


\$2.50

# REMark<sup>®</sup>

Volume 5, Issue 8 • August 1984  
F/N 885-2053



Official magazine for users of  computer equipment.

# Support and More

## Energize your tired old computer with...

### H8 PRODUCTS

The Most Extensive Line of Hardware Support for the H8\*

- **DG-80/FP8**  
Z80® based CPU including the powerful FP8 monitor — both only \$199.00. The acclaimed FP8 monitor package is included with the DG-80 CPU.
- **DG-64D/64K RAM Board**  
Reliable, Low Power, High Capacity Bank-selectable RAM  
Priced from \$233.00 (0K) to \$299.00 (64K)
- **DG Static 64**  
Fully Static, High Capacity, Bank-selectable RAM. Also can be used as EPROM/PROM board (2716 type EPROMS). Priced from \$199.00 (0K) to \$499.00 (64K).
- **DG-32D/32K RAM Board**  
Low cost, Dependable RAM for the H8 32K Version Only \$179.00.
- **DG-ADP4**  
H17-4 MHz disk adaptor — \$19.95



### THE SUPER 89

The DG SUPER 89 is a replacement central processor board for the Heath/Zenith 88-89 series of computers. The DG SUPER 89 offers advanced features not available on the standard Heath/Zenith 88-89 such as 4 MHz operation, real-time clock, optional AM9511A arithmetic processor, up to 256K of bank selectable RAM with parity check, and HDOS, CP/M and MP/M compatibility. By incorporating the state-of-the-art technology of the Z80, the DG SUPER 89 offers the user increased speed and system reliability for years to come. Super 89 BIOS included. Full compatibility with all Heath/Zenith software and hardware products is designed into the DG SUPER 89. Priced from \$829.00 (128K) to \$989.00 (256K).

### HEARTBEAT

The DG Heartbeat is a compact computer system designed to be hardware and software compatible with the popular Heath/Zenith Z89/90 computer product line. The Heartbeat offers advanced features not found on the standard Heath/Zenith computer such as 4 MHz operation, real-time clock/calendar, two RS-232 serial ports, five peripheral expansion slots, 128 Kbytes (expandable to 256 Kbytes) parity checked RAM and provisions for an optional AM9511 Arithmetic Processor. Compatible with HDOS, CP/M and MP/M II (Multi-user) operating systems. Super 89 BIOS included. The Heartbeat may be used with most popular video terminals on

the market although the Heath/Zenith H/Z19, H/Z29 and ZT-10/11 video terminals are recommended for full Heath/Zenith software compatibility. The Heartbeat cabinet design provides for inclusion of hard and/or floppy disk drives as well as other desired peripheral interfaces and is color-coordinated for use with the Zenith Z29 and ZT-10/11 video terminals. Priced from \$1350.00 (Basic Unit).

### SUPER 89 BIOS

The Super 89 BIOS is a greatly expanded operating system interface fully compatible with that provided with H/Z CP/M for use with the Super 89 and Heartbeat. Support is provided for all H/Z disk drive systems as well as Magnolia Microsystems 8"/5" disk controller, CDR systems FDC-880H, and hard disk systems using the popular XEBEC Winchester controller.

Expanded facilities of the BIOS include a built-in RAM Disk with capacity up to 170 Kbytes; a 48 Kbyte RAM print spooler; transparent operation at CPU clock rates of 2 or 4 MHz; increased TPA (Transient Program Area) for larger user programs; detailed error messages and user selection of recovery options; real-time clock support; simplified, self-prompting, configuration utility that is truly user-friendly; also, included are several useful utilities for expanded system operation.

The Super 89 BIOS is available separately, \$60.

CP/M®, MP/M and MP/M II® are registered trademarks of Digital Research of Pacific Grove, California.

H88/89®, Z89/90®, H17®, H77®, H/Z 47®, Z67® and H-88-1® are registered trademarks of the Heath Company and Zenith Data Systems.

Z80® and Z80A® are the registered trademarks of Zilog Corporation.

**D·G ELECTRONIC DEVELOPMENTS CO.**

**Ordering Information:** Products listed available from DG Electronic Developments Co. 700 South Armstrong, Denison, Tx 75020. Check Money Order. VISA or MasterCard accepted. Phone orders call (214) 465-7805. Freight prepaid. Allow 3 weeks for personal checks to clear. Texas residents add 5%. Foreign orders add 30%. Prices subject to change without notice.



**Staff**

Manager ..... Bob Ellerton  
(616) 982-3867  
Software Engineer ..... Pat Swayne  
(616) 982-3463  
Bulletin Board and  
Software Developer ..... Jim Buszkiewicz  
(616) 982-3463  
Software Coordinator ..... Nancy Strunk  
(616) 982-3838  
Secretary ..... Margaret Bacon  
(616) 982-3463

**REMark**

Editor ..... Walt Gillespie  
(616) 982-3789  
Editorial and Advertising  
Assistant ..... Donna Melland  
(616) 982-3794  
Graphics and Layout  
Assistant ..... Greg Martin  
(616) 982-3463  
Printers ..... Imperial Printing  
St. Joseph, MI

	U.S. Domestic	Canada & Mexico	International
Initial	\$20	\$22*	\$30*
Renewal	\$17	\$19*	\$24*

\*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 available 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/Zenith Users' Group  
Hilltop Road  
St. Joseph, MI 49085  
(616) 982-3463

Although it is a policy to check material placed in REMark for accuracy, HUG offers no warranty, either expressed or implied, and is not responsible for any losses due to the use of any material in this magazine.

Articles submitted by users and published in REMark, which describe hardware modifications, are not supported by Heathkit Electronic Centers or Heath Technical Consultation.

HUG is provided as a service to its members for the purpose of fostering the exchange of ideas to enhance their usage of Heath equipment. As such, little or no evaluation of the programs or products advertised in REMark, the Software Catalog or other HUG publications is performed by Heath Company, in general and HUG in particular. The prospective user is hereby put on notice that the programs may contain faults the consequence of which Heath Company in general and HUG in particular cannot be held responsible. The prospective user is, by virtue of obtaining and using these programs, assuming full risk for all consequences.

REMark is a registered trademark of the Heath/Zenith Users' Group, St. Joseph, Michigan.

Copyright © 1984, Heath/Zenith Users' Group

## on the stack

<b>Buggin' HUG</b> .....	7
<b>My Favorite Subroutines</b> .....	9
<b>Heath Company Introduces HERO Jr.</b> <i>Bob Ellerton</i> .....	11
<b>Inside Microsoft™ BASIC</b> <i>D.D. Dodgen</i> .....	13
<b>Getting Started with Assembly Language #11</b> <i>Pat Swayne</i> .....	19
<b>More About CP/M Disk Errors</b> <i>Norman E. Lavigne</i> .....	23
<b>Practical File Management Part 3</b> <i>David E. Warnick</i> .....	27
<b>Patch Page</b> <i>Pat Swayne</i> .....	30
<b>PALETTE - A Powerful Graphics Package for the H/Z100</b> <i>Tom Huber</i> .....	31
<b>New HUG Products</b> .....	34
<b>Local HUG News</b> .....	36
<b>HUG Price List</b> .....	36
<b>Spreadsheet Corner Part 2</b> <i>H.W. Bauman</i> .....	37
<b>Clock Watchers Delight #3</b> <i>Pat Swayne</i> .....	43
<b>COBOL Corner X</b> <i>H.W. Bauman</i> .....	49
<b>HERO SEEKS A Friend</b> <i>Dr. Kenneth R. Hill</i> .....	52
<b>SIG News</b> <i>Bill Parrott</i> .....	55
<b>An Introduction to 'C' Part VII</b> <i>Brian Polk</i> .....	57
<b>Update on Heath/Zenith Related Vendors</b> .....	62

**ON THE COVER:** Pictured is the Heath's latest addition in the field of Robotics, HERO JR.

See article on page 11 for more information.

QUIKDATA INC.

# WE'RE NUMBER ONE BECAUSE YOU'RE #1

Quikdata has been in the computer mail order and newsletter business since shortly after the first Heath H8 computer kits were rolling off the assembly line. When this business was started we made a promise to ourselves - our customers were not to be ignored. We didn't want to be your ordinary mail order business. We implemented a unique set of guidelines to operate by in order to give you fast and efficient service at lower than average prices: 1) We don't sell anything we would not use ourselves; 2) Full stock of what we carry means 24 hour shipping in most cases; 3) We stand behind everything we sell; 4) We are available for software and hardware support and consultation; 5) We repair what we sell; 6) We only support one line of computers, the H/Z line - the one we consider the best. Quikdata has been instrumental in introducing lots of hardware and software to the H/Z world and there is more to come. Your confidence in us has made us the major independent supplier of software and hardware for H/Z computer systems, and we take this space to thank you. Sure, we could fill this page with our major account names to try to impress you, but that's not necessary. At Quikdata, no matter who you are, no matter how large or small your account, **WE TREAT YOU AS IF OUR BUSINESS DEPENDS ON YOU - BECAUSE IT DOES!**

## YOUR DISK DRIVE NEEDS

Our specialty is disk drives. We carry a full line of disk drives, cabinets, cables, and diskettes.

Tandon <b>TM100-1</b> SS ST	\$179
Tandon <b>TM100-2</b> DS ST	\$224
Tandon <b>TM101-4</b> DS DT	\$279
Tandon <b>848-2E</b> 8" DS DD thin	\$395
Mitsubishi <b>M4851</b> 5" ST thin	\$239
Mitsubishi <b>M4853</b> 5" DT thin	\$259

**DUAL 5" CABINET** with fan, power supply and cables - \$175.

**SINGLE 5" CABINET** \$59 (add \$20 for cable).

**COMPLETE DUAL 8" SUBSYSTEM** for use with H89 with MMS controller, Z100 systems and some other controllers. Ready to plug in and go. Gives a total of 2.5 megs storage - \$1009. Same, but one drive system - \$619.

**QUIKSTOR 15 MEG WINCHESTER FOR H8 & H89:** Subsystem ready to plug into H8 with WH8-37 controller or H89. Can handle both HDOS and CP/M software with up to 15 user defined partitions, each which can be bootable - \$1395.

HDOS & CP/M (2.2.03) software - \$149.

## MISCELLANEOUS

<b>SPELLBINDER</b> w/dictionary	\$325
H89 custom SB keyset	\$ 37
<b>THE WORD PLUS</b> spell checker	\$125
<b>WORD PROCESSOR</b> (Bible study)	\$179
<b>dBASE II</b> CP/M and ZDOS	\$389
<b>FRIDAY!</b> database program	\$195
<b>QSALE</b> inventory/POS dBII	\$195
<b>MAGNOLIA 77316</b> DD board	\$369
<b>MMS H89 16K</b> upgrade RAM	\$ 69
<b>DAISYWRITER 2000</b> PRINTER	\$1149
<b>ISOBAR</b> 8 outlet protector	\$ 79
<b>425W UPS</b>	\$429
<b>RS-232 TRANSFER SWITCH</b>	\$ 69
<b>ANGEL 64K PRINT BUFFER</b>	\$269
*MEMORY CHIPS *FANS *RS-232 LINE	
MONITORS *VIDEO BLANKERS *KRES 2/4	
MHZ MOD *BOOKS *MORE	

**CALL OR WRITE FOR FREE CATALOG, OR FOR MORE INFORMATION ON ANY OF OUR PRODUCTS**

## H-SCOOP

The most complete independent newsletter supporting the H8, H89 and Z100 series of computers. GET THE SCOOP today, and get the most from your computer tomorrow.

RATES: \$20 for 12 issue year USA, \$27 foreign AIR MAIL

**ZENITH** | data systems

**QUIKDATA, INC.**

2618 Penn Circle  
Sheboygan, WI 53081  
(414) 452-4172

   
VISA and MASTERCARD





## Challenges of the Future! Memories of the Past....

The other day I had the opportunity of attending the local high school graduation. Sitting with hundreds of other parents and relatives, my wife and I proudly watched as the second of our four sons went through these ceremonies.

As representatives of the class gave speeches and each member was called to receive his or her diploma, I thought of how we, as the first legion of microcomputer users, will help shape their future. It was easier in the 'ole days. After graduating from high school our decisions concerning employment careers were simpler. It was find a job, go into the military or go to college. What subjects do you study in college? Well, for a teacher, engineer, etc., the classes were well defined.

Today we tell these students, "You have to learn about computers no matter what you want in life!" From the auto mechanic to the housewife to the soldier, from the engineer to the teacher, you have to know about computers. The classes a student entering college today might choose could well be outmoded before that same student graduates four years later.

As members of today's microcomputer society we are helping to set the direction this next generation will take. The choices we make as to operating systems, languages and programs will influence the individuals who will produce these products. Present day marketing trends will affect the outcome of R&D projects and corporate decisions concerning future products.

The hobby of yesterday is giving way to the industry of the future. We as microcomputer hobbyist have had a lot to do with the direction of that future.

As to the future of HUG, I would like to take the opportunity to welcome two new members to our staff. Jim "Blackmax" Buszkiewicz has come to HUG from the Heath Technical Consultation Department. Jim has extensive experience in both hardware and software plus a special interest in CP/M. He has assumed the duties of Software Developer and the HUG Bulletin Board on Compuserve.

Greg Martin is the second new individual at HUG. Greg has past experience in the graphic arts and has taken up the duties of Layout Assistant for REMark.

All of us welcome these two new additions to the staff and hope they will enjoy working with our ever growing membership.

Walt Gillespie  
Editor

ZENITH

NOTE: RETAIL STORE  
FOR LOCAL CUSTOMERS

CELESTIAL  
MERIDIAN

COMPETITIVE PRICES

HORIZON

# Studio Computers Inc.

SPECIALIZING IN  
HEATH/ZENITH  
COMPUTERS AND  
ACCESSORIES

WIDE SELECTION  
OF HARDWARE  
AND SOFTWARE

Ray -  
Great  
Concept!  
Nick

## Engineered to Bring You the Best in Price and Performance

Since our conception in 1978, our reputation has been built on supplying Heath/Zenith-users quality software and hardware products at affordable prices. We offer you the complete selection of Zenith data systems as well as a variety of other select computer peripherals for your Z89/90 and Z100 computers. Don't miss out on our latest free catalog. Call or write today.

999 S. Adams, Birmingham, MI 48011 • (313) 645-5365

**ZENITH** data  
systems

# BUGGIN' HUG



## Update to the "ERRORS" Program

Dear HUG,

In the July 1982 issue of REMark (#30), Patrick Swayne had a program called ERRORS that would give the number of soft errors since a cold boot. I used that program alot, but since then have moved to CP/M 2.2.04. The program will not work with this new version because the addresses of the error counts are different. The following are the changes that can be made to the program to allow it to be used for versions 2.2.02 - 2.2.04.

First find the code 14 lines after MAIN as follows:

```
LXI    D,49H-3 ; OFFSET TO H17 ERROR COUNT (.03)
CPI    3        ; VERSION .03?
JZ     VERS3   ; YES
LXI    D,4CH   ; OFFSET TO H17 ERROR COUNT (.02)
VERS3: DAD     D
```

and change to the following:

```
LXI    D,49H-3 ; OFFSET TO H17 ERROR COUNT (.03)
CPI    3        ; VERSION .03?
JZ     VERS3   ; YES
LXI    D,4CH-3 ; OFFSET TO H17 ERROR COUNT (.02)
JM     VERS3   ; IF VERSION .02 BRANCH
LXI    D,4DH-3 ; OFFSET TO H17 ERROR COUNT (.04)
VERS3: DAD     D
```

Next find the code 6 lines after PHERR as follows:

```
LDA    VERS    ; GET VERSION NUMBER
CPI    3        ; VERSION .03?
JNZ    EXIT    ; IF NOT, FINISHED
LHLD   BASE+1  ; ELSE, GET BIOS ADDRESS
LXI    D,4BH-3 ; OFFSET TO H37 ERROR COUNT
DAD    D
```

and change to the following:

```
LDA    VERS    ; GET VERSION NUMBER
CPI    2        ; VERSION .02?
JZ     EXIT    ; YES, FINISHED
LHLD   BASE+1  ; GET BIOS ADDRESS
LXI    D,4BH-3 ; OFFSET TO H37 ERROR COUNT (.03)
CPI    3        ; VERSION .03?
JZ     VERS3A  ; YES BRANCH
LXI    D,4FH-3 ; OFFSET TO H37 ERROR COUNT (.04)
VERS3A: DAD     D
```

I hope this is helpful to others that use this program.

Christopher S. Simmons  
2481 NW Garryanna Street #35  
Corvallis, OR 97330

## Printer Problems

Dear HUG,

I have a slight problem that is really a pain. I have an H-89 (48K, single drive), Peachtree's Magic Wand, and a JUKI model 6100 daisywheel printer. Unfortunately for myself and other JUKI owners, if there are indeed any of you out there, the JUKI's command set is

similar to a Diablo 630 (no problem yet), but is designed mainly for WordStar. The Magic Wand print program has no command for "Shadow" print, which I feel is necessary to give proper emphasis to titles on such things as book reports, reports, project explanation headings, etc. If there is anyone out there who has surmounted this obstacle, please let me know.

If there is anyone out there with configuration problems, (layperson terms: you can't get the printer to work), below I have include how I got the printer to work with CP/M and Magic Wand.

## Configuring a JUKI Model 6100 to Magic Wand

I. Be sure that the JUKI has an RS232C interface card, this won't work without it. Connect the cable to port 340-347.

II. PIP a copy of CONFIGUR.COM onto the disk containing EDIT.COM.

III. Run CONFIGUR and do the following:

- A. Press A (Terminal Printer characteristics), make sure that:
  1. C = LST: BAUD RATE 300 PORT: 0E0H = 340
  2. M = Printer Ready Signal Polarity = HIGH
  3. N = Printer Ready Signal = DTR (PIN 20)
4. Press Y
- B. Press C (Change Default I/O Configuration), make sure that:
  1. D = LST: = LPT:
  2. Press Y
- C. Press Y (Configure Both Memory and Disk)

### Note:

If you ever again run CONFIGUR on that disk, without planning to change something, you must tell CONFIGUR you do not have a "Standard System", or it will change back to original configuration. To prevent this, you can tell CONFIGUR you don't have a standard system and then go to step C only. This will prevent changing of your current configuration.

IV. At this time your printer will (more than likely) work with Magic Wand.

V. To use the PRINT.COM program, perform steps I - IV on a disk containing it.

Richard R. Jefferson  
P. O. Box 707  
Beaufort, NC 28516

## HA-8-3 & HA-89-3 Graphics Users' Library Now Available

Dear HUG,

At long last the graphics users' library is organized and available. This library is only for public domain programs so you will not find any "For Sale" stuff in it and we hope future donations will also be 100% public domain. At present there are ten HDOS and eight CP/M disks in the library. I would like to thank Jim Buszkiewicz who is the largest single contributor and is responsible for most of what is in the CP/M disks. Thanks also goes to Skip Kelly, 1012 Danburry Circle, Virginia Beach, VA 23464, who took the time to organize the disks and will be maintaining and distributing the disks. To reduce the amount of disks being messed up in customs, in Canada contact Tom Few, RR #1, Arva Ont NOM 1C0, If you have anything to add to the library, please send it to Skip Kelly. I believe they will be asking \$2 to cover postage and handling of your disks. Contact them for further details. Best to everyone.

Fred Pospeschil  
3108 Jackson St.  
Bellevue, NE 68005

## Extra Z-100 Opcodes

Dear HUG,

I recently came across some information which was new to me and may be new to many people. It seems that the 10 unassigned opcodes in many 8085 microprocessor chips may actually perform operations. According to my information they are as follows:

08 (HL) => (HL)-(BC)  
10 Arithmetic right shift of (HL)  
18 Rotate (DE) left through Carry  
28 Load DE with HL+Immediate (8 bits)  
38 Load DE with SP+Immediate (8 bits)  
CB Restart at 40H on overflow  
D9 Store (HL) indirect through DE  
ED Load (HL) indirect through DE  
DD Jump on NOT bit 5 of flag byte  
FD Jump on bit 5 of flag byte

I have tested all of the opcodes on my Z-100 except CB since I don't have the definition of "overflow". Opcode 08 affects the sign, zero, and carry flags but not the auxiliary carry or parity. Opcode 10 is interesting for programmers using 2's complement arithmetic since the "sign" for bit (H7) propagates right during the shifting. Opcodes DD and FD are interesting in that the only way I can find to control F5 is to push a word onto the stack and then do a POP PSW.

I would be interested in reading about other peoples' knowledge of, or experience with, these opcodes.

James T. Malone  
461 Shore Acres Road  
Arnold, MD 21012

## Additions to "ZD", Z-100 Sorted Directory Utility

Dear HUG,

I have recently amended two small sections of code to Jeff Kalis' Z-100 sorted directory utility, ZD, which was discussed in the April issue of REMark. The first addition supplies the number of files on the disk using the same format as the Z-DOS DIR command. To allow the hard copy of the directory to be torn from my ProWriter without performing manual line feeds, the second section of code was added which provides the necessary paper advancement.

The additions are simple to understand and can be quickly inserted into the program. I hope others may find these features useful.

### Listing 1. Definition of the number of files field.

```
MONTH DB ' /'  
DAY DB ' /'  
YEAR DB '<<<<',0  
NUMFLS DB ' ,6 DUP (0),' file(s)',0 ;number of  
; files field  
;  
X10000 DW 10000  
X1000 DW 1000,100,10  
STKCNT DW 0  
;  
;  
START PROC NEAR
```

### Listing 2. The modified code which provides the number of files and the hard copy paper advancement.

```
POP DX  
NEXTLP: ADD SI,2 ;Point to next pointer  
LOOP 0TLP  
;  
;
```

```
; Custom section 1 written by Michael Masters on May 24,  
; 1984 to print the number of files on the disk.
```

```
MOV AX,STKCNT ;Load AX with the  
; number of files  
MOV DX,0  
MOV DI, OFFSET NUMFLS+3 ;Point to the  
; number of files.  
CALL CONVRT ;Convert the number of  
; files to ASCII using  
; the existing CONVRT  
; routine.  
;  
; Print out the number of files  
MOV DX,OFFSET NUMFLS  
CALL DOPRT
```

```
; End of the custom routine 1 written by Michael Masters.
```

```
CMP OUTDV,2 ;if out to CON, we're done  
JZ PRTOUT
```

```
; Custom routine number 2 written by Mike Masters on  
; May 25, 1984 to perform several line feeds on the  
; printer in order to bring the directory out of the  
; printer.
```

```
CMP FFPLG,OFFH ;Check to see if the  
; form feed flag is set  
JZ FRMPD ;If so go do a form feed  
; If not print 9 line feeds.  
MOV CX,9AH  
LFL00P: CALL DOCRLF  
LOOP LFL00P  
RET ;We are done, so return.  
;  
FRMPD: MOV DX,OFFSET RESTR ;Perform a form feed.  
; End of custom section number 2  
;  
CALL DOPRT ;print the form feed
```

```
PRTOUT: RET  
PRINT ENDP
```

Michael Masters  
Graduate Teaching Asst.  
Dept. of Electrical Engineering  
Kansas State University  
Manhattan, KS 66506

## A Letter To David Warnick, Author of the BASIC Computing Series

Dear Mr. Warnick,

I have been following your series of articles on BASIC Computing in REMark and was especially pleased to see the segments on random files and sorting. I have never used random files before as the programs I have written lend themselves more toward sequential files. In trying your programs, I noted one apparent discrepancy between the version of MBASIC you use and the one I have (BASIC-80, Rev. 4.82, running under HDOS on an H8). The discrepancy was that I cannot specify the field length in the OPEN statement. Therefore, the following code produced a file 100 sectors long:

```
10 OPEN "R",#1,"FILENAME"  
20 FIELD #1,5 AS A$  
30 FOR X=1 TO 100  
40 LSET A$=MKIS(X)  
50 PUT #1,X  
60 NEXT X  
70 CLOSE #1  
80 END
```

The solution to conserving disk space on my system is only partially documented in the MBASIC manual. However, I devised the follow-

Vectored to 64





# "My Favorite Subroutines"

Dear HUG,

I have a comment on Mr. John Hafey's subroutine in the June issue, page 24. He suggests using Poke 3,170 to output everything on the printer and Poke 3,105 to output everything on the terminal. These work fine but if you encounter an input statement or an error, your system will lock-up because nothing can be entered from the keyboard.

I suggest using Poke 3,171 for outputting to the printer and Poke 3,107 for outputting to the terminal. This way you can still input data and if an error should occur, you can save yourself by putting things back the way they were by typing Poke 3,169.

I also have one of my own MBASIC subroutines. This routine easily converts lower to UPPER CASE and verifies that an alphabet was indeed entered. First let me explain how it is done. The ASCII value of "A" is 41 Hex or 0100 0001 Binary and "a" is 61 H or 0110 0001 B. If it's not immediately obvious, the only difference is Bit 5. When this Bit equals 0 it is UPPER CASE, when it equals 1 it is lower case. In this situation we want capitals so all we have to do is use the boolean function AND 1101 1111 which in Hex is DF. This will cause bit 5 to go low and all others will remain unchanged.

```
10 PRINT "ENTER (A-Z): ";
20 X$=INKEY$:IF X$="" THEN 20
30 PRINT X$:* ECHO CHARACTER *
40 X1$=CHR$(ASC(X$) AND &HDF):* MAKE X1$ CAPITAL *
50 IF X1$<"A" OR X1$>"Z" THEN 10:
  '* VERIFY THAT ITS AN ALPHABET *
60 X$=X1$:* MAKE X$ CAPITAL IF DESIRED *
```

Frank Cepulkowski  
4523 S. Richmond  
Chicago, IL 60632

Dear HUG,

I prefer to load ZBASIC using the expression 'ZBASIC Z' where the 'Z' is a program to set up my favorite working conditions and especially my favorite FUNCTION KEY assignments which change with time by the natural laws of "whim". Many of my programs alter these precious assignments and I demand that they be restored by the programs as I exit from them. Because of "whim", it is not practical to hardwire them into all the exit routines. And because the disk with the 'Z' program may have been removed for the current program, it could be awkward to call. All I want anyway, is to have the F-keys restored.

This is a pair of subroutines to read the current assignments from memory before the program changes them to restore them later. Very convenient!

Two cautions: Because of absolute memory addressing, you may need to alter the variable named 'base' for your versions of Z- DOS and ZBASIC. Also, the key assignments are preserved as a string array so some caution with ERASE is required.

Naturally, you must give due attention to the state of the keys before allowing the program to loop through either routine a second time.

```
5 '(joykeys.bas) ZBASIC ASCII file by Arthur C. Smith
6 DIM KEYZ$(12):DEF SEG=&H1000:BASE=7116
7 FOR K=1 TO 12:FOR N=BASE+16*K TO BASE+16*K+15
8 A=PEEK(N):KEYZ$(K)=KEYZ$(K)+CHR$(A)
9 NEXT N:NEXT K:DEF SEG:<goto or return as you require>
```

```
65500 'restore original function keys as you exit a
      program
65501 KEY OFF:FOR K=1 TO 12:KEY K,KEYZ$(K):NEXT K
```

Arthur C. Smith  
11685 Shangri La Avenue  
Hesperia, CA 92345

Dear HUG,

The following is a short routine that allows your Pascal program to echo asterisks while you enter information into the computer. The variable "typeletter" holds the temporary letter and, if it is not a backspace (CHR(8)), it will CONCATenate it to the String Variable, response. The routine is designed to print the string that you enter at the end of the routine.

This would be a good routine to use for password entry into a bulletin board system or other programs where privacy is required.

```
*)
program hide_characters;
var
  typeletter : char;
  response   : string[80];
begin
  response := '';
  write ('please enter your name...');
  repeat
    read (kbd,typeletter);
    if typeletter = #8 then
      begin
        delete (response,length (response),1);
        write (#8,' ',#8);
        end
      else
        begin
          write ('*');
          response := concat (response,typeletter);
          end;
    until typeletter = #13;
  writeln (#13,#10,'Your entry is ',response);
end.
```

Paul L. Eustace  
Vice President  
North Houston HUG  
8110 Tattershall Circle  
Humble, TX 77338

Dear HUG,

I found the new feature in the April 1984 issue of REMark, "My Favorite Subroutines", to be very useful. I would like to share two of mine which I have found very useful. They are not original with me. I read the algorithms somewhere, but so long ago I've forgotten.

Western human calendars are based on the system established by Pope Gregory XIII in 1582 (sic) the Gregorian Calendar. It is the calendar we all know and love with its twelve months for the twelve Apostles. Some months have 30 days, some have 31, and one has 28 days -- except every four years when it has 29 days. That means some years have 365 days and some 366 days. Despite all the jiggling and tweaking, it still doesn't work out evenly because the earth wobbles in its orbit. But it doesn't matter because humans are very adaptable, and can crunch data regardless of format. Rapid computations based on dates in the Gregorian system, however, are a lot to expect from a little eight bit computer.

Fortunately, Julius Caesar, who was in charge in Rome before Pope Gregory, well before, had a better, if more egotistical idea. He based his calendar on the beginning date of his reign, and dates were simply numbered from that point. Of course, Julius was a heathen, and all his works were suspect.

Nevertheless, the idea of starting a count of dates from a fixed position stuck around. The only problem is; for large expanses of time in a human sense the Julian Date gets too large and meaningless. What day is 2349832 in the infinite scheme of things?

Computers handle Gregorian Calendar dates awkwardly whether the input is September 12, 1984, 09/12/84, or something else. Julian Calendar dates, however, are easily processed by the computer because the Julian Date is represented by a single value; in the case of the BASIC subroutines below a single precision number called DJ!. DJ! can be added to, and subtracted from in amounts of days, weeks, months (average length), years, or centuries. It can also tell you the day of the week for any date in its range (1900 to 2100 AD). If you want to know the day of the week your birthday will fall on in 2001 AD, just ask. If you are still around in 2200 AD, you will have to adjust the constants to get accurate day/date computations.

#### Suggested Uses:

The Gregorian Date can be input in any format convenient for the user. I find the MMDDYYYY convenient because it allows me to enter dates very quickly via the keypad. I input the date into a string which is then parsed into MM, DD, and YYYY sections with MID\$. The sections are converted to numerical values for M%, D%, and Y% using VAL.

My program will then GOSUB to the Gregorian to Julian Converter. On return from the subroutine, DJ! can be renamed so that several Julian dates can be manipulated as needed.

When it is time to print the results of the program, each specific Julian Date is converted to DJ! and sent to the Julian to Gregorian Converter. On return from the subroutine M% contains the number of the month, D% contains the date of the month, Y% is the year AD, and DOW\$ is the number of the day of the week. The names of the months and days for the week are read from tables equivalent to the values of M% and DOW\$, and printed in whatever format desired.

These subroutines have many possibilities. If you are working with a payroll program where special rates of pay for Sunday and Holidays

are involved, or where it is necessary to know when the end of the work week is, etc., the Julian subroutines are the ones for you. It is also helpful for work planning programs, i.e. no one works on weekends.

#### Variables:

M% = The number of the Gregorian month (1 to 12).

D% = The number of the Gregorian date (1 to 31).

Y% = The number of the Gregorian year (1900 to 2100).

DJ! = The Julian Date.

DN! = An intermediate variable.

DOW% = The day of the week represented as a value from 0 to 6.

Zero = Sunday.

```
0400 ' Convert Gregorian to
      Julian Date
```

```
0402 IF M%=1 OR M%=2 THEN Y%=Y%-1:M%=M%+13:GOTO 420
```

```
0410 M%=M%+1
```

```
0420 DJ!=INT(365.25*Y%)+INT(30.6001*M%)+D%+1720982#
```

```
0500 ' Convert Julian to
      Gregorian Date
```

```
0505 DN!=DJ!-1720982#:Y%=INT((DN!-122.1)/365.25):
```

```
M%=INT((DN!-INT(365.25*Y%))/30.6001)
```

```
0510 D%=DN!-INT(365.25*Y%)-INT(30.6001*M%):
```

```
IF M%=14 OR M%=15 THEN M%=M%-13:GOTO 520
```

```
0515 IF M%<14 THEN M%=M%-1
```

```
0520 IF M%=1 OR M%=2 THEN Y%=Y%+1
```

```
0525 DOW%=7*((DN!+5)/7)-INT(((DN!+5)/7)):RETURN
```

Carl Edwin Lovett Jr.

