place orders using your Visa or MasterCard

THE HEATHKIT ONLINE CATALOG is a menu-driven system that allows you to browse 17 different product categories that make up the over 400 kits in the Heathkit line. While not intended to replace the printed editions of the catalog, you have electronic access to bargains, specials and new products without waiting on the postman. You can even place your order online using your Visa, MasterCard, or Heath Revolving Charge. To access the Heathkit Online Catalog just type GO HTH at any '!' prompt or 'OK' prompt on the CIS system.

THE HUG BULLETIN BOARD is reserved for members of the Heath Users' Group. It puts you in touch with hundreds of active HUG members around the country. Members can leave messages, retrieve and reply to other messages and responses, search and scan topic areas, even enter into real-time online discussions with other members in the 'conference' mode. Not to mention that the HUGBB has one of the largest database on CompuServe for members to download software and files from the system to their own computers. You can also upload your own programs to the database. To access the HUG Bulletin Board just type R HUG at any 'OK' prompt, or type GO PCS-48 at any '!' prompt on the CIS system.

What do I need to get on? First a modem to link your computer to the phone lines. It may be acoustic or direct connect. Any of the modems in the Heathkit catalog will fill the bill.

Second, a CIS account and software to make your computer act like a terminal. HUG has a package that includes both, and will let you get on CompuServe the day you receive it. It's called the 'MicroNet Connection' and sells for $16 thru HUG. Order part number 885-1122 (HDOS) or 885-1224 (CP/M).

CompuServe charges are $5.00 an hour for regular hours and open areas. HUG members receive a 10% discount while using the HUG Bulletin Board.

Find out what all the excitement is about. Take advantage of the Heathkit Online services today!

It's like a direct link to Benton Harbor!

CP-217R3

Heathkit®
Heath Company

†Heathkit Electronic Centers are units of Veritechnology Electronics Corporation.
Heath Company and Veritechnology Electronics Corporation are subsidiaries of Zenith Radio Corporation.

Conversion of IBM-PC BASIC Programs to ZBASIC

John Stetson
3144 Hewitt Ave, #143
Silver Spring, MD 20906

Introduction and Overview

Unless you've been living in an ivory tower for the past 2 years, you can't have helped but notice that there are more and more microcomputers around with the letters IBM on them. Now we all know that Heath/Zenith makes a superior microcomputer for the money, but what about all of that software that Z-100 microcomputers should be able to run since they have the Microsoft Disk Operating System (MS-DOS) in common with the IBM-PC?

The single largest block of relatively inexpensive software shared by the Zenith and IBM machines is BASIC programs which run under the BASICA (Advanced BASIC) interpreter on the IBM-PC and ZBASIC on the Z-100. This is certainly the best all around BASIC language ever widely implemented on microcomputers, and is pretty much upward compatible from the venerable MBASIC interpreter we have grown fond of under the HDOS and CP/M operating systems.

IBM-PC BASIC programs can be found in a variety of PC oriented publications, PC bulletin boards, and the IBM-PC Special Interest Group Bulletin Board on Compuserve's MicroNet service, home of the illustrious Heath Users' Group Bulletin Board. So now let's say that you have obtained a listing of a potentially interesting IBM-PC BASIC program - now what do you do? Well, if the program is relatively free of machine hardware dependent features, such as graphics and sound commands, there is a chance that it will run with little or no change. However, most of the really interesting BASIC programs available do use graphics commands, and these are the ones to which we will primarily direct our attention.

This article assumes you are already reasonably familiar with the ZBASIC programming language, and that you wish to determine the feasibility of attempting to convert an IBM-PC program to run under ZBASIC (not all interesting programs can be easily converted). Specific conversion methods will be presented, where possible, but you should keep in mind that there are not always hard and fast rules to follow - be creative! We will also discuss a variety of useful information pertaining to ZBASIC which is not directly related to the conversion process.

Common BASIC Language Features Not Requiring Conversion

What makes this kind of conversion feasible is the fact that both IBM-PC BASIC and ZBASIC were derived from the same original program; they have their differences, but the basic procedural language is almost identical. In general, all of the features in the earlier MBASIC interpreter which are present in the 16 bit BASIC interpreters will require little or no conversion effort. Examples of these types of commands include FOR/NEXT loops, character string manipulation, disk file input/output operations, etc. Virtually all of the conversion effort centers on identifying and changing hardware dependent commands.

Overview and Comparison of IBM-PC Graphics Hardware

In order to perform the conversion of a BASIC program which takes advantage of hardware specific features of the IBM-PC, it is necessary to have at least a passing familiarity with the hardware features of the IBM-PC which are likely to be encountered in BASIC programs. We will compare these to the equivalent Z-100 features, where possible.

The most important hardware characteristics to be considered when converting BASIC programs with graphics commands are listed below.

- Physical screen size in characters and pixels
- Logical to physical mapping of the available colors

The screen size of the Z-100 computer is 25 rows of 80 characters with each character occupying a 9(v) by 8(h) pixel cell. Measured in pixels, this translates to 225(v) by 640(h), for graphics. It is important to remember that the Z-100 is always in both text and graphics mode - no special action need be taken to alternately display either character oriented or pixel (dot) oriented data on the screen. Each pixel can have any of 8 colors, without any restrictions. When equipped with 64k bit Video RAM chips, the Z-100 has a second screen's worth of Video RAM which can be activated under program control. With a full complement of 192k of Video RAM, the Z-100 has the potential for 450(v) by 640(h) pixels using 8 colors, running in interlace mode.

The screen size of the IBM-PC is 25 rows of 80 characters, in text mode with the Monochrome Display, with each character occupying a 14(v) by 9(h) pixel cell. The monochrome display and adapter cannot be used for pixel graphics, only to display textual information.

The IBM-PC Color Graphics Monitor Adapter permits the use of either a B/W or Color Monitor (either NTSC or RGB) to display both textual and graphics information. Three modes are supported:

Alphanumeric Mode - displays 25 rows of 40/80 characters with each character occupying an 8(v) by 8(h) pixel cell. Character attributes include reverse video, blinking, and highlighting. There are 16 foreground colors (8-15 are light versions of 0-7) and 8 background colors that can be set by character. There is sufficient screen RAM to store (8) 25 x 40 screen images or (4) 25 x 80 screen images.

The screen border color may also be changed to any of 16 colors. It is important to note that the alphanumeric mode cannot be used at the same time either of the graphics modes are in use - the desired mode must be explicitly selected using the SCREEN command each time the screen mode must be changed.

Medium Resolution Graphics Mode - displays 200(v) by 320(h) pixels. Each pixel may have any of 4 different colors, 3 of which are selected from the desired "palette". Palette 0 has green, red, and brown as colors, and palette 1 has cyan, magenta, and white. The 4th color may be any of the available 16 colors.

High Resolution Graphics Mode - displays 200(v) by 640(h) pixels. No colors are available in this mode since all available video RAM (a paltry 16K!) is devoted to the pixel on/off information.

Common BASIC Language Features Requiring Conversion

We will now discuss in detail those features which are common to both IBM-PC BASIC and ZBASIC, but requiring special conversion modifications due to machine hardware or firmware differences.

PEEKs and POKEs

Almost all of the well known "toy" or "game" computers require the use of the PEEK function and POKE command to perform graphics functions from BASIC. One of the most significant improvements in both IBM-PC BASIC and ZBASIC is a rich collection of graphics commands and functions which allows relatively complex graphics algorithms to be implemented in an efficient and self-documenting manner. Unfortunately, there are still times when it is necessary to resort to PEEK and POKE and these are usually quite hardware dependent. Further complicating this situation is the necessity of specifying the 8088 address segment (64k memory block) which provides the base address for PEEK, POKE, BSAVE, BLOAD, and CALL. This is done by using the DEF SEG statement prior to the memory reference. In most cases, isolated PEEKs and POKEs in IBM-PC BASIC programs should simply be removed, as their function is not appropriate on the Z-100.

Typical examples are given below:

```
100 DEF SEG : POKE 106,0
    ' Clear Basic Input Line Buffer
200 DEF SEG=0 : POKE 1050,PEEK(1052)
    ' Clear PC-DOS Keyboard Buffer

250 ' Switch from Color to Monochrome Display
300 DEF SEG=0 : POKE &H410,(PEEK(&H410) OR &H30)

350 ' Switch from Monochrome to Color Display
400 DEF SEG=0 : POKE &H410,(PEEK(&H410) AND &HCF) OR &H10
```

One must also be on the lookout for OUTs and INs to and from hardware dependent 8088 I/O ports. These should also be eliminated:

```
100 OUT &H3D8,8
         ' Enable 16 background colors in 40 column text mode
200 OUT &H3D8,9
         ' Enable 16 background colors in 80 column text mode
```

Both BASICs can quickly save and restore a screen video image using the BSAVE and BLOAD commands. The address and length of the VRAM to be saved is different, however:

```
100 ' Save video screen image on IBM-PC
110 DEF SEG=&HB800                    ' Video RAM segment address
120 BSAVE "B:SCREEN.DAT",0,&H4000 ' Write 16K bytes to file
200 ' Reload video screen image on IBM-PC
210 DEF SEG=&HB800                    ' Video RAM segment address
220 BLOAD "B:SCREEN.DAT",0            ' Reload VRAM from file

100 ' Save video screen image on Z-100
110 DEF SEG=&HC000                    ' Blue VRAM Plane address
120 BSAVE "B:BLUE.DAT",0,&HC800       ' Write 50K bytes to file
130 DEF SEG=&HD000                    ' Red VRAM Plane address
140 BSAVE "B:RED.DAT",0,&HC800        ' Write 50K bytes to file
150 DEF SEG=&HE000                    ' Green VRAM Plane address
160 BSAVE "B:GREEN.DAT",0,&HC800      ' Write 50K bytes to file

200 ' Reload video screen image on Z-100
210 DEF SEG=&HC000         ' Blue VRAM Plane address
220 BLOAD "B:BLUE.DAT",0   ' Reload Blue VRAM Plane from file
230 DEF SEG=&HD000         ' Red VRAM Plane address
240 BLOAD "B:RED.DAT",0    ' Reload Red VRAM Plane from file
250 DEF SEG=&HE000         ' Green VRAM Plane address
260 BLOAD "B:GREEN.DAT",0 ' Reload Green VRAM Plane from file
```

Note that each separate color plane must be saved on the Z-100 and that 50K bytes must be saved due to the Z-100 Video Memory Map. Only the green plane need be saved on non-color Z-100's.

SCREEN Command

IBM-PC:

SCREEN <mode><,burst><,apage><,vpage>

mode	=	0 for Text Mode (40 or 80 chars/line set by WIDTH) 1 for Medium resolution Graphics Mode (200 x 320) 2 for High resolution Graphics Mode (200 x 640)
burst	=	1/0 to Enable/Disable Color in Text Mode 0/1 to Enable/Disable Color in Graphics Modes
apage	=	active page = 0-7/0-3 in 40/80 character mode = page number to be written into video RAM
vpage	=	visual page = page number to be displayed

Z-100:

SCREEN <gmode><,vmode>

gmode	=	1 to Enter H19 Character Graphics Mode 0 to Exit H19 Character Graphics Mode
vmode	=	1 to Enter H19 Reverse Video Mode 0 to Exit H19 Reverse Video Mode

The SCREEN command is almost always present in IBM-PC BASIC programs which contain subsequent graphics commands. Its main function is to set the display MODE (with the WIDTH statement). The other options are not easily convertible to ZBASIC. Before removing it, you should note which graphics mode is being selected (medium or high resolution). This will determine both the screen size and use of color in succeeding graphics commands. If the medium resolution graphics mode is being selected, all horizontal (X-axis) coordinates in subsequent graphics commands must be multiplied by 2 to account for the difference in the IBM- PC and Z-100 display resolution (320 vs. 640 pixels). If high resolution mode is selected, no horizontal axis compensation is needed. In either graphics mode, the IBM-PC uses 200 pixels on the vertical axis, versus 225 for the Z-100. Depending on how picky you are, you may wish to multiply all vertical (Y-axis) coordinates in subsequent graphics commands by 225/200 = 1.125 to account for this difference.

The SCREEN command in ZBASIC serves a totally different purpose. It provides a simpler method of entering and exiting H/Z-19 graphics and reverse video display modes than the usual ESC F/G and p/q sequences.

The <apage> and <vpage> parameters of the IBM-PC BASIC SCREEN command can be simulated, to a certain extent, by using the following subroutines to switch between two separate Video RAM pages which are available if the Z-100 in use has a full 192K of Video RAM using 64k bit RAM chips, and the switch on the Video Card is in the 64K position. Since the time required to switch to the alternate Video RAM display page is small compared to the length of time it will be displayed, this technique is especially useful in providing a smooth transition from one image to the next when animation is being performed.

```
100 ' This subroutine alternately switches the screen display between
110 ' two different Video RAM pages by selecting the 6845 CRT Controller
120 ' Screen Start Address MSB Register (0CH), reading the current MSB
130 ' of the Screen Start Address, flipping the 08H bit, and finally
140 ' writing the new Screen Start Address back to the 6845 chip.  Refer
150 ' to the Z-100 Technical Manual for details on the 6845.  This
160 ' requires that the Z-100 have a full complement of 64k bit VRAMS.
170 OUT &HDC,&H0C : OUT &HDD,((INP(&HDD)+&H08) AND &H0F) : RETURN

200 ' This subroutine alternately turns the Z-100 Screen Video RAM
210 ' off and on by reading the 6821 Video Control Chip, flipping the
220 ' 80H bit, and writing the modified byte back to the 6821 chip.
230 ' Refer to the Z-100 Technical Manual for details on the 6821 chip.
240 OUT &HDA,((INP(&HDA)+&H80) AND &HFF) : RETURN
```

SCREEN Function

IBM-PC:

N = SCREEN(<row>,<col>[,z])

row = row number of desired character from 1 to 25.
col = column number of desired character from 1 to 80.

In Text Mode, if z = 0, then N = the ASCII character code of the character at the specified coordinates. If z <> 0, then N = the color code of the character from 0 to 255.

N MOD 16 = foreground color
(N-(N MOD 16))/16 MOD 128 = background color
If N > 127, then the character is blinking.

In Graphics Mode, N = 0 if points or lines are present in the character cell, rather than a character.

Z-100:

If z = 0, then N = the ASCII character code of the character at the specified coordinates. If z <> 0, then N = the color of the character from 0 to 7. See the COLOR Command below for a description of the difference in color code values. Normally it is sufficient to simply change the interpretation of the color value returned in the variable N.

COLOR Command

IBM-PC:

Text Mode (SCREEN 0):

COLOR <foreground><,background>
<,border>

foreground = foreground color from 0 to 31.
background = background color from 0 to 7.
border = screen border color from 0 to 15.

Medium Resolution Graphics Mode (SCREEN 1):

COLOR <background><,palette>

background = background color from 0 to 15.
palette = palette number from 0 to 1.

IBM-PC BASIC Color Definition Table

Number	BASICA Color	Number	BASICA Color
0	BLACK	8	GRAY
1	BLUE	9	LIGHT BLUE
2	GREEN	10	LIGHT GREEN
3	CYAN	11	LIGHT CYAN
4	RED	12	LIGHT RED
5	MAGENTA	13	LIGHT MAGENTA
6	BROWN (*)	14	YELLOW
7	WHITE	15	HI-INTENSITY WHITE

Colors 8-15 are obtained by combining the corresponding color from 0-7 with gray, which produces a lighter version of the same color. Colors 16-31 are the same as colors 0-15 but with the addition of a blinking attribute.

(*) - BROWN can be considered equivalent to YELLOW in ZBASIC.

IBM-PC BASIC Color Palette Definition Table

Number	Palette 0	ZBASIC #	Palette 1	ZBASIC #
0	0-15 (1)	0-7	0-15 (1)	0-7
1	GREEN	2	CYAN	3
2	RED	4	MAGENTA	5
3	BROWN (2)	6	WHITE (2)	7

(1) - Color number 0 is the background graphics color and may be any of the 16 colors listed above.

(2) - These are the default foreground graphics colors.

Z-100: COLOR <foreground><,background>

foreground = foreground color from 0 to 7.
background = background color from 0 to 7.

As you can see from the tables above, the method of assigning color codes is rather complicated in IBM-PC BASIC. The ZBASIC color codes are the same as colors 0-7 for the IBM-PC in Text Mode if you change BROWN to YELLOW. This leads to the following method of changing the color codes in the COLOR command for Text Mode (SCREEN 0):

If IBM_COLOR is either the foreground or background color in the IBM-PC BASIC COLOR command, determine the Z-100_COLOR as follows:

```
If  0 <= IBM_COLOR <=  7, then Z100_COLOR = IBM_COLOR.
If  8 <= IBM_COLOR <= 15, then Z100_COLOR = IBM_COLOR-8.
If 16 <= IBM_COLOR <= 23, then Z100_COLOR = IBM_COLOR-16.
If 24 <= IBM_COLOR <= 31, then Z100_COLOR = IBM_COLOR-24.

Or Z100_COLOR = IBM_COLOR MOD 8.
```

Although ZBASIC seems to perform this automatically for color values greater than 7, it is advisable to do the conversion manually, if for no other reason than to make the program listing more understandable.

If a screen border color is specified, it should be ignored, since there is no way to alter the screen border color on the Z-100.

In Medium Resolution Graphics Mode (SCREEN 1), a background color and palette number are specified. The background color is converted as described above. The palette number (0 or 1) selects a group of 3 predetermined colors as listed in the Palette Table above. Either the "background" color or one of the 3 palette colors is used by any subsequent PSET, PRESET, LINE, CIRCLE, PAINT, and DRAW commands. It is important to note that the color number reference in these commands will always be from 0 (background color) to 3 (default foreground color). The Palette Table above gives the equivalent ZBASIC color number to use according to the palette number (0-1) and color number (0-3). An example of this technique is given below:

IBM-PC Basic:
```
      100 SCREEN 1          ' Medium Res. Graphics Mode
      110 COLOR 10,1        ' Light Green Background, Palette #1
      120 PSET (40,30),2 ' Palette #1, Color #2 = Magenta
      130 PSET (160,100) ' White foreground color
```

Z-100 ZBASIC:
```
      110 COLOR 7,2         ' White Foreground, Green Background
      120 PSET (80,34),5 ' Magenta from Palette Table above
      130 PSET (320,112) ' White foreground color
```

Also note that the X and Y coordinates for PSET have been multiplied by 2 and 1.125, respectively, to compensate for the larger screen size.

Since the High Resolution Graphics Mode (SCREEN 2) of the IBM-PC is limited to Black and White, you may choose whichever colors appeal to you when converting a program which uses this mode. It is also unnecessary to multiply the X coordinate values by 2 in this case.

Pixel Oriented Graphics Commands

The following commands fall into the general category of pixel-oriented graphics commands; they affect individual pixels (picture elements), or dots, on the screen:

PSET, PRESET, LINE, CIRCLE, PAINT, GET, PUT, DRAW, and POINT

Converting these commands involves two main operations:

1) X and Y pixel scaling according to screen size in pixels.

2) Conversion of the color code value used to draw pixels.

As previously mentioned, the IBM-PC BASIC program should be inspected for SCREEN statements. SCREEN 1 signifies a Medium Resolution Graphics program (200(v) by 320(h)), and SCREEN 2 signifies a High Resolution Graphics program (200(v) by 640(h)). Although some programs may yield an acceptable image without any compensation for the difference in screen size in pixels, in general it is necessary to multiply every X coordinate by 2 in a Medium Resolution program, and every Y coordinate by 1.125 in both Medium and High Resolution programs.

It is also necessary to determine the palette number in effect by examining the COLOR statement, and to select the proper ZBASIC color to give the same color. In some cases, however, the choice of colors may be arbitrary, and reasonable results can be obtained by simply replacing the COLOR statement values with whatever seems reasonable.

We will now discuss some of the peculiarities of these commands.

CIRCLE Command

The general form of the CIRCLE Command in both BASICs is as follows:

CIRCLE (<xc>,<yc>),<radius><,color<,start,end<,aspect>>>

xc,yc = pixel coordinates of the center of the circle (or ellipse). These must be rescaled as described above.

radius = the "radius" of the circle or the Major Axis of the ellipse. This is measured along the horizontal X axis of the figure if the aspect ratio is less than 1, and along the vertical Y axis if the aspect ratio is greater than 1. This value must also be rescaled, depending on the aspect ratio.

color = the color with which to draw the figure. This must be converted as previously described.

start = starting angle of the arc to be drawn in radians.

end = ending angle of the arc to be drawn in radians.

aspect = the "aspect ratio" of the figure. This is the ratio of the X radius to the Y radius of the figure and is related to the physical aspect ratio of the display screen (most are 4/3).

The visual appearance of a circle will depend on the specified aspect ratio (or default value) and on the vertical size adjustment of the particular display screen used to view the image. It is helpful to draw a circle with a desired aspect ratio which fills the entire screen and adjust the vertical size of the display screen to obtain a visual circle.

What default aspect ratio should be used to perform this adjustment? The default values in use are listed below:

IBM-PC Medium Resolution Graphics .... 5/6 = 0.833333
IBM-PC High Resolution Graphics ...... 5/12 = 0.416666

ZBASIC 15-Jul-82 Version .... 132/256 = 0.515625
ZBASIC 10-Oct-82 Version .... 112/256 = 0.437500

Since I prefer to reduce the vertical size of the display somewhat for displaying character data, I choose to use 0.5 as the "default" aspect ratio to adjust the vertical size of the display screen. This number is easy to remember and gives a good compromise between text and graphics.

GET and PUT Commands

The GET and PUT commands allow the efficient movement of pixel data between Video Memory and Main Memory as an aid in animating images.

The general form of these commands is:

GET (<x1>,<y1>)-(<x2>,<y2>), <array$name>

PUT (<x1>,<y1>)-(<x2>,<y2>), <array$name>,<action>

where <action> is PSET, PRESET, AND, OR, or XOR (the default).

The amount of main memory which must be reserved for the array which is to hold the pixel data is computed differently for the IBM-PC and the Z-100.

IBM-PC: Array Size = [4 + INT((X*B+7)/8) * Y] / N rounded up

Z-100: Array Size = [4 + INT((X+7)/8) * 3 * Y] / N rounded up

Where:

X = number of columns of pixels to be stored

Y = number of rows of pixels to be stored

B = bits per pixel = 2 for Medium and 1 for High Resolution Graphics Mode on the IBM-PC

N = bytes per array element = 2 for integer and 4 or 8 for single or double precision floating point

For example, let's assume that we wish to save a block of pixels which is 40 pixels across by 30 pixels up and down. This takes 77 integer array elements on the IBM-PC and 227 on the Z-100, assuming High Resolution Mode on the IBM-PC. The reason it takes about 3 times as much storage on the Z-100 is that each pixel can have any of 8 distinct color values and this is reflected in the Video RAM mapping scheme.

DRAW Command

The DRAW Command supports a variety of subcommands which comprise what is called a "Graphics Definition Language" on the IBM-PC and a "Graphics Macro Language" on the Z-100. It can be particularly tedious to convert an IBM-PC BASIC program which contains a large number of DRAW subcommands since most will require pixel rescaling. The U, D, L, R, E, F, G, H, and M commands will contain absolute pixel coordinate references which must be rescaled as described above. Although there is an S subcommand which can be used to apply a scale factor to all following subcommand values, it affects both X and Y coordinates and is therefore useless for multiplying only the X axis values by 2, for example. This brings to mind an important omission in the ZBASIC documentation for the S subcommand. It states that the scale factor may range from 1 to 255. What is not mentioned is the fact that the scale factor is divided by 4 before being used (4 is the default value). The C subcommand for changing colors must also be changed using the current palette number as described above.

BASIC Language Features Which Are Not Easily Convertible

Although IBM-PC DOS V1.0 and 1.1 diskettes are readable by the Z- 100 (and vice-versa), BASIC program files saved in binary (compressed) format are not directly transferable, due to differences in the internal token assignments in the two BASICs. BASIC files must be saved in ASCII format, using the SAVE "FILENAME",A command, so that they can be transferred properly. IBM-PC DOS 2.0 formats floppy disks as either 9 (default) or 8 sectors per track, and the 8 sector format must be used in order to be readable by Z-DOS V1.0 or 1.1.

IBM BASIC supports 256 characters, while ZBASIC supports the standard 128 ASCII characters and the 33 H/Z-19 style graphics characters. In general, references to CHR$(128) through CHR$(256) in IBM BASIC programs are not readily convertible to an equivalent character in ZBASIC. These additional characters (called "Extended ASCII") include foreign character set symbols and special graphics characters. It is true that alternate font tables and the H/Z-19 graphics characters can be used to emulate some of these characters, but it is not easy.

The IBM-PC keyboard has an ALTernate key which is used in a fashion similar to the CTRL (control) key to generate extended character codes. The main function of the ALT key is to transmit an ASCII NULL character (zero) followed by the selected character itself. Usually, the best thing to do is to change this to a function key or control character input which may conflict with previously defined uses for these.

Due to the design of the IBM Color Graphics Display Adaptor, it is possible to change the colors of all areas on the screen painted with a given color number by simply changing the foreground and/or background colors. Although the Z-100 hardware may provide a way to do this, ZBASIC does not take advantage of it, if it exists. Programs which use this feature can usually be modified by judicious use of the COLOR and PAINT commands.

Both ZBASIC and IBM BASIC support "Event Trapping". This feature allows a subroutine to be executed when a specified hardware event occurs (such as the depression of a function key). This is not well documented in the ZBASIC manual, look for it in ZBASIC Vol. II after the sample TTY Communications Program. Both BASICs support event trapping for the Function Keys and the COM device, but the IBM-PC also supports a light pen (PEN) and joy stick (STICK/STRIG). So, if you see the following statements in a program, you will have to remove them in order to try to run it:

PEN ON/OFF/STOP - start or stop reading the light pen status and enable/disable event trapping.

ON PEN GOSUB <line#> - define light pen event handling subroutine.

v = PEN(n) - read and return light pen value.

STRIG(n) ON/OFF/STOP - start or stop reading the joystick button (trigger) status and enable/disable event trapping.

ON STRIG(n) GOSUB <line#> - define joystick button event handler.

STRIG ON/OFF - start or stop reading the joystick buttons.

v = STRIG(n) - read and return joystick button status.

xy = STICK(n) - read x and y joystick coordinate values.

The IBM-PC has a simple sound circuit and speaker which are supported by BASICA commands. The PLAY command's Tune Definition language is similar in concept to the DRAW command's Graphics Macro Language. These commands must be removed from BASICA programs.

SOUND <frequency>,<duration> - generate the specified tone.

PLAY <string> - play the specified series of tones defined with a "Tune Definition Language".

The LOCATE command has two additional operands under BASICA which are used to define the start and stop scan lines of the screen cursor (cursor size). ZBASIC seems to ignore these, if they are present.

```
LOCATE <row>,<col>,<cursor_on/off>,
                              <start_scan>,<end_scan>
```

The IBM-PC supports a cassette tape recorder as a state-of-the- art mass storage device! This is controlled by means of the MOTOR command which, if encountered, is a good reason to abandon the conversion!

BASICA V1.1 supports additional options on the OPEN COM: command which are used to control various modem control signal lines such as RTS, CTS, DSR, and RLSD. An option is also provided to send line feeds after carriage returns are received. These features can not easily be emulated without resorting to assembly language.

