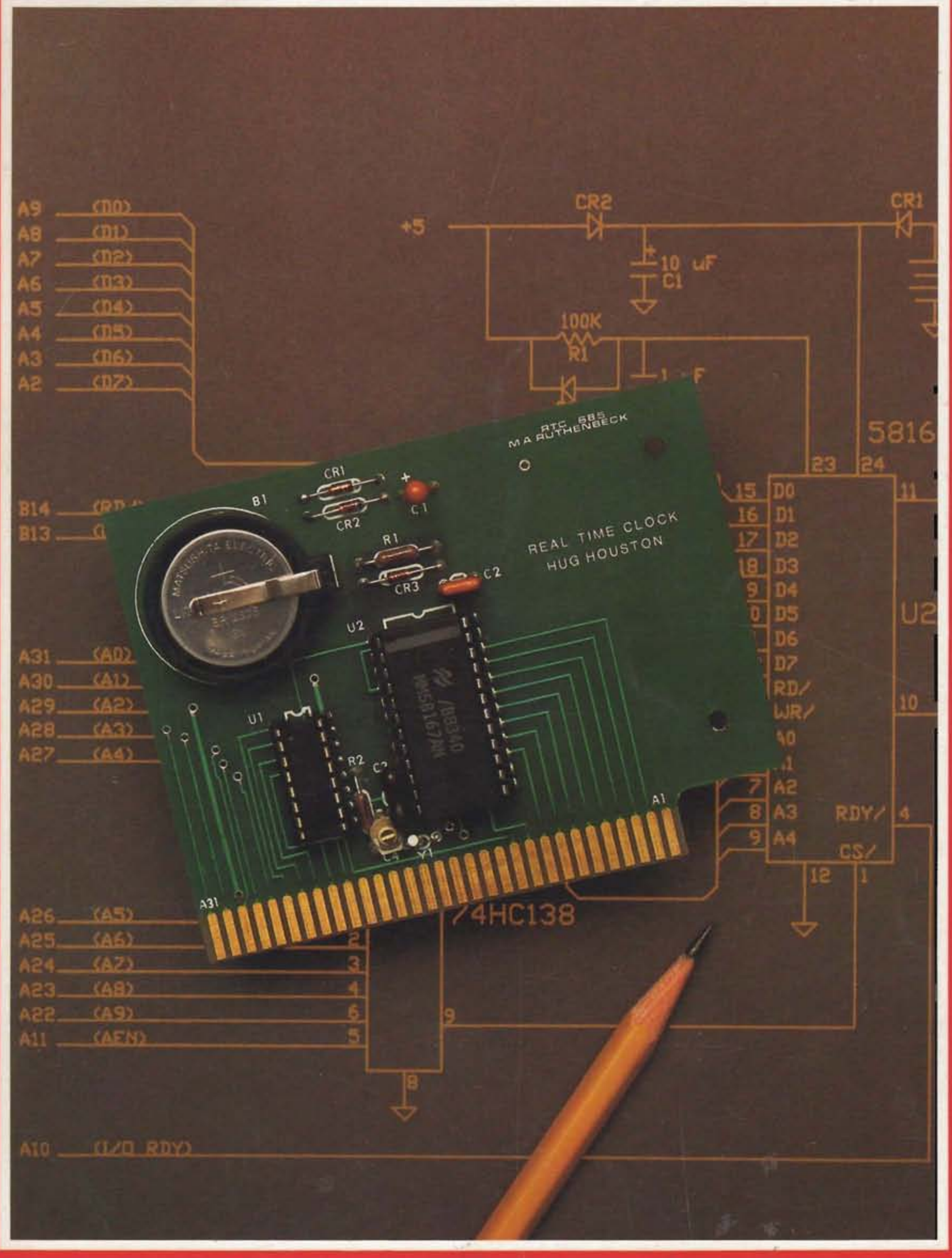
REMark®

Volume 6, Issue 11 • November 1985
P/N 885-2070 Issue 70

Official magazine for users of HEATH ZENITH computer equipment.

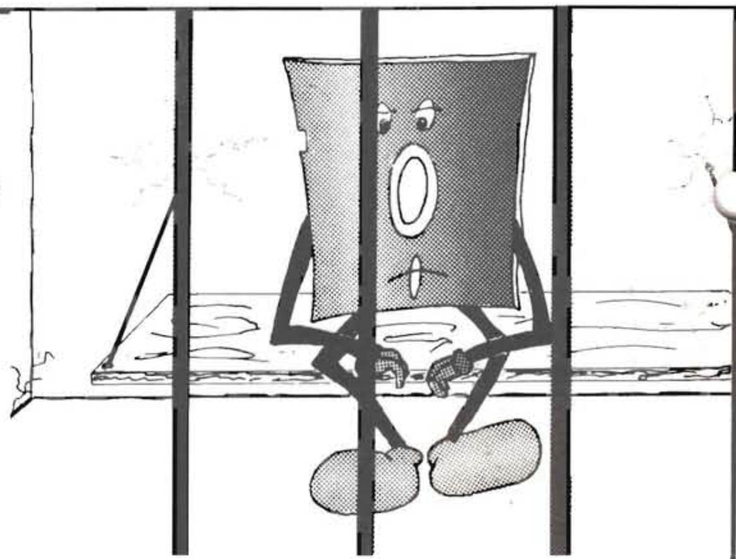
$2.50

Real Time Clock
HUG Houston, See Page 57



micro/VERSAL™

the complete file transfer package For IBM-PCs and Zenith Z100s



Free your data with micro/VERSAL™

With micro/VERSAL™ you can READ, WRITE and FORMAT any of the following formats:

- 8 inch standard (SSSD)*
- Avatar TC1 (SSDD)
- Avatar TC1 (DSDD)
- Cromemco CDOS - I (SSSD)*
- Cromemco CDOS - II (SSDD)
- Cromemco CDOS - II (DSDD)
- Datavue (SSDD)
- Datavue (DSDD)
- Epson QX-10 (SSDD)
- Epson QX-10 (DSDD)
- HDOS (SSSD)*†
- HDOS (SSDD)
- HDOS (DS00)†
- Heath With Magnolia (SSDD)
- Heath With Magnolia (DSDD)
- Heath With Magnolia (SSSD)*
- Heath With Magnolia CP/M 2.24 (SSDD)
- Heath With Magnolia CP/M 2.24 (DSDD)
- Heath With Magnolia CP/M 2.24 (SSSD)*
- IBM-PC/CPM-86 (SSDD)
- IBM-PC/CPM-86 (DSDD)
- Kaypro II (SSDD)
- Osborne (SSSD)*
- Osborne (SSDD)
- Osborne (DSDD)
- Osborne (Ozmosis SSDD)
- Osborne (Ozmosis DSDD)
- Radio Shack TRS-80 CP/M (SSDD)
- Radio Shack TRS-80 CP/M (DSDD)
- Radio Shack TRS-80 CP/M (SSSD)*
- Radio Shack TRS-80 I DOS (SSDD)*†
- Radio Shack TRS-80 DOS 1.3 (SSDD)*†
- Radio Shack TRS-80 DOS 2 (SSDD)*†
- Radio Shack TRS-80 I DOS (SSSD)*†
- Radio Shack TRS-80 NEWDOS (SSSD)*†
- Lobo MAX-80 (SSDD)
- Lobo MAX-80 (DSDD)
- Morrow MD2 (SSDD)
- Morrow MD3 (DSDD)
- Morrow MD11 (DSDD)
- NEC PC (8001) (SSDD)
- NEC PC (8031) (SSDD)
- NEC PC (8031) (DSDD)
- Sanyo (SSDD)
- Sanyo (DSDD)
- Superbrain 2*
- Televideo (SSDD)
- TI Professional (SSDD)
- Xerox 820-I (SSSD)*
- Xerox 820 (SSDD)
- Xerox 820 (DSDD)
- Zenith Z-100/CPM-85 (SSDD)
- Zenith Z-100/CPM-85 (DSDD)
- Zenith Z-90 (SSDD)
- Zenith Z-90 (DSDD)
- Zenith Z-90 CP/M 2.24 (SSDD)
- Zenith Z-90 CP/M 2.24 (DSDD)
- Zenith Z-89 (SSSD)*

*Not available on PC version
† Formatting option not supported

(48 PTI SOFT-SECTORED OR 8 INCH*)

micro/VERSAL™ is easy to use – menu driven, and requires only 64K and 2 floppy drives or 1 floppy and a hard disk; runs under both DOS and DOS2.

You also get utilities to help you transfer your files:

- **ANYCPM™** – define your own CP/M formats for machines not listed on our formats menu.
- **SED** – a disk sector editor lets you examine and modify any byte on any disk, even CP/M disks.
- **COPY/FORMAT** disks to your own specifications.
- **RECOVER** accidentally deleted files.
- Utilities to modify files to **MS-DOS** standards from other operating systems.

only micro/VERSAL™ gives you EVERYTHING YOU NEED FOR ONE LOW PRICE:

$49.95

plus $4.00 shipping & handling

Send check, money order, M/C or VISA numbers to:

ADVANCED SOFTWARE TECHNOLOGIES

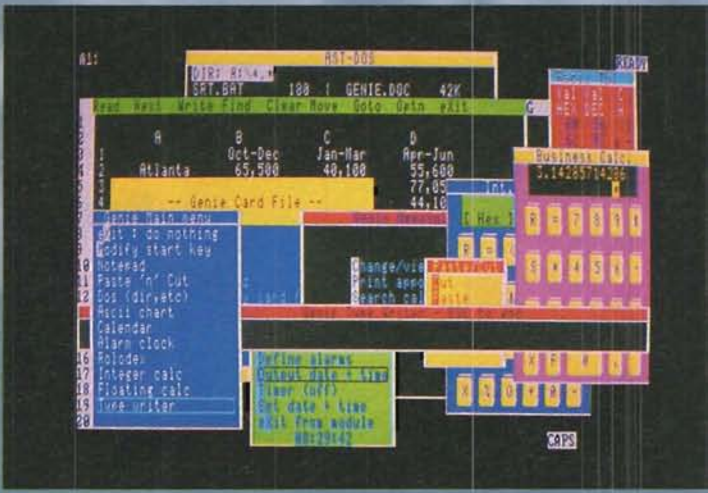
**452 West 47th Street
New York, NY 10036
(212) 247-0150**

Also from AST is our COED full screen editor, with macro commands, user Definable function keys, UNDELETE, Multiple files, built in calculator, on disk tutorial, and much more. COED is FAST, works with BASIC programs and has NO LIMIT on file size . . . Lose those EDLINE blues **$34.95**

NOT PROTECTED

Genie®

Software Magic from Advanced Software Technologies



Shown here is Genie "popped up" on a Z-110 running Lotus 123. From the left are: The Genie main menu, the Genie rolodex style card file, the Genie notepad containing data cut from Lotus, the Genie DOS performing a directory command, the Genie alarm clock (at the bottom,) the Genie typewriter, Genie calendar, Genie Cut and paste, Genie Calculators, and the Genie Ascii table.

Genie

	Sidekick	Perks	Genie
Key Macros	+69.95	No	**YES**
Dos Services	No	Yes	**YES**
Notepad	50K	4K	**No Limit**
Ascii Table	Yes	Yes	**YES**
Calculators	One	One	**TWO**
Typewriter	No	No	**YES**
Calendar	Yes	Yes	**YES**
Rolodex	Yes	No	**YES**
Cut and Paste	No	No	**YES**
Alarm	No	Yes	**YES**
Expandable	No	No	**YES**
	$120	$100	**$49.95**

Special Introductory Offer:

$49.95

plus $4.00 shipping and handling
New York residents add 8.25% sales tax

To order send MC/VISA check or M.O. to:

ADVANCED SOFTWARE TECHNOLOGIES

452 West 47th Street
New York, NY 10036
(212) 247-0150

Genie is completely resident so that once you load it, it is always there. A single key stroke lets you command the powers of Genie. Have Genie take notes, schedule appointments, move data between your applications, perform calculations, and much more. When you finish, Genie goes away and waits for your next command.

Genie is available for the Zenith Z-100 and IBM-PC compatibles. Genie requires MS-DOS version 2 or above.

Heath/Zenith Users' Group

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
Editorial and Advertising Assistant	Lori Lerch (616) 982-3794
Printers	Imperial Printing St. Joseph, MI

	U.S. Domestic	APO/FPO & All Others
Initial	$20.00	$35.00*
Renewal	$17.00	$30.00*

* U.S. Funds

Limited back issues are available at $2.50, plus 10% shipping and handling — minimum $1.00 charge. 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

REMark®

Official magazine for users of HEATH / ZENITH computer equipment.

Volume 6, Issue 11 • November 1985

Index of Advertisers

This index is provided as an additional service. The publisher does not assume any liability for errors or omissions.

21st Century Data, Inc.	14
Advanced Software Technologies	2,3
Analytical Products	38
CDR Systems, Inc.	26
Consumer Product Development	20
DelSoft	51
Diverse Systems	62
EDL Products & Services	73
FBE Research Company, Inc.	62
FINA Software	14
Future Communications	25
Gemini Tech	45
Generic Computer Products, Inc.	6,73
Paul F. Herman	80
Hilgraeve, Inc.	82
Intersecting Concepts	34
M.E. Pittman	51
McGaffey Engineering	56
Redwood Development	81
Ross Custom Electronics	62
S&K Technology	81
Secured Computer Systems	33
Software Toolworks	56
Studio Computers	46
Systems Innovations Inc.	8
UCI	52
Veritechnology Electronics Corp.	74
Barry Watzman	84

On The Cover: The Real Time Clock circuit board from Heath Users' Group Houston. See how you can make one yourself. Article on Page 57.

G.C.P.I.

GENERIC COMPUTER PRODUCTS, INC.

COMPUTER PRODUCTS YOU CAN RELY ON

P.O. BOX 790 · DEPT. 115R
MARQUETTE, MICH. 49855

HOLIDAY SALE

. . . drawn-up especially for YOU!

Add $3.00 for shipping/handling for all orders and 4% sales tax for Michigan residents. Specify computer brand, model, operating system and disk format when ordering.

For Faster Delivery
Call 906-249-9801

ORDER NOW . . . CLIP A COUPON AND SAVE. LIMITED TIME OFFER!

$39.95 NEW! **PERSONAL TASK MANAGER (P.T.M.)**

Now you can get more done in less time! PTM is a task organizer and time management tool designed to help you get organized. Save $10 by purchasing PTM now, and get organized today! Available for IBM-PC and compatible systems.

$125 **TURBO GIFT-PAK**

- ★ **TURBO-PASCAL V3.0**
- ★ **TURBO TUTOR**
- ★ **TURBO DATABASE TOOLBOX**

All 3 for only
$125
You save over $30!
Available for most PC-DOS, MS-DOS, and CP/M-80 computer systems.

$129.95 **SIDEKICK/ SUPERKEY COMBO PAK**

Two great utility programs for your IBM-PC or compatible at incredible holiday savings! In addition, Borland will give you a cash rebate of $15.00. You save $40.00 off the regular price!

$69.95 NEW! **TURBO GAMEWORKS** (from Borland)

- ★ **CHESS**
- ★ **BRIDGE**
- ★ **GOMUKU** *Japanese Tic-Tac-Toe)*

All 3 games, including source code, only
$69.95
Available for IBM-PC and compatibles.

20% OFF **HOLIDAY SPECIAL SALE COUPON**

See something else in the catalog you want, but you don't see a special sale coupon for it? Well, no problem! Take 20% off the list price of any product you see listed in our catalog.
The 20% must be applied to the original list price of the item you want. No other discounts apply.

$39.95 NEW! **PERSONAL FINANCE MANAGER (P.F.M.)**

Now you can get your finances organized into a well-maintained budget. P.F.M. is designed to keep track of your expenses and provide summary reports for personal and tax purposes. Available for the IBM-PC and compatible systems.

$39.95 **FOOTBALL SPECIAL**

Get our popular, fast action arcade-style FOOTBALL game and our point-spread program, FOOTBALL-PICKS. With the NFL in full swing, you'll want both of these favorites! Buy now and save $10.00 off the regular price. Available for HDOS, most CP/M-80 computers, the Z-100, and IBM-PC and compatibles.

$99 **FINANCIAL-PAK**

- ★ **Amortization and Loan Analysis**
- ★ **Lump Sum and Annuity Analysis**
- ★ **Stock and Mutual Fund Buy/Sell Aid**

All 3 for only
$99
Now get the most commonly needed financial software at the lowest price! Available for most PC-DOS, MS-DOS, and CP/M-80 computer systems.

$59.95 **IBM-PC ENTERTAINMENT SPECIAL**

I-BERT
the Q-bert-like Arcade Game
LEM99
Lunar Lander Flight Simulation
CHARADES
Popular Party Game
WORDSPELL
Wheel-of-Fortune-Like Word Game
All 4 games only
$59.95

$49.95 **FLOPPY-DISK ORGANIZER-PAK**

ARCHIVE - Allows you to back up groups of files under one file name. Also compresses information to save disk space!
CATALOG-MASTER - Now get a master directory for all your floppies on one report. Finding any file is now fast and easy!
Get both of these popular disk utilities now and save $25 off the regular price. Available for most PC-DOS, MS-DOS and CP/M-80 computer systems.

$39.95 **C.A.K.E. (Cursor & KEypad Editor)**

However you slice it, our CAKE full-screen editor program is one of the easiest to use. So if you don't want to spend a lot of time to learn a complicated word-processing package with difficult to remember commands, then try a piece of our CAKE. And to put the frosting on the CAKE, save $10 off the regular price by getting it now! Available for Heath/Zenith CP/M-80 systems, Z-100 systems, and IBM-PC and compatible systems.

$49.95 **HEATH/ZENITH CP/M-80 ENTERTAINMENT SPECIAL**

HAUNTED-CAVE
Graphics Adventure Game for any age
CASTLE-GOR
Challenging Text Adventure Game
SPACE-WARRIOR
Arcade-Style Game
All 3 games only
$49.95

BUGGIN' HUG

On The Air With The Z-100

Dear HUG:

After having built two H-89s and operated Radio Teletype on both, I have missed that mode of communications on my Z-100s that replaced the 89s. So, in response to my own needs and those of a few others that would like to get on the air with their Z-100, I have written an rtty program I call ZRTTY. This is a compiled BASIC program with linked assembly language routines for speed in screen handling.

Some of the features of the program are:
* Single key stroke control of all functions.
* Dual, variable size windows for transmit and receive data.
* Disk control for transmit and/or receive data.
* Printer control for transmit and/or receive data.
* Four transmit buffers.
* On screen status indicators of all functions.
* ASCII and Baudot codes supported.
* Transmitter key control lines from modem port.
* Auto update of date and time at transmit time.
* Color support and options.

Hardware required is a Z-110 or Z-120 computer, 128K of RAM, ZDOS or MSDOS. Any radio operators interested in the program or more information may contact me by mail. 73's

Robert McGowan KD7GP
8096 Meridian Road NE
Silverton, OR 98781

Difficulty With WordStar Filters

Dear HUG:

I've received several letters from readers who experienced some difficulty with the two programs listed in my article "WordStar Filters For The Z-100 And Z-100 PC" in the September 1985 issue of REMark.

The symptoms reported were failure of the programs to run properly, accompanied by excessive seeking or "chunking" of the disk drive heads.

The symptoms, after some experimentation, proved to be the result of an I/O problem in MS-DOS — apparently, the default assignment by DOS of only 2 disk buffer sectors on bootup is inadequate for this type of program which uses character by character I/O.

The solution is simple — and highly recommended to speed up all disk I/O under MS-DOS 2.xx — put a CONFIG.SYS file on the boot disk to assign additional buffers. Those readers who may not be familiar with the use of CONFIG.SYS to add to the standard MS-DOS system component, should refer to page 9.19 of the MS-DOS Version 2 manual. There is no "right" number of buffers to add, because the optimum number of buffers will vary depending on the type of program being run, but based on my own "seat-of-the-pants" experience, I'd recommend BUFFERS = 10 for systems with 192K or less of RAM, and BUFFERS = 30 for systems with additional RAM. Each buffer consumes about 1040 bytes of RAM on the H/Z-100 and 528 bytes of RAM on the H/Z-100 PC.

John F. Smith
239 Colonial Drive
Warminster, PA 18974

Baud Rate Solution On The H-8

Dear HUG:

I don't know how many people would be in this situation, but I thought I would send this anyway. I bought an H-8 with the H-8-5 serial cassette interface when Heath was clearing them out. Later I purchased an H-29 terminal. This terminal can operate at 19,200 baud, which is great when using a spreadsheet or word processor. The serial card uses hard wire baud rate selection with a maximum of 9600. I wanted to use 19,200, and my solution is as follows.

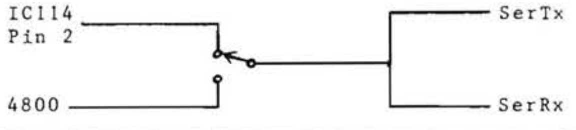
First I had to find the right frequency on the board. Among other places, it is available at pin 2 of IC114 (74LS161). I connected SerRx and SerTx with a jumper, and then soldered a wire on the back of the board from SerRx to pin 2 of IC114. I set the terminal for 19,200 baud, booted SuperCalc. The screen fills in a flash. You have to see it running a program to appreciate the difference.

A week later, I put Y-wing Fighter in and started to play. Everything was fine until I pressed a key. I think I entered hyperspace; the screen filled with garbage, the bell sounded, etc. I discovered graphics games will not work at the higher baud rate. Incidentally, they do not seem to work at 9600 either.

To get around this problem, I installed an SPDT switch on the rear of the computer. One pole is connected to pin 2 of IC114, one pole is connected to the 4800 baud holes, and the common pole is connected to SerRx and SerTx (see Figure). When I use text programs (spreadsheets, etc.) I position the switch for 19,200, set up the terminal for 19,200, and boot the program. When I use graphics programs, I set everything for 4800. This hasn't failed in any program that I have used.

Sincerely,

Michael Eldridge
5498 Tradewinds Walk #1
San Jose, CA 95123



Invent Celestial Objects Of Your Own

Dear Walt:

I was very pleased to see that "Halley's Comet" article was published in the September 1985 issue of REMark. After sending the article to you, it occurred to me that readers may want to experiment by adding a celestial object of their own (either real or invented) to the planets already listed. The simple instructions are below, should anyone like to incorporate them into the arti-

Continued on Page 83

SII VALUE BULLETIN

STRIKE FORCE!

ZF241
- AT-Compatible
- 80286 Processor
- 6 Mhz Clock
- Single 1.2MB Floppy
- 20MB Winchester
- 512K RAM

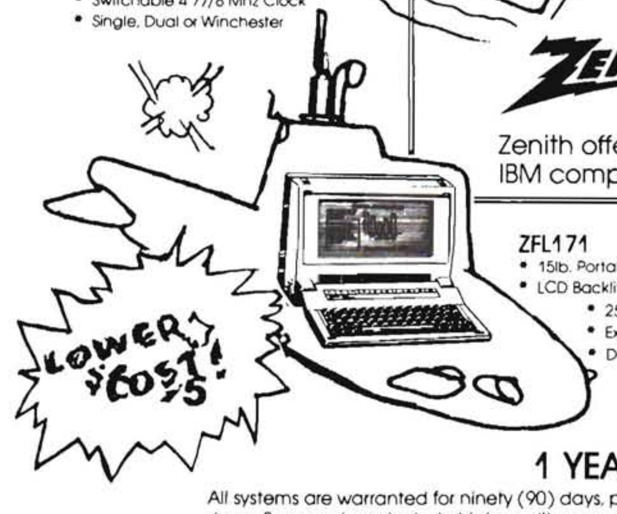
Once again, Systems Innovations Inc. and Zenith Data Systems break thru the lines with Service, Support & Spectacular Value.

ZF148
- 22lb. Desktop
- 256K RAM
- Expandable to 768K
- Switchable 4.77/8 Mhz Clock
- Single and Dual 5¼" Drives

ZF158
- Desktop PC
- 256K RAM
- Expandable to 640K
- Switchable 4.77/8 Mhz Clock
- Single, Dual or Winchester

ZENITH | data systems

Zenith offers you the widest array of IBM compatibles on the market today.



ZFL171
- 15lb. Portable
- LCD Backlit Display
    - 256K RAM
    - Expandable to 1 Megabyte
    - Dual 5¼" Disk Drives

ZF138
- 24lb. Transportable
- 7" CRT Display
- 256K RAM
- Expandable to 768K
- Switchable 4.77/8 Mhz Clock
- Single and Dual 5¼" Drives

1 YEAR FREE SERVICE

All systems are warranted for ninety (90) days, parts, labor and for one (1) full year free service (labor) upon return to us. Some systems include high quality components from third party manufacturers. These components are fully warranted by us for ninety (90) days from date of purchase and should be returned to us for service. Factory Service Centers may decline to service equipment containing third party components. If you suspect equipment malfunction, please call our Technical Support Department.

SYSTEMS INNOVATIONS INC.

HOURS: 9AM-6PM, MON.-FRI. • THURS. TIL 8PM • SAT. 10AM-3PM

44 Stedman Street,
Lowell, Massachusetts 01851
(617) 459-4449
"Your Complete Computer Resource Center"

VISA MasterCard AMERICAN EXPRESS

Upgrade Your CP/M 2.2 With ZCPR3 A Z80-Based Command Processor

Rick Swenton
19 Allen Street
Bristol, CT 06010

Introduction

What is ZCPR3? Technically, ZCPR3 stands for Z-80 Command Processor Replacement version 3. More specifically, it is an enhanced replacement Console Command Processor (CCP) for CP/M 2.2. It is written in Z-80 code so it can pack more features into the same restricted space of 2K bytes than the original CCP occupied. Because it is written in Z-80 code, it will not run on the H-100 which uses the 8085 CPU, nor will it run on the H-8 unless it has the Z-80 upgrade CPU board installed. Included in the ZCPR3 package is a collection of programs and system files which have been integrated tightly together to form a powerful "unified" operating system. ZCPR3 comes on fourteen eight-inch disks and the source code is included! There are over seventy COM files and over one hundred available commands with more added each day by the dedicated and enthusiastic ZCPR3 users who contribute new and updated programs into the distribution network. More on this later. ZCPR3 and the majority of its programs and documentation are copyright by Echelon, Inc., but may be obtained and used free of charge for non-commercial purposes. Further on in this article, I will explain the various ways to obtain ZCPR3. ZCPR3 and its utilities were written by a very brilliant man: Richard Conn. Many of you are familiar with ZCPR3's predecessor, ZCPR. ZCPR is a good enhancement to Digital Research's standard CCP. But ZCPR is "kid's stuff" compared with ZCPR3! If you have implemented Ray Livingston's BIOS-80 to run high capacity disk drives with the H17 controller, you were given the option to include ZCPR with your installation. Once you did, you could never go back to the stone ages. This is how ZCPR3 is to ZCPR1. Once you use it, you can never go back to plain old ZCPR. ZCPR3 gives your CP/M compatible system many of the advanced features found in UNIX and MS-DOS.

Overview Of Features

To the casual user, ZCPR3 will look and operate just like CP/M 2.2. It is downward compatible with CP/M 2.2 and 99% of all programs, which ran on your old CP/M system, will run under ZCPR3. Some highly specialized programs which poke directly into the CCP like SYNONYM.COM will not run and may crash the system. I have experienced problems running a .COM file from Wordstar and not being able to return to Wordstar automatically. I haven't looked into this yet. Most of the ZCPR3 utilities can not run under plain CP/M 2.2, because of the special memory setup required by ZCPR3.

Here are some of the features added to your system by ZCPR3:

1. Multiple commands are allowed on a single line.
2. Enhancement of the directory.
3. Enhanced command processing.
4. Enhanced command search and directory search path.
5. Resident commands.
6. Flow commands.
7. Shells.
8. Environment Descriptor.
9. Over 100 commands are available.
10. There are over 200K of on-line HELP files available which, along with over 70 .COM files are included in the 14 disk set!

I'll try to explain each of these as simply as I can, and I'm sure that because of the vastness of ZCPR3 I will omit something. I learn something new about ZCPR3 each day that I use it and it's fun!

Installation

In order to understand the ZCPR3 system, you must first understand a little about how it is installed. There are two ways to install ZCPR3: One is the manual method, which requires a good understanding of assembly language. You must downsize your existing CP/M to reserve about 4k at the top of your memory. You must modify the BIOS to initialize this memory when you boot your system, and you must integrate the new ZCPR3 command processor with the existing CP/M. As you can see, this is no job for a novice. Even the experienced CP/M systems programmer will have to spend many frustrating hours pondering over the supplied documentation. The second method is to purchase an auto-install version from Echelon, Inc. This company is the official commercial source of ZCPR3 products and software support. The auto-install version is pre-configured to take advantage of all but one of the ZCPR3 features. This version costs about $150, but is worth every penny and more.

You may ask "What if I know a little about assembly language programming and a little about CP/M. Would I be able to install the manually installed version of ZCPR3?" Probably not. You need to be comfortable working in the BIOS. You also need to fully understand MOVCPM, SYSGEN, MAKEBIOS and DDT. In addition, you will need Digital Research's Macro Assembler MAC and Debugger ZSID, and a text editor. A public domain program called RELS.UTL by Robert Van Valzah may be required depending on the revision level of some of the files. Installation is quite a task, but it's well worth the effort. So, as you can see, there's no "in between". Either you are really proficient in assembly language or you purchase the auto-install version. The clear advantage of the auto-install version is that all the work is done for you and you may obtain professional consultation and assistance from Echelon. They have experience installing ZCPR3 on many systems, including the H-8 and H-89.

Figure 1 shows a comparison of my system memory map before and after I installed ZCPR3. The memory sizes shown can vary depending on how many options were selected.

```
 64K    Standard CP/M 2.2               ZCPR3 System
 TOP->--------------------- ----------------------------
   5K ¦ Standard H/Z BIOS ¦¦ ZCPR3 Resident Packages   ¦4K
      --------------------- ----------------------------
 3.5K ¦ CP/M 2.2 BDOS     ¦¦ User Modified H/Z BIOS    ¦5K
      --------------------- ----------------------------
   2K ¦ CP/M 2.2 CCP      ¦¦ CP/M 2.2 BDOS             ¦3.5K
      --------------------- ----------------------------
      ¦ Transient Program ¦¦ ZCPR3 Command Processor   ¦2K
      ¦       Area        ¦----------------------------
      ¦                   ¦¦ Transient Program         ¦
      ¦                   ¦¦         Area              ¦
      ¦                   ¦¦                           ¦
      ¦                   ¦¦ (4K smaller in my system) ¦
 ~54K ¦                   ¦¦                           ¦~50K
      ¦                   ¦¦                           ¦
      ¦                   ¦¦                           ¦
      ¦                   ¦¦                           ¦
100H-->--------------------- ----------------------------
      ¦CP/M Buffers       ¦¦CP/M and ZCPR3 Buffers     ¦
  0H-->--------------------- ----------------------------
```

Figure 1. Comparison of standard CP/M 2.2 and ZCPR3 Memory Maps

Now, let's talk about specific memory locations. I will define how I organized my memory, and later I will explain what each area was used for. Figure 2 shows my ZCPR3 memory map in more detail. The most detailed locations are those shown above the BIOS. They are located in that 4K area which was reserved above the BIOS by running the command MOVCPMxx 60 where xx is either 17,37 or 47 depending on the type of Heath disk system you are using. This will create a 60K CP/M system. The new CP/M will never attempt to load programs in the upper 4K of memory in your system. The ZCPR3 utilities will take care of utilizing the reserved memory.

```
Address
 FFFF   -------------------------------------------
        ¦   External Search Path Buffer           ¦ 64
 FFC0   ------------------------------------------- bytes
        ¦   ZCPR3 External Stack                  ¦\
 FF90   -------------------------------------------  \
        ¦   ZCPR3 Multiple Command Line Buffer    ¦   \
 FEC0   -------------------------------------------    \
        ¦   ZCPR3 Memory-Based Named Directory (S) ¦    ¦
 FDC0   -------------------------------------------     ¦
        ¦   ZCPR3 External File Control Block     ¦
 FD90   -------------------------------------------    1K
        ¦   ZCPR3 Message Buffers                 ¦
 FD40   -------------------------------------------     ¦
        ¦   ZCPR3 Shell Stack                     ¦     ¦
 FCC0   -------------------------------------------    /
        ¦   ZCPR3                ¦     Z3TCAP (S) ¦   /
 FC40   ¦       Environment      -----------------  /
        ¦           Descriptor (S)                ¦/
 FBC0   -------------------------------------------
        ¦   ZCPR3 Flow Command Package (S)        ¦ 0.5K
 F9C0   -------------------------------------------

        -------------------------------------------
        ¦   ZCPR3 Input/Output Package (S)        ¦ 1.5K
        -------------------------------------------
          (I/O Package not implemented in my system)
              (Shown here for reference only)

 F9C0   -------------------------------------------
        ¦   ZCPR3 Resident Command Package (S)    ¦ 2K
 F1C0   -------------------------------------------
        ¦   ZCPR3 BIOS with Modified Cold Boot    ¦
        ¦       Routine to Initialize All Elements ¦ 5K
        ¦       of the ZCPR3 System Above (3)     ¦
 DC00   -------------------------------------------
        ¦   CP/M BDOS                             ¦ 3.5K
 CE00   -------------------------------------------
        ¦   ZCPR3 Command Processor               ¦ 2K
 C600   -------------------------------------------
        ¦   Transient                             ¦
        ¦       Program                           ¦ ~50K
        ¦           Area                          ¦
  100   -------------------------------------------
        ¦   CP/M and ZCPR3 Buffers                ¦256
    0   -------------------------------------------bytes
```

Figure 2: ZCPR3 System Memory Image used in the author's H89

Notes:

1. All Areas Above F1C0H are initialized by the Cold Boot Routine in the BIOS
2. Those Areas marked with (S) are ZCPR3 System Segments
3. The file BIOS.SYS occupies 7K on my disk. It only occupies about 5K in memory because some of the Cold Boot code is overlayed after it is used.

Now, we will break down the memory map into even more detail. Figure 3 shows the greatest detail of the way the ZCPR3 memory is configured in my system. Using Figures 2 and 3 as a reference, we can begin the discussion of all of the features of the ZCPR3 environment. Because installation is complex, I will not cover the details. Experienced assembly language programmers can follow the installation documentation, which is included on the disks. Others will not have to worry about installation if the auto-install version is purchased.

The ZCPR3 Replacement Command Processor (CP)

The ZCPR3.ASM file is assembled after options are selected and the running address is determined. The running address is the same address where the standard CP/M CCP would be located in your 60K CP/M system. The CP/M CCP is then overlayed by the ZCPR3 Command Processor using SYSGEN and DDT. The new CP/M system would be SYSGENed back to the disk and further bootable disks would be made from this master using SYSGEN.

There are many ways of configuring the new Command Processor. You can have all the standard commands like DIR built into the CP, or you can place these commands in a Resident Command Package, which will free the CP to do more sophisticated tasks. This is the best way to go. In my system, the only commands built into my Command Processor are SAVE and GO. A third method would be to have most of your commands available as COM files. There are enough programs on the disk set to drive you crazy while you try to make your decisions.

Let's discuss some of the features of ZCPR3 which lie outside of the Command Processor.