American Consulate General FRDCO

APO New York, NY 09213

Dear HUG,

Here is a small SCREEN DUMP program of mine that avoids "The Dreaded HT Bug" which appeared on page 66 of the April 1984 issue of REMark in a letter from L. T. Scotney. After reading the article "ZBASIC Simulation Technique" by Arthur C. Smith in the March 1984 issue of REMark, I tried some Curve plotting ZBASIC programs of my own. It wasn't long before I desired hard copy, but discovered no Utility for such in Z-DOS. This SCREEN DUMP program fits the bill, if you can stand the 30 minute wait.

```
10000 REM ZDUMP.BAS J.FENNEN 4-6-84
10010 REM DUMPS Z100 VIDEO DISPLAY TO RX-80 PRINTER IN
10020 REM DOT IMAGE MODE 8X640 DATA BLOCKS, OVERLAID
10030 REM SO FIRST PIXAL ROW OF DATA BLOCK OVERLAYS
10040 REM LAST PIXAL ROW OF PRECEDING DATA BLOCK
PRINTED. THE 8TH PIXAL ROW OF EACH DATA DE-
10060 REM CODE FROM VIDEO SCREEN IS ALWAYS EQUAL TO
10063 REM ZERO TO PREVENT THE INFAMOUS BASIC "HT" BUG
10065 REM FROM CORRUPTING THE DOT IMAGE MODE DATA BLOCK.
10070 LPRINT CHR$(10);CHR$(10);CHR$(27);"A"CHR$(7);
10075 Y=0
10080 FOR T=1 TO 31
10090 LPRINT CHR$(27);" *";CHR$(4);CHR$(128);CHR$(2);
10100 FOR X=0 TO 639:W=128:U=0
10110 FOR Z=0 TO 6
10120 IF POINT(X,Y+Z)<>0 THEN U=U+W
10130 W=W/2:NEXT Z:LPRINT CHR$(U);:NEXT X:LPRINT
10140 Y=Y+7:NEXT T
10150 LPRINT CHR$(27);"@"::FOR I=1 TO 25:PRINT CHR$(7);:
LPRINT CHR$(7);:NEXT I
10160 RETURN
```

J. Fennen

5593 Forest Lane

Fort Loudon, PA 17224



# Heath Company Introduces HERO JR.

Bob Ellerton  
HUG Manager

## The First Fully Pre-programmed Personal Robot

Sunday, June 3, 1984, marked the introduction of HERO JR., Heath's fully pre-programmed personal robot. HERO JR. was introduced to the public at the Consumer Electronics Show in Chicago. I was fortunate enough to attend and be a part of the team that demonstrated this unique robot to the world.

Unlike previous entries into the personal robot market, HERO JR. uses a 32K monitor ROM to provide an interesting almost kidlike out-of-the-box personality. HERO JR. is capable of playing games, singing, reciting poetry, gabbing (robot gibberish), speaking and exploring on its own with these user selectable personality traits. When HERO JR. enters into its "personality mode", it can perform true multi-tasking routines. But, probably one of the best features of HERO JR. is that it easy to use. HERO I required extensive user knowledge to program through the HEX keypad on the head. HERO JR., on the other hand, can be programmed easily by a child with simple to use commands clearly labeled on its keyboard. The keyboard on HERO JR.'s head even has a help and demo key to further explain the operations of this friendly little robot.

HERO JR. uses four sensors to detect sound, light, motion and distance. With these sensors, HERO JR. can explore its environment or use these sensors as a security system, alarm clock or measuring tool. HERO JR. is equipped with a remote control so that the user can "drive" the robot from one location to another directly. To say the least, HERO JR. is a combination of unique well thought out technology. Also, HERO JR. can be expanded with plug-in modules that are used to enhance either its utility or personality.

As you have probably noticed, I have described HERO JR. as an "it" previously in this article since this is the way HERO would seem until the user becomes familiar with "Robot Wizard". Robot Wizard is a mode that allows the owner to change or modify some of HERO JR.'s programming such as its name. Therefore, HERO JR. can assume his or her own identity. Further, HERO JR. can be modified to speak the user's name in many of its phrases providing for even more personality.

HERO JR. has an RS-232 interface located on the rear of the robot's head. The interface will allow direct connection between a terminal or computer. With this interface and the "HERO JR. BASIC" cartridge, HERO JR. can easily be programmed for functions not standard to the monitor ROM.

Further information about HERO JR. can be obtained by contacting Heath Company, Department 150-375, Hilltop Road, Benton Harbor, Michigan. HERO JR. will cost approximately \$1000. ✖



While attending the Consumer Electronics Show, Bob and HERO JR. met Leonard Nimoy, better known to most HUGgies as "Spock" in the T.V. and movie series Star Trek. Mr. Nimoy was kind enough to sign the actual demonstration HERO JR. after putting the robot through its paces.

# WIZARDRY NEWS

The Best Software for the Best Computers!

## REACTOR-100

### Nuclear power at your fingertips!

You are at the controls of a high capacity pressurized light water nuclear reactor in this exciting simulation. **REACTOR-100** makes full use of the outstanding color graphics capability of the Z-100. A "real-time" schematic display of the plant and its conditions, as well as a full instrument panel, give you complete control to safely (or not so safely!) operate the reactor.

The program can optionally be used with the Software Wizardry P-SST multifunction board for realistic sound effects. Plugging in the **Wizardry Lightpen** allows complete control simply by pointing at the screen.

Not just a game, this superbly crafted system is also an outstanding educational tool for power engineers, and those simply interested in "the way things work".

**REACTOR-100** runs under ZDOS on the H/Z-100. Also available for the Z-150/160, but schematic does not display in color.

## Palette Version 2.0

### Now with animation

The best gets better! **Palette**, the best selling interactive graphics program for the H/Z-100, lets you draw superb color or monochrome graphics displays. Save them to disk with a unique compression algorithm to save disk space, or dump them to compatible printers, black and white or color.

Other features include support for the **Wizardry Lightpen**, interface for joystick control with the **Wizardry P-SST board**, and support of the high resolution interlace mode of the H/Z-100.

Now, with the newest release, we've added **animation**, and a host of other improvements. If you've always felt you had a little Michaelangelo or Cecil B. DeMille in your blood, now's your chance to express yourself! Create masterpieces. Store them on disk. Manipulate them, animate

them! **Palette** comes with extensive documentation and a complete tutorial.

For those of you already sold on **Palette**, \$20 and return of your distribution disk will get you Version 2.0

**Palette** runs under ZDOS on the H/Z100 and requires color RAM and at least 192K system memory.

### ESP Dial in Communications

#### New Product!

**ESP**, our newest communication software package for Heath/Zenith computers, is a bulletin board system with a difference — it creates **two system consoles**. Anything typed on one will appear on the other, and be read by the system. The second terminal can be a station in another room, or a modem for dial-in capability!

**ESP** runs under CP/M-80 on the H-89 & 90, and under CP/M-85 on the H/Z100

## ZLYNK/II Modem Communications

### New version available

**ZLYNK/II**, THE communications software for the Z-100, even features it's own language — **ZLINGO!** New version now available features **CP/M-86 compatibility!** In addition, the upgrade includes Compuserve "B" protocol, improvements to existing overlays, more configurable options, more **ZLINGO** commands, and user-defined function keys.

The documentation has been revised, and an index included. Now that we've made this powerful system even better, you've no excuse not to talk to the world!

**ZLYNK/II** runs under CP/M-85 or CP/M-86 on the H/Z-100.





# Inside Microsoft™ BASIC (MBASIC)



D.D. Dodgen  
3 Porter Ave.  
Buffalo, NY 14201

Written for the H/Z-89 computer using MBASIC under the HDOS operating system. Most items of information are pertinent to all versions of MBASIC, but the codes used change with each version.

Knowledge of the internal workings of our computers usually turns out to be useful sooner or later. The more involved our programs, the more it becomes important to be familiar with each detail of how the computer operates.

The purpose of this article is to discuss how MBASIC stores its programs in the computer's Random Access Memory (RAM), and how this will affect us as programmers. After I'm done with all of the technical information, I promise to provide a practical example of how all of this can be used to an advantage.

Just to keep your interest up for now, though, I will mention that one of the most obvious advantages to knowing how your computer stores programs is that you can then know what does and does not save memory when you are trying to fit your latest programming effort into the space that HDOS and MBASIC conspire to leave you.

Use Program Listing 1 to look into the area of RAM where MBASIC begins its storage of programs. The first thing that you will notice is that the program is not stored in the same readable, listable format that it was in when you first typed it. This is because MBASIC encodes part of the program in order to save space. The amount of space that the authors of MBASIC save you through this technique can be graphically demonstrated with a simple experiment: On a disk with some spare room left on it and an MBASIC program already saved on it, BOOT the disk and LOAD the program into memory under MBASIC. Now enter this: SAVE"TEMP",A. By using a different file name, we ensure that the original program is not destroyed. The [,A] part of the command tells MBASIC to decode the program back to its listable format while saving it to disk. Use the SYSTEM command to get back to HDOS, and then use the CAT command to look at the disk's directory. Notice how many more sectors the program takes up in its decoded format? It should be about 25-33% depending on the amount of PRINT and REM statements used. Be sure to DELETE TEMP.BAS when you are done with this little experiment.

Now that we know that MBASIC is using some form of code, let's see if we can crack it!

Looking back into the computer's memory again (Example 1 is a copy of what Program Listing 1 will display), we find that MBASIC uses the following technique to store each program line:

2 bytes = a pointer to the beginning of the next program line in memory

2 bytes = the line number of this line

? bytes = variable number of bytes to store whatever is on this line

1 byte = set to zero as a line termination character

**Note:** The two byte pointer to next line and the two byte line number are both encoded in the LSB/MSB format used by assembly language programmers. This is because MBASIC is just another assembly language program itself. To decode LSB/MSB back into decimal format, multiply the second byte by 256 and add the result to the first byte. (Example: In our memory dump, address 28761 contains 104 and address 28762 contains 112 as pointers to the next line. Multiply  $112 * 256 = 28672$ . Now add 104, which was the first byte,  $28672 + 104 = 28776$ . This indicates that the next line will start at address 28776. If you look at 28775, you will see the zero byte terminating the line before it.

This little tidbit of information gives us our first, very important clue to saving memory: USE MULTIPLE STATEMENT LINES WHENEVER POSSIBLE. Each line of your program takes up five extra bytes of memory over and above what the line contains. It only takes one byte to use the colon (:) required for a multiple statement line. Take Program Listing 1 for example. Eight bytes could have been saved by putting the whole program on one line, versus three. Eight bytes may not sound like much, but then this program was only three lines long. Think of the savings for a 16K program! Of course, if your program is definitely going to be a short one, then putting each statement on its own line does increase readability of the code. Everything is a trade-off.

The next step in our search for the key to the code is to notice that MBASIC encodes each RESERVED KEYWORD (a listing of keywords appears on page 7-14 of the MBASIC Manual) such as FOR, PRINT, NEXT, etc., into a one or two byte TOKEN. (See Tables 1 and 2 for a listing of these Keyword Tokens.)

Looking once more at Example 1, you should now be able to see that addresses 28761 and 28762 point to memory location 28776 as the beginning of the next line of the program, addresses 28763 and 28764 indicate that this is line number 10. Next you will discover that address 28765 contains the value 130, which Table 2 indicates is the token for the keyword FOR. After that, you will notice the variable 'I' is stored without being encoded. (No, I have not forgotten my promise to make this all useful. Yes, it has something to do with this 'I' business.) Then there should be an equal sign, but you will find that it has been changed to the Mathematical Token 240. (See Table 3 for the various Mathematical Tokens.)

Now we find what at first glance appears to be a strange garble of information. We should find following the equal sign,

[28761TO28808], but what we do find does not seem to mean that at all (except for you sharp eyed programmers who picked the token for the keyword 'TO' out of the list in Table 2, go to the head of the class). If you spotted that, you may also have noticed further that there are three bytes on either side of that token, and that the first of each of these three byte groups is the value 28. You have probably guessed by now, 28 is another form of token, and the other two bytes are encoded numbers just like the line number and the next-line pointer. (See Table 4 for a listing of the various Constants Tokens.)

Table 4 points out another interesting tidbit of information concerning memory savings. Look at the following program bit:

```
10 PRINT "Hello"
...
1000 GOTO 10
...
2000 GOTO 10000
...
10000 PRINT "Goodbye"
```

Which line do you think takes up more memory, 1000 or 2000? Those of you who chose 2000 go to the back of the class. Those of you who chose line 1000 go to the back with the others. Table 4 shows that a reference to another line of the MBASIC program is stored as: a one byte token (14) followed by the line number encoded into two byte LSB/MSB format. It does not matter if the line number referenced is 1 or 65000, it still only takes up three bytes of memory.

Therefore, putting frequently used sub-routines near the beginning of the program to save memory because they have shorter line numbers, does not work on the H/Z-89 version of MBASIC as it does in some other versions.

This tactic does work well in the other major programming concern -- saving execution time. Each time a program statement such as GOTO, GOSUB, etc. sends the MBASIC interpreter out of line number sequence, the interpreter has to start back at the first line of the program and examine each line number in sequence until it finds the one for which it is searching. Therefore, if the line is near the beginning of the program, then it will be found sooner. A few seconds here and there sometimes add up, especially if a subroutine is called from a FOR-NEXT loop.

Another time saver is to define variables used the most often early in the program. MBASIC stores all variables used by the program in a table, and must start at the beginning of the table each time it looks one up. The variables are put into the table in the order in which there are discovered in the program.

### Example 1

```
28761 104 h 28762 112 p 28763 10
          28764 0      28765 130   28766 73 I 28767 240 p
28768 28      28769 89 y 28770 112 p 28771 206 N 28772 28
28773 128     28774 112 p 28775 0      28776 128 28777 112 p
28778 20      28779 0      28780 145   28781 73 I 28782 59 ;
28783 255     28784 151   28785 40 ( 28786 73 I 28787 41 )
28788 59 ;    28789 255   28790 150   28791 40 ( 28792 255
28793 151     28794 40 ( 28795 73 I 28796 41 ) 28797 41 )
28798 44 .    28799 0      28800 135   28801 112 p 28802 30
28803 0      28804 131   28805 73 I 28806 0      28807 0
28808 0
```

**Note:** This is what the video display will show after you type in Program Listing 1 and RUN it. Notice that the last three bytes are all zero. This is the way MBASIC tell itself that it has reached the end of the program.

Table 1 -- MBASIC KEYWORD to Memory Token Conversion

ABS=152 210	ERASE=177 133	LOF=255 176	RETURN=142
AND=247	ELR=214	LOG=255 138	RIGHT\$=255 130
ASC=255 149	ERR=215	LSET=201	RND=255 136
ATN=255 142	ERROR=168	MERGE=255 197	RSET=202
AUTO=171	EXP=255 139	MID\$=255 131	RUN=138
CDBL=255 157	FIELD=192	MKD\$=255 179	SAVE=203
CINT=255 155	FIX=255 158	MKI\$=255 177	SGN=255 132
CSNG=255 156	FN=211	MKSS=255 178	SIN=255 137
CHR\$=255 150	FOR=130	MOD=252	SPACE\$=255 152
CLEAR=146	FRE=255 143	NAME=199	SPC=212
CLOSE=195	GET=193	NEW=148	SQR=255 135
CONT=154	GOSUB=141	NEXT=131	STEP=209
COS=255 140	GOTO=137	NOT=213	STOP=144
CVD=255 172	HEX\$=255 154	NULL=150	STR\$=255 147
CVI=255 170	IF=139	OCT\$=255 153	STRING\$=216
CVS=255 171	IMP=251	ON=149	SWAP=165
DATA=132	INP=255 144	OPEN=191	SYSTEM=189
DEF=152	INPUT=133	OR=248	TAB=208
DEFDBL=176	INPUT\$=133 36	OUT=157	TAN=255 141
DEFFN=152 211	INSTR=218	PEEK=255 151	THEN=207
DEFINT=174	INT=255 133	POKE=153	TO=206
DEFSNG=175	KILL=200	POS=255 145	TROFF=164
DEFSTR=173	LEFT\$=255 129	PRINT=145	TRON=163
DEFUSR=152 210	LEN=255 146	PUT=194	USING=217
DELETE=170	LET=136	READ=135	USR=210
DIM=134	LINE=177	REM=143	VAL=255 148
EDIT=167	LINEINPUT=177 133	RENUM=172	VARPTR=220
ELSE=58 162	LIST=147	RESET=204	WAIT=151
END=129	LOC=255 175	RESTORE=140	WIDTH=161
EOF=255 174	LOAD=196	RESUME=169	XOR=249

### Shorthand substitutes for Keywords:

```
? vice PRINT===145
' vice REM=====58 143 219
```

Notice that using the shorthand PRINT style does not change memory requirements, but the shorthand REMARK uses even more memory than it's counterpart!

Table 2 -- Memory Token to MBASIC KEYWORD Converter

58 143 219=	161=WIDTH	204=RESET	255 138=LOC
58 162=ELSE	163=TRON	206=TO	255 139=EXP
129=END	164=TROFF	207=THEN	255 140=COS

130	FOR	165	SWAP	208	TAB	255	141=TAN
131	NEXT	166	ERASE	209	STEP	255	142=ATN
132	DATA	167	EDIT	210	USR	255	143=FRE
133	INPUT	168	ERROR	211	FN	255	144=INP
133	36=INPUT\$	169	RESUME	212	SPC	255	145=POS
134	DIM	170	DELETE	213	NOT	255	146=LEN
135	READ	171	AUTO	214	ELR	255	147=STR\$
136	LET	172	RENUM	215	ERR	255	148=VAL
137	GOTO	173	DEFSTR	216	STRING\$	255	149=ASC
138	RUN	174	DEFINT	217	USING	255	150=CHR\$
139	IF	175	DEFSNG	218	INSTR	255	151=PEEK
140	RESTORE	176	DEFDBL	220	VARPTR	255	152=SPACES
141	GOSUB	177	LINE	247	AND	255	153=OCT\$
142	RETURN	177	133=LINEINPUT	248	OR	255	154=HEX\$
143	REM	189	SYSTEM	249	XOR	255	155=CINT
144	STOP	191	OPEN	250	EQV !?!	255	156=CSNG
145	PRINT	192	FIELD	251	IMP	255	157=CDBL
146	CLEAR	193	GET	252	MOD	255	158=FIX
147	LIST	194	PUT	255	129=LEFT\$	255	170=CVI
148	NEW	195	CLOSE	255	130=RIGHT\$	255	171=CVS
149	ON	196	LOAD	255	131=MID\$	255	172=CVD
150	NULL	197	MERGE	255	132=SGN	255	174=EOF
151	WAIT	199	NAME	255	133=INT	255	175=LOC
152	DEF	200	KILL	255	134=ABS	255	176=LOF
152	211=DEFFN	201	LSSET	255	135=SQR	255	177=MKI\$
153	POKE	202	RSET	255	136=RND	255	178=MK\$
154	CON'T	203	SAVE	255	137=SIN	255	179=MKD\$
157	OUT						

**Note:** EQV is an undocumented KEYWORD! Placed in the list as it is, it appears to be meant to be a logical operator (equivalence?). My guess is that it does not work properly, hence Microsoft did not include it in their Manual.

**Table 3 -- Mathematical Tokens**

>	(greater than)	239
=	(equal to)	240
<	(less than)	241
+	(addition)	242
-	(subtraction)	243
*	(multiplication)	244
/	(division)	245
↑	(exponentiation)	246

**Table 4 -- Constants Tokens**

11	= next two bytes comprise Octal number (0-65535)
12	= next two bytes comprise Hex number (0-65535)
14	= next two bytes comprise line number reference (0-65529)
15	= next one byte comprises decimal number (10-255)
17-26	= equals the numbers 0-9 (i.e. 17=0, 18=1, 19=2, etc.)
28	= next two bytes comprise integer (-32768 to +32767)
29	= next four bytes comprise single precision number (comprised of a sign with seven significant digits and a decimal point) also used for exponential notation
31	= next eight bytes comprise double precision number

**Program Listing 1 -- Display Memory Contents**

```
10 FORI=28761T028808
20 PRINTI;PEEK(I);CHR$(PEEK(I));
30 NEXTI
```

**Program Listing 2 -- Demonstate Speed of For-Next Loop**

```
10 FORI=1T065000
20 REM
30 NEXTI
```

While you are defining your variables, the more that you define as INTEGERS (whole numbers in the range -32768 to +32767 without decimal places), the more memory you will save. Integers take up two bytes less memory to store than FIXED-POINT numbers (numbers using decimal places - MBASIC assumes all numbers to be fixed-point unless told otherwise!)

When you are going to use subscripted variables, always DIM them. I know, you are thinking that MBASIC will allow any variable to be subscripted up to 10 places without a DIM statement, so why bother when you only want 4 places? The answer is that MBASIC will DIM that variable for 10 places which takes up memory in the variable table (each element uses two bytes in the table to point to the address where the value of the variable can be found). Why waste this memory if it is not needed?!

Leaving off the variable after a NEXT statement is legal, and it saves one or two bytes of memory each time, as well as execution time, once again at a minor loss in program readability. A simple experiment demonstrates this fact also. Run Program Listing 2. It takes 3 minutes, 23 seconds. If you delete the 'I' in line 30, the program only takes 2 minutes, 41 seconds. Try changing the REM in line 20 to a PRINT. This causes the program to take 9 minutes, 3 seconds! Try to avoid unnecessary PRINT statements within a FOR-NEXT loop!

Speaking of PRINT statements, anything that you put within quotes is stored in memory exactly as you type it. (This is true of REM statements as well.) By carefully wording PRINT statements and REMARKS, a great deal of memory can be saved. A system which I use with some success is to not put any REM statements in a medium to long program (calm down for just a minute!). What I do is list the program out on my printer and put the remark statements on the hardcopy where it does not use up memory. Remark statements are absolutely necessary in a program of any length, no matter how good a programmer you feel you are. If you don't believe me, try modifying something you programmed six months or more ago.

Finally, for those of you who are still awake, I will keep my promise and give you a useful program which uses this information. Program Listing 3 is a CROSS REFERENCE utility. It will give you each line number that a particular variable or line number is used within the program, or can be used to dump a complete listing of all variables/line numbers used. Every programmer has had the occasion to wonder: Have I used XX yet? or Can I delete line 1012 without messing up a



GOTO or GOSUB to that line? I wrote this routine because I was tired of tracking down all the places where I used a variable when there was a bug connected with it's value. If XX is equal to 127 when I wanted 10, then I would have to look at each time it was used to find out where the typo was.

To use CROSSREF, make sure the target program does not have any line numbers higher than 64999. Then MERGE CROSSREF (you must have saved it with the ,A option). Now type RUN 65000, and answer the questions.

If the target program is a long program and you are using option one, the program may run out of memory. Try adjusting the CLEAR statement and the DIM statement in line 65000. Sometimes, it will be necessary to use options two and three rather than one because there just is not enough memory left.

When typing CROSSREF, leave out all REM statements. The program will not RUN with the statements located as they are. I placed a large number of REMarks in like this to allow you to follow the flow of the program, but of course, when REMarks are found in the middle of multiple statement lines like this, they will cause the program to malfunction!

For those of you who do not wish to take the time to type in CROSSREF, you can get a copy of the Microcomputer Software Library, Rt. 2, Box 328, Carthage, NY 13619. Ask for Utility Disk #1 (it has other utility programs on it too!). The cost is a modest \$5.00 (one dollar per disk copy fee, one dollar for postage and handling, and three dollars for a blank disk). If you send them a disk that has been INITed, you save three dollars.

If you have not yet heard of the MSL, it is a not-for-profit division of MySoft which acts as a collection/distribution point for public domain software. Their fee of one dollar per copy covers their costs, so membership in the Library is currently free (except for the cost of the stamp on your letter!).

If you write to them, be sure to specify which computer system you want the program for. They operate a department for each computer system currently being sold. (Some of the departments are still rather small in inventory since they depend solely on submissions by members to build their collection.)

### Program Listing 3 -- Cross Reference Utility

```

64999 END : REM To prevent Target Program from crashing into CROSSREF during
           possible test RUNs of the Target Program
65000 CLEAR 5000 : REM It is necessary to CLEAR enough string space for V$(VL).
           This value made need to be changed if Out of Memory occurs.
           VL=0 : REM Counter for how many variables/line number references have
           been found during search options 1, 2, or 3.
           MP=28761 : REM The first byte of any MBASIC program.
           ED$=CHR$(27)+"E" : REM Control code to erase the video display.
           DIM V$(99) : REM There is no way of knowing how many variable/line #
           references will be found. This number will have to be adjusted
           as necessary if a SUBSCRIPT OUT OF RANGE or OUT OF MEMORY error
           occurs. If either error occurs, check PRINT FRE(X),FRE(X$).
           If there is too much string space left, reduce the CLEAR amount.
65010 PRINT ED$ ; "Variable/Line # Cross-Reference Utility" : PRINT
           : PRINT "1 - List ALL Variables/Line # References" : PRINT
           : PRINT "2 - List ALL Variable References Only" : PRINT
           : PRINT "3 - List ALL Line # References Only" : PRINT
           : PRINT "4 - Search for ONE Variable Only" : PRINT
           : PRINT "5 - Search for ONE Line # Only" : PRINT
65015 INPUT "Enter the # of your choice: ";CH : REM Enter a # 1-5
           IF CH<1 OR CH>5 THEN 65010 REM Test for legal response.
           ELSE IF CH=3 THEN LV=1 : REM Choices 4 & 5 only look for one item.
           PRINT : INPUT "Enter the Var/Line # to search for: ";V$(1) : REM Use
           V$(1) as the target variable/line number for later comparison.
           V$(1)=V$(1)+"-" : REM Use the dash to separate target from the list of
           line numbers where target found.
           J=1 : REM A counter.
           V$(2)="ZZZ" : REM Used as a flag for the end of the variable list
           during alphabetizing later prior to display.
65030 PRINT : INPUT "Do you want output to go to a printer? ";LPS : REM Answer
           is not important if printer is not hooked up. Answering YES
           will cause an error if LP: is not LOADED prior to RUNning
           CROSSREF.
           LPS=LEFT$(LPS,1) : REM Using only the left character of the answer
           makes it easier to check for a YES reply. Answering Y, YES,
           YEH, YEP would all be correct responses.
           IF LPS="Y" THEN OPEN"O",2,"LP:" : REM Get the printer ready if wanted.
65100 GOSUB 65190 : REM Get pointer to next line of program and line number of
           current line being searched.
           IF LN=64999 THEN 65450 REM End of Target Program reached so go to
           printout routine.
           ELSE GOSUB 65200 : REM Subroutine to decode the line of program one
           byte at a time looking for variables/line # references.
           GOTO 65100 : REM Keep going back for another line of the program until
           the previous check for line number 64999 stops us.
65150 X=INSTR(V$(J),"-") : REM Use to find the separator between the name of
           the variable/line number being looked for the the line number
           references where it has been found.
           X=X-2 : REM Back the pointer up two spaces so the "-" which was added
           to the variable name will not be included in this computation.
           IF AV$=LEFT$(V$(J),X) THEN 65160 : REM IF equal then we have found a
           reference to the variable/line number we were looking for.
           ELSE RETURN : REM If not equal then this is not the one we were looking
           for so we don't need to indicate that it has been found.
65160 IF STR$(LN)=RIGHT$(V$(J),LEN(STR$(LN))) THEN F=1 : REM If equal that means
           that this reference has already been found on this line so we do
           not have to keep indicating that fact. F is used as a flag for
           this purpose.
           RETURN REM This will return us from subroutine after setting flag.
65170 V$(J)=V$(J)+STR$(LN) : REM Tag the line number where reference
           found on the end of the V$(J) which is keeping track of it.
           J=LV : REM Set FOR-NEXT counter to end of loop. No use wasting time
           looking at the rest of the V$(J) since we found the one we were
           looking for.
           F=1 : REM Flag the reference as being found so we don't have to look
           again until the next line number.
           PRINT AV$="" : REM Show the people that the computer is really doing
           something since this program will take some time to complete.
           RETURN
65180 MP=MP+1 : REM Increment counter for memory address to be PEEKed next.
           MM=PEEK(MP) : REM Set MM equal to the contents of memory location MP.
           IF MM=34 THEN RETURN REM This subroutine is looking for the end of
           a pair of quotes (ASCII code 34).
           ELSE 65180 : REM It will keep looking until the quote is found. This
           is because no variable or line number reference can possibly
           be inside a pair of quotes. Note that if there is a SYNTAX

```



```

error in the program and the quotes were never closed, this
cause cause unpredictable results.
65190 NL=PEEK(MP)+256*PEEK(MP+1) : REM Find the location of the next line of
the program which has been stored in LSB/MSB format.
LN=PEEK(MP+2)+256*PEEK(MP+3) : REM Decode the line number of the current
line being searched.
PRINT : PRINT "Working on Line #";LN : REM Let the people know that we
have moved on to the next line of the program in our search.
MP=MP+4 : REM Move the PEEK pointer past the NL & LN pointer to the
first actual byte of the program line.
RETURN
65200 FOR MP = MP TO NL-1 : REM Search the entire line up the zero byte end of
line terminator.
MM=PEEK(MP) : REM Find out what is in the next location of memory.
65210 IF MM=255 OR MM=15 THEN MP=MP+1 : REM Skip over all two-byte tokens.
ELSE IF MM=132 OR MM=143 OR (MM=58 AND PEEK(MP+1)=143) THEN MP=NL-1 :
REM The rest of the line is either a DATA or REM statement, so
no references can be there.
ELSE IF MM=28 OR MM=11 OR MM=12 THEN MP=MP+2 : REM Skip numeric data.
ELSE IF MM=29 THEN MP=MP+4 : REM Skip over single precision number.
ELSE IF MM=31 THEN MP=MP+8 : REM Skip over double precision number.
ELSE IF MM=14 THEN GOSUB 65400 : REM A line number reference has been
found. Go see if we are looking for line numbers and update
V$(J) if we are.
ELSE IF MM=64 AND MM<123 THEN GOSUB 65300 : REM We have an ALPHABETIC
character, which must be part of a variable reference. Go
update data.
65220 NEXT : RETURN
65300 AV$="" : REM Clear out holder for which reference has been found.
65310 AV$=AV$+CHR$(MM) : REM Add current legal variable character to name of
variable found.
65320 MM=PEEK(MP+1) : REM Get the next character from memory to see if it is
part of this variable's name.
IF (MM=47 AND MM<58) OR (MM=64 AND MM<123) OR MM=33 OR (MM=34 AND MM<38)
THEN MP=MP+1 : GOTO 65310 : REM Legal characters for variable.
65330 IF MM=32 THEN MP=MP+1 : GOTO 65320 : REM Skip blanks.
65340 IF MM=40 OR MM=91 OR MM=123 THEN AV$=AV$+CHR$(MM) : REM Legal subscripting
characters. At this point the entire name of the variable found
has been loaded into AV$.
65350 IF CH=3 OR CH=5 THEN RETURN : REM Only looking for line number references
so we don't have to save this variable.
ELSE IF CH=4 THEN J=1 : GOTO 65150 : REM Only looking for one variable
so set match string to V$(1).
65370 F=0 : REM Set found-on-this-line flag to zero at start of routine.
FOR J=1 TO LV : REM Match against all variables found so far.
GOSUB 65150 : REM Subroutine to check for a match AV$ against V$(J)
NEXT : IF F=1 THEN RETURN : REM If flag set then this variable already
found on this line number.
ELSE V$(J)=AV$+"-" : REM We have found a new variable not yet in the
list. At this point J=LV+1
LV=LV+1 : REM Since we found a new one we have to add one to the count
of how many have been found.
GOTO 65170 : REM Add LN to end of V$(J) and go on.
65400 AV$="" : REM Zero out holder for line number found
X=PEEK(MP+1)+256*PEEK(MP+2) : REM Find out which line number has been
found.
MP=MP+2 : REM Bump PEEK pointer past line number found.
AV$=MID$(STR$(X),2) : REM Scrape off the leading blank when changing
X from number to string variable.

```

```

IF CH=2 OR CH=4 THEN RETURN : REM No interested in line numbers.
ELSE IF CH=5 THEN 65150 : REM Only working on one line number.
65370 : REM The rest of the process is the same as for a variable.
65450 PRINT ED$,"Sorting" : REM Start alphabetizing the variables found.
IF CH>3 THEN LV=1 : REM We only have one variable/line number to sort
in options 4 & 5.
GOTO 65520 : REM Printout to video and printer the results.
ELSE SW=LV-1 : REM Set end of sort pointer.
65460 X=SW : REM Set end of FOR-NEXT loop to last switch made.
SW=0 : REM Zero out switch counter so we can see if any switches are
made during sort.
FOR I = 1 TO X : REM Sort all entries up to last entry switched.
IF V(I)>V(I+1) THEN SW=I-1 : REM A switch is needed here.
V(0)=V(I) : V(I)=V(I+1) : V(I+1)=V(0) : REM Switch V(I) & V(I+1)
65470 NEXT : IF SW=0 THEN 65460 : REM If SW>0 then a switch has been made.
We need to keep going through this routine until no switches
have been made. (This is a improved bubble sort.)
65480 FOR I = 1 TO LV : REM Examine all variables found for the last line
number reference. Line numbers require additional sorting
because the bubble sort thinks that 10000 comes before 20.
IF VAL(LEFT$(V$(I),1))>0 THEN SW=I : REM If the left digit is a number
then mark this as a reference to a line number.
65490 NEXT : REM SW will point to the last line number when this FOR-NEXT loop
is done.
65500 X=SW : REM Set FOR-NEXT counter to last line number switched.
SW=0 : REM Set switch counter to zero.
FOR I = 1 TO X : REM Look at all references up to last switch.
X1=INSTR(V$(I),"-") : REM Find divider between referenced line and lines
found in.
X2=INSTR(V$(I+1),"-") : REM Do next one too.
IF VAL(LEFT$(V$(I),X1) > VAL(LEFT$(V$(I+1),X2) THEN SW=I : REM A switch
is needed here.
V$(0)=V$(I) : V$(I)=V$(I+1) : V$(I+1)=V$(0) : REM Make the switch.
65510 NEXT : IF SW=0 THEN 65500 : REM Keep going back until no switches made.
65520 FOR I = 1 TO LV : REM This is the printout routine.
PRINT V$(I) : REM Print to the video.
IF LP$="Y" THEN PRINT#2,V$(I) : REM Print to the printer is selected.
65525 NEXT : CLOSE : END : REM Close LP: before program stops.