BASICA has a function called VARPTR$ which returns the memory address of a program variable used with the DRAW or PLAY command as a character string, rather than a number (which VARPTR does). This is intended to be used with the IBM-PC BASIC Compiler and can be replaced with VARPTR, if necessary.

In ZBASIC, the syntax of the WIDTH command is simply:

WIDTH LPRINT <line_length>

In BASICA, there are several different forms:

WIDTH <line_length> - sets screen line width

WIDTH <file_number>,<line_length> - sets file line width

WIDTH <device>,<line_length> - sets device line width where <device> is SCRN:, LPT1-3:, or COM1-2:

An interesting use of the BASICA WIDTH command is to switch from either medium or high resolution graphics mode to the other in conjunction with the SCREEN command:

SCREEN 1,0 - select medium resolution graphics mode (WIDTH 40)
WIDTH 80 - switch to high resolution graphics mode
WIDTH 40 - switch back to medium resolution graphics mode

This is important in determining the current graphics mode (either medium or high resolution) when applying conversion factors to subsequent graphics commands which use x-y coordinates.

Little Known Features of ZBASIC

In collecting information for this article, I uncovered several poorly documented or undocumented ZBASIC features (some "features" may be "bugs") which may be of general interest.

Some H/Z-19 Escape Sequences are suppressed or "trapped" by ZBASIC. The most obvious examples of this are the H/Z-19 Escape Sequences used to display text on the 25th line: ESC 'x1' ESC 'Y8' and to turn the cursor off and on: ESC 'x5' and ESC 'y5'. These work perfectly well from outside of ZBASIC from programs written in other languages. ZBASIC seems to maintain internal information about the current status of the cursor location and on/off state which prevents their change using the H/Z-19 Escape Sequences. The alternative is to use the LOCATE 25,1 and LOCATE ,,0/1 commands to position the cursor to line 25 and turn the cursor off and on, respectively. These changes are required when converting MBASIC programs designed for the H/Z-19 to run under ZBASIC on the Z-100.

Unfortunately, there is an inconsistency in the numbering of some colors used in ZBASIC commands and the MTR-100 Escape Sequence used to set foreground and background colors. This Escape Sequence is:

ESC 'm' <foreground><background>

It seems that the Red and Green Color Planes have been reversed, which alters the color codes for the colors GREEN, CYAN, RED, and MAGENTA as summarized in the table below. The ZBASIC numbering scheme corresponds to that used by the IBM-PC, while the MTR-100 method places the colors in order of intensity from low to high.

Number	ZBASIC Color		MTR-100 Color
0	BLACK		BLACK
1	BLUE		BLUE
2	GREEN	<-->	RED
3	CYAN	<-->	MAGENTA
4	RED	<-->	GREEN
5	MAGENTA	<-->	CYAN
6	YELLOW		YELLOW
7	WHITE		WHITE

The LIST command allows output to be redirected to either a Z-DOS device or a Z-DOS disk file:

LIST 100-999,"B:TEMP.BAS" - writes lines 100-999 of the current program to file B:TEMP.BAS

LIST ,"COM1:" - writes all lines to the COM device

In addition to the TIME$ and DATE$ functions which can be used to get and set the Z-DOS system time and date as character strings, there also exist TIME and DATE integer functions which return the number of seconds since midnight (1-86400) and the day of the year (1-366), respectively. These are handy in numerical calculations.

The following CHR$ function values have the following effects:

PRINT CHR$(11) - home the cursor
PRINT CHR$(12) - clear the screen
PRINT CHR$(20) - turn on 25th line key list (like KEY ON)

The default function key settings in ZBASIC are not particularly useful. The following program can be used to redefine them to more useful values. Keys 1-3 are especially useful for returning the console to its normal state when experimenting with half converted IBM-PC BASIC programs, without the side effects of doing an ESC "z" to reset the console. You can go a step further by modifying the default key definitions in the ZBASIC.COM file using the Z-DOS DEBUG program. The key definitions are at offset address SSSS:843E in ZBASIC.COM and must be 15 characters or less in length and terminated by a zero byte (SSSS is the segment address which may vary depending on the amount of memory in your computer).

```
100 ' ZBASIC program to define soft function keys
110 ' Key definitions are limited to 15 characters
120 ' Only Keys 1-10 are displayed on the 25th line
130 ' Written by John Stetson 01-Apr-83
140 '
150 CR$=CHR$(13) : Q$=CHR$(34)
160 KEY 1,"SCREEN 0,0"+CR$          ' Exit Graphics & RV Modes
170 KEY 2,"LOCATE ,,1"+CR$          ' Turn on the Cursor
180 KEY 3,"COLOR 7,0:CLS"+CR$       ' Reset Colors & Clear Screen
190 KEY 4,"CLS:LIST"+CR$            ' Clear Screen & List Program
200 KEY 5,"RENUM 100,1,10"+CR$      ' Renumber Program
210 KEY 6,"RUN"+CR$                 ' Run Program
220 KEY 7,"FILES"+Q$+"B:*.BAS"+Q$+CR$ ' Display Directory
230 KEY 8,"LOAD"+Q$+"B:"            ' Load Program
240 KEY 9,"SAVE"+Q$+"B:"            ' Save Program
250 KEY 10,"SYSTEM"+CR$             ' Exit to ZDOS
260 KEY 11,"KEY ON"+CR$             ' Turn on 25th line Key List
270 KEY 12,"KEY OFF"+CR$            ' Turn off 25th line Key List
280 KEY ON : CLS
290 END
```

Another very useful, and apparently undocumented feature, is the HELP key. When pressed, it caused the text of the last executed ZBASIC command to be displayed which allows editing changes to be made. After any desired changes are made, pressing RETURN will cause re-execution of the command.

Possible Enhancements to ZBASIC

When Z-DOS V2.0 is released, we can hope for an updated version of ZBASIC similar to the IBM-PC BASICA V2.0. This version contains a number of very useful enhancements, especially to the graphics commands. Since IBM BASIC programs which use these features are starting to appear, it is important to recognize them and besides, it's nice to hope that we'll have these features before long, too. The following is a partial summary of some of the more interesting enhancements to BASICA V2.0.

Enhancements to existing commands:

The DRAW command supports 2 new subcommands:

TAn - turns through angle n, from -360 to +360 degrees. This feature is similar to "Turtlegraphics".

P<paint_color>,<border_color> - similar to the PAINT command.

The LINE command has a new "style" parameter:

LINE (x1,y1)-(x2,y2),<color>,B,<style>

This parameter is a bit mask value which specifies how the line is to be "patterned"; you can create dashed and dotted lines.

The PAINT command has a new "tile" string parameter:

PAINT (x,y),<tile$>

This parameter specifies how the painted region is to be "tiled". You can easily create a variety of rectangular pattern blocks.

New commands:

WINDOW {SCREEN} | (x1,y1)-(x2,y2)

This command will cause subsequent pixel oriented graphics commands to automatically scale and clip the displayed image using the defined "window". This permits "zooming" and "panning" of images.

VIEW {SCREEN}

(x1,y1)-(x2,y2),<paint_color>,<border_color>

This command is used to define one or more "viewports" or logical screens on the physical screen. You can subdivide the physical screen into separate viewports which each automatically provide scaling and clipping of pixels plotted by subsequent graphics commands.

The PMAP function is used to translate a point's physical or pixel coordinates into world or data coordinates and vice versa. The WINDOW command uses "world" coordinates and VIEW command uses "physical" coordinates.

The SHELL command permits execution of MS-DOS commands from inside BASIC, and the CHDIR, MKDIR, and RMDIR commands permit changing, creating, and deleting directories from within BASIC.

Sample Program Conversion

We will now illustrate the conversion process with a specific example. At the end of this article, you will find a sample IBM- PC BASIC graphics program which uses graphics commands and the resulting ZBASIC program which was obtained using the methods described above. This particular program conversion does not illustrate all of the possible problems you may encounter, but was selected as being a rather typical example of the conversion process.

Reference Publications

In general, the features described in this article are for ZBASIC Version 1.0 (10-Oct-82), and IBM-PC BASICA (Advanced BASIC) V1.10.

The following publications provide additional details:

IBM Personal Computer Hardware Reference Library - BASIC, Second Edition (May 1982), Version 1.10, Copyright IBM, 1981

IBM Personal Computer Hardware Reference Library - Technical Reference, First Edition (August 1981), Copyright IBM, 1981

MICROSOFT Z-BASIC (Z-DOS) Volumes I & II, Copyright by Microsoft and Zenith Data Systems, 1982

MICROSOFT BASIC-80 (CP/M-85), Copyright by Microsoft, 1979 and Heath Company, 1981

IBM-PC Version of Block-Buster Game

```
100 ' Block-Buster Game from PC-World Magazine Volume 1, #5
110 ' By Dan Illowsky and Michael Abrash
120 DEFINT A-Z : DIM BALL(10)
130 CLS : SCREEN 1,0 : COLOR 0,1
140 CIRCLE (3,3),2,3 : PAINT STEP(0,0),3
150 GET (0,0)-(5,5),BALL
160 LINE (0,0)-(319,199),2,BF
170 LINE (80,20)-(241,185),0,BF
180 FOR I=0 TO 7 : FOR J=0 TO 3
190 LINE (82+20*I,48+J*12)-STEP(18,8),((I+J) MOD 2)+1,BF
200 NEXT J : NEXT I
210 NBALLS=3 : NBRKS=32
220 PX=150 : PXINC=0
230 ' Update ball count & check for loss
240 LOCATE 2,2 : PRINT "Balls left";NBALLS
250 IF NBALLS=0 THEN LOCATE 15,13 : PRINT "You Lost!!!" : END
260 BX=80 : BY=100 : BXINC=4 : BYINC=4
270 PUT (BX,BY),BALL
280 ' Update paddle direction
290 A$=INKEY$
300 IF A$="c" THEN PXINC=5
310 IF A$="z" THEN PXINC=-5
320 IF A$="x" THEN PXINC=0
330 OLDPX=PX : PX=PX+PXINC
340 IF PX<80 OR PX>221 THEN PX=OLDPX
350 LINE (OLDPX,181)-(OLDPX+20,181),0
360 LINE (PX,181)-(PX+20,181),3
370 OLDPX=PX
380 OLDBX=BX : BX=BX+BXINC
390 IF BX<80 OR BX>234 THEN BXINC=-BXINC : BX=BX+2*BXINC
400 OLDBY=BY : BY=BY+BYINC
410 IF BY<24 THEN BYINC=-BYINC : BY=BY+2*BYINC
420 IF BY<=175 GOTO 470
430 IF BX<PX-5 OR BX>PX+20 GOTO 580
440 BYINC=-BYINC : BY=BY+2*BYINC
450 BXINC=(BX-PX)\2-4
460 ' Check for brick hit and possible win
470 PUT (OLDBX,OLDBY),BALL
480 IF POINT(BX+2,BY+2)=0 GOTO 550
490 PAINT (BX+2,BY+2),0
500 BYINC=-BYINC : BY=BY+2*BYINC
510 NBRKS=NBRKS-1
520 LOCATE 2,20 : PRINT "Bricks left";NBRKS
530 IF NBRKS=0 THEN LOCATE 15,13 : PRINT "You Won!!!" : END
540 ' Update ball position
550 PUT (BX,BY),BALL
560 GOTO 290
570 ' Account for lost ball
580 NBALLS=NBALLS-1
590 PUT (OLDBX,OLDBY),BALL
600 GOTO 240
610 END
```

Z-100 Version of Block-Buster Game

```
100 ' Block-Buster Game from PC-World Magazine Volume 1, #5
110 ' By Dan Illowsky and Michael Abrash
120 ' Converted to ZBASIC by John Stetson 09-Jul-83
130 DEFINT A-Z : DIM BALL(10)
140 ' Initialize for new game
150 CLS : COLOR 7,0 :
    ' SCREEN statement deleted & COLORs changed
160 CIRCLE (3,3),2,7 : PAINT STEP(0,0),7 : ' White color
170 GET (0,0)-(5,5),BALL : CLS : ' Extra CLS for more games
180 LINE (0,0)-(639,199),5,BF : ' Magenta color and X=2*X
190 LINE (160,20)-(482,185),0,BF : ' Black color and X=2*X
200 FOR I=0 TO 7 : FOR J=0 TO 3
210 ' X=2*X, Cyan color and Magenta->Yellow color
220 ' for better visibility on non-color Z-100's
230 LINE (164+40*I,48+J*12)-STEP(36,8),3*(((I+J) MOD 2)+1),BF
240 NEXT J : NEXT I
250 NBALLS=3 : NBRKS=32
260 PX=300 : PXINC=0 : ' X=2*X
270 ' Update ball count & check for loss
280 LOCATE 2,24 : PRINT "Balls Left:";NBALLS
290 IF NBALLS=0 THEN LOCATE 15,36 :
    PRINT "You Lost!!!" : GOTO 660
300 BX=160 : BY=100 : BXINC=8 : BYINC=4 : ' X=2*X
310 PUT (BX,BY),BALL
320 ' Update paddle direction
330 A$=INKEY$
340 IF A$="4" THEN PXINC=-10 : ' Z-100 Keypad 4 is left
350 IF A$="5" THEN PXINC=0   : ' Z-100 Keypad 5 is stop
360 IF A$="6" THEN PXINC=10  : ' Z-100 Keypad 6 is right
370 OLDPX=PX : PX=PX+PXINC
380 IF PX<160 OR PX>442 THEN PX=OLDPX : ' X=2*X
390 LINE (OLDPX,181)-(OLDPX+40,181),0 : ' X=2*X
400 LINE (PX,181)-(PX+40,181),7 : ' X=2*X
410 OLDPX=PX
420 OLDBX=BX : BX=BX+BXINC
430 IF BX<160 OR BX>468 THEN BXINC=-BXINC :
    BX=BX+2*BXINC : ' X=2*X
440 OLDBY=BY : BY=BY+BYINC
450 IF BY<24 THEN BYINC=-BYINC : BY=BY+2*BYINC
460 IF BY<=175 GOTO 510
470 IF BX<PX-10 OR BX>PX+40 GOTO 620 : ' X=2*X
480 BYINC=-BYINC : BY=BY+2*BYINC
490 BXINC=(BX-PX)\2-8 : ' X=2*X
500 ' Check for brick hit and possible win
510 PUT (OLDBX,OLDBY),BALL
520 IF POINT(BX+2,BY+2)=0 GOTO 590
530 PAINT (BX+2,BY+2),0
540 BYINC=-BYINC : BY=BY+2*BYINC
550 NBRKS=NBRKS-1
560 LOCATE 2,43 : PRINT "Bricks Left:";NBRKS : BEEP
570 IF NBRKS=0 THEN LOCATE 15,36 :
    PRINT "You Won!!!" : GOTO 660
```

☞

```
580 ' Update ball position
590 PUT (BX,BY),BALL
600 GOTO 330
610 ' Account for lost ball
620 NBALLS=NBALLS-1
630 PUT (OLDBX,OLDBY),BALL : BEEP
640 GOTO 280
650 ' Delay after end of game
660 FOR I=1 TO 3 : FOR J=1 TO 1000/1 : NEXT J : BEEP : NEXT I
670 GOTO 150
680 END
```

Changing your address?

Let us know. We don't want you to miss a single issue of REMark, send your change of address to:

Heath Users' Group
Hilltop Road
St. Joseph, MI 49085

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

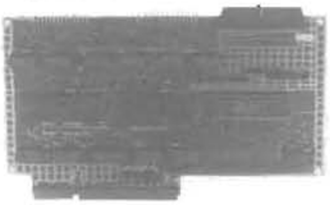
PLEASE SEND CHECK OR MONEY ORDER TO:

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

H/Z89 PERIPHERALS from SECURED COMPUTER SYSTEMS

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

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



Features:

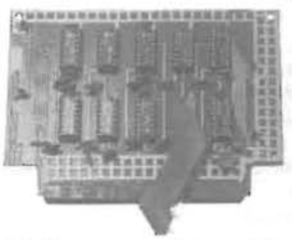
Chip independent design • Reduces computer data buss loading from 3 to 1 • Choice of Centronics or Epson parallel drivers for HDOS or CP/M • Complete documentation and installation instruction.

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

PRICE $199.00
Shipping & Handling $10.00

16K RAM EXPANSION CARD

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



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

NOW INCLUDING SUPPORT MOUNTING BRACKET

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

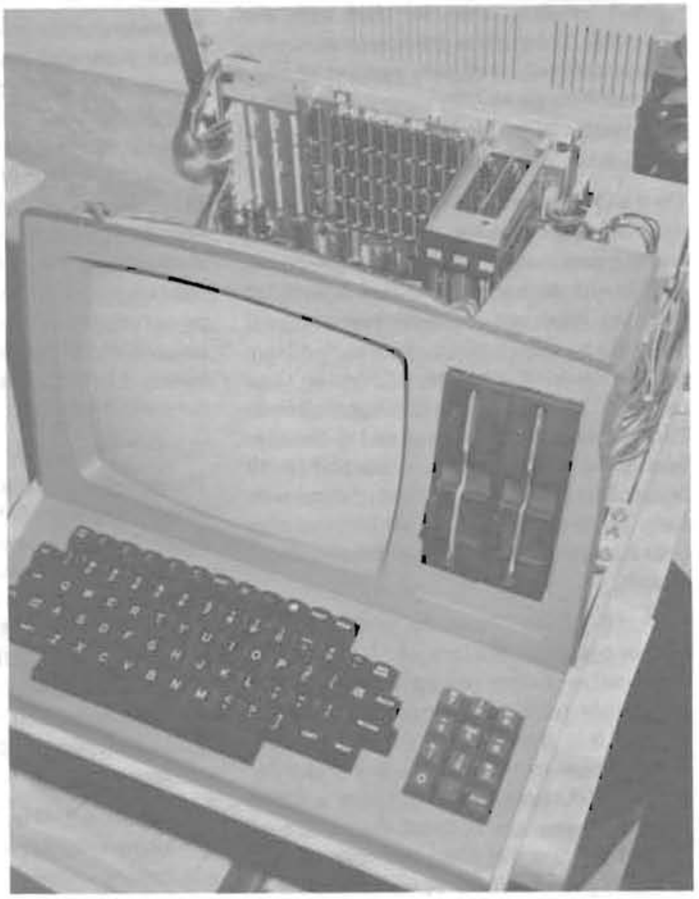
Now Only $65.00

Shipping & Handling $5.00

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
Terms and specifications subject to change without notice.

ZENITH data systems SERVICE CENTER



Turning on the SUPER 89

Walter Gillespie
REMark Editor

I had an older H19 terminal sitting around and had thought many times of turning it into an H89. Also I had noticed a modification board advertised by D-G Electronic Developments Co. of Denison, Texas, the Super 89. So a request went off to Bruce Denton and Brian Grams of D-G for one of their CPU boards.

In the mean time, I set about modifying the H19 into a computer. I won't dwell on the modification process here because the H19 to H89 kit that was offered by Heath is no longer available. You may purchase the manuals and take your chances that all the individual parts listed within might still be available (most are). (To convert the H/Z19 to an H/Z89, order manual #597-2215- 01. To convert the H/Z19A to an H/Z89A, order manual 595- 2709.) The Zenith model ZCN19 can't be upgraded because the neck of the CRT tube is longer than those used in other models. Thus, there is no room for the CPU board. Word from D-G is that they plan on a power supply upgrade kit to be available later this year. No modifications to the power supply are required for the H/Z89, H/Z89A, or Z90 except as noted in the D-G instruction manual.

The arrival of the Super 89 CPU board did little to relieve my excitement for making this modification. The materials were well packaged. Loose parts such as cable shells, screws, and nuts were in one plastic bag. Another contained a DB25 adapter cable assembly similar to those supplied by Heath for their serial board. This connector is used for the on-board serial port. A third plastic bag contained materials for implementing the D-G Electronic Disk feature of the Super 89 board. The master packing also contained 90 (three ring punched) pages that comprised the main instruction manual. Unfortunately you must provide your own binder. Also included was a small product description folder on the AM9511A Arithmetic chip that you can add to the board as an option.

The main bulk of the package consisted of two large sheets of egg-crate foam protective padding that sandwiched the new CPU board. At first sight, this board gives one a definite feeling of quality. The general layout and cleanliness along with the silk screen parts call-outs and smooth soldering are par with those offered by Heath Company.

I must say this modification is not for the "faint of heart!", but it doesn't require one to be an electrical engineer either. If there is an area that is in need of improvement, it would be the manual. First, although generally well written, it lacks the check-as-you-go boxes so familiar to Heath manuals. Second, there are a few areas that seem to jump around due to a difference in ROMs that you might encounter. While talking with Brian of D-G, I learned that they are aware of this particular problem and are working on a new D-G ROM that will not require the Heath monitor ROM or the port decoder PROM from your old H/Z89 (more on this later). Third, a board component layout sheet is almost a necessity for proper location and positioning of the many jumpers required but, one is not provided (again, D-G is aware of this and is working to include this in a new manual to be released in August along with the new ROMs).

As with all manuals, you should read through it two or three times so that you become familiar with the terms and locations used in the instructions. READING and understanding this modification is VERY important.

The Hardware

Except for a change to the cross-over cable (that's the one going from the CPU board to the Terminal Logic board), installation is very straight forward. All Heath Company peripheral boards, disk controllers (H77, H37, H47, & H67), serial I/O and parallel I/O plus many boards manufactured by other H/Z support vendors, may be used with the Super 89. Compatible add-on boards plus various methods and parts needed to convert H/Z-19's to H/Z-89's are covered in a three page addendum included with the instruction manual.

On the CPU board that I received, you were required to remove the monitor code ROM and the port decode PROM from the Heath CPU board. According to D-G, a new ROM that they have under development (should be ready by the HUG Conference) will not require the MTR-89 or 90 ROM from your old CPU. The installation of the port decode PROM requires the removal and re-installation of the memory card on the SUPER 89 board. This is not a difficult task, if done with care. The memory card has sockets that plug onto a set of pins the same as the peripheral boards.

With 16 jumper locations and some 67 jumper positions, the lack of a board layout sheet makes jumper setting difficult but not impossible. I don't claim to understand them all, but it is these jumpers that allow implementation of the features that make the D-G 89 CPU board SUPER. Using a Z80A processor, you are allowed the choice of 2(2.048)MHz or 4(4.096)MHz system clock speed. If you install the optional arithmetic processor, you may choose, through a jumper, 2 or 4MHz operation for it. Many other jumper options are available. There are too many to cover here, Jumper installation is covered adequately in the manual.

The board comes with 128K of memory installed in two banks of 9 64K RAM chips. This is expandable to 256K through two additional banks. This memory is bank-selectable and can be write protected in 4K byte increments. For the novice, the easiest use of this extra RAM (that over the 64K that CP/M or HDOS use) is with the Electronic Disk software now supplied free with the board.

Two 8 position DIP switches are provided on the board for additional system configuration. The first switch is addressed to I/O port 362Q which provides the same functions as SW501 on the Heath CPU board. The second switch is addressed as the upper four bits of I/O ports 200Q and 201Q. The first four switches on this DIP are used to set the Year Offset for use with the on-board clock. The other 4 switches are as yet undefined but could be used in future D-G products.

The installation of Disk Controller cards is the same as used with the Heath CPU Board, with special chip and cable locations for optional boards noted in the manual. I installed both the Hard Sector (H-88-1) Disk Controller and the Soft Sector (H/Z89- 37) on my unit with no problems.

Placement of the board into the '89 is no different than with the older CPU. All cable connection locations are the same with only two exceptions. The first is the power plug, it has been placed on the top right edge closer to the power supply and turned vertically. The other is the addition of the on-board serial interface plug which is directly above the terminal serial plug at the lower right corner. You may add a battery back-up power supply to the on-board clock through an unlabled three pin plug located approximately 3 inches in from the lower right hand corner of the board.

The Software

The system seems to take a little longer to give it's single ready beep on power-up. This delay is due to extensive validation and preparation concerning RAM, clock speed, and the monitor. All standard monitor commands of both the MTR-89 and MTR-90 are supported by the SUPER 89 as well as some new conventions of its own. The date and time can be set and read under the monitor command, plus a sophisticated memory test can be run.

When run at 2MHz, the system would handle all my software with few exceptions. The exceptions were important though, they consist of most all MicroSoft software (MBASIC, MS BASIC Compiler, MS Cobol, etc.). Patches for this software, using DDT, are included in the manual. After patching they ran as before. Patches are also included for running HDOS and CP/M at 4MHz. These changes, like all of the patches, must be made with the system clock set at 2MHz. When finished, you change the system clock jumper to 4MHz and re-boot. Everything should come up running ok. Patches for HDOS are to the SY.DVD and TEST17.ABS. For CP/M, patches must be made to MOVCPM5.COM, FORMAT.COM, DUP.COM, and BIOS.SYS. These patches are simple and easy to install and should not cause even an amateur hacker problems.

(The patches mentioned here will not be required when using the SUPER 89 with the new ROMs mentioned earlier.)

To install the Electronic Disk, a new BIOS must be assembled for CP/M. A second manual and a 5" disk with proper software are included to be used in this process. The changes and subsequent "MAKEBIOS" procedure are tedious and time consuming (not much different than the standard CP/M MAKEBIOS routines), but the results are well worth the effort. On my system, this additional disk is drive G:. The Electronic Disk is always the next drive after your last possible drive. I have both the hard and soft- sector controllers installed. This allows for six possible drives (A,B,C)+(D,E,F), even though I have only one hard sector drive and two soft sector drives installed. You can write to and read from this disk just as you can any other physical disk. The only limitation is that the Electronic Disk cannot be SYSGENed (bootable). With the 128K of RAM that comes installed on the SUPER 89 board (two of the four available 64K RAM banks installed), the Electronic Disk has 62K of space. Adding one or two additional banks of RAM for a total system RAM of 192K or 256K, the Electronic Disk would have 126K or 190K capacity respectively. You must reassemble the BIOS if you make any changes to memory, since it is the BIOS that carries the information for proper initialization of the Electronic Disk at BOOT.

Ultimeth Corporation (Dean Gibson) has HDOS "Cache" and "Pseudo-disk" programs available for use with the SUPER 89. The Cache program operates something like an "auto-loading" electronic disk. As writes are made to the disk, the information is also placed in the memory disk. On a "read from disk", the memory disk is first checked for that file. If it is there, it is read from the memory disk instead of from the physical disk. This greatly speeds up disk accessing. The Pseudo-disk operates in a mannner similar to the D-G Electronic Disk.

When running the system at 4MHz, MBASIC will load from the Electronic Disk in about as much time as it takes you to remove your finger from the return key. The use of this memory disk is an ideal advantage in using business software that requires a lot of reading and writing to temporary files. It, the Electronic Disk, is a new approach to the mass storage problem. It's hard to get used to not having lights blinking and clicks and whirs going on. Blip! There's your info!

Other Enhancements

To round out my new SUPER 89 system, I acquired two Shugart 5" DS/DD Thinline drives from Software Wizardry (Tom Jorgensen), St. Charles, MO. These were placed into the single full size 5" disk opening in the front of the H19. These are also available from Quikdata of Sheboygan, Wisconsin and Floppy Disk Services of Princeton, New Jersey. Mitubushi half- high 5" drives are now offered by Heathkit stores. You should check with these sources for current prices and availablity.

I replaced the backplate on my unit with one from Magnolia Microsystems, Seattle, Washington. This allows for a greater variety of possible peripheral expansion connectors. A similar type replacement backplate is available from Kres Engineering, Irvine, California.