Multiple Commands

ZCPR3 supports multiple commands on the same line, usually separated by semicolons. Thus, the following would be a valid command line:

```
ASM MYFILE.AAZ;LOAD MYFILE;MYFILE
```

In this single command line, the system would assemble MYFILE, load the resulting HEX file into a COM file, and then run it. Up to 200 characters can be placed on one command line. The standard BDOS line editor remains in effect so that Control-E can be used to cause a carriage return/line feed on the screen to allow a long command to be printed on a new line without pressing the return key. The buffer for the Multiple Command Line is defined in the memory maps in Figures 2 and 3. This buffer must be initialized by the BIOS upon Cold Boot before it can be used.

```
Address Range   Size    Function
   Ø -    FF   256 b    Standard CP/M Buffers except
  3B             1 b    Wheel Byte
 1ØØ - C5FF    ~5Ø  K   TPA
C6ØØ - CDFF      2  K   ZCPR3 Command Processor
CEØØ - DBFF      3.5K   BDOS
DCØØ - EFFF      5  K   Modified BIOS with Buffers
F1CØ - F9BF      2  K   Resident Command Package
F9CØ - FBBF      Ø.5K   Flow Command Package
FBCØ - FCBF    256 b    Environment Descriptors
                        Bytes ØØH-7FH  Z3 Parameters
                        Bytes 8ØH-FFH: Z3 Terminal Cap
FCCØ - FD3F    128 b    ZCPR3 Shell Stack
FD4Ø - FD8F     8Ø b    ZCPR3 Message Buffers
                        Byte Ø:  Error Flag (Z/NZ)
                        Byte 1.  IF (8 Levels)
                        Byte 2:  IF Active (8 Levels)
                        Byte 3:  Z3 Cmd Status
                                ØØB - Normal
                                Ø1B - Shell
                                1ØB - Error
                        Bytes 4&5· Error Address if 1ØB
                        Byte 6  Program Error Code
                        Byte 7  ZEX Message Byte
                                ØØB - Normal
                                Ø1B - Z3 Prompt
                                1ØB - Suspend Intercept
                        Byte 8· ZEX Running Flag (Ø=No)
                        Bytes 9-1Ø: Address of Next
                                Char for ZEX to Return
                        Bytes 11-12: Address of First
                                Char in ZEX Memory-
                                Based File Buffer
                        Byte 13· SH Control Byte
                                Bit Ø: Enable SHCMT
                                Bit 1: Enable SHECHO
                                Bit 7· Enable Shell
                                       Entry Wait
                        Bytes 14-15· Shell Scratch
                        Bytes 1ØH-2FH: Error Cmd
                        Bytes 3ØH-39H: Registers
                        Bytes 3AH-3FH  Reserved
                        Bytes 4ØH-4FH. User-Defined
FD9Ø - FDBF     48 b    ZCPR3 External FCB
FDCØ - FEBF    256 b    Memory-Based Named Directory
FECØ - FF8F    2Ø8 b    Multiple Command Line Buffer
FF9Ø - FFBF     48 b    ZCPR3 External Stack
FFCØ - FFFF     64 b    External Search Path Buffer
```

Figure 3: Detailed ZCPR3 Memory Map for the author's system

Directory Enhancement

There are two methods of specifying a directory under ZCPR3. They are symbolically called DU: and DIR: which stand for Disk/User and Named Directory. Under ZCPR3, the USER command is no longer used. If you wish to log into drive B user 3, you would simply type 'B3:'. The system prompt would return with a 'B3>'. Using the DU: form, any of the following commands are valid:

```
A:    (logs you into drive A, current user number)
C5    (logs you into drive C user 5)
8:    (Logs you into user 8 of the current drive)
```

The second method of specifying a directory is called a Named Directory. Under ZCPR3, you can assign a name to a Disk/User area. Let's say that you have three drives and that you have your CP/M system disk in drive A, your Wordstar disk in drive B:, and your dBASE disk in drive C:. Using the file MKDIR.COM, you can create a named directory of your disk environment so that when you are logged into A0:, you would see the following prompt:

```
AØ:SYSTEM>
```

You can now log into your Wordstar disk with either of the following two commands:

```
BØ          or    WORDSTAR:
```

The system prompt will now show:

```
BØ:WORDSTAR>
```

In the same manner, you could log into your dBASE disk with the command:

```
CØ:          or        DBASE:
```

The system prompt will show:

```
CØ:DBASE>
```

As you can see, this feature is most desirable with a hard disk where you have files in many user areas. Named directories can be created with meaningful names assigned to the Disk/User areas.

Resident Commands (RCP)

There are two places where resident commands can be located: One is the ZCPR3 Command Processor, the other is a Resident Command Package (RCP). It is better to have resident commands

CP	-	Copy a File
ECHO	-	Send a message to the console from a multiple command line or a SUB or ZEX file
ERA	-	Erase a file
H	-	Display list of available cmds
LIST	-	Print a file on the printer
NOTE	-	A comment which is not interpreted as a command
PEEK	-	Display memory
POKE	-	Alter memory
PROT	-	Protect files (R/O,R/W,DIR,SYS)
REN	-	Rename a file
TYPE	-	Type a file on the CRT

Figure 4: A List of Commands available in the author's Resident Command Package (RCP)

in the RCP rather than the CP, because this will allow the CP to control more sophisticated system functions. The RCP is loaded into its place in upper memory (see Figures 2 and 3) by the loader utility LDR.COM. It occupies 2K of memory and once loaded is always present. Commands are available immediately on demand. CP/M does not have to do any disk access to find the command, like it would if the command was a COM file. Another advantage of Resident Command Packages is that you may have as many as you wish, each with its own set of standard or custom commands, and each loaded into the RCP memory location when required. The memory resident commands I have available on my system are listed in Figure 4. The only commands I have built into the Command Processor are SAVE, which operates the same as CP/M, except you can specify the number of pages to save in decimal or hex, and GO which allows you to re-execute a COM file which is already in the TPA. This is the same as jumping to address 100H and executing the program there.

Paths

ZCPR3 allows you to specify the search path for the execution of a COM file. Using the PATH command, you may set-up a search path so that if the file you wish to execute is not on the currently logged disk or user area, the ZCPR3 CP will search your other drives for the file in the order you specify in the PATH command. For example, you are currently logged into A7:. Then you issue the command:

```
PATH C0 B3 A4
```

Then, when you attempt to run a COM file, ZCPR3 will first search A7:, then C0:, then B3, then A4:, then finally A0:, the default. Of course, the search will abort as soon as it finds your file. This feature is great for hard disk users. You could set-up a path so that commonly used system programs could reside in a known Drive/User area and be accessible from all other Drive/User areas you happen to be working in.

When using the Named Directory feature previously described, you can also specify the search path in Named Directory form, such as:

```
PATH WORDSTAR DBASE SYSTEM
```

Extended Command Processing And Error Handling

Your ZCPR3 system can be set-up with extended command processing. If a command is issued and ZCPR3 cannot find the COM file along the search path, a default COM file called CMDRUN.COM will be executed. This opens up many diverse possibilities. In my system, I renamed the LRUNZ.COM utility to CMDRUN.COM. I also have a large file of infrequently used commands packed into a library to conserve disk and directory space. LRUNZ.COM can load and execute a COM file from the library. The default library name is COMMAND.LBR. Since I renamed LRUNZ.COM to CMDRUN.COM, it will always be invoked if the requested COM file cannot be found along the search path. ZCPR3 will then pass the command to the CMDRUN program and CMDRUN (LRUNZ in disguise) will try to find the requested COM file in the default library, COMMAND.LBR. Your options here are limited only to your own imagination!

You may choose to implement Error Handling instead of the LRUNZ scheme. There are four different error handlers provided with the ZCPR3 disk set. The fanciest one uses full-screen video features. If you issue a bad command to ZCPR3 from any of the command input sources (multiple command line, SUB, ZEX), the error handler will bring up a full-screen display of what part of your command line is bad and allow you to select options, such as replacing the bad command with a new one, going on to the next command in the sequence, or terminating the balance of the command line and returning to the system. Once you see the powerful features of ZCPR3, you will never go back to plain CP/M again!

Batch Processing

Two very versatile batch processing programs are included with the ZCPR3 package. They are SUB and ZEX. SUB is an enhanced version of SUBMIT.COM which comes with CP/M. SUB is a disk-based batch processor. This means that it uses a temporary disk file to store the sequence of commands it's working on. SUB can accept input from a disk file. It can also accept input interactively from the console without the operator having to edit a submit file. ZEX is a memory-based batch processor. It too can accept input from a disk file or from the console interactively, but it doesn't use a temporary disk file to store its sequence of commands. They reside in memory so execution is very fast. The trade-off is that ZEX uses more system memory than SUB. A tremendous amount of versatility has been built into the ZCPR3 system.

Flow Commands (FCP)

Flow commands reside in the Flow Command Package (FCP) and are loaded into memory with the LDR loader utility. These commands control the flow state of the ZCPR3 system. Commands, such as IF and ELSE, are used to test the TRUE/FALSE flow state. The Command Processor is aware of the flow state. It will allow requested commands to execute if the flow state is TRUE, but will flush requested commands without error if the flow state is false. Here is an example of the use of flow commands:

```
IF EXIST DIR.COM;REN FILES.COM=DIR.COM;ELSE;FILES;FI
```

If I were to explain the above command line in English, I would say "If the file DIR.COM exists in the current directory, then rename it to FILES.COM. Otherwise, run the file FILES.COM". The command FI at the end of the command line means the same as "endif". It is used to tell the Command Processor that it has reached the end of the current IF level. Up to eight IF levels and up to nine flow states can be nested at one time. Figure 5 shows the Flow Commands available in my system.

Shells

The best way to describe a Shell is to talk about that part of the

Command Processor which interprets the command line. This is called the Command Line Interpreter. This is the code which tries to analyze the user's input to the system and decide what to do with it. The ZCPR3 CP does this function the same way that the CP/M CCP does, except that the ZCPR3 CP always checks to see if a Shell has been specified and will run the Shell instead of the Command Line Interpreter. In this manner, ZCPR3 will accept input and run commands from the Shell rather than from user input from the console. A shell can be something as simple as a named variable to something as complex as complete and powerful utilities. For example, suppose you define the named variable "MYFILE" to mean the same as "LETTER.TXT". Now, if you wish to edit this text file, you would usually type "EDIT LETTER.TXT". Because the MYFILE Shell is defined, the command "EDIT %MYFILE" would do the same thing. One obvious advantage of this feature is that you can create a long batch file using named variables, like %MYFILE, and you would not have to change the batch file if you used it to process other files whose names would not always be the same every time the batch file was processed. You would simply use the Shell feature to redefine the names of the named variables. There are two very powerful utilities provided which run as Shells: MENU/VMENU and VFILER. Their operation is impressive.

```
IF     -   Tests a condition for True/False
ELSE   -   Toggles the current condition
           to its opposite condition
FI     -   Terminates the current IF level
XIF    -   Terminates all pending IFs

These are arguments to IF and ELSE

ERROR  -   If the ZCPR3 ERROR FLAG is set
           then the Flow State is True
EXIST  -   If all the files that follow
           this argument exist then the
           Flow state is True
INPUT  -   If the user strikes a T,Y,<cr>, or <sp>
           then the Flow State is True
NULL   -   If the field following this argument
           is blank then the Flow State is True
```

Figure 5: A list of Commands available in the author's Flow Command Package (FCP)

Environment Descriptor

The ZCPR3 Environment Descriptor is a very small area in reserved memory (see Figures 2 and 3) in which is stored some data particular to the user's system. In here is stored particulars about the user's CRT terminal, such as the size of the display and the control codes to perform direct cursor positioning and highlighting. Similar information is stored about the user's list printer, such as page size and the ability to accept form feeds. In addition, the Environment Descriptor contains the addresses of ZCPR3 specific buffers, such as the locations of the RCP, FCP, Named Directory buffer, ZEX registers and others. With all of this information present in the Environment Descriptor, it does not have to be duplicated and be present within every program or utility which needs to know this information. For example, there is a utility called VFILER. This is a full-screen file handling program which is used for copying files between disks. It does direct cursor positioning, reverse video or highlighting and clear screen CRT functions, as well as accessing the correct lines per page on the list printer. It does all this by knowing where the ZCPR3 Environment Descriptor is located in memory and reading the data from it as required.

To permit VFILER to run on ANY ZCPR3 system, the only change that you will have to make is to tell VFILER where the system's Environment Descriptor is located. Once it knows this, it knows where everything else is. This is called "installing". When you "install" a ZCPR3 utility, you merely update the address of the Environment Descriptor within the utility. Now, if you buy a new CRT terminal which uses different control codes than your old one, all you have to do is put the new code in the Environment Descriptor and all of your ZCPR3 utilities will access the new codes! You can even use the old terminal again by loading in the old codes at will. The terminal codes are contained in a separate area of the Environment Descriptor called the Terminal Capabilities (TCAP). Included in the disk set is a menu driven TCAP selection covering over 44 different terminals, as well as a utility to create a new TCAP not on the menu.

Redirectable Input/Output

ZCPR3 allows for custom I/O modules to be dynamically loaded into memory to reconfigure or extend the system I/O capabilities. For example, you may desire to have all of your dialog with the console saved to a disk file for future reference. You may want to have two consoles which operate in parallel. Any number of I/O environments can be created and dynamically loaded.

I have not implemented the I/O Package in my ZCPR3 system. It is shown in Figures 2 and 3 for reference.

Utilities

Over 100 COM files are included on disk. They range from super directory displays to memory dumps, and disk editor to a very fancy full-screen system status display which tells you what's available in your system in a menu-driven screen-by-screen format.

Many of the COM files are enhanced versions of the features available within the Resident Command and Flow Control packages. There is tremendous versatility in setting-up your system. You can choose moderately powerful, but fast commands to be memory-resident, or you could use the more powerful COM files which will take time to load and consume disk space.

Help Feature

Over 200K of online Help Files are present on the disks. If you have the room on your working disks, all the ZCPR3 command instructions and descriptions would be available to be typed on the console by your commmand. Since disk space is at a premium on my system, I printed the Help Files and placed them in a notebook. There is even an included utility which will reformat the CRT-oriented displays and list them on your printer. If you want, you could learn the complete ZCPR3 system without ever opening a book!

SYSLIB3

For all you people devoted to assembly language programming, Richard Conn has made available a library of integrated assembly language modules called SYSLIB3. In this library are hundreds of routines used by the ZCPR3 utilities. If you are interested in "rolling your own" programs, SYSLIB3 will be invaluable.

Echelon

Echelon, Inc. is the commercial source of ZCPR3. This is the only source of the auto-install version of ZCPR3.

Even though ZCPR3 is available from sources which distribute public domain software, ZCPR3 and its utilities are copyright by Echelon, Inc. You may freely use them as long as your use is non-commercial and you make no money. This is where Echelon comes in. If you are manufacturing a new computer and you wish to sell it with ZCPR3 included, Echelon is where you get the license. For us personal users, Echelon provides its customers with expert software consultation and support. They publish a newsletter and operate a bulletin board which contains the latest news and upgraded programs. They are also expecting to release very soon "ZCPR3: The Manual", a 200 page book by Richard Conn.

In addition to ZCPR3, Echelon is the only source of a replacement BDOS called ZRDOS (Z-80 Replacement Disk Operating System). ZRDOS in conjunction with ZCPR3 forms the coveted "Z-SYSTEM". ZRDOS was written to be downward compatible with the standard CP/M BDOS. In addition, it takes into account all the features of ZCPR3. The "Z-SYSTEM" makes you completely independent of Digital Research! You replace their CCP with ZCPR3 and you replace their BDOS with ZRDOS. All that's left is the BIOS and Heath/Zenith wrote that! Echelon will now be your total support.

There is a movement underway right now to promote ZCPR3 and ZRDOS. Echelon is, of course, a major influence. However, a large number of people contribute regularly to the continued enhancement of ZCPR3 and ZRDOS. At least once a week, an improved version of a ZCPR3 utility appears on the Echelon bulletin board. Somebody just had to add those three commands to his favorite program. Fortunately, Echelon is coordinating the work on the official upgrades, so that there won't be more than one upgrade with the same revision number!

One more thing before I stop blowing the horn for Echelon: If you did not obtain ZCPR3 from them, don't ask them for help. It's only fair that you don't get something for nothing. ZCPR3 is among their "bread and butter". Please help by not releasing licensed software like ZRDOS and the auto-install ZCPR3 to public bulletin boards.

Sources For ZCPR3

SIG/M, the Special Interest Group for Microcomputers, a user group of the Amateur Computer Group of New Jersey is a non-profit organization which distributes public domain CP/M software. ZCPR3 and its utilities are available on SIG/M volumes 184 through 192 and Volumes 200 through 202 (twelve eight-inch disks). The cost is $6 per volume, plus shipping. Contact SIG/M at Box 97, Iselin, NJ 08830.

Echelon, Inc. is the official source of ZCPR3 and its utilities, including software support and consultation. A sampler kit with 20 utilities and a manual install ZCPR3 with documentation is available for $39.

If you want more, you can get all of the above in a package containing 70 programs (source code included) in a complete subsystem including the online Help facility for $89.

The ZCPR3 auto-install version, which contains the full-blown complete system and installs in four minutes or less, is available for $149.

Contact Echelon, Inc., 101 First Street, Los Altos, CA 94022. Telephone: (415) 948-3820. The Echelon BBS number is (415) 489-9005.

Summary

Well, what do you think? I have been using ZCPR3 for almost a year now, and I have not even come close to depleting the massive power at my fingertips. ZCPR3 transformed my H-89 into a new and versatile system limited only to my own imagination. Every day I learn something new; every day I have fun using ZCPR3. You could too!

My next step . . . ZRDOS.

H/Z-25 Super Chip Set

- 4 Chip Set
- 12 New Features
- 256 Characters
- Double Strike
- Super/Sub's
- Underline
- Italics
- more

$84.95
+S&H

FINA SOFTWARE
16144 Sunset Bl. #3
Pacific Palisades
California 90272
213/454-6393

recover from a

FATAL ERROR

in just sixty seconds with Rescue & Repair.

At last, there is a fast, simple way to recover data lost because of "Read Only" errors and operator mistakes. In fact, Rescue & Repair will allow you to retrieve your "lost" data after *any* error (except power failure or major technical malfunction), even if the machine has been warm booted or *reset*. Reviewers have been unanimous in their praise. **Interface Age** wrote *"The name says it all. . . The 66-page manual is excellent."* **H-SCOOP** *calls Rescue & Repair "a powerful and very useful program. . . It is extremely easy to use."* **Sextant** calls it *"a real life saver,"* and states that *"Rescue & Repair will recover files lost due to practically anything."* Rescue & Repair is available for MS, PC and Z-DOS* machines for $75 and most CP/M** machines for $55.

Ask your Heath/Zenith dealer or contact:

21st CENTURY DATA, INC.

P.O. Box 1139
Solana Beach, CA
92075
(619) 755-6218
California residents add 6% sales tax.

*Z-DOS is a trademark of Zenith Data Systems
**CP/M is a trademark of Digital Research Corp

Installation Of ZCPR3 For Heath Computers

Richard P. Allen
HHC 1/54th Inf Bn
APO NY, NY 09139

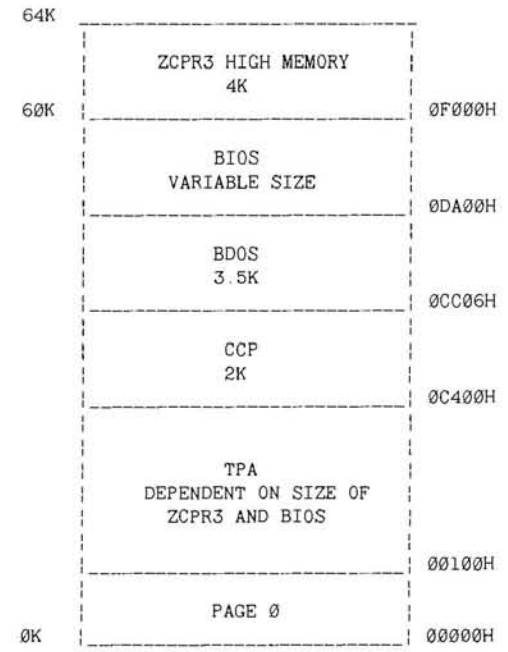
First of all, I would like to introduce myself so all of you timid Heath users out there (if there are any!) will not be too intimidated by the undertaking of installing ZCPR3. I have been a Heath user now for about two years, and spent most of that time working with HDOS. I got my H-89A on the advice of a good friend, Fred Freeland, well known I am sure to many of you in the Tacoma-Seattle area. I have to give most of the credit for me being able to write this article to Fred, because without him I would still probably be playing games on a VIC-20. I have no formal training in computers, except what I have read (and learned from Fred). I am a Physician's Assistant in the U.S. Army by trade and am stationed in Bamberg, W. Germany.)

ZCPR3 is the CP/M 2.2 users answer to UNIX. Move up without getting a new computer and it's all free! I didn't think anything free could be so useful, and until you've tried it, you won't believe it either. I felt an article like this was needed because of all the trouble I had getting my system up, and the fact that I just read an article in SEXTANT about ZCPR1 of all things. I figured everyone in the states would be using ZCPR3 by now, but I have never seen anything in any of the Heath journals about it, so I figured it was about time.

Enough about me, now let's get to ZCPR3. All the source code and many of the assembled files for ZCPR3 are available from the SIG/M users group or Echelon, INC. I got my copy from the SIG/M users group library. ZCPR3 is an incredible replacement for the CP/M CCP and much, much more. It is, in fact, an entirely new operating environment which is compatible with CP/M 2.2 with many of the capabilities of UNIX and some of the other 'more advanced' operating systems. The system includes many new commands, and expansion of almost all of the old commands. It includes the use of named directories and allows for password protection of the those directories. It also enables the user to move from one user area to the other just by typing the user area number. Shells are also supported, as is the ability to create "Supersubmit" files in the form of .COM files. These are just a few oversimplified aspects of ZCPR3, just to wet your appetite. I do not plan to get much more involved in an explanation of the capabilities of ZCPR3, because quite frankly, I am still learning.

In order to gain the tremendous advantage of ZCPR3, there is a small price to pay in the area of TPA. It will take approximately 4K of your available RAM to install all aspects of ZCPR3, except the I/O package which I will not go into as I don't feel it is applicable to the Heath implementation of BIOS.

Heath 64K ZCPR3 Memory Image



Installation

When you begin to read the ZCPR3 Installation manual, which took a couple of hours to print on my Gemini, it says that you need a working knowledge of CP/M, SYSGEN, DDT and Z80/8080 assembly language to perform the procedure. I thought I did have a working knowledge of these, I didn't! I do now! It took me almost 6 weeks from start to finish to figure out or research all of the things I needed to know to do the installation. I can now install a system from start to finish in about two hours, excluding the changes to the BIOS.

Hardware required for Heath installation:

H-89 or H-8 with a Z80 card and 64K of RAM
At least 110K bytes of disk space, two drives are highly recommended, but not required

Software Requirements:

Working CP/M 2.2.03 or 2.2.04 system
Access to source code of your BIOS
MAC.COM assembler AND ASM.COM
DDT.COM,SYSGEN.COM, and MOVCPMXX.COM
ED.COM or any comparable text editor
Microsoft M80 and L80 are required to assemble and link utilities, if so desired (not required for installation)
MAKEBIOS, SUBMIT, XSUB and PREL are also required to reassemble the BIOS.

Once you have decided to install ZCPR3, you have to sit down and make some decisions on what features you wish to include in your system. I made several systems with many of the options excluded, but I finally decided the best option was to install the entire system, except for the I/O packages. In this article, I will describe the installation of the package I chose, which will take 4K of overhead RAM, and includes capability for Named Directories, Resident Control packages, Flow Control packages, External File Control Buffer, Shell Stack, Environmental Descriptor, Multiple Command Line Buffer, External Search Path, and ZCPR External Stack and Wheel capability. I will not go into any long dissertation on any of these, as they are well described in the ZCPR3 documentation.

After you have looked over the ZCPR3 Installation manual and have made your decisions, it is time to begin. The first thing you need to do is to modify your BIOS to initialize the ZCPR3 system on cold boot. The ZCPR3 Doc recommends that you use MAC to assemble the BIOS, however, it is probably just as simple to include the necessary equates in your BIOS and use the MAKEBIOS procedure provided by Heath to assemble your BIOS. All of the equates necessary for this procedure are found in the Z3BASE.LIB file included with the ZCPR3 distribution software, and these need to be set to the values you have chosen for the installation of your system. Below, in Figure 1, are the values which I included as a part of my BIOS. These values assume you are using a 60K ZCPR3 (64K REQUIRED) system, as these are all above 0EFFFH which is the last address in a 60K system.

One change that you may want to make is, if you have room in page zero, to put the Wheel Byte and the External Path in that memory area. That will leave a full page of memory free up high for any other applications you may choose later. The reason I could not do this was that my BIOS uses all of page zero.

All the lines of code in Figure 1 marked by a '#' have to be added to your BIOS to initialize ZCPR3. I have included some additional code above and below the ZCPR3 code to show you where I placed this code in my BIOS. It can probably be placed other places in BIOS, but it should be placed in the area which is written over after the cold boot is completed.

Figure 1

```
IN8255: MOV     B,A
        LXI     H,OUTH84+1
        MVI     A,PPCTL
        ADD     B
        MOV     M,A
        MVI     A,PPMSF+PPGAM1+PPGAPC+PPGBPB
        CALL    OUTH84
        MVI     A,PPDS
        CALL    OUTH84
        RET
        ENDIF
      (End of cold boot code)
#I8080  EQU     FALSE
#RCP    EQU     ØFØØØH  ; RESIDENT COMMAND PACKAGE
#RCPS   EQU     16      ; 16 128-byte Blocks (2K bytes)
#IOP    EQU     Ø       ; REDIRECTABLE I/O PACKAGE
#IOPS   EQU     Ø       ; 12 128-byte Blocks (1.5K bytes)
#FCP    EQU     ØF8ØØH  ; FLOW COMMAND PACKAGE
#FCPS   EQU     4       ; 4 128-byte Blocks (Ø.5K bytes)
#Z3ENV  EQU     ØFAØØH  ; ENVIRONMENT DESCRIPTORS
#Z3ENVS EQU     2       ; SIZE OF ENVIRONMENT DESCRIPTOR
                          IN 128-BYTE BLOCKS
#SHSTK  EQU     ØFBØØH  ; ZCPR3 SHELL STACK
#SHSTKS EQU     4       ; NUMBER OF SHSIZE-BYTE SHELL
                          STACK ENTRIES
#SHSIZE EQU     32      ; SIZE OF A SHELL STACK ENTRY
#Z3MSG  EQU     ØFB8ØH  ; ZCPR3 MESSAGE BUFFER
#EXTFCB EQU     ØFBDØH  ; ZCPR3 EXTERNAL FCB
#Z3NDIR EQU     ØFCØØH  ; ZCPR3 NAMED DIRECTORY AREA
#Z3NDIRSEQU     14      ; 14 18-byte Named Directory
                          Elements permitted
#Z3CL   EQU     ØFDØØH  ; ZCPR3 COMMAND LINE BUFFER
#Z3CLS  EQU     2ØØ     ; SIZE OF COMMAND LINE BUFFER
#EXTSTK EQU     ØFDDØH  ; ZCPR3 EXTERNAL STACK
#EXPATH EQU     ØFEØØH  ; EXTERNAL PATH
#EXPATHSEQU     5       ; 5 2-byte Path Elements
#Z3WHL  EQU     ØFEØBH  ; WHEEL BYTE ADDRESS
;*  END of ZCPR3 BASE EQUATES
```

Beginning of initialization routine:

```
#INZCPR:
#       LXI     D,PATH
#       LXI     H,EXPATH
#       MVI     B,9
#       CALL    MOVLOP
#       XRA     A
#       STA     Z3WHL
#       LXI     H,Z3ENV
#       MVI     B,128+16
#       CALL    ZEROM
#       XRA     A
#       STA     SHSTK
#       LXI     H,Z3MSG
#       MVI     B,8Ø
#       CALL    ZEROM
#       LXI     H,Z3NDIR
#       CALL    ZERO128
#       LXI     D,CMDSET
#       LXI     H,Z3CL
#       CALL    MOVER
#       RET

#MOVER:
#     MVI       B,128
#MOVLOP
#       LDAX    D
#       MOV     M,A
#       INX     D
#       INX     H
#       DCR     B
#       JNZ     MOVLOP
#       RET
#ZERO128:
#       MVI     B,128
#ZEROM:
#       MVI     M,Ø
#       INX     H
#       DCR     B
#       JNZ     ZEROM
#       RET
#CMDSET
#       DW      Z3CL+4
#       DB      Z3CLS
#       DB      Ø
#       DB      Ø
```

```
#PATH: (These values may be changed for your system, but
        the number of bytes used should not be changed)
#       DB        '$','$' (Current disk, Current User area)
#       DB        'A'-'@','$' (Disk A., Current User area)
#       DB        'A'-'@',15  (Disk A., User 15:)
#       DB        'A'-'@',0   (Disk A., User 0:)
#       DB        0
CLEN    EQU       $-HSTBUF
;*******************
;
;  IF COLD BOOT CODE IS SMALLER THAN HOST BUFFER,
;  THEN FILL OUT 'HSTBUF' WITH DS STATEMENT
        IF        (CLEN-HSTSIZ) SHR 15
        DS        HSTSIZ-CLEN
;  OTHERWISE REORG SO RUN-TIME VARIABLES CAN ALSO OVERLAY
;  COLD BOOT CODE
        ENDIF
        IF        NOT ((CLEN-HSTSIZ) SHR 15)
        ORG       HSTBUF+HSTSIZ
        ENDIF
;
;*******************
DIRBUF: DS        128
```

Figure 2 shows where the call for the code in Figure 1 is placed in the cold boot routine in BIOS. This position again may be changed if so desired, but it works fine where it is.

Figure 2

```
CBOOT:  DI
        LXI      SP,STACK
#       CALL     INZCPR
        LDA     DEFIOB
        STA      IOBYTE
        MVI      A,MI$JMP
        LXI      H,CLOCK
        STA      CLKVEC
        SHLD     CLKVEC+1
        IF       INTINP
        LXI      H,CRTISR
        STA      SERVEC
        SHLD     SERVEC+1
        ENDIF
        LXI      H,CTLPRT
        MOV      A,M
        OUT      H88CTL
        INX      H
        MOV      A,M
        ORA      A
        JZ       CBT0
        OUT      H8CTL
```

Only the code marked need be added to BIOS, however, I also added to the CP/M sign-on message. This was just to identify the BIOS as having been changed to initialize ZCPR3. I also added the message '*****WARNING** DO NOT RUN CONFIGUR ON THIS SYSTEM **WARNING*****'. This comes up every time I do a cold boot to remind me that the CONFIGUR program cannot be run on the BIOS once it is mated with the ZCPR3 CCP. This is a problem that hopefully some of you CP/M Hackers out there can help me with, since I have not been able to solve it. I probably could if I had the source code to CONFIGUR. I think the problem lies in the fact that ZCPR3 has an external file control buffer, and the regular CP/M uses an internal FCB, which contains the name of a program to be run on cold boot. When CONFIGUR is run, it writes this file name or a '0' someplace in ZCPR where there should be active code. This is just speculation on my part, but I can tell you for sure CONFIGUR will not work (will destroy!!) on a working ZCPR3 system.

Once you have made all the changes to your BIOS, then you need to assemble it. I found it easiest just to use MAKEBIOS. Below is a step-by-step procedure of the way I assembled my BIOS.

1. Format a blank disk.
2. Move your BIOS.ASM file, that you have made your changes to, to a bootable disk with SUBMIT, MAKEBIOS, MAKEBIOS.SUB, PREL, and ASM.
3. Boot the above system disk and run the following command line:

   ```
   SUBMIT A:MAKEBIOS A:NEWBIOS.SYS
   ```

4. Answer appropriately to the MAKEBIOS prompt, and if you have not made an error in your code, you will shortly have your new ZCPR3 BIOS.
5. Now move NEWBIOS.SYS to the blank disk and rename it BIOS.SYS, put this disk in drive B:
6. For one drive systems, put NEWBIOS.SYS on a bootable disk with MOVCPMXX.COM (MOVCPM17 or MOVCPM37, etc.), and SYSGEN.COM, delete original BIOS and rename NEWBIOS.SYS to BIOS.SYS after you have booted the disk.
7. Now run the following command line:

   ```
   MOVCPMXX 60 B:BIOS.SYS (2 DRIVES)
   MOVCPMXX 60 (1 DRIVE)
   ```

8. Run SYSGEN using the memory image created from the above command. Hit return when asked for source.
   Answer B: for dest for two drives
   Answer A: for dest for one drive
9. Now you should have a bootable disk, with your new ZCPR3 BIOS, and a 60K system mated with your old CCP.
10. Now you can configure the BIOS. See Heath documentation if you are not familiar with CONFIGUR.COM.

(**Note:** You must configure prior to replacing the CCP with the ZCPR3 CCP to avoid crashing the new system.)

You should now have a disk with your new BIOS, configured the way you want it, using your old CP/M 2.2 CCP. You now need to find the location of your CCP base address to use in the file Z3BASE.LIB. This may be done by using the program Z3LOC .COM, provided with ZCPR3 or a similar program, such as TELL.COM. You now need to proceed to modify the Z3HDR.LIB and Z3BASE.LIB in preparation for the assembly of the new ZCPR3 CCP.

To begin this procedure copy, ZCPR3.ASM, MAC, Z3HDR.LIB, Z3BASE.LIB, and a text editor to the system disk with the new BIOS. Load Z3BASE.LIB and begin. The initial Base addresses section can all be deleted, unless you desire to change these for reference purposes. You then need to move directly on to the body of Z3BASE.LIB.

The first three equates may all be deleted as they are not used:

```
Z3REV     EQU  30
CBREV     EQU  41
MSIZE     EQU  48
```

Base equate must be set to the base address for your CP/M:

```
BASE      EQU  0
I8080     EQU  FALSE       ;SET FOR Z80
EXPATH    EQU  0FE00H
EXPATHS   EQU  5
Z3WHL     EQU  0FE0BH
CCP       EQU  0C400H      ;BASED ON  VALUE  DETERMINED
                            ABOVE, THIS IS MY VALUE
```

(**Note:** the CCP value must be based on the modified BIOS and your 60K system, and does not allow the Z3CPR system to be moved via MOVCPM.)

```
RCP       EQU  ØFØØØH
RCPS      EQU  16
IOP       EQU  Ø                 ;NOT IMPLEMENTED,
                                  BUT MUST BE Ø
IOPS      EQU  Ø
FCP       EQU  ØF8ØØH
FCPS      EQU  4
Z3ENV     EQU  ØFAØØH
Z3ENVS    EQU  2
SHSTK     EQU  ØFBØØH
SHSTKS    EQU  4
SHSIZE    EQU  32
Z3MSG     EQU  ØFB8ØH
EXTFCB    EQU  ØFDBØH
Z3NDIR    EQU  ØFCØØH
Z3NDIRS   EQU  14
Z3CL      EQU  ØFDØØH
Z3CLS     EQU  2ØØ
EXTSTK    EQU  FDDØ
```

The final equate may also be deleted:

```
DJEPROM    EQU  (NOT NEEDED BUT YOU COULD USE THIS
                 TO POINT TO THE MEMORY THAT'S LEFT
                 UP HIGH)

(Note:The addresses from ØFEØCH-ØFFFF are unused.)
```

This completes the necessary information in Z3BASE.LIB, anything else that I may have overlooked is unnecessary for installation. This file is used to assemble all of the memory resident support packages in ZCPR3.