```



EXPLORE  
NEW WORLDS  
WITH  
**HUG**  
GAME  
SOFTWARE



# The HDOS Contest Winner

## Getting Started With Assembly Language



Pat Swayne  
HUG Software Engineer

Last month, I presented the winning CP/M program submitted to my "Getting Started With Assembly Language" contest, which was announced in the February issue. This time I will present the HDOS (Heath Disk Operating System) winner.

The contest was to write a program called RENM for renaming files that used the following syntax:

```
RENM dev:OLDNAME TO NEWNAME
```

where dev: is a drive designation (optional), and oldname and newname are file names. I applied the same four tests to the HDOS programs that I applied to the CP/M programs. I will repeat the description of the tests for those who missed last month's article. The first one was to run the program without any arguments.

```
>RENM
```

The program should have detected that there were no arguments and indicated proper usage with a message. I think that all of the HDOS entries passed this one. For the second test, I created a dummy file using an editor (I had used SAVE in CP/M, but HDOS does not have save) called X.XXX, and renamed it with

```
>RENM X.XXX TO X.X
```

In the CP/M versions, many programs failed here, because they did not put the required spaces after the single character extension. HDOS requires nulls instead of spaces, and since HDOS will not accept an improper file name, the programs had to pass this test or not work at all. For the next test, I made a new X.XXX file and entered

```
>RENM X.X TO X.XXX
```

Here, the program should have detected that there was already a file called X.XXX on the disk and not allowed the rename operation. If it didn't, two X.XXX entries were created, which can cause problems. Here is where some programs started failing. The next test was to enter

```
>RENM X.X TO X.*
```

This caused problems in CP/M, but not in HDOS because of its built in protections, so the programs passed this test by not allowing the operation.

After these tests, I still had more than one working program left, so I had to devise another sneakier test to use, to help me decide. The test was suggested by the author of the winning program, because he had discovered it during his own testing. It was to try to have the program rename itself, as in

```
>RENM RENM.ABS TO RENX.ABS
```

Every program but the winner bombed out here, because HDOS will not allow any operations except disk read by a file on itself unless the channel the file was brought in from the disk with (always -1) is first closed. This is something that I had not covered in my assembly language articles, so perhaps it would have been unfair to use this test to select a program that wasn't real good in other areas. However, the winning program, submitted by Stephen Liddle of Pleasant Grove, Utah, was good in all areas, and he even helped with this article on the program, which, if you remember, was one of the criteria for judging a winner. The following paragraphs are his description of the program, which follows this article.

"RENM starts by checking for a command line on the stack. If the command line is not present, RENM prints a standard help message and exits. If the command line is there, RENM finds the beginning of OLDFILE.NAM and saves that address. RENM then looks for a 'TO', with spaces on either side. Then RENM finds NEWFILE.NAM, and .DECODes it into NEWNAM. Then RENM performs a check to see if the user put a device specification on the NEWFILE.NAM. If so, RENM prints a message saying that it is illegal to give a device specification for NEWFILE.NAM. This is followed by the standard help message."

"If all is clear to this point, RENM checks to make sure the command line has all been processed (except for blanks, which we always skip). Then RENM .DECODes the <dev:>OLDFILE.NAM into OLDNAM. At this point, RENM has parsed the whole command line, and is ready to process the parsed information."

".RENAME requires the device specification to be present in the new name string, so RENM puts the device from OLDNAM into the NEWNAME string, followed by the NEWFILE.NAM as entered by the user. The string in NEWNAME is of the form: 'DVn:FILENAME.EXT'. Note the delimiting blank at the end of the string."

"Next RENM checks to see if NEWFILE.NAM already exists on <dev:>. This is done by trying to .OPENR the NEWNAME string. If the open is successful, RENM prints a message indicating that NEWFILE.NAM already exists, and that OLDFILE.NAM was not renamed."

"Finally(!), RENM actually .RENAME's OLDFILE.NAM to NEWFILE.NAM. If this is successful, RENM lets the user know that the rename worked."

The program is laid out in good style, with a definition section, the main body, subroutines, and a data area. Congratulations to Stephen Liddle for producing this fine program.

```

*****
TITLE 'RENAM - HDOS FILE RENAMING UTILITY'
RENAM - HDOS FILE RENAMING UTILITY

THIS PROGRAM DEMONSTRATES FILE
RENAMING IN HDOS.

RENAM RENAMES FILES USING THE FOLLOWING
SYNTAX:

>RENAM <dev:>:OLDFILE.NAM TO NEWFILE.NAM

<dev:> is an optional device specification
which defaults to SYD:.

THE GENERAL FLOW OF THE PROGRAM IS AS FOLLOWS:

1. GET COMMAND LINE
2. PARSE THE COMMAND LINE
3. VERIFY THAT NEWNAME.EXT IS UNIQUE
4. RENAME FILENAME.EXT TO NEWNAME.EXT

WRITTEN BY STEPHEN LIDDLE, WITH SPECIAL THANKS
TO PAT SWAYNE OF HUG FOR TECHNIQUE TIPS

FEBRUARY, 1984

```

```

***** ASSEMBLY CONSTANT DEFINITIONS

.EXIT EQU 0
.SCOUT EQU 2
.OPENR EQU 42Q
.CLOSE EQU 46Q
.RENAME EQU 51Q
.DECODE EQU 53Q
.ERROR EQU 57Q
$TYPTX EQU 31136A
BELL EQU 7
CHAN1 EQU 1
EC.FNF EQU 14Q
EOM EQU 80H
NL EQU 10
NUL EQU 0
PRGCHAN EQU -1
SPACE EQU 32
STAKTOP EQU 200Q

ASCII BELL
CHANNEL 1
FILE NOT FOUND
END-OF-MESSAGE MARKER
ASCII NEWLINE
ASCII NULL
PROGRAM CHANNEL
ASCII SPACE
LOWER BYTE OF 42200A

```

```

***** RENAM MAIN PROGRAM CODE

RENAM MVI A,PRGCHAN
SCALL .CLOSE
JC ERROR
LXI H,0
DAD SP
MOV A,L
CPI STAKTOP
IS THE STACK EMPTY?

CLOSE PROGRAM CHANNEL
IF WE COULDN'T CLOSE IT
PUT STACK POINTER IN HL
IS THE STACK EMPTY?

```

```

INX D
INX D
LXI H,NEWNM
MVI B,B
CALL COPY
MVI A,'.'
STAX D
INX D
MVI B,3
CALL COPY
MVI A,SPACE
STAX D
MVI A,CHAN1
LXI D,DEFAULT
LXI H,NEWNAME
SCALL .OPENR
JC GOTERR
MVI A,CHAN1
SCALL .CLOSE
CALL $TYPTX
DB NL,'The new name file already exists!'
DB NL,'File name NOT changed.' NL+EOM
JMP EXIT

GOTERR CPI EC.FNF
JNZ ERROR
POP H
LXI D,DEFAULT
LXI B,NEWNAME
SCALL .RENAME
JC ERROR
CALL $TYPTX
DB NL,'Successfully renamed'.' '+EOM
XRA A
SCALL .EXIT
GO BACK TO HDOS

***** SUBROUTINES
*** COPY - COPY NON-NULL CHARACTERS
* COPIES CHARACTERS FROM (HL) TO (DE),
* WITHOUT COPYING ANY NULL CHARACTERS.
* ENTRY (B) = COPY COUNT
* (HL) = COPY FROM
* (DE) = ADDRESS OF COPY BUFFER
* EXIT (DE) = POINTS PAST LAST CHARACTER
* (HL) = ORIGINAL (HL) + COPY COUNT
* USES ALL EXCEPT (C)
COPY MOV A,M
INX H
GET NEXT CHAR
INX H
INCREMENT POINTER

SKIP PAST THE PRE-SET COLON
STUFF THE NEW FILE NAME
INTO NEWNAME STRING
PUT '.' INTO NEWNAME STRING
PUT EXTENSION INTO NEWNAME STRING
MARK END OF FILE NAME
OPEN ON CHANNEL 1
USE STANDARD DEFAULTS
WE'RE GOING TO OPEN NEWNAME
TRY TO OPEN IT
WE MUST HAVE AN ERROR TO CONTINUE
CLOSE IT BACK UP
NL,'File name NOT changed.' NL+EOM
IS IT FILE NOT FOUND?
NO- PROCESS HDOS ERROR
YES- RENAME IT
USING STANDARD DEFAULTS
TO NEWNAME
ERROR IN RENAMING

```



JZ HELP YES- PRINT HELP MESSAGE  
 CALL SKIPSPC SKIP OVER ANY LEADING SPACES  
 PUSH H SAVE ADDRESS OF OLDFILE.NAM  
 CALL SKIPNAM SKIP OVER OLDFILE.NAM  
 MOV A.M MUST HAVE A SPACE AFTER OLDFILE  
 CPI SPACE SKIP TO THE 'TO'  
 JNZ HELP  
 CALL SKIPSPC MUST HAVE 'TO' NEXT  
 MOV A.M FOUND THE 'T'  
 CPI 'T'  
 JZ COMPO  
 NOTO \$TYPTX  
 CALL NL, 'Expected to find " TO "', EOM  
 DB HELP  
 JMP  
 INX H DIDN'T FIND 'TO'  
 MOV A.M FOUND 'TO'  
 CPI 'O'  
 JNZ NOTO  
 INX H MUST HAVE A SPACE AFTER 'TO'  
 MOV A.M MORE THERE THAN 'TO'  
 CPI SPACE SKIP TO BEGINNING OF NEWFILE.NAM  
 JNZ NOTO  
 CALL SKIPSPC  
 PUSH H SAVE LOCATION OF NEWFILE.NAM  
 LXI D, DEFAULT USE STANDARD DEFAULTS  
 LXI B, NEWNAM  
 SCALL .DECODE  
 JC ERROR  
 POP H  
 PUSH H IS THERE A DEVICE ON NEWFILE.NAM?  
 CALL CHECKDV  
 POP H  
 CALL SKIPNAM SKIP OVER NEWFILE.NAM  
 CALL SKIPSPC SKIP ANY TRAILING SPACES  
 MOV A.M  
 CPI NUL  
 JNZ HELP  
 POP H GET ADDRESS OF OLDFILE.NAM  
 PUSH H SAVE IT FOR THE SCALL .RENAME  
 LXI D, DEFAULT POINT TO DEFAULT STRING  
 LXI B, OLDNAM  
 SCALL .DECODE  
 JC ERROR  
 NODEV H, OLDDEV  
 LXI D, NEWNAME  
 LXI B, 2  
 MVI B, 2  
 CALL COPY  
 MOV A.M  
 ORI 'O'  
 STAX D  
 YES- PRINT HELP MESSAGE  
 SKIP OVER ANY LEADING SPACES  
 SAVE ADDRESS OF OLDFILE.NAM  
 SKIP OVER OLDFILE.NAM  
 MUST HAVE A SPACE AFTER OLDFILE  
 SKIP TO THE 'TO'  
 MUST HAVE 'TO' NEXT  
 FOUND THE 'T'  
 \$TYPTX  
 NL, 'Expected to find " TO "', EOM  
 HELP  
 DIDN'T FIND 'TO'  
 FOUND 'TO'  
 MUST HAVE A SPACE AFTER 'TO'  
 MORE THERE THAN 'TO'  
 SKIP TO BEGINNING OF NEWFILE.NAM  
 SAVE LOCATION OF NEWFILE.NAM  
 USE STANDARD DEFAULTS  
 PUT DECODED NEWFILE.NAM INTO NEWNAM  
 ILLEGAL FILE SPECIFICATION  
 IS THERE A DEVICE ON NEWFILE.NAM?  
 SKIP OVER NEWFILE.NAM  
 SKIP ANY TRAILING SPACES  
 IS THE COMMAND LINE BUFFER EMPTY?  
 NO- PRINT HELP MESSAGE  
 GET ADDRESS OF OLDFILE.NAM  
 SAVE IT FOR THE SCALL .RENAME  
 POINT TO DEFAULT STRING  
 PUT OLDFILE.NAM INTO OLDNAM  
 ILLEGAL FILE SPECIFICATION  
 NOW WE MUST SET UP THE NEWNAME  
 STRING AS 'DYN:NEWFILE.NAM'  
 MOVE DEVICE NAME TO NEWNAME  
 TURN DEVICE UNIT INTO ASCII CHAR  
 STUFF IT INTO NEWNAME STRING

CPI NUL DO WE COPY THIS?  
 JZ NEXT NO- DO NEXT CHAR  
 STAX D YES- STORE IT  
 INX D AND INCREMENT  
 DCR B ONE DOWN--  
 JNZ COPY --- AND (B) TO GO  
 RET  
 \*\*\*  
 \* SKIPSPC - SKIP OVER SPACES  
 \*  
 \* ENTRY (HL) = BUFFER ADDRESS  
 \* EXIT (HL) = POINTS PAST SPACES  
 \* USES (HL), (A)  
 SKIPSPC MOV A.M GET CHARACTER  
 INX H POINT TO NEXT CHAR  
 CPI SPACE IS IT A SPACE?  
 JZ SKIPSPC YES- SKIP IT  
 DCX H NO- BACK UP (HL)  
 RET  
 \*\*\*  
 \* SKIPNAM - SKIP OVER FILE NAME  
 \*  
 \* SKIPS ALL CHARACTERS UNTIL  
 \* SPACE, NEWLINE, OR NUL.  
 \*  
 \* ENTRY (HL) = BUFFER ADDRESS  
 \* EXIT (HL) = POINTS PAST FILE NAME  
 \* USES (HL), (A)  
 SKIPNAM MOV A.M GET CHARACTER  
 INX H  
 CPI SPACE IS IT A SPACE?  
 JZ PAST YES- WE'RE PAST  
 CPI NUL IS IT A NEWLINE?  
 JZ PAST YES- WE'RE PAST  
 CPI NUL IS IT A NULL?  
 JNZ SKIPNAM NO- DO ANOTHER  
 DCX H BACK UP POINTER  
 RET  
 PAST  
 \*\*\*  
 \* CHECKDV - CHECK FOR DEVICE SPECIFICATION  
 \*  
 \* CHECKS TO SEE IF THERE IS A DEVICE NAME  
 \* AT THE CURRENT MEMORY LOCATION.  
 \*  
 \* ENTRY (HL) = ADDRESS AT WHICH TO CHECK  
 \* EXIT EITHER RETURNS TO MAIN PROGRAM  
 \* OR EXITS WITH A HELP MESSAGE.  
 \* USES (HL), (A), (B)  
 CHECKDV MVI B, 4 COLON WITHIN 4 CHARACTERS  
 CHECK1 MOV A.M GET NEXT CHARACTER  
 INX H POINT TO NEXT CHARACTER  
 CPI ' ' IS THIS A COLON?  
 JNZ NEXTCH NO- GET CHECK NEXT CHAR  
 CALL \$TYPTX YES- PRINT HELP MESSAGE  
 DB NUL, 'Illegal to specify device for new name.', BELL+EOM  
 JMP HELP

```

NEXTCH DCR B CHECK1 GO FOR ANOTHER
JNZ RET
RET

***
ERROR - PROCESS HDOS ERROR
PRINTS HDOS ERROR MESSAGE, THEN
PRINTS RENM HELP MESSAGE, AND EXITS.

ENTRY (A) = ERROR CODE
EXITS TO HELP

MVI H,BELL RING THE BELL
CALL .ERROR PRINT HDOS ERROR MESSAGE
HELP $TYPTX

DB NL,NL,'The proper use of this utility is',NL
DB NL,' >RENM <dev:>OLDFILE.NAM TO NEWFILE.NAM'
DB NL,NL,'<dev:> is an optional disk drive name'
DB ', and defaults to SYD:.'
DB NL,'OLDFILE.NAM is the file to be renamed.'
DB NL,'NEWFILE.NAM is the new name for OLDFILE.NAM.'
DB NL,NL+$OM
JMP EXIT

```

```

***** DATA SECTION
NEWNAM DS 3 FOR DEVICE
DB ' '
DS 13 FOR NEWNAME.EXT

NEWNAM DS 1 RESERVED
NEWDEV DS 2 DEVICE NAME
DS 1 DEVICE UNIT
NEWNM DS 8 FILE NAME
NEWEXT DS 3 FILE EXTENSION
DS 4 RESERVED

OLDNAM DS 1 RESERVED
OLDDEV DS 2 DEVICE NAME
DS 1 DEVICE UNIT
DS 8 FILE NAME
DS 3 FILE EXTENSION
DS 4 RESERVED

DEFAULT DB 'SYD' DEFAULT DISK DRIVE
DB 0,0,0 NO DEFAULT EXTENSION

END RENM

```

# H-1000

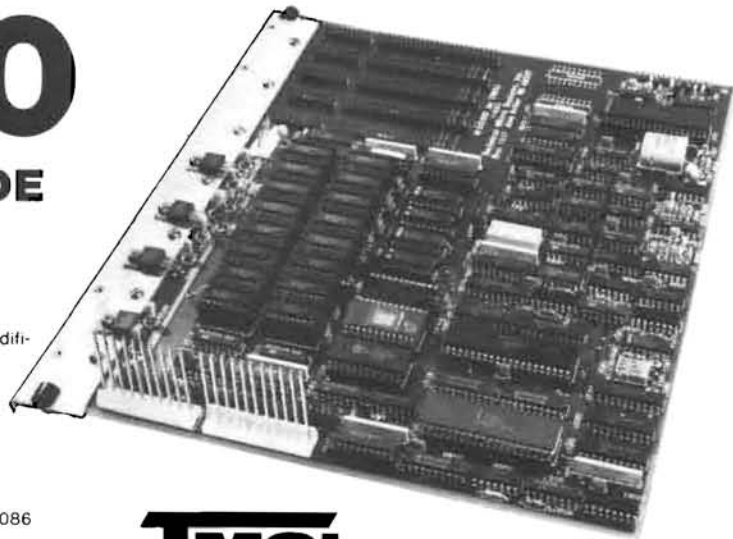
## A Z80/8086 UPGRADE FOR THE H89/Z89

### HARDWARE

- plug-in replacement for the H89/Z89 CPU board; no modifications required
- dual CPUs: Z80 and 8086
- 256K RAM standard; sockets for up to 1 megabyte RAM
- 5 I/O slots
- faster program execution: 2/4 MHz for Z80, 8MHz for 8086
- fully compatible with all Heath/Zenith peripherals

### SOFTWARE

- runs all Heath/Zenith software without modification
- compatible with Zenith Z100 and IBM Personal Computer
- choice of MSDOS or CP/M-86 for the 8086
- supplied with diagnostic software package
- "soft disk" feature: copies an entire disk in RAM for instant disk access
- supports multi-user and multi-task operating systems



## TMSI

Technical Micro Systems Inc.

P.O. Box 7227, Dept. H  
366 Cloverdale • Ann Arbor, Michigan 48107  
(313) 994-0784

We accept MasterCard and VISA.  
Serious Dealer Inquiries Invited

The H-1000 is a quality product of Technical Micro Systems, Inc., manufacturers of innovative microcomputer systems since 1979

H-1000 and TMSI are trademarks of Technical Micro Systems, Inc. H89, Z89, Z100 and HDOS are trademarks of Heath/Zenith Corp., Benton Harbor, Michigan. MSDOS is a trademark of Microsoft, Bellevue, Washington. CP/M and CP/M-86 are trademarks of Digital Research, Inc., Pacific Grove, California.

# More About CP/M Disk Errors



Normand E. Lavigne  
184 Washington St.  
New Bedford, MA 02740

The disk drives in our H/Z-89/90 computers are belt driven mechanisms. Since the absolute speed attained with any belt drive can be adversely affected by changes in temperature and humidity, it is important to have a running indication as to whether or not the speed of our drives is being altered by changes in climate. This article presents a modification of an earlier program presented by Pat Swayne in REMark #30. The present modifications allow the program to run under CP/M 2.2 versions 02, 03, and 04. Furthermore, the program now reports which version is in current use since not all users are likely to change all of their large collection of disks over to CP/M 2.2.04 just because it is now available. This article is presented in tutorial form so that even the beginner can take advantage of the benefits of program ERROR234.

In New England, where the temperature can be anywhere from -5 to +105 degrees Fahrenheit and the relative humidity can easily vary from 20% through 100% from season to season, it has proven to be quite useful to have an indication of disk drive soft error rates. With these broad changes in climate, I have observed that the resulting speed changes in my H-17 and H/Z-37 drives can cause anything from several soft errors through to occasional hard errors especially on hot, humid summer days. I have, in fact, found that there is no one speed setting which will provide error-free operation for summer and winter especially on the H/Z-37's, here in southern New England. This is not a problem unique to my own particular drives. It has also been observed by several other computer users in our area. With a program that reports the number of soft disk errors, we are forewarned that the drive speeds need to be adjusted to accommodate our outlandish seasonal changes. Furthermore, if you smoke in your computer room, an occasional cleaning of the drive belt and worm drive shaft with alcohol also helps tremendously. (Note that if you clean the worm drive shaft, it should then be lightly lubricated with a light machine oil before use.) Since our Z-89 with both hard and soft sectored controllers is in operation from 8 to 10 hours a day using word processing for textbook-writing, the risk of losing large amounts of text looms over the system practically all day every day. You know the old saying, "Forewarned is Forearmed"? Well we just feel more comfortable knowing when to adjust drive speeds before that fatal hard error ever occurs.

This is probably a good time to discuss hard versus soft errors, especially for the newcomer to CP/M and computing. Whenever the operating system is attempting to read or write information on disk, anything that interferes with the operation causes the system to re-try that operation. Interference can come from dust on the disk surface which the envelope liner has not yet trapped, small scratches on the surface, drive speed irregularities, etc. Anything the operating system

or the controller doesn't like causes what is called a soft error and makes CP/M record the soft error occurrence in memory and the operation is tried again. If CP/M can successfully perform the operation on one of these subsequent attempts, you have no way of knowing that soft errors ever happened. All you see is the end result of the operation like you always expected to see anyway. However, there is a potential problem here. This can develop into a situation where ignorance is definitely NOT bliss! The horrible fact is that when CP/M cannot perform a successful information transfer after ten (10) re-tries, the BDOS portion of CP/M objects, a hard error is produced and the file you were transferring is lost. Without a special utility program to inform you that soft errors are beginning to occur, these harmless errors are completely transparent and you will only be made aware of the problem after a hard error crash. Because I depend so much on my computer, I feel that I can't afford to wait for this type of problem to creep up on me. I want to know there's a potential problem long before the crash, especially since it's so easy to do.

The program presented here is a modification of the assembly language program called "ERROR" which Pat Swayne offered in REMark issue #30. Although ERROR operates properly for CP/M 2.2.02 & 2.2.03, some further modifications are needed for CP/M 2.2.04 since the addresses for SECNT17 and SECNT37 have been changed in the new BIOS. SECNT17 and SECNT37 are the memory locations where CP/M keeps its soft error counts for the H-17 and H/Z-37 drives. Whenever you cold boot your computer from the H: prompt, CP/M sets these RAM memory locations to zero, then increments them by 1 each time a soft error occurs. SECNT17 gets incremented for every soft error on an H-17 (hard sectored) drive and SECNT37 is incremented for every soft error on an H/Z-37 (soft sectored) drive. This new version of ERROR is called "ERROR234" since it detects and reports H-17 soft errors for CP/M 2.2.02 and H-17 & H/Z-37 soft disk errors for both the 03 and 04 versions of CP/M. ERROR234 also reminds the user of the CP/M version in current use on his disk since the version number is included in the displayed messages.

Let's face it, although we all should know that we can occasionally run drive diagnostic tests to check the condition of our drives, most of us find that this is too time consuming or we just don't think of doing them. Some of us may not even be aware that there are such tests. Anyway, it's only human to tend to forget to run any kind of tests while everything is operating properly and find out there's trouble when the inevitable crash occurs. Since no one really can afford the system down-time associated with catastrophic failure, it would be wise to have a daily indication of impending problems. Also, with a

warning that problems might just be lurking around the corner, now would be the time to adjust or clean the disk drives and run our speed and/or other diagnostics before the problem gets so severe that we can no longer boot-up.

For you pros out there, if you are interested in finding out how the program works, the documentation included in the listing in the form of remarks in the rightmost column explain what every line of code is accomplishing. Also, the program has been broken down into logical sections with their own titles so that the structure of the program is easy to follow. Use the following listing as a short guide to the flow of the program:

- 1) The CP/M stack is first saved to preserve its integrity.
- 2) A new stack is established for ERROR234 use.
- 3) The version number of the booted CP/M is found.
- 4) The message "CP/M VERSION 2.2.0X" is printed to the console where "X" is 2, 3, or 4 depending on which version you are currently using.
- 5) The FIND AND PRINT H-17 ERROR COUNT section first determines the proper offset to SECNT17 for the CP/M version in use and places it into the (DE) register pair.
- 6) The DECOUT routine is then called to convert the count into decimal and to print the decimal result out to the console.
- 7) On return from DECOUT, BDOS is called upon to print the H-17 message.
- 8) The FIND AND PRINT H/Z-37 ERROR COUNT section operates the same as the H-17 section described above but for SECNT37 and the H/Z-37 message.
- 9) Another difference between the H-17 and H/Z-37 sections is that the H/Z-37 routine is aborted if the current version of CP/M is 2.2.02 since this older version apparently did not support the H/Z-37 soft-sectored drives. The program therefore goes directly to the EXIT label if version 02 has been found.
- 10) The EXIT section calls on BDOS to print the "SINCE LAST COLD BOOT" message, restores the original CP/M stack, then returns to CP/M.

For you newcomers, even if you've never used assembly language before, don't be afraid of it. Look at the listing of PROGRAM ERROR234. It's only about 150 lines of code and it's really nothing to be afraid of. If you're not really interested in the inner workings of the program, you can still get it working quite easily so don't bother worrying about it. Too many people cringe at the thought of assembly language and there's really no good reason to feel this way when you have a listing of a working program. You don't have to know anything about assembly language to make use of a source code listing. Just type it into a file like you would type any other document, like a letter to a friend. However, just as there are accepted rules as to how a letter should be laid out, there are also some rules as to how a source code listing should be laid out. So, when you type in this listing, follow it EXACTLY, character-by-character and space-by-space. The lineup of the various columns is very important and can easily be achieved with your TAB key. As far as actually typing this listing into a file, you can use CP/M's ED.COM program if you are familiar with its operation, or use your own word processor. Since most users have some form of word processing software like WordStar, Magic Wand, PIE, etc., this is probably the easiest way to get your source file for this program onto a diskette. I always use WordStar in the "N" or non document mode since the page-breaks of the document mode tend to confuse the assembly language com-

piler. It is also important that you call your file ERROR234.ASM since the ASM part is what the assembler will be looking for in the next step.

Now for the easy task of actually assembling your source code file. If you have H/Z-37 external drives, there's ample room on one disk to do the whole job. Just make sure that ASM.COM, LOAD.COM, and ERROR234.ASM are on one of the drives then type the following command at the A> prompt:

```
A>ASM ERROR234.AAZ
```

The AAZ portion of the command line will instruct the ASM program that your ERROR234.ASM is currently on drive A (the first A), that you want the HEX file that the assembler will create to be placed also on drive A (the second A), and that you do not want a print file of the assembly process (the Z). If you've never done this before, you're not interested in the print or .PRN file since it involves Machine Language, another frightening subject to many users. So, we're not going to scare you with it now that you just got the guts to do an assembly! Incidentally, since there are so many different system configurations out there, you may need to juggle the drive designations around to suit yourself. Just make sure that the ASM.COM program knows where to find your source file, (ERROR234.ASM), and that you know where the HEX file will be produced. This ERROR234.HEX file will be your key to getting the ERROR234.COM file you need to run this program from CP/M.

Now it's time to produce the program file, ERROR234.COM. To do this, you must run the CP/M program called LOAD.COM in the following way:

```
A>LOAD ERROR234
```

This command line will run the LOAD program from your disk and go looking for the ERROR234.HEX file that the assembler program produced. It will make the appropriate changes needed to create the ERROR234.COM file and place this file back on your disk automatically. Now, whenever you type:

```
A>ERROR234
```

from the CP/M prompt, the ERROR234.COM program will run and report to you, on the console screen, the version of CP/M that you are using as well as the number of soft errors that have occurred on your drives. Simple as that! Piece of cake, right? If everything went as planned, this is the message you should see on your screen:

```
A>ERROR234
```

```
CP/M VERSION 2.2.0X      (where X is your CP/M version number)
Y H17 SOFT DISK ERRORS  (      Y is the number of H17 errors)
Z H37 SOFT DISK ERRORS  (      Z is the number os H37 errors)
SINCE LAST COLD BOOT.
```

If the system is in good shape, Y and Z will be zero. If a problem is just starting to creep up, you may see 2 or 3 errors reported. Don't get upset with a couple of reported errors since these could have been caused by random things such as a power line disturbance while access to the disk was being attempted. However, keep a watchful eye on the report to observe if the errors increase over the next few days. A steady increase in soft errors indicates that the drives need attention, and as stated earlier, if you wait till you're getting 10 soft errors, the system will crash. Now you'll be in real trouble since you won't even be able to Boot up on a diagnostic disk! An increase from 0 to 5 or 6 errors within a one week period definitely indicates that attention is needed immediately.

Here at CCS where we are so dependent on the integrity of our computer system, all of the bootable WordStar disks run the CP/M





```

; OFFSET TO SECTN37 FOR VERSION 4
; GO PRINT H37 ERROR COUNT
; GET BIOS ADDRESS
; OFFSET TO SECTN37 FOR VERSION 3

D,4PH-3
PR37
BASE+1
D,4BH-3
D
E,M
H
D,M

; (DE) = H37 ERROR COUNT
; PRINT NUMBER OF H37 ERRORS
; POINT TO H37 MESSAGE
; GET READY TO START PRINTING
; PRINT H37 SOFT ERROR MESSAGE

D,LBMSG
C,PSTRING
BDOOS
H
SPHL
RET

; POINT TO "LAST BOOT" MESSAGE
; GET READY TO PRINT A STRING
; PRINT IT
; GET OLD STACK
; SET IT
; GO BACK TO CP/M

PRINT NUMBER IN (HL) IN DECIMAL
DECOU: PUSH B
        PUSH D
        PUSH H
        LXI B,-10
        LXI D,-1
        DAD B
        INX B
        JC DX
        LXI B,-10
        DAD B
        XCHG
        MOV A,H
        ORA L
        CNZ DECOU
        MOV A,E
        ADI '0'
        MOV E,A
        MVI C,CONOUT
        CALL BDOOS
        POP H
        POP D
        POP B
        RET

; SAVE REGISTERS
; RADIX FOR CONVERSION
; SUBTRACTION COUNTER
; SUBTRACT 10
; INCREMENT COUNTER
; REPEAT UNTIL OVERFLOW
; ADD RADIX BACK IN ONCE
; (DE) = DIGIT, (HL) = NUMBER/10
; DONE?
; CALL RECURSIVELY UNTIL DONE
; GET CHARACTER TO PRINT
; ADD ASCII BIAS
; PUT RESULT IN E
; SET UP FOR CONSOLE OUT SERVICE
; PRINT DIGIT
; RESTORE REGISTERS
; SAVE VERSION NUMBER HERE

; CP/M VERSION 2.2.0', '$'
; H17 SOFT DISK ERRORS', CR,LF, '$'
; H37 SOFT DISK ERRORS', CR,LF, '$'
; SINCE LAST COLD BOOT.',
; CR,LF, '$'

DBS
DS
DS
END

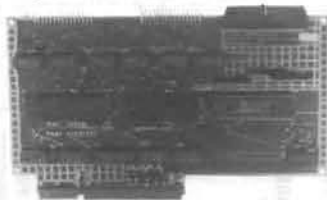
; STACK
; STACK SPACE
; MAIN

```

## H/Z89 PERIPHERALS from SECURED COMPUTER SYSTEMS

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

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



#### Features:

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

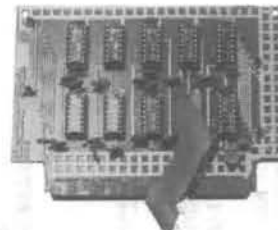
- 2 Serial Ports
- Supports: Ring Input, External Clock, Auto Dialer
- 3 Port Parallel with 2 Level Interrupt Control
- Fully compatible with all models of H/Z 88, 89, 90 using CP/M or HDOS.
- Fully tested, 90 day warranty, two serial cables and a parallel cable (internal to computer) and software driver.

**PRICE \$199.00**

Shipping & Handling \$10.00

### 16K RAM EXPANSION CARD

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



#### Fully compatible with:

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

**NOW INCLUDING SUPPORT MOUNTING BRACKET**

#### Featuring:

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

**Now Only \$65.00**

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

Shipping & Handling \$5.00



PRICES ARE LESS SHIPPING AND TAX IF RESIDENT OF CALIFORNIA  
MAIL ORDER: 12011 ACLARE, CERRITOS, CA 90701 (213) 924-6741  
TECHNICAL INFO/HELP:  
8575 KNOTT AVE., SUITE D, BUENA PARK, CA 90620 (714) 952-3930

Terms and specifications subject to change without notice.