In the area of other support, Ultimate Computer Systems, Hastings, Michigan, has available a set of "library routines" which allow the AM9511 arithmetic processor to be used for mathematical functions in Microsoft Fortran, Basic Compiler, and PL/1 programming languages. Use of the 9511, according to Bruce Denton of D-G, makes a tremendous difference in the speed and accuracy with which these operations are performed.

A simular replacement CPU board is available from Technical Micro Systems, Inc., Ann Arbor, Michigan. This could possibly be the subject of review in a future issue of REMark.

Conclusion

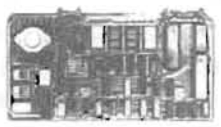
It's hard within the limits of an article to explain all of the features of a product like the D-G SUPER 89 CPU board. So, I have just touched on the highlights. But, a few other features need to be mentioned such as the Date and Time abilities. Two sample programs are included along with the Electronic Disk software. They are CLOCK and TIME. CLOCK allows you to set, change, and display the current date and time information held in memory. It should be noted that the CLOCK 25th line feature should be turned off when running some programs as it interferes. TIME is a program which displays in large, 6" high numerals the current time held in memory. A second counter in the form of a ruler progresses across the bottom of the screen until 60 seconds is reached, at this point the display is updated. To disable TIME and return to command level, you press the "erase" key on the H/Z-89.

The D-G SUPER 89 modification board is indeed super. Its features as mentioned previously, plus many advanced features, make it a very good buy at the current price of $829.00. The board is available for immediate shipment from D-G Electronic Developments Co., 700 South Armstrong, Denison, Texas 75020, or at your local Heathkit Electronics Center.

NEXT MONTH IN REMark -- THE COVERAGE OF SECOND NATIONAL HUG CONFERENCE

ATTENTION Z-100 USERS!

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



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

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

Promblaster — programs 19 different eproms from software.

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

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

Vectored from 6

Q. How do I modify my H/Z-89, Z-90 programs to run on the H/Z-100 or under ZBASIC?

A. The Z-100 codes are generally the same as the H-19 codes with several new ones. ZBASIC though has a different set of codes. The use of CRT control codes (i.e. any PRINT string character sequence containing CHR$(27) or the equivalent), can occasionally lead to unpredictable results. Where ever possible, the appropriate ZBASIC command must be used instead. A complete list of ZBASIC screen commands can be found on page 3.7 of the ZBASIC manual.

Q. Can I sell a program I have written using Heath/Zenith software?

A. In general, you can offer to sell a program that you own exclusively as long as you do not offer it for sale with any of the programs from your ZDS distribution diskettes.

For example, the source code for a BASIC program can be sold but the users are required to buy their own copy of the BASIC interpreter to run it. You are not permitted to supply a version of CP/M, ZDOS, or BASIC.

In the case of compilers and assemblers, you can offer to sell your source file, the .REL or .HEX file compiled or assembled from your source, and generally the .COM or .ABS file created by the linker or loader from your source files. The .COM and .ABS files which result from the use of Microsoft BASIC Compiler 5.3 (and later) requiring BRUN require a separate license agreement with Microsoft. The same is true for .COM and .ABS files which result from the use of the Cobol Compiler 4.6 which requires the run- time module. If the person you sell the program to wishes to recompile, reassemble, relink, or reload the program, he must purchase his own copy of the compiler to do so.

A special case occurs involving the CBASIC ® Digital Research program which requires a run-time program from the distribution diskette in order to run. According to Digital Research News, July 1982, Volume 2, No. 2, royalty requirements and the run-time module licensing have been dropped. Inquiries concerning the details of the policy should be directed to Digital Research Customer Service, 160 Central Ave., Pacific Grove, CA 93950.

Controlled Data Recording Systems Inc.

ANNOUNCING THE FDC-H8

DOUBLE DENSITY 8" AND 5.25" CONTROLLER FOR THE H8 COMPUTER

Has all of the capabilities of our popular FDC-880H controller, with the added features of;

- Direct memory access (DMA) data transfer.
- Hard sectored controller (H17) incorporated on the board.
- Runs with the standard 8080 CPU card and with Z80 CPU upgrades.
- Accesses both hard sectored disk formats and soft sectored disk formats through the same drives attatched to the FDC-H8 without hardware additions. **Price $495.00**

NEW PRODUCTS FOR THE FDC-880H

DM-1 DUAL BOARD MODIFICATION KIT **$29.95**
Allows for both the FDC-880H and the H88-4 controller cards to interface with the same 5.25" drives. Drives will run as both hard sectored format and soft sectored format depending upon the logical drive letter.

CDR BIOS by Livingston Logic Labs **$60.00**
Enhanced version of Heath/Zenith CP/M 2.203 BIOS with ZCPR. Supports all Heath/Zenith disk formats through the FDC-880H and the H17 controllers.

CDR DVD by Livingston Logic Labs. **$40.00**
HDOS driver for running double density HDOS through the FDC-880H

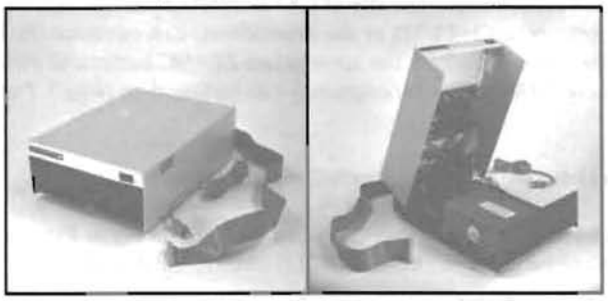
Shugart Slimline 5.25" 40 track double sided drives **$275.00**

Shugart Slimline 8" double sided drives **$525.00**

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

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

NOW 12 MEGABYTE
(CDR-10M) $2995



WINCHESTER SYSTEM
For the Heath/Zenith Computer

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

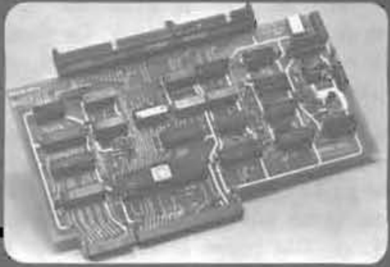
- Switching power supply
- Expansion for backup installations
- Auto attach BIOS
- Hard disk utilities
- Formatting program
- 1 year parts & workmanship warranty

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

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

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

1 CONTROLLER
FOR 8" & 5.25" DRIVES



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

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

NEW PRICE $495

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

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

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

Getting Started With Assembly Language

(Ramblings and Advice)

Pat Swayne
Software Engineer

First of all, let me apologize for not having an installment of "Getting Started with Assembly Language" in the last issue of REMark. Walt, your REMark editor, had pushed up his deadlines because the upcoming HUG Conference had pushed him up, and I just didn't have anything ready. Besides, when I look back through the last 7 or so issues of REMark at the articles and new software products I have done, I have to sit down and rest just from thinking about it. One thing is certain, though. I couldn't have done many of the new programs in anything other than assembly language.

In the last installment of "Getting Started", I said that we would take a break from learning new stuff and look at some other programs to see how they followed some of the techniques we have discussed so far. I have decided since then not to mention any specific programs, but to present some general problems that I have observed in several programs. In case you have not been following this series, we have covered console input and output, and printer output in both HDOS and CP/M. If you want to "catch up", the series starts in the April issue (#39) of REMark.

You may be wondering how a program can be messed up in such simple areas as console input and output and printer output, so let me give you a little background. I did not "grow up" on Heath computers, as many of you did (shocking, but true!). My first exposure to microcomputers was at the place where I worked before I came to HUG, on an Intel MDS 800, which is sort of like a very high priced H8, and the terminal was a Hazeltine 1500. They had both the CP/M and ISIS operating systems (ISIS is one of Intel's own operating systems). They also had remote terminals connected to a PDP-11, and since CP/M was used just for playing games, I became accustomed to the way ISIS and RSX-11 (the PDP-11 operating system) did things. One of the things I got used to was using the DELETE key to correct typing mistakes. When I pressed it (when using ISIS or a remote terminal), the last character was removed from the screen, and the cursor backed up. The fact that the character was echoed instead of deleted in CP/M didn't bother me too much since you could at least type Control-R and see what you really had entered.

Then I got my H8 with the cassette software and discovered that they wanted me to use Control-H (or BACK SPACE if you had one!) to correct typing errors instead of DELETE, while the DELETE key was used to discard a whole line. Well, it wasn't long before I had at least my own system working the way it should: DELETE deleted characters, and Control-X discarded a line. Then along came HDOS and then CP/M, and I was only mildly disappointed when they echoed characters when I pressed DELETE, because they could both be configured to delete when DELETE was pressed.

The trouble came when I started reviewing programs submitted to HUG, and found that with some of them, the DELETE key did not delete even though my system was configured that way. It was then that I discovered that programmers, even good ones, really can mess up on something as simple as console input and output. I have seen programs using every conceivable method of console I/O, including the use of MTR-88 routines and direct port access. The problem with the DELETE key is just one of the many problems that arises when console I/O is done improperly. Some problems are more serious, such as the inability of a program to work on certain systems that it otherwise should be able to work on. So I have prepared the following guidelines for console I/O in assembly language programs, and I can only hope that experience programmers as well as beginners are reading this article.

Guidelines For Programming Console I/O in Assembly Language

1. Use operating system routines for all I/O, and use the highest level routines that will do the job. For example, if you are doing line input (where the user types in a line of information and then hits RETURN to enter it), use the operating system's line input routine, rather than building up your own around the systems character input routine.

2. NEVER use monitor ROM routines or direct port access for I/O.

3. If you absolutely must break one of the above rules, inform the user in the documentation (especially in the case of the second rule), and make every effort to have your I/O routines work like the operating system. You can really irritate a potential software customer with something as simple as the way the DELETE key works.

4. Allow either upper or lower case for input expecting alphabetic responses, especially "Y" or "y" for "yes", and "N" or "n" for "no". There is probably nothing more irritating than typing "y" and having the program assume you meant "no", or spit out some nasty "illegal entry" message. A simple ANI 5FH instruction will convert any letter in the A register to upper case if it is lower case. HDOS can perform the conversion for you (see the SCALL .CONSL section of your HDOS System Programmer's Guide).

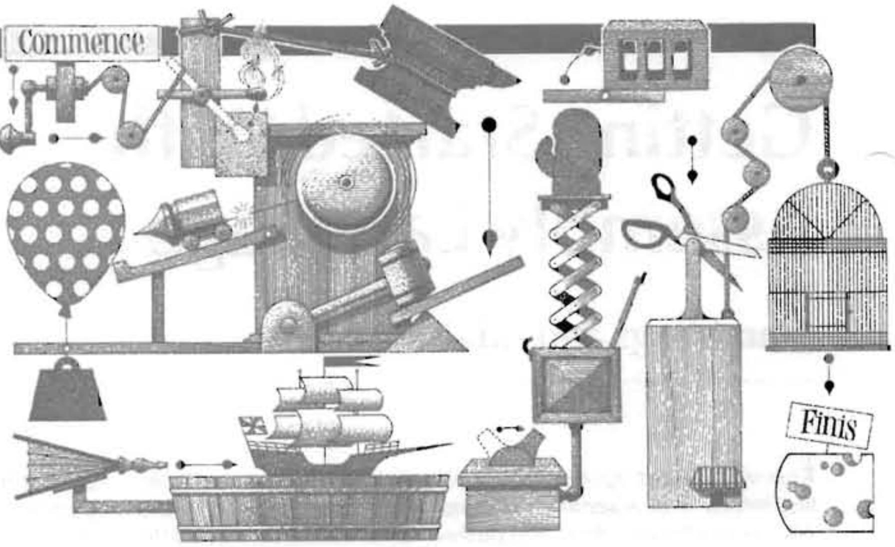
5. HDOS users, if you alter any of the input parameters such as line width, tab processing, or letter case mapping, be sure to restore things the way they were when your program ends. The same system call (.CONSL) used to set parameters can also be used to read the existing ones, which you can store and reset at the end of your program.

I must admit that in the past, I have broken some of those guidelines myself, but then there was a time when I couldn't write programs at all. We learn as we go. At least some of us do, but sometimes even the big guys get stuck in a rut. The patches to Microsoft BASIC in REMark #41 attest to that.

I guess this has not been much of an article on assembly programming. Next month I plan to have my fire re-lit, and we will start in on disk operations.

Programming Simplified

Our Words Game Continues



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

This month we will continue programming our game of WORDS. So far, we've completed its introduction and will get into the meat of it now. Figure 1 is a flow chart of the input subroutine. It begins at the connector matching the point we left off last month.

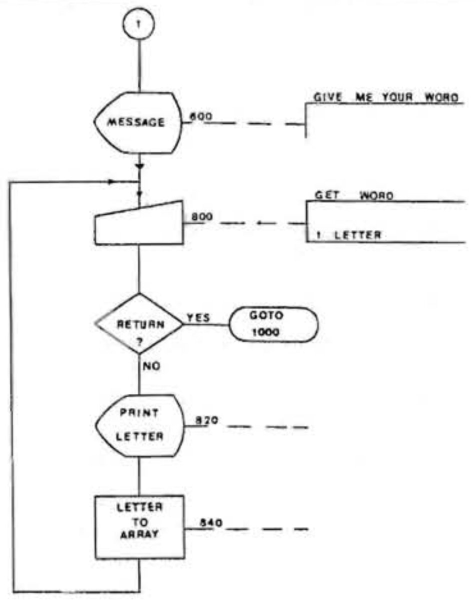


Figure 1

Begin at line 600 which asks for our input. We've got to consider everything which will be on the screen just as we did last month. Let's start on screen character 6 and line 1.

```
600 X=37:Y=32                          'CHARACTER 6, LINE 1
610 GOSUB 5060:GOSUB 5080:GOSUB 5210:GOSUB 5040:
    X=X+4  'GIVE (SPACE)
620 GOSUB 5120:GOSUB 5040:X=X+4             'ME (SPACE)
630 GOSUB 5240:GOSUB 5140:GOSUB 5200:GOSUB 5170:
    X=X+4  'YOUR (SPACE)
640 GOSUB 5220: GOSUB 5140:GOSUB 5170:GOSUB 5030          'WORD
```

That completes the instruction. Next we'll take the input. We'll also print the input in large letters. The problem is where to begin on the screen. How big will the word be? How will we center it? Putting the last question first, we won't attempt centering of the input. It can be done, but we'll leave that for later. As for the word length, we know from our work with our library program LETTERS.BAS, that most letters need four character widths. That means a maximum of twenty letters can be shown on one line. Most words (definitely all the words I can spell) are shorter than that. We'll just put the first letter directly under the left edge of our message. We'll use screen character 6 and move down to line 10.

We'll also need to know whether this letter is the first, second, etc., letter in the unscrambled word. For that we'll use the variable "A" to keep track of which letter has been input. Then we'll take the input one letter at a time. It goes like this:

```
700 X=37:Y=41                        'CHARACTER 6, LINE 10
710 A=1                              'COUNTER FOR INPUT LETTER
800 I$=INPUT$(1)                     'GET INPUT.  ASSIGN TO I$
```

We are now to the point in our program where a decision must be made. If the player presses the return key, he is finished with his input. How does the computer know that the RETURN key has been pressed? It's time to refer to our Operating Manual, Section 12, ASCII Characters. We don't need to look far until we find that CARRIAGE RETURN is decimal code 13. All we have to do is test for CHR$(13). If the RETURN key has been struck, we will direct program execution to another area where we can mix up the word and continue the game. Our program line numbers are already in the 800's and we have some additional steps to perform, so let's decide now to begin that part of the program on line 1000 when we write it. By now you should begin to recognize some of the real advantages of the flow chart and modular programming. We will have program branches, but are in complete control. Also, the return routine can't sidetrack us from completing our input routine. We've assigned it a place and can write it later. Our program continues.

```
810 IF I$=CHR$(13) GOTO 1000              'RETURN WAS STRUCK
```

If RETURN was not struck, we've got to print the letter which was. It was easy to print a letter when we just used GOSUB commands for known letters, but this could be any letter and we must direct program execution to the correct one of twenty-six possible program lines. As we scan our MBASIC manual, we find only two instructions which can direct program execution to different line numbers depending on the value of an argument or a variable. These are ON....GOTO and ON ....GOSUB, but how do we get the letters A to Z to equal the numbers 1 to 26? Again, the ASCII values of the letters will come to the rescue. Another look at Section 12 of our Operating Manual shows that the ASCII value of "A" is 65 and increases by 1 for each letter of the alphabet, so that the value of "X" is 90. But that's 64 away from the value that we want, so we'll have to get the ASCII value of the letter that was typed to the input, then subtract 64. We can use this value in an ON....GOSUB instruction to direct our program to the first line of the correct letters' graphics instructions. It looks like this:

```
820 L=ASC(I$)-64                       'CALCULATE CONTROL VALUE
830 ON L GOSUB 5000,5010,5020,5030,5040,5050,5060,5070,5080,
    5090,5100,5110,5120,5130,5140,5150,5160,5170,5180,5190,
    5200,5210,5220,5230,5240,5250
```

Now that the letter is printed, we must store it in memory so we can find and use it later. For this we'll use arrays as follows:

L$(A)	to store the letter in the unjumbled word
W$(A)	to store the letter for mixing
C(A)	to store the width of each letter in the unjumbled word
W(A)	to store the width of each letter for mixing

We can see that these are ARRAYS, and must be DIMensioned. In the last article, we left line 250 for that purpose. How large should we make these arrays? A few paragraphs ago we said that the most letters we can fit on one line of the display is 20, so that is how big our arrays will be. Once we've assigned our array variables with the input letter and the letter size taken from the control code in the letter-generating subroutine, we can increment the variable "A" which keeps track of which letter we're working with in our input word. Then we've got to direct program execution back to input the next letter. Until the RETURN key is struck, we'll stay in the loop. Check Figure 1 again. This loop operation should be clear to you. The only exit is the RETURN key decision. Let's complete our input program subroutine.

```
840 L$(A)=I$          'PUT INPUT LETTER INTO UNJUMBLED ARRAY
850 W$(A)=I$          'PUT INPUT LETTER INTO ARRAY FOR MIXING
860 C(A)=C            'PUT LETTER WIDTH INTO UNJUMBLED ARRAY
870 W(A)=C            'PUT LETTER WIDTH INTO ARRAY FOR MIXING
880 A=A+1             'INCREMENT THE LETTER NUMBER VARIABLE
890 GOTO 800          'GO BACK FOR THE NEXT INPUT LETTER
```

Don't forget to dimension the arrays.

```
250 DIM L$(20),W$(20),C(20),W(20)        'DIMENSION ARRAYS
```

Now relax, turn on your computer, make sure last month's WORDS.BAS is on your disk, and call up MBASIC. Then type lines 600 to 640, 700, 710, 800 to 890, and 250. THIS TIME DON'T SAVE "WORDS"!!! You'll overwrite last months work and lose it. Instead type:

SAVE "WORD2",A

Then type: NEW

That will clear the memory. Now type:

LOAD "WORDS(IN HDOS IT'S - LOAD "WORDS")

Finally type: MERGE "WORD2"

Now you can: SAVE "WORDS",A

The reason we did it this way is that the file MERGEd will overwrite any lines with similar numbers in the original program. If the old temporary line 600 used for testing last month was in your program, this will replace it with the new line 600 you just added. If you had just MERGEd "WORDS", the old line 600 would have overwritten and thus replaced your new work.

Once again we're almost ready to test a new subroutine, and again we'll need a temporary line to supply a procedure the completed program will give us later. We must:

1) Exit HOLD SCREEN Mode
2) Exit GRAPHICS Mode
3) Turn on the cursor
4) END program execution

We'll use line 1000 as our temporary line because that's where the RETURN key is supposed to direct execution. Add line 1000 and type RUN.

```
1000 PRINT E6$;E8$;E4$:END
```

If all goes well, and it should, your computer will print the two messages it did last month, then it will ask for a word. When it does, any CAPITAL LETTER you type will be displayed on the screen until you press the RETURN key. Then the program will end but your word will stay. At this point in our program development, only capital letters may be typed. Anything else will not be understood by the program as we have written it, and will cause an operating error. Go ahead and try it to see what it does. You won't hurt anything. This is what I was talking about in past articles when I said that good programs can be made better. We'll take care of this shortcoming in a later article. For now, stick with the CAPS LOCK key and type letters only. We've got another subroutine to write.

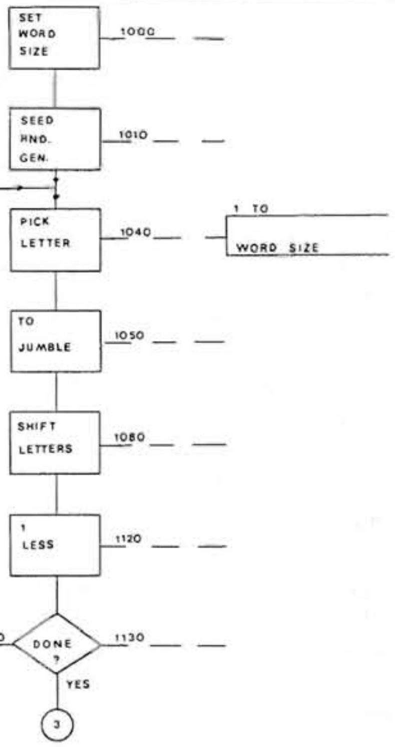


Figure 2

The flow chart of Figure 2 shows how we'll mix up the word that was input. This will give us the jumbled letters the players must sort out. Because the variable "A" has been incremented after the last letter was input, it is one greater than the length of the word. To keep track of how long our word is, we'll use the variable "F".

```
1000 F=A-1                'SET VARIABLE F TO LENGTH OF WORD
```

Now we've got to pick a letter at random from the word. It will be the first letter of the jumble. We can't use a constant or a fixed formula because words of the same length would always mix the same way. So, we'll use the computer's random generator. It can pick a different random number every time, right? Wrong!!! These computers are so darn perfect that if you type the same word 5 times, it'll mix the same way every time, but there is a way to get around that. It's called seeding the random generator. Refer to your MBASIC manual, Section 3, for the RND function, and Section 2 for the RANDOMIZE command. In order to provide a different seed for each word, and one the players can't control, we'll use READ and DATA commands. We'll put 21 different "Prime Numbers" in our data bank so no two numbers have a mathematical relationship. We'll also include a RESTORE command so there's no way to run out of seeds for the random generator. Be sure to read the READ, DATA, and RESTORE commands in Section 2 of your MBASIC manual.

The program lines we'll need are:

```
1010 READ B                          'GET A NEW SEED NUMBER
1020 IF B>0 THEN RESTORE:GOTO 1010
    'IF WE'RE AT END OF DATA START OVER
1030 RANDOMIZE B                     'RESEED RANDOM GENERATOR
1040 E=INT(F*RND(1))+1
    'GET RANDOM NUMBER BETWEEN 1 AND WORD LENGTH
6000 DATA -3,-5,-7,-9,-11,-13,-19,-23,-29,-31,-37,-43,-47,-51
6010  DATA -57,-61,-67,-73,-79,-89,-97,1
```

Be careful with the last data entry. There's no minus sign. That's so it can trigger line 1020 to RESTORE, which makes line 1010 start to READ from the beginning of the DATA again.

We now have a random number between 1 and the length of the word. That's the number of the letter we'll pick to add to our jumble. Just as we use the variable "F" to keep track of the length of the un-jumbled word, we'll use "G" for the length of the mixed-up word. Again, we'll need arrays for the length of each letter. We'll use M$(G) and M(G). We'll have to add these new arrays to the DIMension statement on line 250.

Here's the logic for our mix routine in plain English. First, we number each letter in the word. Then we pick a number at random. It must be between 1 and the number of letters in the word. We'll take the letter matching that number and add it to the jumble. We'll keep track of both the letter and its size so we can print it in special graphics. We'll use arrays to store each of these things as follows:

L$ and C	for the original word
W$ and W	temporary use to take letters for the jumble
M$ and M	for the jumbled word

Our program continues.

```
1050 G=G+1                   'INCREMENT LETTER NUMBER IN JUMBLE
1060 M$(G)=W$(E)             'LETTER IN JUMBLE IS RANDOM LETTER
1070 M(G)=W(E)               'KEEP TRACK OF LENGTH OF LETTER
```

At this point, we've taken a letter and left a hole in the middle of the word. How do we prevent trying to pick that letter again? (It's actually still there in the array W$.) We simply take each letter in a position after the one removed and move that letter down one space. For example, if we had a 5 letter word and took letter three ,W$(3), we'd just have to tell the computer that now W$(3) equals W$(4). That would move one letter down. The next step would be W$(4)=W$(5). Then we'd change our variable "F" to be one less so our program would work on a shorter word made up of only the remaining letters. After that we'd get another random number until all the letters were used. We'd know this when F=0.

This procedure will find a lot of use in games and computer aided education. Any time you take one item from a list and want the rest available on a random basis, you can do it with the routine just described, whether the list is states and capitols or the cards in a deck for poker or blackjack. Keep this routine on file, you'll use it often. This is how to put the explanation into program steps.

```
1080 FOR U=E TO F-1         'FROM RANDOM LETTER TO END OF WORD
1090 W$(U)=W$(U+1)          'MOVE ONE LETTER DOWN
1100 W(U)=W(U+1)            'MOVE SIZE OF LETTER TOO
1110 NEXT U                 'GO BACK AND CONTINUE TO END OF WORD
1120 F=F-1                  'ONE LESS LETTER LEFT TO MIX
1130 IF F>0 GOTO 1040       'PICK ANOTHER RANDOM TILL ALL USED UP
```

Now take another break, look over the subroutine flow chart and the program lines we've just completed to be sure you understand what we're doing. Then type lines 1000 thru 1130. Also type the DATA fields in 6000 and 6010. Finally update the DIM line 250 to read:

```
250 DIM L$(20),W$(20),C(20),W(20),M$(20),M(20)
```

As before, pick a name for this work and save it. SAVE "MIX",A will do nicely. Then, just as we did before, type NEW followed by LOAD "WORDS" and MERGE "MIX". You've now added another module to your program.

Normally when we create a subroutine like this, we test it before going any farther. This time, however, there's no output indication of what was done. So we'll write the next subroutine which prints out the jumble of letters we just created and we'll test both sections together. Be sure to SAVE "WORDS",A.