The next step in the installation is to edit the Z3HDR.LIB. If you are short of disk space, you can delete all comments in this file. Below are the values which should be used to get ZCPR3 up initially. If you desire to modify any of these equates at a later time, that is OK, but too much modification will cause the code to exceed 2K and you will get an assembly error. I will only discuss those items in this area that are necessary for installation, anything not mentioned may be deleted from the file. Detailed explanations of all of these equates are found in the ZCPR3 installation manual. This part of the article should be used as a guide to modify these .LIB files.

```
CPRLOC     EQU  CCP             (To be installed via
                                 SYSGEN/DDT)
COMTYP     MACRO
           DB        'COM'
           ENDM
SUBTYP     MACRO
           DB        'SUB'
           ENDM
SUBON      EQU       TRUE
DRVPREFIX  EQU       TRUE
COMATT     EQU       Ø1H
DIRON      EQU       FALSE
LTON       EQU       FALSE
GOON       EQU       TRUE
ERAON      EQU       FALSE
SAVEON     EQU       TRUE
RENON      EQU       FALSE
GETON      EQU       FALSE
JUMPON     EQU       FALSE
NOTEON     EQU       FALSE
```

No changes need to be made to the wheel byte equates for the initial assembly. All should be left FALSE.

```
NCHAR      EQU       4
```

CTABLE MACRO should not be changed but must be included!

```
WIDE       EQU       TRUE
FENCE      EQU       '|'
SYSFLG     EQU       'A'
SOFLG      EQU       'S'
ERAOK      EQU       FALSE
ERAV       EQU       FALSE
ERDFLG     EQU       'V'
PGDFLT     EQU       TRUE
PGDFLG     EQU       'P'
NLINES     EQU       24
SECTFLG    EQU       'S'
PATH       EQU       EXPATH
MINIPATH   EQU       TRUE
SCANCUR    EQU       TRUE
INCLDU     EQU       TRUE
ACCPTDU    EQU       TRUE
NDINCP     EQU       TRUE
INCLNDR    EQU       TRUE
ACCPTND    EQU       TRUE
DUFIRST    EQU       FALSE
PWCHECK    EQU       TRUE
MULTCMD    EQU       TRUE
CMDSEP     EQU       ';'
CMDRUN     EQU       FALSE
IFON       EQU       FALSE
MAXUSR     EQU       31

MAXDISK    EQU       '4'    ;STANDARD VALUE, MAY BE
                             CHANGED TO ACCOMMODATE
                             YOUR SYSTEM
SUPRES     EQU       FALSE
SPRMPT     EQU       '$'
CPRMPT     EQU       '>'
NUMBASE    EQU       'H'
CURIND     EQU       '$'
COMMENT    EQU       ';'
```

This completes the modification of the ZCPR3.LIB files. If all of the above files (ZCPR3.ASM,Z3BASE.LIB,Z3HDR.LIB,MAC) are on your present disk, type the following command line:

```
MAC ZCPR3 $-S PZ
```

If all goes well and you have no errors, you will create a file called ZCPR3.HEX, which is the file needed to replace your old CCP.

The next step in the installation is the replacement of the CCP. Prior to going any further, you should back-up copies of the above .LIB, .HEX and .ASM files, and erase all but Z3BASE.LIB and ZCPR3.HEX from your working 60K system disk. Now copy SYSGEN and DDT to the disk with the above files and run SYSGEN on the 60K system. When prompted for source from SYSGEN type A:, respond no to copy BIOS question and hit return when asked for destination. Next type the command line:

```
SAVE 38 CPMIMAGE.BIN
```

Next, type the command line:

```
DDT CPMIMAGE.BIN
```

Your terminal should respond with:

```
DDT VERS 2.Ø
NEXT  PC
27ØØ  Ø1ØØ
```

You now need to find the start of the CCP, which should be at either 0980H or 0900H. You can determine this by typing the -d command followed by the appropriate memory locations starting at 0900H. (0900H,0980H,0A00H,etc)

When you see two hex jump instructions (C3 XX XX C3 XX XX) followed by the words COM and SUB and a copyright message in the ASCII area, you have found the start of the CCP. I will assume from here that your CCP starts at 0980H. The next step is to zero out the CCP area using the command:

```
    -f980,117f,0          (Zeros out 2K or 800H bytes)
```

Next, we need to find the offset to load the new CCP at our new location. The following command will give the offset:

```
    -h980 C400 (The first number is the place in memory
                you want your  file to  load or where your
                your started, the second number is the
                address of the CCP of the 60K system with
                the modified BIOS/real time load addr.)
```

DDT will respond with two numbers, the second is the offset:

```
    -CD80 4580
```

Now dump memory at 980 again to insure that it has been zeroed out, use the -d980 command. Next, we need to patch our new CCP into the memory image. The following commands will accomplish this:

```
    -iZCPR3.HEX     (Must be on same disk as DDT)
    -r4580
```

DDT will respond with:

```
    NEXT  PC
    2700  0000
```

(**Note:** If the 'NEXT' address is not the same as the 'NEXT' address you got when you loaded the CPMIMAGE.BIN file, something is wrong. Recheck the offset and make sure you used the proper location for the CCP.)

Again, dump memory at 980H to confirm that the new CCP has been loaded at the proper location. If, again, you see the two jumps and the COM and SUB in the ASCII area, the procedure has been accomplished successfully. Now Control-C out of DDT and run SYSGEN. When prompted for source, just hit return to use the image we just created which is still resident in memory. SYSGEN will not ask if you want BIOS copied, but when prompted for the destination type A:, and any other disks you want to sysgen with the new system. Now do a shift-reset and do a cold boot with the disk with the new BIOS and CCP and you should see A0> come up. If you type one of the normal CP/M resident commands such as DIR or ERA, you will get:

```
    A0>DIR    or    A0>ERA
    ?DIR            ?ERA
```

since these commands are no longer resident in the CCP. Another test is to type the following, 15:, and you should get the following response, A15>. If both of these responses are appropriate, your installation has been successful to this point. As you can see at this point ZCPR3, without the resident support packages, is not very useful, so our next step is to install the first of the support packages which is the Environment Descriptor.

Before moving on, do some clean-up by saving and erasing all files on the newly created ZCPR3 system disk, except MAC and Z3BASE.LIB. Now copy SYSENV.LIB, SYSENV.ASM, a text editor, and LOAD.COM onto the ZCPR3 system disk.

Now load SYSENV.LIB into the text editor, and edit to fit your system. The following are the changes I made to this file prior to assembly.

1. Processor speed equate to 2mHz
2. Terminal and printer specific data can be changed to your specs, TCAP entry should not be altered at this time.

After modifications are complete, assemble SYSENV with the following command line:

```
    MAC SYSENV $-S PZ
```

This will create a HEX file which needs to be loaded to create a COM file. Use the following command line:

```
    LOAD SYSENV.HEX
    (This will create a file called SYSENV.COM)
```

Next, rename SYSENV.COM to SYS.ENV to create the environmental descriptor package. Type the following command to clean up the disk after saving the SYSENV.LIB file:

```
    ERA SYSENV *
```

In order to create the terminal specific portion of the Environmental descriptor package, we need to copy the files, TCSELECT.COM and Z3TCAP.TCP, onto the system disk and type the following command line:

```
    TCSELECT MYTERM.Z3T
```

You will be presented with a menu which doesn't contain any options for Heath terminals. By typing '+', you will get menu 2 which contains two options for Heath terminals. And by making a selection, you will create a file called MYTERM.Z3T which can be loaded into memory later with the LDR.COM program.

Now if creation of MYTERM.Z3T was successful, erase TCSELECT and Z3TCAP.TCP.

If you so desire, you can also assemble the files SYSNDR.LIB and SYSNDR.ASM to create the file SYS.NDR, which is the Named Directory file which can be loaded with LDR.COM. This file is assembled and loaded in the same manner as the SYSENV files. If you do not desire to assemble this file, it can be created later using the MKDIR.COM program.

Your system disk should now contain BIOS.SYS, SYS.ENV, MYTERM.Z3T, LOAD.COM, Z3BASE.LIB, a text editor, MAC .COM and optionally SYS.NDR. We are now ready to assemble the first of the Resident Command packages. Begin by copying SYSRCP2.LIB and SYSRCP.ASM to your new system disk (or any other disk where you have room to assemble it). I do not recommend changing the SYSRCP?.LIB files the first time through the installation process, as these are very sensitive to changes in size. They will assemble without problem the way they are presently set up. If you want to go back later and make changes when you are more familiar with the function of ZCPR3, you will be more able to make useful modifications.

If you have copied the above two files to your disk with MAC, then rename SYSRCP2.LIB using the following command line:

```
    REN SYSRCP.LIB=SYSRCP2.LIB
```

Now, begin the assembly with the command:

```
    MAC SYSRCP $-S PZ
```

If you get no errors, you will end up with a new file by the name of SYSRCP.HEX, which again needs to be loaded to make a COM file. Take note here, though, the program LOAD.COM will not work! I found this out the hard way, trial and many errors. The ZCPR3 documentation says you can use a program called MLOAD.COM, which I don't have, nor have I ever heard of. I developed a procedure using DDT to do the same thing, which I will describe in the following instructions.

Type the command line:

```
    SAVE 8 JUNK.BIN (This will save 8 pages of Junk
                     to a file on disk.)
```

Next type:

```
    DDT JUNK.BIN      (This loads 8 pages of Junk into
                       memory with DDT up high.)
```

DDT should respond with:

```
DDT VERS 2.0
NEXT  PC
0900  0100
```

Fill these 8 pages with zeros with the command:

```
-f100,900,0
```

Control-C out of DDT and type the following command line:

```
SAVE 8 JUNK.BIN
```

We now have an 8 page file full of zeros on disk. Now, type the following command line:

```
DDT JUNK.BIN
```

DDT will respond:

```
DDT VERS 2.0
NEXT  PC
0900  0100
```

Now we have 8 pages of zeros in memory, and we are ready to load our first SYSRCP hex file, but we need to figure the offset first using the following command:

```
-h100 0f000        (The first number is the place we
                    want the file to load in DDT, the
                    second number the real time load
                    addr from Z3BASE.LIB)
```

DDT will respond with two numbers, the second is the offset:

```
f100 1100
```

We can now load the SYSRCP hex file with the command:

```
-iSYSRCP.HEX
-r1100
```

Now using the -d100 command check to insure that the SYSRCP hex file has loaded at 0100H, if so Control-C out of DDT and type the following command line:

```
SAVE 8 SYS2.RCP
```

This will create a file called SYS2.RCP on disk which can be loaded later with LDR.COM. This completes the assembly of the first of the four SYSRCP.LIB files, the remaining three are all assembled in the exact same manner. The final thing we have to assemble is the SYSFCP? files. There are two of these in the distribution ZCPR3 files. These are also assembled and loaded in almost the same manner as the SYSRCP files that we have just finished, the only difference is that instead of saving 8 pages of memory out of DDT we only save 2. The address used to figure the offset is 0F800H, the same address as was used in Figure 1 for FCP.

After you complete the above assemblies, you should have the following files on your ZCPR3 system disk:

```
SYS.ENV
MYTERM.Z3T
SYS1.RCP
SYS2.RCP
SYS3.RCP
SYS4.RCP
SYS.FCP
SYS1.FCP
SYS.NDR    (Optional)
Z3BASE.LIB
BIOS.SYS
MAC.COM
LOAD.COM
TXTPRO.COM (Text editor)
```

We are now ready to try and get the whole system up and running. But, before we do, you need to backup all of the files you have just created, just in case! After you have done this, put the following files on your system disk from the ZCPR3 distribution files:

```
LDR.COM          Z3INS.COM
MCOPY.COM        WHEEL.COM
PWD.COM          SHOW.COM
MKDIR.COM        SUB.COM
XD.COM           DIR.COM
XDIR.COM         ERROR1.COM
ERROR2.COM       ERROR3.COM
ERROR4.COM       ERRORX.COM
```

Having completed this, create a text file with a list that contains the preceding files, one file per line only, and save it as COMFILES.INS.

Now type the following command line:

```
Z3INS SYS.ENV COMFILES.INS
```

This will install the .COM files above for the system you have just created. This is necessary, prior to using the above files, so they will function correctly with your terminal. When the installation is complete, type the following command line to get the system up and running:

```
LDR SYS.ENV,MYTERM.Z3T,SYS2.RCP,SYS.FCP,SYS.NDR;ERROR3
```

The prompt you see should now show you the following:

```
A0> (If you did not yet assemble SYS.NDR) or

A0:BASE> (Including SYS.NDR)
```

You should now have ZCPR3 installed, up and running. Run the program, SHOW.COM, to see some of the capability of ZCPR3.

***** ENJOY!! *****

This completes the installation of ZCPR3 on your Heath 8-bit system. I hope this has been of some value in helping get ZCPR3 up and running, as it is a real pleasure to compute under this operating system.

This article has not touched on the capabilities of ZCPR3, It has tremendous capability and for all of you out there who have 8-bit machines, and no plans to go to 16-bit in the near future, you can now have some of the capability of UNIX and MS-DOS under CP/M 2.2. Hope you enjoy ZCPR3 as much as I have, and if problems arise or you have a solution to any of the problems I have evidenced in this article, feel free to write me.

TIRED OF WAITING, WAITING, WAITING,
ON YOUR COMPUTER ? SO, SPEED IT UP !

INTRODUCING

THE TIME WORKS'

CPDZ-100 & CPDZ-150

INSTALLS IN JUST MINUTES WITH NO PERMANENT MODIFICATIONS TO YOUR EXPENSIVE MAIN BOARD. A SMALL SWITCH, MOUNTED ON A PANEL FITTED ON THE BACK OF YOUR COMPUTER, ALLOWS THE USER TO INCREASE THE 8088'S SPEED FROM STOCK UP TO 40 - 50%. COMPATABLE WITH MOST HARDWARE AND SOFTWARE. VERSIONS FOR H/Z-100'S AND 150/160.

JUST $34.95 PLUS $2.00 S & H. TX RESIDENTS ADD 5.125% SALES TAX.

SEND CHECK, MONEY ORDER TO >

CPD
Consumer Product Development

CONSUMER PRODUCT DEVELOPMENT
6242 B RUMFORD LN
HOUSTON, TX 77084

Using The H-100 As A Time Machine (I)

Arnold W. Seibel
851 Cypress Street
Monterey, CA 93940

This adventure was inspired by Randy Meyers' procedure SETPOINT ("Computer Graphics on the H/Z-100," REMark V. 5, No. 5, May 1984). SETPOINT lights up a specified pixel, on the display screen, in a specified color. If you've read Randy's article, I'm sure you appreciated it. Addressing a pixel in the video RAM is not a simple matter.

But, SETPOINT is a subroutine, not a free-standing program. It won't run by itself; it requires another program to set up a stack, push three numbers onto the stack, and instruct the 8088 to CALL SETPOINT.

Another problem with SETPOINT, from my point of view, was that I didn't know how fast it was. I translated the program (roughly) into LMI 8086 Forth and found that version rather slow. But how slow was it? I needed to run SETPOINT in the original version, together with a routine that would tell me how long it took SETPOINT to execute, say, 10,000 times.

To do that, I had to learn to use a lot more of the power of MASM than I had dared to mess with before. While I was at it, I decided to try writing a program that would run as an .EXE file — the kind of program that's not limited to 64 Kbytes in size. I didn't write anything that big, but I learned how to; and in the process, I learned a lot about LINK and LIB, and the way the 8088 organizes memory and generally takes care of business. Here's what I learned — although not necessarily in the same order, and certainly not with the same number of false starts.

A Simple Step: H2ASCII

The program I ultimately wrote to benchmark SETPOINT is made up of several small modules. Modules are simply .ASM files that are assembled separately, then LINKed together to make an .EXE file that will run. One of the modules has to contain a main procedure (that's what I call it) that does three things: provides the operating system with an entry point, executes the other procedures as subroutines, and exits to the operating system. SETPOINT is a subroutine; so is H2ASCII.

H2ASCII does a little job that has to be done in order to show the time on the screen. It takes a hexadecimal byte (the value is assumed to be less than decimal 100), translates that byte into two binary-coded-decimal digits, and finally translates those digits into the corresponding ASCII characters, which can then be sent to the screen. H2ASCII expects to find the original byte in the 8088's AL register; it leaves the two ASCII characters in AL and AH, which together are called the AX register.

Registers are memory locations within the 8088. They all have special functions — the 8088 implicitly uses certain registers when you give it certain instructions — and most of them can also be used in other ways that you make up yourself. All the registers are 16 bits (four hex digits, or two bytes) wide. Four of them — AX, BX, CX, and DX — can also be used as pairs of one-byte registers called AH and AL (H and L for High and Low), BH and BL, and so on.

H2ASCII uses both AL and AX; it also uses CL. The procedure expects to find a one-byte number in AL. The first thing it does is to CBW — Convert the Byte to a Word. This just means that it copies the highest (most significant) bit of AL throughout AH. The result is that AX contains a two-byte number with exactly the same value as the one-byte number in AL. We want that to be a two-byte number because the next thing we're going to do with it is DIVide it by ten. The DIV instruction has to be accompanied by a register name or memory location, which tells the 8088 where it will find the divisor. If that is a one-byte value (which CL is), then the dividend is expected to be a two-byte value, and to be in AX. H2ASCII MOVes the value 10 into the one-byte register CL. The instruction DIV CL then tells the 8088 to divide whatever is in AX by the value in CL.

When DIV divides a two-byte number by a one-byte number, it returns the quotient in the AL register and the remainder in the AH register. When the divisor is ten, these numbers represent the two-digit decimal equivalent of the number that was originally in AX. For example, suppose we are translating the byte that contains the hour, and the hour is 12. The DOS function that tells the time will give us that in the form of the byte 0C hex. The procedure that calls H2ASCII will MOVe that byte into AL. CBW converts that byte to the two-byte number 000C hex. Dividing it by ten returns, the quotient (01 hex) in AL and the remainder (02 hex) in AH. Those are one step closer to the ASCII '1' and '2' in '12'.

All that remains is to convert the numbers 01 hex and 02 hex to the numbers that represent the characters '1' and '2' in the ASCII system. The hex column in an ASCII chart shows that the hex number for an ASCII digit is 30 hex plus the value of that digit: 30 hex represents '0', 31 hex represents '1', and so on. So, to convert a hex number from 0 through 9 to its ASCII equivalent, all we have to do is add 30 hex to it. The two ADD instructions in H2ASCII do that to the quotient and remainder left in AL and AH by the DIV instruction. The instruction ADD AH,'0' is the same as ADD AH,30h; the assembler automatically converts the character '0' to its numerical equivalent.

Since this is the first procedure, I'll briefly explain the code that appears before and after the instructions we've just gone through. The PAGE and TITLE lines influence only the listing that the assembler produces, not the program. The comments, which are preceded by ';', are ignored by the assembler; they're for people only.

In the line "CODESEG SEGMENT PARA PUBLIC 'CODE'," the word SEGMENT signifies that this is a segment directive. It marks the beginning of a segment. The word preceding it, CODESEG, is the name I chose to use as a label for this point. The remaining three words will be passed on to LINK when this procedure is linked with others. PARA directs that this segment shall be placed in memory on a paragraph boundary. That is, its first address is to end in 0 hex. (LINK requires segments to begin on paragraph boundaries.) PUBLIC directs that this segment will be concatenated (joined) in memory with any other segments in the same LINK session that have the same name (CODESEG) and the same class ('CODE'), and they will all be treated as one segment in the final .EXE program. 'CODE' is the name I chose to give to the class that identifies all my code segments.

The next line of code is ASSUME CS:CODESEG. This tells the assembler that the CS register will contain the segment number for all the symbols in CODESEG. For the 8088, every memory address must have two parts: a segment number and an offset. Both are four-digit hex numbers. The offset is the distance in bytes from the beginning of a memory segment to the address in question. The segment number is the absolute memory address of the beginning of a segment, but with the last byte omitted. To compute the memory address for a symbol, the 8088 puts another hex zero on the end of the segment number (multiplying it by 16) and adds the offset to the result. The segment number is usually looked up in one of the four segment registers, CS, DS, SS, or ES.

The only symbol in CODESEG is H2ASCII, the name of the procedure. Because that symbol comes just before the first memory location in this module (the first byte of the CBW instruction), its offset will be 0000. If you assemble this code and have MASM make you a .LST file, you can see this very clearly. The 8088 must add this offset to a segment address (constructed from the segment number in CS) in order to find the code for CBW in memory. This segment number, the one marking the beginning of the instructions, will be placed in CS when the operating system executes the program.

PUBLIC H2ASCII provides the linker with some information about the procedure H2ASCII, which is named on the next line. Declaring this procedure PUBLIC makes it eligible to be called as a subroutine by procedures outside this module.

In H2ASCII PROC NEAR, the word PROC signifies that the following code is a procedure — a body of code that can be executed, whether on its own or as a subroutine. H2ASCII is the name I chose to give it. NEAR essentially means that, when this procedure is called by other procedures, they will be in segments with the same segment directive as this one — CODESEG SEGMENT PARA PUBLIC 'CODE' — and addressable through the CS register. If a procedure is not NEAR, it is FAR. If you get this wrong, the 8088 won't be able to find its way back to the calling procedure after it executes the subroutine. When it calls a procedure, it saves the next address so that it can pick up where it left off when it returns from the procedure. If it's a NEAR call, it only has to save the offset; if it's a FAR call, it has to save the segment number, too.

The RET after the operative code tells the 8088 to pick up where it left off. The next line is H2ASCII ENDP. The ENDP means this is the end of a procedure, and the label H2ASCII tells the assembler which one it's the end of. Similarly, in the next line, ENDS means this is the end of a segment, and CODESEG is the name of the segment that's ending. Finally, END tells the assembler that this is the place to stop assembling.

Now, one thing about this little procedure may have bothered you. We left the results of our computations with an ASCII '2' in AH and an ASCII '1' in AL. If you look at all of AX, we've left the string '21' in it, which is '12' spelled backwards. We could swap the bytes with the instruction XCHG AL,AH, but we don't want to. What's actually in AX is 3231 hex. Later (in the procedure that calls this one) we're going to MOVe that two-byte number from AX into a memory location in the midst of a character string. When 3231 hex gets moved into memory, it will be stored with its low byte high and its high byte low. In other words, it will be stored as 3132 hex, or '12', which is just how we want it.

Getting Down To Business: TIMERON

My program to benchmark SETPOINT is constructed as a pixel sandwich on time: SETPOINT is in the middle, and two subroutines that do the timing are on both sides of it. TIMERON is the first slice.

Like H2ASCII, TIMERON does something very simple. However, its structure is a little more complex. In addition to a CODESEG, it also has a DATASEG, this adds a few housekeeping lines. The SEGMENT directive at the top of the data segment is exactly analogous to the one we have seen already for a code segment. Within the data segment is one directive:

```
MESSAGE DB 'Setting timer.   ','$'
```

This is a Define Byte directive. It tells the assembler to reserve some memory space at this point in the segment; to allow the first byte of this space to be referred to by the symbol MESSAGE; to allow information to be read from it or written to it in byte-size pieces (signified by the B in DB); to make the space exactly large enough to hold what follows; and to put what follows into that space. What follows is a sequence of bytes; again, the assembler translates characters inside single quotes into their hex equivalents, and again, if you have MASM assemble this and make a .LST file, you can easily see what is going on. After the message is one extra byte, '$'. This character will not appear on the screen, it merely marks the end of the string for the DOS function that sends strings to the screen. It could have been included in the same set of quote marks with the string, following the three periods; however, putting it in its own quotes makes it easier to remember what it's there for.

In CODESEG this time, we have a longer ASSUME directive telling the assembler that the DS register holds the segment number for DATASEG, as well as that CS holds the one for CODESEG.

Immediately after the PROC directive marking the beginning of the procedure, we MOVe the segment number of DATASEG into the DS register. We do this indirectly by way of AX, because it's illegal to move an immediate value into a segment register. The symbol DATASEG in MOV AX,DATASEG represents an immediate value because it is a segment label; the assembler substitutes DATASEG's segment number for it automatically. Other addresses, such as the offset of MESSAGE inside the data segment, cannot be gotten at quite this easily. (We do not have to move CODESEG's segment number into CS, because the operating system will do that when it executes the program.)

The next instruction (after DS is fixed up) will show how to get at the offset of MESSAGE. These three lines, ending with INT 21h, print the string in MESSAGE on the screen. Taking them from the bottom up, INT 21 tells the 8088 to look in the AH register for the number of a DOS function, and to execute that function. The preceding instruction is MOV AH,9, which MOVes into AH the number of the Print String function. When the 8088 executes that function, it will be told to look in DS for the segment number and in DX for the offset of the first byte of the string. The first instruction in this block takes care of DX by MOVing the OFFSET of the label MESSAGE into it. DS already contains the correct segment number, since MESSAGE is defined inside DATASEG and DATASEG's segment number is in DS.

The next two lines of code set the registers CX and DX to zeros. This is being done because the DOS Set Time function expects to find the new time in these registers. For the purpose of timing the execution of SETPOINT, I don't care what time it is, I just want to know how much time elapses while it's running. So I'll set the time to 00:00:00.00 before it starts, then run it, then read the time. The time I read will be the elapsed time, no subtraction required.

The registers could have been set to zeros by the instructions MOV CX,0 and MOV DX,0. In fact, there's no reason why they shouldn't be, because those instructions would take up exactly the same number of bytes and execute in exactly the same amount of time as the XOR instructions. However, this gives you a chance to investigate the meaning of XOR. XORing a register with itself sets all the bits in that register to zero, regardless of what they were before.

Finally, we come to another function call. This time we have MOVed the number 2D hex into the AH register. This is the number of the DOS Set Time function. It causes the 8088 to move CH into the hour, CL into the minute, DH into the second, and DL into the hundredths of a second byte where the DOS keeps track of the time. The time is thus set to zero, and we waste no time getting out of this function so we can get into SETPOINT. At most, we waste very little time. RET takes four microseconds to execute, which isn't much when your timer is accurate only to the nearest hundredth of a second.

Getting Into The Heavy Stuff: TIMEROFF

TIMEROFF is the second slice of time on my pixel sandwich. It reads the time and prints it on the screen, followed by the word "elasped." It calls H2ASCII as a subroutine to translate the time into printable characters.

This module begins with two structure definitions. These directives don't produce any executable code, and they don't set aside any memory space. Instead, they provide measuring instruments for locating certain bytes or words (two-byte values) in memory. The measuring will be done relative to some specified beginning point in memory. Each label inside the structure definition is a definite number of bytes from the beginning. These labels will be used to measure from the beginning point we specify in memory.

TIMEROFF will use these structures to measure memory in the data segment. When we get to TIMEROFF's DATASEG, we will set aside the regions to be measured. For now, just keep in mind that the structure definitions at the beginning of the module, outside all the segments, are only providing the measuring sticks.

The phrase, TIME_BYTES STRUC, tells the assembler that we are defining a structure and that the label TIME_BYTES will be used to refer to the structure as a whole (as a set of measuring distances, not as bytes set aside in memory). The first line after that, BHR DB ?, means that the first byte in this structure will be called BHR. The next line, BMIN DB ?, means that the second byte will be called BMIN. And so on.

When all is said and done, TIME_BYTES defines a structure four bytes long, and each byte has its own name. The question marks mean that we do not care what is in the bytes at this point. Later, we will reserve some space in DATASEG where we can use the labels BHR, BMIN, BSEC, and BFRAC to refer to four consecutive bytes. Those are the places where we will store the four bytes of the time after using the DOS Get Time function to retrieve them, and before using H2ASCII on them.

The other structure, TIME_STRING, is a bit more complex. We will use it to measure the area in DATASEG where we store the ASCII characters of the time, in preparation for printing them on the screen. Some of the parts of this structure are defined as words (DW, two bytes, four hex digits) instead of bytes. In DATASEG, these will be used to store two-byte ASCII strings. The word-wide portions are all uninitialized (the question marks) because we don't care at this point what is in them. The byte-wide portions, however, are initialized with specific things: two colons to separate the hour, the minute, and the second; a decimal point to set off the hundredths of a second; a space and the word "elasped"; and the '$' that marks the end of a printable string. When we use this structure to set aside some memory space in DATASEG, these initial values will automatically be put into the appropriate bytes.

Inside DATASEG, the first line is an enigmatic:

```
    IN_BYTES     TIME_BYTES      <>
```

TIME_BYTES is the name of the first structure defined above. This directive tells the assembler to set aside just enough space at this point in DATASEG to hold something the size of TIME_-BYTES, to allow the first byte of that space to be referred to as IN_BYTES, and to allow bytes within that space to be addressed by the labels within the definition of TIME_BYTES. The angle brackets are necessary to make this work. If we wanted to initialize any of the bytes at this time, we could do so by inserting the values within the brackets, separated by commas.

Similarly, the next line:

```
    OUT_STRING     TIME_STRING      <>
```

sets aside enough space to hold something the size of TIME_-STRING, causes its first byte to be called OUT_STRING, and allows its parts to be addressed by the labels in TIME_STRING. At this time the colons, the decimal point, the 'elapsed,' and the

'?' are placed in the appropriate bytes as measured from the first byte of OUT__STRING.

If at this point you're confused, let me recommend one more time that you assemble this code and get a .LST file out of it. There you will see the bytes and words that are set aside, and the initial values that are entered by the above two directives.

Moving on to CODESEG, we find another new thing:

```
     EXTRN      H2ASCII:NEAR
```

This directive — notice that it is inside CODESEG — means that a procedure called H2ASCII is going to be called; that it is EXTeRNal, i.e., not in this module, but assembled separately in some other module; and that the call to it will be a NEAR call, i.e., a call to another location within CODESEG. The NEAR in this directive matches up with the NEAR in the directive H2ASCII PROC NEAR at the beginning of H2ASCII. The EXTRN in this directive matches up with the PUBLIC in the directive PUBLIC H2ASCII just before the beginning of H2ASCII.

Inside the procedure TIMEROFF, after DS has been set up, we come to another function call: INT 21h preceded by MOV AH,2Ch. 2C hex is the number for the function Get Time. That function moves the hour into CH, the minute into CL, the second into DH, and the hundredths of a second into DL. The time, of course, goes on but we now have captured whatever the time was when this pair of instructions was executed. This is the time when SETPOINT finished executing for the 10,000th time — plus the time required to call TIMEROFF and execute the two MOVes that set up DS, which is about six microseconds. Still pretty close.

The next thing we do is to move those four bytes into the structure IN__BYTES, which we have set aside in DATASEG. This is necessary because we're going to be using the registers for computations. The four MOVes following the function call are fairly straightforward. As an example of the structure notation, the expression IN__BYTES.BHR refers to the address that is offset from the beginning of IN__BYTES by the same number of bytes that BHR is offset from the beginning of the defining structure. The period between the two labels makes this work.

The next four blocks of code in this procedure take the four time bytes, one at a time; MOVe them into AL; CALL H2ASCII to convert them to ASCII characters in AX; and MOVe the character pairs into the appropriate words in our other data structure, TIME__STRING. The first block, for example, moves the value in IN__BYTES.BHR (which was MOVed there from CH) into AL, then CALLs H2ASCII. That subroutine returns the ASCII digits for the hour (in reverse order, remember) in AX. The next line after the CALL moves that pair of characters as a word (i.e., a two-byte number) into the word measured by .HR from OUT__STRING. The bytes are switched in the process, so the tens' digit will come out first when the string is printed.

The procedure ends with one more function call — the same one used to print the string "Setting timer..." in the procedure TIMERON. In this instance, the string to be printed starts at the location in DATASEG that we labeled OUT__STRING, so what's MOVed into DX is OFFSET OUT__STRING. DS is already correct because OUT__STRING is inside DATASEG.

Coming Up

Next, we'll make some very slight modifications to the original SETPOINT. Then, we'll write a procedure to combine TIMERON, SETPOINT, and TIMEROFF into a program that will run on its own. And we'll look at different ways you can use MASM, LIB, and LINK to generate the executable file — and, if you wish, to make these subroutines available for use in other benchmarking programs.

```
page   54,132
title  H2ASCII: convert a hex byte to two ASCII characters

; The byte is passed in al and the pair of characters
  is returned in ax
; The value of the byte may not be less than zero or
  more than 99
; Uses registers ax, cx

; Code segment

codeseg  segment para   public   'code'

         assume  cs:codeseg

; Procedure

         public  h2ascii

h2ascii  proc    near

         cbw                    ; convert byte in al to a word
         mov     cl,10          ; put divisor (10) in cl
         div     cl             ; divide word in ax by 10
         add     ah,'0'         ; convert ones' digit to ASCII
         add     al,'0'         ; convert tens' digit to ASCII

         ret                    ; return to calling procedure

h2ascii  endp                   ; end of procedure

codeseg  ends                   ; end of code segment

         end                    ; end assembly
```

```
page   54,132
title  TIMERON.ASM: set the time to 00:00:00.00 via DOS
                    Set Time

; This procedure prints a message,
  sets the time to 00:00:00.00 and exits
; Uses registers ax, cx, dx

; Data segment

dataseg  segment para  public  'data'

message  db     'Setting timer.  ','$'  ; message to appear
                                          on screen

dataseg  ends                    ; end of data segment

; Code segment.

codeseg  segment para  public  'code'

         assume cs:codeseg,ds:dataseg

; Procedure

         public  timeron

timeron  proc    near

         mov     ax,dataseg  ; initialize dataseg's register
         mov     ds,ax

         mov     dx,offset message  ; put offset of
                                      string into dx
```

```
        mov     ah,9            ; function no  for Print String
        int     21h             ; interrupt for Function Request

        xor     cx,cx           ; set time registers to zeros
        xor     dx,dx

        mov     ah,2dh          ; function no  for Set Time
        int     21h             ; interrupt for Function Request

        ret                     ; return to calling procedure

timeron endp                    ; end of procedure

codeseg ends                    ; end of code segment

        end                     ; end assembly
```

```
page    54,132
title   TIMEROFF.ASM: print the time as hh:mm:ss:ff
        (24-hr clock)

; This procedure prints the time (24-hour clock) followed
  by the word 'elapsed'
; Calls the external procedure h2ascii
; Uses registers ax, cx, dx

; Define a structure for saving the bytes from the
  registers

time_bytes   struc
bhr          db      ?                 ; gets hour from ch
bmin         db      ?                 ; gets minute from cl
bsec         db      ?                 ; gets second from dh
bfrac        db      ?                 ; gets 100ths of
                                         second from dl
time_bytes   ends

; Define a structure for building the string as the bytes
  are translated

time_string  struc
hr           dw      ?                 ; hour will go here
             db      ':'
min          dw      ?                 ; minute will go here
             db      ':'
sec          dw      ?                 ; second will go here
             db      '.'
frac         dw      ?                 ; 100ths of second
                                         will go here
             db      'elapsed'
             db      '$'               ; end-of-string
                                         for printing
time_string  ends

; Data segment

dataseg      segment para  public  'data'

in_bytes     time_bytes    <>          ; reserve space
                                         for the bytes
out_string   time_string   <>          ; reserve space
                                         for the string

dataseg      ends                      ; end of data
                                         segment

; Code segment

codeseg      segment para  public  'code'

             extrn   h2ascii:near      ; declare near
                                         external proc

             assume  cs:codeseg,ds:dataseg

; Procedure
```

```
                public   timeroff

timeroff        proc     near

                mov      ax,dataseg
                mov      ds,ax

                mov      ah,2ch               ; function number
                                                for Get Time
                int      21h                  ; interrupt for
                                                Function Request

                mov      in_bytes.bhr,ch      ; move hour from ch
                mov      in_bytes.bmin,cl     ; move minute from cl
                mov      in_bytes.bsec,dh     ; move second from dh
                mov      in_bytes.bfrac,dl    ; move 100ths of
                                                second from dl

                mov      al,in_bytes.bhr      ; get hour in al
                                                to work on it
                call     h2ascii              ; convert it to
                                                ASCII
                mov      out_string.hr,ax     ; build it into
                                                the string

                mov      al,in_bytes.bmin     ; get minute in al
                call     h2ascii              ; convert it to
                                                ASCII
                mov      out_string.min,ax    ; build it into
                                                the string

                mov      al,in_bytes.bsec     ; get second in al
                call     h2ascii              ; convert it to
                                                ASCII
                mov      out_string.sec,ax    ; build it into
                                                the string

                mov      al,in_bytes.bfrac    ; get 100ths of a
                                                sec in al
                call     h2ascii              ; convert it to
                                                ASCII
                mov      out_string.frac,ax ; build it into
                                                the string

                mov      dx,offset out_string  ; offset of
                                                  string into dx
                mov      ah,9                 ; function no. for
                                                Print String
                int      21h                  ; interrupt for
                                                Function Request

                ret                           ; return to calling
                                                procedure

timeroff        endp                          ; end of procedure

codeseg         ends                          ; end of code segment

                end                           ; end assembly
```

All tangled up in red tape?