**ZENITH**  
data systems  
SERVICE CENTER



## Part 3 - We Can Add and Print Data

# Practical File Management

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

In the past two articles of this series, we began a modularized program designed to manage data. The menu and control portions were completed in Part 1. Part 2 explained some programming philosophies and the use of key files. We then added a module which lets us name and switch the file we're working on, and sets up an array in RAM containing our key information.

This month's article will add more modules and get us closer to a final program. When this month's additions are made, you'll be able to use the program as is. Only improvements will follow. We'll continue to specify our numeric variables in the format using the fewest bytes of data acceptable for the job to be done. You'll also notice that we add lines of programming in these new modules which are identical to other lines found elsewhere. This situation will be remedied in the final part of the series when we discuss shrinking the size of a program, optimizing RAM, and modifying the program to fit your own special application.

Now that we can open a data file, the next thing we've got to do is put something into it. That means the add sub-routine must be developed now. A look at line 1180 of the control program tells us we'll start on line 2000. Before adding any of the new programming to your existing work, make sure you've got a backup copy of the latest version on a separate disk. It's a good idea to put a write-protect tab on the backup disk (after you've written the backup file to it, of course). That way you can read it if anything happens to the original, but it can't be overwritten and lost till you remove the tab. Learn to back up your work at the end of a session. Do it more often during long sessions. We're all guilty of saving a few minutes or a \$3.00 disk by not making copies. I for one learned my lesson when I lost the directory tracks on a 640K disk which was nearly full. Fortunately most of it was backed up, but I still lost many days of work. Somehow the extra five minutes to update backups doesn't seem as long as it once did.

Figure 1 shows the first steps of our data adding module. As with each module, we must first clear all the menu information from the screen. However, a check of line 2000 shows that we begin on line 2 of the screen and erase to the end of the screen. Checking back to the menu module, you'll see lines 1040-1060 provided us with file name and drive information for the file we're working on. That stays on line 1 until the new file option is selected again. It may have been more

convenient to use the 25th line for this status line, but I personally prefer that kind of information at the top of the screen. Another bit of my programming philosophy shows up on line 2110. Many file handling programs such as the very good MAILPRO require that the operator strike an option key from the menu for every addition to a file. I for one don't make an update each time I have a new piece of data, but prefer to add several entries to a file at a time. When the program enters an option which could perform multiple operations (Add, Change, Delete, or Look up), the operator has to request an exit from that option by pressing the red key. If you prefer the other type of operation (press A for every addition) just delete lines 2110 and 2160, and change line 2460 to GOTO 1000.

Figure 1

```
2000 PRINT FNCA$(2,1);EES          'CLEAR LINES 2 - 24
2010 PRINT FNCA$(9,21);"MONTH"
2020 PRINT FNCA$(11,21);"DATE"
2030 PRINT FNCA$(13,21);"YEAR"
2040 PRINT FNCA$(15,21);"VALUE"
2050 PRINT GMS                     'ENTER GRAPHICS MODE
2060 PRINT FNCA$(10,27);"zz"       'UNDERLINE
2070 PRINT FNCA$(12,27);"zz"
2080 PRINT FNCA$(14,27);"zz"
2090 PRINT FNCA$(16,27);"zzzzzzz"
2100 PRINT GOS                     'EXIT GRAPHICS MODE
2110 PRINT FNCA$(24,21);
      "PRESS THE RED KEY TO RETURN TO THE MENU";FNCA$(1,1)
```

Figure 2 shows the next logical steps for adding data. We'll input the data to be filed away for future reference. As the program follows stocks, we need the date and a value (I use the weekly closing price). Each element of the date is input as two digits. The second day of a month must be entered as 02. I did this so we could use the INPUT\$ command, thus eliminating the need to press the carriage return. Note that very clear instructions to the operator will appear on the screen. When it comes time to enter the value, we want decimal values. Stocks are quoted in fractions of dollars. To aid the operator, program lines 2290 and 2300 print a conversion table on screen lines 20 and 22. Never overlook the fine points of programming. Terminal and computer users should be told what to do on the screen. Think about this the next time you reluctantly dig out the documentation to a program you're running which isn't quite as user friendly as you'd like. Our programming lines look like this:

Figure 2

```

2120 PRINT FNCAS(18,21);
      "ENTER THE MONTH AS 2 DIGITS (01 - 12)"
2130 PRINT FNCAS(9,27);      'MOVE CURSOR TO UNDERLINE
2140 M$=INPUT$(2)
2150 PRINT M$
2160 IF M$=CHR$(27)+"Q" GOTO 1000
2170 PRINT FNCAS(24,1);EL$;FNCAS(1,1)
2180 PRINT FNCAS(18,1);EL$
2190 PRINT FNCAS(18,21);
      "ENTER THE DATE AS 2 DIGITS (01 - 31)"
2200 PRINT FNCAS(11,27);
2210 D$=INPUT$(2)
2220 PRINT D$
2230 PRINT FNCAS(18,1);EL$
2240 PRINT FNCAS(18,21);
      "ENTER THE LAST 2 DIGITS OF THE YEAR"
2250 PRINT FNCAS(13,27);
2260 Y$=INPUT$(2)
2270 PRINT Y$
2280 PRINT FNCAS(18,1);EL$
2290 PRINT FNCAS(20,16);
      "1/8 = .125 1/4 = .25 3/8 = .375 1/2 = .5"
2300 PRINT FNCAS(22,21);
      "5/8 = .625 3/4 = .75 7/8 = .875"
2310 PRINT FNCAS(18,21);
      "ENTER THE STOCK VALUE IN DECIMAL FORM"
2320 PRINT FNCAS(15,27);
2330 INPUT "",V

```

Figure 3 is quite simple and completes our data entry procedure. Line 2340 builds the date to be stored. It is important to note that we took the data as strings even though only numbers will be encountered. We did this so the strings could be added. With string variables, addition is performed by concatenation of the individual strings. That is, they are hooked together like the cars of a train, not summed as numbers are.

Notice that when we add these strings they are in the order of years, months, days. We won't print or display them this way. Nobody does. So why store them in this oddball fashion? To permit sorting by date. The farther left a character is in a string, the greater its value in comparisons. It's the same as numeric data in this respect. We have columns for units, tens, hundreds, etc. By putting the years first, they have the greatest effect on the order of the sort output. If we had put months first, all January values would print, then all February values, etc. regardless of the year specified. The rest of the module provides data to the operator.

Figure 3

```

2340 LSET A$=Y$+M$+D$      'PREPARE DATE FOR FILE ENTRY
2350 LSET B$=MK$(V)        'PREPARE VALUE FOR FILE ENTRY
2360 PUT #1,NR%+1          'PUT IN FILE AFTER LAST RECORD
2370 NR%=NR%+1            'INCREMENT RECORD COUNTER
2380 PRINT FNCAS(18,1);EE$
2390 PRINT FNCAS(19,21);
      "RECORD NUMBER ";NR%;"ADDED TO THE FILE"
2400 FOR X=1 TO 500:NEXT X
      'DELAY FOR OPERATOR TO READ MSG.
2410 PRINT FNCAS(19,1);EL$ 'ERASE LINE
2420 PRINT FNCAS(9,26);LE$ 'ERASE DATA FROM UNDERLINE
2430 PRINT FNCAS(11,26);LE$
2440 PRINT FNCAS(13,26);LE$
2450 PRINT FNCAS(15,26);LE$
2460 GOTO 2110            'BACK FOR NEW INPUT

```

Add all the programming presented above (lines 2000-2460) to your program and save it. Now when you run it, you can add data to your file. For those of you who don't have stock values handy, the following chart gives some actual weekly closing values for AT&T. You can use them.

MONTH	DATE	YEAR	VALUE
2	11	83	64 1/4
2	18	83	67 3/4
2	25	83	68 3/8
3	4	83	66 1/4
3	11	83	66 3/8
3	18	83	66
3	25	83	65 7/8
4	1	83	64 3/8
4	8	83	64 3/4
4	15	83	66 5/8
4	22	83	66 1/4
4	29	83	68
5	6	83	69 1/2
5	13	83	67 1/2
5	20	83	65 5/8
5	27	83	66 3/4
6	3	83	65
6	10	83	63 7/8
6	17	83	64 1/4
6	24	83	62 7/8

Now that we have data in the file, we can print it using the LIST utility from CP/M or similar HDOS utilities as explained in my previous articles on random files. However, we want to access our data from within this program using Lookup or Print options. Because all reference to the data file is by way of record numbers in the key file and the key array B%(), we must first generate an accurate key file. This is done by our Sort routine. We must run the sort every time data is added to or deleted from our file if we want the Change, Delete, Lookup, or Print options to work.

Figure 4 shows the steps in beginning our sort option. After the screen is set up, we'll dimension the necessary arrays to the number of records now in the data file. We'll need an array for record numbers. We decided last month to use B%. (That is array B with all the data stored as integers so we'll only need 2 bytes of RAM to store each entry). We'll also need an array for the key information to be sorted (the dates we put into our data file). This will have to be a string array as the data we're entering will be strings. We'll use A\$. As in last month's article, we must ERASE the arrays before we can re-DIMension them. That requires us to add the new array A\$ to line 400.

With the arrays dimensioned, we'll KILL our old key file to get it off the disk before we write a new one. Then we'll open and field a new key file. Why don't we just write over the old key file? We could, but if we're sorting because of a deletion, there would be useless information stored at the end of the file. It would never be read because reading the key file is controlled by the number of records in the data file. However, it would make the key file longer than necessary. While this is a minor point to those of us with two controllers and multiple disk drives, it is an inconvenience to those with hard sectoring (90K/Drive Max.) and only one drive. It's also a very poor practice to keep junk on disks and we should all learn to purge our disks of the trash that is stored on them. It's a practice that, if adopted, could save a lot of us the 25-buck price of a box of disks.

Finally, we'll read the dates of each record in our data file into the array A\$() and place the associated record number into B%() on lines 20090-20130 and use line 20140 to inform the operator that this has been done.

Figure 4

```

400 DIM A$(10),B%(10)
20000 PRINT FNCAS(2,1);EE$
20010 ERASE A$,B%          'ERASE ARRAYS
20020 DIM A$(NR%),B%(NR%) 'RE-DIMENSION ARRAYS
20030 CLOSE #2            'CLOSE THE KEY FILE
20040 KILL FK$            'KILL KEY FILE FROM DISK
20050 OPEN "R",#2,FK$,2   'OPEN NEW KEY FILE

```



```

20060 FIELD #2,2 AS C$      'FIELD NEW KEY FILE
20070 PRINT FNCA$(3,21);
      "YOU HAVE SELECTED THE SORT ROUTINE"
20080 PRINT FNCA$(5,21);
      "NOW LOADING INFO INTO MEMORY TO SORT"
20090 FOR X%=1 TO NR%      'FOR EACH RECORD IN DATA FILE
20100 GET #1,X%           'GET A RECORD
20110 A$(X%)=A$          'PUT DATE IN ARRAY TO SORT
20120 B$(X%)=X%          'PUT RECORD NUMBER IN ARRAY
20130 NEXT X%             'NEXT RECORD
20140 PRINT FNCA$(7,21);"A TOTAL OF ";NR%;
      "RECORDS HAVE BEEN READ INTO MEMORY"

```

Figure 4 tells us that the next step to perform is a Quicksort. This sort was explained in the second part of my series on sorting in REMark issue 51 so its operation will not be explained here. We sort on the dates in A\$( ) and when a SWAP is required, the record numbers in B%( ) are also exchanged. Within the sort routine two arrays LL%( ) and UL%( ) are dimensioned. They must be included on line 400. The Quicksort additions to our program are:

**Figure 5**

```

400 DIM A$(10),B$(10),LL$(10),UL$(10)
20150 PRINT FNCA$(9,21);"NOW SORTING ";F$
20160 SL%=INT(NR%/3):IF LS%<1 THEN SL%=1
20170 ERASE LL%,UL%
20180 DIM LL$(SL%),UL$(SL%)
20190 SP%=1
20200 LL$(1)=1
20210 UL$(1)=NR%
20220 LL1%=LL$(SP%)
20230 UL1%=UL$(SP%)
20240 SP%=SP%-1
20250 LL2%=LL1%
20260 UL2%=UL1%
20270 K$=A$(INT((LL1%+UL1%)/2))
20280 IF A$(LL2%)>=K$ GOTO 20310
20290 LL2%=LL2%+1
20300 GOTO 20280
20310 IF K$>=A$(UL2%) GOTO 20340
20320 UL2%=UL2%-1
20330 GOTO 20310
20340 IF LL2%>UL2% GOTO 20380
20350 SWAP A$(LL2%),A$(UL2%):SWAP B$(LL2%),B$(UL2%)
20360 LL2%=LL2%+1
20370 UL2%=UL2%-1
20380 IF LL2%<=UL2% GOTO 20280
20390 IF LL2%>=UL1% GOTO 20430
20400 SP%=SP%+1
20410 LL$(SP%)=LL2%
20420 UL$(SP%)=UL1%
20430 UL1%=UL2%
20440 IF LL1%<UL1% GOTO 20250
20450 IF SP%>0 GOTO 20220
20460 PRINT FNCA$(9,1);EL$
20470 PRINT FNCA$(9,21);"SORT IS FINISHED"

```

The sort module is completed by writing the key data to the new key file and returning to the menu. Again, we attempt to prevent operator errors by making sure the correct disk is in the correct drive on lines 20480 and 20490. Line 20500 makes program execution wait until all is ready and the operator presses any key.

**Figure 6**

```

20480 PRINT FNCA$(11,21);
      "MAKE SURE THE CORRECT DISK FOR THE KEYFILE"
20490 PRINT FNCA$(13,21);"IS IN DRIVE ";D2$;
      " AND PRESS ANY KEY TO CONTINUE"
20500 I$=INPUT$(1)
20510 FOR X%=1 TO NR%      'FOR EACH RECORD
20520 LSET C$=MKI$(B$(X%)) 'READY KEY INFO FOR FILE
20530 PUT #2,X%           'PUT KEY INFO IN FILE
20540 NEXT X%            'NEXT RECORD
20550 GOTO 1000          'BACK TO THE MENU

```

The sort module presented above as lines 20000-20550 should work with any file. The only change required would be on line 20110 to

assure that we read the correct field from each record for sorting. Add line 400 and lines 20000-20550 to your program. Save it (and a backup), then run the program. If you have not previously added data to the file, do so now. Use the AT&T I gave above if you like. Then sort the file. You are now ready to print a graph of stock performance.

The full print sub-program will take 96 lines and requires a lot of explanation. Because we're so close to producing something visible, I'll present a much abbreviated version of the print module here without explanation. This way when you select the Print option, you'll get a printout of all values in your file. (HDOS users, LPRINT is a CP/M instruction to print on the line printer. You'll have to OPEN #3 as your LP:. Don't forget to LOAD LP: before MBASIC when you run this. Also substitute PRINT #3 for my LPRINT.) Next month we'll cover the full print option thoroughly. The final array to be dimensioned is V. It will hold the individual stock values in single-precision form.

**Figure 7**

```

400 DIM A$(10),B$(10),LL$(10),UL$(10),V(10)
17000 PRINT FNCA$(2,1);EE$
17010 PRINT FNCA$(3,21);
      "YOU HAVE SELECTED THE PRINT OPTION"
17020 PRINT FNCA$(10,21);
      "INPUT MONTH TO START THE CHART AS 2 DIGITS"
17030 FR%=1
17040 LR%=NR%
17620 IF FR%<1 THEN FR%=1
17630 IF LR%>NR% THEN LR%=NR%
17640 LV=999
17650 HV=0
17660 Z%=LR%-FR%+1
17670 ERASE A$,V
17680 DIM A$(Z%),V(Z%)
17690 FOR X%=FR% TO LR%
17700 W%=X%-FR%+1
17710 GET #1,B$(X%)
17720 A$(W%)=A$
17730 V(W%)=CVS(B%)
17740 IF V(W%)<LV THEN LV=V(W%)
17750 IF V(W%)>HV THEN HV=V(W%)
17760 NEXT X%
17770 PRINT FNCA$(11,21);
      "MAKE SURE THE PRINTER IS READY AND TURNED ON"
17780 PRINT FNCA$(13,21);
      "THEN PRESS ANY KEY TO PRINT THE CHART"
17790 P$=INPUT$(1)
17800 LPRINT "PERFORMANCE OF ";F$
17810 LPRINT "CHART STARTS ";P$
17820 LPRINT "CHART ENDS ";P$
17830 LPRINT "THE LOWER LIMIT OF THE CURVE IS ";LV
17840 LPRINT "THE UPPER LIMIT OF THE CURVE IS ";HV
17850 LPRINT
17860 LPRINT
17870 LPRINT TAB(10)"LOWER LIMIT";TAB(37)"MIDDLE";
      TAB(60)"UPPER LIMIT"
17880 LPRINT TAB(10) LV;TAB(37) (LV+HV)/2;TAB(60) HV
17890 LPRINT TAB(15)"V";TAB(40)"V";TAB(65)"V"
17900 FOR X%=1 TO Z%
17910 LPRINT MID$(A$(X%),3,2);"-";RIGHT$(A$(X%),2);"-";
      LEFT$(A$(X%),2);
17920 LPRINT TAB(INT(((V(X%)-LV)/(HV-LV))*50)+15)"*";
      TAB(70) V(X%)
17930 NEXT X%
17940 LPRINT CHR$(12)
17950 GOTO 1000

```

Add the above lines to your program and save it. Lines 17030 and 17040 are temporary and replace the final lines 17030-17610 as we'll explain next month. Now run the program and specify the file you sorted above. Select the print option and I think you'll be pleasantly surprised.

See you next month.



# Patch Page

Pat Swayne  
HUG Software Engineer

This article presents patches for the HUG Z-DOS Editor (885-3010-37) and CP/EMulator (885-3007-37).

## HUG Z-DOS Editor Patch

The Z-DOS version of the HUG editor will not properly load a text file that is not a multiple of 128 bytes in size. To fix the problem, locate the label READF and find the lines

```
OR      AL, AL      ;TEST FOR EOF
JZ      GTBYT1     ;OTHERWISE, PROCESS BYTE
```

Change these lines to

```
CMP     AL, 1      ;TEST FOR EOF
JNZ     GTBYT1     ;OTHERWISE, PROCESS BYTE
```

Re-assemble EDIT.ASM to produce a corrected editor. If the lines have already been patched in your copy of the editor, your .COM files have also been fixed and no changes are needed.

## CP/EMulator Patch

If you are using CP/EMulator with Z-DOS version 2 and your system drive is not A:, you should make the following patch to the file CMD.SYS supplied with CP/EMulator. You can make the patch with

either DDT or DEBUG, and both patches are shown. The first one is the DDT patch.

```
A>DDT CMD.SYS
NEXT PC
0900 0100
-S206
0206 32 0
0207 1A 0
0208 F5 0
0209 C1 .      (Type a period.)
->C           (Control-C)
A>SAVE 8 CMD.SYS
```

Here is the patch with DEBUG.

```
A>DEBUG
-NCMD.SYS
-L
-E206
0xxx:0206 32.0 1A.0
0xxx:0208 F5.0
-W
-Q
```

Without this patch, CP/EMulator will always attempt to access drive A: when you run it under Z-DOS 2, even if your system drive is not drive A:.



## ANNOUNCING THE MODIFIER

A disk utility that modifies the CP/M BIOS to be able to read and write to a number of 5.25" CP/M disk types.

There is a growing need for the everyday user of computer systems to be able to take data files home from the office to continue to work on them. The computers at home and at work may both run a version of CP/M, but the disk structures may be incompatible. This is especially a problem in the 5.25" world. MODIFY 89 was designed to address this problem. MODIFY 89 makes the CP/M operating system access a specified 5.25" drive as one of the below disk types. Disks placed in that drive that are of the specified type can be used as if they were one of the standard disk types accepted by the H8, H/Z89 or H/Z90 computers. Thus PIP, STAT, DIR and others will work for that disk also. The price for MODIFY 89 is \$49.95.

MODIFY 89 is set for the following disk types:

- Access S.S. D.D.
- Cromemco S.S. D.D.
- DEC VT180 S.S. D.D.
- IBM PC/Zenith 100 (CP/M) S.S. D.D.
- IBM PC/Zenith 100 (CP/M) D.S. D.D.\*
- Kaypro II S.S. D.D.
- Morrow Micro Decisions S.S. D.D.
- NEC PC-8001A S.S. D.D.
- Osborne S.S. S.D.
- Osborne S.S. D.D.
- Otrona D.S. D.D.\*
- Superbrain Jr. S.S. D.D.
- TI Professional S.S. D.D.
- TRS-80 Model I (Omnikron CP/M)
- TRS-80 Model III (MM CP/M)
- Xerox 820 S.S. S.D.
- Xerox 820-II S.S. D.D.
- Standard  
(Tests for H/Z37 and C.D.R. Disk types)

\* = Double sided 5.25" drive required

S.S. = single sided, D.S. = double sided, S.D. = single density, D.D. = double density

Limitations: MODIFY 89 is not a disk duplicate program. It is currently available for use with an H/Z89 or H/Z90 computer that has an FDC-880H double density 8" and 5.25" controller, using C.D.R.'s BIOS V.2.9 or with an H8 computer using the FDC-H8 by C.D.R. Systems, Inc.

MODIFY 100 will soon be released for the Z100 line of computers at a price of \$75.00.

Contact:

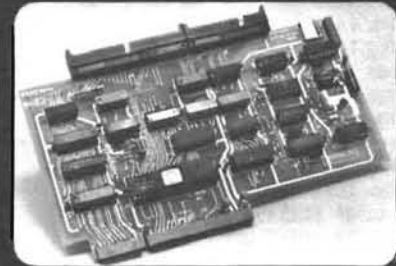


**C.D.R. Systems, Inc.**  
7210 Clairemont Mesa Blvd., San Diego CA 92111  
Telephone: (619) 560-1272

Or a C.D.R. Systems, Inc. dealer near you.

## 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.

**PRICED AT \$395**

**Includes controller board CP/M boot prom, I/O decoder prom, hardware/software manuals BIOS source listing. HDOS driver now available for \$50.00.**

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

# PALETTE -- A Powerful Graphics Package for the H/Z-100



Tom Huber  
Related Products Editor

The H/Z-100 Series of Computers are powerful, fast (enough for my needs), and colorful. About the only regret anyone could have in choosing a H/Z-100 is the lack of good game software, which is typically available for the more popular (and lower cost, thus, more widely distributed) home computers. However, when it comes to video power, I don't think there is a better computer on the market. (Incidentally, through the rest of the review, the references to the Z-100 apply equally to all Heath and Zenith and H/Z-100 non-PC series computers.)

Getting the most out of all this power is another story. I have tried writing my own artist program, only to find it slow and tedious. Then I discovered PALETTE from Software Wizardry.

PALETTE is a graphics design package, appropriately named because it is a paint brush that can be used to create masterpieces of color on the computer's screen. The program needs at least 192K of main memory and all three video banks fully populated for color (64K devices are faster than 32K devices in video memory, however, 32K devices do work). Palette supports, but does not need, a light pen and a graphics printer. Future versions (now in testing) will offer other input devices, trakballs, joysticks and digitizing pencil-pads.

## A Note About Color Monitors

PALETTE does best when used with a color display. I personally like the Zenith ZVM-135, although the ZVM-133 or ZVM-136 also will work. There are probably good alternative choices from other manufacturers, but when looking for a display, don't let price be your only guide.

The ZVM-135 is the most versatile of the color monitors from Zenith. It offers good resolution (compatible with the Z-100), and both composite and RGB inputs. In addition, there is also a small speaker and audio amplifier built into the unit, which makes it ideal as a high-quality video monitor for video recorders and disk players.

The ZVM-133 offers only an RGB input and features an exceptional display since it uses a "black matrix" color picture tube. This results in very good contrast without setting the brightness and contrast (black level) to their extremes. If I didn't like the idea of being able to use the audio and composite inputs of the ZVM-135 from other video sources, this monitor would be my first choice. Incidentally, in a test

unit that I checked out, the cable (ZVM-133-2) used reversed polarity of the sync inputs at the monitor. If you buy and use the ZVM-133 monitor with the ZVM-133-2 cable, you will have to reverse the sync settings on the video board of the Z-100.

The ZVM-136 is like the ZVM-133 in that it features RGB input only. It uses long-persistence phosphors in the picture tube, which I do not care for since scrolling and animation in this type of display typically produce a noticeable smearing of the images as they "move". Although I have not seen one of these monitors in action, I must assume that they will exhibit similar smearing. I have been told that the ZVM-136's overall picture quality is similar to the ZVM-133. If you are not bothered by the smearing during scrolling, but need an ideal monitor for use with the Z-100's interlaced mode, then this is the only color monitor in Zenith's line that will fit your needs.

## The Program Disk

PALETTE comes on a single disk (non-bootable) and requires only the minimum operating system (Z-DOS) to work. According to the documentation, it may be transferred and used on any disk format, including 8-inch and Winchester. If you use it on a system with a single floppy disk, you will end up with less than 60K of disk space. However, a number of files can be removed from your working disks, enough to give you well over 100K of free disk space.

Like all good software, PALETTE is being continually updated. Therefore, a README.DOC file which includes all the updates for the particular version you receive is also on the disk. The compression source file and at least one printer driver source file is included on the disk. Including a printer drive source file should give experienced programmers the information needed to interface PALETTE with almost any printer on the market.

Apparently, the majority of the routines used in PALETTE are written in C with machine language routines for direct video interfacing. C is one of the better high level languages for speed. Some versions are even available now that support the 8087 co-processor for the Z-100.

Disk storage space for displays is always at a premium, a typical 5-1/4-inch floppy (data) disk will handle about five or six raw (unprocessed), full-color screen images. Dale Wilson, the creator of



PALETTE, got around this problem by using a compression technique that will allow around twenty images on the same-sized disk.

### Running the Program

Running PALETTE is easy. There are three ways to start the program: with a clear screen, with a file read in from disk, and without clearing the screen. Once PALETTE is running, a help file can be called up from the keyboard and even though the help message overwrites the current screen, its contents are saved for redisplaying after you have read the desired help screen(s).

The cursor is represented by a set of cross hairs, which allow accurate pinpointing of a single pixel or center of a wider "brush stroke".

Movement of the cursor is accomplished, not through the arrow keys, but through the keypad, which is a logical (although not marked) arrangement for directing movement. In addition to single step movements, the automatic repeat and fast repeat features can be used. However, you don't have to worry about getting in too many keystrokes from the typeahead feature; PALETTE controls the keyboard and does not allow typeahead.

Even with the lack of typeahead, PALETTE is **not** slow. Movement of the cursor (brush) is good. If you don't like to hold keys, PALETTE also features continuous movement. Here, the keypad is used in conjunction with the shift key to direct movement. Stepping for both manual and continuous movement is controllable, so you aren't tied to continuous lines or brush strokes. The step spacing is entered through the normal number keys associated with the main typewriter keyboard of the Z-100. Also, by using the shift key in conjunction with these keys, the numeric values are added, resulting in step rates of virtually any size (though I can't see any use for values much above twenty).

PALETTE uses the 5 key (keypad) to control whether the brush paints (writes) or doesn't; the 0 (zero) key is used to stop or pause the continuous motion; and the period key turns either the cursor display (or status line) on or off. The arrow keys are disabled and the home key returns to cursor to a center (or predetermined) point on the screen. Incidentally, this status line is updated each time the cursor is moved, so turning off the line will speed up continuous movement, whether controlled by the computer or manually entered by holding down the movement key (in conjunction with fast repeat).

PALETTE features a number of commands entered through the keyboard. With one minor exception, all the commands are logical. A defines an area on the screen, B selects brush shape (D for dot, E for ellipse, R for rectangle, L for line), C selects color and type of paint (cover, wash, stain, reverse), D calls various disk input/output functions (R for read, W for write, D for displaying the directory, and E for erase a file), F is used for filling the screen or area with a color, G is used for drawing geometric shapes (E for ellipse, R for rectangle, L for "rubber band" lines), H is used to set the home position, K is the one exception -- it is used for copying one area of the screen to another (since C is used to set the color, K was the only logical choice left), L is for light pen use, M is used to move one area of the screen to another (the original is erased), N clears the screen for a new picture, Q allows the users to quit and return to the operating system, R is the rotate command (L for left, R for right, F for flip, and U for rotate upside down), S is used for selecting a different working screen (S for swap, K for copy -- consistent with the copy command, W for write, and R for read), T is used to enter the text mode, V is used for "low usage" commands (E sets to erase color, I turns video interface on and off), and M is used to preview a move.

Obviously, some of the single keystroke commands can get you into trouble; PALETTE comes back on these commands with an "are you

sure" type of message to make sure that you really meant to issue a particular command. In addition, if you start into a command, but don't want to continue, you may use the escape key to cancel (abort) the command.

Defining an area of the screen has its advantages. If you are working on a very complex design and don't want to accidentally destroy any previously created image, you may define a working area of the screen. Once you do this, you can't damage or change anything outside that area, even though you can move the cursor. Well, almost; the move and copy commands will let you move this pre-defined area. One word of caution here -- copy and move do not ask "are you sure" before executing the command. However, you do have the VM command which lets you preview the move (or copy) before you execute the command permanently.

The brush shape is fully controllable, except in simulating the sweeping movement where the brush is gradually lifted from the canvas as it is moved; brush up and down is very abrupt. I really don't know if one could control gradual lifting of the brush while the cursor is moving.

Both ellipses and rectangles can be fully adjusted for height and width and the resulting center can be left empty or filled in. The line shape is the most versatile, offering something that even a traditional paint brush can't do: broken lines. Up to fifty points can be defined as solid or transparent, resulting in the ability to paint a picket (without the points) fence in one stroke.

Any two of the computer's eight colors can be mixed and you can dictate whether the color is to completely cover the background (what is already there), stain it so that the background is partially filtered out (somewhat like staining wood, the original color and design remains, but is altered by the color of the stain), washed (the color is added to the background), or reverses the background colors.

If two colors are mixed, they produce vertical strips. While this is perhaps the easiest to do (from a programmer's standpoint), it does not always produce the desired results.

The disk functions, read, write, and directory, are expected in a program such as this. The function to erase files was an unexpected bonus; you can erase one or more files from any disk installed in the system. The standard wild cards (\* and ?) work for both the directory and erase functions. The directory is displayed in a slightly modified form, showing only the filename, the size in bytes and the date created or last modified.

You may swap disks (remove the PALETTE disk) anytime except when you need to use the help key or after you have activated but not completed the D function. This allows single-disk drive owners to have access to almost the full power of PALETTE.

Occasionally you'll want to drop a shape on the screen without having to change the brush shape, particularly if the shape is complex. The draw geometric shapes command allows you to draw an ellipse (modified circle), rectangle, or line on the screen. The line is interesting because it is a "rubber band" line. Once the first end-point is set, you may use the movement keys to move the other end to any point on the screen. While you are doing this, a line is displayed, running from the first end-point to the cursor. This line stretches, contracts, and moves as you move the cursor around the screen, just as if you were holding one end of a rubber band at a fixed point and were moving the other end around; hence the name, rubber band.

I mentioned earlier that the home key returned the cursor to the center of the screen or a predetermined point. The H command, set



home position, allows the user to move the home position to any point on the screen. This can be very useful when you want to work in just one area of the screen and want to establish a starting point different than the center of the display.

The light pen has an unusual, but needed, feature. It allows you to select the "flash" color. Light pens and displays must be matched to work correctly. The sensitivity of the light pen varies from color to color, depending upon the phosphors used in the display unit's CRT. In some systems, all colors are used, producing a momentary white screen. However, if the monitor is overdriven, it is possible that the picture will distort, resulting in wrong point recognition. In these cases, it would be ideal if you could select a color other than white. With PALETTE, the calibrate section of the light pen command allows you to select a flash color to which your pen will respond. In addition, you also can adjust the sensitivity of the light pen. This results in very reliable light pen operation.

The rotate commands allow you to turn an area of the screen ninety degrees to the left or right, or flip the image left to right or up to down. Obviously, a square area works best (although, due to screen ratio, it really isn't a true square). I would like to see partial turns, with increments fully definable from 1 to 359 degrees. Maybe future versions of PALETTE will feature this.

Because the Z-100 has a large amount of video memory. It is possible to have two complete "pages" of video in memory at one time. Even ZBASIC can access these two pages through PEEK and POKE commands. PALETTE is no exception and features a full compliment of read, write, swap and copy commands for performing the functions.

Most pictures require at least some text. PALETTE accommodates this with the text command. Full control over cursor movement is maintained allowing text to be inserted at any point desired by the operator. By defining an area to work in, it is possible to rotate text ninety degrees without having to do much fancy footwork in the process.

The Z-100 is capable of producing a full 1K by 1K display -- true, you have to get in and work with the programmable video ICs and have a monitor that has the capability to display information on this scale. However, on a more practical note, the best we can do with current display devices (monitors) is 640 points by about 480 lines in an "interlace" mode. PALETTE allows the interlace mode to be turned on and off, but only to double the depth (height) of individual pixels, producing a solid picture.