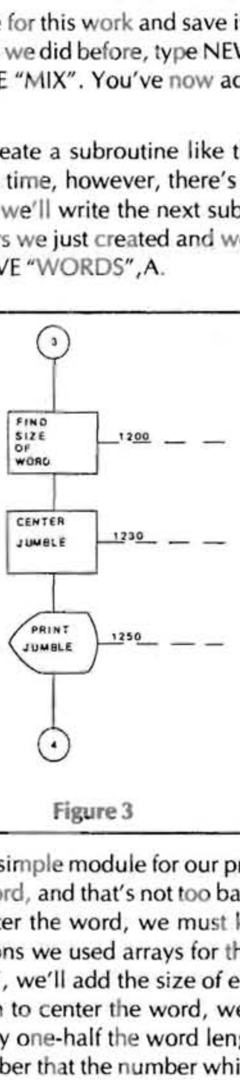


Figure 3

Figure 3 shows a short simple module for our program. The toughest part is centering the word, and that's not too bad, so let's get on with it. Before we can center the word, we must know how long it is. That's one of the reasons we used arrays for the size of each letter. Using the variable "H", we'll add the size of each letter to see how long the word is. Then to center the word, we'll back up from the middle of the screen by one-half the word length. For direct cursor addressing, we remember that the number which instructs the computer is 31 plus the position number on the screen. That means that to print in the middle of the screen at position 40 we use the number 31+40 or 71. We'll set X at 71 minus one-half the word length. We'll begin printing on screen line one to allow room for instructions, player input, and the correct word as it is figured out. Finally, this subroutine will take each letter of the jumble and use ASCII values with an ON....GOSUB statement to print each letter just as we did on lines 820 and 830 for our input. Our program looks like this:

```
1200 FOR H=1 TO A-1   'FOR EACH LETTER OF THE WORD
1210 J=J+C(H)         'ADD THE LENGTH OF A LETTER
1220 NEXT  H          'CONTINUE FOR EACH LETTER IN  THE  WORD
1230 X=71-(J/2):Y=32  'CENTER WORD HORIZONTALLY, LINE 1
1240  FOR  K=1 TO A-1 'FOR EACH LETTER OF THE  JUMBLE
1250 N=ASC(M$(K))-64  'CALCULATE VALUE FOR  ON....GOSUB
1260 ON N GOSUB 5000,5010,5020,5030,5040,5050,5060,5070,5080,
     5090, 5100,5110,5120,5130,5140,5150,5160,5170,5180,5190,
     5200,5210,5220,5230,5240,5250         'PRINT THE LETTER
1270 NEXT K                'CONTINUE TO THE END OF THE WORD
```

Now type lines 1200 thru 1270 and add them to the existing WORDS program as we did with the last two modules. Be careful not to erase or lose what you did in the past. In fact, it would be a good idea to make a backup copy of your work at the end of each session. Your program is getting rather large and there's a lot of time and effort invested. Making backups of important work is a very good habit to get into.

We've completed another module and this time we can test our work. Debugging has been made easier because we know that if there is a problem, it must be in one of the last two modules added. All the others have been tested before. We've also got our flow charts to aid in checking program execution. As in all our previous tests, we'll have to add a temporary line to end execution and get our terminal back to normal. Add:

```
1300 PRINT E6$;E8$;E4$:END
```

Now type: RUN

When the screen says "GIVE ME A WORD" type: HEATH and <RETURN>

Do you know what went wrong? I intentionally left out an instruction. Before you read any farther, try to figure out what you need to fix the program.

If you said that we didn't erase the screen before we printed the jumble, you're right. You're also correct if you want to add the instruction PRINT E2$. The best place to add it is between lines 1220 and 1230, so type:

```
1225 PRINT E2$
```

Now type: SAVE "WORDS",A

Don't let all the junk on the screen bother you. If you really want to clear it off, press the "OFF LINE" key so it latches down. Then hold the "ESC" key down while you type an "E". Press the "OFF LINE" key so it pops up. What you just did was disconnect the computer from the terminal. Then you sent the Clear Display code "ESC E". Now we're back on line so we can add line 1225 and SAVE WORDS if you didn't do it before. Type RUN again. When asked for a word this time, type HEATH again. Then press RETURN. This time it should have worked fine. You should have a jumble of the five letters from HEATH on the screen.

This is a good time to make a backup copy of your file "WORDS.BAS". You can also RUN the game as often as you like and try any word you want. Next month we'll add the rest of the subroutines to give you a complete game. So far it's got over 240 lines and takes up about 10K on your disk. You've come a long way and were never lost thanks to flow charting and testing each module as you wrote it.

Please remember the articles in this series are copyrighted by the author and the programs are copyrighted by APPLIED COMPUTING. You may use them all you want, but may not sell or give them to anyone else.

If you have any questions about what we've done this month, or suggestions for a way we can improve this column, be sure to write the author at the address above. If you require an answer, please include a self-addressed stamped envelope. See you next month.

PASCAL/MT+™
FOR HDOS

- VERSION 5.5 [LATEST VERSION]
- USES STANDARD HDOS DEVICE DRIVERS
- ISO STANDARD PASCAL
- EMITS NATIVE 8080 OR Z80 CODE
- ALL HDOS SYSTEM CALLS CAN BE MADE FROM PASCAL
- AUTOMATIC CHANNEL ALLOCATIONS
- MODULAR COMPILATION
- CHAINING FROM ONE PROGRAM TO ANOTHER SUPPORTED
- UP TO 225 OVERLAYS SUPPORTED

- SELECTABLE BCD REALS [FOR BUSINESS APPLICATIONS] OR FLOATING POINT REALS [FOR SCIENTIFIC APPLICATIONS]
- FLOATING POINT REALS MAY BE EITHER SOFTWARE OR HARDWARE [AMD 9511 / INTEL 8231]
- SUPPORTS PACKED RANDOM FILES
- RELOCATING LOADER
- SYMBOLIC DEBUGGER [INCLUDES BREAKPOINTS]
- LIBRARIAN
- DISASSEMBLER
- INTERRUPT LEVEL PROCEDURES

- GENERATES ROMABLE CODE
- INLINE MINI-ASSEMBLER ALLOWS YOU TO WRITE ASSEMBLY LANGUAGE CODE AS PART OF THE PASCAL/MT+ SOURCE CODE
- PROCEDURES MAY BE DECLARED EITHER RECURSIVE OR STATIC [STATIC PROCEDURES EXECUTE CONSIDERABLY FASTER THAN RECURSIVE PROCEDURES]
- NO ONGOING CHARGES FOR RUN TIME PACKAGES

PRICE **$ 350**
SPECIFY H-17/37/47

PASCAL/MT+ IS A TRADE MARK OF DIGITAL RESEARCH

NEW ORLEANS
GENERAL DATA SERVICES, INC
7230 CHADBOURNE DR
NEW ORLEANS LA 70126
(504) 241-9495

HUG NEW PRODUCTS

885-1110 HDOS AUTOFILE Data Base (Z80 only) $30.00

This is an announcement that AUTOFILE has been updated to version 3.0. AUTOFILE is a data base management system (DBMS) for organizing, filing, and retrieving information.

The functional description in the HUG Software Catalog is still accurate. No function has been removed. This version has significant new features, two of which are:

1) The ability to search multiple data bases with one definition or a search argument. A dismount/mount feature makes it possible to search as many data bases as the user wishes.

2) A significant feature for unloading and deleting items, controlled from the hit list display. This feature makes it much easier to modify data base content.

This version of AUTOFILE is completely compatible with version 2.0. The documentation has been enhanced to clarify program function for the benefit of users having little prior experience with data base systems.

AUTOFILE requires HDOS 2.0 on an H/Z-89 with 48K of memory. It makes extensive use of the Z80 instruction set, and therefore will run on H/Z-89s or H8 Z-80 based systems only.

Note: The programs are written using the Zilog mnemonics. The source files and .ACM files are included.

To anyone wishing to update from AUTOFILE version 2.0 to 3.0, send your original two disks and $10.00 to Nancy Strunk, HUG, Hilltop Road, St. Joseph, MI 49085.

885-8020[-37] CP/M RF Computer-Aided-Design Package ... $30.00

Introduction: RFCAD is a collection of 35 programs written in Microsoft BASIC which aid the user in designing many standard audio, HF, VHF, UHF, and microwave circuits and antennas. These programs are made readily accessible by use of a user-friendly menu. This package is primarily aimed at electronic engineers and Ham's who have frequent need to design circuits and antennas. It will also be of tremendous benefit to electronic students who want to experiment with circuit configuration to see what happens. Several programs are also included which will be helpful to anyone interested in setting up a satellite earth station.

Requirements: These programs require the CP/M operating system on an H8/H17/H19, H/Z89 or Z90 with 40K of memory. Two 5 1/4 inch single-sided drives are required to run from the menu program. A printer device is not required.

The programs are written in Microsoft BASIC and will run on version 5.03 or newer. The only ESCape code or cursor control used for the H19 terminal is the "clear screen". At least 11K free bytes of MBASIC program memory space are required by the largest program.

Note: The H19 terminal is required for the clear screen control used in the program. The clear screen code can be changed so that the H19 terminal would not be required.

The following files are included on the HUG P/N 885-8020[-37] CP/M RF Computer-Aided-Design package. (This is a two disk set.)

```
CADMENU .BAS        MFILTER   .BAS
AELIPT  .BAS        MOD       .BAS
AFILTER .BAS        MSCONV    .BAS
AFPLOT  .BAS        MSTRIP    .BAS
BFILTER .BAS        NE555     .BAS
CHORN   .BAS        NFCONV    .BAS
COIL    .BAS        OSC       .BAS
DBCONV  .BAS        PATH      .BAS
DISH    .BAS        PELIPT    .BAS
FED     .BAS        PFILTER   .BAS
FREQWAV .BAS        RESNET    .BAS
HELIX   .BAS        RESONNCE  .BAS
HELP    .BAS        SATANT    .BAS
HFILTER .BAS        WNDLD     .BAS
HORN    .BAS        TLINE     .BAS
LFILTER .BAS        YAGE      .BAS
LPI     .BAS        ZCONV     .BAS
MATCHER .BAS
```

Author: Gary A. Field, WA1GRC

Program Content: The subjects covered in the RFCAD program list are as follows.

Active filters,
Passive filters,
Pi and Tee attenuators,
Power dividers,
L, Pi & Pi-L matching networks,
NE555 circuits,
Yagi, Horn, Helix and Dish antennas,
Microstripline matching,
Receiver front end design,
Wind loading on antennas,
Radio path loss calculator,

Beam heading and distance to remote station,
Oscillators, and
Inductor design.

There are also several handy conversion utilities which convert between electrical units such as dB, dBm, dBmV, Watts, Volts, VSWR, and degrees Kelvin, as well as dimensional units like meters, centimeters, feet, and inches.

The documentation for RFCAD contains schematic diagrams which are referred to by the programs. This package has been in use in an R.F. Research and Development group for over a year and is believed to be thoroughly debugged.

885-8021 HDOS
Student's Statistics Package $20.00

Introduction: This package is a collection of programs which conveniently calculate the most needed basic statistics. The programs cover the most frequently used statistical analysis covering most of the designs included in introductory statistics textbooks. The programs are menu driven.

Requirements: These programs require the HDOS operating system version 2.0 on an H8/H17, H/Z-89, or Z90 with 48K of memory. The menu program is written for a two 5 1/4 inch single-sided drive system, however, the user has the option of selecting one disk drive in the menu program. A line printer is required for hardcopy output.

The programs are written for either Benton Harbor BASIC or Microsoft BASIC.

Note: The H19 terminal is not required for these programs. An 80 character by 24 line terminal is assumed.

The following files are included on the HUG P/N 885-1021 HDOS Student's Statistics Package:

STATMENU	.BAS	TTEST	.BAS
BASTAT	.BAS	A1	.BAS
BASALT	.BAS	A1R	.BAS
CORLTN	.BAS	A2	.BAS
LINREG	.BAS	A2R1	.BAS
CHISQR	.BAS	NORRND	.BAS
STDSCR	.BAS		

Author: Theordore J. Stolarz

Program Content: The following is a brief description of each program available in this statistics package:

BASTAT — Basic Statistics for a Single Variable - This program computes basic descriptive statistics and builds a frequency distribution and graph of the data with expected frequencies calculated from the normal curve.

BASALT — Alternate for program BASTAT - This is a less elaborate program than BASTAT and will handle data that may not run on BASTAT. It does not provide a frequency distribution or graph of the output.

CORLTN — Linear Correlation - This program provides a correlation matrix for 2 to 20 variables and provides both Fisher's Z' for each value of r and a test for the significance of each correlation coefficient.

LINREG — Linear Regression (2 variables) - This program provides both regression equations for a 2 variable correlation problem. It allows for predicting unknown values of both X and Y using the equations. It will accept raw data for input or previously calculated statistics.

CHISQR — The CHI SQUARE Statistic - This program calculates the value of Chi Square for both the Row only (1xC) and the contingency table designs. Yate's correction for discontinuity is provided as an option where appropriate.

STDSCR — Standard Score Conversion Program - This program converts a set of scores to Z scores or standard scores with any base. It also provides descriptive statistics on entered scores.

TTEST — T-tests for Mean Differences - This program calculates the value of Student's t for both independent and dependent samples. It provides degrees of freedom and descriptive statistics for the samples.

A1 — Single Factor Analysis of Variance - This program provides a single factor ANOVA on any number of independent groups with any number of subjects. For unequal sample sizes, a choice of the weighted or unweighted means analysis is provided.

A1R — Single Factor within Subjects Analysis of Variance - This program provides a single factor ANOVA for the within subjects or repeated measures design.

A2 — Two Factor Analysis of Variance - This program provides a factorial ANOVA for the two factor design. The unweighted means analysis is used for unequal sample sizes.

A2R1 — Two Factor Mixed Design Analysis of Variance - This program provides an ANOVA on the "split plot" or mixed within subjects - between subjects two factor design.

NORRND — Normal Distribution of Random Numbers - This program provides pseudorandom numbers selected from a population with a specified mean and standard deviation. Also generates the usual rectangular distribution of random numbers.

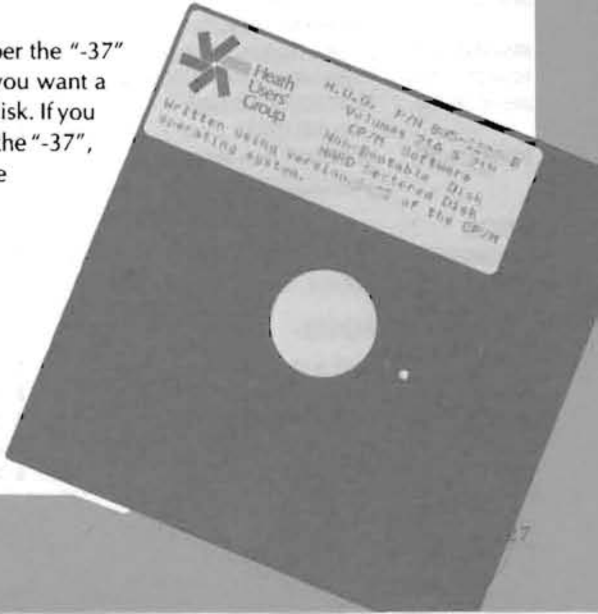
The documentation contains sample sessions using the programs, as well as example output from the execution of the program.

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

885-1092[-37] HDOS RDT Relocating Debugging Tool
885-1111[-37] HDOS MBASIC Games
885-1055[-37] HDOS MBASIC Inventory
885-1022[-37] HDOS HUG Editor (ED)
885-1113[-37] HDOS Fast Action Games

Refer to the HUG Software Catalog for descriptions of these products.

Please remember the "-37" indicates that you want a soft-sectored disk. If you do not include the "-37", you will receive hard sectored.



HUG Price List

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

Part Number	Description of Product	Selling Price	REMark Issue
HDOS			
885-1029[-37]	Disk II Games 1 H8/H89	$ 18.00	40
885-1038[-37]	Wise on Disk H8/H89	$ 18.00	42
885-1042[-37]	PILOT on Disk H8/H89	$ 19.00	42
885-1044[-37]	Utilities Disk VI	$ 18.00	43
885-1060[-37]	Disk VII H8/H89	$ 18.00	40
885-1062[-37]	Disk VIII H8/H89 (2 Disks)	$ 25.00	40
885-1064[-37]	Disk IX H8/H89	$ 18.00	42
885-1067[-37]	Disk XI H8/H89 Games	$ 18.00	40
885-1071[-37]	MBASIC SmBusPk H8/H19/H89	$ 75.00	41
885-1078[-37]	HDOS Z80 Assembler	$ 25.00	42
885-1079[-37]	Page Editor PAGED	$ 25.00	43
885-1083[-37]	Disk XVI Misc. Utilities	$ 20.00	43
885-1086[-37]	Tiny HDOS Pascal H8/H89	$ 20.00	40
885-1089[-37]	Disk XVIII Misc H8/H89	$ 20.00	41
885-1090[-37]	Disk XIX Utilities H8/H89	$ 20.00	41
885-1093[-37]	Dungeons and Dragons	$ 20.00	43
885-1097[-37]	MBASIC Quiz Disk H8/H89	$ 20.00	41
885-1107[-37]	HDOS Data Base System H8/H89	$ 30.00	42
885-1108[-37]	HDOS MBASIC Data Base System	$ 30.00	41
885-1112[-37]	Graphics Games Disk	$ 20.00	43
885-1121	Hard Sectored Support Package	$ 30.00	37
885-1122	MicroNET Connection	$ 16.00	37
885-1123	XMET Robot Cross Assembler	$ 20.00	40
885-1124	HUGMAN & Movie Animation Pkg	$ 20.00	41
885-1125	MAZEMADNESS	$ 20.00	41
885-1126	HDOS UTILITIES by PS:	$ 20.00	42
885-8015	TEXTSET Formatter	$ 30.00	42
885-8016	Morse Code Transceiver Ver 2.0	$ 20.00	41
885-8017	HDOS Programmers Helper	$ 16.00	42
CP/M			
885-1211 [-37]	Sea Battle	$ 20.00	36
885-1222 [-37]	Adventure	$ 10.00	36
885-1223 [-37]	HRUN HDOS Emulator	$ 40.00	37
885-1224 [-37]	MicroNET Connection	$ 16.00	37
885-1225 [-37]	Disk Dump and Edit Utility (DDEU)	$ 30.00	38
885-1226 [-37]	CP/M Utilities by PS:	$ 20.00	38
885-1227 [-37]	CP/M Cassino Graphic Games	$ 20.00	38
885-1228 [-37]	CP/M Fast Action Games	$ 20.00	39
885-1229 [-37]	XMET Robot Cross Assembler	$ 20.00	40
885-1230 [-37]	CP/M Function Key Mapper	$ 20.00	42
885-1231 [-37]	Cross Reference Utilities for MBASIC	$ 20.00	43
885-3003 [-37]	ZTERM Modem Package	$ 20.00	36
885-8012 [-37]	Modem Appl. Effector (MAPLE)	$ 35.00	36
885-8018 [-37]	FAST EDDY Text Editor and BIG EDDY	$ 20.00	43
885-8019 [-37]	DOCUMAT Formatter and DOCULIST	$ 20.00	43
ZDOS			
885-3004-37	ZBASIC Graphic Games Disk	$ 20.00	37
885-3005-37	ZDOS ETCHDUMP	$ 20.00	39
MISCELLANEOUS			
885-0004	HUG 3-Ring Binder	$ 5.75	
885-4001	REMark VOLUME 1, Issues 1-13	$ 20.00	
885-4002	REMark VOLUME 2, Issues 14-23	$ 20.00	
885-4003	REMark VOLUME 3, Issues 24-35	$ 20.00	
885-4600	Watzman/HUG ROM	$ 45.00	41

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

Special Note: We have found that some members are purchasing HDOS disks and trying to "run" them on CP/M when they do not have the HDOS operating system or HRUN (P/N 885-1223[-37]), the HDOS emulator for CP/M. Please read carefully the requirements for a particular product before you order. Make sure your system (hardware and software) matches what is specified in the abstract.

Back issues of REMark are available in bound volume form at $20.00 each as follows:

Volume 1	1978-1980	Issues 1-13	Part #885-4001
Volume 2	1981	Issues 14-23	Part #885-4002
Volume 3	1982	Issues 24-35	Part #885-4003

Send orders to:
Heath Company
Attn: Parts Dept.
Benton Harbor, Mi 49022

HONOR, GLORY — AND CASH

Many of you have heeded the pleas for help from the people of Schedar and Almach in GRAVITRON and GRAVITRON II. The evil Cephans are under attack on Heath/Zenith screens across the country, and we're pleased.

There's just one problem. The Cephans are winning. We've heard from distressingly few people who've completed the entire mission.

To correct that situation, we'll send $10 to the next five registered owners who complete either game, and $50 to the first owner to complete both. To qualify, send us your winning score and the message that appears at the end of the game.

Go ahead. Try it. Where else can you save an entire civilization -- and pick up a little money in the process?

GRAVITRON and GRAVITRON II require H/Z89, Z90, H8-H19, or Z100 with HDOS, CP/M, or CP/M-85 and 48k. $19.95 each + $1.50 shipping. Free catalogue available.

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

Creating a Disk Master Catalog

Frank E. Hutchison
Mary Lynn Hutchison
5327 Edgewater Drive
Ewa Beach, HI 96706

```
                          DISK MASTER CATALOG

110  Check register

     Name     .Ext     Size      Date            Flags    24-Dec-82

     CHKONE   .BAS     11       06-Nov-82
     CHKTWO   .BAS     7        16-Nov-82
     CHKTHR   .BAS     24       16-Nov-82
     CHECK    .BAS     6        10-Dec-82
     CHECK    .DOC     6        28-Oct-82
     PIE      .ABS     21       07-Apr-82
     CHKACCT  .DAT     9        22-Dec-82
     INFO     .DAT     2        22-Dec-82
     RECONCIL.DAT      14       10-Dec-82

        9 Files, Using 100 Sectors (264 Free)

10  Basic Diskette (Church)

     Name     .Ext     Size      Date            Flags    24-Dec-82

     COUNT    .BAS     8        11-Dec-82
     UPDATE   .DAT     3        20-Dec-82
     CHURCH   .DOC     5        28-Nov-82
     ROSTERAL.DAT      138      20-Dec-82
     YOUNGWMN.CAL      9        20-Dec-82
     PRNTRSTR.BAS      22       19-Dec-82
     ROSTERMZ.DAT      76       20-Dec-82

        9 Files, Using 268 Sectors (96 Free)

160  Writing One

     Name     .Ext     Size      Date            Flags    24-Dec-82

     SIDEBAR  .ONE     3        21-Aug-82
     FOLLOWUP.ENS      4        07-Jul-82
     IDEA     .FIL     46       07-Jul-82
     ARTICLE  .ONE     54       21-Oct-82
     WRITERS  .GUI     11       07-Jul-82
     COVERLET.ENS      3        28-Oct-82
     ARTICLE  .TWO     44       16-Jun-82
     ARTTWO   .NOT     4        26-Mar-82
     DADDY    .ART     55       28-Oct-82
     WAITING  .GME     14       15-Nov-82

        15 Files, Using 282 Sectors (80 Free)
```

Figure 1. With a Disk Master Catalog, you'll find the right disk the first time.

Have you ever wanted one file from among your many disks but couldn't remember which disk it was on? You mounted a disk, checked its directory, dismounted the disk, and repeated the process until you found the file. This procedure is annoying if you have a few disks. As your disk library grows, it quickly becomes both time-consuming and frustrating.

Keep track of all your files easily with software that's already at your fingertips. A Disk Master Catalog can be created using the Peripheral Interchange Program (PIP) found in HDOS and a line or text editor.

(If you are familiar with PIP and its commands, you can skip the next section which briefly covers the commands of PIP necessary to create a Disk Master Catalog.)

PIP

The Peripheral Interchange Program (PIP) is a very powerful file-manipulating tool found in HDOS. For a full understanding of what it can do, you should STUDY your user's manual. With PIP you can delete, write over, or rearrange files with ease. We strongly advise that you keep back-ups of all your files when you start to use PIP. Mistakes can be deadly.

To create a Disk Master Catalog, you must know how to access a disk directory, how to copy files, and how to concatenate (join or combine) files with PIP.

1. To access a disk directory - place PIP in command mode. At the system prompt (> for HDOS), type in PIP and hit RETURN. The system prompt will be replaced by:

```
:P:
```

To list the non-system files on the disk, enter the designation of the drive on which it is mounted followed by /L. For example,

```
:P:SY1:/L
```

will list the non-system files on the disk in drive SY1:. If you want a list of the system files, use the command:

```
:P:SY1:/S
```

2. To copy files - the PIP command used to copy a file onto another disk (or the same disk) is:

```
:P:Destination=Source
```

Destination and Source are in the form DVn:filename.ext, where DVn: is the drive designation (SY0:,SY1:, etc.), and filename.ext is any valid file name and extension. See your user's manual for rules concerning file names and extensions.

3. To concatenate files - creating one file out of several is accomplished by the command

```
:P:Destination=Source1,Source2,Source3,...
```

where Destination, Source1, Source2, Source3, etc., follow the file name conventions above.

Disks may be mounted and/or dismounted without hanging up PIP. To mount a disk, use :P:DVn:/MOU. To dismount, use :P:DVn:/DIS. To dismount a disk and remount another with a single command, use :P:DVn:/RES, then remove the disk, and replace it with another.

Creating Your Disk Master Catalog

(The following discussion assumes a system with multiple drives. The procedure for systems with a single drive will follow.)

Creating a Disk Master Catalog is accomplished in three steps:

1. Put PIP in command mode. With the catalog destination disk in SY0:, and the disk to be cataloged in SY1:, the command

```
:P:SY0:CATALOG1.DAT=SY1:/L
```

will place the directory of the disk in SY1: in a file named CATALOG1.DAT on the catalog disk in SY0:.

```
:P:SY1:/RES
```

will allow you to replace the disk in SY1: with another disk. Repeat the process, changing the file name each time, until all your disks' directories have been read onto the catalog disk.

2. When all the directory files are on one disk, the command

```
:P:MASTCAT.DAT=CATALOG1.DAT,CATALOG2.DAT,CATALOG3.DAT
```

will combine them into one file to create a Disk Master Catalog.

Because PIP will not allow the same file to be used as both a source file and a destination file, if you have too many source files to list on a single line, you must alternate a temporary file name as the destination. For example:

```
:P:CATTEMP.DAT=MASTCAT.DAT,CATALOG4.DAT,CATALOG5.DAT
```

3. You are now ready to edit your directory into final form. MASTCAT.DAT does not contain a label for each disk's directory. Labels may be added with the line or text editor of your choice. Access MASTCAT.DAT with your editor and insert the disk labels and ID numbers at the appropriate places in the file.

Single Drive Systems

If your system has only one disk drive, you can still create a Disk Master Catalog by modifying step 1 and adding another step before step 2.

In step 1, use PIP to place the source disk's directory file back onto the source disk with the command:

```
:P:CATALOG.DAT=SY0:/L
```

(Using SY0:/S will generate an error message, but SY0:/L/S will give you a listing of all the files on that disk.)

Reset SY0:, mount another disk, and repeat the process until each of your disks contains an accessible directory file of its contents. Use ONECOPY to copy the directory files onto the catalog disk. See your user manual for details. Now, go back to step 2 and concatenate the files to create your disk master catalog.

If you have a word processing program (such as Software Toolworks' TEXT) which can access text files, you can create the Disk Master Catalog directly from the individual directory files. In the process, some simple touches can be added to improve the appearance of your printed copy. Create a master read file which reads each directory file, places the disk label at the head of the directory, emboldens the disk label, indents the directory file, and skips a line between each directory file. The result will look like Figure 1.

A word processing program makes updates easier, since only the catalog files which have changed must be updated. Otherwise, you must repeat the entire process for each update.

A Disk Master Catalog is a convenient organizer of your disk software. It is a handy reference which can save time and frustration when you're looking for a specific file. And it's available with software you already have.

About the Authors:

Frank is an Engineering Duty Officer in the U.S. Navy presently assigned to Pearl Harbor Naval Shipyard. He has a Ph.D. in Physics. Mary Lynn is a free lance writer and has a B.S. in Physics. The Hutchisons have three and a half children and often must wait in line to use their H-89.

Announcing . . . **WATCHWORD**

The word processor and full screen editor for the Z100 and ZDOS.