Let CHECKZ uncomplicate your home or business records & increase productivity.

CHECKZ

- Continuous check printing
- Report sort on user defined categories
- Report sort on deductibles
- Maintains list of accounts, addresses & account numbers
- Password security
- Searching, reconciling, & archiving facilities
- Ideal for small business or individual

CP/M-Z 80, PCDOS, MSDOS Formats
$49.95 Disk and Manual
$5.00 Manual Alone

For further information or orders:
FUTURE COMMUNICATIONS
210 West Benson Street, Atlanta, Georgia 30030

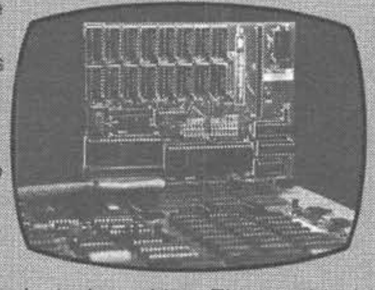
Controlled Data Recording Systems Inc.
Quality Products and Support for the Heath/ Zenith Community

New for the H/ Z89-90 Computer Users:
The Super RAM 89 Package

Heath/ Zenith 89 and 90 computer users can now get the speed and power of a high capacity Ram Drive System at a reasonable price.

Using the Ram 89 package, standard software shows an immense improvement in speed. Depending on the software being run, programs may execute 10 times faster than when run through standard floppies.

Ram 89 comes as a two card set that plugs into the left hand (16k expansion) side of the computer. No computer modifications required.



Board 1 has two banks of 256k chips possible for a total of 512k. Either or both banks are able to use 64k chips instead. Board 1 can be used by itself, with board 2 added at a later date.

Board 2 has an additional 512k, plus it has a real time clock capability, and a SASI interface hardware capability. Board 2 piggybacks onto board 1.

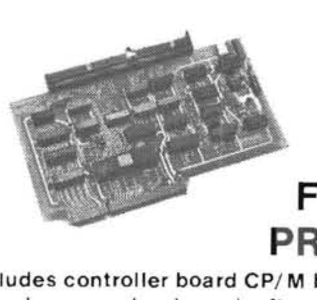
The Ram Drive Software (SRAM) allows one or two logical ram drives. The ram drive(s) can be located starting anywhere from logical A: to O: (standard drives get relocated). SRAM can be set to start at logical A: and warm boot with ram(no floppy disk accesses needed). Ram drive attatches to any of the versions of CP/M 2.2 bios used in the H/Z89-90.

GET A FULL MAGABYTE OF RAM FOR YOUR H/ Z89-90

Board 1 includes hardware manual and ram drive software with no ram: $190.00. Each 256k bank, add $56.00. Board 2 (must have board 1). With no ram, no clock, no SASI $90.00. Each 256k bank, add $56.00. Ask about clock, SASI pricing.

FOR THE H/ Z89-90 COMPUTERS

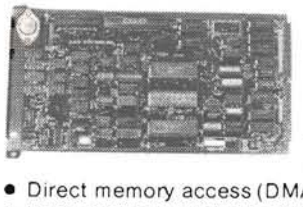
THE ONE CONTROLLER FOR 8" & 5.25" DRIVES THE FDC-880H PRICE:$395



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

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. The FDC-880H operates with or without the Heath hard sectored controller.

THE FDC-H8 DOUBLE DENSITY 8" AND 5.25" CONTROLLER FOR THE H8 COMPUTER PRICE $495



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

- 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.

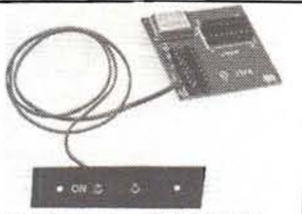
MODIFY89 Allows owners of the FDC-880H or the FDC-H8 to read/write to the following disktypes:

Access S.S.D.D., Cromemco S.S.D.D., DEC VT180 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-8001 A S.S.D.D., Osborne S.S.D.D., Osborne S.S.D.D., Otrona D.S.D.D.*, Zorba D.S.D.D., TI Professional S.S.D.D., TRS-80 Model I (Omnikron CP/M), TRS-80 Model III (MM CP/M), Xerox820 S.S.D.D., Zerox820-II S.S.D.D. Plus more, Standard (Tests for H/Z37 and C.D.R. Disk types)

PRICE $49.95

FOR THE Z100 COMPUTERS

THE ORIGINAL Z100 SPEED MODULE. RUN YOUR Z100 PROGRAMS FASTER



The ZS100 runs the Z100 CPU 50% faster, (7.5 MHZ) in 8088 mode. The ZS100 installs easily with no soldering. The ZS100 is externally switchable between speed mode and normal. The ZS100 improves the time performance of applications packages with no software modifications needed. The ZS100 is for all Heath/Zenith 100, 110 and 120 series computers.

New: a powerful floppy drive package for any computer that can use 8" floppy drives.

Two extended technology 5.25" drives in a package with interface cables case and power supply, with the capability of looking exactly like two standard 8" drives to the computer.

Now have **2.4 megabytes** of **5.25" floppy file storage capacity** in one package. A switch enables the drives to look like 8" drives or 5.25" 96 TPI drives.

ASK FOR PRICE.

For information about these products contact:
Controlled Data Recording Systems Inc.
7210 Clairemont Mesa Blvd.
San Diego, Ca. 92111 Phone (619) 560-1272 or contact a Heath/Zenith dealer that carries the C.D.R. Systems Inc. line of products.
New - C D.R. Systems Bulletin Board (619) 560-8929

MUVIT

Louis M. St.Martin
860 Hillcrest Drive
Pomona, CA 91768

A BASIC Boost For Spreadsheets

A Problem Is Recognized And Its Solution Appears

An annoying repetitive frustration I've experienced since getting into personal computing in 1978, is the often encountered need to manually transfer data between files to accomplish some task. One such boring job is accumulating the data in my checkbook spreadsheet and re-entering it in my monthly cash summary spreadsheet. I'd been doing this for two years and knew of no satisfactory alternative until Software Toolworks' MYCALC(tm) spreadsheet provoked me to explore an avenue out of my difficulty.

On page 31 of the MYCALC instructions, Software Toolworks describes the expression used to refer a position in a spreadsheet to other spreadsheets on the current disk; e.g., the expression "=data[x57]" can be used to transfer data from box "x57" of the spreadsheet named "DATA.MC". It is just that simple; to pick up the amount of a check written to pay the gas bill, for example, it is only necessary to enter in my cash summary spreadsheet the checkbook spreadsheet coordinates of the box whose contents are the amount of the gas bill.

However, it was quickly apparent I would have to enter my gas bill payment on exactly the same line month after month if the cash summary spreadsheet was to find it faithfully. And, what happens if I write more than one check against an account; say I paid two months' gas bills at two different times in a given month? How could the cash summary sheet know it was to sum all the payments to the gas company? MYCALC assumes a one-to-one correlation between source and destination spreadsheets. So, after reconciliation to this disappointment, I set out to develop means for picking up data at random locations and summing multiple entries.

Listings 1 through 4 constitute an MBASIC(tm) program that does that. It picks up data entered at random in a checkbook spreadsheet, sums each account, and posts the results to an intermediate spreadsheet which my cash summary spreadsheet reads.

So you may better understand what I set out to accomplish and how MUVIT works, I will describe my MYCALC spreadsheets and their data files, then the solution I've developed.

Description Of MYCALC Files

Figure 1 is an example of a checkbook spreadsheet, CHEX-784.MC, for July 1984. The layout of this spreadsheet is such that the account any check was written to cover is in column "f" and the amount of the check is in column "g".

MYCALC disk files are in ASCII format and Figure 2 is the first page of that generated by MYCALC from the checkbook spreadsheet of Figure 1. Scanning Figure 2, we can easily spot those of interest here, i.e., those in which either an "f" or a "g" follows the character ">" which is interpreted by MYCALC as "Go to". The characters between the ">" and the colon define the spreadsheet box address where data or instructions are to be placed. The colon, ":", delimits the box address, separating it from the data or instruction in the box. Text data are preceded by a double quotation mark. Numerical data are in exponential format. Expressions containing back slashes, "\", are instructions to MYCALC for setting column widths, formulas for calculations, etc.

Figure 3 is a copy of a section of a cash summary spreadsheet. The data from CHEX784.MC have already been posted using MUVIT. For clarity, data for months other than July have been suppressed.

MUVIT Is Described, Briefly

The purpose of MUVIT is to transfer data from a Source spreadsheet to a Destination spreadsheet. Specifically, the Source is a spreadsheet serving as a check register and the Destination is a cash summary spreadsheet. To each of these spreadsheets there is a corresponding disk file; i.e., a Source file and a Destination file. MUVIT creates two intermediate disk files, the first of which, the Transfer file, contains the list of accounts in the Source file to be scanned, summarized, and assigned to specific addresses in the Destination spreadsheet. The second file created by MUVIT is the Link file to be read by MYCALC as input to the Destination spreadsheet. The transfer file is used repeatedly, every time data is to be transferred from the check register spreadsheet to the cash summary spreadsheet. A link file, on the other hand, is used only once. A new link file is created each time a transfer of check register data to the cash summary occurs.

MUVIT consists of four modules, MUVIT.BAS, TWO.BAS, THREE.BAS, and FOUR.BAS. Listing 1, MUVIT.BAS — the entry module, initializes controls for the display and routes the user to the other modules depending on whether a) an existing transfer file is to be used, b) a new transfer file must be created, or c) an existing transfer file is to be modified. In the most frequent circumstances, an existing file will be used, so the user is routed immediately to FOUR.BAS where final steps are taken to create the link file. If a new transfer file is to be created, the user is routed to TWO.BAS where the needed data is entered and written out to the disk for use in the final step. If an existing file is to be modified, the entry module routes the user to THREE.BAS where the changes are made and incorporated in the transfer file. Both these latter modules finish by routing the user to FOUR.BAS. In FOUR.BAS the user is asked to name the source file to be scanned for data for creating the link file to be read by MYCALC into the destination spreadsheet.

Running MUVIT

Assuming you have your check register and cash summary spreadsheets designed and in use, copy Microsoft BASIC — any flavor (MBASIC, ZBASIC, GW-BASIC, etc.), MUVIT's four modules, and your check register spreadsheet(s) to a disk. Boot up and enter "MBASIC MUVIT" at the prompt. Answer the request for a transfer filename with anything suiting your fancy. Respond with your intent to create a new file. Enter the information relating to your check register.

The first time through you will create a Transfer file by entering the account names and their locations taken from your cash summary spreadsheet. Having completed this, the program takes over to construct your transfer file, giving you opportunity to review and, if necessary, correct it. When you are satisfied, MUVIT proceeds to the final phase to construct the link file, the end product you will read into your cash summary spreadsheet. You will be asked to enter the filename of your check register. MUVIT's final effort is to write the link file to disk for your use.

The time for this first excursion depends on how long you spend entering your cash summary spreadsheet data. Subsequent runs using a transfer created from the data in the example in this article consume about one minute and ten seconds.

When MUVIT completes its task, go to MYCALC, call up your cash summary spreadsheet and when it appears, read in your link file. If all is well, your check data will overlay the cash summary spreadsheet. Remember — when you save the updated cash summary to change the filename from the link filename to that of your spreadsheet, otherwise, you will have an updated cash summary spreadsheet living under a link filename and your link file will have vanished. Fortunately, your transfer file will be intact so you can go back to MUVIT and quickly recreate the link file, if you feel you need it.

A More Detailed Look — Much More

The Entry Module — MUVIT.BAS (Listing 1)

Line 1140 notifies MBASIC of MUVIT's intent to use some variables generated in the first module, MUVIT.BAS, in later modules. Lines 1170 and 1180 establish strings to position the cursor at the vertical center of the screen, to clear the display, and get the user's attention by sounding the bell.

Lines 1230 through 1250 accept a transfer filename entry from the keyboard and search for it on the default disk. If it is found, line 1310 asks the user if the file is to be modified. If modification is not desired, the program chains to FOUR.BAS for immediate preparation of the link file. On the other hand, if the user responds with an intent to modify the transfer file, the program chains to THREE.BAS. If the program does not find the desired transfer filename (entered at line 1250) on the default disk, the program — via the ON ERROR and ERL commands — quizzes the user as to whether a new transfer file is to be created or was the entry an error. If it is an error, the user is given opportunity to re-enter. If a new transfer file is to be created, the program chains to TWO.BAS for that purpose. If none of this pleases the user, the program responds with the message in line 1410 and quits.

Listing 1 — MUVIT.BAS

```
1000 'MUVIT - Transfers data between MYCALC(tm)
     spreadsheets (see line 1420)
1010 '
1020 '              by Louis St.Martin
1030                 860 Hillcrest Drive
1040 '                Pomona, CA  91768
1050 '                  714/622-3248
1060 '
1070 'Written in Microsoft BASIC-80(tm), Revision 5.21
     (see line 1430)
1080 '
1090 '
1100 '         ENTRY MODULE - as of 3/28/85
1110 '
1120 ' *************** INITIALIZATION *****************
1130 '
1140 COMMON CL$,TRANSFER$
1150 '                  DISPLAY CONTROLS
1160 '
1170 HERE$ = CHR$(27) + CHR$(89) + CHR$(43) + CHR$(32)
1180 CL$ = CHR$(27) + CHR$(69) + HERE$ + CHR$(7)
1190 '
1200 ' ************ SELECT TRANSFER FILE ***************
1210 '
1220 PRINT CL$
1230 PRINT "In the next instruction you will be asked to
      enter a TRANSFER filename which"
1240 PRINT "is the list of accounts and line numbers in
      your DESTINATION spreadsheet."
1250 INPUT "Enter the TRANSFER filename, ALL CAPS, PLEASE ";
      TRANSFER$
1260    ON ERROR GOTO 1340
1270 OPEN "I",1,TRANSFER$
1280    ON ERROR GOTO 0
1290            CLOSE#1
1300 PRINT CL$
1310 INPUT "Do you want to modify this file" ; A$
1320    IF A$ = "Y" OR A$ = "y" THEN CHAIN "THREE"
1330            CHAIN "FOUR"
1340 '
1350    IF ERL = 1270 THEN PRINT
            "File not found. Existing files are:" :
            PRINT CHR$(7) : FILES . PRINT   PRINT
1360 INPUT "Is the file you want on this list", A$ . PRINT
1370    IF A$ = "N" OR A$ = "n" THEN
            INPUT "Is this a new TRANSFER filename" ;
            A$ . GOTO 1400
1380    IF A$ = "Y" OR A$ = "y" THEN RESUME 1250
1390            GOTO 1360
1400    IF A$ = "Y" OR A$ = "y" THEN CHAIN "TWO"
1410 PRINT CL$ : PRINT "PROGRAM TERMINATED" : END
1420 ' MYCALC is a trademark of SOFTWARE TOOLWORKS
1430 ' BASIC-80 is a trademark of Microsoft Corporation
```

Getting started — TWO.BAS (Listing 2)

Since we must always start from scratch, it is appropriate the next module in the program and logically the next one we discuss is that which creates our first transfer file, TWO.BAS. At line 2030 the user is asked to repeat the name of the transfer file just in case there's been a change of heart about the name entered in the previous module. Line 2040 notifies MBASIC the user still intends to retain some assigned variables for later use. In lines 2060 through 2080, details about the source spreadsheet essential to creating the transfer file are requested and entered.

Line 2060 is the place you will begin to regret not having spent time designing your source (check register) and destination (cash summary) spreadsheets (see Spreadsheet Design below). First, at line 2060, you will enter the number of accounts in your destination spreadsheet. Next, you'll need some information on your source spreadsheet. Of course, if you slavishly adopted my layout, you will count the number of columns in my check register spreadsheet and enter that quantity — it's 10. Your guess at the number of checks you write each month is what you should enter in answer to the question in line 2080. I write about 30 to 40 but enter 100 in response to the request; it's a safe round number that makes me feel very prosperous. Don't get too ambitious, though. You'll run out of memory if your guess is too high! Line 2090 sets up the arrays for your account names and their line numbers, using the parameters just entered.

Beginning with line 2110 and ending at line 2150, the user enters the names of the accounts to be transferred from the source to the destination spreadsheet. When all data are entered, lines 2170 through 2290 display it on the video, and the printer if desired. At line 2300 the user is given opportunity to initiate corrections to the list of accounts and/or line numbers just input. Lines 2310 through 2340 accomplish the changes. After each change is entered, the program cycles back to line 2170 for another peek at the corrected list on the video and, if so desired, the printer for a recheck.

When satisfaction with the list is indicated by the user at line 2300, the program jumps to line 2350 to write the transfer file to disk. Then it chains to module "FOUR.BAS".

Listing 2 — TWO.BAS

```
2000 ' Module TWO - Installs a NEW TRANSFER file
      {as of 3/24/85}
2010 '                A MODULE OF MUVIT
2020 PRINT CL$
2030 INPUT "Enter filename of NEW TRANSFER file" ,
      TRANSFER$
2040 COMMON CL$, TRANSFER$
2050 PRINT
2060 INPUT "NUMBER OF ACCOUNTS" , N
2070 INPUT "Number of COLUMNS in check book spreadsheet" ;
      B1
2080 INPUT "Maximum number of checks per month" ;
      C : B = B1*C
2090 DIM ACCOUNT$(C), LLINE$(N)
2100 PRINT CL$
2110 PRINT CHR$(7) , "ENTER ACCOUNT NAMES AND LINE NUMBERS
      THEY OCCUPY ON DESTINATION SPREADSHEET"
2111 ' *********** REVIEW TRANSFER FILE ***************
2120 FOR A = 1 TO N
2130     PRINT   PRINT A,
2140    INPUT "ACCOUNT NAME, LINE NUMBER   " ;
        ACCOUNT$(A), LLINE$(A)
2150 NEXT A
2170 PRINT CL$
2180 FOR A = 1 TO N STEP 2
2190    PRINT "ITEM # " , A , ACCOUNT$(A) ; "   ,   " ;
        LLINE$(A) ;
2200                IF ACCOUNT$(A+1) = "" THEN 2230
2210    PRINT TAB(40) ; "ITEM # " , A+1 ; ACCOUNT$(A+1) ;
        "   , " , LLINE$(A+1)
2220 NEXT A
2230 PRINT : PRINT
2240 INPUT "Would you like a hard copy of this list" ,
      A$
                IF A$ = "Y" OR A$ = "y" THEN 2250 ELSE 2310
2250 PRINT CL$   INPUT "When printer is ready, hit RETURN",
      A$
                IF A$ = "" THEN 2260
2260 FOR A = 1 TO N STEP 2
2270    LPRINT "ITEM --> " ; A , ACCOUNT$(A) , ",   " ;
        LLINE$(A) ;
2280                IF ACCOUNT$(A+1) = "" THEN LPRINT " " :
                GOTO 2310
2290    LPRINT TAB(40) , "ITEM --> " ; A+1 ; ACCOUNT$(A+1) ,
        ",   " ; LLINE$(A+1)
2300 NEXT A
2301 ' *********** CORRECT TRANSFER FILE **************
2310 INPUT "Is this list of ACCOUNT NAMES and LINE NUMBERS
      correct (Y/N)", A$ .
                IF A$ = "Y" THEN 2360
2320    INPUT "Enter 'ITEM #' of incorrect ACCOUNT or
        LINE NUMBER" ; I
2330                PRINT TAB(30) ACCOUNT$(I),LLINE$(I)
2340    PRINT "Enter correct ACCOUNT NAME and LINE NUMBER"
2350                INPUT ACCOUNT$(I), LLINE$(I) : GOTO 2180
2351 ' *********** WRITE TRANSFER FILE TO DISC **********
2360 OPEN "O",1,TRANSFER$
2370    WRITE#1, B,C,N
2380                FOR A = 1 TO N
2390                        WRITE#1, ACCOUNT$(A), LLINE$(A)
2400                NEXT A
2410                        CLOSE#1
2420 CHAIN "FOUR"
```

Pass THREE, And Proceed To --> FOUR.BAS (Listing 4)

In keeping with our earlier determination to an orderly talk-through, we will move on to the final module, FOUR.BAS, to describe a complete transfer of data. We will return to discuss the modification module later.

Listing 4 is where it all comes together. Our transfer file is compared with the source file and where agreement exists, data is moved from the source to the link file to be read by the destination spreadsheet. Lines 4030 through 4160 open the transfer file generated by TWO.BAS to first determine the size of the arrays necessary later, input the account names and line numbers, and for surety, clears an array, EXPENSE(E) to all zeros because I don't trust MBASIC much more than I trust myself not to screw up when it comes to money matters!

In lines 4570 through 4660, MUVIT looks at what it sees in the arrays created from column "f" of the check register spreadsheet. If line 4620 finds a match with a string in the transfer file created in TWO.BAS, it links the amount it finds in column "g" on the same line of the check register spreadsheet with that account name. If an account name is encountered more than once, the associated amounts are summed so a total of all checks written to that account is created. Also, MUVIT eliminates blank entries (line 4590) and the label "FOR" from the top of column "f" (line 4600). Line 4610 eliminates strings containing a back slash, "\"; which are MYCALC instructions and of no interest to us in pursuit of MUVIT's tasks. The combination of lines 4620, 4640, and 4650 mentioning "FLAG2" alert the user via the video display of the existence of accounts in the check register spreadsheet that do not match the transfer file contents. The user takes appropriate action on being notified of such anomalies. He may ignore these entries, he may want to go back to the check register spreadsheet and correct some spelling, or perhaps a revised transfer file is in order.

If all has gone to the user's satisfaction to this point, the program writes the link file, notifies the user of its name, and returns to the operating system. The user now enters MYCALC, calls for reading in the cash summary spreadsheet and when it appears, calls for the link file to be read in as an overlay. Et, Voila! There appears, as by magic, the fruits of all this labor and fun; a new column in the spreadsheet displaying all the amounts of checks written and recorded in the check register spreadsheet, properly juxtaposed opposite the appropriate accounts.

Listing 4 — FOUR.BAS

```
4000 '   Module FOUR - MAIN TRANSFER PROGRAM {as of 3/20/85}
4010 '             A module of MUVIT
4020 '
4030 OPEN "I",1,TRANSFER$
4040 '
4050    INPUT#1, B,C,N
4060    DIM CONTENT$(B), FFOR$(B),ACCOUNT$(C),EXPENSE(N),
        AMOUNT$(B)
4070    DIM LINK$(N), LLINE$(N)
4080             FOR A = 1 TO N
4090                      IF EOF(1) THEN 4130
4100               INPUT#1, ACCOUNT$(A), LLINE$(A)
4110             NEXT A
4120 '
4130    FOR E = 1 TO N
4140             LET EXPENSE(E) = 0
4150   NEXT E
4160                      CLOSE#1
4170 '
4180 PRINT CL$
4190 '
4200 ' ****** OPEN DISC FILE AND INPUT CHECK DATA ******
4210 '
4220 PRINT CL$, "Enter name of check spreadsheet--
     ALL CAPS, PLEASE"
4230 INPUT FILENAME$ :
                 IF RIGHT$(FILENAME$,3) = ".MC" THEN 4240
                 ELSE  LET FILENAME$ = FILENAME$ + ".MC"
4240 ON ERROR GOTO 4270
4250 OPEN "I",1,FILENAME$
4260 GOTO 4320
4270 IF ERL = 4250 THEN PRINT "File not found.
                 Existing files are " . PRINT CHR$(7)
                 FILES : PRINT : PRINT
4280 INPUT "Is the file you want on this list"; A$ ·
                 PRINT
4290    IF A$ = "Y" OR A$ = "y" THEN RESUME 4220
4300 PRINT CL$   PRINT "PROGRAM TERMINATED" : END
4310 '
4320 PRINT CL$; "CHECK DATA BEING LOADED. PLEASE WAIT "
4330    FOR BOX% = 1 TO B : LET LAST = BOX%
4340                      IF EOF(1) THEN 4390
4350             INPUT#1, CONTENT$(BOX%)
4360               IF MID$(CONTENT$(BOX%),2,1) = "f"
                          THEN FFOR$(BOX%) = CONTENT$(BOX%)
4370               IF MID$(CONTENT$(BOX%),2,1) = "g"
                   THEN AMOUNT$(BOX%) = RIGHT$(CONTENT$(BOX%),
                   LEN(CONTENT$(BOX%)) -
                   INSTR(1,CONTENT$(BOX%), "·"))
4380    NEXT BOX%
4390                      CLOSE#1
4400 '
4410 PRINT CL$ , "LINK FILE BEING PREPARED. PLEASE WAIT "
4420 '
4430 ' ************ CATEGORIZE CHECK DATA **************
4440 '                    CONVERT LOWER CASE TO UPPER
4450 FOR T = 1 TO LAST
4460               IF FFOR$(T) = "" THEN 4530
4470      LET J = LEN(FFOR$(T))
4480        FOR I = 1 TO J
4490                      IF ASC(MID$(FFOR$(T),I,J)) > 122
                            THEN 4520
4500                      IF ASC(MID$(FFOR$(T),I,J)) < 97
                            THEN 4520
4510               MID$(FFOR$(T),I,J) =
                   CHR$(ASC(MID$(FFOR$(T),I,J)) - 32)
4520        NEXT I
4530 NEXT T
4540 '
4550 '********* SUM & ASSIGN EXPENSE TO ACCOUNT **********
4560 FLAG2 = 0
4570 FOR T = 1 TO LAST
4580    FOR A = 1 TO N
4590             IF FFOR$(T) = "" THEN 4660
4600             IF INSTR(1,FFOR$(T),"FOR") > 0 THEN 4660
4610             IF INSTR(1,FFOR$(T),"
4620             IF INSTR(1,FFOR$(T),ACCOUNT$(A)) > 0
                         THEN LET EXPENSE(A) = EXPENSE(A)
                         + VAL(AMOUNT$(T+1)) : FLAG2 = 1
4630      NEXT A
4640      IF  FLAG2 = 0 THEN PRINT FFOR$(T) ,
          " is an INVALID Account Name "
4650      FLAG2 = 0
4660 NEXT T
4670 '
4680 ' *********** WRITE LINK FILE TO DISC *************
4690 '
4700 PRINT CL$ + HERE$ ;CHR$(7)
           : INPUT "ENTER MONTH & YEAR {mm,yr} ";
           MONTH$,YEAR$
4710 LINK$ = "LINK"+MONTH$+YEAR$+".MC" .
         PRINT "Your Link filename is ", LINK$
4720 OPEN "O",1,LINK$
4730 '
4740    FOR D = 1 TO N
4750             IF EXPENSE(D) = 0 THEN 4780
4760         LINK$(D) = ">" + CHR$(VAL(MONTH$) + 97) +
             LLINE$(D) + ":" + RIGHT$(STR$(EXPENSE(D)),
             LEN(STR$(EXPENSE(D))) - 1)
4770        PRINT#1, LINK$(D)
4780    NEXT D
4790                      CLOSE#1
4800 '
4810 PRINT "JOB COMPLETE - GO TO MYCALC "; CHR$(7)
4820 SYSTEM
4830 END
```

Tidying Up — THREE.BAS (Listing 3)

THREE.BAS is included to accommodate those of us who are imperfect planners or who cannot foresee the inevitable expansion of data to clutter an otherwise tidy page. It allows the user to adjust an existing transfer file to accommodate changes in a destination spreadsheet. Account names, can be added, deleted, changed, or moved to different lines. As in TWO.BAS, provisions are made for reviewing and correcting the product before final writing to disk.

Upon exercising the option in the entry module, "MOVIT.BAS", to modify an existing transfer file, the program chains to "THREE.BAS" where the user is asked to enter the number of accounts in the modified file. The existing file is opened and array variables used in creating a new transfer file are established, (lines 3000–3160). The program jumps to line 3470 and prints out the existing (OLD) list of account names and line numbers on the video display (and printer, if desired) then resumes at line 3180.

In lines 3180 — 3200, the OLD and new (NNEW) numbers of accounts are compared and the program jumps to the appropriate area depending on whether no change, an increase or a decrease in the number was entered at line 3060. Let's now assume no change in number was entered and the user wants to correct an account name or destination line number assighment. The program jumps to line 3620 where the user is asked if the existing list is correct. If the answer is "N" (meaning "NO", of course), the program requests the Item number of the account name or line number to be corrected. It then prints the current data and waits for entry of the corrected data. Then it loops through the print routine again to give the user an opportunity to review the list after the modification. If the user agrees the list is now satisfactory, the program returns to line 3190 and falls through to line 3680 where the modified transfer file is written to disk.

Similarly, requests to make changes to the number of accounts are directed to line 3230 for increases or 3310 for decreases. Each time an account is added or deleted, the program displays the modified list and the user is afforded opportunity to correct as necessary. When the list is satisfactorily updated, the modified transfer file is written to disk.

A special test in the decrease routine checks to see if the user got into this territory by mistake. If so, the user is given opportunity to exit gracefully by entering a '0' at line 3340.

Listing 3 — THREE.BAS

```
3ØØØ ' Module THREE - Revises an EXISTING TRANSFER file
      (as of 4/15/85)
3Ø1Ø '                    A MODULE OF MUVIT
3Ø2Ø PRINT CL$
3Ø3Ø COMMON CL$,TRANSFER$
3Ø4Ø PRINT "Filename of EXISTING TRANSFER file is " ;
     TRANSFER$
3Ø5Ø    PRINT   PRINT "Enter the following. "
3Ø6Ø            INPUT "NUMBER OF ACCOUNTS" , NNEW
3Ø7Ø PRINT CL$
3Ø8Ø OPEN "I",1,TRANSFER$
3Ø9Ø    INPUT#1,B,C,N
31ØØ            OLD = N
311Ø DIM ACCOUNT$(OLD + NNEW), LLINE$(OLD + NNEW)
312Ø            FOR A = 1 TO OLD
313Ø                    IF EOF(1) THEN 316Ø
314Ø                    INPUT#1, ACCOUNT$(A),LLINE$(A)
315Ø            NEXT A
316Ø                    CLOSE#1
317Ø GOSUB 347Ø
318Ø IF NNEW = OLD THEN GOSUB 362Ø
      ' NO CHANGE IN # OF ACCOUNTS
319Ø IF NNEW > OLD THEN GOSUB 323Ø : ' INCREASED #
32ØØ IF NNEW < OLD THEN GOSUB 331Ø : ' DECREASED #
321Ø GOTO 368Ø
3211 '
322Ø ' ******** ADD ACCOUNTS TO TRANSFER FILE ***********
3221 '
323Ø PRINT CL$ ; "ENTER NEW ACCOUNT NAMES AND LINE NUMBERS
     THEY OCCUPY ON DESTINATION WORKSHEET."
324Ø            FOR A = OLD+1 TO NNEW
325Ø                PRINT   PRINT A,
326Ø        INPUT "ACCOUNT NAME, LINE NUMBER  " ;
                 ACCOUNT$(A), LLINE$(A)
327Ø NEXT A
328Ø LET OLD = NNEW : GOSUB 347Ø · GOSUB 362Ø
329Ø RETURN
3291 '
33ØØ ' ****** DELETE ACCOUNTS FROM TRANSFER FILE ********
33Ø1 '
331Ø Q = Ø
332Ø PRINT CL$ ; "You have indicated you want to decrease
     the number of accounts
333Ø PRINT "TRANSFER file  " ;
334Ø PRINT "Enter the ITEM number of the account to be
     deleted, OR enter a 'Ø'"
335Ø INPUT "to exit " ; D
336Ø IF D = Ø THEN PRINT CL$ , "PROGRAM TERMINATED " : END
337Ø FOR I = D TO OLD
338Ø    SWAP ACCOUNT$(I), ACCOUNT$(I+1)
339Ø    SWAP LLINE$(I), LLINE$(I+1)
34ØØ NEXT I
341Ø LET NNEW = NNEW+1 : Q = Q+1
342Ø    IF NNEW < OLD THEN 334Ø ELSE 343Ø
343Ø NNEW = NNEW - Q : GOSUB 347Ø
344Ø GOSUB 362Ø
345Ø RETURN
346Ø ' ******* PRINT CURRENT LIST OF ACCOUNTS *********
347Ø FOR A = 1 TO OLD STEP 2
348Ø    PRINT "ITEM --> " , A ; ACCOUNT$(A) ; ",  " ,
        LLINE$(A) ;
349Ø            IF ACCOUNT$(A+1) = "" THEN 352Ø
35ØØ    PRINT TAB(4Ø) ; "ITEM --> " , A+1 ; ACCOUNT$(A+1) ;
        ", " ; LLINE$(A+1)
351Ø NEXT A
352Ø PRINT : PRINT
353Ø INPUT "Would you like a hard copy of this list" , A$ .
                IF A$ = "Y" OR A$ = "y" THEN 354Ø ELSE RETURN
354Ø PRINT CL$ : INPUT
     "When printer is ready, hit RETURN"; A$
                IF A$ = "" THEN 355Ø ELSE 354Ø
355Ø            FOR A = 1 TO OLD STEP 2
356Ø                    LPRINT "ITEM --> " , A ,
                        ACCOUNT$(A) ; ",  ", LLINE$(A),
357Ø                       IF ACCOUNT$(A+1) =
                           "" THEN LPRINT " " : RETURN
358Ø                    LPRINT TAB(4Ø) ,
                         "ITEM --> " ; A+1 , ACCOUNT$(A+1) ;
                         ",  " , LLINE$(A+1)
359Ø            NEXT A
36ØØ RETURN
361Ø '* CHECK LIST OF ACCOUNTS FOR ACCURACY & COMPLETENESS *
362Ø INPUT "Is this list of ACCOUNT NAMES and LINE
     NUMBERS correct (Y/N)", A$ :
                IF A$ = "Y" THEN 368Ø
363Ø    INPUT "Enter 'ITEM #' of incorrect ACCOUNT or
        LINE NUMBER" ; I
364Ø    PRINT TAB(3Ø) ACCOUNT$(I),LLINE$(I)
365Ø    PRINT "Enter correct ACCOUNT NAME and LINE NUMBER"
366Ø            INPUT ACCOUNT$(I), LLINE$(I) :
                GOSUB 347Ø   GOTO 362Ø
367Ø RETURN
3671 '
368Ø ' * REVISION COMPLETE - WRITE TRANSFER FILE TO DISC *
3681 '
369Ø OPEN "O",1,TRANSFER$
37ØØ    WRITE#1,B,C,NNEW
371Ø            FOR A = 1 TO NNEW
372Ø                    WRITE#1,ACCOUNT$(A), LLINE$(A)
373Ø            NEXT A
374Ø                    CLOSE#1
375Ø CHAIN "FOUR"
376Ø END
```

General Comments

MUVIT is written in MBASIC Version 5.21.