With the ZVM-136, any flicker while in the interlace mode is reduced to an absolute minimum due to the long-persistence tube used in this display. Other displays will produce some flicker in varying degrees, depending upon the phosphors used in the CRT design and the viewer's sensitivity to flickering images (some people can even see, and are bothered by, the flicker in a fluorescent light).

Unfortunately, true interlacing is not supported, something I would like to see in a future version of this software. I would also like to see a variable display supported where the user can increase or decrease any of the screen parameters, including scan rates, width, and height.

### The Documentation

Software documentation can be as important to a program's success as the program itself. You can have the best program in the world, but it won't be worth anything if you don't have good documentation to teach people how to use it. PALETTE is no exception to this rule. Fortunately, PALETTE has some of the best documentation I have

ever seen in a program of this nature. It is complete and yet concise enough to fit on 50 pages.

The excellent 27 page tutorial will take even the most novice of users through the steps needed to get PALETTE running with all its many bells and whistles. However, be aware that the tutorial is straightforward; it won't bore you by repeating every facet of the operation three or more times. Instead, the instructions are clearly given once. At least, you won't end up feeling like an idiot because you needed to use the tutorial.

The reference section repeats all the instructions given in the tutorial. Because the operation of PALETTE is well thought out and structured, the reference section is almost as easy to understand as the tutorial section. For experienced computer users, particularly programmers, I recommend skipping the tutorial and using the reference section to learn PALETTE.

My pet peeve of using the term "invocation" (meaning "to start") applies to this documentation. I don't know why, but like the naming of early unlimited hydroplanes after women (My Sweetie, Miss Slo Mo Shun, Miss Thriftway) because they were unpredictable, maybe, just maybe, people use the term invoke to mean what it says: to call upon the spirits to start or keep the darn thing running. I hope programmers and authors eventually get away from the practice of using invoke and its variations (just as people got away from naming unlimited hydroplanes after women). PALETTE appears to have no problems either starting or running smoothly and consistently.

Some of the shortcomings have already been mentioned: rotate commands that are too coarse, lack of true interlace, and so on. In PALETTE's documentation, I would like to have seen a quick reference guide for the commands in PALETTE. It would add a great deal to the usefulness of the package. I sincerely hope Dale Wilson and Software Wizardry see fit in future editions to include such a one-page command sheet or card.

### The Future

Just a couple of weeks ago, I received a test version of PALETTE. Some of the enhancements that are being tested include animation, a feature that may easily make this one of the very best packages on the marketplace for the H/Z-100. Also included was the ability to mix brush shapes and defining them by name, allowing the user to form libraries (or tool boxes) of brush shapes for use with his palette. Increased disk support now features areas, brushes (mentioned earlier), coordinates (not fully developed for testing), and keystrokes (for saving a sequence of keystrokes and recalling them later). A sketch function was added to the Geometric command allowing quick sketching from the keypad (again, without changing the brush shape or size).

If all these features (and several others) are incorporated in PALETTE (I think they probably will), along with some of those I proposed in this article, this package will end up being a real masterpiece. As it now stands, PALETTE is the best on the market for doing artwork on the computer. It is fast, easy to learn and use, and does an excellent job of compressing the picture information for compact disk storage.

Obviously, I have not covered everything PALETTE can do here. To fully understand and appreciate the power of this package, contact the vendor for more information.

<b>Vendor:</b>	Software Wizardry 122 Yankee Drive St. Charles, MO 63301 (314) 946-1968	<b>Price:</b>	\$69.95
		<b>Vers. 2</b>	\$89.95



# HUG NEW PRODUCTS

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



885-1237[-37]

**CP/M UTILITIES ..... \$20.00**

**Introduction:** This disk contains a new collection of utilities for users of 8-bit CP/M on H8, H/Z-80,90, and H/Z-100 computers.

**Requirements:** Any Heath/Zenith CP/M compatible computer, any version of 8-bit CP/M 2.x, and at least 32K of RAM. An H/Z-19 or compatible terminal is required for use on an H8 computer.

This disk contains the following files:

README	.DOC	SCRNCLK	.ASM
SEE	.COM	CLOCK	.COM
SEE	.ASM	CLOCK	.ASM
SCR89	.COM	CLKDEMO	.BAS
SCRH85	.COM	SWAP	.COM
SCR100	.COM	SWAP	.ASM
SCR	.ASM	COLOR	.COM
UNSCR	.COM	COLOR	.ASM
UNSCR	.ASM	IBM	.CHR
SCRNCLK	.COM		

#### Authors:

SWAP -- Mark Aagenas, Zenith Data Systems  
COLOR -- from a Z-DOS program by Bob Metz  
All other programs by P. Swayne, HUG

SEE -- This program is a fancy replacement for the CP/M TYPE command. It allows you to scroll forward or backward through a file, to search for strings within the file, and to print screens of the file on a printer. It can also scroll sideways to allow you to view files up to 132 characters wide.

SCR89, SCRH85, SCR100 -- SCR is a textual (not graphic) screen dump program. While it is loaded, any text on the screen is dumped to your printer when you type control-D. SCR becomes part of the system when it is loaded so that any program can be run, and screens of it "dumped". SCR89 is for use on H/Z-89,90 computers, and on H8 computers that use the H8-4 serial card. SCRH85 is for H8's with the H8-5 card. SCR100 is for H/Z-100's.

UNSCR -- This program unloads SCR from your system.

SCRNCLK -- This program maintains a real time clock that can be optionally displayed as a digital clock on your computer screen, even while you are running other programs. It is similar to the screen clock presented in the November 1981 issue of REMark except that it requires no BIOS modifications, and the clock is maintained while the screen display is off. The time maintained by this program can be accessed by other programs. This program will not run on H/Z-100 computers.

CLOCK -- This program is used to control the SCRNCLK program. It can enable or disable the screen clock display, set the time, display the time (useful when the screen clock is off), or unload the SCRNCLK program from your system.

CLKDEMO.BAS -- This is a Microsoft BASIC program that demonstrates how the time can be accessed from SCRNCLK.

SWAP -- This is the console swapper program presented in REMark that allows you to use an external terminal on your computer, and to switch back and forth between the normal keyboard/screen and the other terminal.

COLOR -- This program is for H/Z-100's only, and lets you set the foreground and background colors on your screen.

IBM.CHR -- This is a replacement for the ALTCHAR.SYS file on H/Z-100 CP/M system disks that provides an IBM PC-like character font. The characters are bolder, and some may find them easier to read, especially on color monitors. The normal H/Z-19 graphic characters are supported. For H/Z-100's only.

885-3015-37 ZDOS

**SKYVIEWS ..... \$20.00**

**Introduction:** SKYVIEWS is a Z-BASIC program for plotting the positions on the observers horizon of sun, moon, planets, and selected stars, with constellations marked. Using graphics, the observers horizon is plotted on the Z-100 screen.

**Requirements:** The program requires an H/Z-100 with 192Kb of system RAM, Z-DOS Version 1.25, one disk, and a color monitor, although the displays will work on a monochrome monitor.

The following programs are included on the HUG P/N 885-3015-37 SKYVIEWS disk:

SKYVIEWS	.EXE	BRIGHTST	.BAS
PLANET	.BAS	SKYVIEWS	.BAS
CSTNM	.BAS	SKYVIEWS	.DOC

**Author:** Eugene L. Davis

**Program Content:** SKYVIEWS.EXE is compiled Z-BASIC program for

calculating and plotting the positions of sun, moon, major planets, and stars brighter than 4.0 Magnitude on the observer's horizon and the observer's zenith (overhead). The phases of the moon are shown in the plots. The major constellations in the northern and southern hemispheres can be marked for viewing on the screen. For aid in telescope viewing, the azimuth and elevation angles, as well as the right ascension and declination of the major planets are displayed at the time and for the place selected for the viewer. The data input requirements are simple: date, local standard time, latitude and longitude, and time zone of the observer. The program is menu driven and all data inputs have default values for ease of data entry and demonstration purposes. The algorithms and equations used in this program were taken directly from the ASTRONOMICAL ALMANAC for 1984, and references are commented into the source.

The source program is included and can be executed directly under the ZBASIC interpreter. New default values can be added by creating a new default value file.

**Comments:** Extensive documentation is included with SKYVIEWS making this an ideal program for beginning Astronomers as well as seasoned ones.

**TABLE C Rating:** (0),(7),(9)

### 885-8030-37 ZDOS

**MATHFLASH** ..... \$20.00

**Introduction:** MATHFLASH is an H/Z-100 color educational program designed for use by children who are learning the basic mathematical operations of addition, subtraction, multiplication, and division. It emulates the traditional "flash card" method of learning.

**Requirements:** MATHFLASH requires the Z-DOS operating system on an H/Z-100 computer, 128Kb of RAM, one disk drive, and a video monitor (either color or monochrome, though color is recommended).

The following files are included on the HUG P/N 885-8030-37 MATHFLASH disk:

README .DOC  
MATHFLAS.EXE

**Author:** Mark C. Morrow

**Program Content:** MATHFLASH offers the following features:

- \* MATHFLASH allows complete user control of the colors used for each character on the monitor screen.
- \* MATHFLASH remembers which problems are answered incorrectly and repeats them occasionally, interspersed among other problems, until the child answers them correctly.
- \* MATHFLASH works with true random sequences of exercises which do not repeat; i.e., answers can not be "memorized" by successive program sessions.
- \* MATHFLASH will perform any user-selected combination of addition, subtraction, multiplication, and/or division. For example, "+" and "/" may be selected for drill, excluding "-" and "X"; "+" and "-" may be selected, excluding "X" and "/"; "+" may be selected, excluding the other three operations, etc.
- \* MATHFLASH allows you to specify the operands used for specific operations. For example, the "+2's" may be requested, or the "X7's"

and "/7's" interspersed, or any other conceivable combination of operands (0,1,2,...,9) and operators (+,-,X,/).

\* MATHFLASH entertains children while they practice in the areas you have specified.

\* MATHFLASH statistics are available at any time in the program showing:

- 1) Number of problems worked.
- 2) Percent of total answered correctly.
- 3) Average elapsed time required by child to answer the problems.
- 4) Exactly which problems were missed and have not yet been answered correctly. This feature allows you to review your child's progress and to emphasize the areas he/she may be having difficulty in.

These statistics are saved on disk for review after the program has finished.

**Comments:** From all appearances, this program is child-proof, so to speak. Wrong entries will do no harm to the program itself or its data file. Typing CTL-C is the only way to exit the program. An excellent child education addition to your software library.

**TABLE C Rating:** (0),(1),(9)

### 885-1236[-37] CP/M

**MBASIC FUN DISK I** ..... \$20.00

**Introduction:** This first MBASIC FUN disk contains five games. Two of these games are adventure type games and the other three are mind-teaser types. All four games use H/Z-89 graphics to a small degree. The adventure games, ESCAPE and DRACULA'S CASTLE, are slightly different types of adventures than you may be used to. Commands are given with two letters such as: 'GW' for Get Water. DRACULA'S CASTLE is in real time and you must be clever or the king of canines will get you. CONCENTRATION uses H/Z-89 type graphic characters and plays similar to the ever popular TV game show. MASTER-MIND also uses H/Z-89 graphic characters instead of pegs to play this old favorite. Finally, MERLIN has you remembering sequences of numbers in their proper order after being flashed on the screen momentarily.

**Requirements:** This MBASIC FUN DISK requires the CP/M operating system on either an H8 (with an H/Z-19 terminal), H/Z-89, or H/Z-100. 64K of RAM and MBASIC Ver 5.2x are also needed. Only one drive is needed for any of these games.

The following programs and files are included on the HUG P/N 885-1236[-37] CP/M MBASIC FUN DISK I:

README .DOC  
DRACULA .GAM  
MERLIN .BAS  
ESCAPE .BAS  
CONCEN .BAS  
MERLIN .DAT  
DRACULA .BAS  
MASTMIND.BAS

**Author:** Iraj Aidun

ESCAPE -- This is an adventure program that does not run in real time mode so you can take your time in solving it. The object is to escape from an abandoned house... alive!

DRACULA'S CASTLE -- This is a highly advanced adventure game



which, unlike ESCAPE, operates in real time mode. This game is not recommended for people who have a shortage of time or temperament. If it wasn't for the prisons, slaves, guards, surprises, and blood-sucking DRACULA, this game would be easier!

**CONCENTRATION** -- Up to four people can play this game at once. The object is to beat your opponents by matching the most pairs of graphic symbols on the grid provided.

**MASTER-MIND** -- The object of this game is to see who is the real master of your mind.. you or your computer. The game is played with graphic symbols instead of colors. The object of the game is to guess the symbols of the computer in their correct order, in as few turns as possible.

**MERLIN** -- This game puts up a picture of the computer's numeric keypad and then flashes sequences of numbers on that pad. You then must duplicate each sequence. At first the game is easy, but watch out, a photographic memory becomes a must.

**Comments:** None

**TABLE C Rating:** (0),(1),(2),(3),(7),(9)

# Local HUG Club News

Korea, Yongsan, Seoul

## K-HUG

C/O LTC Wm. Dismuker  
HHC 1st SIG BDE  
APO San Francisco, CA 96301  
293-7750 office  
293-4132 home  
Group Size: 6  
Monthly meetings not established yet.

## YOU-HUG

Madesa Compu-Systems Inc.  
1502 S. Raccoon Ave.  
Youngstown, OH 44515  
Contact: Mario DeSantis  
216-799-5028  
Starting new Users' Group

France, Paris

## GUFIH (Groupe des Utilisateurs Francophones d'informatique Heath- Zenith)

37, Boulevard Saint-Jacques  
75014 Paris

Contact: Dr. Bernard Pidoux  
(1)-565-10-11  
Group Size: 350  
Meet every Wed. at 7 p.m. at above address  
Computerized BB (1)-565-10-09  
Monthly Newsletter

To Heath/Zenith Users' of Waterloo-Cedar Falls area of Iowa. Subject: Possibility of starting a local HUG Club. If anybody is interested, please contact me at the address below:

Bob Brownlee  
1129 Lantern Square Apt. 1  
Waterloo, IA 50701  
Phone: 319-291-7220 after 4 p.m.

CO, Lowry AFB  
**ZUG (Zenith Users' Group)**  
AFAFC/MPA  
Building 444  
Lowry AFB, Denver, Colorado 80279  
Contact: S. L. Brantley  
303-370-7153

Group Size: 75  
Meet 2nd Wed., Gilchrist Building

**CCCC (Champaign County Computer Club)**  
1004 Kinch  
Urbana, IL 61801  
Contact: Jim Mullen  
217-344-2178  
Group Size: 110  
Meet 1st Wed. 7:30 p.m. at Urbana Civic Center  
Have 24 hr. BB 217-359-9090

# 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	Volume - Issue
<b>HDOS</b>			
885-1030[-37]	Disk III, Games II .....	\$ 18.00	5-2
885-1096[-37]	MBASIC Action Games .....	\$ 20.00	5-2
885-8026	Space Drop .....	\$ 16.00	5-2
885-8027	HDOS SCICALC .....	\$ 20.00	5-3
<b>CP/M</b>			
885-1234[-37]	CP/M Ham Help .....	\$ 16.00	5-2
885-1235-37	CP/M RDZDOS .....	\$ 20.00	5-7
885-5001-37	CP/M-86 KEYMAP .....	\$ 20.00	5-4
885-5002-37	CP/M-86 HUG Editor .....	\$ 20.00	5-5
885-5003-37	CP/M-86 Utilities by PS: .....	\$ 20.00	5-7
885-8025-37	CP/M 85/86 FAST EDDY .....	\$ 20.00	5-2
<b>ZDOS</b>			
885-3009-37	ZBASIC Dungeons & Dragons .....	\$ 20.00	5-3
885-3010-37	ZDOS KEYMAP .....	\$ 20.00	5-4
885-3011-37	ZDOS ZBASIC Games Disk .....	\$ 20.00	5-5
885-3012-37 ‡	ZDOS HUG Editor .....	\$ 20.00	5-5
885-3013-37	ZDOS Checkbook Manager .....	\$ 20.00	5-7
885-3014-37 ‡	ZDOS MSDOS Utilities II .....	\$ 20.00	5-7
885-8028-37	ZDOS SCICALC .....	\$ 20.00	5-2
885-8029-37	ZDOS FAST EDDY .....	\$ 20.00	5-6
<b>MISCELLANEOUS</b>			
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	
885-4004	REMark Volume 4, Issues 36-47 .	\$ 20.00	
885-4700	HUG Bulletin Board Handbook ....	\$ 5.00	5-2
<b>NOTE:</b> The [-37] means the product is available in hard sector or soft sector. Remember, when ordering the soft sector format, you must include the "-37" after the part number; e.g. 885-1223-37.			
<b>Products marked with (‡) will also run on the H/Z-150 PC series computers.</b>			





should find the "worksheet cursor" that indicates the active or current block or cell "highlighted" in the upper left corner of the blank worksheet. This block or cell will be called "A1" (or similar, depending on the software you are using). Remember the roadmap in Part 1! The nominal size of this cell will be one (1) row high and 8, 9, or 10 digits (characters) wide depending on what spreadsheet software you are using. You will be entering data into this block or cell. Only one cell is current at any moment (The size can be varied as we will discuss later.). This Active cell is the only one immediately available for use! When you enter data from the keyboard, it goes to this Active cell. The row and column that contain the Active cell are called the current row and current column. Again, the Active cell is ALWAYS indicated on the worksheet by this "highlighted" worksheet cursor. The computer cursor (blinking underline, blinking block, etc.) can be at another part of the screen.

You can easily MOVE the worksheet cursor to any location that you specify for the Active cell, sometimes it can be directed automatically. Also, remember that the worksheet can be much larger than your screen area. Your screen acts as a "display window" through which you see part of the worksheet and manipulate its contents. As you enter data and the worksheet begins to fill up, you can move (or it will move for you) the window as necessary to keep the Active cell in view. We will discuss this in detail as we need this. Our current P&L assignment will fit on the one "window".

I do not want to use the name "worksheet cursor" all through this series. How about some alternate names? I just thought of a couple of these--"reverse-video-bar" (Isn't that bad?), "cell pointer", or how about just "pointer"? That should do it for now! How are we going to move this pointer? The keys that do this vary a little from computer-to-computer and from software-to-software. We will use the "Arrow-Keys" and the "Home-Key". Now, try moving the cell pointer around on the screen, push the right arrow key twice and the down arrow key once. Where are you? Can you name the Cell? Is it at "C2"? Try moving the pointer yourself until you feel at ease with it and you can name the cell you are at. Don't worry about the edges of the display window. If you try to move the pointer above row "1" or left of column "A", the computer will beep at you to say "you can't do that"! You can move the pointer below or right of this window and you will find that the window will move. This is called scrolling. The window can scroll up-or-down, left-or-right. We will discuss this in detail later. For now just press the Home key to get back to where we started.

By now, I am sure that you have noticed that the screen has a "work area" either at the top or the bottom of the screen, depending on which software you are using. If you watched what was happening as you moved the pointer, you would have noticed that the cell location name was always shown in this area. Did you see it? Was your cell name the same as the one given? Try moving the pointer again if you did not follow this. Now, press the Home key again and the worksheet cursor should be back in the upper left hand corner of the worksheet. Do you agree? There are other methods of moving the pointer that we will discuss as we try them.

There are three (3) basic ways to make an entry into the Active cell:

- 1) Type it on the keyboard and press Return.
- 2) COPY or MOVE an existing entry into another cell.
- 3) RETRIEVE data from a disk file, one created by you using your software or one created by someone else with another program.

We will start with Number 1, typing entries. To type an entry into a particular cell as you have shown on your Preparation Form, just MOVE the pointer using the arrow keys to that cell and type the entry

and ending with Return. As you type, each character will appear on a certain line of your work panel area at the top or bottom of the screen as we have discussed before. As with most computer programs, a blinking cursor will indicate where the next character that you will type will appear. This may seem unnecessary, but it is important. You will need to move this cursor when we use the "EDIT MODE". We will discuss this when we use it. As with any other keyboard entry, you can make typing error corrections by using Backspace and re-typing an entry until you press Return! When you press Return, the software causes the entry to be stored into the current cell. If something was already stored in that cell, the New entry replaces it! There is no way to recover the old contents of the cell! (Moral: Look and check before you press Return!)

There are four (4) types of entries that you can enter. (Each software may require a little different way to inform the software what you want.) I will start with the LOTUS 1-2-3 Software method:

1) Numbers. If you start an entry with any of these characters, 1-2-3 will know you are entering either a number or a formula:

0 1 2 3 4 5 6 7 8 9 + - . ( @ # \$

It is easy to see why 1-2-3 would assume the first few characters would be a numeric entry. They are the basic numbers and simple arithmetic expressions. The characters @ and # could be used to form formulas. The first key you type determines the entry type (The header "READY" changes to "VALUES" in the work panel.). Here are the 1-2-3 rules for entering numbers:

- a) A number may begin with a digit (0..9), a plus (+) or a minus (-) sign, a period (decimal point), or (\$) sign.
- b) A number may end with % (indicate percent). It divides the preceding number by 100.
- c) A number cannot have more than one period (decimal point).
- d) You cannot use any commas or spaces when you enter a number! We will see how 1-2-3 can format numbers with a comma later.
- e) You can end the number with a power-of-10 scaling factor (Called the scientific format, more about this later).
- f) There are some special numbers that we will discuss later.

2) Labels. If the first character that you type is not one of the number types from above, 1-2-3 decides that you are entering a label. It shows its decision by changing the "READY" indication to "LABEL" on the work panel portion of the screen. (Exceptions: The slash "/" starts a 1-2-3 command. Function keys (F1...F10) invoke 1-2-3 functions.) A few keyboard characters are special to LABELS. They tell 1-2-3 how to display the LABEL in the cell--left aligned, right aligned, centered or repeatedly:

' (apostrophe) -- left aligned  
" (double quotes) -- right aligned  
↑ (carat) -- centered  
\ (backslash) -- repeating

I can hear you asking, what if the Label starts with a number like 15 B Street? Answer, start your label with one of the above label prefix characters:

"15 B Street

The prefix character will not appear in the displayed cell!

3) Value vs Label. 1-2-3 must distinguish between a Value and a Label. A Value is an instruction that can be used for a calculation. A Label is a string of characters to be displayed.

4) Formulas. The first character that you enter always determines

what you are entering. Thus, a formula like a number must begin with one of the characters in (1) above. But, if this is the case, starting a formula with the letter "A", for instance, would have 1-2-3 decide that a LABEL is being entered. So, I will nearly always begin formulas with the "+" character. Our formulas will look like the following:

- a) +1640-510+1102-1202 and "Return".
- b) +A1+C2-A4+B3 and "Return".

Formulas are like numbers, they may not contain any space characters. Also, as you type the characters from the keyboard, they appear on the control panel portion of the screen as described before.

One of the advantages of 1-2-3 and most spreadsheet software is the ease with which it will allow you to make changes to your entry. You can always change the entry while you are in the process of typing it by backspacing and retyping, if you have not typed Return. You can go back later, either to revise or replace it. We will explore various methods as we use them. One of the Cardinal Principles that I always tell my trainees is "The more characters that you type, the greater your chance of making an error"! We will examine many ways to reduce the number of entries that we will have to enter into our spreadsheets. And, one of the main questions asked by all trainees is "what do I do next?" Recognizing this, most spreadsheet programs provide many MENUS and HELP supports! Be sure you know how to use both with your software!

There are two (2) ways to select from a MENU, pointing and typing:

- 1) When 1-2-3 displays a Menu after you type a "/", it changes the Mode indicator to "MENU". You can now MOVE the "menu pointer" by means of the arrow keys to highlight the selection you want and then type "Return" to select the item from the Menu items.
- 2) As an alternative, you can make your choice by typing the first letter of the Menu item. (Note: When creating a new filename, range name or graph name you must type out the complete name.)

The 1-2-3 "HELP" facility is nearly always available by pressing the F1 (HELP) key. You will find that HELP explains itself very clearly on how to use HELP. Try HELP many times! After getting HELP, the session continues exactly where you left off! Type the "ESC" key to end the HELP screen.

Now, while you are at the computer keyboard, with the spreadsheet blank worksheet in front of you on the screen, we will start to enter our "model"! (BOY! I didn't think we would ever get started. Did you?) Here is the step-by-step procedure I will use this time (using 1-2-3):

- 1) With 1-2-3, have the CAP lock and SCROLL "F12" lock keys OFF! Try them, on and off, and watch your work panel. Did you see the display? Do you have the pointer at "HOME"? Do you see "A1" in the upper, left hand corner and "READY" in the upper right hand corner? You should, and if you do, lets go to the next step.
- 2) To move the pointer to cell C1, press the right arrow key twice.
- 3) Type ↑, P, space, &, space, L and press Return.
- 4) Move the pointer to B3 by pressing the left arrow once and the down arrow twice.
- 5) Type ↑, JAN and press the right arrow (NOT RETURN)! (Note: Did you see the pointer automatically move where we wanted it to go?)
- 6) Type ↑, FEB and press the right arrow.
- 7) Type ↑, MAR and press the right arrow.

8) Type ↑, YTD and press Return.

9) Press the GOTO (F5) key, type A4 and press Return. (Note: Moving one cell at a time is not the fastest way to move the cell pointer when you have a ways to go. How do you like this method?)

10) Type ', SALES and the DOWN ARROW (NOT Return). (Note: The (') prefix causes the LABEL to be left aligned. If you do not type ('), the LABEL will be left aligned by default, so we can eliminate the (').)

11) Type COST and press the down arrow.

12) Type G, ., P, . and press Return.

13) Press F5, type A8 and press Return.

14) Type G, space, &, space, A and press the DOWN arrow.

15) Type SELLING and press Return.

16) Press F5, type A11 and press Return.

17) Type NET and press the Right Arrow (not Return). (You should now have the pointer at cell "B11". Do you?)

18) Type +, B6, -, B8, -, B9 and press the right arrow. (Note: 1-2-3 will calculate the formula and it will equal a Zero "0" because the referenced cells are blank (Zero to 1-2-3). This is true for the following formulas also.)

19) Type +C6-C8-C9 and press the right arrow.

20) Type +D6-D8-D9 and press the right arrow. (Your pointer should be at cell "E11".)

**Note:** Spreadsheet software has built-in functions that help us with formulas. Our YTD cell "E11" will contain the SUM of the VALUES in cells "B11", "C11", and "D11". We can handle this several ways. This time we will use 1-2-3's SUM function. We tell 1-2-3 that we want this function by typing the "@" character first! So, the 1-2-3 SUM function call looks like this:

@SUM(list)

A 1-2-3 "list" is a "range" of cells like:

A1, A2, A3, A4

We can tell 1-2-3 that we have a "range" for a "list" like this:

(A1..A4) Note: The two (2) periods ".." means "range"!

Thus, for this example our function will look like the following:

@SUM(A1..A4)

This will replace our typing out the following formula:

+A1+A2+A3+A4

Now, isn't SUM easier?

21) Type @SUM(B11..D11) and press Return.

22) Press the F5 (GOTO) key, type B6 and press Return.

23) Type +B4-B5 and press the Right Arrow.

24) Type +C4-C5 and press the Right Arrow.

25) Type +D4-D5 and press the Right Arrow.

26) Type @SUM(B6..D6) and press the Up Arrow.

27) Type @SUM(B5..D5) and press the Up Arrow.

28) Type @SUM(B4..D6) and press Return.



**Note:** We have now entered everything on our Preparation Form; however, we cannot easily check our entries because our window is not displaying the formulas we have entered. So now, we must learn "Worksheet Commands"! Just a few to start. We also must consider "Worksheet Global Commands" (Global means overall). These are best learned by using them over-and-over until they are firm in our mind!

29) Type /WG and press Return, then type T. Did you watch your control panel? Did you see the "MENU" change? This is a good time to try "HELP"! Did you find the "HELP" screens useful? Type "ESC" key and you are back to the worksheet. Now, all the formulas are shown in "TEXT" format! But, did you find that some of the longer formulas have been truncated? This is because 1-2-3 allows for 9 characters per cell by "default". Our formula in cell E11 will require 14 characters. What do we do now? Watch!

30) Type /WC and press Return, type 14 and press Return.

**Note:** Now, carefully compare your "Template" with your Preparation Form. Does it check out? Think of your computer's main memory (RAM) as being your workspace where you have created your "template". Your disks are permanent storage areas where many worksheets (and other data) can be stored, each in their own file. Be sure that you store your creations on disk in a permanent file often!

31) Type /FS and press return.

32) Now type the name that you are going to give your "template". I typed B:P\_LTEMP1 and Return. (I keep my files on the disk in drive B:.)

**Note:** If at any time you want to "QUIT" for a rest or for any other reason, "SAVE" your work as we have just completed, before you leave 1-2-3. You can use a series of names like-- B:P\_LTEMP2, etc.

33) Type /Q and Return.

34) Type Y.

35) Type E.

36) Type Y.

**Note:** To restart your "model", put your disks in their drives A and B as at the beginning of this step-by-step procedure.

37) Type 123 and press Return.

38) Press "any key" when requested.

39) Type /F and press Return.

40) MOVE the pointer to P\_LTEMP1 and press Return. (If you left your worksheet while in the "TEXT" format [formulas printed out], type /WG and Return and type G.)

**Note:** We are now ready to add "DATA" into our "template"! Here is a list of data that we will enter into the blank cells:

```
a--JAN--SALES(cell "B4")-----12456.
b--FEB--SALES(cell "C4")-----13567.
c--MAR--SALES(cell "D4")-----14678.
d--JAN--COST(cell "B5")----- 6234.
e--FEB--COST(cell "C5")----- 6545.
f--MAR--COST(cell "D5")----- 6787.
g--JAN--G&A(cell "B8")----- 1122.
h--FEB--G&A(cell "C8")----- 1234.
i--MAR--G&A(cell "D8")----- 1356.
j--JAN--SELLING(cell "B9")----- 456.
k--FEB--SELLING(cell "C9")----- 567.
l--MAR--SELLING(cell "D9")----- 678.
```

Now, we will enter this data in the blank cells of the "template":

41) Press F5, type B4 and press Return.

42) Type 12456 and press the Right Arrow (NOT Return).

**Note:** The Zeros "0" now disappear and we now start getting VALUES that have been calculated by the formulas using the data that we have started to supply and that the numbers are displayed right aligned by default.

43) Type 13567 and press the Right Arrow.

44) Type 14678 and press Return.

45) Press F5, type B5 and press Return.

46) Type 6234 and press the Right Arrow.

47) Type 6545 and press the Right Arrow.

48) Type 6787 and press Return.

49) Press F5, type B8 and press Return.

50) Type 1122 and press the Right Arrow.

51) Type 1234 and press the Right Arrow.

52) Type 1356 and press Return.

53) Press F5, type B9 and press Return.

54) Type 456 and press the Right Arrow.

55) Type 567 and press the Right Arrow.

56) Type 678 and press Return.

**Note:** We have now completed our P&L statement with the first quarter data. It does not look like much of a Financial Report, does it? Lets start to "dress" it up! This might be a good time to "SAVE" your work again. Can you do it without the following instructions? After you do it a few times you will remember.

57) Type /FS and press Return (accept the same file name.).

58) Type R (Replace the file.).

59) Type /WG and press Return.

60) Type C and press Return (You are accepting 2.)

**Note:** We now have our "model" in "currency" format. I think that did a lot for the looks of our report. Do you agree? Now, lets do something to improve our LABELS. Did you try HELP?

61) Press F5, type C1 and press Return.

62) Type ↑, PROFIT AND LOSS and press Return.

63) Press F5, type B3 and press Return.

64) Type ↑JANUARY and press the Right Arrow.

(**Note:** Steps 62 and 64 are ways to make changes to LABELS by over-writing them! Here are some more.)

65) Type ↑FEBRUARY and press the Right Arrow.

66) Type ↑MARCH and press the Right Arrow.

67) Type ↑YEAR TO DATE and press Return.

68) Press F5, type A4 and press Return.

69) Type TOTAL SALES and press Return.

70) Type TOTAL COSTS and press Return.

71) Type GROSS PROFIT and press Return.



- 72) Press F5, type A8 and press Return.
- 73) Type G&A EXPENSES and press the Down Arrow.
- 74) Type SELLING COSTS and press Return.
- 75) Press F5, type A11 and press Return.
- 76) Type NET PROFIT and press Return.
- Note:** That is looking better! What are we missing? What year is this for? What is the name of the business? That is a start, but lets "SAVE" again (safety first!).
- 77) Type /FS and press Return (use same file name).
- 78) Type R (over-write the disk file).
- 79) Press F5, type A3 and press Return.
- 80) Type ↑1984 and press Return.
- 81) Type /WIR and A4..E4 and Return (Use "HELP" for information!)
- 82) Type /WIR and A1..E1 and Return.
- 83) Press F5, type B1 and press Return.
- 84) Type S P R E A D S and press the Right Arrow.
- 85) Type H E E T C O R and press the Right Arrow.
- 86) Type N E R C O. and press Return.
- 87) Press F5, type B12 and press Return.
- 88) Type \ (backslash), - and Return (use "HELP" for "backslash").
- 89) Type /C and Return (accept B12), type C12..E12 and Return.
- 90) Press F5, type A5 and press Return.
- 91) Type \\* and press Return.
- 92) Type /C and press Return, type B5..E5 and press Return.
- 93) Press F5, type B14 and press Return.
- 94) Type \= and press Return.
- 95) Type /C and press Return, Type C14..E14 and press Return.