FRIENDLY - FAST - FLEXIBLE

- *See subscripts, superscripts, underlining and boldface directly on the screen*
- *Create your own fonts and special characters*
- *Remap any key and configure to your printer*
- *Written in native 8088 assembly language for fast response*

Other features include: centering, formatting, right justification, arbitrary line length, automatic horizontal scrolling, split screen, and macros.

For additional details, send a SASE
Demo Disk: $3.00
WatchWord with Manual: $100.00
User Group Price (5 or more to same address): $50.00 each

S & K Technology, Inc.
4610 Spotted Oak Woods
San Antonio, TX 78249
512-492-3384
ATTN: Steve Robbins

MAGNOLIA
MICROSYSTEMS

CP/M Plus™ Released!

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

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

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

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

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

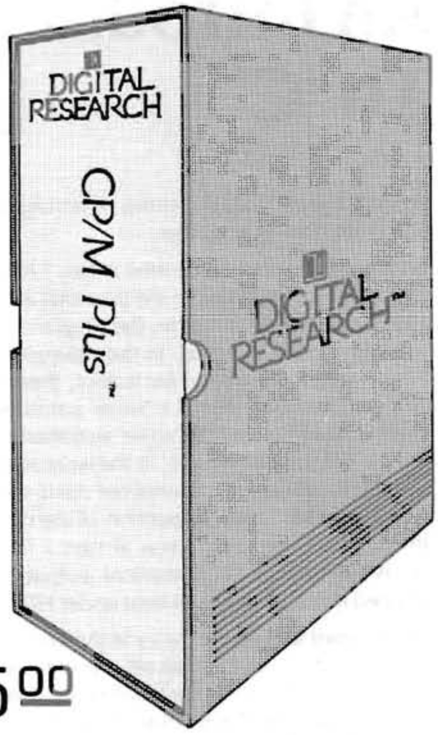
We are including the SOURCE code for our BIOS, together with Digital Research's MAC, RMAC, LINK, and SID software development tools, so you may make whatever modifications necessary for your specific application.

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

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

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

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



$ 595.00

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

Double Density Controller reduced $100

now only $495.

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

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

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

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

Part II

An Introduction To 'C'

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

This is the second of a series of articles designed to introduce the 'C' programming language.

Before we get into some new areas, I have a few comments about the program presented in the previous article. For those of you who did not see the prior article, the program presented printed a "hello" message on the terminal. In the example, we had to declare a variable, open the terminal for output, then use the declared variable as the channel number in a 'write' statement. The 'C' book indicated that channel 1 is used in 'write' statements to direct output to the terminal. But just putting a '1' in the write statement without the 'fopen' statement produced 'channel not open' messages when the program was executed. Upon inspection of the channel number assigned in the 'fopen' statement, it was always a '1'. Therefore, I assume that '1' is the channel for terminal output, but it must be explicitly opened in the program, at least under HDOS.

I mentioned an exercise to try in the prior article in order to see what happens when the ';' is left off one of the lines. If you tried this, you saw that the 'C' compiler printed an error message indicating that a ';' was missing. But the line it points to is the line after the missing semicolon. When you do get error messages from the 'C' compiler, be sure to check the lines before and after the indicated line for the problem. Also, some errors can cause other errors which can cause other errors... In other words, when you have many errors, try correcting errors one at a time. You will find in many cases, that correcting one mistake will remove several error messages.

Another exercise I mentioned trying was to change the word 'write' to 'qrite'. The interesting thing about this modified program is that it will pass through the 'C' compiler without errors. Only when the output from the compiler is assembled will an error occur - 'undefined qrite'. I point this out just to let you know that passing a program through the 'C' compiler without errors does not necessarily mean that there are no errors in the program. The first time I used the 'fopen' statement, I neglected to put double quotes around the 'w'. This passed through the compiler without error but got an undefined variable message from the assembler. Since the program only had three lines in it, I was able to quickly determine the problem. If the program had been larger, I would have had to spend more time figuring out how an undefined variable could have generated from a clean-compiled 'C' program.

Speaking of variables, the version of 'C' I am using supports two types of variables: character and integer. Character variables (char) reserve one byte of storage, integer variables (int) reserve two bytes. The process of declaring a variable assigns an address to the variable name. The variable name can be as long as you like, but only the first seven characters are recognized, so they must be unique. Underscores are permitted, and upper and lower case characters are considered different. Variables must start with a letter.

One of the most important aspects of a computer program is the output, and the output is not much good if it can't be formatted the way you want it. In 'C' the statement we use to format output is appropriately called 'format'. In this statement, we will indicate any literals and/or variable formats we will be printing. Variable formats are indicated by a '%' followed by a format character. Format characters we will be using are:

d = decimal, c = ASCII character, x = hex, o = octal.

We can also indicate the number of positions to print, so that '%6d' means print six decimal digits. Once the formats and literals are established via the 'format' command, subsequent 'printf' statements will utilize the indicated formats in sequence. As way of example, and in order to introduce new concepts, let's look at a sample program which will print the 'C' special character constants in decimal, octal, and hex.

```
#include "printf.c"
main()
/* This is a sample 'C' program which prints out
   the 'C' special character constants in decimal,
   octal and hex */
     (
     format("newline: dec=%d oct=%o hex=%x\n");
     printf('\n');
     printf('\n');
     printf('\n');

     format("tab: dec=%d oct=%o hex=%x\n");
     printf('\t');
     printf('\t');
     printf('\t');

     format("backspace: dec=%d oct=%o hex=%x\n");
     printf('\b');
     printf('\b');
     printf('\b');

     format("carriage return: dec=%d oct=%o hex=%x\n");
     printf('\r');
     printf('\r');
     printf('\r');

     format("form feed: dec=%d oct=%o hex=%x\n");
     printf('\f');
     printf('\f');
     printf('\f');

     format("backslash: dec=%d oct=%o hex=%x\n");
     printf('\\');
     printf('\\');
     printf('\\');

     format("apostrophe: dec=%d oct=%o hex=%x\n");
     printf('\'');
     printf('\'');
     printf('\'');

     format("bit pattern: dec=%d oct=%o hex=%x\n);
     printf('\123');
     printf('\123');
     printf('\123');

     )
```

Some things to note about this program:

1) #include "printf.c" - This statement adds the code found in file "printf.c" into this program. This is necessary to bring into this program the code to recognize 'format' and 'printf'. Make sure that this file is on your disk.

2) Notice how we mixed literals and formats in the 'format' statement. Anything that is not preceeded by a '%' is printed as is.

3) Also notice how we use three 'printf' statements to refer back to the three formats in the 'format' statement. The 'printf' statements will utilize the pre-defined formats in sequence.

4) Comments can be added to your program by enclosing the text between '/*' and '*/'. The comment text can extend over many lines and can be embedded within statements.

Now that you can output from a 'C' program, the next step is input. That's what we will look at next time.

'C' you later!

CHEAPCALC
Another Look

B. L. McFarland
17175 Gunther Street
Granada Hills, CA 91344

In Issue 37 of REMark, I presented an MBASIC spread sheet program for the H89 or H8+H19 terminal. In Issue 40, Clifford Lundburg presented an errata for the program that covered most of the problems existing with the program. However, there are a few "bugs" remaining in the program that need fixing. This article gives a third (and closer) look at the functioning of the program for those who are not familiar with spreadsheet programs. It should be noted that the allowable variable names are A-O, not A-P as stated in the article, unless the dimension statements in line 130 are changed to 16 instead of 15. The errata below assumes that the changes suggested by Lundberg have been made, and should produce a working program in both HDOS and CP/M systems.

Errata for CheapCalc

1. Delete IF statement on line 530; it is not needed.

2. Change the 130 back to 250 in line 2780, and add line 255 to work in both CP/M and HDOS systems.

```
255 FOR X%=1 TO MC%:FOR Y%=1 TO CW%(X%):
    A$(Y%,X%)="":NEXT:NEXT
```

3. Lines 1140 & 1150 do not function properly with some versions of MBASIC, and may not allow input of negative numbers correctly. In addition, a round-off problem exists when using the $ format with small numbers which is illustrated by the program examples.

```
1140 H$=STR$(A1):
     IF INSTR(H$,"E") THEN 1160
1145 H$=H$+".00":LP=INSTR(H$,"."):
     H$=LEFT$(H$,LP+2)
1150 IF RIGHT$(H$,1)="." THEN
     H$=LEFT$(H$,LEN(H$)-1)+"0"
```

The changes below were discussed in Lundberg's article to achieve the same effect as those suggested here, but these changes have a special purpose as discussed later in the article.

4. This change was misprinted in Issue 40, so use the following:

```
3880  NEXT
3890  CR=0:GOSUB 350:GOTO 2090
```

5. Problems occur when the data is larger than the screen display because of scroll-up of the screen. These changes also speed up the program by eliminating some string manipulation. The variables are all right adjusted as discussed by Lundberg but S$ has been eliminated to minimize "garbage" collection by the program. The SPACE$ function is used in place of S$ to achieve proper spacing of the variables.

```
Delete lines 230 & 280 where S$ is defined.
        780 B$(Y%,X%)=MID$(A$(Y%,X%),1,L%)
 Delete 786-800 as they are not needed now.
 1770 PRINT FN PC$(Y%+5-SY%,CO%);SPACE$(CW(A)-LEN(B$(Y%,X%));
        1785 PRINT B$(Y%,X%);NV$;
        1820 Change LEFT$(S$,5) to SPACE$(5)
        1870 H1%=CW%(FX%)/2:H2%=CW%(FX%)-H1%-1
        1880 PRINT SPACE$(H1%);MID$(T$,FX%,1);SPACE$(H2%);
Delete  1890 and 1980
In line 1910 change PRINT FX%; TO PRINT IV$;FX%;.
        1920 PRINT NV$;CL$:NEXT.
In line 1950 Add :YD%=YX%(A)-SY%+1:IF YD%>18 THEN YD%=18.
        1960 FOR Y1%=SY% TO SY%+YD%.
        1965 IF LEN(B$(SY%+Y1%,A)<1 THEN 1990
        1967 Z=T+CW%(A)-LEN(B$(Y1%,A).
In line 1970 change T to Z.
In line 2000 change XM% to MT%.
        3660 PRINT#1,SPACE$(CW(X1%)-LEN(B$(Y1%,X1%));B$(Y1%,X1%)
        3670 'LPRINT SPACE$(CW(X1%)-LEN(B$(Y1%,X1%));B$(Y1%,X1%)
```

6. In line 2890, change the last YH% to XH%.

7. Change line 2900 to GOTO 1570 instead of GOSUB 1570.

8. Delete line 2910.

9. Change 1.0 to 1.1 in line 260 to identify a modified program.

The above changes complete the errata for the program. The changes suggested below however, should be made to improve the operation of the program.

1. With a text editor that has global change capability, add line

```
125    DEFINT B-Z
```

and remove the % from all variables. This was suggested by Jim Tennent and saves memory.

2. In line 1470 delete the GOSUB 680, and in line 1520 change the GOSUB 1080 to GOSUB 370. This allows specifying the format of each cell individually in the A$ array (e.g. :FA1+A2, or :$1.23). The format is then retained until changed later in the problem calculation order.

3. Add the following line (See Appendix for an explanation).

```
2195 if A$="L" or A$="M" THEN 3780
```

```
Delete 3800 and add
       3785  A$(Y%,X%)=A$(YH%,XH%)
```

Program Checkout

After all the changes have been made to CHEAPCALC, the proper functioning of the code can now be checked. To simplify the description of the user operations, the keys to press will be enclosed in brackets [ ] followed by a brief description of what the user should be observing.

When the program is first run, the screen should show an A1 in the upper left hand side of the screen, followed by a blank line and then by an inverse border the has the letters A-H across the top and the numbers 1-19 down the left side. A beep should sound followed by an inverse blank field appearing in the A1 location. This blank inverse field is the cursor for the CHEAPCALC program.

The program is now ready for data entry in the default :$ format mode.

[1.23],[rtn] will cause an inverse 1.23 to appear in A1.

[shift-down arrow] will move the cursor to A2, and display the 1.23 in A1 in normal video. Data can now be entered in A2.

[-1.23],[rtn] will show an inverse -1.23 in A2.

[shift-down arrow] will move the cursor to A3, and display the HA-1.23 in A2 in normal video. Data can now be entered in A3.

[A1+A2],[rtn] will cause the sum of A1 and A2 to be displayed in A3. This is 0.00 for these numbers.

[ctrl-r][shift-down arrow][ctrl-r] will cause the sum of A2+A3 to be displayed in A4 by replicating the command in A3. The value displayed in A4 is -1.23 & A2+A3 is displayed on the second line of the screen under the A4 cell designation displayed on the first line (See the APPENDIX for a description of how replicate works, and how to add a copy command to the program).

[shift-down arrow],[A1*A2+1],[rtn] produces the expected result in A5 of -.51.

[shift-down arrow],[1+A1*A2],[rtn] produces the unexpected result in A6 of -2.74. This is caused by the order of operations which proceed from the left and cause 1+A1*A2 to be executed the same as (1+A1)*A2 in an ordinary BASIC program. The user must therefore keep the order of operations in mind at all times.

[shift-up arrow] 4 times is used to move the cursor to A2. [2.77],[rtn][!] recalculates the array displaying 4.00 in A3, 6.77 in A4,4.40 in A5, and 6.17 in A6.

[/] will display the available commands menu on line 2 of the CRT for selection of the desired user option.

[1],[rtn] enables the user to change the width of the cursor column and prompts the user with "WIDTH=". A number <30 is ok.

[15],[rtn] changes the A column width to 15 spaces, and redisplays the data array in the new widths.

[/],[2],[rtn] is used to save a data file, and has the prompt "SAVE FILE TO DISK:FILENAME=". Any legal filename is ok.

[SAMPLE.DAT],[rtn] saves this file on the current disk.

[/],[4],[rtn] is used to delete the complete data file from memory.

[/],[3],[rtn] is used to load a data file, and uses the prompt "READ FILE FROM DISK:FILENAME=".

[SAMPLE.DAT],[rtn] reads the above file back from the disk.

[/],[5],[rtn] is used to move the screen window to display other cells of the data array.

[M30],[rtn] moves the H20 cell to the upper left hand corner of the CRT. Only one fourth of the screen should be usable. Use the shift-arrow keys to verify.

[/],[5],[A1],[rtn] moves the A1 cell back to the upper left hand corner.

[6],[rtn] is used to print the data array to a line printer. The upper left hand corner to be printed is entered first, after the prompt "UPPER/LEFT CORNER:", followed by the lower right hand corner after the prompt "LOWER/RIGHT CORNER:". The printer is assumed to be set to print all columns on a single line.

[7],[rtn] is used to display the DOC file, and is helpful if some command has been forgotten. The DOC file was published in the March REMark, p. 15. You may wish to modify it.

If the entries described above have been made and the results were also as described, the program is probably "bug-free", and we can move on to a simple use of a spread sheet program.

Investment Comparison

As a simple case, we will compare a bank investment at a fixed interest rate compounded at different intervals (daily, weekly, monthly, semi-annually, & annually).

[/],[4],[rtn] if needed to clear all cells.
[/],[1],[rtn],[15] to give us a larger title row (A).
[shift-right arrow],[/],[1],[rtn],[15] to give us a larger data row (B). This is needed for floating point format.
[/],[5],[rtn],[A5] to move down several rows to leave room for titles. This is usually quicker than the arrow keys.

Now we enter the problem data.

```
KEYBOARD INPUT                                                LOCATION

[INTEREST RATE],[rtn],[shift-right arrow],[.10],[rtn] (A5,B5)
[shift-down arrow],[shift-left arrow]
[INVESTMENT],[rtn],[shift-right arrow],[2500],[rtn]    (A6,B6)
[shift-down arrow],[shift-down arrow],[shift-left arrow]
[COMPOUNDED FOR],,[rtn],[shift-right arrow],
[" ONE YEAR     ],[rtn]                                  (A7,B7)
[shift-down arrow],[shift-left arrow],[shift-left arrow]
[DAILY],[rtn],[B5/365],[rtn]                             (A8,B8)
[shift-down arrow],[shift-left arrow]
[WEEKLY],[rtn],[B5/52],[rtn]                             (A9,B9)
[shift-down arrow],[shift-left arrow]
[MONTHLY],[rtn],[B5/12],[rtn]                          (A10,B10)
[shift-down arrow],[shift-left arrow]
[SEMI-ANNUALLY],[rtn],[B5/2],[rtn]                     (A11,B11)
[shift-down arrow],[shift-left arrow]
[ANNUALLY],[rtn],[B5],[rtn]                            (A12,B12)
[shift-right arrow],[1+B12*B6],[rtn]                      (C12)
[shift-up arrow],[1+B11^2*B6],[rtn]                       (C11)
[shift-up arrow],[1+B10^12*B6],[rtn]                      (C10)
[shift-up arrow],[1+B9^52*B6],[rtn]                       (C9)
[shift-up arrow],[1+B8^365*B6],[rtn]                      (C8)
```

[!],[rtn] This causes the recalculation of the problem and should result in the following display from A5 to C12. Since the calculations proceed by row, the dollars and cents format is used for all numbers that will fit the format automatically. All fields are right adjusted. However, if you want the text to be left adjusted, just add enough spaces to fill the field shown on line 1 of the CRT.

```
  INTEREST RATE            .10
AMOUNT INVESTED        2500.00
     COMPOUNDED   FOR ONE YEAR
          DAILY   2.73973E-04    2762.79
```

```
         WEEKLY   1.92308E-03    2762.66
        MONTHLY   8.33333E-03    2761.78
  SEMI-ANNUALLY           .05    2756.25
       ANNUALLY           .10    2750.00
```

Save the data file.

[/],[2],[rtn],[SAMPLE2.DAT],[rtn]

Move the cursor to B5

[.07],[rtn],[!] to recalculate and obtain the following display.

```
  INTEREST RATE             .07
AMOUNT INVESTED         2500.00
     COMPOUNDED FOR ONE YEAR
          DAILY  1.91781E-04     2681.23
         WEEKLY  1.34615E-03     2681.13
        MONTHLY  5.83333E-03     2680.73
  SEMI-ANNUALLY          .03     2652.25
       ANNUALLY          .07     2675.00
```

Move the cursor to B6

[5,(CTRL-U 3 times)],[rtn],[!] changes 2500 to 5500 by copying the text in line 2, and gives the following display after recalculation.

```
  INTEREST RATE             .07
AMOUNT INVESTED         5500.00
     COMPOUNDED FOR  ONE YEAR
          DAILY  1.91781E-04     5898.71
         WEEKLY  1.34615E-03     5898.49
        MONTHLY  5.83333E-03     5897.60
  SEMI-ANNUALLY          .03     5564.35
       ANNUALLY          .10     5665.00
```

These examples illustrate most of the features of CHEAPCALC and should start the new users on their way to formulating problems with the program, customizing the code to meet their specific needs.

Appendix

Description of the Replicate Command

Several readers have requested a discussion of the replicate command logic contained in lines 3780-3890.

The replicate command [ctrl-r],[shift-arrow],[ctrl-r] sequence is usable only with arithmetic function commands and its use by the user results in the following program actions.

[ctrl-r] sends the program to line 3780 where the current cell indices are stored in XH% & YH%, and the flag (CR) is set equal to 1 after the IF test is passed.

The cursor is moved to a new cell location, and the next [crtl-r] sends the program again to 3780 where the IF test is failed and the difference in the old & new cell location is calculated in line 3790 after the new cell A$ is set equal to the old cell A$ in line 3785. If the old cell A$ was non-zero, the FOR loop (lines 3810-3880) then checks for a legal cell character (A-O) and if found, changes it by the difference found above. The numerical index is changed in the same manner. This continues until the end of the A$ string is found. If the replicate function is used on a text string, it will be changed to "garbage", while numbers will not be affected. The change suggested in Issue 40 by Lundberg keeps the old and new strings aligned.

The reader that has experimented with this command will agree that the choice of [ctrl-r] to start the replicate sequence leaves something to be desired since it requires a large movement of the hands to replicate a cell. Line 2195 changes this since it allows the [shift-DL] and [shift-IL] keys to be used in the sequence instead of [ctrl-r]. This change also allows the easy addition of a copy command (requested by several readers) by changing line 3785 to the following:

```
3785 A$(Y%,X%)=A$(YH%,XH%):IF A$="L" THEN 3890
```