The program takes advantage of the Chain command to reduce program load time for runs with an existing transfer file.

Spreadsheet Design for Compatibility with MUVIT

We started this project because MYCALC showed promise of liberation from much of the drudgery of transferring data between spreadsheets. We feel we've made considerable progress in removing the constraints on data transfer, but we are not completely free of some restrictions. The data MUVIT is to manipulate must be in specific columns in the check register and cash summary spreadsheets else gibberish results. The name of accounts checks are written to cover must be in column "f" and amounts associated with those accounts in column "g". Likewise, the month of January must be allocated column "b" in the cash summary spreadsheet. If these data are located otherwise, lines 4360, 4370, and 4760 in FOUR.BAS must be revised so the program will track the check register and cash summary spreadsheets.

At line 4220 the user is asked to enter the filename of the particular check register spreadsheet to be scanned for data to be transferred. I create a separate check register spreadsheet for each month; naming them "CHEX" followed by the month and year. Thus, I label my check register spreadsheet for July 1984, "CHEX784" — MYCALC very obligingly adds the extension "MC"; but have no fear, whether the user includes or ignores the extension, MUVIT looks after us and makes the appropriate adjustment to the filename in the IF-THEN-ELSE bits of line 4230.

In the event the filename that is entered does not exist on the default disk, rather than sending the user off to mill about in confusion, lines 4240 through 4290 gently query one about what is going on and display all the filenames on the disk. The user is given opportunity to correct his ways so matters can proceed. However, if the user is adamant and rejects MUVIT's understanding and kind treatment, it boots him off at line 4300 without ceremony.

Assuming peaceful relations are still extant, MUVIT notifies the user to be patient and begins at line 4320 to load the check register data to be bumped against the transfer file. Since we are interested in data in two specific columns in the check register spreadsheet, lines 4360 and 4370 ignore all data except that in columns "f" and "g". Line 4360 tucks strings matching transfer file strings into an array, "FFOR$(BOX%)". Line 4370 does an additional something; it strips the source data from column "g" of all characters to the left of the colon, ":", thus converting it to an exponential (albeit alphabetic) expression which it tucks into another array, "AMOUNT$(BOX%)".

At completion of the initial scan and transfer of the desired data from the check register spreadsheet to temporary arrays in memory, MUVIT again asks forebearance of the user while it modifies the language found in the check register spreadsheet so it can do its job faithfully. This is dictated of course, by MBASIC's intolerance to human inconsistency in mingling lower upper case text. This process begins at line 4450 to completion at line 4530.

The titles on the cash summary spreadsheet may be spelled out or abbreviated depending on your taste. I spell them out. However, I use abbreviations in my cash register entries (compare Figures 1 and 3). In Figure 3 Food & Household, Magazines, Maintenance, Miscellany, etc. are spelled out. In Figure 1 I have abbreviated some of these. When I create a transfer, I use abbreviations such as F&H, MAG, MAINT, MISC, ELEC, NEWS, PHONE, etc. Thus, my transfer file recognizes shorthand designators for those accounts. Make sure your abbreviation is a contiguous continuous string within the whole word you use in your check register, e.g., TelePHONE or MAGazine. Be warned, if abbreviations in the check register compress a name, the transfer file will not recognize it; do not add an "S" to MAG for example. No lasting harm is done if you do, but MUVIT will notify you the entry is INVALID and you will have to change an entry, either in the check register or the transfer file.

MUVIT affords the option of directing check data to a line differing from the line its account name appears on in the cash summary spreadsheet. In Figure 3, for example, the account name Food & Household is on line 29, but my transfer file directs check transaction results to line 31.

And Finally

The writer will dump MUVIT to a double density disc on receipt of a blank disc and $3.00 to cover handling and postage.

```
a   b   c       d     e               f              g         h         i
      CHECK/
 DATE  VISA  No.      TO              FOR        AMOUNT   DEPOSIT   BALANCE
             Balance 7/1                                             781.01
  7/1       272 Claremt Animal  Pet care      7.58  250.21   1023.64
  7/3       273 Pool & Patio    Home Maint 15.25 1743.09   2751.48
  7/5       274 Chevron         Gasoline    29.04            2722.44
            275 Auto Club       Car Ins.    98.44            2624.00
            276 YMCA            Charity     15.00            2609.00
            277 Heathkit        Fred       100.00            2509.00
            278 Pomona City     Water       63.00            2446.00
            279 Cal Fed         Mortgage   203.70            2242.30
            280 LA Times        News        20.00            2222.30
            281 Edison Co.      Elec        43.03            2179.27
            282 SoCal Gas       Gas         21.54            2157.73
            283 Gemco           Alice       81.00            2076.73
            284 Dr  Heathman    Medical     42.00            2034.73
  7/6       285 Angel's         Home Maint 13.15            2021.58
            286 D&S             Car Maint  47 11            1974.47
            287 Cash            Alice       80.00  134.03   2028.50
  7/10 5261386 Acme Towing      Car Maint  25.00            2003.50
       5716828 F*H Tire         Car Maint 109.14            1894.36
       5067824 Pomona Newstand Mags        16.60            1877 76
  7/17                                             80.00    1957 76
  7/18      288 Dover Books     Books        4.85            1952.91
            289 Postmaster      Postage     20.00            1932.91
  7/20      290 Stater Bros     F&H         19.67            1913.24
  7/21 5301250 Walters'         ACLU        19.76            1893.48
  7/25      291 Stater Bros     F&H          8.95            1884.53
            292 Pool            Home Maint  6.38            1878.15
  7/26      293 Ole's           Home Maint 14.41            1863.74
  7/30 5986232 Xochomilco       F&H         10.06            1853.68
```

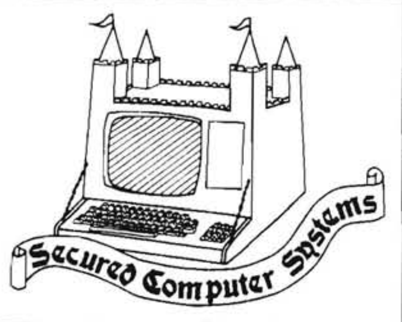
Figure 1 — Check Register Spreadsheet

```
>a1."a
>a1:\awc1
>b1:" DATE
>b1:\awc6
>c1:"CHECK/VISA No.
>c1:\afcf0
>c1:\awc14
>d1:\awc1
>e1:"           TO
>e1:\awc20
>f1:"       FOR
>f1:\awc15
>g1:" AMOUNT
>g1:\awc9
>h1:" DEPOSIT
>i1:" BALANCE
>b3:"
>d3:"
>e3:"Balance 7/1
>i3:7.8101E+02
>b4:"  7/1
>c4:2.72E+02
>e4:"Claremont Animal
>f4:"Pet care
>g4:7.58E+00
>h4:2.5021E+02
>i4:=i3+h4-g4#1.02364E+03
>b5:"  7/3
>c5:2.73E+02
>e5:"Pool & Patio
>f5·"Home Maint.
>g5:1.525E+01
>h5:1.74309E+03
>i5:=i4+h5-g5#2.75148E+03
>b6:"  7/5
>c6:2.74E+02
>e6:"Chevron
>f6:"Gasoline
>g6:2.904E+01
>i6:=i5+h6-g6#2.72244E+03
>c7:2.75E+02
>e7:"Auto Club
>f7·"Car Insurance
>g7:9.844E+01
>i7:=i6+h7-g7#2.624E+03
>c8:2.76E+02
>e8:"YMCA
>f8:"Charity
>g8:1.5E+01
>i8:=i7+h8-g8#2.609E+03
>c9:2.77E+02
>e9:"Heathkit
>f9:"Fred
>g9:1.E+02
>i9:=i8+h9-g9#2.509E+03
>c10:2.78E+02
>e10:"Pomona City
>f10:"Water
>g10:6.3E+01
>i10:=i9+h10-g10#2.446E+03
>c11:2.79E+02
```

Figure 2 — Partial Listing Of CHEX784.MC In MYCALC Format

```
 1                   a                          h
 2  CASH SUMMARY - 1984 :                      JULY
 4
 5          INCOME:
 6
 7      Salary:                               2,370.21
 8      Interest (+Earned -Withdrawn):          729.03
 9      Medical Insurance Benefits:
10
11         Total Income for Month:            3,099.24
12
13         EXPENDITURES:
14
15      Personal Allowances:
16        Fred                                  100.00
17        Alice                                 161.00
18
19      Taxes:
20        Income Tax
21        Property Tax
22
23      Operating Expenses:
24        Utilities:
25          Electricity                          43.03
26          Gas                                  50.58
27          Water                                63.00
28          Telephone
29      Food & Household:
30        Cash                                  400.00
31       Checks                                 38.68
32      Home Ownership:
33          Mortgage                            203.70
34          Home Insurance
35          Home Maintenance                              49.19
36        Bank Charges
37
38      Medical - Paid Out                                42.00
39
40      Car Ownership:
41        Gasoline                                        29.04
42        Car Maintenance                                181.25
43        Car Insurance                                   98.44
44        Licenses
45
46      Donations:
47        Humane Society
48        Charity                                         15.00
49        Political
50        Other Donations
51
52      Pet Care                                           7.58
53
54      Newspapers                                        20.00
55      Magazines                                         16.60
56      Books                                              4.85
57      Postage                                           20.00
58      Miscellany                                        19.76
59
60        Total Expenditures:                          1,563.70
61        Budgeted Expenditures:                       1,548.22
62        Actual NET=Budget-Actuals:                    (15.48)
63          DELTA NET=Planned/Actuals:                   (0.01)
64
65  ***** INCOME LESS ACTUALS:      1,535.54
66  ***** SAVINGS IMPACT :                               806.51
```

Figure 3 – Cash Summary Spreadsheet



HEATH/ZENITH 88, 89, 90 PERIPHERALS

16K RAM EXPANSION CARD

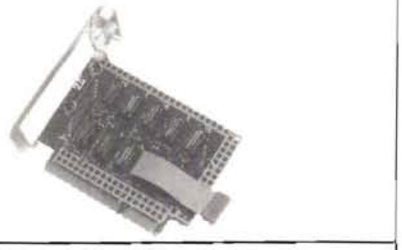
Expand your H/Z 88, 89 RAM Memory to a **FULL 64K** and begin using larger and more powerful programs with our **16K RAM** card.

Fully compatible with: Magnolia Microsystems and CDR CP/M and disk I/O interface cards.

Featuring: Complete installation instructions • Mounting bracket • 90 day warranty

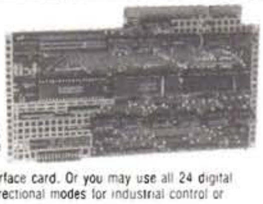
Field reliability record exceeding 3 years.

Only $65.00
Shipping & Handling $5.00



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

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



Your H/Z 88, 89, 90 can now directly connect and operate EPSON, IDS, ANADEX, GEMINI, SILVER REED, NEC, SUPER 5, PROWRITER, OKIDATA, and many more line printers using CENTRONICS style parallel interface with our **2/3rds**, 2 port serial, 3 port parallel interface card. Or you may use all 24 digital lines in various configurations of input, output, or bidirectional modes for industrial control or data sampling.

Features:
• 2 Serial Ports Supporting Ring Input, and External Clock • 3 Parallel Ports, 24 Total Digital Input/Output Lines • Fully Compatible with All Models of H/Z 88, 89, 90 using Heath/Zenith CP/M or HDOS • Now Supporting CP/M Version 2.2.04 • Choice of Centronics Line Printer Support Software for the CP/M or HDOS Operating Systems • Reduced Computer Bus Loading and Chip Independent Design

Complete with Installation Instructions, Documentation, 90 Day Warranty, Two Serial Cables and a Parallel Cable Internal to the Computer.

Price $199.00 **Second Operating System Driver $25.00**
Shipping & Handling $10.00

REAL TIME CLOCK

You will be able to perform time and date stamping for point of sales software, and bulletin board software or perform time studies as well as real time data sampling with our **REAL TIME CLOCK.** This peripheral card is a perfect companion to our 2/3rds card for industrial control and data sampling. STOP WATCH time study and alarm demonstration software is included for the CP/M operating systems. You will be able to view the current date and time on screen continuously or simply listen to an audible beep every fifteen minutes and the hour chimed or disable the clock entirely at your option

Features:
• True I/O Addressing, Not Memory Mapped • User Selectable Address • Rechargeable Battery Backup Using Commonly Available Batteries • Installable on the Left or Right Side of the Computer (Left side operation requires our I/O expansion module.) • Month, Day, Year, Hour, Minutes, Seconds, 1/10's, 1/100's and 1/1000's of Second of Accuracy • Interrupt Capability Based on Tenths of Seconds, Seconds, Minutes, Hour, Day, Week or a Specific Date and time • CP/M Operating System Software Driver and Demonstration Programs • HDOS is not Supported.

Price $130.00 with Batteries **$114.00 without Batteries**
Shipping & Handling $5.00

HDOS is a reg. trademark of the Heath Co. CP/M is a reg. trademark of Digital Research
PRICES ARE LESS SHIPPING AND TAX IF RESIDENT OF CALIFORNIA
MAIL ORDER, 12011 ACLARE ST., CERRITOS, CA 90701 (213) 924-6741
TECHNICAL INFO/HELP: 8575 KNOTT AVE., SUITE D, BUENA PARK, CA 90620 (714) 952-3930

ZENITH data systems SERVICE CENTER

TERMS AND SPECIFICATIONS SUBJECT TO CHANGE WITHOUT NOTICE — VISA AND MASTER CARD GLADLY ACCEPTED

Transfer "Foreign" Files in Seconds

NOW AVAILABLE FOR Z-100 AND Z-150

No modems and RS-232 links required!

ONLY $39.95!

READ, WRITE and FORMAT more than 70 different CP/M and MS-DOS disks on your computer

with MEDIA MASTER™

Tired of hunting for software in your computer's disk format? Do you need to exchange your Lotus 1-2-3, dBase II, Multiplan, and Wordstar data files among different CP/M and MS-DOS computers?

Then MEDIA MASTER is for you!

MEDIA MASTER supports the following disk formats:

Actrix (SSDD)	H/Z Z-100 Z-DOS 1.xx (SSDD)	LNW-80 (SSDD)	Sanyo (DSDD)
Actrix (DSDD)	H/Z Z-100 Z-DOS 1.xx (DSDD)	Lobo Max-80 (SSDD)	Superbrain (DSDD)
Avatar TC10 (DSDD)	H/Z Z-100 Z-DOS 2.xx (SSDD)	Lobo Max-80 512 (SSDD)	Superbrain Jr. (SSDD)
Casio FP 1000 (DSDD)	H/Z Z-100 Z-DOS 2.xx (DSDD)	Micral 9050 CP/M-80 (DSDD)	Systel II (SSDD)
Chameleon CP/M-80	H/Z Z-90 40 trk, 1 k blk (SSDD)	Morrow MD 11 (DSDD)	Systel III (DSDD)
Columbia MPC CP/M-80	H/Z Z-90 40 trk, 2 k blk (SSDD)	Morrow MD 2 (SSDD)	TI Professional CP/M-86 (DSDD)
Cromemco CDOS (SSDD)	Heath w/Magnolia CP/M (SSDD)	Morrow MD 3 (DSDD)	TRS-80 III FEC CP/M (SSDD)
Cromemco w/Int'l Term (DSDD)	IBM PC CP/M-86 (DSDD)	NCR Decision Mate 5 (DSDD)	TRS-80 III FEC T805 (SSDD)
Cromemco w/Int'l Term (SSDD)	IBM PC CP/M-86 (SSDD)	NEC PC-8001A (SSDD)	TRS-80 III Hurr. Labs (SSDD)
DEC VT180 (SSDD)	IBM PC-DOS 1.xx (DSDD)	NEC PC-8001A (DSDD)	TRS-80 III Mem. Merch (SSDD)
Davidge (DSDD)	IBM PC-DOS 1.xx (SSDD)	Olympia ETX II (SSDD)	TRS-80 IV CP/M+ (SSDD)
Digilog (DSDD)	IBM PC-DOS 2.xx (DSDD)	Olympia EX100 (DSDD)	TRS-80 IV Mont. Micro (SSDD)
Epson Multifont (DSDD)	IBM PC-DOS 2.xx (SSDD)	Osborne (SSDD)	Teletek 40 trk (SSDD)
Epson QX-10 (DSDD)	IDEA Bitelex (SSDD)	Osborne 4 (DSDD)	Toshiba T100 (DSDD)
Fujitsu Micro 16s (DSDD)	ISM CP/M (DSDD)	Osborne Osmosis (SSDD)	TurboDos (DSDD)
Groupil III CP/M (DSDD)	Insight Dev. IQ-120 (SSDD)	Otrona (DSDD)	Wang Maws CP/M (DSDD)
H/Z Z-100 CP/M (DSDD) later	Kaypro II/2 (SSDD)	PMC Micromate (DSDD)	Xerox 820 II (SSDD)
H/Z Z-100 CP/M (SSDD)	*Kaypro 4, 10 (DSDD)	Reynolds & Reynolds (SSDD)	*Zorba 40 trk (DSDD)

*****_Requires access to "foreign" computer_

MEDIA MASTER _is also available for DEC Rainbow, Osborne, and Kaypro computers._

NEW! — FILE MATE™

A Dozen Utilities in One!

ONLY $29.95

Finally — a powerful, easy to use file utility that performs a host of functions. **FILE MATE's** numerous text sifting and paragraph reformatting modes help you exchange "foreign" text files. Other file inspection and repair tools include two binary dumps, an ASCII/HEX file compare, and CRC "fingerprint." FILE MATE is available for popular CP/M and MS-DOS computers.

INTERSECTING CONCEPTS
(formerly MDG & Associates)

4573 Heatherglen Ct., Moorpark, CA 93021

ORDERING INFORMATION:
Include $3 per order for postage/handling. Overseas airmail add $10. To place COD orders:

Call TOLL FREE 24 hours:
800-824-7888 (Ask for Operator 251)
Alaska, Hawaii: 800-824-7919 (Ask for Operator 251)
Technical questions: call 805-529-5073

An Update Of CONDOR 3

Relational Database Management System (rDBMS)

Richard L. Mueller, Ph.D.
11890–65th Avenue N.
Maple Grove, MN 55369

In the June issue of REMark, Stephanie Butzbach wrote a super article on my favorite database management system, CONDOR 3. The article, "A Review of the CONDOR Database Management System", covered CONDOR (really CONDOR 3, a relational database management system) in an overview fashion. She discussed briefly what is a database, what resources are required to run CONDOR, discussed some of the features and capabilities of CONDOR in a general way, without introducing specific CONDOR commands, and concludes with a Table of CONDOR Database File Specifications (containing such information as records per database file, bytes per record, etc.). An excellent article. Well done, Stephanie!!

What I want to do in this article is to expand on what Stephanie covered in a general way, and then be more specific by discussing the additions/enhancements that are available in the latest release of CONDOR 3 (2.11.12). The additions/enhancements will be the changes made since release 2.09, the version currently available from Heath/Zenith Data Systems for the H/Z-100 series. According to the Zenith Software Consultation Bulletin Board, the release level of CONDOR 3 for the H/Z-150 PC series is 2.11. However, it does not indicate the minor revision level (release 2.11 actually had a series of releases some having significant enhancements: 2.11.01 thru 2.11.12). My feeling is that the release level of CONDOR 3 for the H/Z-150 PC is somewhere between '.01' and '.12'; and not '.12' which is the current release.

The first part of my article is for all users interested in database management, and the second part is for current users of CONDOR who have release 2.09 or a later release, but have not yet gotten release 2.11.12. If there is enough interest in database management, CONDOR 3 in particular, I would like to follow this article with a series of articles covering the individual features of CONDOR by creating databases, maintaining them, sorting them, manipulating the data, generating reports, etc. Please let me know if such a series would be of interest to you by contacting me directly or through HUG.

Now let's talk about database management systems (usually referred to as DBMS packages or applications), in general. First of all, what do we mean by a "database"? There must be hundreds of answers to that question and each person you ask will give a different answer. As defined in the CONDOR documentation, "A database is an organized collection of information about one or more subjects. The subjects in the collection are related to one another, such as customer and inventory information as it relates to sales activity. Large or small, a database resembles a library or a set of file cabinets in an office. The user of a database can select and work with an entire database or any of its parts in many ways."

I don't have to tell any of you microcomputer users out there that there are numerous DBMS packages available; they are advertised in almost any computer magazine, as well as some non-computer magazines, such as Business Week. The most popular DBMS packages seem to be dBASE II followed by dBASE III. However, there are other database management systems just as good and just as powerful. CONDOR 3 is not as powerful in some ways or as flexible as dBASE II and III, but it is, in my opinion, much easier to use and has every feature and capability that I need in a DBMS package. You don't have to be a programmer to use CONDOR 3, whereas dBASE II requires one to be more than a casual user to use the package, like a programmer in some cases.

Over the past few months, I have looked at other DBMS packages and I have not found one yet that I would prefer over CONDOR 3. I purchased CONDOR for my H-100 about a year and a half ago and just updated it to the latest release, which I will talk about shortly. As Stephanie put it, "CONDOR is one of the best database management systems on the market."

PRICE is definitely a factor in choosing a DBMS package. Generally, the higher priced packages have more capabilities and features than the lower priced packages. As you can see, I said "generally". You will always find a few cases where the above is not true. You may come across a low priced DBMS package that has some of the features and capabilities of the high priced ones. And I'm sure the opposite is true.

Recently, I was looking at a review of something like 20 or so different DBMS packages, and the prices ranged from $389 to $995. This was looking strictly at DBMS packages as such. There are some "integrated" packages, such as LOTUS 1-2-3 that have some low-level database management capabilities. These I did not consider in the above pricing range.

Suggested price for CONDOR 3 is $650, for either the H/Z-100 or H/Z-150 PC series; this is the price that is shown in the Heathkit catalogs. However, I have seen this lower in advertisements by some of the discount software houses. To get started, one may wish to start with a lower priced CONDOR package called CONDOR Jr. (suggested price is $299). The Heathkit catalog and Zenith Data Systems refer to CONDOR Jr. as the CONDOR File Management System (FMS). This package, as you may guess by the price, does not have all the features of CONDOR 3, a relational database management system (referred to as rDBMS). Any of the "relational" features are missing from CONDOR Jr., as well as the powerful CONDOR Report Writer capability. One can start with CONDOR Jr. and upgrade later to CONDOR 3 at an upgrade price. Database files created by CONDOR Jr. are fully compatible with CONDOR 3.

The questions new users may have are: "Why should I purchase a database management system? What would I use it for, if I did purchase one? Would it benefit me?" And the list goes on. My feeling is that one's imagination is the limiting factor for uses of a DBMS package.

Let me just list a few uses for a DBMS package around the house. In fact, I have implemented a number of these for my own use. Once you start creating databases, you get "hooked" on it. You can't stop looking for more ideas of what to put into a database. For starters, how about a list of all your valuables such as TVs, VCRs, cameras, radios, portable stereos, etc. You can build a database to contain the name of the item, the type, the purchase price, the purchase date, the store or brand name, in what room is it kept, who in your family does it belong to, etc. Once the database is created, you can then list the valuables by individual, by type, by room in the house, or by whatever other field you may have in your database.

Other uses for databases are as follows: a mailing list which includes names, addresses, phone numbers, and other type of information; recipes; an inventory list of all furniture and appliances in your house; categorize all important magazine articles; maintain a database of all your computer books; car maintenance records; sporting events that your children (and yourself) participate in (save all kinds of records/statistics); stocks and bonds; coin or stamp collections; insurance policies; VCR tapes; all your LP albums and cassettes; and the list goes on and on.

For businesses, some examples are inventory, list of customers, list of vendors, employee records, etc. You may also want to maintain some information for various charitable organizations you may belong to. Again the uses are many.

With CONDOR 3, it's very easy to set up a database, once you determine what types of information you want to collect and maintain. You define the fields (the individual items of information, such as name or an address; both of these items are fields), the lengths of them (in terms of characters), and the type of data (such as alpha, numeric, dates, etc.). Once the database is defined (fields are defined), data information can be entered. After the data has been entered (this will really be an on-going activity), it's very easy to retrieve the data that you want, in the order that you want, in the format you desire, etc.

A nice feature of CONDOR 3 is the capability of writing selected (or all) fields from your database to one of several ASCII type format files that are supported by CONDOR. This allows your database data to be used with other applications written in PASCAL or assembly language, for example; the data can be used by BASIC programs; can be used by word processing packages and maillist packages, such as Wordstar's Mailmerge; and the data can be used by FORTRAN and COBOL programs. What this means is that you can use the "friendliness" and the "ease of use" of the CONDOR 3 package to input, update, and maintain data, then use other applications or packages stated above to process the data. This is definitely a major feature of CONDOR 3.

At this time, I would like to turn my discussion away from generalities of database management systems and CONDOR 3, and talk about specifics; namely, the enhancements, additions, and deletions that are in the latest release of CONDOR 3. Again, let me say that the following discussion assumes the reader is familiar with CONDOR 3 release 2.09 or later. Some information covered here may be repetition of features and capabilities already available in the CONDOR 3 release for the H/Z-150 PC series. However, I'm sure that some of the features are new and will be of interest to you.

First, let me talk about two features that relate to what was covered in Stephanie's article. The Specification Table listed the "records per database file" as 32,767. This has been increased to 65,534 and this already is available in the H/Z-100 PC version of CONDOR. The other information in this Table remains unchanged.

In addition to the data types already defined in release 2.09 and below of CONDOR, there is a new data type introduced in an early release of 2.11 of CONDOR (in 2.11.01 — 2.11.07). This new data type is called "decimal numeric", which can have up to 18 decimal places. This makes the use of CONDOR adaptable to maintaining data for FORTRAN and PASCAL programs, for example.

Other changes and enhancements made in the earlier releases of 2.11 of CONDOR 3 are the capability of having two-screen formats, adding a "where" clause and multiple equations to the COMPUTE command, and a new command PRINTER. Before the two-screen format capability was added, a user was restricted to one screen for defining the format of a database. Everything had to be cramped on one screen. This two-screen capability doubles the space for format definition. A toggle key, CRTL-N, allows the user to toggle back and forth between the two screens.

The PRINTER command directs output to the printer, screen, text file, or eliminates the output altogether. 'PRINTER FILE filename' sends the output to an ASCII file. 'PRINTER CRT' or 'PRINTER SCREEN' redirects printer output to the video screen. 'PRINTER OFF' eliminates output; and 'PRINTER ON' restores all print functions.

Somewhere along the long line of CONDOR releases, a TERM command was introduced (it is not in my release of CONDOR 3, 2.09), but now has been deleted. The purpose of TERM was to inform CONDOR of the type of computer terminal you had. Eight different standard terminals were specified for your use. However, there also was a capability to create a terminal identification file for a nonstandard terminal. But as I said above, it is no longer in the current release of CONDOR 3. It seems to me that CONDOR Computer Corporation provides the user with the proper version of CONDOR 3 based on the user's microcomputer.

A MENU system with on-line HELP was added to CONDOR 3, in release 2.11.08. The CONDOR 3 MENU system is a complete teaching aid designed to simplify the learning and use of CONDOR 3. Your job of creating a database, entering data, updating it, adding to it, and generating reports from it, will be done much easier if you use the MENU system when you first begin to use CONDOR 3. Once you become experienced and feel comfortable with building your own commands, the MENU system can be disabled. The user then enters all commands at the CONDOR Prompt.

All of the CONDOR commands can be accessed through the various options on these menu pages except for two commands, EXEC and CHDIR, which I will discuss shortly. To execute commands, you simply enter the number of the operation you want to perform and follow the instructions on the screen.

There is a main MENU with a series of second level submenus. The main MENU lists the most central operations to building your database and maintaining it. The second level menus lead to third level menus, the command building menus.

CONDOR 3 release 2.11.09 introduced a number of changes. First of all, the PRINT command was deleted. The function of the PRINT command (i.e., print out data) is now handled by the LIST command by specifying the '[P]' option. STAX was replaced with STATS. Special spacing characters were added to the LIST command to allow for printing of mail labels. The "where" clause was added to the LIST, CHANGE, PROJECT, STATS, and TABULATE commands.

Two utility commands were added for users of MS-DOS V2.11 or later. The PAUSE command, when used in a command procedure, halts the processor until any key is pressed. However, the user can abort the procedure during a PAUSE by using the F10 function key. The other utility command is the CD or CHDIR command, which allows the user to change directories or get the name of the current directory without leaving CONDOR.

Four new commands were added in release 2.11.09: COLOR, EXEC, GRAF, and TALK. The latter two are not yet available for the H/Z-100 series, but are available for the H/Z-150 PC series (IBM compatible version). I contacted CONDOR Computer Corporation and they assured me that the GRAF and TALK commands would be made available in the H/Z-100 version later this year and all registered owners will get an automatic update. They would not commit to an exact date, however.

The COLOR command allows the user to choose the background and foreground colors for five different areas on the CONDOR screen. The areas for which colors can be set are BORDER, TEXT, NAMES, VALUES, and ERRORS. BORDER refers to the square around the outer edge of the CRT screen. TEXT is any text which CONDOR prints (status messages, prompts, etc.) which is not a field name, field value, or error message.

NAMES refers to field names as they are shown whenever the dataset form is displayed. VALUES are words or numbers you type into the field, which are listed in commands, such as LIST and TABULATE. ERRORS refers to error messages. Any area may be set to blink on and off by using the word BLINKING in the command line.

The EXEC command allows the user to run other programs and commands from within CONDOR 3. I am quite pleased to see this command introduced. This allows me to call and run programs and utilities as part of my CONDOR procedure files. EXEC works with MS-DOS V2 and later. The programs and commands must have the extensions .COM and .EXE in order to be executed from within CONDOR 3. Please note that the extension must be specified when you try to call another program or command.

GRAF produces graphs from CONDOR 3 files. The graphs currently supported are the vertical bar chart, the horizontal bar chart, standard line chart, pie chart, and the scatter chart. Remember this is not available today in the H/Z-100 version of CONDOR 3.

The last of the four new commands is the TALK command. This command allows users to communicate with information services and other companies. A script file is necessary for the TALK command. There are 2 ways to use TALK with script files. One for connecting to an information service such as COMPUSERVE, SOURCE, etc. The second way is for batch communications.

The last significant change occurred with release 2.11.10: the "SMART" feature. The SMART feature allows users to save a sequence of commands in a file on disk and play back that sequence whenever it is needed. The SMART feature only works in conjunction with the CONDOR 3 MENU System.

From the documentation that I received with my release (2.11.12), I could not find anything that went into either release 2.11.11 or release 2.11.12. My feeling is that mostly minor changes were made and those changes were probably to fix some "bugs".

Hopefully, this article gives the non-database users some additional information on Database Management Systems (DBMS), and hopefully provides the current H/Z-100 CONDOR user (and an H/Z-150 PC user) with a summary of the new features and enhancements in CONDOR 3. As I said earlier, if there is enough interest in more information on CONDOR 3, I would be happy to put together a series of articles on the subject. Please let me know directly or through HUG.

Software Packages mentioned in this article:

Wordstar and Mailmerge — Micropro International
LOTUS 1-2-3 — LOTUS Development Corporation
dBASE II and III — Ashton-Tate

The H89 *SPEED* Center

4MHz mod $45

An easy to install plug-in module. No trace cutting or soldering. Speed may be toggled with software. Includes a replacement Z80A (4MHz). Software supports Heath CP/M 2.2.03 and 04 and CDR Systems BIOS. Call or write for support of other BIOS or HDOS. Specify disk format.

CDR Super RAM 89

Make your H89 a speed whiz with this board and the included RAM drive software. With our 6MHz mod and the RAM drive, the Heath BIOS assembles (using MAKEBIOS) in less than 1½ minutes! Note our discount prices!

Main board w/o RAM - specify disk format	**$175**
Expansion board w/o RAM	**$69**
Main board w/512K RAM	**$239**
System w/MEGABYTE RAM	**$369**
Clock option	**$45**
SCSI option	**$85**

REP3 - Automatic Key Repeat

Stop wasting time - make full use of the repeat function just like the Z100. Simple plug-in installation on your keyboard assembly. Hold any key down for half a second and the key begins repeating. Combine this with our 4MHz mod and make WordStar fly! Provision for a defeat switch.

Kit	**$35**
Assembled	**$45**

WSPATCH $20

Adds H19/H89 function key patches to versions 3.0 or 3.3 of WordStar. Key functions similar to the PIE editor. Includes provision for redefining the keys by the user. Also includes a printer driver for the Epson MX80/FX80 printers.

EMULATE

Allows the H89 to read/write to the following disk formats.

Actrix	Eagle II	Morrow MD	Superbrain Jr.
AMPRO	Epson QX-10	NCR DecMate 5	Televideo
Beehive Tpr	Fujitsu CP/M86	NEC PC-8001A	TRS80-1 CP/M
CDR	IBM CP/M86	Osborne 1	TRS80-3 CP/M
Cromemco	IMS 5000	Otrona	TRS80-4 CP/M
DEC VT180	Kaypro II	PMC MicroMate	Xerox 820
DEC Rainbow	Magnolia	Sanyo 1100	Zorba

Now includes 42 formats! Uses a modified version of CP/M 2.203 or .04 BIOS which is included with the program. Allows the use of virtual drives and reading of 40-track disks in an 80-track drive.

For H37 with Heath CP/M	**$59**

Check for C.D.R. and Magnolia versions.

6MHz mod $59

Similar to our 4MHz modification, but increases the CPU speed to 6MHz. Requires some soldering on the CPU circuit board. Includes a Z8OB (6MHz) CPU replacement. Some technical knowledge is recommended for installation. Call or write for more details. Specify disk format.

CDR Controllers

Double density disk controllers. Includes board, instruction manuals, ROMs and software. Allows control of 5 and 8 inch drives. Includes BIOS source code and some useful utilities. Please include your CP/M s/n when ordering.