**Note:** I think that we have turned our "model" into a rather nice Financial Statement. We will do other things in future "models". Do you like it? If you would like to try some of your ideas, I would "SAVE" it first. Then, go ahead and try your improvements.

- 96) Type /FS and press Return.
- 97) Type R.
- 98) Type /WGFT and press Return.

**Note:** We have added extra ROWS but our formulas have been automatically adjusted to their new cell locations by 1-2-3 and the "DATA" that we entered into the blank cells has also been relocated to their new correct cells. Pretty handy?

- 99) Type /WGFG and press Return.
- 100) Press F5, type B6 and press Return.
- 101) Type 22456 and press Return.

**Note:** Did you see the "model" re-calculate all the formulas automatically and almost so fast that you could not see it happen?! This is the way that you can do a "WHAT IF" with 1-2-3. Try some other changes yourself. You cannot do any damage, we SAVED our as-

signment! When you get finished, do as follows:

- 102) Type /QY and press Return.
- 103) Type E and Y.

**Note:** We have completed our first 1-2-3 assignment -- P&L statement! Did you find it hard? It looks worse than what it is. Right? Remove both disks and turn off your computer.

### Closing

Now, please try doing this "model" without the step-by-step instructions using only your "Spreadsheet Preparation Form" and the "HELP" screens. If you must, refer to the above instructions, but keep repeating your attempts of doing the model without the instructions until you can complete it without any help. When you can do that, you are ready to try a new model! I will see you next month and we will go through the step-by-step procedure for this model for both the Multiplan and the PeachCalc/SuperCalc Spreadsheet Software.



## NEW Z-DOS WORDKEY™ SIMPLIFIES WORDSTAR™

Introducing WordKey. WordKey replaces ALL of WordStar's control-key commands with much simpler and faster keypad and function-key commands. Each function and keypad of your H/Z-100 is used two or three times over to define more than 90 key combinations. You'll never need to hold down the control key again! Look at these features:

- Keys are logically grouped so that the most often-used commands are single keystrokes. Move around the screen quickly and easily. Lesser used commands are two separate keystrokes that can be entered with one hand.
- Full onscreen help is always instantly available, including two different key diagrams and a separate explanation of each key's function. Just press the HELP key.
- Blank ruler line..Block cursor and key-click on/off...Detailed User's Manual and printed keyboard diagrams.. And much more.
- Use with either WordStar version 3.21 or 3.30.

**Please send me:**

Z-DOS WordKey: \_\_\_\_\_ copies @ \$49.95 \$ \_\_\_\_\_  
 Special: C.Itoh Prowriter graphics screen  
 dump program \_\_\_\_\_ copies @ \$19.95 \$ \_\_\_\_\_  
 Calif. residents please add 6% tax \$ \_\_\_\_\_  
 Total Enclosed (includes mailing) \$ \_\_\_\_\_

Name \_\_\_\_\_

Address \_\_\_\_\_

### DeISoft

Gary Deley, 564 Calle Anzuelo, Santa Barbara, CA 93111  
 (805) 967-9566 eves and weekends REM

## H/Z-100 AND 150 PC GRAPHICS SOFTWARE

MICROSERVICES continues to expand its support of Heath/Zenith computers.

For H/Z-100s:

- ZANIMATE.** Make color animation sequences using ZBASIC ..... **\$64.95**
- ZPALETTE.** Draw images in 92 colors using ZBASIC. Improved with three (3) painting modes and graphics generator ..... **\$59.75**
- ZPATTERN.** Test your color monitor ..... **\$24.95**
- Z3D** from **COLORWORKS.** Make three-dimensional objects in color ..... **\$75.00**

For H/Z-100s and 150 PCs:

**SOFTKITS** from **KERN INTERNATIONAL.** We are now carrying a full line of this excellent series of books and disks for graphics and business/scientific applications.

*We are continuing to introduce new products. Write for our catalog.*



**MICROSERVICES**  
P.O. Box 7093  
Menlo Park, CA 94026  
Phone: (415) 851-3414



Include 3% p/h (\$3.00 min.). California add 6 1/2% tax.

## LEARN TO PROGRAM H89/H19 GRAPHICS IN MBASIC

**BASIC GRAPHICS PRIMER** is a computer based course that will teach you fundamental MBASIC programming techniques to control the H19 terminal or H89 computer graphics. Seven lessons will teach you many forms of graphic plots, data plotting, multiple sequential and random access files used for graphics, basic figure animation techniques, and simple computer graphics game design. Each lesson consists of simple illustrative program segments with complete explanations of the programming techniques involved, questions to reinforce the lesson material, and one or more example programs. Most H19/H89 terminal commands are covered in the lessons, including use of the special function keys. **BASIC GRAPHICS PRIMER** is a "HOW TO" course with plenty of examples. If you want to learn how to take full advantage of the H19/H89 graphics features in your programs, **BASIC GRAPHICS PRIMER** is a must. For CP/M only. Requires a printer.

**ONLY \$49.95!**

CP/M version for H89, H8, H19 requires CP/M and MBASIC

## NEWLINE SOFTWARE

P.O. Box 402, Littleton, MA 01460 (617) 486-8535

NAME _____	Send me _____ BASIC
STREET _____	GRAPHICS PRIMER
CITY _____	program(s) for CP/M
STATE _____ ZIP _____	at \$49.95 each.
Check one: <input type="checkbox"/> payment enclosed <input type="checkbox"/> send COD (add \$3.00)	
Send order to:	
NEWLINE SOFTWARE, P.O. BOX 402, LITTLETON, MA 01460	
Foreign orders: add \$3.00 Airmail, \$10.00 for non-U.S. checks	

CP/M is a trademark of Digital Research, Inc.  
MBASIC is a trademark of Microsoft, Inc.



## 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 attached to the FDC-H8 without hardware additions.
- Price \$495.00**

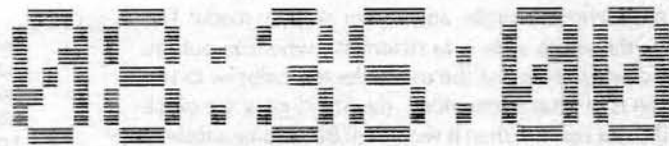
### NEW PRODUCTS FOR THE FDC-880H

- 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

# Clock Watcher's Delight #3



## A Clock for the Z-150, and Updates to the Z-100 Clock

In last May's issue, I presented a screen clock program for H/Z-100 computers that causes a digital clock display to be maintained in the upper right corner of the computer's screen. Since then, I have made some improvements to that program, and I have also written one for the Z-150. I will describe the Z-150 program first, and then present the Z-100 program changes.

Listing 1 is the Z-150 screen clock program source code. To use the program, type in the code from Listing 1 using an editor. You may want to change the color in the line `COLOR EQU WHITE`, which controls the display color of the clock. On a monochrome monitor, a color other than white will show up as a dimmer display, which you may prefer since the clock is less distracting. If you have an IBM PC computer, this program will work on it, but you should set the color to white unless you have a color monitor, because the IBM cannot do "grey scale".

After you have typed in the program, you can assemble it by entering

```
A>MASM B:SCRNCLK.B:SCRNCLK;  
A>LINK B:SCRNCLK.B:SCRNCLK;  
A>ERASE B:SCRNCLK.OBJ  
A>EXE2BIN B:SCRNCLK.B:SCRNCLK.COM  
A>ERASE B:SCRNCLK.EXE
```

This assumes that your source file is called `SCRNCLK.ASM`, and is on drive B:, and that `MASM`, `LINK` and `EXE2BIN` are on drive A:. When you have assembled the program, you will have a file on drive B: called `SCRNCLK.COM`. Copy it to your system drive and enter

```
A>SCRNCLK
```

to start the clock. The clock display should appear in the upper right corner of your screen, and should indicate the time as set by the system `TIME` command.

### Controlling the Clock

Although the time as displayed by the clock program is set to the system time, it is maintained internally (for reasons explained later), and so further uses of the `TIME` command will not change the time on the screen. Also, the program is locked into memory at start up and cannot be removed unless you re-boot the system. The program in Listing 2 overcomes these problems by providing a way to set the clock and to turn it off temporarily. If you type in the listing as `CLOCK.ASM` and assemble it to `CLOCK.COM`, you can turn off the clock by entering

```
A>CLOCK OFF
```

And re-enable it by entering

Pat Swayne  
HUG Software Engineer

```
A>CLOCK
```

The clock will not only be turned back on, but will be re-set to the system time. Therefore, if you want to change the time on the clock, enter a new time with the `TIME` command and run `CLOCK`.

### How It Works

When you run the `SCRNCLK` program, it gets the current time from the system and stores it within itself. Then it sets up the vector for the BIOS-generated timer interrupts so that the interrupts are processed by `SCRNCLK`. `SCRNCLK` then exits via interrupt 27H, which causes the program to remain in memory.

The timer interrupts occur approximately 18.2 times a second. `SCRNCLK` counts them and updates the display after every 18 "ticks". If that was all it did, the clock would run quite fast, and the time displayed would soon be inaccurate. So the program maintains a counter that is incremented each time the display is updated, and after 89 updates, it skips one update. The figure 89 is derived by first dividing .2 into 18, which gives 90. Since the timer interrupts are actually slightly faster than 18.2 times a second, the clock is still slightly fast if you skip an update every 90 seconds, so we skip one every 89 seconds. This value is set up in the program at the line `CALFAC EQU 89`, and you can change it if you want to make the clock run slower or faster. Increase it for a faster clock, or decrease it for a slower clock.

It would be nice if we could access the system time in this program and use it for the screen display, as is done in the Z-100 screen clock program. Unfortunately, there is no way to access the system time from the BIOS on a Z-150 (which is a ROM BIOS, made to work like IBM's BIOS), and using the MS-DOS system call for getting the time will not work because MS-DOS cannot process more than one system call at a time. During any given timer interrupt, the chances are good that a system call is in progress, so another one cannot be

made by a program such as SCRNCCLK. If it attempts to do it, the system will probably crash.

When it is time to update the display, SCRNCCLK converts the time values (hours, minutes, seconds) to ASCII digits. Then it makes a BIOS call to get the current video state. A Z-150 can be in any one of seven video states which are combinations of 40 or 80 column mode, color or monochrome mode, and text or graphic mode. The purpose in testing the video state is to determine where to put the clock display, and whether to use the user selected color or to use white. If the screen is in 40 column mode, the first digit of the clock must start in a different column than it would for 80 column mode in order for the display to be in the upper right hand corner.

When the color and position of the display have been determined, SCRNCCLK saves the current cursor position, writes the time to the screen, and restores the cursor. This is all done using low level BIOS routines that accomplish the task quickly and do not interfere with the operating system or any running program. Those BIOS routines are a kind of mixed blessing. They allow very sophisticated screen control, such as scrolling text up or down within a section of the screen, with text outside the section unaffected. But they are only accessible by a direct call to them. In other words, there are no escape sequences as on a Z-100, and if a high level language you might be working with does not have any extensions to access the BIOS screen routines, you are going to have to do some assembly programming, or do without.

### Improvements to the Z-100 SCRNCCLK Program

Those of you who are using the Z-100 screen clock program from the May issue may have been disturbed by the way it sometimes causes the entire top line of the screen to be distorted while certain programs are running. I have found that if you modify the program so that it does not use the processor's ES register, it will not affect any part of the screen outside of the clock area itself. To modify the program, first locate the label UPDATE and remove the following lines a few lines below it.

```
PUSH    ES
MOV     AX,CS
MOV     ES,AX                ;PUT DS HERE
```

Change the line

```
MOV     OLDSEC,DH
```

to

```
MOV     CS:OLDSEC,DH
```

Similarly, add a CS: to the labels HOUR, MINUTE, SECOND, VPVAL, and SCRPTTR wherever they occur in MOV instructions throughout the program. Find the line

```
MOV     ES,AX                ;POINT ES AT VIDEO RAM
```

and change it to

```
MOV     DS,AX                ;POINT DS AT VIDEO RAM
```

Locate the label CLRLP, and replace the code from there to the line JNZ CLRLP with this code.

```
CLRLP:  PUSH    CX                ;SAVE COUNTER
CLRLPD: MOV     [DI],AL            ;CLEAR A LINE OF VRAM
        INC     DI
        LOOP   CLRLPD
        ADD    DI,80H-9           ;MOVE TO NEXT LINE
        POP    CX                ;RESTORE COUNTER
        DEC    DL
        JNZ   CLRLP              ;LOOP UNTIL 9 LINES CLEAR
```

At the label PTMLP, change AL,[BX] to AL,CS:[BX]. In the line just above that one, change 7FH to 80H. Below the label PDONE, remove the line with POP ES. Finally, at the label PDIGLP, change the code from there to the RET instruction to what is shown here. (And make sure that you didn't forget to put a CS: before SCRPTTR in the line above this label).

```
PDIGLP: DB     00101110B          ;SEG CS
        LODSB                    ;MOVE A BYTE TO VRAM
        MOV    [DI],AL
        ADD    DI,BP              ;UPDATE VRAM POINTER
        LOOP  PDIGLP              ;LOOP UNTIL DONE
        INC   CS:SCRPTTR          ;UPDATE SCREEN POINTER
        RET
```

Now, here's a question for those of you who know more about the MS-DOS assembler (MASM) than I do: Can anyone tell me the proper syntax for specifying a segment over-ride for the LODS instruction? I know that with the CP/M-86 assembler, you can use

```
LODS    CS:AL
```

but that does not work with MASM. I had to use a DB statement, as in the code above.

### Listing 1

```
TITLE  SCREEN CLOCK FOR MS-DOS
PAGE   ,132
SCRNCCLK -- SCREEN CLOCK FOR MS-DOS (Z-150 OR IBM)

THIS PROGRAM PROVIDES A SCREEN CLOCK FOR MS-DOS THAT FUNCTIONS
LIKE THE HDOS AND CP/M SCREEN CLOCK PRESENTED IN REMARK #22,
NOV.. 1981.

THE CLOCK IS TIMED INTERNALLY DUE TO THE INABILITY TO ACCESS THE
SYSTEM TIME FROM THE BIOS. IT CAN BE SET TO THE SYSTEM TIME WITH
THE COMPANION PROGRAM "CLOCK". THE COLOR OF THE DISPLAYED CLOCK
IS DETERMINED BY THE LABEL "COLOR", BELOW. IF YOU ARE IN A B/W
MODE, THE COLOR IS AUTOMATICALLY SET TO WHITE.

BY P. SWAYNE, HUG 22-MAY-84

DEFINITIONS
BLUE EQU 001B
GREEN EQU 010B
CYAN EQU 011B
RED EQU 100B
MAGENTA EQU 101B
YELLOW EQU 110B
WHITE EQU 111B

COLOR EQU WHITE

; COLOR CONTROL BITS
; (IBM CALLS IT BROWN)
; CLOCK COLOR
```



```

ROW EQU 0 ; PUT CLOCK ON TOP ROW
CALFAC EQU 89 ; CALIBRATION FACTOR
SCRINT EQU 10H ; SCREEN CONTROL INTERRUPT
SCP EQU 2 ; SET CURSOR POSITION
GCP EQU 3 ; GET CURSOR POSITION
WCHAR EQU 9 ; WRITE CHARACTER
GVS EQU 15 ; GET VIDEO STATE
TIMEINT EQU 1CH*4 ; Z-150 TIMER INTERRUPT VECTOR

CLK SEGMENT
ASSUME CS:CLK,DS:CLK,ES:CLK,SS:CLK
ORG 100H

START: JMP SETUP ; SET UP CLOCK INTERRUPT, ETC.
;
; CLOCK DATA AREA
CLKFLG DB 1 ; CLOCK ON/OFF FLAG
SEC DB 0 ; SECOND (BINARY)
MIN DB 0 ; MINUTE
HOU DB 0 ; HOUR
TICCNT DB 18 ; TIC COUNTER
SECCNT DB 0 ; SECOND COUNTER (FOR CALIBRATION)
OLDIP DW 0 ; OLD INSTRUCTION POINTER
SYSSTK DW 0 ; SYSTEM STACK
SYSSTKS DW 0 ; SYSTEM STACK SEGMENT
CPOS DW 0 ; CURSOR POSITION
CMODE DB 0 ; CURRENT VIDEO MODE
MODES DB 0,1,2,3,1,0,2,2 ; DECODED MODES
TIMSTR DB ; SPACE BEFORE TIME
HOUR DW 0 ; HOUR (ASCII)
MINUTE DW 0 ; MINUTE
DB ;
DB ;
SECOND DW 0 ; SECOND
DB OFFH ; STACK SPACE
DB 256 DUP (?)
MYSTAK EQU $

; PROCESS CLOCK INTERRUPTS HERE
DB 'CK'
MYTIME: POP CS:OLDIP
PUSH AX
DEC CS:TICCNT
JNZ TIMEINT
MOV CS:TICCNT,18
MOV AL,CS:SECCNT
INC AL
CMP AL,CALFAC
MOV CS:SECCNT,AL
JNZ NOTCAL
MOV CS:SECCNT,0
SHORT TIMEINT
JMP AL,CS:SEC
NOTCAL: MOV AL,60
INC AL
CMP AL,60
MOV CS:SEC,AL

```

```

;SAVE IT
;GET ROW FOR CLOCK
;ASSUME COLUMN 31
;80 COLUMN MODE
;NO
;ELSE, SET COLUMN 71

;SET CURSOR TO TIME AREA
;SET A COUNTER
;ASSUME WHITE COLOR
;40 COL B/W?
;YES
;80 COL B/W?

;ELSE, USE COLOR
;POINT TO TIME STRING
;GET A DIGIT
;SAVE COUNTER
;SAVE POINTER
;PRINT DIGIT

;LOOP UNTIL DONE

;RESTORE CURSOR TO USER'S POSITION

;RESTORE REGISTERS

;RESTORE SYSTEM STACK

;RE-ENABLE CLOCK
;AND EXIT

;CLEAR AH
;GET RADIX
;DIVIDE BY IT
;ADD ASCII

;PRINT A DIGIT ON THE SCREEN

PDIGIT: MOV AH, WCHAR
MOV CX, 1
INT SCRINT
MOV AH, GCP
INT SCRINT
INC DL
MOV AH, SCP
INT SCRINT

;SET NEW CURSOR POSITION

```

```

CPOS,DX
DH,ROW
DL,31
BYTE PTR CMODE,2
PTIME0:
DL,71
AH,SCP
SCRINT
CX,9
BL,WHITE
BYTE PTR CMODE,0
PTIME1:
JZ
CMP BYTE PTR CMODE,2
JZ
MOV BL,COLOR
SI,OFFSET TIMSTR
LODSB
PUSH
CALL
POP
POP
POP
PDONE: MOV AH,SCP
MOV DX,CPOS
INT SCRINT
POP BP
POP DS
POP DI
POP SI
POP DX
POP CX
POP BX
CLI
MOV SS,CS:SYSSTKS
MOV SP,CS:SYSSTK
MOV CS:CLKFLG,1
JMP TIMEXIT

; CONVERT NUMBER IN AL TO ASCII DIGITS IN AL, AH
CONASC: MOV AH,0
MOV BH,10
DIV BH
ADD AX,'00'
RET

; PRINT A DIGIT ON THE SCREEN
PDIGIT: MOV AH,WCHAR
MOV CX,1
INT SCRINT
MOV AH,GCP
INT SCRINT
INC DL
MOV AH,SCP
INT SCRINT

;SET NEW CURSOR POSITION

```

```

RET
;END OF RESIDENT CODE

LEND:
;
;
;
SETUP: MOV AH,2CH
INT 21H
MOV SEC,DH
MOV AH,2CH
INT 21H
CMP SEC,DI
JZ WFSSEC
MOV SEC,DI
MOV WORD PTR MIN,CX
PUSH DS
XOR AX,AX
MOV DS,AX
MOV SI,OFFSET TIMEINT
MOV DI,DWORD PTR [SI]
CMP WORD PTR ES:-2[DI],'KC'
JZ ITSIN
CLI
MOV WORD PTR [SI],OFFSET MYTIME
MOV 2[SI],CS
POP DS
MOV WORD PTR TIMEX+1,DI
MOV WORD PTR TIMEX+3,ES
STI
MOV DX,OFFSET LEND
INT 27H
TIN: INT 20H
CLK ENDS
END START

```

**Listing 2**

```

TITLE SCREEN CLOCK CONTROL PROGRAM
PAGE ,132
THIS PROGRAM ALLOWS YOU TO SET THE SCREEN CLOCK
MAINTAINED BY THE SCRCLK PROGRAM TO THE SYSTEM
TIME IT ALSO ALLOWS YOU TO TURN THE CLOCK OFF
IF YOU SO DESIRE. THIS VERSION IS FOR THE Z-150
OR IBM PC. TO USE THIS PROGRAM, ENTER
A>CLOCK OFF TO TURN THE CLOCK OFF
A>CLOCK TO TURN ON AND SET CLOCK
BY P. SWAYNE, HUG 22-MAY-84
DEFINITIONS
TIMEINT EQU 1CH*4
CMDBUF EQU 80H
CLKFLG EQU 103H
CLOCK SEGMENT

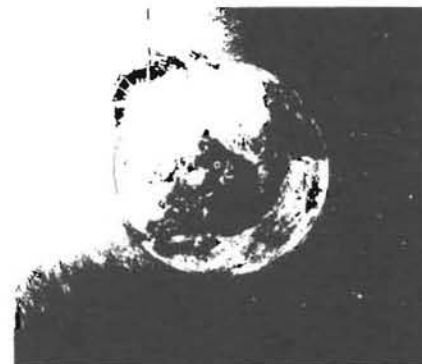
```

```

ASSUME CS:CLOCK,DS:CLOCK,ES:CLOCK,SS:CLOCK
ORG 100H
START:
PUSH DS
XOR AX,AX
MOV DS,AX
SI,OFFSET TIMEINT
DI,DWORD PTR [SI]
WORD PTR ES:-2[DI], 'KC'
JNZ EXIT
;IF NOT, EXIT
;ELSE, RESTORE DS
DS
BX,OFFSET CLKFLG
SI,OFFSET CMDBUF
LDSB
OR AL,AL
JZ CLKON
SOS:
CMP AL, ' '
JZ SOS
CMP AL, '0'
JNZ CLKON
MOV BYTE PTR ES:[BX],0
EXIT:
INT 20H
CLKON:
MOV BYTE PTR ES:[BX],1
MOV AH,2CH
INT 21H
MOV TSEC,DH
MOV AH,2CH
GETTIM:
INT 21H
CMP TSEC,DH
JZ GETTIM
ES:1[BX],DH
ES:2[BX],CX
MOV BYTE PTR ES:4[BX],18
INT 20H
TSEC DB 0
CLOCK ENDS
END START
;SAVE DS
;DS AT 0
;POINT TO TIMER INTERRUPT
;GET VECTOR IN ES:DI
;IS CLOCK INSTALLED?
;IF NOT, EXIT
;ELSE, RESTORE DS
;POINT TO CLOCK FLAG
;POINT TO COMMAND LINE
;GET COUNT BYTE
;ANY ARGUMENT?
;NO, TURN CLOCK ON
;GET NEXT BYTE
;SPACE?
;IF SO, SKIP IT
;OFF?
;NO, TURN CLOCK ON
;KILL CLOCK
;AND EXIT
;ENABLE CLOCK
;GET TIME
;SAVE SECOND
;GET TIME AGAIN
;SAME SECOND?
;IF SO, WAIT FOR NEXT ONE
;ELSE, SAVE SECOND
;SAVE MINUTE, HOUR
;SET TIC COUNTER
;AND EXIT
;TEMPORARY SECOND STORAGE

```

# DISCOVER THE EXCITING WORLD OF HUG GAME SOFTWARE



## (LISP) for H89

Artificial Intelligence Language  
 UO-LISP Programming Environment  
 The Powerful Implementation of LISP  
 for MICRO COMPUTERS  
 Excellent for developing A.I., ROBOTIC, EXPERT  
 and INTELLIGENT SYSTEMS



The usual LISP Interpreter Functions  
 Data Types, Structure Editor,  
 Screen Editor with Mouse Support,  
 Optimizing LISP Source Code  
 Compiler & Assembler, Assembly &  
 LISP Code Intermixing, Compiled  
 Code Library Loader, Numerous Utility  
 Packages, Hardware and Operating  
 System Access, Session Freeze and  
 Restart, Comprehensive 350 page  
 Manual with Usage Examples, and  
 much more is available with the  
 ADVANCED INTERPRETER, your  
 environment may cause constraints,  
 review catalog.

Other UO-LISP products include META a translator writing system,  
 RLISP a high level language, and LISPTX a text formatter.

**!!NEW!! LEARN LISP FOR \$39.95**

The "LEARN LISP" Interpreter includes a subset of the above  
 description, manual includes tutorial.

The UO-LISP Programming Environment runs on the H89 with CPM.

Package Prices range from \$39.95 to \$300.00.

Manuals may be purchased separately.

TO ORDER: First send for your FREE catalog which lists technical details,  
 distribution formats, and ORDER FORMS.

VISA and Mastercard accepted.

**Northwest Computer Algorithms**

P.O. Box 90995, Long Beach, CA 90809 (213) 426-1893

## Graphics For Z-100 AND Z-150 FlexiTek Tektronix 4010 Emulator

Use your Z-100 or Z-150/160 to access mainframe graphics packages. Draws fast.  
 Selectable baud rates to 9600. Complete set-up menu for terminal parameters. Capture  
 graphics to disk. Print out on dot-addressable matrix printer using FlexiPrint (in-  
 cluded FREE, see below) OR send picture to HP plotter. You get TWO versions,  
 one for Z-100, and one for PC-compatibles such as Z-150/160. Z-100 version uses  
 400-line interface mode, with 35 lines by 74 column text, a very close imitation of the  
 4010. The PC version supports only 32 lines of text, and 200 lines res, but otherwise  
 is similar. PLUS you get ALL source code (MASM), so you can see how it's done,  
 modify it if you wish, or use our drawing routines for output from your own pro-  
 grams.

Notes: Some 4010 packages for the Z-100 claim more lines of resolution. We  
 deliberately avoided this since many Z-100s have only 32K video RAMs and cannot  
 support greater than 400 line res. In any case, greatly exceeding 400 lines stretches  
 the ability of your monitor to sync, and may obscure the edges of the picture. Also,  
 some packages advertise more than 35 lines of text. This may make diagram labels  
 come out in the wrong locations. Changing features, and configuration for non-  
 standard graphics boards, is possible with FlexiTek, usually by changing EQUates  
 in our source files. Requires ZDOS/MSDOS/PCDOS 1.25 or above.

Price as of July 1, 1984: \$129.95

## FlexiPrint Screen Printer:

Print out screen graphics on dot-addressable printer. You get .100 AND PC-  
 compatible versions. Activated at any time by Shift or PrtSc, or triggerable by your  
 own programs. You get source code (MASM), so you can make versions for almost  
 any printer by changing EQUates. We pre-configure versions for OKI-92 etc., EP-  
 SONS, CITO, NEC, Apple Imagewriter etc. You can also choose dumps of Z-100  
 hi-res, normal-res, PC screen, color-to-grey translation, or dumping of any arbitrary  
 area of memory. Requires ZDOS/PCDOS/MSDOS 1.25 or above.

Price as of July 1, 1984: \$34.95

For more info send business size SASE. Dealer and local author  
 enquiries welcomed. ORDERING: Please add tax (Cal res only, 6%) first, then  
 shipping and handling (\$3 per item in U.S. or Canada, elsewhere \$12). Check or  
 money order OK, money order only outside U.S. (sorry Canadians).

**Various  
Ware**

VariousWare  
 P.O. Box 21070  
 El Cajon, CA 92021  
 (619) 448-7126

IBM would like you to think of them when you see PCDOS. Likewise Zenith for ZDOS, Z-100.

## The Software Toolworks® presents: THE FUTURE



Languages, games, productivity software, computer cookbooks and more are formatted for IBM PC, PCjr, most CP/M systems and Heath/Zenith HDOS. They range in price from \$19.95 to \$59.95. For a free 41 product catalog contact The Software Toolworks, 15233 Ventura Blvd., Sherman Oaks, CA 91403, (818) 986-4885.

CP/M is a registered trademark of Digital Research.

## HOW MUCH FREE SOFTWARE COULD YOU USE?

FIND OUT WITH OUR GIANT PUBLIC DOMAIN DIRECTORY

- SUPPLIED ON DISK FOR EASY COMPUTER ACCESS
- MORE THAN 4,500 ENTRIES
- SUBJECT AREAS INCLUDE:

ASTRONOMY, AVIATION, BUSINESS, EDUCATION, ENGINEERING, GAMES, GRAPHICS, HAM RADIO, MUSIC, PROGRAMMING, TEXT EDITING, VOICE SYNTHESIS, UTILITIES AND MUCH MORE.

Yes! I need to know what free software is available. Send me the public domain directory on the Heath CP/M 5 1/4 format checked  ON 3 HARD DISK \$19.95  ON 2 SOFT DISKS \$12.95  ON 1 DOUBLE DISK \$9.95

**HEADWARE**  
2865 AKRON STREET  
EAST POINT, GA. 30344

TERMS: NO RISK, MONEY back guarantee. Add \$2 domestic \$4 foreign per order for S&H. Enclose your check or M.O. with your order. Sorry, no charge or phone orders.

Name \_\_\_\_\_

Address \_\_\_\_\_

City, State, Zip \_\_\_\_\_

\*CP/M Reg. TM Digital Research Corp.

Now for  
CP/M too!

## HOME FINANCE SYSTEM VERSION 2

—An extensive Home Finance System that keeps track of checking, asset accounts (cash, savings, IRAs, CDs), and regular bill payments. Let your printer write your checks for you on any business-sized check (design your own check format).

—Checks have user defined codes and a separate flag for tax deductible items.

—Many reports, including listing all checks, or checks by codes or tax flag.

—System consists of 130 page users manual with 5 program disks (5-1/4") and a sample data disk.

Hardware: H8/HZ89 (64K) or HZ100 with 2 disk drives. Any Heath®, Zenith® or other printer.

Software: HDOS 2.0 and MBASIC 4.82 for HDOS, or CP/M or CP/M-85/86 (Ver. 2.2) and MBASIC 5.21 for CP/M.

Order: Complete System \$89† (specify hard or soft sector 5 1/4", HDOS or CP/M, HZ89 or HZ100). Manual alone \$21.†

Master Card/Visa accepted, please include your phone number.

†Prices include shipping.



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

## Double the Speed of your H/Z89-90 Cut computing time by as much as 50%

### Fast Operation

With the Dual Speed Module, software that used to be slow and inefficient now zips right along. Run programs like SuperCalc (in "auto mode") and Spellstar without long waits.

### Easy

Simply replace your existing microprocessor IC with the Dual Speed Module and stop wasting time. Installation requires no trace cutting or soldering.

### Fully Supported

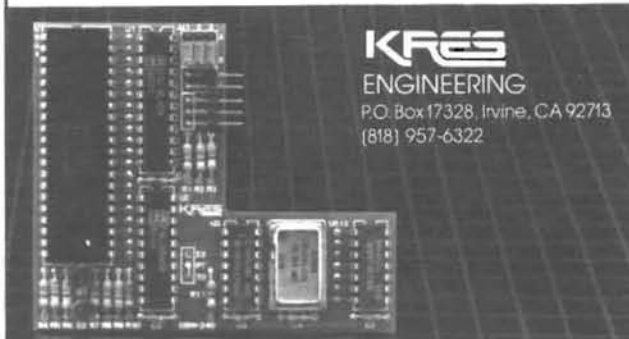
Software support for CP/M, HDOS, and CP/M+ Works with Magnolia Microsystems\*, CDR, and many other disk systems.

Dual Speed Module DSM-240 \$99.97\*\* + \$3.50 shipping and handling (includes CP/M or HDOS, add \$10.00 for both)

\*Requires Software Patch PMM-100 \$49.95 and Monitor ROM KMR-100 \$59.95

\*\*All prices subject to change without notice.

CP/M is a registered trademark of Digital Research



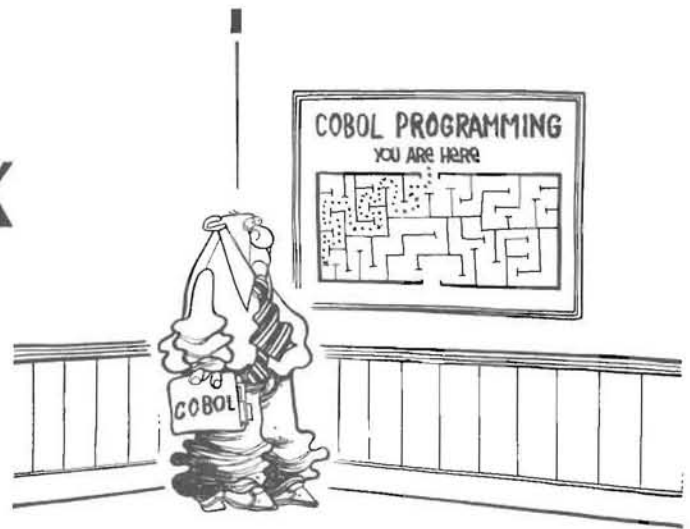
**KRES**  
ENGINEERING

P.O. Box 17328, Irvine, CA 92713  
(818) 957-6322



# COBOL Corner X

H. W. Bauman  
493 Calle Amigo  
San Clemente, CA 92672



## Introduction

Did your Program #3 print out the report as we specified? If not, and you do not believe I have made the program clear, let me know with details of your problems, suggestions, etc. (enclose a business size SASE). Learning a new computer high level language, like COBOL, requires working with it over and over. Program #4 is another program with total-lines similar to Program #3. This time I would like you to do the program (including all PHASES) completely without my hints!