Now the sequence [shift-DL],[shift-arrow],[shift-DL] will replicate an arithmetic command and the sequence [shift-IL], [shift-arrow],[shift-IL] will copy any cell contents unchanged to a new location, without requiring a large amount of hand movement. The reasons for several of the earlier changes should now be apparent.

```
10   '  *************************
20   '  *                       *
30   '  * ELECTRONIC WORKSHEET  *
40   '  *  WRITTEN FOR THE      *
50   '  *   APPLE II BY         *
60   '  *  WILLIAM V R SMITH    *
70   '  *                       *
80   '  * MODIFIED FOR MBASIC   *
90   '  *   BOB MCFARLAND       *
100  ' *       5/26/83         *
110  ' *                       *
120  ' *************************
130 DEFINT B-Z
140  MC=15:MR=40:DIM A$(40,15),B$(40,15),CW(15),YX(15)
150 SY= 1:XM = 1:SX= 1:WIDTH 255
160 BP$=CHR$(7)
170 E$=CHR$(27):EH$=E$+"H":'home cursor
180 CS$=E$+"E":' CLEAR SCREEN
190 CL$=E$+"K":' CLEAR TO EOL
200 IV$=E$+"p":' inverse video
210 NV$=E$+"q":' normal video
220 EN$=E$+"x1":' enable 25th line
230 DEF FN PC$(R,C)=E$+"Y"+CHR$(R+31)+CHR$(C+31):'direct screen addressing
250 DS$=E$+"y1":'disable 25th line
260 PRINT E$+"w";CS$;EN$;FN PC$(25,1);CL$
270 FOR X=1 TO XM:FOR Y=1 TO YX(X):A$(Y,X)="":B$(Y,X)="":NEXT:NEXT
280 PRINT  EN$;FN PC$(25,25);"Mbasic Basicalc Program Version 1.1";EH$
290  FOR X= 1 TO MC:YX(X)=1:CW(X) = 9: NEXT:OF=1
300 T$ = "ABCDEFGHIJKLMNOP"
310 T1$ = "****************"
320  GOTO 2090
330  '
340  '  * VARIABLE PARCER  *
350  '
360 L=  LEN (A$(Y,X)):F = 2:A1 = 0:A2 = 0:P= 1:H$ = ""
370  IF L= 0 THEN 790
380  IF P> L THEN 790
390  GOSUB 1050
400 IF C> 64 THEN  GOSUB 860: IF P> L THEN  RETURN:'character input
410  IF C= 46 THEN 480:'decimal point
420  IF C> 41 AND C< 48 THEN GOSUB 690:F = C- 41: GOTO 380:'operands
430 IF C=94 THEN GOSUB 690:F=7:GOTO 380
440  IF C= 38 THEN 1330:'& operator
450  IF C> 47 AND C< 58 THEN 480:'numbers
460  IF C= 58 THEN 1510:': sets output format
470  GOTO 790
480 H$ = H$ +  CHR$ (C): IF P> L THEN  GOSUB 690: GOTO 1100
490 GOSUB 1050:GOTO 400
500  '
510  ' * INPUT STATEMENT **
520  '
530 IV=1:I$=A$:PRINT A$;:PRINT FN PC$(2,1);
540  A$=INPUT$(1):A=ASC(A$):IF A=27 THEN 2200
550 IF A<>8 THEN 590:'routine to handle backspace
560 L=  LEN (I$):I$ = MID$ (" " + I$,2,L - 1):PRINT FN PC$(2,1);I$;
```

```
570 IF LEN(I$)=0 THEN A$="":RETURN
580  GOTO 540
590  IF A  = 21 THEN 640:'ctrl-u
600  IF A  = 13  THEN 650:'ctrl-M(CR)
610  IF  A< 31 THEN 540:'any other ctrl character is ignored
620  IF A  = 34  THEN GOTO 540:'" mark for string input
630 I$ = I$ + A$:PRINT FN PC$(2,1);I$;:IV=LEN(I$):GOTO 540
640 A$ =  MID$ (A$(Y,X),IV+1,1): GOTO 630
650 A$=I$:RETURN
660  '
670  ' * PERFORM MATH FUNCTION
680  '
690 A2 =  VAL (H$):H$ = "":'math function subroutine
700 F1 = F:F = 2
710  ON F1 GOSUB 730,740,720,750,720,770,760
720  RETURN
730 A1 = A1 * A2: RETURN
740 A1 = A1 + A2: RETURN
750 A1 = A1 - A2: RETURN
760 A1 = A1 ^ A2: RETURN
770  IF A2 <  > 0 THEN A1 = A1 / A2
780  RETURN
790 H$ =  MID$ (A$(Y,X),1,L)
800 B$(Y,X)=LEFT$(H$,CW(X))
810  GOSUB 1270: '   * XM AND YX TEST
820  RETURN
830  '
840  ' * FIND MATH VALUE OF  SCREEN
850  '
860 X3=C-64:IF C=94 THEN RETURN
870  IF X3 >MC THEN  GOSUB 790: RETURN
880 H$ = "": IF L= 1 THEN 790
890  GOSUB 1050: IF C< 48 OR C> 57 THEN  GOTO 790
900  GOTO 920
910  GOSUB 1050
920  IF C< 48 OR C> 57 THEN 960
930 H$ = H$ +  CHR$ (C)
940  IF P> L THEN 960
950  GOTO 910
960 Y3 =  VAL (H$)
970  IF Y3 > MR OR X3 > MC THEN H$ = "ERROR":P = L+ 1: GOTO 1120
980 H$ = B$(Y3,X3)
990  GOSUB 690
1000  IF P> L THEN  GOSUB 1100: RETURN
1010  RETURN
1020 '
1030  ' *** PARCE LINE FOR CHAR
1040 '
1050 C=  ASC ( MID$ (A$(Y,X),P,1)):P= P+ 1
1060  RETURN
1070  '
1080  ' * ASSIGN ANSWER
1090  '
1100  IF A$(Y,X) = "" THEN  B$(Y,X)="":RETURN
1110  IF  LEN ( STR$ ( INT (A1))) > CW(X) THEN H$ = "ERROR"
1120  ON OF GOSUB 1160,1210,1220,1230
1130  IF OF = 4 OR OF = 1 THEN 1270
```

```
1710  GOSUB 1860
1720  PRINT FN PC$(1,25);CL$;
1730  GOTO 2130
1740  '
1750  ' * INDIVIDUAL VALUE SCREEN PRINTER
1760  '
1770  A=X:Y1=Y-SY:IF LEN(A$(Y,X))>0 THEN GOSUB 360
1780 CO=5:IF X=SX THEN 1800
1790  FOR X2 = SX TO X- 1:CO = CO + CW(X2): NEXT X2
1800 PRINT FN PC$(Y1+5,CO);SPACE$(CW(A)-LEN(B$(SY+Y1,A)));
1810 PRINT B$(SY+Y1,A);NV$;
1820  RETURN
1830  '
1840  ' * COMPLETE SCREEN PRINT
1850  '
1860  PRINT CS$;FN PC$( 4,1);:PRINT IV$;SPACE$(5);:MT=MC
1870 PP = 0: FOR FX= SX TO MC:PP = PP + CW(FX)
1880 IF PP > 77 THEN MT = FX - 1:FX= MC+1
1890 NEXT
1900  FOR FX= SX TO MT
1910 H1= CW(FX) / 2:H2 = CW(FX)-H1-1
1920 PRINT SPACE$(H1);MID$(T$,FX,1);SPACE$(H2);
1930  NEXT:PRINT CL$;FN PC$(5,1);
1940 YD=18:IF SY+YD>MR THEN YD=MR-SY
1950  FOR FX= SY TO YD+ SY: PRINT IV$;FX;: IF FX < 10 THEN  PRINT " ";
1960  PRINT NV$;CL$: NEXT
1970 A = SX:T = 5
1980 PRINT FN PC$( 5,1);:YD=YX(A)-SY+1:IF YD>18 THEN YD=18
1990  FOR Y1 = 0 TO YD
2000 IF LEN(B$(SY+Y1,A))=0 THEN 2030
2010 Z=T+CW(A)-LEN(B$(SY+Y1,A))
2020  PRINT FN PC$(Y1+5,Z);B$(SY+Y1,A);
2030  NEXT
2040 T = T + CW(A):A = A + 1: IF A = < MT THEN 1980
2050 RETURN
2060  '
2070  ' * PROMPT OF INPUT
2080  '
2090 DF = 1
2100  PRINT CS$
2110  GOSUB 1860
2120 X= 1:Y= 1
2130  PRINT FN PC$(1,1);MID$ (T$,X,1);Y;SPACE$(CW(1)-4);":<-";BP$
2140  PRINT FN PC$( 2,1);CL$;A$(Y,X);
2150 PRINT IV$;:GOSUB 1770
2160  '
2170  ' * INPUT AND PERFORM
2180  '
2190 A$=INPUT$(1):A=ASC(A$):IF A<>27 THEN 2250:' Test for ESC
2200 A$=INPUT$(1):IF A$="C" THEN DF=-1: GOTO 2340:'Move back
2210 IF A$="L" OR A$="M" THEN 3840
2220 IF A$="D" THEN DF=-1:GOTO 2470:'Move forward
2230 IF A$="B" THEN DF=1:GOTO 2340:'Move up
2240 IF A$="A" THEN DF=1:GOTO  2470:' Move down
2250  IF A = 47 THEN 2800:'/= GOTO command menu
2260  IF A=18 THEN 3840:' Replicate function
2270 IF A=4 THEN 3800:'CTRL D =quit program
2280 IF A=35 THEN A$(Y,X)="":GOTO 2650
```

```
1140  B$(Y,X)=STR$(A1)
1150  GOTO 1270
1160 H$=STR$(A1):IF INSTR(H$,"E") THEN 1200
1170 H$=H$+".00":LP=INSTR(H$,"."): H$=LEFT$(H$,LP+2)
1180  H$=STR$(A1)+".00":LP=INSTR(H$,"."): H$=LEFT$(H$,LP+2)
1190 IF RIGHT$(H$,1)="." THEN H$=LEFT$(H$,LEN(H$)-1)+"0"
1200 B$(Y,X) =  H$: RETURN
1210 A1 =  INT (A1): RETURN
1220  RETURN
1230 A1 =  INT (A1): IF A1 > 20 THEN A1 = 20
1240  IF A1 < 1 THEN A1 = 1
1250 B$(Y,X)=LEFT$(T1$,A1)
1260  RETURN
1270  IF X=> XM THEN XM = X
1280  IF Y> YX(X) THEN YX(X)=Y
1290  RETURN
1300  '
1310  ' **** SUM(FUNCTION)
1320  '
1330 P= P+ 4: GOSUB 1050
1340  GOSUB 860:Y4 = Y3:X4 = X3
1350  GOSUB 1050: GOSUB 860
1360 A1 = 0:A2 = 0:X5 = X3:Y5 = Y3
1370  IF Y4 = Y5 THEN 1430
1380 X3 = X4: FOR Y3 = Y4 TO Y5
1390 P= 1
1400  GOSUB 980
1410  NEXT
1420  GOSUB 1100: RETURN
1430 Y3 = Y4: FOR X3 = X4 TO X5
1440 P= 1
1450  GOSUB 980
1460  NEXT
1470  GOSUB 1100: RETURN
1480  '
1490  ' * OUTPUT FORMAT *
1500  '
1510 GOSUB 1050
1520  IF C= 36 THEN OF = 1:' $
1530  IF C= 73 THEN OF = 2:' I
1540  IF C= 70 THEN OF = 3:' F
1550  IF C= 42 THEN OF = 4:' *
1560 GOTO 380
1570  '
1580  ' * VIDEO SCREEN LAYOUT
1590  '
1600 X2=X:Y2=Y
1610  FOR X1 = 1 TO XM
1620  PRINT FN PC$(1,25);: PRINT IV$;: PRINT "WORKING";NV$;
1630  FOR Y1 = 1 TO YX(X1)
1640  IF A$(Y1,X1) = "" THEN  1680
1650  PRINT FN PC$(1, 34);: PRINT  MID$ (T$,X1,1);Y1;
1660 X= X1:Y= Y1
1670  GOSUB 360
1680  NEXT
1690  NEXT
1700 X=X2:Y=Y2
```

```
2290 IF A=33 THEN GOTO 1600:'execute functions
2300  IF A > 43 THEN 2620:'operands
2310  IF A = 38 THEN 2680:'& command
2320  IF A = 34 THEN  A$=INPUT$(1): GOTO 2620:'"=text input to cell follows
2330  A$=INPUT$(1): GOTO 2190
2340  GOSUB 1770
2350  ON DF + 2 GOTO 2360,2130,2390
2360 X= X+ 1: IF X> MC THEN X= MC: GOTO 2380
2370  IF X> MT THEN SX= SX + 1: GOSUB 1860: GOTO 2370
2380  PRINT IV$;: GOSUB 1770:GOTO 2130
2390 Y= Y+ 1: IF Y> MR THEN Y= MR: GOTO 2430
2400 IF Y>18+SY THEN X3= -1:SY=SY+10:Y=SY+10
2410 IF Y>MR THEN Y=MR:SY= MR - 18
2420  IF X3 =  - 1 THEN  GOSUB 1860:X3 = 0
2430  GOTO 2380
2440  '
2450 ' * MOVE CURSOR
2460 '
2470  ON DF + 2 GOTO 2480,2610,2530
2480  GOSUB 1770
2490 X= X- 1: IF X> = SX THEN  GOTO 2380
2500 SX= SX - 1: IF X= 0 THEN X= 1:SX= 1: GOTO 2520
2510  GOSUB 1860
2520  GOTO 2380
2530  GOSUB 1770
2540 Y= Y- 1: IF Y= > SY THEN GOTO 2380
2550 SY= SY - 10:Y= SY: IF Y<  = 0 THEN Y= 1:SY= 1
2560  GOSUB 1860
2570  GOTO 2380
2580  '
2590  ' ** INPUT STRING FOR PAGE
2600  '
2610  IF A$ = "&" THEN 2680:'command for summations
2620  PRINT FN PC$( 2,1);A$(Y,X): PRINT FN PC$( 2,1);: GOSUB 530
2630  IF A$ = "" THEN 2660
2640 A$(Y,X) = A$
2650  GOSUB 360
2660  GOTO 2130
2670  '
2680  ' ** SUM STATEMENT **
2690  '
2700  PRINT FN PC$( 1,1);CL$;
2710  PRINT FN PC$( 2,1);:INPUT "SUM(START = ";A$
2720  PRINT FN PC$( 2,1);CL$;"SUM(";A$;" THRU ";: INPUT "";B$
2730  PRINT FN PC$( 2,1);"SUM(";A$;" THRU "B$;")"
2740  IF A$ = "" OR B$ = "" THEN 2130
2750  A$(Y,X) = "&SUM(" + A$ + "-" + B$ + ")"
2760  GOSUB 360: GOTO 2130
2770  '
2780  ' * HANDLE GLOBAL COMMAND
2790  '
2800 CLOSE#1: PRINT FN PC$( 2,1);CL$;
2810  INPUT"1-WIDTH 2-SAVE 3-LOAD 4-CLEAR 5-GOTO LOCATION 6-PRINT 7-HELP";A$
2820  ON  VAL (A$) GOTO 2890,3000,3220,270,3460,3630,2830
2830 ON ERROR GOTO 2880:OPEN "I",1,"MBCALC.DOC"
2840 FOR I=1 TO 15:LINE INPUT#1, A$:PRINT FN PC$(I+5,8);CL$;A$:NEXT
2850 PRINT FN PC$(23,8);"HIT ANY KEY TO CONTINUE/ESC TO STOP":A$=INPUT$(1)
```

```
2860 IF A$<>CHR$(27) THEN 2840
2870 CLOSE#1:ON ERROR GOTO 0:GOTO 2090
2880 A$=INPUT$(1):RESUME 2870
2890  PRINT FN PC$( 2,1);CL$;:INPUT "WIDTH = ";A$:A=VAL(A$):IF A > 30 THEN 2890
2900  CW(X) = A
2910  YH = Y:XH=X
2920  FOR Y= 1 TO YX(X): GOSUB 790: NEXT
2930 Y2=YH:X2=XH
2940 GOTO 1610
2950  ' GOSUB 3340: GOTO 2090
2960  '
2970  ' * DISK I/O
2980  ' * FILE OUT *
2990  '
3000  GOSUB 3390
3010  PRINT FN PC$( 1,1);CL$;
3020  PRINT "SAVE FILE TO DISK FILENAME = ";:INPUT A$
3030  IF A$ = "" THEN  GOSUB 3390: GOTO 2130
3040  PRINT FN PC$( 1,1);CL$
3050 OPEN "O",1,A$
3060  PRINT#1, XM
3070  FOR X= 1 TO XM
3080  PRINT#1, CW(X),YX(X)
3090  FOR Y= 1 TO YX(X)
3100  PRINT#1, CHR$ (34);A$(Y,X); CHR$ (34)
3110  PRINT#1, CHR$ (34);B$(Y,X); CHR$ (34)
3120  NEXT
3130  PRINT#1, "<>"
3140  NEXT
3150  PRINT#1, "<>"
3160 CLOSE#1
3170 Y= 1:X= 1
3180  GOTO 3780
3190  '
3200  ' * FILE IN *
3210  '
3220  PRINT FN PC$( 1,1);
3230  PRINT "READ FILE FROM DISK  FILENAME = ";:INPUT A$
3240  IF A$ = "" THEN  GOSUB 3390: GOTO 2130
3250  OPEN "I",1,A$
3260  IF EOF(1) THEN 3360
3270  INPUT#1, XM
3280  FOR X= 1 TO XM
3290  INPUT#1, CW(X),YX(X)
3300  FOR Y= 1 TO YX(X)
3310  INPUT#1. A$(Y.X).B$(Y.X)
3320  NEXT
3330  INPUT#1, B$: ' ERROR IF NOT <>
3340  NEXT
3350  INPUT#1, B$: ' ERROR IF NOT <>
3360 CLOSE#1
3370  GOSUB 3390
3380 GOTO 3780
3390  PRINT FN PC$( 1,1);CL$;
3400  PRINT FN PC$( 2,1);CL$;
3410  PRINT FN PC$( 3,1);CL$;
3420  RETURN
3430  '
3440  ' ** GOTO LOCATION
3450  '
3460  GOSUB 3470: GOTO 2130
3470  GOSUB 3390
3480  PRINT FN PC$( 2,1);: INPUT "GO TO PAGE LOCATION :";A$
3490  GOSUB 3540
3500  IF X1 * Y1 = 0 THEN  RETURN
3510 X= X1:SX= X1:Y= Y1:SY= Y1:PRINT CS$
3520  GOSUB 1860
3530 GOTO 1730
3540 L=  LEN (A$): IF L< 2 THEN X1 = 1:Y1 = 1: RETURN
3550 X1 =  ASC ( A$) - 64
3560  IF X1 < 1 OR X1 > MC THEN X1 = 1: RETURN
3570 Y1 =  VAL ( RIGHT$ (A$,L- 1))
3580  IF Y1 < 1 OR Y1 > MR  THEN X1 = 1:Y1 = 1
3590  RETURN
3600  '
3610  ' *** PRINT OUT
3620  '
3630  GOSUB 3390
3640  PRINT FN PC$( 2,1);: INPUT "UPPER/LEFT CORNER:";A$: GOSUB 3540
3650 X3 = X1:Y3 = Y1
3660  PRINT FN PC$( 2,1);: INPUT "LOWER/RIGHT CORNER :";A$: GOSUB 3540
3670 X4 = X1:Y4 = Y1
3680  'OPEN "O",1,"LP:"
3700  FOR Y1 = Y3 TO Y4
3710  FOR X1 = X3 TO X4
3720 'PRINT#1, SPACE$(CW(X1)-LEN(B$(Y,X)));B$(Y1,X1);
3730  LPRINT SPACE$(CW(X1)-LEN(B$(Y1,X1)));B$(Y1,X1);
3740  NEXT
3750 'PRINT#1,
3760  LPRINT
3770  NEXT
3780 'CLOSE#1
3790 X1 = 1:Y1 = 1:GOTO 3510
3800 PRINT DS$;CS$;E$+"z":X=FRE(0):PRINT X:END
3810 '
3820 ' REPLICATE FUNCTION & COPY FUNCTIONS
3830 '
3840 PRINT BP$:IF CR<1 THEN XH=X:YH=Y:CR=CR+1:GOTO 2190
3850 LD=0:A$(Y,X)=A$(YH,XH):IF A$="L" THEN 3960
3860 DX=X-XH:DY=Y-YH:LA=LEN(A$(YH,XH)):IF LA<1 THEN 2190
3870 IS=1:IF ASC(A$(Y,X))=38 THEN IS=5
3880 FOR I=IS TO LA:A$=MID$(A$(YH,XH),I,1):A=ASC(A$)
3890 IF A<65 OR A>65+MC THEN 3950
3900 IF I=1 THEN A$(Y,X)=CHR$(A+DX):GOTO 3920
3910 A$(Y,X)=MID$(A$(Y,X),1,I-1+LD)+CHR$(A+DX)
3920 N=VAL(MID$(A$(YH,XH),I+1))
3930 N$=STR$(N):LN=LEN(N$):N=N+DY:N$=STR$(N):LD=LEN(N$)-LN
3940 A$(Y,X)=A$(Y,X)+MID$(N$,2)+MID$(A$(YH,XH),I+LN)
3950 NEXT
3960 CR=0:GOSUB 360:GOTO 2130
```

Patch Page

Pat Swayne
Software Engineer

KEYMAP Patch

I have discovered that the KEYMAP program (HUG disk 885-1230[-37]) does not work properly under the old version of Heath/Zenith CP/M (version 2.2.02). The mapping works OK, but the ESCape key does not work while the program is loaded. This problem may also affect non-Heath versions of CP/M. To test if the problem affects you, load any configured or unconfigured version of KEYMAP and press the escape key (at the CP/M A> prompt). If the symbols " ↑ [" do not appear, you need to make the following patch.

```
A>DDT KEYMAP.COM      (or KEYBAS.COM, etc.)
NEXT  PC
0F00 0100
-S192
0192 2E 0E
0193 64 .             (type a period)
-S19B
019B 2D 0D
019C C2 .
-S1C4
01C4 2E 0E
01C5 64 .
-S1CD
01CD 2D 0D
01CE C2 .
-^C                   (Control-C)
A>SAVE 14 KEYMAP.COM
```

If you would like to patch the source code, locate the instruction MVI L,100, which occurs two places in the program, and in each case change it to MVI C,100. A few lines below each place, change DCR L to DCR C.

SDUP Patch

A problem has been found in the SDUP program on disk 885-1121 in that it does not copy the last sector on a disk. Sometimes this affects the files on a disk, and sometimes it does not, which is why I did not catch it sooner. The problem must be fixed at the assembly level as follows. Copy SDUP.ASM to a disk that will have at least 20 free sectors on it after SDUP.ASM is copied. Now, locate the section of the code that looks like this:

```
        XCHG                    ELSE PUT TOTAL IN DE
        LHLD    SECPTR          GET POINTER AGAIN
        MOV     A,E
        SUB     L               SUBTRACT IT FROM TOTAL
LOOP1   STA     SECTORS         UPDATE SECTOR COUNT
```

Edit that part of the program, so that it looks like this:

```
        XCHG                    PUT TOTAL IN DE
        LHLD    SECPTR          GET POINTER AGAIN
        CALL    $CHL            COMPLEMENT IT
        XCHG
        DAD     D               SUBTRACT IT FROM TOTAL
        MOV     A,L
        INR     A
LOOP1   STA     SECTORS         UPDATE SECTOR COUNT
```

Locate the following line near the beginning of the program.

```
$HLIHL  EQU     30211A
```

Add this line just below it.

```
$CHL    EQU     30224A
```

Copy HOSDEF.ACM from your HDOS Software Tools disk to either the disk with SDUP.ASM or your system disk, and assemble SDUP as in this example.

```
>ASM SY1:SDUP=SY1:SDUP
```

Here, we assume that ASM.ABS is on your system disk (SY0:), and that the SDUP disk is in SY1:. The assembler will create a new SDUP.ABS that you can use in place of the original one.

DDEU Patch

If you would prefer an underline cursor to a block cursor in the DDEU program (HUG disk 885-1225[-37]), make the following patch.

```
A>DDT DDEU.COM
NEXT  PC
2880 0100
-S27B1
27B1 78 79
27B2 0E .
-^C
A>SAVE 40 DDEU.COM
```

HDOS ASM Patch Revisited

In the HDOS ASM patch in the July issue of REMark (#42), there is a typographical error in the Patch Check Code. Since this patch corrects a bug in the assembler, I will repeat the entire patch for those who missed it, with the correct Check Code.

```
>PATCH

PATCH Issue #50.06.00
File Name? ASM
Patch ID? IFOJIC
Prerequisite Code? IFBEIADPGEFFCF
Address? 64210
064210 = 022/21
064211 = 032/^D
Address? ^D
Patch Check Code? EDKIJHLG
```

ARE YOU MOVING?

Don't leave your REMark behind.

Send your change of address in now to:

Heath Users' Group
Hilltop Road
St. Joseph, MI 49085

A Software View of Operating Systems

(Note: The following material was taken from DUP & DUMP, the Newsletter of the San Diego Heath Users' Group, 660 Flora Drive, Oceanside, CA 92056.)

George Coade
San Diego Heath Users' Group
660 Flora Drive
Oceanside, CA 92056

January's meeting and discussion of the various merits of CP/M and HDOS got me to thinking about the software and hardware trade-offs that make up any computer. Some nice opportunities for advanced design are available in the H/Z-89 and I hope you will "come with me" for a brief historical review of CP/M and HDOS with a view to the future of these systems.

CP/M was first written by Gary Kildall in 1971 to make use of Intel's 8008 CPU chip (which was soon upgraded to the 8080). Gary's aim was to use the operating system as a buffer between the system hardware and user programs so that programs would be easily interchangeable. He also wanted the system to handle the filing requirements of the (then) newly developed floppy disks from IBM. Intel, his employer at the time, was offered the system but turned it down. Gary left Intel to form his own company, Digital Research, as the sales group for what became CP/M Version 1. He had very little luck selling the new concept because the manufacturers of the time wanted to write their own systems and sell their own software. Not until 1975 did the idea of a standard operating system running interchangeable user programs make Digital Research successful financially.

Gary continued to run the company from his home until about 1980 when he began to expand into language systems and operating systems beyond the Intel series. This date also marked the transition from the balky CP/M Version 1.4 to the much smoother Version 2.2.

HDOS, from the Heath Company, was planned just before CP/M was taking off as a standard and during the time that Intel was promoting universal adoption of their Multibus design for computer construction. Multibus was based on a plug-in backplane with removable system boards in 8K segment divisions. The CPU and its control programs occupied the first board in low address space with boards for memory above that level and I/O boards and disk handling boards somewhere in high memory space. Boards could be added or removed in 8K multiples as cost and function required. Anyone with an H-8 will remember this construction well (except that Heath did not use the Multibus connector in the H-8).

Heath also wanted their own operating system to be easy to use for beginning hobbyists. CP/M's reputation for crashing disks and wiping out systems was not too compatible with Heath's marketing base of hams and schools and training groups. Besides, at that time, creating your own operating system looked a lot easier to do (famous understatement #1).

Heath turned to a consulting firm (Wintek) to produce the software and soon found that the young man doing most of the writing would rather work for them directly. So began J. Gordon Letwin's association with Heath. Letwin is reportedly one of the great brains of the world and can talk almost as fast as he thinks. His system made use of the interrupt system of the 8080 CPU to access the operating system, a feature of the chips that Kildall had omitted in CP/M. Only now with the introduction of CP/M-86 is the interrupt system used and, incidentally, the entire operating system is placed in low memory. [JGL's operarting system also bears a remarkable resemblance to that of the Digital Equipment Corporation's PDP series with .MCALLs instead of .SCALLs, et cetera. The first time I attended a DEC operating systems class, I felt like I was at home! - ed.]

With Heath having only ROM in lower memory, the interrupt structure branches to a table of jumps in low RAM memory and from there to the correct system function. User programs can modify those jumps if desired and substitute new functions as needed. The disk dump program from Barry Watzman is a good example of how to modify the jump table.

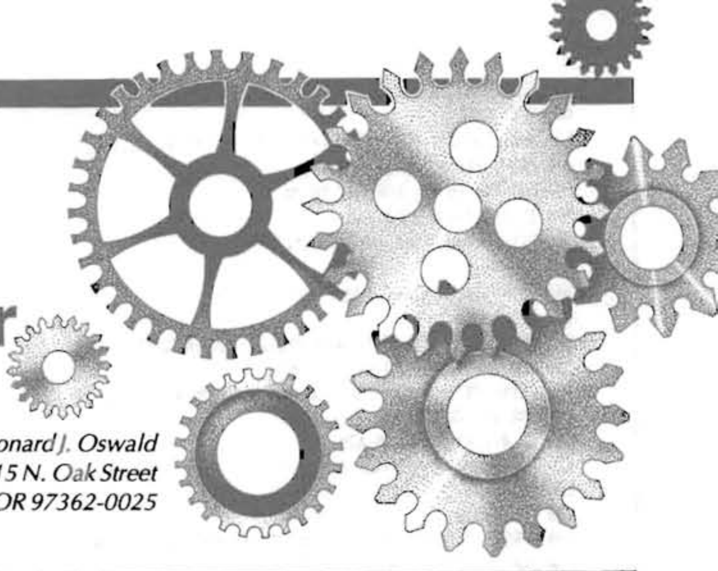
Other "goodies" are a system that sizes itself to fit the available memory without having to run a special program such as MOVCPMxx whenever new memory becomes available. The programs also carry their own markers to tell the operating system where in memory they should be loaded (i.e., debugging programs that load themselves into high memory). The special programs to run printers and modems [xx.DVD] are separate but callable from the main body of the operating system to allow for easy upgrading to new accessories as they are developed.

HDOS remains a great system with plenty of useful programs and there are many parallels between it and the new MSDOS that Letwin has fathered in his new job at MicroSoft. I happen to think that Heath has the nucleus of a better system than MicroSoft's but then they still have Letwin. HDOS 2.0 is very good for most uses but is limited to sectors of 256 bytes and therefore is not able to handle hard disks and their vastly expanded storage space.

As a brief "dream list" we could ask for sufficient system calls to manage disk access on a track and sector level, free disk interchange in a drive with the system remounting by itself whenever necessary, switching to disks other than SYO: while the programs are being run, allowance for extended memory to be used as a virtual disk and hard disk capacity. Making the entire system compatible with the calls and memory structure of CP/M would also be a great advantage. With new schemes of memory management it should be possible to run CP/M programs under HDOS just as easily as running HDOS programs under CP/M.

What, then, is available to expand our present machines? One of the new ideas is memory beyond the 64K address range of the Z-80 CPU. This extended memory would be used much as a disk drive but with a 16X increase in access speed. Magnolia Microsystems makes the XM-318 which you can see at the Heath store. It plugs into the spot for the Heath 16K memory expansion and provides that memory plus 110K of "disk" storage (128K of chips total). The store is using the extra memory to run MP/M, the multiuser version of CP/M, on several terminals which is the announced purpose of the board. The price is $595.00 which places the board in the expensive category for a single user system. Then there is a presently unanswerable question of how well this arrangement would function with CP/M Plus that expects 196K [good gosh!] of memory (three banks of 64K). The needed answers will have to wait for the software to come through the pipeline from Heath/Zenith as they add the drivers and modifications for the H/Z-89 family of computers.

WORD-SEARCH Puzzle Generator



Leonard J. Oswald
115 N. Oak Street
Mt. Angel, OR 97362-0025

This article and the accompanying program came into being as a result of an original program and article appearing in the June 1981 issue of the Softside magazine starting on page 92. This is a copyrighted program written by David W. Durkee for the Apple computer with translations for the Radio Shack TRS-80 and the Atari computers by Jon Voskuil.

Do you or your friends like to work crossword puzzles? Have you tried to find the imbedded words in the word-search puzzles? If your answer to either question is yes, then you might like this little program.

I have used the original Radio Shack version with a few modifications for over a year and have submitted to our local paper (a very small paper in size and circulation) one puzzle each week for the enjoyment of my neighbors and friends. A few months ago, my Radio Shack computer was having interface problems and I was forced to translate this fine program so that it would run under Microsoft BASIC in the Heath/Zenith computer that I have at my business. The following information will guide you through this short program and hopefully, with some of the enhancements that I suggest, you will have much fun and a use for this program.

I have not seen this program run under the Apple or Atari versions. I will say that the Heath/Zenith version runs considerably faster than the Radio Shack TRS-80 version and the Z-90 with Z-25 printer now gets to do all the work.