For the H89 FDC-88OH	**$349**
For the H8 FDC-H8	**$429**

TIM2 - Real Time Clock

This circuit installs in the left hand expansion slots of the H89 or H89A. Can provide application programs with time and date information. Includes battery backup for continuous timekeeping when the computer is turned off. Program access to the clock is through an I/O port. The port address is user selectable by means of a jumper. Requires soldering 4 wires to the CPU board. A prepared ribbon cable is included.

Kit	**$65**
Assembled	**$75**
Software on disk - specify disk format	**$10**

DATESTAMPER

Product of Plu*Perfect Systems. Provides automatic time and datestamping for CP/M 2.2 files. Works with many real time clocks, including our own TIM2 product.

CP/M - specify disk format	**$45**

NEWWORD - the better word processor from NewStar Software

Faster than WordStar, same commands as WordStar, MailMerge feature is built-in, supports more printers, has an undelete function and excellent documentation. We offer it at a discount and also include a function/cursor key patch for the H19/H89. Part of the bundled software is the WORD plus spelling checker from Oasis - normally a $150 item by itself!

CP/M - specify format	**$85**
CP/M with the WORD plus	**$135**
MSDOS (Includes WORD plus)	**$195**

DISKETTES

High quality private label diskettes at an economy price.

5" SSDD 10 hard sector	**$16/box of ten**
5" DSDD	**$18/box of ten**
8" SSDD	**$26/box of ten**
8" DSDD	**$29/box of ten**

ANALYTICAL PRODUCTS **209/564-3687**

20663 Ave. 352 Woodlake, CA 93286

Specify Disk Format on Software

CALL OR WRITE FOR CATALOG PRICES SUBJECT TO CHANGE

Terms: Check or Money Order - VISA/MC - C.O.D.
Add $3 per order for shipping and handling
California residents add 6% tax

ZPC Update #1

Pat Swayne
HUG Software Engineer

ZPC is a HUG program that emulates an IBM PC or Z-150 on an H/Z-100 (dual processor) series computer. It was announced in the September REMark, and since then we have received many questions about it. This article will attempt to answer some of those questions. It also presents a small patch to the initial release of ZPC, and some corrections to the documentation.

PC Compatibility

Just about all of the questions are concerned with just how compatible ZPC is with the IBM PC. ZPC supports enough features of the PC ROM BIOS (including all text and graphics modes) that it can run just about any PC program that confines itself to performing I/O via the BIOS or the operating system. If you have 768k of memory in your system, it also emulates the memory that is on the color/graphics adapter card in a real PC. It can, therefore, run programs that write directly to video memory for either text or graphics displays, if your system has 768k of memory. Since approximately 80 to 90 percent of all IBM PC programs access video memory directly, ZPC is severely limited in its ability to run PC programs unless you have 768k of memory in your H/Z-100. If you have that much memory, ZPC will use the last segment (segment address B000 hex) for emulating PC video memory, and allow 640k or 704k (user configurable) of the remaining memory to be used by PC programs.

Many programs that access video memory directly on a PC also access other aspects of the hardware, which are not supported by by ZPC. Some programs will check the status of the video board before and/or after writing to video memory by reading a port, or they may change certain video parameters by writing to ports. The ports normally used for these operations are 3D8 (hex), 3D9, and 3DA. The Z-100 is only capable of decoding 8-bit ports, so when a program, for example, writes to port 3D9, the Z-100 "sees" port D9, which is one of the Z-100's own video control ports that does not operate the same way as the PC video port.

ZPC does not require that the video status be checked before a program writes to or reads from its emulated video memory, so the status checks in a program can be patched out. The altering of video parameters can also be either patched out or replaced by other code, and so programs can be made to run under ZPC that at first will not. There are other hardware problems that can also be patched out.

Patches for some programs are included on the ZPC disk that were worked out before ZPC was released. As patches for other programs are worked out, they will be published in future ZPC Update columns. In addition to the programs mentioned in the September REMark, patches for Microsoft Word and GW-BASIC (Z-150 version) are provided on the ZPC disk. These patches consist of batch files that apply the patches automatically.

For those who are technically capable and would like to try patching programs themselves to get them to run under ZPC, the documentation lists the areas where patching may be required. Sample code from some programs that were patched is included to to give you an idea of what to look for.

Not all programs that access video memory require patching. Compiled GW-BASIC programs will run without patching unless they use sound (music, noise, etc.). Compiled GW-BASIC programs access video memory directly only if they use graphics, so they will run in systems with less than 768k of memory if they do not use graphics. Another program that runs without being patched is the RUN/C C-language interpreter from Lifeboat Associates. This program does not write to video memory itself, and can be run in less than 768k of RAM. However, one of the sample programs included with it does write to video memory.

Copy Protected Programs

Copy protected programs usually will not work under ZPC, unless the copy protection is somehow defeated. Some of the major business programs for the IBM PC, including dBASE III and FRAMEWORK are copy protected, and for that reason if no other, will not work under ZPC. We have a utility on order as of this writing that we think will allow such programs (assuming no other problems arise) to run under ZPC. For those who do not want to wait for the next ZPC Update in REMark, and would like to experiment for themselves, the utility is called UNLOCK. It is available from TranSec Systems, Inc., 701 East Plantation Circle, Plantation, FL 33324 (305) 474-7548. Note: An IBM PC or Z-150 is required to make copies of a protected disk using a utility such as UNLOCK. You will find that most copy protected disks cannot be copied on a Z-100. Copies should be made only for personal archival purposes.

ZPC Patch

An error has been found in ZPC in a routine that reads pixels from the screen in the medium resolution graphics modes. So far, this error has not affected any of the programs that run under ZPC, but you should patch it because of a potential for trouble. The error involves a single byte at location 176A in ZPC1.COM, 191F in ZPC2.COM, and 1A04 in ZPC3.COM. This location contains a D2 (hex), which should be patched to D0. You can use DEBUG to make the patch, as in this example, which is for ZPC3.

```
A>DEBUG ZPC3.COM
-E1AØ4                 (use 176A for ZPC1, 191F for ZPC2)
1234:1AØ4    D2.DØ
-W
Writing 34ØØ bytes
-Q
A>
```

If you would like to patch the source code, the error is in the file PIXEL.ACM. Locate the label READDOT:, and a few lines below it, locate these lines:

```
    MOV       BX,OFFSET MASKTBL  ;POINT TO MASK TABLE
    XLAT                         ;GET MASK
    NOT       DL                 ;INVERT IT
```

Change the NOT DL instruction to NOT AL to correct the file. **Note:** This patch affects the first thousand or so copies of ZPC made. If your ZPC .COM files are dated 9-11-85 or later, the patch has already been made.

DEMO1 Patch

The DEMO1.EXE program provided with ZPC will not run under systems having less than 768k of RAM, because the program attempts to read from high memory even though it does not use it. You can patch DEMO1 with DEBUG as follows:

```
A>REN DEMO1.EXE DEMO1.BIN
A>DEBUG DEMO1.BIN
-E5696
1234:5696    BØ.Ø2
-W
Writing 12345 bytes
-Q
A>REN DEMO1.BIN DEMO1.EXE
```

ZPC Documentation Corrections

On page 7 of the ZPC documentation, there is an illustration of the Z-100 keyboard with the keys labeled as they are used when ZPC is in the PC mode. On the original of this illustration, some of the key cap legends are pasted on, and one came off when the first batch of copies were printed. If your illustration shows a blank key next to the key labeled PRT SC (the I CHR key on the actual keyboard), it should be labeled as follows:

```
/-------\
| PAGUP |
|       |
| PAGDN |
\-------/
```

On page 15 of the documentation, under BIOS RAM Locations, the table should show the printer addresses starting at 8 instead of 10. An additional entry should be added as shown below:

```
1Ø   Equipment flag.  This flag is set to default Z-15Ø
     values, and the bits in it are defined as follows:
     Ø          1 = diskette drives are present.
     1          1 = 8Ø87 installed.
     3,2        ØØ = 64k RAM chips, 11 = 256k chips.
     5,4        Initial video mode  Set to 1Ø for 8Øx25
                color.
     7,6        Number of diskette drives.
     8          Unused
     11,1Ø,9    Number of RS-232 cards (Ø1Ø = 2).
     12         1 = Game port attached
     13         Unused
     15,14      Number of parallel printer ports.
```

Appendix A of the documentation is in error in stating that compiled GW-BASIC programs will run in less than 768k of RAM if they do not use graphics. They will run, but they must be patched first. Using DEBUG, search for this string of numbers: 00,B8, 33,DB. When debug finds them, add 1 to the address reported, and change the value there (B8) to 02.

GW-BASIC Patch

When I wrote the patches for Z-150 GW-BASIC that are included with ZPC, I neglected to consider that the BEEP command, or PRINT CHR$(7), might use the interpreter's sound generating routines, which will not work under ZPC. Well, they do, and if you execute either BEEP or PRINT CHR$(7) in GW-BASIC under ZPC, it will hang up. You can fix GW-BASIC by patching it with DEBUG, as follows:

```
A>REN GWBASIC.EXE GWBASIC.BIN
A>DEBUG GWBASIC.BIN
-A7B9A
1234:7B9A  MOV AX,EØ7
1234:7B9D  INT 1Ø
1234:7B9F  NOP
1234:7BAØ  NOP
1234:7BA1                   (hit return)
-W
Writing 114CØ bytes
-Q
A>
```

This patch is for the release of GW-BASIC version 2 dated 7-15-85. For the other GW-BASIC version 2 for which a patch was provided with ZPC, use 7BE6 instead of 7B9A as the address to patch. For GW-BASIC version 1, use 40BE.

8 MHz 256K Update #2

In the July issue of REMark, we published an article on how to upgrade an older model H/Z-100 main circuit board to run at 8 MHz and use 256K RAM chips. We also published an update to that article in the September issue. In that issue, we recommended, if you get a "Default controller not ready" message upon powering up after making all of the modifications, that you remove the section of the modification under the heading "Wait State Modification". Since then, we have learned that 8-inch drives may not work properly with the computer running at 8 MHz with the wait state modification out.

If your 8-inch drives do not work, and you have done all of the modifications, except for the wait state modification, go ahead and install the wait state modification, and then replace U233 with a 74S74 (443-900) or a 74AS74 (443-1209).

If you have come across any problems with the modifications that were not covered in REMark, and have found solutions to them, let us know at HUG by mail or phone, so that we can pass your findings on to other HUG members.

Heath/Zenith Users' Group

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.

HDOS HARDCOPY SOFTWARE

Part Number	Description of Product	Selling Price	Vol. Issue
885-1008	Volume I Documentation	9.00	
885-1013	Volume II Documentation	12.00	
885-1015	Volume III Documentation	9.00	
885-1037	Volume IV Documentation	12.00	8
885-1058	Volume V Documentation	12.00	

MISCELLANEOUS HDOS COLLECTIONS

Part Number	Description of Product	Selling Price	Vol. Issue
885-1032	Disk V H8/H9	18.00	8
885-1044-[37]	Disk VI H8/H89	18.00	
885-1064-[37]	Disk IX H8/H89 Disk	18.00	
885-1066-[37]	Disk X H8/H89	18.00	10
885-1069	Disk XIII Misc H8/H89	18.00	

GAMES

HDOS

Part Number	Description of Product	Selling Price	Vol. Issue
885-1010	Adventure Disk H8/H89	10.00	4
885-1029-[37]	Disk II Games 1 H8/H89	18.00	8
885-1030-[37]	Disk III Games 2 H8/H89	18.00	8
885-1031	Disk IV MUSIC H8 Only	20.00	25
885-1067-[37]	Disk XI H8/H19/H89 Games	18.00	12
885-1068	Disk XII MBASIC Graphic Games	18.00	10
885-1088-[37]	Disk XVII MBASIC Graph. Games	20.00	14
885-1093-[37]	D&D H8/H89 Disk	20.00	16
885-1096-[37]	MBASIC Action Games H8/H89	20.00	18
885-1103	Sea Battle HDOS H19/H8/H89	20.00	20
885-1111-[37]	HDOS MBASIC Games H8/H89	20.00	23
885-1112-[37]	HDOS Graphic Games H8/H89	20.00	23
885-1113-[37]	HDOS Action Games H8/H89	20.00	23
885-1114	H8 Color Raiders & Goop	20.00	23
885-1124	HUGMAN & Movie Animation Pkg	20.00	41
885-1125	MAZEMADNESS	20.00	41
885-1130	Star Battle	20.00	47
885-1133-[37]	HDOS Games Collection I	20.00	59
885-8009-[37]	HDOS & CP/M Galactic Warrior	20.00	32
885-8022	HDOS SHAPES	16.00	45
885-8026	HDOS Space Drop	16.00	49
885-8032-[37]	HDOS Castle	20.00	59

CP/M

Part Number	Description of Product	Selling Price	Vol. Issue
885-1206-[37]	CP/M Games Disk	20.00	11
885-1209-[37]	CP/M MBASIC D&D	20.00	19
885-1211-[37]	CP/M Sea Battle	20.00	20
885-1220-[37]	CP/M Action Games	20.00	32
885-1222-[37]	CP/M Adventure	10.00	35
885-1227-[37]	CP/M Casino Games	20.00	38
885-1228-[37]	CP/M Fast Action Games	20.00	39
885-1236-[37]	CP/M Fun Disk I	20.00	55
885-1248-[37]	CP/M Fun Disk II	35.00	69

ZDOS

Part Number	Description of Product	Selling Price	Vol. Issue
885-3004-37	ZDOS ZBASIC Graphic Games	20.00	37
885-3009-37	ZDOS ZBASIC D&D	20.00	50
885-3011-37	ZDOS ZBASIC Games Disk	20.00	52
885-3017-37	ZDOS Contest Games Disk	25.00	58

UTILITIES

HDOS

Part Number	Description of Product	Selling Price	Vol. Issue
885-1022-[37]	HUG Editor (ED) Disk H8/H89	20.00	20
885-1025	Runoff Disk H8/H89	35.00	
885-1060-[37]	Disk VII H8/H89	18.00	
885-1061	TMI Load H8 ONLY Disk	18.00	
885-1062-[37]	Disk VIII H8/H89 (2 Disks)	25.00	
885-1063	Floating Point Disk H8/H89	18.00	
885-1065	Fix Point Package H8/H89 Disk	18.00	10
885-1075	HDOS Support Package H8/H89	60.00	
885-1077	TXTCON/BASCON H8/H89	18.00	
885-1079-[37]	HDOS Page Editor	25.00	15
885-1080	EDITX H8/H19/H89 Disk	20.00	
885-1082	Programs for Printers H8/H89	20.00	
885-1083-[37]	Disk XVI Misc H8/H89	20.00	11
885-1089-[37]	Disk XVIII Misc H8/H89	20.00	20
885-1090-[37]	Disk XIX Utilities H8/H89	20.00	22
885-1092-[37]	Relocating Debug Tool H8/H89	30.00	14
885-1098	H8 Color Graphics ASM	20.00	19
885-1099	H8 Color Graphics Tiny PASCAL	20.00	19
885-1105	HDOS Device Drivers H8/H89	20.00	24
885-1116	HDOS Z80 Debugging Tool	20.00	27
885-1119-[37]	BHBASIC Support	20.00	29
885-1120-[37]	HDOS 'WHEW' Utilities	20.00	33
885-1121	HDOS Hard Sec Sup Pkg 2 Disks	30.00	37
885-1123	XMET Robot Cross Assembler	20.00	40
885-1126	HDOS Utilities by PS:	20.00	42
885-1127-[37]	HDOS Soft Sector Support Pkg	30.00	45
885-1128-[37]	HDOS DISKVIEW	16.00	46
885-1129-[37]	HDOS CVT Color Video Terminal	20.00	46
885-8001	SE (Screen Editor)	25.00	28
885-8003	BHTOMB	25.00	28
885-8004	UDUMP	35.00	28
885-8006	HDOS SUBMIT	20.00	31
885-8007	EZITRANS.	30.00	30
885-8015	HDOS TEXTSET Formatter	30.00	42
885-8017	HDOS Programmers Helper	16.00	42
885-8024	HDOS BHBASIC Utilities Disk	16.00	46

CP/M

Part Number	Description of Product	Selling Price	Vol. Issue
885-1210-[37]	CP/M ED (same as 885-1022)	20.00	20
885-1212-[37]	CP/M Utilities H8/H89	20.00	21
885-1213-[37]	CP/M Disk Utilities H8/H89	20.00	22
885-1217-[37]	HUG Disk Duplication Utilities	20.00	26
885-1223-[37]	HRUN HDOS Emulator 3 Disks	40.00	37
885-1225-[37]	CP/M Disk Dump & Edit Utility	30.00	40
885-1226-[37]	CP/M Utilities by PS:	20.00	40
885-1229-[37]	XMET Robot Cross Assembler	20.00	40
885-1230-[37]	CP/M Function Key Mapper	20.00	42
885-1231-[37]	Cross Ref Utilities for MBASIC	20.00	43
885-1232-[37]	CP/M Color Video Terminal	20.00	46
885-1235-37	CP/M COPYDOS	20.00	54
885-1237-[37]	CP/M Utilities	20.00	55
885-1245-37	CP/M-85 KEYMAP	20.00	63
885-1246[-37]	CP/M HUG File Manager & Utilities	20.00	64
885-1247-37	CP/M-85 HUG Bkgrd Print Spooler	20.00	67
885-5001-37	CP/M-86 KEYMAP	20.00	51
885-5002-37	CP/M-86 HUG Editor	20.00	52
885-5003-37	CP/M-86 Utilities by PS:	20.00	54
885-5008-37	CP/M 8080 To 8088 Trans. & HFM	20.00	64
885-5009-37	CP/M-86 HUG Bkgrd Print Spool	20.00	66
885-8018-[37]	CP/M Fast Eddy & Big Eddy	20.00	43
885-8019-[37]	DOCUMAT and DOCULIST	20.00	43
885-8025-37	CP/M-85/86 Fast Eddy	20.00	49

ZDOS

Part Number	Description of Product	Selling Price	Vol. Issue
885-3005-37	ZDOS Etchdump	20.00	39
885-3007-37	ZDOS CP/EMulator	20.00	47
885-3008-37	ZDOS Utilities	20.00	47
885-3010-37	ZDOS Keymap	20.00	51
885-3022-37	ZDOS/MSDOS Useful Programs I	30.00	63
885-3023-37	ZDOS/MSDOS EZPLOT	20.00	63
885-3026-37	MSDOS SMALL C Compiler	25.00	65
885-3030-37	ZDOS/MSDOS Z-100 PC Emulator	40.00	68
885-3031-37	ZDOS/MSDOS Graphics	20.00	69
885-8029-37	ZDOS Fast Eddy	20.00	53

H/Z100 ZDOS/MSDOS - H/Z150 PC MSDOS

Part Number	Description of Product	Selling Price	Vol. Issue
885-3012-37§§	ZDOS HUG Editor	20.00	52
885-3014-37§§	ZDOS/MSDOS Utilities II	20.00	54
885-3016-37§	ZDOS/MSDOS Adventure	10.00	57
885-3020-37§	MSDOS HUG Menu System	20.00	62
885-3021-37§§	ZDOS/MSDOS Cardcat	20.00	63
885-3024-37§	ZDOS/MSDOS 8080 To 8088 Trans	20.00	64
885-3025-37§§	ZDOS/MSDOS Misc. Utilities	20.00	64
885-3029-37§§	ZDOS/MSDOS HUG Bg. Print Spool	20.00	66

§ All program files run on both
§§ Program files run partially on both

PC/IBM COMPATIBLE

Part Number	Description of Product	Selling Price	Vol. Issue
885-6001-37	MSDOS Keymapper	20.00	59
885-6002-37	CP/EMulator II & ZEMulator	20.00	59
885-6003-37	MSDOS EZPLOT	20.00	65
885-6004-37	MSDOS CheapCalc	20.00	67
885-6005-37	MSDOS Skyviews	20.00	67
885-6006-37	MSDOS Cardcat	20.00	69
885-8033-37	MSDOS Fast Edit	20.00	62

PROGRAMMING LANGUAGES

HDOS

Part Number	Description of Product	Selling Price	Vol. Issue
885-1038-[37]	Wise on Disk H8/H89	18.00	
885-1042-[37]	PILOT on Disk H8/H89	19.00	
885-1059	FOCAL-8 H8/H89 Disk	25.00	13
885-1078-[37]	HDOS Z80 Assembler	25.00	21
885-1085	PILOT Documentation	9.00	
885-1086-[37]	Tiny HDOS PASCAL H8/H89	20.00	13
885-1094	HDOS Fig-Forth H8/H89	40.00	18
885-1132-[37]	HDOS Tiny BASIC Compiler	25.00	59
885-1134	HDOS SMALL-C Compiler	30.00	63

CP/M

Part Number	Description of Product	Selling Price	Vol. Issue
885-1208-[37]	CP/M Fig-Forth H8/H89 2 Disks	40.00	18
885-1215-[37]	CP/M BASIC-E	20.00	26

BUSINESS, FINANCE AND EDUCATION

HDOS

Part Number	Description of Product	Selling Price	Vol. Issue
885-1047	Stocks H8/H89 Disk	18.00	
885-1048	Personal Account H8/H89 Disk	18.00	
885-1049	Income Tax Records H8/H89 Disk	18.00	
885-1055-[37]	MBASIC Inventory Disk H8/H89	30.00	
885-1056	MBASIC Mail List	30.00	
885-1070	Disk XIV Home Fin H8/H89	18.00	
885-1071-[37]	MBASIC SmBusPk H8/H19/H89	75.00	17
885-1091-[37]	Grade/Score Keeping H8/H89	30.00	14
885-1097-[37]	MBASIC Quiz Disk H8/H89	20.00	18
885-1118-[37]	MBASIC Payroll	60.00	30
885-1131-[37]	HDOS CheapCalc	20.00	47
885-8010	HDOS Checkoff	25.00	32
885-8021	HDOS Student's Statistics Pkg	20.00	44
885-8027	HDOS SciCalc	20.00	50

CP/M

Part Number	Description of Product	Selling Price	Vol. Issue
885-1218-[37]	CP/M MBASIC Payroll	60.00	31
885-1233-[37]	CP/M CheapCalc	20.00	47
885-1239-[37]	Spread Sht. Contest Disk I	20.00	
885-1240-[37]	Spread Sht. Contest Disk II	20.00	
885-1241-[37]	Spread Sht. Contest Disk III	20.00	
885-1242-[37]	Spread Sht. Contest Disk IV	20.00	
885-1243-[37]	Spread Sht. Contest Disk V	20.00	
885-1244-[37]	Spread Sht. Contest Disk VI	20.00	
885-8011-[37]	CP/M Checkoff	25.00	32

ZDOS

Part Number	Description of Product	Selling Price	Vol. Issue
885-3006-37	ZDOS CheapCalc	20.00	47
885-3013-37	ZDOS Checkbook Manager	20.00	54
885-3018-37	ZDOS Contest Spreadsheet Disk	25.00	58
885-8028-37	ZDOS SciCalc	20.00	50
885-8030-37	ZDOS Mathflash	20.00	55

DATA BASE MANAGEMENT SYSTEMS

HDOS

Part Number	Description of Product	Selling Price	Vol. Issue
885-1107-[37]	HDOS Data Base System H8/H89	30.00	23
885-1108-[37]	HDOS MBASIC Data Base Sys.	30.00	23

Continued on Page 83 ☛

HUG NEW PRODUCTS

HUG P/N 885-3032-37 MSDOS
Halley's Comet Locater $20.00

Introduction: This disk contains the program for locating Halley's Comet that was presented in the September 1985 Issue of REMark. Versions for both H/Z-100 and H/Z-150/160 series computers are provided, and the programs have been compiled for faster operation.

Requirements: The Halley's Comet Locater requires an H/Z-100, H/Z-150/160, or IBM PC compatible computer, 128k of RAM, and the Z-DOS or MS-DOS operating system. The H/Z-100 version will run in color on a computer equipped for it. The HUG Skyviews program (885-3015-37 for H/Z-100 or 885-6005-37 for H/Z-150/160) is recommended as a companion program to aid in locating the comet.

This disk contains the following files:

README .DOC	HALLEYZ .BAS
HALLEYZ .EXE	HALLEYPC .BAS
HALLEYPC .EXE	

Printed documentation is enclosed with the program.

Program Authors: Jim Tursa. Modified for H/Z-150 by P. Swayne.

HALLEYZ.EXE, HALLEYPC.EXE — This is the compiled Halley's Comet Locater, ready to run. HALLEYZ is for H/Z-100 computers, and HALLEYPC is for H/Z-150/160 computers. This program can plot the course of Halley's Comet in two ways: A right ascension-declination chart for locating the comet, and a space view that lets you see how the comet looks in space as it approaches and leaves the sun. The space view can be plotted from any angle at a user selectable distance from the sun. All of the planets in the solar system can be plotted in addition to Halley's Comet.

HALLEYZ.BAS, HALLEYPC.BAS — These files are the source for the above programs in BASIC. They will run in ZBASIC and/or GW-BASIC on the appropriate computer, but much more slowly than the compiled versions.

TABLE C Rating: (10)

HUG P/N 885-6007-37 MS-DOS
DND (Dungeons and Dragons) $20.00

Introduction: DND is the HUG version of the popular game "Dungeons and Dragons" (tm), played in real-time. This version displays graphic representation on the screen of the rooms, halls, and doors in the area which the player is in. The object of DND is to find the lord master of the dungeon.

Requirements: DND requires the MS-DOS operating system and the GW-BASIC interpreter version 2.0 or greater on the H/Z-100 PC series computer (H/Z-150/160 etc.).

The following programs and files are included on the HUG P/N 885-6007-37 MSDOS DND disk:

DND	.BAS	DND	.DAT
DND1	.BAS	DND	.DOC
DND2	.BAS	README	.DOC
LAIR	.BAS		

Program Authors: Original by Robert E. Wild, this version by Robert J. Sciamanda.

Program Content: The search for the lord master of the dungeon is made by exploring the 50 level dungeon, with the search beginning on level 1. DND is played in real-time, which means the program waits only a short time for a reponse and will continue playing without a response.

At the starting point there is a roadside tavern with many other taverns located on level one. It is at these taverns that treasures are cashed in for experience points. An accumulation of experience points will allow the player to become a higher level character, which will increase the chance of survival as the user moves deeper into the dungeon.

During the quest for the lord master, many obstacles will be encountered. The obstacles will include monsters which may try to steal any treasures which the player may have. Sometimes the monster may attack first. The player may fight, cast a spell, or evade. The player must watch his 'Hit Points' and 'Spell Units' to determine which option is best. These units can be refreshed by returning to a tavern or worshipping at an altar.

The user will find objects along the way which may be of help with the search. As easily as the objects appear, they may disappear.

The lord of the dungeon will be found in a HEATHKIT VAULT. As the game progresses, the combination to the safe will be given. The lord master may not be in that vault as there are many vaults throughout the dungeon.

The DND.DOC file contains information on some of the areas of the game. It is recommended that a hardcopy of the documentation be made for future reference. There are useful tables that will be nice to reference while playing the game.

There are many aspects of the game that must be learned while exploring the dungeon in DND.

Comments: DND is an excellent adventure game with the added feature of the graphic display with real-time mode.

TABLE C Rating: (1), (2), (5), (7), (10)

HUG P/N 885-8035-37 MS-DOS DOCUMAT Formatter And DOCULIST $20.00

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

Requirements: These programs require the ZDOS or MSDOS Operating System on an H/Z-100 computer system (not PC). A printer is required for hardcopy printouts of the documents created with DOCUMAT.

The following programs are included on the HUG P/N 885-8035-37 DOCUMAT Formatter And DOCULIST Disk:

DOCUMAT	.COM	TESTFILE	
DOCULIST	.COM	README	.DOC
MACLIB	.DOC		

Program Authors: Neal A. and Susan VanEck

Program Content: This program is a word processing system which will produce neatly formatted documents. DOCUMAT takes a file of text which you have already entered using your editor and including DOCUMAT commands, and formats it for final printout according to your specifications. Great flexibility is allowed for page layout and text control. Simple commands you insert into your text file specify such things as margin control, paragraph indentation or page size, and tell when to indent, tab, or underline. Sentences are automatically capitalized and separated by two spaces, and new pages are started automatically or whenever you wish.

The program allows for setting right justification of text. It provides a set of ten counters (numeric and alphabetic) for use in numbering pages, sections, chapters, prefixes, etc. It allows up to nine (9) footnotes per page, automatically numbering them as they appear. It automatically generates a Table of Contents of the text. It has the facility to INCLUDE (or merge) a separate file anywhere in the main text. It provides for interactive sessions while processing. The most useful tool is the powerful macro facility.

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

GLOBAL Commands:

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

CONTROL Commands:

A	– all lines of text following will be copied without change
AR	– align right (move the text to the right margin)
B	– begin bold face text

TABLE C

Product Rating

10 - Very Good
9 - Good
8 - Average

Rating values 8-10 are based on the ease of use, the programming technique used, and the efficiency of the product.

7 - Has hardware limitations (memory, disk storage, etc.)
6 - Requires special programming technique
5 - Requires additional or special hardware
4 - Requires a printer
3 - Uses the Special Function Keys (f1,f2,f3,etc.)
2 - Program runs in *Real Time**
1 - Single-keystroke input
0 - Uses the H19 (H/Z89) escape codes (graphics, reverse video)

Real Time — a program that does not require interactivity with the user. This term usually refers to games that continue to execute with or without the input of the player, e.g. p/n 885-1103 or 885-1211[-37] SEA BATTLE.

ORDERING INFORMATION

For Visa and MasterCard phone orders; telephone Heath Company Parts Department at (616) 982-3571. Have the part number(s), descriptions, and quantity ready for quick processing. By mail; send order, plus 10% postage and handling ($1.00 minimum charge, up to a maximum of $5.00. UPS is $1.75 minimum -- no maximum on UPS. UPS Blue Label is $4.00 minimum.), to Heath Company Parts Department, Hilltop Road, St. Joseph, MI 49085. Visa and MasterCard require minimum $10.00 order.

Any questions or problems regarding HUG software or REMark magazine should be directed to HUG at (616) 982-3463. REMEMBER-Heath Company Parts Department is NOT capable of answering questions regarding software or REMark.

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.

C	- center current line
Dn	- set dot leaders (for....example)
Hn	- indent each successive line after current line
In	- indent n spaces each line including current line
Jn	- jump n lines
Ln	- start a new line leaving n-1 blank lines
N	- start a new page immediately
P	- start a new paragraph
R	- reset, start new line and turn off temporary options
S	- justify current line and start new line
Tn	- set tab, space to column n
U	- begin underlining
Wn	- skip to new page

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

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

Fashioned after the CP/M version of DOCUMAT (HUG P/N 885-8019), this newer version (ver. 2.0) has several new features not found in the older one. Some of these features include:

1. The ability to create and modify and index.
2. Additional information on how to use the printer control macros to adapt to virtually any printer, including the new laser printers.
3. The addition of a PAUSE function for use in manually changing fonts and print wheels as needed when printing a document with DOCULIST.
4. The addition of PNUM, a page number function for increased flexibility in page number placement and formatting.
5. The addition of left-hand and right-hand running footers.
6. Expansion of the running headers to include both left and right-hand running headers.
7. The addition of a "revision bar" facility.
8. DOCULIST has the added features of displaying boldface as reverse video when used with an H/Z-100 or H/Z-19 terminal, and the ability to use ANSI carriage control characters.

Details of all commands can be found in the printed manual which is included with this software.

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

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

TABLE C Rating: (0), (4), (10)

HUG P/N 885-8036-37 CP/M
Grade $20.00

Introduction: Grade allows the teacher to enter grades into his or her gradebook in random order just as he used to do in his old-fashioned pencil and paper gradebook. It provides a hard copy suitable for permanent records, as well as scores and summaries suitable for display to the class. The program is fully menu driven and is fairly self explanatory. The maximum number of entries is 100 students and 60 scores for each student. Students are identified by a consecutively numbered "STUDENT INDEX" and all score entries are identified by a consecutively numbered "SCORE INDEX".

Requirements: This version of Grade requires MBASIC on an H/Z-89/90 series computer. A total of 64k of memory is needed along with one disk drive. This software will also run on an H/Z-100 system with either CP/M-85 or CP/M-86 and MBASIC.

In addition to the printed users' manual that comes with this product, the following programs or files are included on the HUG P/N 885-8036-37 CP/M Grade Disk:

```
GRADE    .BAS
ELEC103
README   .DOC
```

Program Author: Robert R. Ludeman

Program Content: When Grade is executed, the following commands or menu is available:

A	Add A Student To Present Class File
B	Build A New Class File
C	Change A Student's Score In Present Class File
D	Delete A Student From Present Class File
E	Enter A New Score Set Into Present Class File
F	Fixed Order Score Entry Into The Present Class File
H	Help - Prints This List
K	Kill (Deletes) An Entire Class File From A Named Disk
L	List Student Scores
O	Order Score Sets By Date In The Present Class File
P	Print Scores And Score Summary On Printer
R	Recall A File Previously Stored On A Named Drive
S	Save Present Class File On A Named Drive
X	Exit This Program
Z	Zap (Deletes) A Score Set From The Present Class File
/	Terminates Any Activity And Returns To Command Prompt

ELCT103, a sample class data file is also included with this disk.

TABLE C Rating: (10)

HUG P/N 885-8037-37 MS-DOS
Grade $20.00

Introduction: Grade allows the teacher to enter grades into his or her gradebook in random order just as he used to do in his old-fashioned pencil and paper gradebook. It provides a hard copy suitable for permanent records, as well as scores and summaries suitable for display to the class. The program is fully menu driven and is fairly self explanatory. The maximum number of entries is 100 students and 60 scores for each student. Students are identified by a consecutively numbered "STUDENT INDEX" and all score entries are identified by a consecutively numbered "SCORE INDEX".

Requirements: This version of GRADE requires the MS-DOS operating system and GW-BASIC version 1.0 or greater on an H/Z-100 PC series computer (H/Z-150/160 etc.). A total of 320k of memory is needed along with one disk drive.

In addition to the printed users' manual that comes with this product, the following programs or files are included on the HUG P/N 885-8037-37 MS-DOS Grade Disk:

```
GRADE    .BAS
ELEC103
README   .DOC
```

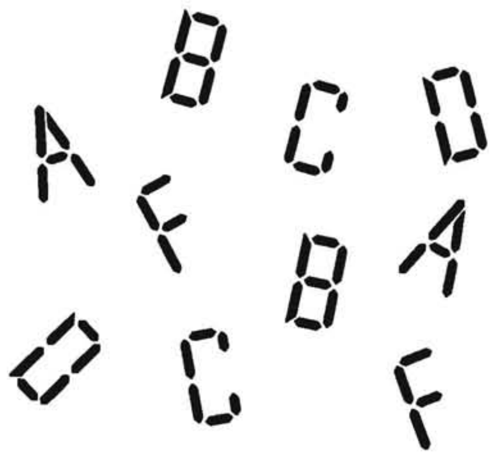
Program Author: Robert R. Ludeman

Program Content: When Grade is executed, the following commands or menu is available:

A	Add A Student To Present Class File
B	Build A New Class File
C	Change A Student's Score In Present Class File
D	Delete A Student From Present Class File
E	Enter A New Score Set Into Present Class File
F	Fixed Order Score Entry Into The Present Class File
H	Help - Prints This List
K	Kill (Deletes) An Entire Class File From A Named Disk
L	List Student Scores
O	Order Score Sets By Date In The Present Class File
P	Print Scores And Score Summary On Printer
R	Recall A File Previously Stored On A Named Drive
S	Save Present Class File On A Named Drive
X	Exit This Program
Z	Zap (Deletes) A Score Set From The Present Class File
/	Terminates Any Activity And Returns To Command Prompt

ELCT103, a sample class data file is also included with this disk.