## Program #4 Information

### PROGRAM SPECIFICATIONS

PROGRAM NAME: DISCOUNT REPORT      PROGRAM ID: PRGM04

## Program Description:

This program prints a Discount Report from the Order Record Data file.

## Input File:

FILEL3.DAT

## Output File:

Discount Report as specified below.

## List of Program Operations:

- A. Read each input order record.
- B. For each record, the program should calculate the Discount Amount and the Net Amount.

1. Discount Amount is calculated by multiplying Purchase Amount by Discount Percentage obtained from each input record.

2. Net Amount is calculated by subtracting Discount Amount from Purchase Amount.

C. For each record, print the following fields on the Discount Report detail line in accordance with the format specified for the Output Report Line:

1. Customer Name.
2. Purchase Amount.
3. Discount Amount.
4. Net Amount.

D. Double-space each detail line.

E. After all the input records have been processed, the program should print the following total fields on the Discount Report's two total-lines in accordance with the format specified for the Output Report Line:

1. Total line #1
  - a. Total Number Transactions (computed by counting 1 for each order record).
  - b. Total Purchase Amount (computed by summing each order record Purchase Amount into Accumulator).
  - c. Total Discount Amount (computed by summing each Discount Amount into Accumulator).
  - d. Total Net Amount (computed by summing b. and c. into Accumulator).
  - e. Triple-space Total line #1 down from last detail line.
2. Total line #2
  - a. Average Purchase Amount (computed by dividing Total Purchase Amount by Total Number Transactions and call for Rounded option).
  - b. Average Discount Amount (computed by dividing Total Discount Amount by Total Number Transactions and call for Rounded option).
  - c. Average Net Amount (computed by dividing Total Net Amount by Total Number Transactions and call for Rounded option).
  - d. Double-space Total line #2 down from Total line #1.

F. COBOL will be the programming language.

## Output Report Line Format

PRINT POSITIONS	FIELD NAME	COMMENTS
		DETAIL LINE *****
1-5	Filler	Provide left margin
6-29	Customer Name	
30-32	Filler	
33-44	Purchase Amount	Zero-suppress non-significant dollar position zeros Insert commas & decimal point
45-48	Filler	
49-58	Discount Amount	Zero-suppress non-significant dollar position zeros Insert commas & decimal point
59-62	Filler	
63-74	Net Amount	Zero-suppress non-significant dollar position zeros Insert commas & decimal point

```

75-132 Filler
                TOTAL LINE #1
                *****
1-7 Filler Provide left margin
8-25 Filler Print TOTAL TRANSACTIONS
26 Filler
27-30 Total NBR Transactions Zero-suppress non-significant
zeros
31 Filler
32-44 Total Purchase Amount Zero-suppress non-significant
dollar positions zeros
Insert commas & decimal point
Print an asterisk
45
46 Filler
47-58 Total Discount Amount Zero-suppress non-significant
dollar position zeros
Insert commas & decimal point
Print an asterisk
59
60 Filler
61-74 Total Net Amount Zero-suppress non-significant
dollar position zeros
Insert commas & decimal point
Print an asterisk
75
76-132 Filler

```

```

                TOTAL LINE #2
                *****
1-7 Filler Provide left margin
8-22 Filler Print AVERAGE AMOUNTS
23-32 Filler
33-44 Average Purchase Amount Zero-suppress non-significant
dollar position zeros
Insert commas & decimal point
Print two (2) asterisks
45-46
47-48 Filler
49-58 Average Discount Amount Zero-suppress non-significant
dollar position zeros
Insert commas & decimal point
Print two (2) asterisks
59-60
61 Filler
62-74 Average Purchase Amount Zero-suppress non-significant
dollar position zeros
Insert commas & decimal point
Print two (2) asterisks
75-76
77-132 Filler

```

### Input Record Format

FIELD POSITIONS	FIELD NAME	DATA CLASS	COMMENTS
1-2	Record Code	Alphanumeric	Code "L3"
3-5	Filler		
6-29	Customer Name	Alphanumeric	
30-66	Filler		
67-68	Discount Percentage	Numeric	
69-70	Filler		
71-79	Purchase Amount	Numeric	Assumed decimal point between columns 77 & 78
80	Filler		

**Note:** The Input Transaction File (FILEL3.DAT) is the same one that we used for Program #3. It contains some data that we will not use in Program #4.

### COBOL Verb "Subtract"

The subtract statement subtracts one or more NUMERIC data items from a specified item and stores the difference. The general format is:

```

SUBTRACT {data-name-1}...
        {numeric-literal-1}...
FROM {data-name-m}
     {numeric-literal-m}
[GIVING data-name-n] [ROUNDED] [ERROR-SIZE-clause].

```

The same three (3) options, GIVING, ROUNDED, and ERROR-SIZE, as described in the last COBOL Corner article under MULTIPLY apply to SUBTRACT in the same way. The effect of the subtract statement is to sum the values of all the operands that precede FROM

and subtract that sum from the value of the item following FROM. The result (difference) is stored in data-name-m. If the GIVING option is specified, the result is stored in data-name-n. A couple of examples of the subtract statement follows:

```

SUBTRACT WS-DISC-AMT-WRK
FROM OF-PURCHASE-AMT
GIVING DR-NET-AMOUNT.

```

OR

```

SUBTRACT WS-TOT-DISCOUNT-AMT
FROM WS-TOT-PURCHASE-AMT
GIVING WS-NET-PURCHASE-AMT.

```

### Notice:

The above information should provide you with everything needed for Program #4! You are now on your own. Please Design, Compile and Link/Execute your program before the next COBOL Corner article!

### More On Structured COBOL

There are two (2) basic principles to be followed when coding structured programs:

1. Only three (3) control structures will usually be used:
  - a. Sequence -- usually COBOL statements.
  - b. Iteration (loops) -- PERFORM/UNTIL.
  - c. Selection -- IF statements.
2. Each program MODULE should have only one entry and one exit point.

Since the GO TO statement was not represented, it generally should NOT be used in a structured program. However, due to COBOL syntax requirements, there are a few situations where the GO TO statement might be required--Internal SORT. If GO TO statements are not used and PERFORM statements naming only a single procedure-name are used, a MODULE will have only a single entry and exit point. That is, each MODULE will be entered before the first statement and exited after the last statement of the MODULE.

Our programs are going to become more complex. Therefore, we will break down the PROCEDURE DIVISION into a greater number of MODULES. For the larger COBOL programs, the use of module numbers will help our program documentation. (Like trying to find a certain item in a book without an index.) Several numbering systems are in general use. We will work with one type at this time and another type in later articles so that you will be able to choose which method you prefer. This first system uses a three-digit ascending number as described in the following table:

MODULE NUMBER	MODULE FUNCTION
000	Main Processing Module
100-199	Initialization Module
200-699	General Processing Modules
700-799	End-of-program totals, statistic, etc.
800-849	Input (Read) Modules
850-869	General non-report output (Write) Modules
870-879	Report Heading Modules
880-889	Report Top-line Output (page-skipping)
890-899	Report-line Output (line spacing)
900-999	Inter-program Communication

I will be showing you examples of the module numbering in some of the next programs we will be working with. Each module must still have a unique name that will describe (self-documenting) the func-

tion of the module. We must also remember that these module names must be within a 30-character user-defined maximum. The use of readable abbreviations will be used.

For our purpose, we will define structured programming as a program design, documentation, coding, and testing methodology that utilizes techniques in program development to create proper, reliable, and maintainable software products on a cost-effective basis. Note that this definition is good for languages other than COBOL also!

### Added Style Summary

Whenever fields are to be operated upon arithmetically, each field must be defined as NUMERIC (not EDITED), and the field size, decimal point location, and arithmetic sign must be considered.

When designing a report program that accumulates and prints totals, the following considerations must be provided for:

1. DATA DIVISION.
  - A. Record-description for each total-line must be specified.
  - B. Input record fields that are operated on arithmetically must be defined as NUMERIC.
  - C. Fields must be defined in WORKING-STORAGE to accumulate the totals.
2. PROCEDURE DIVISION.
  - A. Every WORKING-STORAGE accumulator field MUST BE initialized to ZERO.
  - B. After all detail processing for all the input records has been completed, the total accumulations should be made.
  - C. A procedure to print each of the total-lines must be performed after all input records have been processed, but before the output report file has been closed.

### Complete Report Program

A complete report program is one that contains heading lines, detail lines, and total lines! All of our remaining programs will always meet the above description. To add these Heading Lines, we must add many new MODULES, with module numbering, to our Structure Chart.

Our TOP level module will always be similar in function to this one:

```
000-PRINT-ACCT-REC-REPORT.
```

Our FIRST level modules will be something like the following:

```
100-INITIALIZE-VARIABLE-FIELDS.  
200-PROCESS-ACCT-REC-RECORD.  
800-READ-ACCT-REC-RECORD.  
700-PRINT-TOT-LINE1.  
750-PRINT-TOT-LINE2.
```

Next, we expand on our FIRST level modules and choose SECOND and THIRD levels modules, as necessary. They could look like the following:

```
870-PRINT-REPORT-HEADING.  
880-WRITE-REPORT-HEADING.  
890-WRITE-REPORT-LINE.
```

Did we make good use of the above module numbering table? Are our module names readable and self-documenting? Do you have better ones? I want you to be creative! If I have made a good choice, you should have a good idea of what this program will do and how it will be designed without ever seeing the program specifications. Can you visualize the program?

### Closing

The next COBOL Corner will start you on Program #5, which will have specifications requiring a complete report program with Heading, Detail, and Total Lines. The program will be an Account Receivable Report. This program will expand on our use of the WORKING-STORAGE SECTION. We will add the following:

1. WS-REPORT-CONTROLS. This area will keep count of report lines used, lines per page, and page count.
2. WS-DATE-AREA. This will provide us with one way to insert a date into our report heading.
3. HEADING-LINES. We will define our heading-line format in WORKING-STORAGE.
4. DL-DETAIL-LINE. We will now define our report detail-line in WORKING-STORAGE and not in the FILE SECTION as we have done previously.
5. TL-TOT-LINES. The total-lines will now be defined in the WORKING-STORAGE SECTION and not in the FILE SECTION.

Program #5 will use the COBOL verb "IF" for the first time.

For your "homework", please do the following:

1. From the MODULE names that I provided above, prepare a complete Structure Chart with a TOP level, FIRST level, SECOND level, and a THIRD level. If you do not remember how we did Structure Charts, go back and review earlier articles.
2. Study the IF/THEN COBOL verbs in your COBOL-80 Manual.



### FOR PRESCHOOLERS EDUCATOR APPROVED EDUCATIONAL SOFTWARE FOR THE H8/H89/Z100

#### LETTERS AND NUMBERS \$29.95

Give your child a head start by introducing the world of letters, numbers and computers. This set of 8 moving graphic programs will stimulate learning and teach the fundamental relationship of letters to words and numbers to objects. For HDOS, CP/M, CP/M-85. Listable MBASIC programs. 48K memory required.

#### PRESCHOOL CHALLENGE \$29.95

Challenge your child to recognize letters, count objects and learn the concepts of same/different, right/left, over/under etc. This set of 9 moving graphic programs provide an enjoyable learning environment for your child. For HDOS, CP/M, CP/M-85. Listable MBASIC programs. 48K memory required.

#### MOTHER GOOSE \$29.95

Teach your child early reading concepts with these 11 moving graphic Nursery Rhymes. As the words appear on the screen, the mouse runs up the clock, the cow jumps over the moon, Wee Willie Winkie runs through the town etc. For HDOS, CP/M, CP/M-85. Listable MBASIC program. 48K memory required.

#### ALIEN FACES \$29.95

Allow your child to be creative with a program that will generate over a million happy, sad, scary or goofy faces. Faces change with single keystrokes, the "5" changes the nose, the "4" changes the left eye etc. For HDOS, CP/M, CP/M-85. 48K memory required. Z-100, HDOS version requires MBASIC.

Programs feature easy to use menus and are available on hard or soft sector 5 and 1/4 inch distribution disks. The Z-100 versions are in color.

**INTUITIVE LOGIC**  
412 Taylor Street  
Rochester, MI 48063-4327  
(313) 651-3859

# HERO SEEKs A Friend

Dr. Kenneth R. Hill  
404 Ben Oaks Drive East  
Severna Park, MD 21146

The robot wanders around the room or from room to room, avoiding obstacles, until commanded to "STOP". Another vocal command (any loud sound) will restart or jump to a different utility program (supplied by user) as desired.

```
                                0200-0375
0200 B3          M.L.           )
0201 B6 00      LDA A w $0      )
0203 97 10      STA A @ 0010   )
0205 97 40      " " " 0040    )
0207 97 41      " " " 0041    )--Initialize
0209 B6 EE      LDA A w $EE     )
020B 97 11      STA A @ 0011   )
020D 3F         R.L.           )
020E 45         enable sonar   )
020F B3          M.L.           )
0210 BD 03 2B   JSR 3.          ) START of main program.
0213 7C 00 40   INC (M+1->M; M=0040) ) (Check for Voice command)
0216 96 40      LDA A w val. in 0040
0218 BD F6 4E   JSR (REDIS)    ) Display count value.
021B BD F7 AD   JSR (OUTBYT)   )
021E BD 03 70   JSR 4. (delay for display)
0221 B1 50      CMP A w $50 =.B0
0223 24 02      BCC
0225 20 07      BRA
0227 3F         R.L.
0228 21         zero all motors; center some
0229 B3          M.L.
022A B6 00      LDA A w $10
022C 97 40      STA A @ 0040
022E 3F         R.L.
022F C3 D0 61   center head,med.,abs.,wait
0232 C3 F0 4A   center steer,med.,abs.,wait
0235 CC 0B FF   move forward,slow,abs.,cont. (or CC 13 FF for medium motion.)
0238 B3          M.L.
0239 CE 00 03   LDX w $03
023C 96 11      LDA A w val. in 0011
023E BD F6 4E   JSR (REDIS)    ) Display range value.
0241 BD F7 AD   JSR (OUTBYT)   )
0244 0C         CLC
0245 B2 48      SBC A (A-M-C->A) ) Range limit = 48H
0247 25 02      BCS
0249 20 EE      BRA
024B 96 11      LDA A w val. in 0011
024D 0C         CLC
024E B2 20      SEC A (A-M-C->A)
0250 24 0A      BCC
0252 CE 00 0A   LDX w $0A
0255 BD 03 0B   JSR 1.
0258 25 07      BCS
025A 20 DD      BRA
025C BD 03 20   JSR 2.
025F 24 DB      BCC
0261 3F         R.L.
0262 02         abort drive motor
0263 72 FB 7B   speak,wait "There is something in my way"
0266 B3          M.L.
0267 01 01 01   NO-OPS (or JSR 3.)
026A 3F         R.L.
026B C3 C8 5E   turn head,CCW(L),slow,abs.,wait -3 )
026E C3 D0 5B   " " " ,med., " " -3 ) -16 "Look left"
0271 C3 DB 51   " " " ,fast, " " -10 )
```

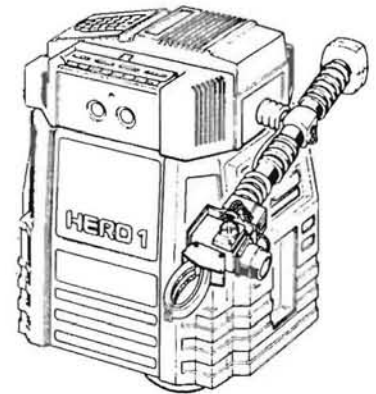


```

0274 BF 00 02  pause,1/8 sec
0277 83          M.L.
0278 CE 00 03  LDX w $03
027B BD 03 0B  JSR 1.
027E 24 0E      BCC
0280 3F          R.L.
0281 C3 EC 93   steer,CW,slow,abs.,wait  )
0284 D3 08 0D   move forward,slow,rel.,wait  ) - Turn R 90
0287 C3 E8 4A   steer,CCW,slow,abs.,wait  )
028A 83          M.L.
028B 7E 02 13   JMP (back to start+1)
028E 96 11      LDA A w val. in 0011
0290 BD F6 4E   JSR
0293 BD F7 AD   JSR
0296 97 41      STA A @ 0041
0298 3F          R.L.
0299 BF 00 03   pause,3/16 sec.
029C 83          M.L.
029D 96 11      LDA A w val. in 0011
029F B1 00 41   CMP A w val. in 0041
02A2 26 EA      BNE (br. if not = 0)
02A4 96 41      LDA A w val. in 0041
02A6 16         TAB (A->B;store L range in B)
02A7 3F          R.L.
02A8 BF 00 02   pause,1/8 sec.
02AB D3 C8 03   turn head,CW(R),slow,rel.,wait  +3 )
02AE D3 D0 03   " " " ,med., " " +3 ) +32 "Look Right"
02B1 C3 D8 71   " " " ,fast,abs., " +26 )
02B4 BF 00 02   pause,1/8 sec.
02B7 83          M.L.
02B8 CE 00 03   LDX w $03
02BB BD 03 0B   JSR 1.
02BE 24 0E      BCC
02C0 3F          R.L.
02C1 C3 E8 00   steer,CCW,slow,abs.,wait  )
02C4 D3 08 0D   move forward,slow,rel.,wait  ) - Turn L 90
02C7 C3 EC 4A   steer,CW,slow,abs.,wait  )
02CA 83          M.L.
02CB 7E 02 13   JMP (back to start+1)
02CE 96 11      LDA A w val. in 0011
02D0 BD F6 4E   JSR
02D3 BD F7 AD   JSR
02D6 97 41      STA A @ 0041
02DB 3F          R.L.
02D9 BF 00 03   pause,3/16 sec.
02DC 83          M.L.
02DD 96 11      LDA A w val. in 0011
02DF B1 00 41   CMP A w val. in 0041
02E2 26 EA      BNE
02E4 96 41      LDA A w val. in 0041
02E6 10         SBA (A-B->A) if R > L, turn right, etc.
02E7 24 11      BCC
02E9 3F          R.L.
02EA C3 F0 35   steer,CCW(L),med.,abs.,wait  )
02ED D3 08 10   move forward,slow,rel.,wait  ) - Turn L 15
02F0 C3 F0 4A   center steer,med.,abs.,wait  )
02F3 C3 D0 61   center head,med.,abs.,wait  )
02F6 83          M.L.
02F7 7E 02 10   JMP (back to START line)
02FA 3F          R.L.
02FB C3 F0 5F   steer,CW(R),med.,abs.,wait  )
02FE D3 08 10   move forward,slow,rel.,wait  ) - Turn R 15
0301 C3 F0 4A   center steer,med.,abs.,wait  )
0304 C3 D0 61   center head,med.,abs.,wait  )
0307 83          M.L.
0308 7E 02 10   JMP (back to START line)
030B BD F6 4E   JSR REDIS Subr. 1.
030E 96 11      LDA A w val. in 0011 Check for range value < 20H
0310 BD F7 AD   JSR OUTBYT Go to Subr. 2. if true.
0313 0C         CLC
0314 B2 20      SEC A (A-M-C->A)
0316 25 01      BCS
0318 39         RTS
0319 BD 03 20   JSR 2.
031C 25 FA      BCS

```

031E 20 EB	BRA	
0320 0C	CLC	Subr. 2.
0321 09	DEX (X-1->X)	Check for 3 consecutive
0322 26 01	BNE	identical range values.
0324 0D	SEC	
0325 3F	R.L.	
0326 8F 00 03	pause,3/16 sec.	
0329 83	M.L.	
032A 39	RTS	
032B 3F	R.L.	Subr. 3.
032C 42	enable sound	Listen for loud sound.
032D 8F 00 02	pause,1/8 sec.	Stop and wait if sound greater
0330 83	M.L.	than AOH is heard.
0331 CE 08 00	LDX w #0800	
0334 BD F6 4E	JSR	
0337 B6 C2 40	LDA A w val. in C240	
033A BD F7 AD	JSR	
033D 0C	CLC	
033E B2 A0	SBC A (A-M-C->A)	
0340 24 08	BCC (br. if A>M;M=A0)	
0342 09	DEX	
0343 26 EF	BNE	
0345 CE 00 03	LDX w #03	
0348 0D	SEC	
0349 39	RTS	
034A 3F	R.L.	
034B 8F 00 04	pause,1/4 sec.	
034E 72 FD 46	speak,wait "Your wish is my command"	
0351 8F 00 04	pause,1/4 sec.	
0354 83	M.L.	
0355 BD F6 4E	JSR	Listen for second loud sound.
0358 B6 C2 40	LDA A w val. in C240	Continue with main program or
035B BD F7 AD	JSR	jump subr. if heard.
035E 0C	CLC	
035F B2 A0	SBC A (A-M-C->A)	
0361 24 08	BCC	
0363 BD 03 20	JSR 2.	
0366 20 ED	BRA	
0368 01 01 01	NO-OPS	
036B 39	RTS (replace with a jump to a user utility	
	outline if desired - change line 0361 accordingly)	
036C 01 01	NO-OPS	
036E 01 01	NO-OPS	
0370 3F	R.L.	Subr. 4.
0371 8F 00 10	pause, 1 sec.	1 sec. pause where needed
0374 83	M.L.	
0375 39	RTS	



# SIG News



Bill Parrott  
7010 Caenen  
Shawnee, KS 66215

This is the first of what I hope will be a fairly regular column (notice that I didn't say monthly) to keep folks informed of what's happening on the HUG SIG and to provide special recognition for those members of the SIG who have provided outstanding support. For those of you who might not be aware, the HUG SIG (Special Interest Group) is basically an electronic bulletin board which is run on the CompuServe Information Service (CIS). It incorporates, however, much more than a simple message exchange facility. It includes an extensive data base of member contributed software and real-time interaction with other members via the conference feature.

## Members of the Month

I want to make my first order of business to recognize and thank several members of the SIG who have been designated "Member of the Month" <MOM>. The SYSOPs (Jim Buszkiewicz and myself) get together each month and select a member who has contributed in some significant way to the SIG during that month as <MOM>. The selected person is "tagged" with <MOM> next to his/her name and runs free on the SIG for the entire following month. Because this is the first time this has been mentioned in REMark, I'll go back to the beginning and recognize each <MOM> and say a little bit about his contributions.

### January, 1984 - Dale Wilson (72155,1402)

Dale is a long-time HUGgie and SIG member and was chosen as our first <MOM> for his interest and particularly his support of the Z-100. He has helped many members with graphics concepts in the Z-100, particularly relating to the "C" language.

### February, 1984 - Mike Cogswell (70140,363)

Most people will know Mike as former President of the Capitol Heath Users' Group (CHUG). Mike has provided extensive support of the Z-100 and has become the resident authority on 8088 assembly language programming under Z-DOS (MS-DOS). If anyone has a question in this regard, Mike has the answer!

### March, 1984 - Steve Whitney (72355,1100)

Steve was named as our March <MOM> for his continued support of members of the SIG. Whenever anyone has a question, Steve wastes no time jumping right in there with whatever assistance he can give. This is exactly what the SIG is about... helping others and learning from others. There is not one of us who cannot use a little help from time-to-time.

### April, 1984 - Frank Ivan (70003,2244)

Frank was named <MOM> for the month of April for his candid view and opinions of the Z-150 and what he saw as its probable effect on the Z-100. Frank led an "active" discussion for several weeks on this topic. (It was in fact this very discussion that prompted my comments in the May issue of REMark.) Frank pointed out to us, however, that because he does work for CIS he already has a "free" user number and requested that we name someone else who might better benefit as <MOM> so we named:

### April, 1984 - Dale Lamm (70555,302)

Dale Lamm is best known as the author of the ever-popular modem packages ZLYNK (for HDOS) and ZLYNK/II (for CP/M). However Dale has contributed more to the HUG SIG software data base than any other member and it was for this that he was recognized.

### May, 1984 - Rick Schaeffer (70120,174)

Rick's contribution to the SIG data base was the Kernighan line editor, adapted from "Software Tools" for the "C" language and Heath/Zenith computers. For those not familiar with this editor, its primary advantage is the extremely powerful pattern matching which is employed. Rick proposed a SIG effort to expand the capabilities of this editor to include full-screen operation and I know of several users who are participating in this effort.

### June, 1984 - Mike Curiale (71715,1365)

Mike is one of the members of the SIG whose interest lies strictly with the HERO I robot. Mike has written some nifty routines for his robot and has shared them with members via the program data base. He has also fielded questions from other members regarding construction, operation, and programming of HERO I.

## Current Events

Rick Schaeffer was recently making some remarks about the superiority of Pascal over "C" (or was it the other way?). Anyway, he basically said that he could produce good software for any given application in his language faster than a programmer proficient in and using the other language. Not one to pass up an opportunity to start a good war, the SYSOPs have proposed a contest to settle once and for all (yuk, yuk!) the question of which is the better language. The contest (currently under way) pits Pascal programmers against C programmers in the production of a unique application. The specifications for the contest defined a "Programmers" calculator using the popular Reverse Polish (RPN) notation. We felt that this particular problem would be equally suited for C and Pascal and that no group would have an advantage over the other by virtue of their language of choice.

Now, no contest is complete without prizes so we're offering prizes. The author of the program judged best to conform to the specifications and completed in the most timely manner will receive his or her choice of any Heath/Zenith software product AND one month free operation on the SIG. The author of the second best entry will receive his choice of either any Heath/Zenith software product OR one month free operation on the SIG. The author of the third best entry will receive any HUG Software product he or she chooses. Needless to say, with this kind of booty up for grabs, interest is strong. We hope to be able to publish the winning entries in an upcoming issue of REMark, so stay tuned!

## We Want You!

If you're not a part of the action on the SIG, you're missing out on one of the best things about being a Heath/Zenith user. All it takes to get in on the fun is to subscribe to CompuServe. The easiest way to do this is to obtain the MicroNET Connection from HUG (p/n 885-1122[-37] for HDOS, 885-1224[-37] for CP/M) which contains everything you need to join CompuServe and the HUG SIG, including your user number, password, and modem software. This package costs only \$16.00 and is the cheapest way to subscribe to CIS.

Now that you see what we do, what are you waiting for? Join us on the SIG for a rewarding experience (not to mention a good time)! I'll be looking for you soon.



# THE QUALITY SOFTWARE SOLUTIONS YOU NEED

Now for ZDOS!

## SOLUTION #1: CATALOG-MASTER

If trying to find the disk file you need right now is a little like looking for a needle in a haystack, you need **CATALOG-MASTER**.

You will realize the following benefits:

- you can catalog your disks quickly and easily
- you can catalog all your disks, even different types
- you can find the disk files you need quickly
- you can get more information by using file descriptions

Find that file you need quickly and easily with **CATALOG-MASTER**

Requires 2-drives and 48K RAM  
Specify HDOS or CP/M when ordering.  
Regular Price . . . \$29.95

Special HUG price . . . only **\$24.95!**  
(First Next 100 orders!)

Get both the HDOS and CP/M versions for only . . . \$45.00  
**LIMITED TIME OFFER — CALL YOUR ORDER IN TODAY**

## SOLUTION #2: FUND-MASTER

If you would like to know not only when to buy or sell stock but also how much to buy or sell, you need **FUND-MASTER**.

You will realize the following benefits:

- you can know when to buy or sell your shares
- you get timely help on how much to buy or sell
- you get the current status of your investment
- you get a complete history of your stock transactions
- you do not have to be a stock market expert to benefit

Get the tool that provides effective information for the management of stocks and mutual funds — get

**FUND-MASTER** ★ Requires 48K RAM

Specify HDOS, CP/M or ZDOS when ordering.

Regular Price . . . \$39.95 (HDOS or CP/M)

Special HUG price . . . only **\$34.95**

Regular Price . . . \$59.95 (ZDOS or MS-DOS)

Special HUG price . . . only **\$49.95**

**LIMITED TIME OFFER - CALL YOUR ORDER IN TODAY**

## SOFTWARE PRODUCTS FOR YOUR H8, H89, Z90, and H/Z100 COMPUTER

### ATTENTION SOFTWARE AUTHORS

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

All products are available in HDOS or CP/M format, except where noted. Add \$2 for shipping. Michigan residents please add 4% for sales tax. PAYMENT IN U.S. FUNDS PLEASE.  
Call or write for more information and our **FREE** catalog.



**For Faster Delivery**  
**Call 906-249-9801**



**GENERIC SOFTWARE**  
P. O. Box 790 - Dept. 884R  
MARQUETTE, MI 49855  
906-249-9801  
10 AM - 5 PM EST

## ASSEMBLER CAN BE FUN

### Announcing the OMNICODER™ Assembler

Meaningful prompts. Descriptive error messages. Unlimited nesting of INCLUDE and macro. Macro definitions with both positional and keyword parameters. Global macro symbols. Subscripted macro symbols. User controlled disk search order. Symbols up to eight characters. List and display files can be directed to terminal, printer, or disk. Uses disk work files for expansions. Produces object modules to be linked by linkage editor. Object format designed to be compatible with multiple processors (up to 32-bit addressing) in order to support planned additions (initial release will only assemble Z80® code).

### System Requirements

Z80®, 64K. CP/M®. One or two disk drives (depending on drive and storage capacity). Formats: 5-inch hard or soft-sectored (48 t.p.i.), 8-inch.

### Future Plans

We plan to add multiple CPU and MNEMONIC capabilities. This assembler is intended to become part of our PANDORA OPERATING SYSTEM (watch for announcement).

### Ordering Information

Price: \$90 per site (USA only, no foreign orders). Ohio residents add 5.5% sales tax. MC®/Visa®: include name, number, expiration date, signature, and phone number. Allow six to twelve weeks for delivery. Send order to:

### MOCKINGBIRD DATA SYSTEMS

2296 Hoover Rd.  
Grove City, OH 43123

Trademarks: Zilog Z80®, Digital Research CP/M®, Heath Company: HDOS™  
Mockingbird Data Systems: OMNICODER™, PANDORA™

## IBM-PC/ZENITH Z-100 users.

Expand your computer universe with—

### micro/VERSAL™

A utility program to READ/WRITE over 20 different 5¼ (CPM, CPM/86, MS-DOS & USER DEFINABLE) disk formats. Now you can easily transfer text, data, or programs between many different micro computers by simply loading **micro/VERSAL™** and the disk you want to READ/WRITE from or to. **micro/VERSAL™** also includes comprehensive utilities to DUMP any 5¼ disk by track, RDSECT to read disk sectors and FAPP, a program to append files together to produce a large file. For disk formats not directly supported, **micro/VERSAL™** provides customization routines that allow users to write their own directory routines.

Now With Formatting

**micro/VERSAL™ \$79.99**

plus \$4.00 shipping & handling

Also **COED™** full screen editor..... **\$34.97**  
Includes: Stack arithmetic, MACRO commands, multiple files, definable function keys and much more.

CREDIT CARD ORDERS: Master Charge / VISA  
MAIL ORDERS: Checks or money orders  
N.J. resident add 6% sales tax.

### ADVANCED SOFTWARE TECHNOLOGIES

417 Broad Street  
Bloomfield, N.J. 07003  
**(201) 783-7298**



# An Introduction To 'C'

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

This is the seventh in a series of articles intended to introduce the 'C' programming language.

The program presented in Part Six was a simplified phone directory. In it we allocated several arrays in order to store the directory entries. The memory addresses for these arrays were automatically allocated for us in the requested size whenever our program was entered. We arbitrarily allocated 30 character positions for each of the ten array elements. For a small number of array elements it does not matter if the 30 positions are more than we need. But if the number of array elements is large, then the number of character positions allocated for each array element makes a big difference, unless we have unlimited memory capacity (or virtual memory). One alternative is to allocate the storage at run-time through the use of the C/80 function 'alloc'. This function returns a pointer to the requested number of bytes. Not only is this method better in that we can control the amount of space allocated, but is more efficient, because array references require the use of a pointer, whether specified or implied (so we might as well specify one).