Basically, the program does the following when you run it. The screen is cleared and the title page with the appropriate author, translator, and copyright information is displayed. After a few seconds, the screen is again erased and the instructions are displayed. You're allowed two choices, either of which will require a simple one key input. The choices will be so that (1) you may see the puzzle generated and put together or, (2) for a blank screen to be shown during puzzle

```
1 '     Version 11/5/1982 by LJO
2 '     Softside June 1981
3 E$=CHR$(27): CL$=E$+"E": Y$=E$+"Y": J$=E$+"J"
10 PRINT CL$:Y$"#=";"WORD SEARCH PUZZLE":
       PRINT Y$"%=";"by David W. Drurkee":
       PRINT Y$"'?";"translated by":
       PRINT Y$"(=";"Leonard J. Oswald":
       PRINT Y$"*=";"Copyright (C) 1981"
20 FOR I=1 TO 1000:NEXT I
40 PRINT Y$"%'";J$"To create a puzzle, simply type in a
word after the '?' prompt,":PRINT Y$"&'";"and press <RETURN>.
When you're finished, enter 'STOP' as your":PRINT Y$",.";
last word, and the computer will do the rest."
50 PRINT Y$"*'";"Please choose:":PRINT Y$",.";
"1 - for normal display during entry":PRINT Y$"..";
"2 - for blank display (so that you can't see the puzzle)"
55 K$=INPUT$(1): K=VAL(K$): IF K<1 OR K>2 THEN 55
60 DEFINT A-Z: DIM W$(200),A(52,52),B(3,3),C$(52,52)
65 DEF FN C$(Y1,X1)=Y$+CHR$(Y1+31)+CHR$(X1+31)
70 RANDOMIZE PEEK(11)
80 PRINT CL$;
90 Z=Z+1
92 PRINT Y$;"6 ";J$;"WORD #";Z;: INPUT A$: IF A$="" THEN 92
95 W$(Z)=A$
100 IF A$="STOP" THEN 500
120 U=INT(RND(1)*20)+1: L=INT(RND(1)*20)+1
160 FOR X=-1 TO 1: FOR Y=-1 TO 1
170 IF X=0 AND Y=0 THEN 270
180 X1=L: Y1=U
190 FOR C=1 TO LEN(A$)
200 X1=X1+X: Y1=Y1+Y
210 IF X1>20 OR X1<1 OR Y1>20 OR Y1<1 THEN B(X+2,Y+2)=0: GOTO 270
220 IF A(X1,Y1)=0 THEN 250
230 IF A(X1,Y1)<>ASC(MID$(A$,C,1)) THEN B(X+2,Y+2)=0: GOTO 270
240 B(X+2,Y+2)=B(X+2,Y+2)+1
250 NEXT C
260 B(X+2,Y+2)=B(X+2,Y+2)+1: B=B+1
270 NEXT Y,X
280 IF B=0 THEN 120
310 R=2: D=2
320 FOR X=1 TO 3: FOR Y=1 TO 3
330 IF B(X,Y)>B(R,D) THEN R=X: D=Y
340 NEXT Y,X
350 X=R-2: Y=D-2
360 IF X=-1 AND Y=-1 AND B(1,1)=1 THEN 380
370 GOTO 400
380 X=INT(RND(1)*3)-2: Y=INT(RND(1)*3)-2
390 IF (X=0 AND Y=0) OR B(X+2,Y+2)=0 THEN 380
400 X1=L: Y1=U
420 FOR C=1 TO LEN(A$)
430 X1=X1+X: Y1=Y1+Y
440 A(X1,Y1)=ASC(MID$(A$,C,1))
445 IF K=2 THEN 460
450 PRINT FN C$(Y1,X1*2);CHR$(A(X1,Y1));
```

☞

```
460 NEXT C
470 B=0: FOR X=1 TO 3: FOR Y=1 TO 3: B(X,Y)=0: NEXT Y,X
480 PRINT Y$;"6 ";J$;: GOTO 90
500 FOR X=1 TO 20: FOR Y=1 TO 20
510 IF A(X,Y)<>0 THEN 530
520 A(X,Y)=45: PRINT FN C$(Y,X*2);"-";
530 NEXT Y,X
540 PRINT Y$"6 ";J$;: INPUT"Position paper and push <RETURN>";K$
547 GOSUB 670
550 LPRINT" ": LPRINT "  WORD PUZZLE ANSWER KEY"
560 FOR I=1 TO 3: LPRINT " ": NEXT I
570 PRINT Y$"6 ";J$;"PLEASE WAIT A FEW MOMENTS . . . .";
590 FOR X=1 TO 20: FOR Y=1 TO 20
600 IF A(X,Y)<>45 THEN 620
610 B=INT(RND(1)*26)+65: A(X,Y)=B
620 NEXT Y,X
630 GOSUB 670
640 LPRINT " ": LPRINT "   COMPUTER-GENERATED": LPRINT "    WORD-SEARCH PUZZLE"
650 FOR I=1 TO 3: LPRINT " ":NEXT I: GOTO 720
670 LPRINT " "
680 FOR X=1 TO 20: FOR Y=1 TO 20
690 LPRINT CHR$(A(X,Y));" ";
700 NEXT Y: LPRINT " ": NEXT X
710 RETURN
720 LPRINT " ": LPRINT " WORD LIST:": LPRINT " "
730 FOR I=1 TO Z-1
735 CT=CT+1
740 LPRINT W$(I),
745 IF CT=3 THEN LPRINT" ": CT=0
750 NEXT I
753 LPRINT CHR$(15)
755 FOR A=1 TO 3: LPRINT" ": NEXT
760 PRINT Y$"6 ";J$;: INPUT "WOULD YOU LIKE ANOTHER COPY";K$: IF LEFT$(K$,1)="N" THEN END
770 PRINT Y$"6 ";J$;: INPUT "ADVANCE PAPER AND PUSH <RETURN>";K$: GOTO 630
780 END
```

generation, thus you will not know where the words are being put in relation to each other.

I have set up the matrix for the Heath/Zenith translation to be a 20 x 20, half the size of the original Apple version. This was a size that was decided on by the editor/publisher of our local paper for the normal puzzles. Occasionally we will run other sizes such as 40 x 15, 30 x 15, 25 x 15, and 40 x 20. The size could be changed to meet your requirements and is covered later in this article.

After your input selection of either a 1 or a 2, you will be prompted for each new word with "WORD # " with the designated number. As a help when you are making up your list of words, assign each a number, and then you will be able to keep track of what has been put in so far. You may input words, as prompted, until you finally input the word STOP. All input will cease when you enter STOP and all the remaining blank spaces of the matrix will be filled with a "-" (dash or hyphen).

At this point, you will be prompted to check if the paper is in place on your printer (you might check also to see if the printer is turned on). On pushing RETURN, a printout of the answer sheet will be given. You will be prompted again to see if you are ready and then the program will proceed to change all of the dashes with a randomly selected group of letters and will print a completed puzzle with a 20 x 20 matrix with your words well hidden. Below the puzzle, you will also see the list of the words that have been hidden in the puzzle. You will be asked if you would like another copy.

If you are having a large group or a party, you could make up a copy of the puzzle for each person to enjoy. In one of the computer classes that I teach, I use this program to make up puzzles to hide the computerese words and have the students look for them.

A few words of explanation on the program listing for those who will be wondering why things are done a certain way. The line numbering sequence is not uniform. I have tried to use all the line numbers as they appeared in the original S-80 version, except for my add in. This is done so that you could refer to the original article for any program problems.

I like to use a couple of REM lines to let me know when I make the last revision and from where I might have gotten the program. I have used the E$ for the usual ESCape sequence, CHR$(27). The CL$ is the string I set up to CLear the screen (a carry over from the TRS-80 that uses CLS). The Y$ is for using the Direct Cursor Addressing sequence. The J$ is the escape sequence for erasing to the end of the page.

Line 10 clears the screen and prints the title page by using the direct cursor positions to make it neat, and line 20 leaves it on the screen for a few seconds.

Line 40 erases all but the first line printed by line 10 and then prints the simple instructions helped out with line 50, and all lines are positioned with the direct cursor addressing.

In line 55, we used the input feature that will allow a single key input to the question and if you do not input a 1 or a 2, then you may be here all day.

In line 60, we have defined all variables from A-Z as integer numbers and we have dimensioned the strings and variables of the matrix. In line 65, I have defined as our own function name, C$ with the x and y coordinates as our variable to be used with the direct cursor addressing method, using the numbers generated by the working portions of the program to find the correct positions of the matrix.

Line 70 is put in to get the random generator started with a changing seed each time you run the program, and then at line 80 we clear the screen and start the word-entry routine in lines 90 to 100.

Line 120 sets up the random starting position for each new word. Line 160 through line 280 will check each direction to see if the word may be written in a specific direction. Lines 310 through 400 will check to see if that direction will intersect with any other words. If all is well, lines 420 through 470 will print this word on the screen, that is, unless you have chosen to use the blank screen input. Line 480 clears the word entry position so that it will be ready for the word entry loop that restarts in line 90.

If you input the word STOP, line 100 will send you to the routine that will fill in the blank spaces that remain in the puzzle with the little "-" (dashes or hyphens). In line 540, you are asked to position the paper (and maybe turn the printer on), then with a gosub to the printer subroutine in lines 670 through 710.

The complete word list of the puzzle is printed by line 730 and then you will be asked in line 760 if you would like another puzzle printed. If so, you will get another printout of the puzzle and another word list.

Now for a few changes or suggestions for alteration of the puzzle as it was presented originally. If the size of the matrix is to be changed, be sure to change the dimensions of A and C$ in line 120, as well as the integers of lines 210, 500, 590, and 680. For a matrix of 40 x 20, make the following changes:

```
   60 DEFINT A-Z: DIM W$(200),A(72,52),B(3,3),C$(72,52)
   120 L=INT(RND(1)*20)+1: U=INT(RND(1)*40)+1
   210  IF X1>40 OR X1<1 OR Y1>20 OR Y1<1 THEN B(X+2,Y+2)=0:
GOTO 270
   500 FOR X=1 TO 40: FOR Y=1 TO 20
   590 FOR X=1 TO 40: FOR Y=1 TO 20
   680 FOR X=1 TO 40: FOR Y=1 TO 20
```

I have added line 645 to LPRINT my by-line, so that I get a little credit from the newspaper. I have also changed the word list printout so that it prints the words in three columns rather than in the single column format by doing the following:

```
   730 FOR I=1 TO Z-1
   735 CT=CT+1
   740 LPRINT W$(I),
   745 IF CT=3 THEN LPRINT" ": CT=0
   750 NEXT I
   755 LPRINT" "
```

If you would like to use the 16.5 letters per inch size letters for the word list, then in line 3 add these two strings: P1$=E$+"[1w": P2$=E$+"[4w". The first is for normal printing at 10 letters per inch and the second is for the 16.5 letters per inch. Then I added line 725 LPRINT P2$ and line 757 LPRINT P1$ and changed line 745 so that CT=5 and this prints 5 columns of smaller letters.

By the way, I believe I forgot to mention that the words have the potential of being placed in the printout in eight different directions, even 45 degrees. So, if the same words are used in more than one puzzle, you would never be able to get the same results more than one time. Well, here is hoping that you enjoy this program half as much as I have, and then you will be in the puzzle making business.

NEWLINE SOFTWARE

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

SOFTWARE FOR HEATH/ZENITH COMPUTERS

HDOS 2.0 upgrade for the Z-100 now shipping!

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

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

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

HERO-1 ROBOT SOFTWARE

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

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

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

FREE CATALOGUE – Send for yours today!

NOVA WordStar(tm) Enhancement Available!

* NOVA provides a series of extremely powerful **Enhancements to your copy of WordStar.**
* NOVA enables **full use of ALL your keypad AND special function keys** to greatly increase the speed and ease of editing documents.
* Full use of special function keys displayed on the **25th line of your terminal** at all times.
* Speeds text entry by replacing those functions which would require multiple control codes.
* Requires CP/M 2.2.03. Specify hard or soft sectored 5.25" disk. NOVA is ONLY $24.95!

M-eXcellent Graphics for the MX-80!

* **Turn your MX-80 printer into a graphics plotter for your artistic or business needs** using this menu driven, highspeed, assembly language graphics package.
* Designed for the artist and the business person to display line graphs, bar charts, pie charts, **artistic designs,** and sketches.
* Requires an H/Z-89 and EPSON MX-80 printer capable of bit-plot graphics. Specify hard or soft sectored 5.25" disk, HDOS or CP/M. MX-80 graphics plotting is ONLY $59.95!

MICRO INNOVATIONS
2455 Sylvania, Toledo, Ohio 43613
VISA and MASTERCARD Orders Welcome!
Phone Orders CALL: (419) 471-1245

Generic Software

QUALITY SOFTWARE FOR HEATH/ZENITH SYSTEMS
Generic's DATA ENTRY Utilities make getting information into your computer easier!

SFDVD (HDOS) / SFCOM (CP/M) *New!* **$19.95**
The **SF** utility allows H19 screen form access to user programs written in Benton-Harbor BASIC, MBASIC, FORTRAN-80, and Assembly language. **SF** will manage all of the terminal input and display functions during program execution. Supports numeric only, alpha only, upper case, and non-echo data input fields. With **SF**, data collection and file updating can be more "user friendly."

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! The **"FORMS"** kit can be used for both sequential and random access file structures. Screen form input can greatly enhance any database/data entry application, because it simplifies data input and modification. CP/M and HDOS.

Call or write for a free Summer 1983 catalog.

P. O. BOX 790 • MARQUETTE, MICHIGAN 49855
PHONE: (906) 249-9801 • 10 AM to 5 PM EST

ATTENTION SOFTWARE AUTHORS: GENERIC SOFTWARE is interested in high quality and well documented software for Heath/Zenith computers. GENERIC offers professional packaging, international marketing, and high royalties. If you are interested in making money from the software you develop, then request our FREE booklet, ***Software Author's Kit.***

Learning To Write Assembly Language Subroutines For MBASIC Under HDOS

Fred Hochschild
1214 Deuts Ave.
Trenton, NJ 08611

Assembly language subroutines can add speed and flexibility to a BASIC program while using a minimum of memory. At times, the added speed can be all important, as in real time programs. The economy of memory is often its own reward. If you have, as I do, an H-8 with 64K of memory, only the upper 56K is recognized by HDOS and MBASIC. The lowest 8K of addresses are where the ROMs live. There is an unused 4K block of addresses between the H-8 ROM and the H-17 disk controller ROM that can be used to store the assembly language routine(s). This means that none of the memory accessible to MBASIC need be used for the routine(s).

In order to use assembly language routines, you of course need to know BASIC (this I will assume), and at least a beginning knowledge of assembly language (R. A. White's article in the March '83 REMark is a good place to start). Most important, is an understanding of how MBASIC calls assembly routines and how data is transferred from the BASIC program to the assembly language subroutine. There is really only one way to do the former, but many ways to do the latter. I will discuss only the transfer of data via the argument of the assembly language subroutine call from MBASIC.

MBASIC uses the USRn(<variable>) function to call an assembly language subroutine. There can actually be as many as ten different subroutines for MBASIC to call. These are distinguished by an integer 'n' which can be 0-9. A single variable can be passed via the USR argument. The variable can be an integer, a single or double precision floating point, or a string. Table 1 summarizes the contents of the CPU's registers for each type of data transferred.

Numeric data is placed in the FAC (floating point accumulator). The type of numeric data is identified by the contents of REG A. The FAC is eight bytes long which provides enough room for a double precision number. The HL register pair always points to FAC-3 which is the fourth byte of the FAC (the bytes are labeled FAC-0 through FAC-7). Table 2 shows how the data is arranged in the FAC. Since the example I am going to discuss does not use the FAC, I will not pursue it further at this time.

Table 1

Type of Data Transferred

Register	Integer	Single Precision	Double Precision	String
A	2	4	8	3
HL	FAC-3 address	FAC-3 address	FAC-3 address	FAC-3 address
DE	----	-------	-------	descriptor address

Table 2

	FAC-7	FAC-6	FAC-5	FAC-4	FAC-3	FAC-2	FAC-1	FAC-0
Integer*	---	---	---	---	LSB	MSB	---	---
Single Precision	---	---	---	---	LSB	thru	MSB	EXP
Double Precision	LSB			thru			MSB	EXP

Notes

*-number stored in two's complement form
LSB-least significant byte
MSB-most significant byte
EXP-exponent

When a string is passed to the subroutine, a '3' is placed in register A and the DE register pair contains the address of a three byte string descriptor. The first byte of the descriptor contains the length of the string (0-255). The second and third bytes of the descriptor contain the address of the string itself. The second byte holds the low order address, the third byte holds the high order address of the beginning of the string. This means that access to the string is several steps away (DE descriptor string). If this seems a bit confusing, it will clear up as we go through the example, speaking of which, let us begin.

Listing 1 is an assembly language routine that will provide MBASIC with a real time character input. This short but useful routine will illustrate most of the necessary concepts needed to develop your own subroutines. Let's take a detailed look at this routine.

```
TITLE     'INKEY subroutine for MBASIC'
```

A title is necessary for the assembler and should be descriptive. The term INKEY comes from a function available in some BASICs that provides the same function as this routine.

```
XTEXT     HDOS.ACM
```

Listing 1

INKEYLOW.ASM

```
TITLE   'INKEY subroutine for MBASIC'

*     This routine will allow for
* real time input in MBASIC.
* The USR should be defined to start
* at 4355 (i.e. DEFUSR=4355).
* The routine will return a null string
* when no character has been typed
* and will return the character if
* one has been typed.

        XTEXT   HDOS.ACM
INKEY   ORG     1100H
        XRA     A
        SCALL   .EXIT
        SCALL   .SCIN
        JC      NONE
        INX     D
        XCHG
        MOV     B,A
        CALL    $HLIHL
        MOV     M,B
        RET
NONE    XRA     A
        STAX    D
        RET
        END     INKEY
```

DEMOLOW.BAS

```
10 DEF USR1=&H1103
20 X$=USR1("A"+"")
30 IF X$<>"" THEN PRINT X$:
   PRINT "IT WORKED":PRINT
40 GOTO 20
```

INKEYHI.ASM

```
TITLE   'INKEY subroutine for MBASIC'

*     This routine will allow for
* real time input in MBASIC.
* The USR should be defined to
* start at 4355 (i.e. DEFUSR=4355).
* The routine will return a null string
* when no character has been typed
* and will return the character if
* one has been typed.

        XTEXT   HDOS.ACM
INKEY   ORG     0AC00H
        XRA     A
        SCALL   .EXIT
        SCALL   .SCIN
        JC      NONE
        INX     D
        XCHG
        MOV     B,A
        CALL    $HLIHL
        MOV     M,B
        RET
NONE    XRA     A
        STAX    D
        RET
        END     INKEY
```

DEMOHI.BAS

```
10 DEF USR1=&HAC03
20 X$=USR1("A"+"")
30 IF X$<>"" THEN PRINT X$:
   PRINT "IT WORKED":PRINT
40 GOTO 20
```

The XTEXT command allows you to add other files to your source code (that's what the written assembly language program is called) at the time of assembly. The file we wish to add, HDOS.ACM, is an indispensable one. It is not a program in itself, but a listing of where to find useful routines in the operating system. The System Programmer's Guide section of the HDOS manual shows what should be included in this file. It is also a good idea to include EQUate statements for the addresses of the routines in ROM. For example, we will use the routine called $HLIHL. The proper EQUate statement to use for accessing this routine is $HLIHL EQU 030211A. You can find ready made HDOS.ACM files on the distribution diskettes that come with the HA-8-3 color graphics board and the HA-8-2 music board.

INKEY ORG 1100H

The ORiGinate command tells the assembler where the program is to be put in memory. In this case, the address (1100H) is in hexadecimal and is a location in low memory between the H-8 and H-17 ROMs. You can only do this if you have RAM at those addresses. Most systems have their RAM addresses starting at the 8K address. In that case, you should ORG the program in high RAM but below HDOS. The memory used for the routine must also be protected from MBASIC when it is loaded by use of the /M: switch (more on this later). INKEY at the beginning of the line is a tag that will be referred to by the END statement.

XRA A
SCALL .EXIT

These two lines provide the normal sequence by which an assembly language program returns control to HDOS. Why, you may ask, is the exit at the beginning of the program? Good question! This is actually a neat trick for avoiding all the mess that is usually associated with loading an assembly language subroutine for BASIC. These statements are not really part of the routine that we wish to use from BASIC. This means that the useful part actually starts three bytes past the ORG address. Keep this in mind, we will need this information later. So, what will happen when the program is run from HDOS, is that it will be loaded starting at the ORG address, it will start executing and will immediately exit back to HDOS. The entire program will, however, remain in memory, ready to be called by MBASIC. By the way, the SCALL .EXIT is one of those useful routines in the HDOS operating system and should always be used to exit to HDOS.

SCALL .SCIN

This is another call to a routine in HDOS. The .SCIN routine looks in the type ahead buffer to see if a character has been typed in. If it has, the character is placed in register A and the carry flag is reset to 0. If there is nothing in the type ahead buffer, the carry flag is set to 1. All I/O for HDOS driven devices should be through HDOS routines. This eliminates the possibility of conflict with HDOS which can cause unpredictable results.

JC NONE

This is a conditional transfer command that jumps to the line labeled NONE if no character is found in the type ahead buffer.

INX D

This moves the DE register pair to point to the second byte of the string descriptor, the low order address of the string that is being used to transfer data between MBASIC and the subroutine.

XCHG

The DE register pair is eXCHanGed with the HL register pair. HL now points to the address of the string in the string descriptor.

MOV B,A

The contents of register A, the character found in the type ahead buffer, is put in register B for safe keeping. This is done because the ROM routine that is called in the next line uses register A as a working register, and our character would be lost.

CALL $HLIHL

This line calls a routine in the disk controller ROM that takes the byte that is being pointed to by the HL register pair, loads it into register L, takes the next byte, and loads it into register H. The result of this is that the HL register pair now contains the address of the string that was transferred to the routine. (Remember: DE→descriptor→string.)

MOV M,B

The contents of register B, the character we wanted, is MOVed to the memory location pointed to by the HL register pair (i.e. the string address).

RET

RETurns to the MBASIC program that called this routine, the character safely tucked in the string used to transfer data to and from the BASIC program.

NONE XRA A

If no character is found in the type ahead buffer by the .SCIN routine, control is transferred to this point. The XRA command does an exclusive OR of register A with whichever register follows, in this case register A, and places the results in register A. When you exclusive OR something with itself, the result is always zero. This is just one way of putting a 0 in register A.

```
STAX      D
```

This command causes the contents of register A to be stored in the memory location pointed to by the DE register pair. The DE register pair, remember, is pointing to the first byte of the string descriptor which holds the length of the string. So, we have put 0 in for the length of the string (i.e. a null string). This will inform BASIC that no character was found in the type ahead buffer.

```
RET
```

Again, this causes a return to BASIC.

```
END       INKEY
```

This informs the assembler that the program that began at INKEY is at an end.

If you don't have RAM in the lowest 8K of addresses, you will have to use a different ORG statement. You will want the subroutine to be located just under HDOS. You can calculate this carefully if you wish, but I will estimate. Let's say you have 48K of RAM. HDOS will use maybe the top 12K. That brings us down to 36K. This is a small routine that will require little memory but to make things easy we will reserve 1K for it. Now, the RAM addresses start at 8K. So, you should reserve the locations 43K- 44K from MBASIC (43K=35K+8K). Since 1K=1024, you get 43K=44032 (AC00 in hex) and 44K=45056 (B000 in hex). You can change the ORG statement to:

```
INKEY  ORG  0AC00H
```

The leading zero is needed to let the assembler know this is a number.

Now the source code can be entered using EDIT or some other text editor. It should be saved as INKEY.ASM. The source code can be assembled using ASM by typing >ASM INKEY=INKEY. You will now have a file, INKEY.ABS, which is an executable binary object code. If you have only one disk drive, you will need to have both INKEY.ABS and MBASIC.ABS on the same disk.

Now we're ready to load the subroutine and enter BASIC to make some use of it. Start by typing INKEY at the HDOS prompt (i.e. >INKEY). Then, how you enter BASIC depends on the ORG statement you used. If the routine is in low memory, just type MBASIC at the HDOS prompt. If the routine is in high memory, type MBASIC/M:44032(45056). This will protect the routine from being overwritten by BASIC.

You should now see the familiar MBASIC prompt, Ok. You should also notice that the amount of free memory is less than usual because some was set aside for the assembly language subroutine. Let's write a short program to make use of our new subroutine.

The first thing you need to do is to let BASIC know where the subroutine is located. This depends on the ORG statement you used. In either case, remember that the useful part of the routine starts three bytes past the ORG address. The statement used to define the starting address for an assembly language subroutine is the DEF USRn. So, depending on your ORG address, the first line of the program should be:

```
10 DEF USR1=&H1103
```

or

```
10 DEF USR1=&HAC03
```

Next, we want to call the subroutine. The subroutine will return a string of length 0 or 1. Assembly subroutines can shorten a string, but they should never lengthen it, because BASIC will not have set aside enough room for the lengthened string and it may overwrite something important. Therefore, we should pass a string of length one to the routine. It does not matter what the string is because it is only what the routine returns that is of interest. A note of caution, never use a string constant as the argument of a USR call. Any changes in the string will cause changes in your BASIC program and could really foul things up. You can force BASIC to create a string variable by concatenating a string constant with a null string in the USR argument. To illustrate, the second line of our program can be:

```
20 X$=USR1("A"+")
```

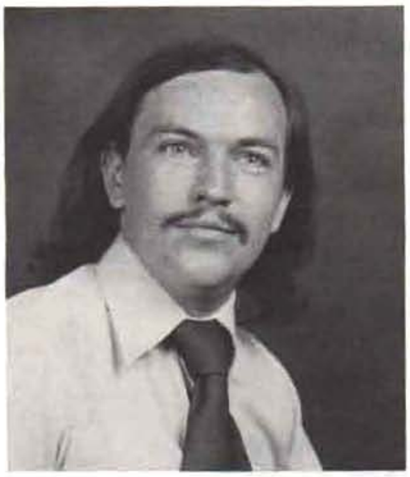
The argument of the USR1 function is a string of length one, but MBASIC must create a string variable in order to evaluate the concatenation. Every time line 20 is executed, X$ will be a null string if nothing has been typed since the last execution, and a character if one has been typed since the last execution.

To complete our little demonstration, we need to include line 20 in a loop that will check to see when X$ is not a null string and print something appropriate. Run the following BASIC program to see how the subroutine works.

```
10 DEF USR1=&H1103 (or &HAC03)
20 X$=USR1("A"+")
30 IF X$<>" THEN PRINT X$:PRINT"IT
WORKED":PRINT
40 GOTO 20
```

Now each time you strike a key, you should see that letter printed on the screen with IT WORKED printed underneath. Control C will stop execution.

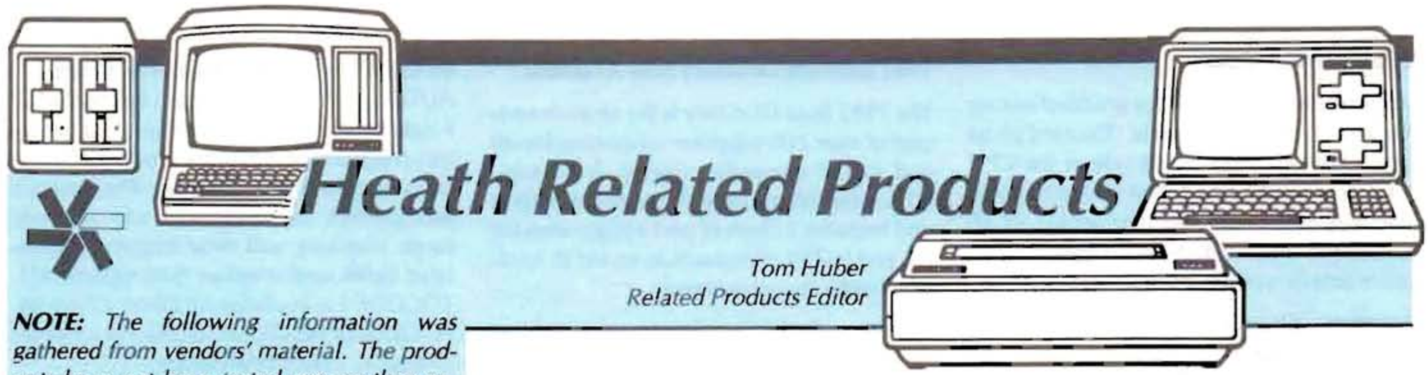
Well, that ends my little tutorial on assembly language subroutines. This has really only scratched the surface, but I hope it will enable you to pursue some subroutines on your own. I think you will find that once you get used to assembly language, its a lot of fun.