TABLE C Rating: (10) ✱



Four Important Announcements From the Manufacturer of The Gemini Emulator Board

1

Gemini Emulator Boards Are Now Being Shipped.

Thank you for your patience. We are now in production and shipping boards in quantity.

2

Back-Logged Orders Are Being Met As Quickly As Possible.

When the Gemini Emulator Board went into production, many thousands of orders were waiting to be filled. We anticipate that we will be completely caught up with back orders by mid-November 1985.

3

Gemini Technology Inc. Has Been Formed To Handle Manufacturing And Marketing.

With the R&D on the Gemini Emulator Board completed, DEL Professional Systems Ltd. has established a sister company to handle manufacturing and marketing. The new company, Gemini Technology Inc., is a public company with shares trading on the Vancouver (British Columbia) Stock Exchange under the trading symbol, GMT.

4

The People Are The Same, But The Look Is A Little Different.

The people behind the Gemini Emulator Board are the same ones who designed and developed it, but there is a new look to the corporate symbol of Gemini Technology Inc. It appears on the bottom of the page.

If you have any questions about any of the information above, please contact us.

GEMINI technology inc.

11-11151 Horseshoe Way, Richmond, B.C., Canada V7A 4S5 (604) 274-1283

ZENITH, PANASONIC & STUDIO COMPUTERS *Together!*

Panasonic monitors and printers make a great companion to Zenith Data Systems computers. Panasonic RGB color and monochrome monitors offer excellent resolution, .31 mm dot pitch on most color models, tilt/swivel bases, and PC compatibility. Choose from CRT sizes from 10" to 45" projector screens and even a 14" television/monitor combination. And all are backed by a one year warranty.

Panasonic dot-matrix and letter quality printers offer exceptional features and quality at very economical prices. Choose from several models, all backed by a super two year warranty. Check our 1986 catalog for details on all Panasonic products.

For a limited time, you'll receive absolutely free a fine Panasonic stereo or television with selected systems as described below.

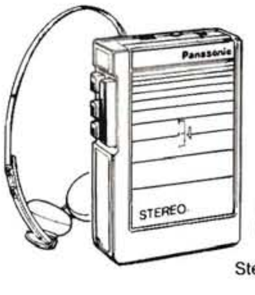
OFFER #1 Purchase any Panasonic monitor or printer, and select your free gift from group #1.

OFFER #2 Purchase any Panasonic monitor or printer along with any floppy disk Zenith computer, and select your free gift from group #2.

OFFER #3 Purchase any Panasonic monitor or printer along with any Zenith hard-disk drive computer, and select your free gift from group #3.

It's that simple! Call, write or stop in our store to place your order. This offer valid only for cash, check, or money order purchases.

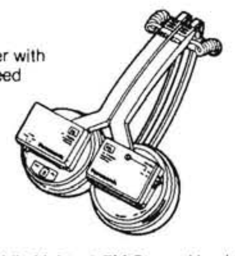
GROUP #1


Mini Stereo Cassette Player with Lightweight Stereo Headphones

Compact FM/AM/FM Stereo Radio with Lightweight Stereo Headphones

GROUP #2

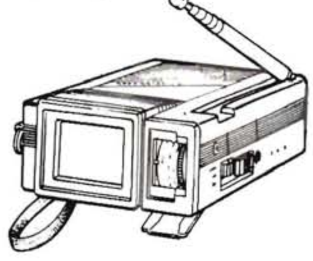
AC/Battery FM Stereo Microcassette Recorder with Built-in Speaker, 2-Speed Tape Selector and Lightweight Stereo Earphones


World's Lightest FM Stereo Headphone Radio with RHC™ Technology and Collapsible Design

GROUP #3

5" diagonal B & W Outsider TV, also an FM/AM radio with slide-rule tuning

3-Piece FM/AM/FM Stereo Radio Cassette Recorder with Separate Speaker System, Auto-Reverse and 5-Band Graphic Equalizer


Travelvision, 20% smaller than previous models. 4-way operation, 360° electronic tuning system. (1.5" diagonal)

Studio Computers inc.

999 South Adams
Birmingham, MI 48011
Phone (313)-645-5365

MasterCard VISA

Customizing MS-DOS WordStar Professional For The H-100 Computer

Frank T. Clark
402 West Ferry Street
Berrien Springs, MI 49103

Introduction

I have been using Wordstar with great satisfaction for several years now. I first started using a very old version under CP/M on my H-89 computer. I then updated and also went to using it under CP/M-85 on my H-100 computer. Later, I began using version 3.21 under Z-DOS, and even later got WordStar version 3.30 Professional for my H-150 computer. I recently replaced my H-100 WordStar with WordStar version 3.31 Professional. I found a lot of things I didn't like about the package, which I decided to change and that is the subject of this article.

I have enjoyed WordStar for many reasons. It was helpful to have this versatile and powerful word processor available for every computer I used. What a difference it made to have the same program (from the user viewpoint) to use on all my different computers. Possibly, unlike some WordStar users, I have appreciated WordStar's control key sequences much more than other programs that require the use of the function keys or even the different implementations of the function keys in WordStar. Part of the reason for this preference is that there are different numbers and names of function keys when you switch computers. The locations of the function keys change even on computers made by the same manufacturer! Keeping all this straight as I switch from one computer to another is just too difficult, so I stick to the control keys.

There is also another reason why I prefer to use the control key sequences. I learned how to type in high school so I am not your usual H&P (hunt and peck) typist. The use of the control key sequences allows me to type, edit and perform all sorts of control functions at my greatest speed without requiring me to look at the keyboard or remove my fingers from the home keys to reach for the function keys. Of course, the ultimate flexibility of WordStar is that whichever way you prefer, you can use it that way.

Customizing WordStar

One of the superior features of WordStar, of course, is the ability to customize it to your specific machine and purposes. This dates from the days before the whole computer market decided to play copycat with a single limited design. Some of us, though, still do not fit into the general mold and have special needs and desires for our word processor. If this describes you, then you will like this article.

Pat Swayne has provided us with some very good articles in the past on customizing earlier versions of WordStar. In this article, I will describe similar patches optimized for this particular version and go even further into some other patches which I have found useful. One of the things I have not found useful is the WINSTALL program. This is one of the slowest programs I have ever seen, and one of the most difficult ways I can imagine to make some simple changes to the program. I find that I prefer to make my changes with DEBUG, and I will provide a list of common patch locations at the end of the article to facilitate this possibility.

All information in this article is based on my observations and testing of the H-100 MS-DOS WordStar version 3.31 Professional software package model number MP-463-17. I had a lot of help from my old CP/M WordStar manuals, which contained some detailed information on patch areas that have not changed significantly in the MS-DOS versions. I cannot guarantee that Zenith or Micropro will not make some changes in this software package which will invalidate my information, so please exercise caution with any patches you make to your software. As a bit of a confidence builder, you should check that the file WSU.COM on your distribution disk is of the size 41856 bytes and created on February 14, 1985 at 11:19:11. It probably will not make any difference if the dates are different, but it would dictate extra caution. In any event, you obviously should not make any changes at all to your original distribution disks, and in fact, I would recommend that you do not change your working copies either. Make all your changes to an extra copy of the software and test it very carefully to make sure it works before you use it on a regular basis.

Comparison To IBM WordStar

One of the first things I did was to compare this version of WordStar to the IBM version WordStar. I was shocked to discover that the H-100 WordStar was 20480 bytes bigger than the IBM version! Except for the size difference the programs were extremely similar right down to the IBM 25th line function key headings, which are actually in the H-100 version! I was, of course, very curious as to what was different in the H-100 version to cause all this extra code.

You will be shocked (maybe) to discover there are more than 16k of X's at the end of the normal WordStar file followed by a small section of code for special initialization and console input handling. This section of code is the same code Pat described in his earlier article, only now it is a normal part of the WordStar code and not something that is relocated to another segment of memory. This has the advantage of using less RAM memory (though more disk memory) and solves the problem Pat mentioned of not being able to be implemented due to RAM memory limitations.

Regrettably, I am not fortunate enough to be able to afford a winchester and I frankly didn't feel like giving up over 16k (about 5%) of my limited floppy space for just X's. The extra space also takes longer to load especially on a floppy. Since I had no use for the extra functions of the additional code, (function keys) it was easy to get rid of the excess space and fit in quite well with my other plans for WordStar.

As Pat mentioned in his articles, one of the big problems with WordStar is its slow speed on the H-100. This is especially a problem under MS-DOS version 2. At one time, I would only use Z-DOS and avoided the use of MS-DOS version 2 solely because WordStar was so slow under it. The IBM WordStar is blindingly fast even under MS-DOS version 2 because of special patches it contains. Many of the following patches are just to speed up output to match the IBM version. The H-100 computer cannot be quite as fast as the IBM due to its superior graphics and software definable characters set, but the difference does not have to be much with the proper patches.

WordStar On A Remote Console

One of the other things I do that may not be common for other computer users is that I use WordStar quite a lot on a remote console. The original design of WordStar was for terminals, so it takes very few patches to the H-100 version for it to work on a terminal. The H-100 is very similar to the VT52 type terminal, which is compatible with the H-19, H-29 and a lot of other terminals. One of Pat's earlier patches was undesirable in this respect because it tied console output directly to the H-100 ROM. As long as all I/O is through the Z-DOS/MS-DOS BIOS, it is very easy to divert it using console SWAP, CONFIGUR or other similar programs to a remote terminal. The loss of speed by doing this is negligible.

I work with bulletin boards a lot and I use WordStar remotely on some of these bulletin boards. At 300 baud, WordStar is a real bear, but it is still usable. At 1200 baud, WordStar is really not too bad. As long as I call up the bulletin board with a VT52 type terminal, I can use WordStar the same as if I was right there, just a lot slower. Editing bulletin board display files has some special requirements for line width, the use of tabs and other things which also call for some special patches that I will describe later.

I also use a remote console at home. I have an H-19 terminal in the living room that is hooked up to my H-100 (and sometimes my H-150) upstairs in my office and running remotely at 19200 baud. I do a lot of work on the H-19 while I am watching television (like I am right now while typing this article) running WordStar, ZBASIC and other things.

WordStar Patches Introduction

Each of the following patches and their uses is individually described. In the patches, I have shown only the exact characters that would be typed in using DEBUG. Where code patches are easier to make I have shown alternate patches using MS-DOS version 2 DEBUG if you are using that. Refer to Pat's excellent articles in the May and February 1985 Remark issues and your operating systems manuals for more detailed information and explanations about DEBUG and patching.

Remember that you must not change your original distribution disks and I recommend that you do not change your working disks either. Make an extra copy of WordStar to perform these patches on and test them thoroughly before transferring them to your working disks.

All patches are performed using the following commands:

```
debug ws.com
(make your.patches)
w
q
```

WordStar Patches

These patches allow you to customize the startup display with your own special messages or notices. Do not make these messages any longer than they already are or you will overwrite other messages! These patches do not add or detract from the efficiency of WordStar, but they add the personal touch.

```
Custom display messages
e7c ' (customized) '
e18f ' VT52 standard terminal interface '
e1d7 '  All I/O H100 MS-DOS customized  '
e1fb '   personally by Frank T. Clark   '
```

I often use WordStar in interlaced mode which has 30 scrolling lines on the screen instead of 24 (the 25th line won't scroll so it cannot be used). The following patch in a copy of the WS.COM file called WSI.COM makes this easy and useful.

```
Set the number of scrolling lines on the screen
e248 1e
```

The following two patch sets are used to set the escape sequences that WordStar uses for normal text versus menus, blocks and other special displays. I use both of these at different times. I have one file called WS.COM which uses the normal VT52 escape sequences that I use remotely and another called WSC .COM which is the file I use at the H-100 screen with a color monitor. Using these patches and changing the appropriate values, you can have any four colors on the screen and not just the two that WINSTALL will let you have. My favorites are yellow characters on a blue background for the normal text and red characters on a cyan background for the menu text. Simple changes to the values given will allow you to select any colors you prefer.

VT52 Device Or Monochrome Video Memory

```
reverse video menu text
e284 2 1b 70
normal video text
e28b 2 1b 71
```

H-100 Color Screen

```
color menu
e284 4 1b 6d 32 35
color text
e28b 4 1b 6d 36 31
```

Another of the undesirable things standard WordStar does is to perform a total terminal reset on entry and exit. This causes problems for people who set their keyclick off, remote terminals operating at other than their normal baud rate, programs that set the 25th line, like Pat's KEYMAP and all kinds of other situations. Normally, the only initialization or cleanup necessary is a clear screen. I find that I prefer to use a block cursor while in WordStar and an underline cursor at other times, so I have included an optional patch for this also.

Minimum Initialization And Cleanup

```
clear screen on entry
e292 2 1b 45
clear screen on exit
e29b 2 1b 45
```

Preferred Initialization And Cleanup

```
clear screen on entry and set block cursor
e292 5 1b 45 1b 78 34
clear screen on exit and set underline cursor
e29b 5 1b 45 1b 79 34
```

Fast operation is always desirable and I strongly recommend all 0 values for WordStar's software delays which Pat mentioned in his previous article.

```
eliminate archaic and undesirable delays
e2ae Ø Ø
e2d2 Ø
```

It is always helpful to have WordStar look on the correct drive for its overlays and the main program when you reload after running another program. I have shown the usual values for floppy and winchester machines. These patches do not work in the standard version of WordStar Professional! The patches will not work until you perform the speedup patches indicated below.

Floppy System

```
default system drive A.
e2dc 1
```

Winchester System

```
default system drive E
e2dc 5
```

As mentioned earlier, fast operation is always a goal, so after a person is no longer a beginner I fully recommend a default help level of 0 to suppress all helpful menus and keep things moving right along. If I have a temporary lapse of memory the helpful menus can always be turned back on by typing ^JH3.

Novice

```
default help displays enable
e36Ø 3
```

Expert

```
default slow help displays disable
e36Ø Ø
```

Also for speed purposes, I usually prefer to have the default directory display off so that I don't have to wait for WordStar to read the disk when I happen to pause a second before typing the next command. If I really want to see the file names one key press will bring it back very quickly.

```
default slow directory display off
e363 Ø
```

I have a specially patched version of WordStar for bulletin board use where some things are different from normal. I find it to be easier to have these things default rather than remembering to type in the necessary control sequences to set them every time I enter WordStar. I prefer things like using most of the 80 column display on a bulletin board and not just 60 columns. I also like to have the justification off so that extra spaces do not get put in to slow things down. You know how I hate things that slow you down. For the same reason, I turn the vari-tabs off so that real tabs will not get replaced with multiple spaces which again take a lot longer to output over a modem. I also turn the page break off since bulletin board files do not have pages. You may ask "Why bother to use document mode at all just use non-document mode?" since I disable all the features of document mode. I used to use non-document mode in earlier versions of WordStar, but someone at Micropro in their infinite wisdom decided that paragraph reform should not work in non-document mode. That is the one feature of document mode that I really do like and use even on a bulletin board.

```
document mode default right margin
   (maximum Wordstar allows)
e38Ø 4b
On screen justification off
e386 Ø
vari-tabs off
e387 Ø
page break off
e38d Ø
```

I also hate the hyphen help feature (you guessed it) because it slows down the time it takes to reformat paragraphs. I also find excess hyphens in a file ugly.

```
disable hyphen help
e389 Ø
```

I usually remove the ruler line from the top of the screen because it gives me an extra display line, and also speeds up screen changes (very noticeable at 300 baud).

```
ruler line off
e38b Ø
```

I use batch files a lot (faster than typing the commands over and over) and sometimes I want to call WordStar in non-document mode with a file name on the command line to edit things like program source files. I keep a special version of WordStar called WSN.COM which does just that using the following patch.

```
non-document mode startup
e392 ff
```

One of the things good about WordStar overlays (the bad is that they make it run slower) is that I can have several custom versions of the root WS.COM which is not too big at about 21k, and they all use the same overlays which are 143k counting MailMerge and CorrectStar. The one problem is that if I run an MS-DOS program from inside one of these specially named versions of WordStar it will reload WS.COM unless I use the following patch to change the internal name of the program for reloading.

```
internal program name change
e3e7 'WSC      ' or 'WSN      ' or 'WSI      ' or whatever
```

The following patches must all be done together for any one of them to work. The purpose of each is defined separately. The only drawback to these patches is the use of WordStar function keys are no longer supported (which breaks my heart). If you really like function keys, then I would recommend Pat Swayne's KEYMAP program.

```
reduce the size of the program
rcx
```

```
5380
e2a4 c3
e2a7 c3
e35c 28 99
```

Z-DOS DEBUG Version Of Patches

```
use printer busy status from BIOS directly
e7ca ff
e7cc b8 0 2 9a 4b 0 40 0 80 e4 80 75 1 f9 c3
bypass OS printer output to BIOS directly
e7dd 9a c 0 40 0 c3
bypass OS console status from BIOS directly
e23d7 9a 3 0 40 0 b0 ff 75 2 b0 0 c3
bypass OS console input from BIOS directly
e240d 9c fc 9a 6 0 40 0 9d c3
bypass OS console output to BIOS directly
e247f 9c fc 88 c8 9a 9 0 40 0 9d c3
```

MS-DOS Version 2 DEBUG Version Of Patches

```
use printer busy status from BIOS directly
e7ca ff
a7cc
mov ax,200
call 40:4b
and ah,80
jnz 7da
stc
ret
bypass OS printer output to BIOS directly
a7dd
call 40:c
ret
bypass OS console status from BIOS directly
a23d7
call 40:3
mov al,ff
jz 23e2
mov al,0
ret
bypass OS console input from BIOS directly
a240d
pushf
cld
call 40:6
popf
ret
bypass OS console output to BIOS directly
a247f
pushf
cld
mov  al,cl
call 40:9
popf
ret
```

You may also occasionally find it useful to customize the default tab settings to those that you prefer. WINSTALL doesn't let you do that! The table begins with a byte indicating how big it is, then each nibble (hex character) following is either a 1 indicating a tab column or a 0 if it is not.

```
default tabs every 8 columns
e43b9 4b 00 00 00 01 00 00 00 01 00 00 00 01  .
```

Conclusion

I love WordStar because it is so easy to customize for different uses and comes in 8 bit CP/M versions and 16 bit MS-DOS versions (my apologies, but I have no use for CP/M-86) for use on all my computers. With the above patches implemented I do not suffer the indignity of having my superior H-100 run a lot slower than my H-150 copycat computer, and I get to make good use of my color monitor.

Postscript

After finally getting WordStar to work well, I started using CorrectStar, and as you might have guessed discovered some problems with its use. The system drive option used by the rest of the overlays is not used for CORRSTAR.OVR, but it can be patched in.

```
system drive A: for corrstar.ovr
debug wsovly1.ovr
e2a13 1
w
q
```

If you wish to contact the author please write to the above address (do not telephone) and include a self-addressed, stamped-envelope, if you expect a reply. *

About The Author

Frank Clark is the senior consultant on systems software for Zenith Data Systems in Saint Joseph, Michigan. His background includes both mainframe computers and microcomputers, programming in 6502 assembler, 8080 assembler, 8088 assembler, APL, BASIC, C, COBOL, FORTRAN and even some PASCAL. He graduated from Andrews University, Michigan in August 1979 with a B.S.C.S degree and in August 1982 with an M.S.C.S. degree. His interests center on anything even remotely related to computer software, particularly graphics. He also enjoys rock music, as well as reading, mostly computer books and science fiction.

H-100 WordStar Version 3.31 Professional Patches Quick Reference

```
Custom display messages
e7c ' (customized) '
e18f ' VT52 standard terminal interface '
e1d7 '  All I/O H100 MS-DOS customized  '
e1fb '   personally by Frank T. Clark   '
Set the number of scrolling lines on the screen
e248 1e
reverse video menu text
e284 2 1b 70
normal video text
e28b 2 1b 71
color menu
e284 4 1b 6d 32 35
color text
e28b 4 1b 6d 36 31
clear screen on entry
e292 2 1b 45
clear screen on exit
e29b 2 1b 45
clear screen on entry and set block cursor
e292 5 1b 45 1b 78 34
clear screen on exit and set underline cursor
e29b 5 1b 45 1b 79 34
eliminate archaic and undesirable delays
e2ae 0 0
e2d2 0
default system drive A:
e2dc 1
default system drive C:
e2dc 3
default help displays enable
e360 3
default slow help displays disable
e360 0
default slow directory display off
e363 0
document mode default right margin (maximum Wordstar allows)
e380 4b
On screen justification off
e386 0
```

```
vari-tabs off
e387 0
page break off
e38d 0
disable hyphen help
e389 0
ruler line off
e38b 0
non-document mode startup
e392 ff
internal program name change
e3e7 'WSC      ' or 'WSN      ' or 'WSI      ' or whatever

reduce the size of the program
rcx
5380
e2a4 c3
e2a7 c3
e35c 28 99
use printer busy status from BIOS directly
e7ca ff
e7cc b8 0 2 9a 4b 0 40 0 80 e4 80 75 1 f9 c3
bypass OS printer output to BIOS directly
e7dd 9a c 0 40 0 c3
bypass OS console status from BIOS directly
e23d7 9a 3 0 40 0 b0 ff 75 2 b0 0 c3
bypass OS console input from BIOS directly
e240d 9c fc 9a 6 0 40 0 9d c3
bypass OS console output to BIOS directly
e247f 9c fc 88 c8 9a 9 0 40 0 9d c3
use printer busy status from BIOS directly
e7ca ff
a7cc
mov ax,200
call 40:4b
and ah,80
jnz 7da
stc
ret
bypass OS printer output to BIOS directly
a7dd
call 40:c
ret
bypass OS console status from BIOS directly
a23d7
call 40:3
mov al,ff
jz 23e2
mov al,0
ret
bypass OS console input from BIOS directly
a240d
pushf
cld
call 40:6
popf
ret
bypass OS console output to BIOS directly
a247f
pushf
cld
mov  al,cl
call 40:9
popf
ret
default tabs every 8 columns
e43b9 4b 00 00 00 01 00 00 00 01 00 00 00 01
```

Heath/Zenith Users' Group

MAP PROJECTIONS OF THE WORLD

- -Produce high-res color graphics maps for the H/Z-100 and H/Z-150 (or IBM)
- -Track up to 10 hurricanes in animation Two historical ones are included
- -Choose any of 7 different projections- 5 Azimuthal & Polyconic & Mercator
- -Pick either of 2 data files: World Coastlines with 5070 digitized points or a very-high-res file of the Caribbean from the Carolinas to Mexico to the Amazon (11940 points)
- -Vary the origin, rotation, window and scale(Global or down to 20 miles/inch on most projections)
- -Choose any grid spacing for the parallels and meridians
- -User's manual includes screen dumps of 16 sample maps to help you learn how to reproduce these and many others
- -Superfast execution- over 1000 pts/min from an optimized compiled basic program that requires only DOS and color graphics. A color monitor does help

"...fascinating look at the world from every conceivable angle...The maps are excellently drawn. It's amazing that Dr Pittman could squeeze that much detail onto a single 5 1/4-inch diskette..."--Dave Felstul--BUSS # 91

Pick up a copy at many Heathkit Centers or order direct by sending $49.95 post-paid (specify Z-100 or Z-150)to: Dr M E Pittman, 57 Emile Av, Kenner, LA 70065

WORDKEY™ TAMES THE POWERFUL WORDSTAR

WordKey makes the popular WordStar™ program the easy-to-use professional word processor it should be. More than 90 simple keypad and function-key commands help you use and remember ALL of WordStar's many capabilities, without EVER using WordStar's awkward 'CTRL' key. Your secretary will love it, and so will you and your family.

Full onscreen help is always instantly available. And you'll like its many other features. A letter to H-Scoop from a satisfied user says, "I found [WordKey] made WordStar much easier to use and ...I learned some capabilities that I did not know WS had...I am extremely pleased with it."

Use WordKey with either WordStar version 3.21 or 3.30, Z-DOS 1.25 or MS-DOS 2.13, on your H/Z-100 computer To order, just mail a check to DelSoft for $49.95. (Calif residents please add 6% tax.)

PSC-PROWRITER GRAPHICS SCREEN DUMP UTILITY

PSC-PRO prints an exact copy of your H/Z-100 monochrome (B&W or green) screen on the C.Itoh Prowriter printer from within ANY program simply by pressing Shift-F12. $19.95 (Calif. residents please add 6% tax.)

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

Performance.

Introducing EasyPC™ from UCI. The most advanced IBM PC emulator you can buy for your Z-100. First to provide the compatibility, speed and audio support to make your Z-100 run like a PC. Since the EasyPC provides complete PC circuitry, you actually run PC programs on PC hardware. Quite simply, it's like installing a PC right inside your Z-100. In the PC mode, your Z-100 can run virtually the entire library of PC programs, including LOTUS 1-2-3, dBase III, FLIGHT SIMULATOR, and most other copy-protected software. In the Zenith mode, your Z-100 is a Z-100, with no change in performance.

EasyPC is fast. In fact, it is noticeably faster than both the Z-150 and the IBM PC. And EasyPC can run at either 5 or 8 MHz. It supports both floppy and Winchester drives, provides crisp color video graphics, supports PC-compatible sound generation, and even comes with its own self mounting speaker.

Quality.

You already know and appreciate the kind of quality and reliability that is built into your Z-100. UCI matches that commitment to quality from first step to last.

Sophisticated engineering and quality components combined with precision automatic assembly, component burn-in, and both in-circuit and full functional testing, mean superior performance and long term reliability.

EasyPC's advanced circuitry utilizes two, high quality, S-100 multilayer boards and a piggyback board. Yet its ingenious design requires only *one extra slot.* So you get all the advantages of PC compatibility, with room to spare.

Availability.

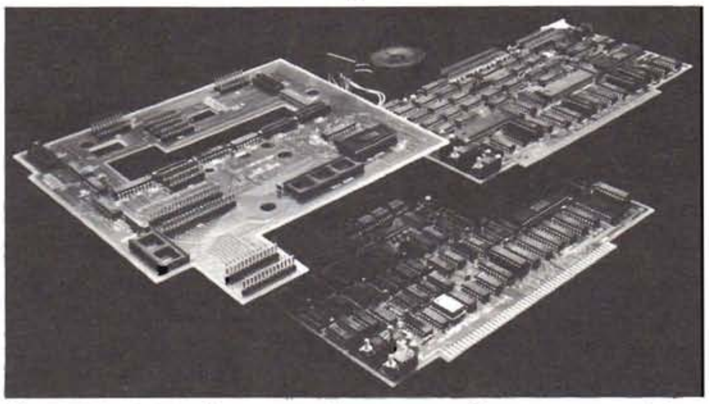
EasyPC is ready when you are. If you've been waiting too long for PC compatibility, your wait is over. EasyPC is here now. Complete documentation is included for quick and easy installation, so you'll be up and running fast. Your only problem will be which PC programs to get first.

EasyPC is available through your independent Zenith dealer. Ask him for a demonstration.

Price.

At $699, you'll pay a little more. But you'll get the best. Total compatibility. High speed performance. Unsurpassed quality and serviceability. In a word, value. You also get UCI's one year Solid Service Guarantee and full customer support.
If you have any questions or problems with any UCI product, or need the name of your nearest dealer, just call 800-UCI-COMPUTER.

Easy PC™



Full IBM PC emulation for your Z-100.

UCI CORPORATION
948 Cherry Street, Kent, Ohio 44240
(216) 673-5155 / 800-UCI-COMPUTER

EasyPC is a trademark of UCI, Inc. Zenith and Z-100 are trademarks of Zenith Electronics Corp. IBM and IBM PC are trademarks of International Business Machines. Lotus 1-2-3 is a trademark of Lotus Development Corp. dBase III is a trademark of Ashton-Tate. Flight Simulator is a trademark of Microsoft Corp.

Image Analysis With The Z100

Lynn R. Jarvis
Pathology Department
Flinders Medical Centre
Bedford Park, S.A. 5042
AUSTRALIA

The Zenith Z100 has impressive graphics which add dramatic visual impact to any software, from spreadsheets to word processors or from painting editors to space games. You would think that all that video memory must be useful for some other purpose. But what? Well, here is an application which really squeezes every drop out of the Z100 video capacity, and it shows that the Z100 is the right machine for the job if you can find the right job for the machine.

Introduction

An image analysis is used to extract information from an image and make measurements on the image features obtained. It can be used to analyze images from optical instruments such as telescopes and microscopes, from photographs of any sort and from real life objects. There are numerous applications for these machines in many fields such as metallurgy, materials science, medicine, electronic component inspection, mineralogy, fibre analysis, astronomy, remote sensing, atomic energy, and many applications too numerous to mention briefly. Basically, all involve image acquisition, feature extraction and measurement.

In medicine, there is a growing area of research involving automatic extraction of diagnostic information from microscope images of cell preparations by the use of image analyzing computers. The current diagnostic procedures require a highly trained technician to examine the microscope slide and report on evidence of malignancy. The aim of computer image analysis is to give the task of slide screening to the machine and to calculate probabilities and give an objective diagnostic report. At Flinders Medical Centre in Adelaide, South Australia we are interested in applying computer image analysis for diagnosis in a wide range of diseases, but our problem has been for many years that image analyzers are very expensive (typically around $80,000). Furthermore, the machines are not easily adapted to specific problems, and it is common to see research efforts diverted to molding the diagnostic method around a general purpose machine. The ideal solution is to have the expertise, manpower and funding to develop equipment and software especially tailored for the research problem. There are several centres where such development is being actively pursued, but in Australia funding is scarce and there has to be a cheaper way, both in equipment and manpower.

There have been several noteworthy attempts to construct image analyzers based on microcomputers, such as the Apple. But the need to hold multiple images in memory and for reasonably fast processing time have limited their practical success. My approach has been to spend enough, but not too much, on a computer with better performance, higher memory capacity, and above all, a good image display without resorting to the extra expense of specialist boards. The aim was to produce an image analyzing machine, which would be easy to use and affordable by smaller hospitals outside the main research and teaching centres. The main benefit of the effort in development would be in quick and easy modification of the software for special applications as the need arose. The machine could then be adapted to the problem, not the other way around.

I selected the Zenith Z100 for two main reasons. Firstly, the outstanding feature of the Z100 for image analysis purposes is the massive video memory of 2 pages of 3 planes each at full screen resolution. To my knowledge such capacity is unequaled by any contemporary microcomputer. The other advantage, from a developer's point of view, is the S100 bus which allows a good selection of video frame grabbers to be installed. Other options, such as interface boards and stepper motor controllers for the planned requirement to drive an automated microscope stage are also readily available. Another benefit of Zenith computers is the extent of the documentation, which gives all the information necessary to develop software which requires direct access to the video memory.

At the time of equipment selection, I was well aware that software development takes time, and the problem with microcomputer products is that there is a real danger that once the software is complete the hardware will be out of date. I already had a Zenith Z89 and was impressed by how that machine had been supported for so long, so I hoped that the Z100 would be supported equally well. Now that our development on the Z100 is complete, it is heartening to find that the continuing support of the Zenith/Heath company means that the computer is not now out of date and will no doubt be supported for many years to come. Also, the emphasis by Zenith to maintain functional compatiblity throughout their design range means that the software development effort will not be wasted, but can be built upon as newer machines become available.

System Development

The description below is a reasonably detailed introduction to the development of the system, which I hope may be of interest to members. The system uses an unmodified Z100 computer with the addition of a low cost video digitizer board, monochrome video camera and monitor, and digitizer tablet for program control. The software was designed and developed by L. Jarvis in the Department of Pathology, Flinders Medical Centre, Adelaide. The ZBASIC software code was written by P. Stoll, assembler code by L. Jarvis.

Video Digitizer Board

The Z100 machine was principally a development tool and in order to justify the expense of purchase, it was necessary to achieve at least observable success for image acquisition and display. This meant that from the outset, the video digitizing board had to be easily installed and a success beyond doubt. How to ensure such success before purchasing the equipment is a mystery, but many potential problems were avoided by making the equipment selection conservatively.

The video digitizer board is the primary determinant of the performance of the whole system, and the choice of the slow random-access board may be open to criticism. However, there were many aspects which led to the board selection and it is of interest to outline the rationale here.

For a reasonable cost, the choice at the time was between the TECMAR, SCION and MICRO-WORKS boards. The TECMAR equipment was a set of 3 and required in addition 64Kb of static RAM for image storage. This was one too many boards for a hard disc Z100 and there were unpredictable compatibility problems for wait-states, clock timing, addressing requirements, and the possible need for a bus extender. The SCION boards seemed to need SCION display hardware — more possible problems of incompatibility, and one of the aims was to get away from expensive peripheral boards. So I was left with the 1979 vintage MICRO-WORKS random access digitizer. There were still potential problems, such as unknown compatibility of an old board with the revised IEEE-696 bus standard and with a 5 Mhz clock. Also the possible need for a 60 Hz vertical sync video camera (which could then give trouble due to our 50 Hz mains frequency) remained an unknown factor. But the board was cheap ($700), so if it didn't work, at least, I could throw it away and learn from the experience. However, the board performed flawlessly from first installation. The minimum grab time is 4 secs including storage into RAM.

Video Camera

One of the important aspects of image analysis is the ability to measure the density of the pixels of the image. For many measurements, it is necessary to relate the pixel value to an optical-density value, so the transfer characteristics of the video tube become important. My research into the densitometric characteristics of video scanners has given considerable experience of the problems that can arise. Non-linear tube transfer characteristics require complicated calibration and correction procedures and the ever present burn-in phenomenon and image lag of vidicons, and even the newer saticon tubes create considerable difficulty for image analysis. For these reasons, I selected a solid-state camera based on a CCD device of 320 × 244 resolution. The resolution was compatible with that of the video digitizer board and the camera had the benefit of zero lag and strictly linear transfer characteristic. One factor which could have been either a benefit or a disaster was the 60 Hz vertical sync for this camera. Fortunately, no problems arose for compatibility with the board and the video display monitor.

Software Specifics

Image Storage

The first requirement was to get the image into memory, so that it could be manipulated easily and it was, therefore, necessary to fit the image into the 64Kb data allowance for compiled ZBASIC. To leave room for the strings and data of the program, this meant trimming the image down from the maximum of 256 × 256 to a size of 192 × 189 pixels. But this fitted nicely into 48 screen columns and 21 screen rows and allowed for a permanent on-screen menu. For speed, the scanning algorithm was written as an assembly language subroutine. The Micro-Works digitizer is a random access device and works by sending it an X-Y address through an S100 I/O port, after which it sends back the intensity value (0-63) of a single pixel. Because of the nature of the video scan, if the computer is not quick enough, it has to wait a complete frame period for every pixel. This can mean up to 10 minutes per scan! The trick is to time the port accesses so that the computer can read the data for one pixel and send the address of the next before a video frame has finished. The computer is not fast enough to grab pixels sequentially, but it is possible to grab a column of pixels allowing one video scan line period for each pixel. This scheme needs only as many video frames as the number of pixel columns (192) and results in a 4 second scan time.