We will keep track of the values returned by 'alloc' in an array. Our declaration will be 'char \*name[10]' which allocates ten pointers to character arrays. Next, we need to allocate the storage, but how do we know how much storage to allocate? There are two things we can do. One, we can estimate our storage requirements and supply that number as the argument passed to 'alloc'. This is probably OK if the storage requirements are constant. In our case, the size of the telephone directory is likely to change, so we don't want to be restricted by a fixed storage allocation number. We could allocate a large amount of storage, even if we didn't need it all, as long as there is enough memory to satisfy the allocation. But that seems like a waste of space. Why don't we try a different approach. Let's read each string into a temporary variable, determine how long the string actually is, and then only allocate the amount of space needed. This works something like this:

```
scanf("%s", s_name);
name[0] = alloc(strlen(s_name));
```

'Strlen' is a C/80 function which returns the length of a string. If we use the value returned by the function as the input supplied to 'alloc', we will only allocate the number of bytes we need. (Remember that each character occupies one byte of storage.) But wait! The value returned by 'strlen' does NOT include the terminating '\0' which we DO need to account for in our allocation. So we need to allocate one more byte than is returned by 'strlen'.

```
name[0] = alloc(strlen(s_name)+1);
```

We also would want to check that our allocation command executed successfully. 'Alloc' returns the 'NULL' value if the allocation is not successful.

```
if ((p = alloc(strlen(s_name)+1)) != NULL)
    name[0] = p;
else
```

```
{
    printf("Allocation Error.\n");
    exit(12);
}
```

Now, let's look at breaking our program into logical pieces. One thing 'C' easily lends itself to (as do other structured languages) is a separation of functions into logical units. This adds overhead to program execution, but if there is one routine that is used many times within the body of the program, it makes a lot more sense to have a separate subroutine which is called each time the routine is needed. In the 'phone' program, there weren't any routines which were executed more than once, which is one reason why I didn't use subroutines. But in most instances it is good programming practice to have a subroutine perform a logical piece of work, regardless of whether it will be executed more than once or not.

A subroutine is actually a program in itself, and must be able (at least logically) to stand on its own. Any variables passed into the subroutine must be declared before the opening brace. A subroutine can pass a value back to the calling routine via the 'return' statement, or it can return automatically when the end of the subroutine is reached. For example, here's a subroutine which will clear the H-19 screen.

```
clrscr()
{
    putchar(27);
    putchar('E');
}
```

This routine does not accept or return any variables, so the application is very simple. Anywhere in the program ('main' or another subroutine), we can invoke this subroutine simply by using its name at the point we want to clear the screen.

```
clrscr();
```

The parentheses are required even if no variables are passed in order for 'C' to differentiate between a variable and subroutine name.

Let's write a subroutine which passes a variable back. This subroutine should change to 'character' mode, accept a character from the terminal, change back to 'line' mode, and return the character to the calling procedure. It looks like this:

```
cget()
{
    int c;

    #asm
        XRA    A
        MVI    B,201Q
        MVI    C,201Q
        SCALL  SCOUT
    #endasm

    c = getchar();

    #asm
        XRA    A
```





If you have followed the seven articles in this series, and have taken the time to delve further into 'C' on your own, you have surely amassed at least a basic understanding of all that is necessary to program successfully in 'C'. As with any programming language, if you don't continue to use it, you will quickly lose what you have already learned. One of the best ways to learn more about a language and the things you can do with it is to incorporate the programming ideas and techniques of other people into your own work. This will ultimately spawn some of your own ideas when you fully understand what others have done.

Along these lines, I would like to introduce an addendum to this series. Starting with this article, I will attach a question-and-answer section to the end of the written text. If you have any questions, comments, suggestions or program samples, please feel free to submit them to me or to the editor of this magazine. I am looking to do many things by this.

- 1) To get a feeling for the types of things I should be covering in these articles.
- 2) To get feedback on if these articles are useful at all, or if they could be refined to be more so.
- 3) To answer questions which were not properly addressed by me, or were initiated by something I touched on.
- 4) To get other people's input on ways of writing programs or subroutines.

Don't feel that these comments need only be from professional programmers. Remember, the obvious solution to a problem is often the best, and sometimes programmers (myself included) are so busy trying to find the most intricate way to approach a problem that we lose sight of this. Please feel free to participate in this in any way you like.

With that behind us, here we go!

Mr. William S. Hall from Ann Arbor, MI, has made several relevant comments regarding the series so far. He points out that the C language "is a fine language that can suffice for nearly every microcomputer activity." He raises a few points about my lack of using the structured concepts which are a basis of 'C'. He is correct, and more attention will be paid to this in the future (as was already done in this article). He has provided a 'modularized' version of the 'more' program (presented in an earlier article). His program is certainly to be commended. Keep one thing in mind -- each separate subroutine adds additional overhead to the 'C' program, in general and specifically each time a subroutine is entered. This is usually not a concern, but it may well be when speed is of the essence. One of the most obvious advantages to his program is that even though it is written specifically for the H-19 terminal, it can be easily modified for other terminals by replacing one or more of the subroutines, without having to touch the main program. Here then is his program:

```

/*.....*/
/*
/*
/*      more.c
/*
/*      A structured version by W. S. Hall
/*.....*/
#include <printf.h>
#define EOF -1
#define ESC 27
#define stderr 0 /* terminal */

main()
{
    int c, line_number;
    line_number = 0; /* initialize the count */
    while ((c = getchar()) != EOF) {
        putchar(c);
        if (c == '\n')
            line_number += 1;
        if (line_number == 23) {
            pause();
            line_number = 0;
        }
    }
}

curspos(x,y) /* position the cursor. Enter with line and
column number */
{
    int x,y;
    x = x + 31; /* add the offset */
    y = y + 31;
    printf("%c%c%c",ESC,x,y);
    return;
}

savgurs() /* save the cursor position */
{
    printf("%cj",ESC);
    return;
}

rstcurs() /* restore the cursor position */
{
    printf("%ck",ESC);
    return;
}

cursoff() /* turn off the cursor */
{
    printf("%cx5",ESC);
    return;
}

curson() /* turn the cursor on */
}

```



```

pause()
{
  open25();
  savcurs();
  curspos(25,1);
  cursoff();
  revvid();
  printf("—More—");
  normvid();
  while (getc(stderr) != '\n') /* wait */
  ;
  rstcurs();
  cursor();
  close25();
  return;
}

open25()
{
  printf("%cx1",ESC);
  return;
}

close25()
{
  printf("%cy1",ESC);
  return;
}

```

```

{
  printf("%cy5",ESC);
  return;
}

revvid() /* go to reverse video */
{
  printf("%cp",ESC);
  return;
}

normvid() /* normal video */
{
  printf("%cq",ESC);
  return;
}
}
/***** end of program *****/

```

The next letter is from Charles McClelland from Shell Beach, CA. He is an assembly language programmer who has a problem he has been unable to solve regarding the use of '#asm' and '#endasm' in 'C'. He is trying to convert an action game from CP/M to HDOS. He states, "in CP/M when you go to direct input/output with function #6 and (you have coded for the input mode) have yet punched a key, function 6 returns a 0 to register A (or RETURN in C). Clem used this fact in SWITCH so his little moving object keeps moving and a key hit (arrow keys) controls it a bit. With HDOS, for lack of that 0 returned constantly, Clem's game comes to a halt. Problem: how to write a 0 again and again as input cycles waiting for a keystroke. So far, I have had no success..."

Any (more) experienced 'C' and/or assembly language programmers have a suggestion? I am not at all familiar with 'C' or assembler under CP/M, so I'll have to pass on this one.

Thanks for your letters -- hope to 'C' more next time.



## 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

# DOODLER

Graphics Package

... for the Zenith Z-100

- Full Featured Graphics Design Package.
- Palette of 36 different color choices.
- Store Designs on Disc for later Reference.
- Playback Mode for Error Correction.
- Extended Text Capabilities including...  
User Designed Character Fonts.  
Italic or Backslant Style and scaling.

See DOODLER at your local HEATHKIT Store.

...or Send \$ 79.95 directly to...



DATA SYSTEMS CONSULTANT  
P. O. Box 535  
St. James City, FL 33956

**813-283-2227**

Specification Sheet available on Request.

# Update On Heath/Zenith Related Vendors

*(These are in addition to vendors listed in the January 1984 issue)*

## **Articulate Publications, Inc.**

402 N. Larchmont Blvd.  
Los Angeles, CA 90004

Contact: Darnell M. Hunt Phone: 213-871-1350

Products: MicroMed & MicroDent, Version 4. Base Price - \$2700. The MicroMed and MicroDent packages are comprehensive, state-of-the-art medical and dental office management systems that run on most of the popular microcomputers on the market. Available options: 8/16 bit processor, monochrome or color display, configurations for single user, multiuser 6, multiuser 16. Call or write to vendor for further information.

## **Budget Software**

P. O. Box 221  
Berrien Springs, MI 49013

Comments: Mail order and correspondence only. Products: Software available at \$14.99 a disk, sources always included. Specializing in providing quality personal/home use software at minimal cost for people on a budget. Software is always compiled or assembled and in normal use requires no additional interpreters or compilers. For use only with Heath/Zenith computers using CP/M-80 and CP/M-85. Send a self-addressed, stamped envelope for a copy of the free catalog. Software line includes games, Diablo and MX-80 printer handlers for graphics lettering, CAI and other specialty programs.

## **CLEO Software, A Division of Phone 1, Inc.**

461 N. Mulford Road  
Rockford, IL 61107

Contact: Roy Lane Phone: 815-397-8110

Comments: Software and telephone consultation available. Products: CLEO-3270 supports the full features of IBM's 3276/2 cluster controller to enable your Z-100 to emulate a 3278 display station with 3287 printer support. CLEO-3780 supports the full features of IBM's 3780 protocol for high speed (up to 9600 baud) transmission and reception of data files over ordinary telephone lines. No additional hardware is required with either package.

## **Computer Consultants to Business**

1033 Bishop Walsh Rd.  
Cumberland, MD 21502

Contact: Barbara or George Sellers Phone: 301-759-1260

Comments: Software, Hardware, and consultation available. Products: Authorized Zenith Data Systems Dealer and Service Center. Epson and Daisywriter printers and Racal-Vadic modems. IBM software, computers, and peripherals. TecMar products for instrumentation, AST and Quadram. DISK-TRAN® disk format conversion programs Osborne, ZDOS (MSDOS & PCDOS), KayPro,

Cromemco, Direct 1000, and others to/from Z-90 CP/M-80, Z-100 CP/M-85 and ZDOS, and Z-150/160 PC and IBM. Construction Calc™ templates for SuperCalc 1 or 2 for construction cost estimating.

## **Digital Marketing Corporation**

2363 Boulevard Circle  
Walnut Creek, CA 94595

Contact: Tim Clark Phone: 415-947-1000  
800-826-2222

Products: Project management and time scheduling program, text-oriented data base management system, appointment scheduling program, word processing enhancement for WordStar, and more. Formats include MS/PC-DOS, CP/M-86, CP/M.

## **Headware**

2865 Akron St.  
East Point, GA 30344

Contact: Dick Riggs

Products: Software available. Composite directory of public domain CP/M programs. Contains over 4,500 entries. Subjects include astronomy, business, education, engineering, radio, music, speech, more. Directory is on disk. On Heath 5 1/4 inch CP/M disks: 3 HS/\$17, 2 SS/\$12, 1 SSDS/\$7. Add \$2 domestic S&H.

## **Paul F. Herman, Data Systems Consultant**

P. O. Box 535  
St. James City, FL 33956

Contact: Paul F. Herman Phone: 813-283-2227

Comments: Hardware, Software, and consultation available. Products: Business and Graphics Software for the Z-100. Custom software and turn-key systems. Authorized Dealer/Service Center for Zenith Data Systems and Star Micronics.

## **Hoyle and Hoyle Software**

716 S. Elam Avenue  
Greensboro, NC 27403

Contact: Janet C. Hoyle Phone: 919-378-1050

Comments: Software and consultation by phone or mail available. Products: Five products available in ZDOS and all standard CP/M formats. Query!2, database management system with 4000 character/record potential, fast sorting and up to 40 Search conditions permitted. SOON! CALC with arithmetic functions and utilities with programs to modify existing databases, read in standard files and merge databases.

## **Insurance Sales Systems**

8015 West 63rd Street, Suite 4  
Merriam, KS 66202

Contact: Don Sprowl Phone: 913-722-0065

Comments: Software and consultation available. Products: Auto PIF, Homeowners PIF, Life PIF, Auto Rating, Proposal Software, Masterfile, IRA, HR-10, Loan Amortization & Depreciation, Mortgage Acceleration, Financial Goals Proposals, Total Needs Analysis. Unique electronic data base management files designed by an insurance agent for insurance agents.

**Magnolia Microsystems, Inc.**

2264 15th Avenue West  
Seattle, WA 98119

Contact: Sales Dept. Phone: 800-426-2841

Comments: Hardware, Software, and consultation available. Products: MAGNet™ Local Area Network for Z89/90, Z110/120, and Z150/160 PC-Compatible computers. 128K RAM Board w/CP/M Plus, Winchester Subsystems, DD controller, SASI-bus interface for Z89/90. Please call for catalog or further information. MC, VISA, American Express accepted.

**Mako Data Products, Inc.**

1441-B N. Red Gum St.  
Anaheim, CA 92806

Contact: Diane Barron Phone: 714-632-8583

Comments: Hardware, Software, and consultation available. Products: (Hardware) MH89+3 Buss Expander, PSGx2 Sound Board for H89/Z90, PSGx4 Sound Board for H8. (Software) Unit Conversion Master, Micro Piano, PSG Demo, Load-'n-go.

**MMORROWARE**

12503 Gristmill Cove  
Austin, TX 78750

Contact: Mark Morrow Phone: 512-250-1303

Comments: Software available. Products: Z-100 software. Colorful flashcard program for children, powerful data reduction/analysis routines for the scientific community, accurate matrix diagonalization/eigenvector calculations for normal mode problems, etc. All programs are compiled ZBASIC, FORTRAN, or assembly, and require only ZDOS and 128K.

**Northwest Computer Algorithms**

P. O. Box 90995  
Long Beach, CA 90809

Contact: William Giarla Phone: 213-426-1893

Comments: LISP Software. Products: VO-LISP Programming Environment, supports the usual LISP Interpreter Functions, Screen Editor with Mouse, Compiler & Assembler, access to hardware and CP/M, inexpensive LEAR LISP Tutorial System. Write for free catalog, Visa and MasterCard accepted.

**Software Support, Inc.**

1 Stalker Ln.  
Framingham, MA 01701

Contact: Malcolm Gulden Phone: 617-872-9090

Comments: Hardware and consultation available. Products: Disk drives, diskettes, printers, cables, and peripheral products.

**Taranto & Associates, Inc.**

121 Paul Drive  
San Rafael, CA 94903

Contact: A. Schaffer Phone: 800-227-2868

Comments: Hardware, software, and consultation available. Products: The Taranto & Associates, Inc. completely integrated accounting Package - (can be stand alone, too) - General Ledger, Accounts Receivable/Invoicing/Sales Analysis, Accounts Receivable

Balance Forward, Accounts Payable/Purchase Order, Payroll, Inventory. Some features include integration to WordStar; multiple departmental general ledger distributions; general ledger distributions by invoice line item.

**Techni Service Corporation**

106 Arbuelo Way  
Los Altos, CA 94022

Contact: Ken Barnes Phone: 415-949-1765

Comments: Software and consultation available. Products: We prepare typography for books and other publications, working from text submitted on floppy disk or via modem. The customer may insert typesetting codes, or use substitute codes for word processor codes.

**The Software Toolworks**

15233 Ventura Boulevard, Suite 1118  
Sherman Oaks, CA 91403

Contact: Susan Hayes Phone: 818-986-4885

Products: The largest line of H/Z specific software, including text processing, editors, languages, utilities, games, spelling. Programs available for H-8, H/Z-89, Z-90, H/Z-100/120, H/Z-19.

**U.S. Robotics**

1123 W. Washington Blvd.  
Chicago, IL 60607

Contact: John B. Cleave

Products: Hardware available. 300 & 1200 baud modems, some with auto-dial/answer, diagnostics. 2 year warranty on all hardware products.

**Virtual Devices, Inc.**

P. O. Box 30440  
Bethesda, MD 20814

Contact: Kent Myers Phone: 301-986-1702  
800-762-7626

Comments: Hardware, Software, and consultation available. Products: Program development system and remote control equipment for HERO 1. C Compiler, assembler, add-on ROMs and boards.

**Barry A. Watzman**

560 Sunset Rd.  
Benton Harbor, MI 49022

Contact: Barry Watzman Phone: 616-925-3136

Comments: Hardware, Software, and consultation available. Products: Serial I/O boards for Z-100; Multi-user operation systems for Z-100; CP/M-86 for Z-100; CP/M-86 applications software for Z-100; consulting.

**ZPAY Payroll Systems**

3516 Ruby Street  
Franklin Park, IL 60131

Contact: Paul Mayer Phone: 312-671-3130

Comments: Software and consultation available. Products: Payroll System for hourly, salaried, or commissioned employees. User choice of pay periods. Prints many reports including W2-forms, plus features found in more expensive systems. Available for CP/M, ZDOS, and IBM-PC compatible computers. Price: \$100 plus \$4.00 shipping.

↳ Vectors from 8

ing code to produce a 2 sector file from the same 100 records (HDOS sectors are 256 bytes long):

```
10 DIM A$(49),A(100)
20 OPEN "R",#1,"FILENAME"
30 FOR X%=0 TO 49
40 FIELD #1,5*X% AS D$,5 AS A$(X%)
50 NEXT X%
60 FOR X=1 TO 2
70 FOR Y=0 TO 49
80 LSET A$(Y)=MKI$(A(Y+1+(X-1)*50))
90 NEXT Y
100 PUT #1,X
110 NEXT X
120 CLOSE #1
130 END
```

**Note:** D\$ is a dummy variable, i.e. D\$="".

For your information, the following table is a comparison of the running times of the four sorts. Tests 7 and 8 are 100 and 1,000 member files respectively with 20% of the numbers out of order. This is to simulate the sorting of an existing file like a mailing list to which a number of additions and corrections have been made.

Sorting Times (seconds)

TEST	BUBBLE	SHELL	HEAP	QUICK
1	2.5	18.6	42.8	15.6
2	24.9	302.0	662.1	194.1
3	300.5	27.8	35.7	16.4
4	>900.0MIN	408.8	579.4	201.5
5	250.6	34.0	40.1	29.5
6	>900.0MIN	560.2	626.2	369.4
7	229.3	28.9	42.2	18.2
8	>900.0MIN	463.4	587.2	239.0

Richard A. Berger  
28 Fawn Meadows Drive  
Eureka, MO 63025

## H/Z-100 Graphics Program

Dear HUG,

Here is a routine for H/Z-100 users interested in creating some of their own graphics. This routine as written is for the Epson MX-80 printer. A page of 'dots' is printed to represent the pixel pattern on the screen. All possible 640 horizontal scan lines cannot be put on a page but as written this routine will create a grid 232 x 225.

```
10 '
20 '
30 '   Written by: John C. Elam
40 '               Rt. 3, Box 96A3
50 '               Denton, TX 76205
60 '               817-497-3267
70 '
80 '   Creation Date: June 1, 1984
90 '   File Name: GRAPH.BAS
100 '
110 '
120 '   This routine will create a grid which can be used
130 '   by H/Z-100 users to lay out graphic designs. Each
140 '   dot on the page represents a pixel.
150 '
160 '   LPRINT CHR$(27)"A"CHR$(3);
```

```
170 '
180 '   print a ruler for 25 rows of 9 pixels (25x9=225)
190 '
200   LPRINT CHR$(27)"K";CHR$(204);CHR$(1);
210   FOR A=1 TO 10
220     LPRINT CHR$(0);
230   NEXT A
240   FOR B=1 TO 25
250     FOR A=1 TO 8
260       LPRINT CHR$(0);CHR$(1);
270     NEXT A
280     LPRINT CHR$(0);CHR$(127);
290   NEXT B
300   LPRINT
310   LPRINT
320   LPRINT
330 '
340 '   print a series of 8 pixel wide columns
350 '
360   FOR C=1 TO 29
370     FOR B=1 TO 7
380       LPRINT CHR$(27)"K";CHR$(204);CHR$(1);
390       FOR A=1 TO 10
400         LPRINT CHR$(0);
410       NEXT A
420       FOR A=1 TO 225
430         LPRINT CHR$(0);CHR$(1);
440       NEXT A
450       LPRINT
460     NEXT B
470     LPRINT CHR$(27)"K";CHR$(204);CHR$(1);
480     FOR A=1 TO 8
490       LPRINT CHR$(1);
500     NEXT A
510     LPRINT CHR$(0);CHR$(0);
520     FOR A=1 TO 225
530       LPRINT CHR$(0);CHR$(1);
540     NEXT A
550     LPRINT
560   NEXT C
570 '
580 '   reset the printer to power up configuration
590 '
600   LPRINT CHR$(27)"@"
```

John C. Elam  
Rt. 3, Box 96A3  
Denton, TX 76205

## A Simple Stepping Motor Control Experiment

Dear HUG,

The Microprocessor Trainer ET-3400 Instruction Manual (contains 19 experiments) and the Microprocessor Interfacing Manual (contains 10 experiments) do not have a stepping motor control experiment. The Robotics Manual of HERO I has a stepping motor control experiment (experiment number 14), it works well, but it seems there is still a gap between the keyboard and the motor.

We combined the data output experiment circuit<sup>1</sup> with 4 operation amplifiers to step up the output voltage to 12 volts, then connected to the the stepping motor<sup>2</sup> to have it in operation. The program is as the following:

```
000 86 03 LDA      00FF B7020F STAA
      BDO0FF JSR      CEFFFF LDX
      86 09          09      DEX
      BDO0FF          26 FD   BNE
      860C           39      RTS
      BDO0FF
```



(1) Microprocessor Student Workbook page 140

(2) Catalogue Number 420-625

Y. P. Hwu  
Bruce Jacobs  
Bradley Wilson  
James Whalen  
Claude Otey  
Wytheville Community College  
Wytheville, VA 24382

### OSCAR help needed!

Dear HUG,

Would you please insert the following plea for help in the "Buggin' HUG" column in the next available issue of REMark? Many thanks.

Does anyone out there have a program for tracking satellites in polar orbits, such as OSCAR or other radio amateur satellites? The language doesn't matter (BASIC, FORTRAN or Pascal), and a printout would be great. So far, all the programs I've found have been for geosynchronous satellites.

D.C. Shoemaker  
HQ USEUCOM Box 897  
APO NY 09128

### Help Needed With FORTRAN-80

Dear HUG,

I am having a slight, yet annoying problem with the FORTRAN-80 version 3.4. The problem is printing a line feed when writing to the disk. Consider the following segment of a program:

```

PI=3.1416
STEP=PI/20.0
TIME=0.0
100 IF(.NOT.(TIME.LE.PI)) GOTO 102
    ANGLE=SIN(TIME)
    WRITE(1,50) ANGLE, TIME
    WRITE(7,50) ANGLE, TIME
50  FORMAT('X',2F10.5)
    TIME=TIME+STEP
    GOTO 100
102  CONTINUE

```

where LUN 1 is the CRT and LUN 7 is the disk file. Each time the variables are printed to the screen, they are preceded by a carriage return and a line feed, which makes an orderly, line-by-line printing. However, when the variables are written to the disk file using the same format statement, the carriage return is present, the line feed is not present, and the character "X" appears in the file. The result is a file which cannot be sent to a line printer without first being edited. A screen dump of the disk file reveals carriage returns (hex 0D) but no line feeds. I have unsuccessfully tried various format specifications with and without carriage control characters in attempts to send the line feeds. Do you have any tips on how I might force the line feed to appear in the disk file?

William E. Crocker  
RR #1  
Scheller, IL 62883

### The ILLUSTRATOR

The ILLUSTRATOR is a full-featured graphics drawing program for use with the Z-100 pixel graphics (w/w/o color) or the H89/H19 IMAGINATOR pixel graphics option. No need for a lightpen.

#### Features Include:

- Turtle Graphics
- Rubber Banding
- Line Drawing
- Box Drawing
- Circle Drawing
- Ellipse Drawing
- Diamond Drawing
- Screen Print
- Dot Cursor Control
- Area Fill
- Box Fill
- Circle Fill
- Diamond Fill
- Copy Area
- Erase Area
- Text Mode
- Invert Mode
- Help Display
- 64 Colors
- Color Define
- Color Pattern
- Screen Save
- Screen Restore
- Area Save
- Area Restore
- Compacted Files
- more...

ONLY \$89.95!

HDOS version for H89, H8, requires IMAGINATOR  
ZDOS version for Z-100, color memory optional  
Supports many dot graphics printers. Call for info.

### NEWLINE SOFTWARE

P.O. Box 402, Littleton, MA 01460 (617) 486-8535

NAME _____	CHECK ONE
STREET _____	<input type="checkbox"/> H89, H8/H19, HDOS
CITY _____	<input type="checkbox"/> Z-100, ZDOS
STATE _____ ZIP _____	
Send me _____ "The ILLUSTRATOR" program(s) at \$89.95 each.	
Check one: <input type="checkbox"/> payment enclosed <input type="checkbox"/> send COD (add \$4.00)	
Send order to:	
NEWLINE SOFTWARE, P.O. BOX 402, LITTLETON, MA 01460	
Foreign orders: add \$3.00 Airmail, \$10.00 for non-U.S. checks	

HDOS is a trademark of Heath Company  
ZDOS, Z-100 are trademarks of Zenith Data Systems, Inc.  
IMAGINATOR is a trademark of Cleveland Codonics, Inc.

### The Software Toolworks presents: MYCALC™

Full featured MyCalc with sort, bar graphs, multiple files and more is ideal for financial planning and budgets. It sells for \$59.95 from The Software Toolworks, 15233 Ventura Blvd., Suite 1118, Sherman Oaks, CA 91403 (818) 986-4885.

## A Plea For National Standards For RBBS

Dear HUG,

After purchasing a modem, and spending a few hours on working with the software to run the modem, along with contacting several bulletin boards in Cincinnati, I find a great need to reform the RBBS and user modem software systems.

The need to reform the whole system would bring cries of intervention with the individual rights, but I'm darn tired of finding numerous single and multiple letter codes that are different on each bulletin board.

When you switch between different RBBS's, it can only drive you nuts to remember which one is which now for operational codes. What is needed is a National Standard, or suggested standards that will help everyone.

I hope that both the REMark staff, and the HUG members would try to contact other clubs and organizations to get something started.

The software I purchased to make the system work, although not expensive, was aimed at using Compuserve and generally only talked about how to contact Compuserve. There was no information on how to save or transfer files, or what you could expect when connecting to an RBBS. As I am sure you are well aware of, some systems require several hits of the CR key to start the sequence of events, and some do not. This would have been helpful for a beginner to know, for I sat there several times, for several minutes wondering what to do. One system I sent a CR to several times did not like it at all, and I got a firm note several days later from the SYSOP not to do that again, please.

Do we need Standards? We sure do! Do we need better written books on how to operate the systems? We sure do!

Bert Rathkamp  
5950 Park Road  
Cincinnati, OH 45243

## H-8 & H-89 Color Graphics

Dear Walt,

Just a short note to let the HA-8-3 and HA-89-3 graphics users know that all of the questionnaires have been entered into a data base. When compared to the total graphics boards out there the list is fairly small - but growing. Based on my contacts the list does seem to represent the more active graphics users.

The following data, if provided, is listed for each person: name; address; home and work phone; primary and secondary operating system; printer(s); type of computer; disk drives; languages used; and graphics board options such as AMD9511, Vortrax, DACS, joysticks, etc. Three lines of free text are also provided for areas of interest and other information which did not fit into the formatted fields.

For ease of use the data is output to disk print files in three sorts: last name, ZIP, and state orders. These files are formatted for direct output to the printer. Page headings and blank lines for the fanfolds are included. With the amount of data included in the file there are four people listed on each page. Each file is currently 13K bytes.

For a copy of the files send a formatted disk and \$1.00 for postage. I can support the following formats: HDOS and CP/M 55 SD 40 track,

DS SD 80 track, and CP/M DS DD 40 track. Just send a blank soft sector disk. If you want to be included in the list send me the info listed above.

Best Regards,

Fred Pospeschil  
3108 Jackson Street  
Bellevue, NE 68005

## Byorhythm leap years

Dear HUG,

In 1582 Pope Gregory the thirteenth decreed that years ending in hundreds should not be leap years unless they are divisible by 400. Thus, Mr. Baileys correction to the Biorythm program for leap years (REMark, June 1984) will fail in the year 2100. The problem can be corrected in the following way.

```
5130 IF (INT(Y1/4)*4)=Y1 THEN 5131
5131 IF (INT(Y1/100)*100=Y1 AND
      (INT(Y1/400*400) <> Y1) THEN 5140
```

I think you will notice a measureable improvement in accuracy after this correction is made.

Ralph L. Seiler  
70 E. Sunset Avenue #1  
Salt Lake City, UT 84115

## Article Correction

Dear Walt,

When I read through my article, "Terminal Control with H/Z-100 Using MS: FORTRAN and Pascal" (June 1984), I noticed a rather serious typographical error. In listing 4 on page 33, the macro START contains the statement POP DS. It should not be there; instead it should be in the macro FINISH. The corrections are as follows:

```
START MACRO
      PUSH      BP          ;Save caller's frame pointer
      . . . . .
      MOV       DS,AX      ;Get address of message
      . . . . .
      . . . . .
FINISH MACRO
      POP      DS          ;Restore caller's Data segment
      POP      BP          ;Restore caller's frame pointer
      RET
      ENDM
```

I apologize to the readers for any inconvenience these errors may have caused them.

Sincerely yours,  
M. Manivannan  
Department of Chemical Eng.  
Clarkson University  
Potsdam, NY 13676

**ZBASIC Re-seeder**

Dear HUG,

The enclosed listing is an adaption of a random number generator re-seeding routine I used in a card sorting program. It re-seeds with a random number depending on the time key F1 is struck. It does require the user to respond to on-screen instructions but maybe Mr. Harvey (Buggin' HUG, June 84) can use it for his classes. I included five "attaboy" statements but any number could be used by changing the size of the RMK\$ array and changing line 140 accordingly.

Has anyone figured out how to make ZBASIC accept the fact that some machines have more than 128K of memory yet? I have some graphics routines which require large arrays and the furshlugginer language won't let me have them even though there is 64K sitting there unused.

Sincerely Yours,

Robert D. Williams  
100 Garrett Drive  
Hampton, VA 23669

```
10 'ROUTINE TO AUTOMATICALLY RESEED RANDOM NUMBER
    GENERATOR EACH TIME USED
20 'AND PRINT A COMMENT BASED ON THE NUMBER RETURNED
30 DIM RMK$(5)
40 RMK$(1)="ATTA BOY"
50 RMK$(2)="DO YOU WANT TO TRY AGAIN?"
60 RMK$(3)="SORRY, YOU GOOFED. TRY AGAIN."
70 RMK$(4)="WOW!—YOU'RE A REAL PROFESSIONAL."
80 RMK$(5)="KEEP UP THE GOOD WORK"
90 KEY (1) ON
100 ON KEY (1) GOSUB 170
110 PRINT "PRESS FUNCTION KEY F1 TO CONTINUE"
120 Y=INT(RND*32767)
130 GOTO 120
140 N = CINT(RND*5)
150 PRINT RMK$(N)
160 END
170 KEY (1) OFF
180 RANDOMIZE (Y)
190 RETURN 140
```

**Likes Spreadsheet Corner**

Dear Walt,

I just read "Spreadsheet Corner, Part I" in the July issue and am REALLY excited about this new feature. It's just what I've been

looking for. Thanks to you and H.W. Bauman. Keep up the good work.

Sincerely,

Jim O'Brien  
1915 Labrador Lane  
Vienna, VA 22180

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

maybe now is the time to join the National Heath/Zenith Users' Group. You will receive:

- a copy of **REMark** filled with new and exciting articles and programs each month
- access to the **HUG** library filled with a large variety of programs
- discounts on a variety of Heath/Zenith computer products (see **REMark** January, 1984 issue for more details)

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

*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

HUG ID Number: \_\_\_\_\_

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	\$20 <input type="checkbox"/>	\$17 <input type="checkbox"/>
CANADA	\$22 <input type="checkbox"/>	\$19 <input type="checkbox"/> US FUNDS
INTERNAT'L*	\$30 <input type="checkbox"/>	\$24 <input type="checkbox"/> US FUNDS

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

You don't have to be a WESTERNER to attend  
the

## Western Regional HUG Conference

Speakers • Panels • Displays • Prizes

Grand Prize: H/Z-100 Computer with Winchester Drive\*

**November 10th & 11th, 1984**

**Disneyland Hotel • Anaheim, California**

The Official Hotel At The Magic Kingdom

Write for Registration Information to:

Conference Coordinator  
1555 North Orange Grove Avenue  
Pomona, California 91767

\* You must be present to win.

### **Index of Advertisers**

Advanced Software Technologies .....	56	Jay Gould Software .....	48	Software Support .....	18
C.D.R. Systems Inc. ....	30,42	Kres Engineering .....	48	Software Toolworks .....	48,65
Del Soft .....	41	Micro Services .....	42	Software Wizardry .....	12
D-G Electronic Development Co. ....	2	Newline Software .....	42,65	Studio Computers .....	6
Generic Software .....	56	Northwest Computer Algorithms .....	47	TMSI .....	22
Headware .....	48	Quick Data .....	4	Variousware .....	47
Paul F. Herman .....	61	Secured Computer Systems .....	26		



Hilltop Road  
Saint Joseph, Michigan 49085

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

**Volume 5, Issue 8**

**POSTMASTER: If undeliverable,  
please do not return.**

**885-2055**