About the Author:

***Fred Hochschild** is a chemistry teacher at Notre Dame High School in Larenceville, NJ. He received his BS in Physics from Stevens Institute of Technology in 1976. His interests other than computers and his teaching range from astrophysics to adventure gaming. He belongs to the Mercer Adventure Gamers club (MAG). Fred spends a lot of his time reading and likes to study textbooks on astrophysics in his spare time. He has had his H-8 for only a few months now and hopes to learn a great deal in this year.*

SEE PAGES 26 + 27 FOR NEW HUG PRODUCTS



Heath Related Products

Tom Huber
Related Products Editor

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

FONTGEN Aids H-8 & H-89 Color Graphics

FONTGEN, including a font generator for creating and modifying character sets, allows users to send text easily to their graphic monitor. The software is compatible with Pascal MT+, Lucidata Pascal, and M-80 graphics capable systems. For more information, contact the vendor.

Vendor: Straightshooter Computer Prod.
1855 NE Dixie Hwy.
Jensen Beach, FL 33457
(305) 334-1755
Price: $12.00

5 Megabyte Subsystems Available for '89 & '100 Products

Computer Dynamics has announced a series of 5M mass storage systems built around the Shugart 8-inch SA1002 5.33M Winchester disk drive. The Controller card is manufactured by Western Digital and will support up to four drives. The WIN-5 is an assembled, tested system that includes software. Available for the IBM PC, S-100 (Z-100), STD BUS, H/Z-89/90, Apple II, and others. For complete details and pricing (including kits), contact the vendor and specify your computer model.

Vendor: Computer Dynamics Incorporated
105 S. Main Street
Greer, SC 29651
(803) 877-7471
Price: $1695.00 WIN-5.

User Group Digital Cassette Source

NDC is a supplier of digital cassettes, including DATALOCK at special prices to User Groups. Their services also include duplication services. Contact the vendor for full details.

Vendor: National Distribution Center
117 West 23rd Street
Independence, MO 64055
(816) 254-0400

New Products from Generic Software

DEPRECIATION-MASTER is a comprehensive set of program modules that allow the user to maintain asset inventories and depreciation schedules. Included is the capability to add, delete, and modify items in multiple asset files inventory using screen "forms". Depreciation schedules can be created, modified, and deleted under user-specified methods or the twenty-seven standard ACRS depreciation schedules that are supplied with the program. The system requires MBASIC, 64K RAM under CP/M(-85) or 56K RAM under HDOS, and two disk drives.

MAIL-IT II, a new version of MAIL-IT, is a database editor for name, address, and phone number information. The new version provides a new assembly language sort routine, screen "forms" control, 1- to 8-across labels, an update mode that supports common data between records, single record LIST option in update mode, file directories while in the program, and file extension capabilities for expanding existing files. System requirements are 56K RAM, HDOS or CP/M, and MBASIC.

Vendor: Generic Software
P.O. Box 790
Marquette, MI 49855
(906) 249-9801
Prices: DEPRECIATION-MASTER
$95.00 + $4.00 S&H
MAIL-IT II
39.95 + $2.00 S&H
(MI residents add 4% sales tax)

Digital Research Sponsers CP/M'83 East

CP/M'83 East will be held September 29th through October 1st, 1983, in Boston's Hynes Auditorium. Industry workshops (for the trade), panel discussions, and seminars are scheduled for each day of the show. For more details contact the producer.

Producer: Northeast Expositions Inc.
824 Boylston Street
Chestnut Hill, MA 02167
(617) 739-2000

Future Technology Offers Extender Board for H/Z-89/90

An extender board, allowing the user to test and troubleshoot both sides of I/O and controller boards is now available from Future Technology, Inc. The board, which allows the board to be tested above the computer also contains test points for all leads and should be good for service center use as well as for individuals. Available assembled or in kit form. Contact the vendor for more information.

Vendor: Future Technology, Inc.
13 New Salem Street
Wakefield, MA 01880
(617) 245-4223
Prices: $30.00 Assembled
$25.00 Kit

CDC Disks Available in Case Lots

Control Data Corporation disks are now available in all popular 5.25- and 8-inch configurations. These premium disks meet or exceed ANSI, EMNA, ISO, and Shugart specifications, include hub rings, and are individually factory tested. Contact the vendor for full pricing information.

Vendor: Midwestern Diskette
Distribution Center
1301 Clayton Road
Creston, IA 50801
(515) 782-5190
Prices: Contact Vendor

Z-ZAP Z-DOS Disk Utility

Z-ZAP is a powerful utility that can be used to access a disk by absolute sector. It will display sector content in hexadecimal, octal, decimal, and ASCII formats; provides a buffer area for editing, copying, and filling sector contents; output to a printer; and writing back to disk. Z-ZAP requires one disk drive, Z-DOS, and 128K Memory.

Vendor: Westcomp
P.O. Box 494
Temple City, CA 91780
Price: $29.95 + $2.00 S&H
(CA residents add 6% sales tax)

H-89 Extender Card

An extender card, useful for troubleshooting the H/Z-89 is now available. The card plugs into either the right or left side of the CPU card. The card being tested then plugs into the extender card, allowing access to all components and signals on the test card. For more details, contact the vendor.

Vendor: Ohio Digital Systems, Inc.
Dept. H
P.O. Box 252
Dayton, OH 45404-0252
(513) 233-0326
Price: $29.95 + $1.00 S&H
(OH residents add 6% sales tax)

Z-100 Software

Software Toolworks is now providing a list of Z-100 compatible software from that company. The software includes C/80 Compiler, UVMAC Macro Assemblers, PACK and CRYPT, TEXT, LISP, RATFOR, ELIZA, SPELL, ADVENTURE, COMPUTER CHEF, AUTODIFF, PIE, ED-A-SKETCH, ZENCALC, MUNCHKIN, SNAKE, SUPER ZAP, SPACE PIRATES, AIRPORT, and WORD WIGGLE. For more information, contact the vendor.

Vendor: The Software Toolworks
15233 Ventura Boulevard
Suite 1118
Sherman Oaks, CA 91403
(213) 986-4885

Livingston Logic Software

CDRBIOS CP/M for the CDR FDC-880H double density floppy controller (H/Z-89 systems) is a replacement for the standard Heath/Zenith BIOS. It allows the FDC-880H to be used with the H-17 under CP/M 2.2.03 and supports all H-37/47 formats.
Version 4.0 of BIOS-80 contains many enhancements over previous releases and allows direct SYSGEN operations between any two BIOS-80 disks.
The CDRDVD HDOS device driver for the CDR FDC-880H has been revised to take care of a problem previously reported.

Vendor: Livingston Logic
P.O. Box 5334
Pasadena, CA 91107
(213) 303-5342
(5 PM to 8 PM weeknights)
Prices: CDRBIOS
$60.00 + $2.00 S&H
BIOS-80 ver 4
$40.00 + $2.00 S&H
$10.00 (update-return your original disk)
CDRDVD driver update
no charge (return your original disk)

1983 Software Directory Now Available

The 1983 Buss Directory is the seventh revision of over 260 suppliers supporting Heath and Zenith computer systems. It contains 20% more listings than the previous edition and includes a chart of port assignments for H8 and H/Z89 computers as an aid to hardware and software designers.

Vendor: BUSS
716 E Street, S.E.
Washington, DC 20003
(202) 544-0900
Price: $12.50

Query!^{2} General Database Management System

Query!^{2} allows the user to create up to 255 fields of 255 characters each. A screen hold feature prevents records with more than 22 fields from scrolling off the screen. The printer program will support 1 to 4 across labels, continuous or single sheet feed, test pattern for forms alignment, and special format creation for labels, as well as other common features. Contact the vendor for full details.

Vendor: Hoyle and Hoyle Software, Inc.
716 S. Elam Avenue
Greensboro, NC 27403
(919) 378-1050
Price: $29.95

Two New Software Packages from F & F Software

Math of Finance is designed to easily solve quantitative financial problems. Functions include present and future values of a single payment or an annuity (fixed payment on a time basis), sinking fund payment, and installment loan and amortization schedules.
General Purpose Decision Maker helps make decisions concerning major purchases, relocation (job, business, home), project selection, priority establishment, and investment alternatives. The user assigns weights to factors which reflects the relative importance of each factor and then assigns a rating to each factor. The program uses the weights and ratings to calculate a relative value to each item.
Both programs are available for CP/M-80, Z-DOS, or PC-DOS (IBM- PC). Specify computer and operating system when ordering. For more details on these programs, contact the vendor.

Vendor: F & F Software
3632 Evans Road
Atlanta, GA 30340
Prices: Math of Finance $17.95
G.P. Decision Maker . . $24.95
Both $34.95

dBASE II™ Program Generator Available

AUTOCODE 1™ is a program for the dBASE II data base management system that generates programs based on the answers of questions supplied by the program. The program incorporates alphanumeric and numeric range checking and error trapping, calculated fields, and selective field reports. AUTOCODE 1 is available for CP/M, CP/M-86, PC-DOS, or MS-DOS(Z-DOS). For more information, contact the vendor.

Vendor: Alex Johnson Corporation
Attn: Malcom McVickar
666 Howard Street
San Francisco, CA 94105
(415) 777-3800
Orders:
(800) 227-1617, ext 417
(800) 772-3545, ext 417 (Calif.)
Price: $195.00

More Heath/Zenith Products

Joy-8 is an analog joystick that connects directly to the NOGDS HA-8-3 color graphics board.
Joy-89 connects to the NOGDS HA-89-3 color graphics board.
Joy-Kit is a kit version of Joy-8/Joy-89.
H-Sink is a heat sink assembly for H-8 boards that get too hot for the mounting bail to dissipate the excess heat. They mount on all standard 6 by 12 inch H-8 boards.
HERO T-SHIRT is a first quality cotton/polyester blend with blue collar and sleeve rings, a picture of HERO silk screened to the front with the message, "Have you HUGged your HERO today??". Specify size—S, M, L, or XL—when ordering.

Vendor: Weitzman Associates
580 N.W. 99 Way
Pembroke Pines, FL 33024
Prices: Joy-8 . . $14.00 + $1.50 S&H
Joy-89 . $14.00 + $1.50 S&H
Joy-Kit . . $9.00 + 1.50 S&H
T-Shirt . $8.95 + $1.00 S&H

Serial to Parallel Converter

The Serial to Parallel converter is designed to interface standard RS-232 printer signals with those printers that have the standard Centronics parallel input. The hardware device uses +5 Volts from pin 18 of the standard centronics printer interface for power and may be driven by CP/M or the universal HDOS driver. For more information, contact the vendor.

Vendor: Micro-Applications
35 W. Wainwright Drive
Poquoson, VA 23662
(804) 868-8764 (after 6 PM)
Price: $95.00 + $3.00 S&H

sions and, as he states, may only work with Zenith CP/M 2.2.03.

I have a much simpler way that will automatically return the user to BASIC, his or her program intact, use no disk space (save for one directory entry, no bytes!), will require no calculations or reading of BASIC tokens in RAM, does not require DDT and is, as far as I've been able to tell, foolproof on ALL versions of CP/M.

Let us create the BDOS ERROR condition Mr. Simpson speaks of in his article. First, place the diskette with MBASIC on it into your default drive and bring up BASIC. Remove the disk and write a short program such as:

10 REM Nothing to it

Now, placing in a different diskette, type 'SAVE "MYPROG" <cr>'. This should result in the message "Bdos Error On d:" where 'd:' is the default drive. Press any key and you'll end up back with the CP/M prompt and an apparently lost program.

Okay, without DDT, or looking at memory, just type:

SAVE 0 CONT.COM <cr>

Now, type:

CONT <cr>

At this point, you should see the MBASIC "Ok" prompt back and typing "LIST" should list out your program without a character lost! You can now type "RESET" (although it is no longer required, as a warm boot has already been performed) and the program can now be saved as it normally would.

And that is it. No calculations, no examining memory, moving RAM contents, finding beginning or endings of programs, checking on BASIC tokens and strings...nothing. Just a simple, automatic procedure that not only will help save the BASIC program, but will return the user back into BASIC as well!

In addition, once CONT.COM is saved out on the diskette, it can be used to get the user back into many (but not ALL) programs that would otherwise be lost.

Although it is not necessary to understand WHY this works, the explanation is as follows.

When a warm boot is performed, most of the TPA (Transient or user Program Area) is not affected. Thus, MBASIC and the user's program should still be there. The 'SAVE' command is an intrinsic CP/M command and does not need to be loaded from disk. By typing the command

SAVE 0 CONT.COM <cr>

a file is created on the disk called CONT.COM with zero pages of bytes saved. In other words, this is a null file...it contains nothing. Now, in CP/M, when a user types the name of a transient command (such as MBASIC), the program is loaded at the FWA or, for standard CP/M, 0100H. It will then run this program.

Remember in our example that we have previously loaded MBASIC which, at the moment, is still out there at 0100H. By typing CONT, CP/M will try and load the program CONT at 0100H and jump to that point. Since CONT contains no bytes, it takes up no disk space and only one directory entry. It can be created at any time the user finds he is dropped from a program and can be left on the diskette or removed as necessary.

To sum it up, here are my steps to BASIC Program Recovery caused by a BDOS ERROR:

1. SAVE 0 CONT.COM <cr>
2. CONT <cr>

That's it. Hope this will satisfy the matter for anyone who has made that BDOS ERROR.

Craig A. Pearce
2529 S. Home Avenue
Berwyn, IL 60402

Corrections to Hayes Chonograph Article

Dear HUG,

It has been brought to my attention that several typos got through to my Hayes Chronograph article which appeared in REMark Issue #35, December, 1982. Please make the following corrections to your copy:

(1) Page 10, left column, near middle, change the line:

```
    DB     16     ;LOCAL STACK SPACE
to
    DS     16     ;LOCAL STACK SPACE
```

(2) Page 10-11, references to CB12: should be CBI2: (see-bee-EYE-two).

(3) Page 11, left column, near middle, change the line:

```
    SETDDV   DB       'ATDT/',CR     ;SET DATE DIVIDER
to
    SETDDV   DB       'ATVD/',CR     ;SET DATE DIVIDER
```

(4) Page 12, left column, near top, replace the lines:

```
    INX      H      ;POINT TO CHARACTER ENTERED
    INX      H
    MOV      A,M    ;GET IT AND SEE IF IT WAS A 'P'
with
    LDA      DMA+2     ;GET  CHARACTER  ENTERED  AND
CHECK FOR 'P'
```

(5) Page 13, left column, change the line:

```
    170 PRINT"12. Remove the date separator"
to
    170 PRINT"12. Remove the time separator"
```

(6) For users' with BIOS 2.2.03, do NOT remove the EI instruction near the label CBI2:.

Please accept my apologies for any inconveniences I may have caused.

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

Transferring HDOS Text Files to ZDOS

Dear HUG,

My H89 has been providing faithful service for 3 years, thanks in part to the useful hints and articles appearing in REMark. It is now headed for retirement, giving way to a new H120.

In changing over to the H120, the first problem I encountered was in transferring my HDOS text files to ZDOS. The friendly people at the Heathkit store were not thrilled at the prospect of copying these files for me because they would first have to be converted to H89 CP/M, then to H120 CP/M, and finally to H120 ZDOS. I took pity on them and decided to transfer the files using a serial port. The following instructions may be useful to your readers as one way to accomplish the transfer.

H89:

Copy the "ATH84.DVD" device driver from the HDOS 2.0 Software

Tools disk to your working disk, renaming it "AT.DVD". Don't forget to "BYE" and re-boot afterwards.

Set the following options on AT:

SET AT: NOMLC
SET AT: WIDTH 132 (It wouldn't accept 255)
SET AT: PORT 340
SET AT: BAUD 4800 (Or any other baud rate you desire)

H120:

Set the following options for AUX with CONFIGUR:

I	User defined
B	Serial device
N	Do not strip parity on input
N	Do not strip parity on output
N	Do not map lower case to upper on input
N	Do not map lower case to upper on output
B	Serial B(J2) (ECH)
M	4800 BAUD
A	No handshaking
A	1 stop bit
N	No parity
C	7 bit words
(RETURN) (0)	No padding required
F	Make changes to both disk and memory
C	Exit program

Connect the H89 and the H120 together with a serial cable (male/female, only pins 1,3, and 7 are used). Use the port 340 connector on the H89 and the J2 connector on the H120. Now, with an editor or word processing program, put the CONTROL-Z at the end of each file you wish to transfer so that ZDOS will recognize the end of the file.

To copy a file, type "COPY AUX FILENAME.EXT" on the H120 and "COPY AT:=FILENAME.EXT" on the H89. Always remember to start with the H120 entry first because there is no handshaking and the H89 will transmit the file whether the H120 is ready or not.

I am sure that there are many other ways to accomplish a file transfer, but this method required no software in addition to HDOS (with an editor) and ZDOS, and took only a few minutes to implement. However, the editor must be able to enter control codes (Heath's EDIT will not). Perhaps someone can suggest a way to transmit the CONTROL-Z with HDOS 2.0 distribution software without resorting to a BASIC or ASM program.

MBASIC programs may be transferred (when SAVEd in ASCII), LOADed into ZBASIC, and run (except for some terminal command differences). The only problem I encountered involved the MBASIC "@" when loading but does not see the <CR> in the ASCII file, causing a "Direct statement in file" error or possibly an error when executed. To avoid this problem, simply edit out the @'s in the ASCII file before loading.

I hope this information will be useful for those "HUGgies" like myself who are graduating from their hard sectored H89 with HDOS to an H120.

Larry W. Oberholtzer
2704 East Melrose
Walla Walla, WA 99362

More on 'C' Language

Dear HUG,

Thank you for the article in REMark issue #42, "Usinc 'C' For Fast Action Games". I hope to see more articles about this facinating language.

The author, Mr. Clement S. Pepper, posed a question for which I would like to provide an answer. The printf statement which would not work:

printf("[e[x5[x1[Y5648[p C IS THE GREATEST [q[y5");

should be replaced by

printf("[e[x5[x1[Y80[p C IS THE GREATEST [q[y5");

which will work. In this case, you must use the ASCII equivalent to set the cursor using direct cursor addressing. Thus the 25 has a decimal equivalent of 56 (plus 31) and the ASCII equivalent of 56 is 8. By the same line of reasoning, the decimal equivalent of 48 (17th column) has an ASCII equivalent of 0.

I would also like to suggest an improvement in the program given in Listing 2 on page 36 of the article which makes use of [j (save cursor position) and [k (set cursor to saved position). Combining this with the changes noted above results in a new Listing 2 program as follows:

```
/* PUTTSTA.C 10-23-82 A printf screen printing test program */
#include "printf.c"

main()
    printf("[E[j[x5[x1[Y80[p C IS THE GREATEST [q[y5[k");  |

/* Note: [ is escape representation. If using PIE will be reverse video.
*/
```

When the program completes running, the cursor and the CP/M prompt will now go to the top of the screen in the usual left hand corner rather than being fixed on the 25th line.

Charles J. Gillies
1050 Crestview Dr., H51
Mountain View, CA 94040

Correction to SYSET Program Published in Issue #42

Dear HUG,

Thanks for publishing my SYSET program in REMark Issue #42. Some lines were somehow omitted, however. The list of HDOS SYSCALLs on the first page of the source code were left out. Was this intentional? If so, it might be a bit confusing to an early learner of assembly programming. I can't see how else it could have happened, since you evidently just photocopied the printout I sent. Perhaps you might find it useful to print the list of definitions in the next issue. At any rate, I hope your readers will find it handy, and you might even consider including it along with the HSY.DVD in future editions.

William N. Tavolga
5151 Windward Avenue
Sarasota, FL 33581

*[ED: Sorry Bill, this information was inadvertently deleted. Here it is for those who need it. The lines below should be inserted right after the remark statement **HDOS system calls.**]*

```
.EXIT  EQU  0     return to HDOS
.SCOUT EQU  2     output char. to console
.PRINT EQU  3     print line on console at addr. in HL
.READ  EQU  4     read from opened disk file
.OPENR EQU  42Q   open file to read
.POSIT EQU  47Q   position at sector # in HL
.ERROR EQU  57Q   call error file
```

H8 Service Note

Dear HUG,

My H8 is nearly six years old, and has never been turned off in that time other than for service or board replacement, etc. It has become increasingly susceptible to A.C. line 'glitches' over the last three years. Recently, the problem became much more frequent, and a second symptom appeared. The front panel 7-segment LED displays became dim to the point where the multiplexing action was visible.

Measuring the voltage on the 8 volt buss line indicated it to be down to 6.5 volts with the 'LOW/NOR' position, and this voltage was changing erratically by as much as plus/minus 0.5 volts.

Removing any board would allow the voltage to rise to about 7.5 volts with the same erratic change. (Several years ago I had to put the 'LOW/NOR' switch in the 'LOW' position because the buss voltage had dropped below 8 volts. I have all but two slots filled with boards.)

On a hunch, I checked for any voltage drop between the heatsink and either of the two diodes pressed into it (Heath part #57- 35). One diode showed a drop of 0.5 volts, and the other a drop of nearly 5.0 volts! With the transformer leads disconnected from the diodes, I measured a resistance of 2 ohms between one diode case and the heatsink, and 14 ohms between the other and the heatsink! Both diodes had been silver plated which should have been compatible with the aluminum heatsink they were pressed into, however, both had tarnished badly (and I live in an area where air polution isn't a problem!) to the point where they were black.

The solution is to clean the end of each diode case where it protrudes through the heatsink, tin it with solder, and solder a short length of heavy braid or wire to each/both diodes. The other end of the braid/wire should have a #6 solder lug attached to it, and this lug attached to the upper mounting bolt of the heatsink where the negative lead from the filter capacitor (Heath part #25-842) is attached.

The buss voltage after this change is now 11 volts with the 'LOW/NOR' switch in the 'LOW' position and 9 volts with the switch in the 'NOR' position, and with five slots filled.

The H8 will now function correctly with the A.C line input voltage reduced to as little as 80 volts, and appears to be completely immune to A.C. line noise.

W. H. Craig
P. O. Box 311
Grayson, KY 41143

Public Domain "C" Programs

Dear HUG,

I was happy to see Clement Pepper's article on C/80 in Issue #41. I had just ordered C/80 from the Software Toolworks and set a goal of learning C by the end of the summer.

However, in his list of references, Mr. Pepper omitted an invaluable resource, the C Users' Group. The group maintains a library of about 30 volumes of public domain C programs in 6 or 7 different disk formats, including Heath. In addition, it publishes a newsletter at irregular intervals. The address for the group is:

C Users' Group
112 N. Main
Yates Center, KS 66703

The one year membership fee is $10.00.

David Tichane
60 Plaza Street, Apt. 4B
Brooklyn, NY 11238

Assembly Language

Dear HUG,

I recently became interested in assembly language programming and found that the distribution disks for the HA-8-2 and HA-8-3 have the HDOS.ACM files on them. This can save a considerable amount of typing. These files also contain a list of address constants for useful routines in the disk controller ROM. However, I found one very small mistake. The comments on the $DADA and the $DADA. equates are reversed. The correct lines are as follows:

```
$DADA   EQU   030072A   (HL)=(HL)+(A) REG A IS NOT DESTROYED
$DADA.  EQU   030101A   (HL)=(HL)+(A) REG A IS DESTROYED
```

I hope this is of some use to someone.

Fred Hochschild
1214 Deutz Ave.
Trenton, NJ 08611

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

CUT ALONG THIS LINE

HUG MEMBERSHIP RENEWAL FORM

When was the last time you renewed?

Check your ID card for your expiration date.

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

Name ______________________

Address ______________________

City-State ______________________

Zip ______________________

REMEMBER - ENCLOSE CHECK OR MONEY ORDER

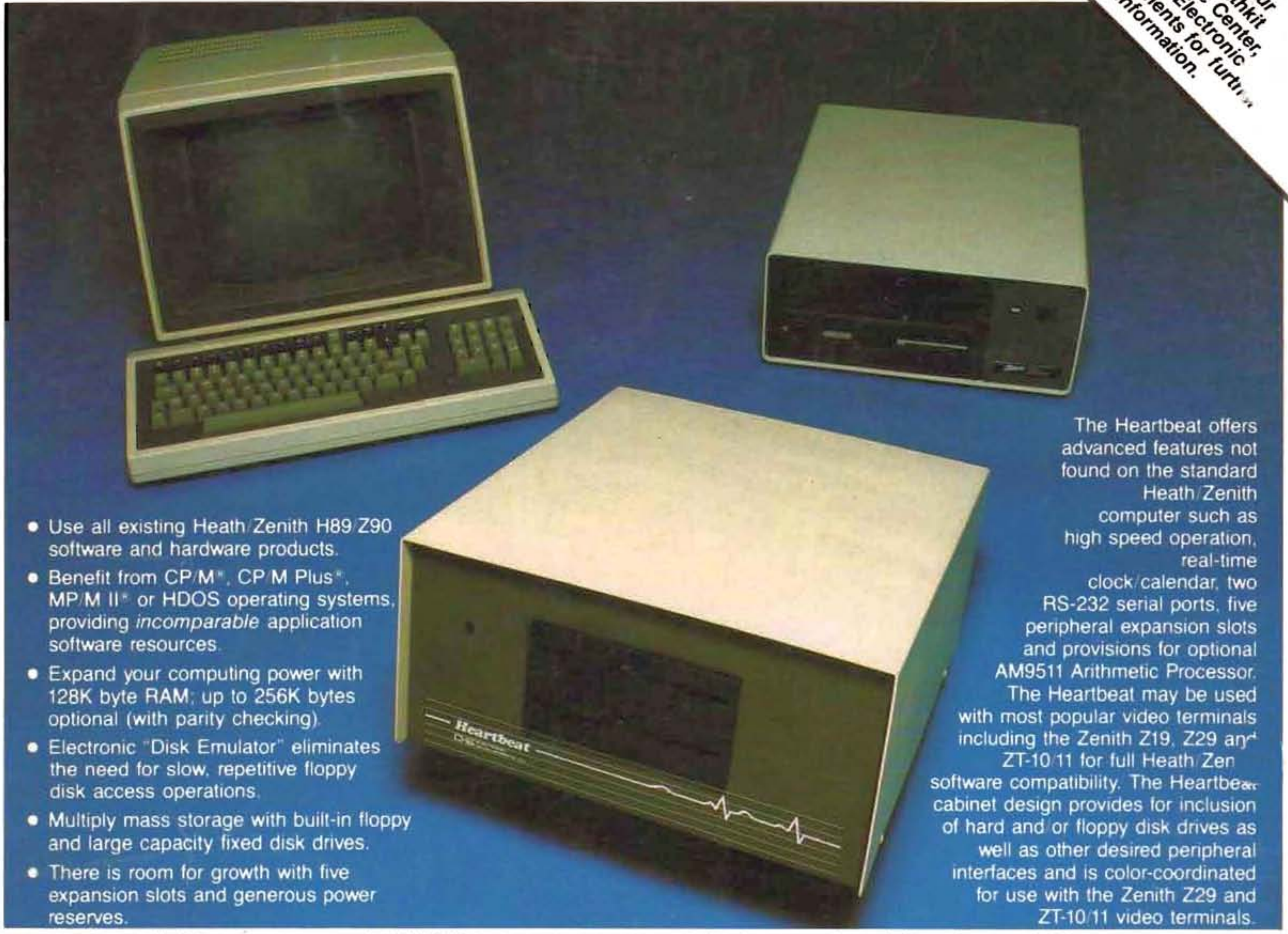
CHECK THE APPROPRIATE BOX AND RETURN TO HUG

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

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

In a Heartbeat...

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



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

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

D·G ELECTRONIC DEVELOPMENTS CO.

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

Heath Users' Group
Hilltop Road
Saint Joseph, Michigan 49085

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

Volume 4, Issue 9

POSTMASTER: If undeliverable, please do not return.

885-2044