Colour Image Display

The next task was to display the image. The trouble is that the image pixels are 6-bit giving 64 grey levels and the Z100 only has 8 colours. The problem was solved by mapping the 6 bits of each image pixel into two screen pixels using the RGB planes to effectively create a screen pixel-pair. The screen pixels are so close together that the colours fuse with potentially 64 possible combinations. However, half of the combinations are the same and there are effectively 32 possible colours. Combinations of these can be stored in lookup tables, so that the image can be transferred to the display memory in any sequence of these colours. The pixel mapping and screen display program took the most effort of the whole programming task. I did have some trouble in working out how to address the video RAM, since the Z100 displays only 9 out of the 16 available scan lines in every row (more on this below). But the technical manual gave me enough information to write an assembler code subroutine for pixel mapping and address calculation to transfer the 36Kb of image array to the screen memory. The image to screen transfer takes approximately 4 seconds.

Background Correction

The illumination of any image is never uniform and this becomes a problem in image analysis, because the thresholding for and feature selection and the measurements are made using the absolute pixel intensity. Any variation of illumination across the image causes trouble. So it is necessary to store a complete image of the background illumination and subtract it every time a scan is made. This is where microcomputers start to weaken at the knees for image analysis applications. Multiple images take a lot of memory and if a language, such as BASIC, is used there is only 64Kb of data space accessible. The main image in RAM already took up 36Kb, so the problem was where to find another 36Kb to permanently store the background image. The answer was to put it in the Z100 screen RAM using the 7 scan lines in every 16 which are not used by the visible screen display. The only problem that arose in doing this was that this area of video RAM was disturbed whenever text was written to the screen. The solution was simply not to write text horizontally adjacent to the image. There is still room for another image in the unused lines of the second page of screen RAM and it is "spare" for special applications. We are currently using it to construct multiple colour binary images from a 64 level original.

Image Enhancement

The distribution of grey levels in an image (such as in a poorly exposed photo) is often lacking many of the available levels. If this picture is mapped using the colours or grey levels directly relating to the pixel intensities, the result is a washed-out image which is difficult to see and work with. A much better picture can be made of the same image if the grey level distribution is reworked to make it as flat as possible, so that all of the available display grey levels are used. This technique is known as histogram equalization and vastly improves the appearance of low contrast images. The algorithm only takes a few lines of BASIC code and is very useful for discriminating parts of the image which are to be measured. It is also useful for enhancing whatever you might want to see better, such as the moon through a telescope or the face of a friend.

Thresholding And Binary Image Display

A basic requirement of image analysis is to select a threshold and display only those pixels darker or lighter, or in a slice between two thresholds. This was no real problem because by comparing each pixel value against a threshold, the blacker or whiter region could be made black or white by pixel conversion within the image to screen display subroutine. But having displayed this image, it is necessary to compare it with the original so that the correct levels can be determined. This requires two images which need to be instantly compared. The 4 sec display time was far to slow to allow alternate display of the two images. But the Z100 has two pages of video display and the image comparison function was elegantly performed by displaying the thresholded image in the second page of the Z100 video memory. Then, by toggling the value in the video display start address register whenever the space bar was pressed, the two pages could be instantly flicked between to compare the thresholding result.

So by effective use of the screen memory, the Z100 allows storage and manipulation of five images each of 36Kb. Each of these could be expanded to approximately 40Kb for a bigger image, if needed. If extra memory is added via the S100 bus, then many more images could be stored. But the existing scheme means that the memory does not have to be expanded beyond the main board limit of 192Kb.

Binary Image Amendment

Classical techniques in image analysis are erosion (removing a layer of pixels from the outside of a binary image) and dilation (adding a layer) and combinations of the two known as opening and closing. These processes were easily performed by two assembler routines which compare each pixel against the threshold value and either remove or include those on the "edge".

Editing

When a binary representation of the required features is roughly selected by thresholding and amendment, it is then necessary to be able to manually edit off unwanted features and to add or touch up any bits that have been undetected. This was achieved by displaying the binary image in green and painting out unwanted parts in red using a block "eraser" controlled by a digitizer tablet. Parts can also be added in green. Also features can be selected by surrounding them with boxes and filling outside the boxes in red. When the image is painted to the required result, the screen display is scanned by the image display subroutine "in reverse" and only those pixels of the image with a corresponding green pixel pair on the screen are marked as valid. Others are marked as not-valid. Bits 7 and 6 of each image byte are used as the valid/not-valid markers.

Measurement

The most important function, of course, is to measure the features which have been selected by the image amendment and editing processes. The problem is that automatic measurement is harder to do than you would think. The computer must be able to distinguish the patches of image belonging to each separate feature and make measurements of its area and perimeter, and also of the distribution of optical density of the pixels of the feature.

Features are found by searching for "valid" pixels (bit 6 set). When the first valid pixel of a feature is found, a loop is entered which propagates around the feature systematically accumulating the pixel numbers and values and setting them as not-valid as they are scanned. The measurement routine is written in BASIC because extra measurements can more easily be added to such a routine. Also specialized measurement routines, such as checking for or measuring inclusions within detected features can be added into the BASIC code.

Program Control

The input device is a digitizer tablet and a menu of commands is written on the screen and selected by moving a cross-hair to them and pressing the digitizer button. The main reason for using a tablet, rather than a mouse or a light pen is that some image analysis problems are just not possible by automatic analysis of the image, and it is necessary to trace around the profiles manually. The permanent connection of a digitizer tablet means that another tablet program can be run just for this purpose.

Programming

Once a series of imaging functions has been selected for a particular job, the same procedure must be followed over and over again. A facility for storing the commands and optionally saving them in a batch file was, therefore, included to give a degree of automation to the image analysis system.

Operation

Overall, the system is simple to use and capable of solving many tricky image analysis problems, because the software is easily modified for the introduction of whatever enhancements are required such as edge detectors, skeleton operators, inverters, smoothers, noise cleaners, curve fitters, circle fitters, and the like. The only problem is that the system will not cope with a moving video image due to the 4 second scan time. The advantage, though, is the low cost of the digitizer board.

Future Development

A more modern video digitizer board will be the first device to consider for short term future development of the existing system. There are now a number of real time video digitizer boards available, which produce images typically of 128 grey levels and 512 X 512 resolution, input and output lookup table facilities, etc. These have not been used for the present development, partly because of the cost ($2-4000), but also because I believe the computer required to store and handle such an image (or multiple images) in a realistic time has to be more powerful than the 8088 based machines that the boards are designed for. The other solution is to use specialist boards to handle the graphics and image manipulations independently, but this all costs extra and hardware development becomes reality. Also my applications require a solid-state video camera to enable accurate densitometric measurements. This means that a digitizer resolution greater than 256 X 256 pixels is wasted until higher resolution cameras become available at a reasonable cost.

Certainly though, the power of microcomputers has caught up with image analysis requirements and systems based on 16 or 32 bit microprocessors give impressive graphics performance. The high cost is the only problem at present. But microcomputer development is rapid and systems based on the newer type of microprocessor, such as the 80286 may soon go a long way to meeting memory and speed requirements at low cost.

But at the present time, the strength of the Zenith Z100 is that it is a complete off-the-shelf solution to a possible nightmare of incompatibility with various specialist graphics boards. The Zenith system is certainly limited compared to state of the art image analysis machines, but considering that it is an unmodified computer which anyone could set up, perhaps in another way it is state-of-the-art itself.

NEW

PARALLEL
INTERFACE BOARD
FOR YOUR H/Z89

ADD WHAT HEATH LEFT OUT

RUNS ALL PARALLEL*
PRINTERS/PLOTTERS

> NO SOFTWARE TO PURCHASE / MODIFY

> WORKS WITH ALL CP/M's AND HDOS

> USES STANDARD IBM / CENTRONICS CABLE

Quick 'P'

H89 PARALLEL PRINTER PORT

$69.50
$4 shipping
Calif. add 7% tax

Custom Circuit Design
ME
MCGAFFEY ENGINEERING
1556 Halford Avenue
Suite 169
Santa Clara, CA 95051
(408) 926-2132

* centronics standard

"C/80 . . . the best software buy in America!"
—MICROSYSTEMS

Now available in MS-DOS

Other technically respected publications like *Byte* and *Dr. Dobb's* have similar praise for **The Software Toolworks' $49.95** full featured **'C' compiler** for CP/M® and HDOS with:

- I/O redirection
- command line expansion
- execution trace and profile
- initializers
- Macro-80 compatability
- ROMable code
- and much more!

"We bought and evaluated over $1500 worth of 'C' compilers . . . C/80 is the one we use."

—Dr. Bruce E. Wampler
Aspen Software
author of "*Grammatik*"

In reviews published worldwide the amazing **$49.95 C/80** from **The Software Toolworks** has consistently scored at or near the top — even when compared with compilers costing ten times as much!

The optional **C/80 MATHPAK** adds 32-bit floats and longs to the C/80 3.0 compiler. Includes I/O and transcendental function library all for only **$29.95!**

C/80 is only one of 41 great programs each **under sixty bucks.** Includes: LISP, Ratfor, assemblers and over 30 other CP/M® and MSDOS programs.

For your **free** catalog contact:
The Software Toolworks
15233 Ventura Blvd., Suite 1118,
Sherman Oaks, CA 91403 or call 818/986-4885 today!

CP/M is a registered trademark of Digital Research.

Real Time Clock

Mark A. Ruthenbeck
Rt. 3, Box 3709
Pearland, TX 77581

This project, like many others, are started because of a simple need. I had acquired a Zenith Z-150 that had nearly everything I could ask for on the factory boards. The system came complete with 640K RAM, two serial ports, a Centronics printer port, and the standard color graphics. All this without any expansion cards! All I needed (wanted) was a real time clock. Since I work with PC's on a day-to-day basis, I am familiar with the AST and Techmar add-on MULTI function boards. But, all I needed was a simple little clock. So I decided to build my own. Hence, the project was born.

The Hardware

The circuit (Figure 1) is a simple one. There is the clock chip (U2), the I/O decoder (U1) and the clock support circuitry. Refer to Figure 2 for the parts list.

The time keeper of the circuit is a National-Semiconductor MM58167A Microprocessor Real Time Clock. The clock has eight addressable real time counters, 56 bits of RAM, two interrupt outputs, and a power down mode. See Figure 3 for the clock's address codes and functions.

The real time counters are divided into 2 BCD digits each. The counter format is shown Figure 4.

With BCD counting, there is a possible range of 00 to 99 in each byte, but as the table shows there are maximum allowable codes, 23 hours:59 minutes:59.999 seconds. The unused bits are kept at a logical 0. These unused bits are indicated by a "-".

The 56 bits of RAM can be used for power down storage or alarm times to be compared to the real time counters. The alarming functions only compare the day of week, hours, minutes, seconds, hundredths and tenths of seconds. However, this application uses the RAM for year storage, leap year indication, initialization of the clock flag, and month of last access. The leap year flag, initialization flag, and the last access date are all used by the Clock device driver and not the HUGCLOCK or SETRTC programs.

There are two interrupt outputs available from the clock. The first one is the INTERRUPT OUTPUT (pin 13, true high) and it is the most flexible. This signal can be programmed to give up to 8 different signals. They are: (1) 10 times a second, 10 Hz, (2) once a second, 1 Hz, (3) once per minute, (4) once per hour, (5) once per day, (6) once per week, (7) once per month, (8) when real time counters and RAM comparison occurs. To enable the interrupt output, a one is written to the interrupt control register (2D1H) at the bit location that gives the desired output frequency. See Figure 5 for interrupt register format. After the interrupt bit(s) have been set in the control register, the corresponding counter's rollover will clock the interrupt status register and cause the interrupt output to go high. To reset the interrupt register, you then read the interrupt status register.

The STANDBY INTERRUPT (pin 14, active low) is the second interrupt available from the clock. This interrupt is enabled by writing a one to the standby interrupt register. This interrupt is triggered by the level of the compare signal rather than the leading edge.

The power down mode of the clock is used for battery backup and power conservation. The clock goes into this mode when pin 23 is pulled low. When in the power down mode all the inputs and outputs are disabled, except for the STANDBY INTERRUPT. Also in this mode, all time keeping functions are maintained.

The PC Board

The PC board layout is a two-sided layout. The main reason for using two sides of the board is to pick up the computer's signals from the two-sided edge card connector. See Figure 6 for the artwork. The layout shown is 1:1.

The Software

There are two programs needed to operate the clock properly. The functions that need to be performed are (1) getting the time and date from the real time clock and then setting the DOS time and DOS date, and (2) setting the real time clock (time and date). The programs that accomplish these tasks are HUGCLOCK.EXE and SETRTC.EXE. The programs were written in C and compiled using the Manx Aztec C compiler. Some of the functions included in the code are Aztec extensions to the C language.

Setting the DOS time and date is handled by HUGCLOCK.EXE. This program sets the DOS time and DOS date by using the Set Current Time (function 2DH) and Set Current Date (function 2BH) DOS system calls.

To set the DOS time, you set the AH register with 2DH, CH register gets the hour, CL register is the minutes, and the seconds get set in the DH register, the hundredths of a second goes into the DL register. When all the registers are set, do an INT 21 (interrupt 21). Prior to loading the data in the appropriate registers it is first read out of the hardware clock. Since the hardware stores its data in BCD format, the data must first be converted into binary. The reading of the hardware is done by the inportb() function and the bcd2bin() function converts the BCD to BINary.

Likewise, to set the DOS date, you set the AH register with 2BH, CX register is the year, DH register the month, and the DL register is the day of the month. Next, of course, is the INT 21.

The registers AH, AL are register pairs that make up the AX register. The H is the high order byte and the L is the low order byte. The AX register is a 16 bit register. This goes also for the registers BX, CX, DX, etc.

The program's other components consist of the define statements, variable declarations, storing of variables, and printing statements. The define statements at the opening of the program are setting up the constants that are used. The variable declarations are setting up the structures that will store the values to be loaded into the hardware registers (AX, BX, etc.). The sysint() function uses the REGister structure and moves the data from the structure variable inregs to the hardware registers and on return the values of the hardware registers are placed in the structure variable outregs. The printing statements, printf() report to the screen the current date and current time that was placed in the DOS.

The SETRTC.EXE program works in the same manner, but only in reverse. The hardware clock is set from the data in the DOS time and DOS date. Before the clock is set, the program calls the resident command TIME and DATE. The internal TIME and DATE functions show the current DOS TIME or DOS DATE, then prompts the operator for the TIME or DATE for any changes. Then, the program reads the DOS time and DOS date, converts the BINary to BCD, then puts it in the clock (outportb()).

When the program calls for the the resident commands, time and date, it does it through the system() function call. This function finds the command interpreter through the "comspec" and invokes COMMAND.COM.

The only drawback of the design is that it does not have a year counter, so when you use the HUGCLOCK and the SETRTC programs, the year is not incremented on December 31 at 23:59:59. However, the device driver software handles this chore along with keeping up with leap years.

The second choice for software is the clock device driver. This installable device driver handles all the hardware I/O whenever there is a call for the date and time. With the device driver installed you do not need the HUGCLOCK or SETRTC programs. This device driver is available from the public domain and was published in the January 1984 issue of BYTE magazine. The primary advantage of running the device driver is that it handles leap years and year rollovers. To install the device driver you must have a line in your CONFIG.SYS file that says DEVICE = CLOCK.COM.

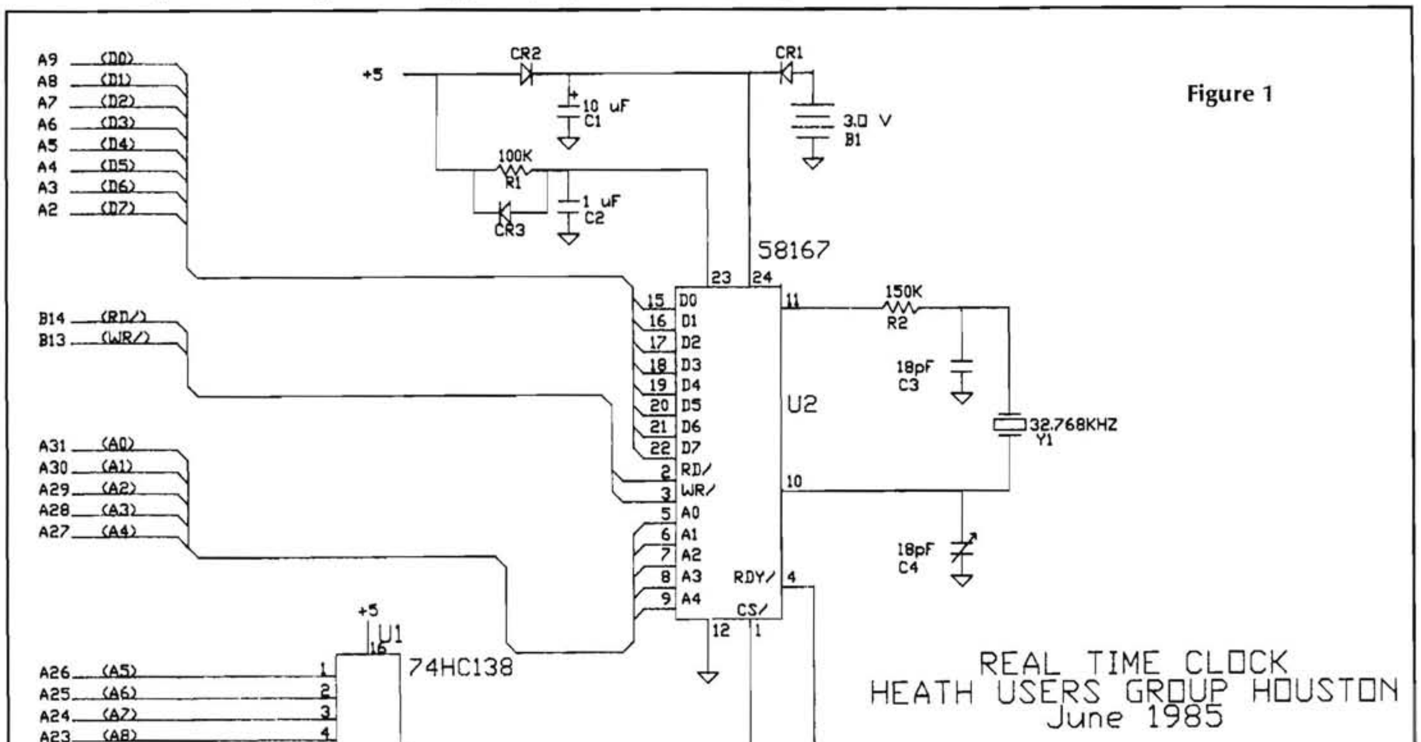


Figure 1

The Real Time Clock (RTC 685) is available, in kit form, to all HUG members for $29.95 + shipping from:

Heath Users' Group Houston
1704 West Loop North
Houston, Tx 77008

The PC Board and the software is available from me for $11.50, plus 1.00 for shipping and handling.

References

IBM Personal Computer Hardware Reference Library, IBM 1983.

MS-DOS Version 2 Programmers Utility Pack, Zenith Data Systems 1984.

David Broadwell, "Real-Time Clocks and PC-dos 2.0," Byte, Jan. 1984.

CMOS DATABOOK, National Semiconductor 1984

About The Author

Mark Ruthenbeck is an Electronic Design Engineer for a Houston based oil field services company. He is responsible for designing micro based proprietary data acquisition systems for use in the oil patch. He is currently serving as secretary to the HUG Houston group and the group's sysop for the club's FIDO BBS.

Address Codes And Functions

Clock Address (Hex)	Function
2C0	Counter Ten Thousandths of Seconds
2C1	Counter Hundredths & Tenths of Seconds
2C2	Counter Seconds
2C3	Counter Minutes
2C4	Counter Hours
2C5	Counter Day of Week
2C6	Counter Day of Month
2C7	Counter Month
2C8	RAM Ten Thousandths of Seconds
2C9	RAM Hundredths & Tenths of Seconds
2CA	RAM Seconds (used for year storage)
2CB	RAM Minutes
2CC	RAM Hours
2CD	RAM Day of Week
2CE	RAM Day of Month
2CF	RAM Month
2D0	Interrupt Status Register
2D1	Interrupt Control Register
2D2	Counters Reset
2D3	RAM Reset
2D4	Status Bit
2D5	GO Command
2D6	STANDBY INTERRUPT
2DF	Test Mode

Figure 3

Real Time Clock RTC 685

Part Description	Reference Designator
PC Board	RTC 685
MM58167AN 24 pin chip	U2
MC74HC138 16 pin chip	U1
32.768 KHZ Crystal	Y1
3.0 V Battery Sanyo CR2430	B1
Diode 1N914	CR1,CR2,CR3
Resistor 150k ohm RN55D 1503F	R1,R2
10 uF/16 volt Capacitor (orange drop)	C1 (Polarized)
1.0 uF 50 volt Capacitor 105M 50V	C2
18 pF Capacitor CM05CD180J03	C3
5-40 pF Capacitor adjustable	C4
24 PIN SOCKET 224-AG-29D, Augat	For U1
16 PIN SOCKET 216-AG-29D, Augat	For U2
Battery Holder Keystone #105	For B1

Figure 2

Counter Addressed	TENS D4 D5 D6 D7	Max BCD Code	UNITS DØ D1 D2 D3	Max BCD Code
1/1Ø,ØØØ of Seconds	D4 D5 D6 D7	9	- - - -	Ø
Hundredths & Tenths	D4 D5 D6 D7	9	DØ D1 D2 D3	9
Seconds	D4 D5 D6 D7	9	DØ D1 D2 D3	9
Minutes	D4 D5 D6 D7	5	DØ D1 D2 D3	9
Hours	D4 D5 - -	2	DØ D1 D2 D3	9
Day of the Week	- - - -	Ø	DØ D1 D2 -	7
Day of the Month	D4 D5 - -	3	DØ D1 D2 D3	9
Month	D4 - - -	1	DØ D1 D2 D3	9

Figure 4

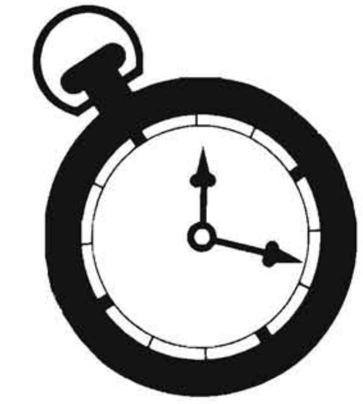
Interrupt Register Format

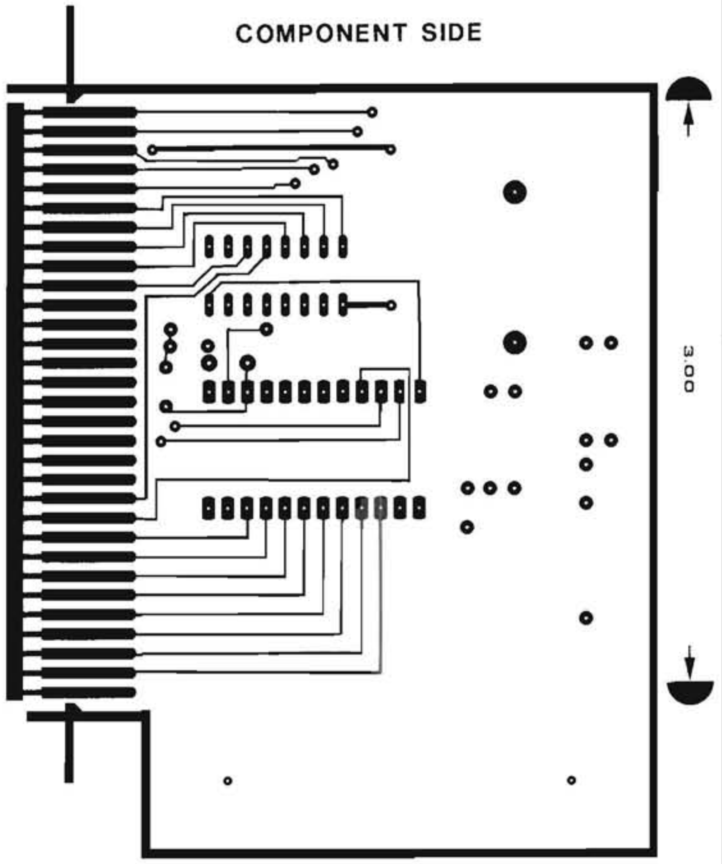
D7	D6	D5	D4	D3	D2	D1	D0
1/Mon	1/week	1/day	1/hr	1/min	1/sec	1/MSec	Compare

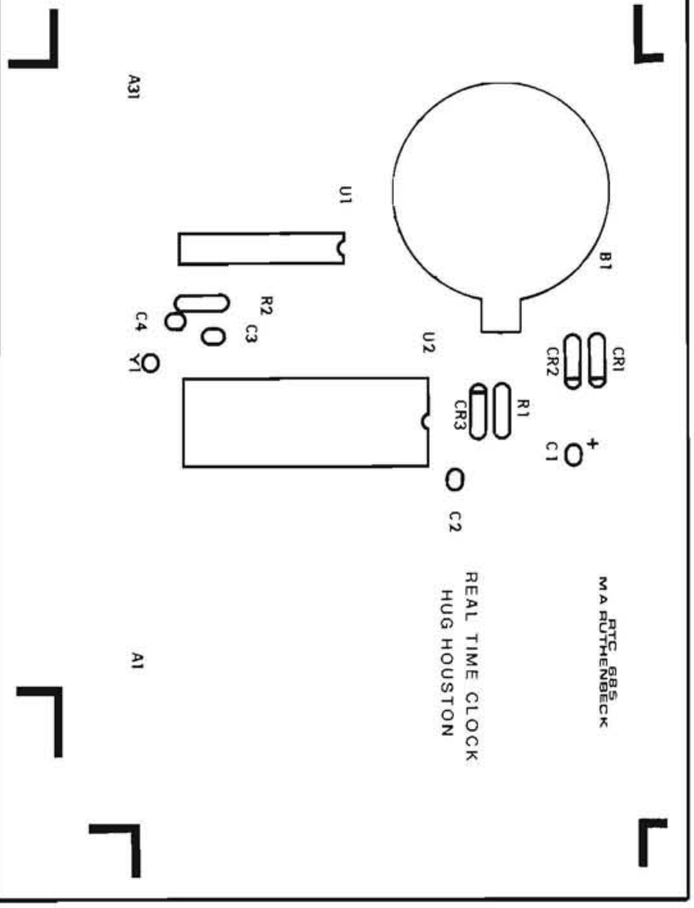
Figure 5

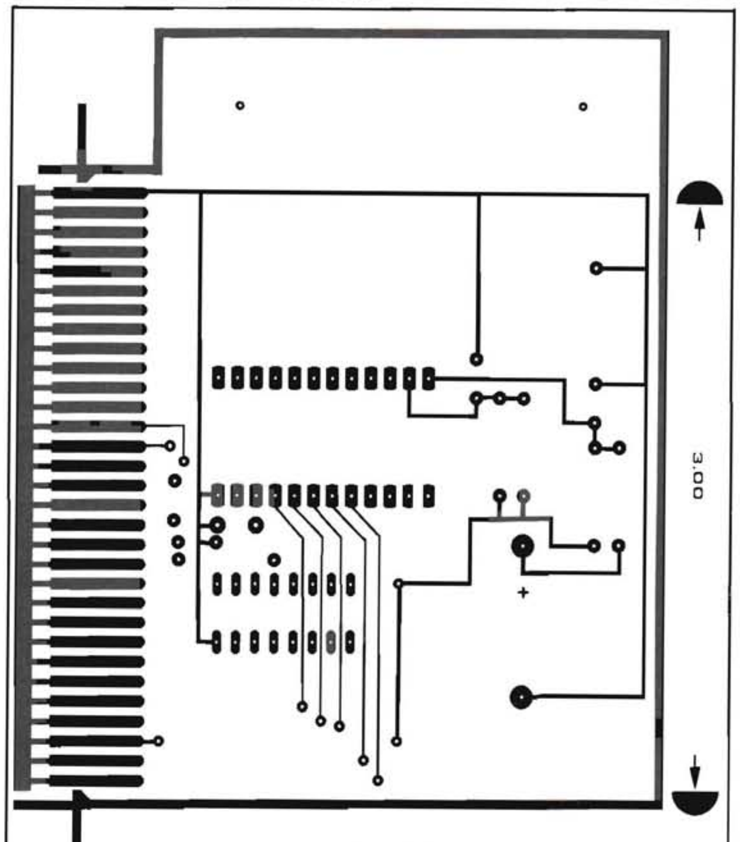
```
/********* set dos time from hardware clock ************
Comments added for clarity       MAR 08/28/85

This program gets the time and date from the real time
clock and sets the dos time  the clock is located at
2c0h - 2dfh.

This program is for non-commercial use only. (C)copyright
1985
        Mark A Ruthenbeck                  04/27/85
*******************************************************/
#include   <stdio.h>

            /* HARDWARE CLOCK PORTS  */

#define     CLK_MILLI      0x2c0     /* milliseconds */
#define     CLK_TENTH      0x2c1     /* tenths of seconds */
#define     CLK_SEC        0x2C2     /* seconds  */
#define     CLK_MIN        0x2c3     /* minutes  */
#define     CLK_HR         0x2C4     /* hours  */
#define     CLK_DOW        0x2c5     /* day of week */
#define     CLK_DOM        0x2c6     /* day of month */
#define     CLK_MON        0x2C7     /* month  */
#define     CLK_YAR        0x2ca     /* year in decimal
                                        - 80 */

            /* END CLOCK PORT ASSIGNMENTS  */


              /* dos function equates */

#define     INT21          0x21     /* interrupt 21 */
#define     GET_TIME       0x2c     /* function #  */
#define     GET_DATE       0x2a
#define     SET_TIME       0x2d
#define     SET_DATE       0x2b

struct      REGS
   {
      int  ax;               /* a place for all the
                                hardware registers */

      int  bx;
      int  cx;
      int  dx,
      int  si;
      int  di;
      int  ds;
      int  es;
   };

main()
{
   struct  REGS      inregs,outregs; /* hardware registers
                                        before  */
                                     /* and after a system
                                        call */

/*********************** NOTE ***********************


 The real time clock stores the time and date (except
 for the year ) in packed bcd.

 So all hardware calls must be converted (bcd2bin)
*******************************************************/


/* to set time ah = 2dh, ch = hour (0-24), cl = min (0-59),
   dh = sec (0-59) dl = hundredths of sec (0-99)          */

   inregs.ax = (SET_TIME << 8);
   inregs.cx = ((bcd2bin(inportb(CLK_HR)) << 8) +
               (bcd2bin(inportb(CLK_MIN)))),
   inregs.dx = ((bcd2bin(inportb(CLK_SEC)) << 8) +
               (bcd2bin(inportb(CLK_TENTH))));

   sysint(INT21,&inregs,&outregs);       /* all set so call interrupt */

   if (outregs.ax & 0x00ff)
     {
        printf("\n.    Invalid Time in RTC      \n");
        exit(-1);
     }

printf("Current time is %d:%02d:%02d.%d \n",(inregs.cx)>>8,
      (inregs.cx & 0x00ff),(inregs.dx)>>8,(inregs.dx & 0x00ff));

/*    The time is set so lets set the date..
        to set the date ah = 2bh, cx = year (1980 - 2099),
        dh = month (1-12), dl = day (1-31)         */

   inregs.ax = (SET_DATE << 8);
   inregs.cx = (inportb (CLK_YAR) + 1980);
   inregs.dx = ((bcd2bin(inportb(CLK_MON)) << 8) +
               (bcd2bin(inportb(CLK_DOM))));

   sysint (INT21,&inregs,&outregs);

   if (outregs.ax & 0x00ff)
     {
       printf("\n.     Invalid Date in RTC.    \n");
       exit (-1);
     }

printf("Current date is %d/%02d/%d \n",(inregs.dx) >> 8,
      (inregs.dx & 0x00ff),inregs.cx);

}

bcd2bin(arg)
  int arg;

{
   return  ((arg/16 * 10) + (arg % 16)) ;
}

/********* set hardware clock time from dos ************
Comments added for clarity              MAR     08/26/85

This program sets the time and date of the real time clock,
it gets the the time from dos.  the clock is located at
2c0h - 2dfh.

This program is for non-commercial use only. (C)copyright
1985

        Mark A Ruthenbeck                  05/17/85
*******************************************************/
#include    <stdio.h>

             /* HARDWARE CLOCK PORTS  */

#define     CLK_MILLI    0x2c0    /* milliseconds */
#define     CLK_TENTH    0x2c1    /* tenth of seconds */
#define     CLK_SEC      0x2C2    /* seconds */
#define     CLK_MIN      0x2c3    /* minutes */
#define     CLK_HR       0x2C4    /* hours */
#define     CLK_DOW      0x2c5    /* day of week */
#define     CLK_DOM      0x2c6    /* day of month */
#define     CLK_MON      0x2C7    /* month */
#define     CLK_YAR      0x2ca    /* year in decimal - 80 */

             /* END CLOCK PORT ASSIGNMENTS  */

                /* dos function equates */

#define     INT21        0x21     /* interrupt 21 */
#define     GET_TIME     0x2c     /* function #  */
#define     GET_DATE     0x2a
#define     SET_TIME     0x2d
#define     SET_DATE     0x2b


struct      REGS
   {
     int    ax;       /* a place for all the hardware
                         registers */
     int    bx;
     int    cx;
     int    dx;
     int    si;
     int    di;
     int    ds;
     int    es;
   };

main()
{
   struct   REGS    inregs,outregs; /* hardware registers */
                                    /* before and after a */
                                    /* system call */

/* before we set the real time clock (rtc) lets get the
   current time and date using the time & date resident
   commands. */
   system ("time");              /* set dos time */
   system ("date");              /* and date    */

/************************ NOTE **************************

 The real time clock stores the time and date (except
 for the year ) in packed bcd.

 So all hardware calls must be converted (bin2bcd)
*********************************************************/

/* to get the dos time ah = 2dh and then INT 21          */
/* after the get time functions returns the time is :
 ch = hour (Ø-24), cl = min (Ø-59), dh = sec (Ø-59)
   dl = hundredths of sec (Ø-99)          */

   inregs.ax = (GET_TIME << 8);

   sysint(INT21,&inregs,&outregs);      /* all set so call
                                           interrupt */

/*  the time is in outregs so lets put it in the clock  */
   outportb(CLK_MIN,(bin2bcd(outregs.cx & Øx00ff)));
   outportb(CLK_HR,(bin2bcd(outregs.cx >> 8)));
   outportb(CLK_TENTH,(bin2bcd(outregs.dx &Øx00ff)));
   outportb(CLK_SEC,(bin2bcd(outregs.dx >> 8)));

/*   The time is set so lets set the date.
       to get the date ah = 2bh,then INT 21, on return
       cx = year (198Ø - 2Ø99), dh = month (1-12),
       dl = day (1-31)                 */

   inregs.ax = (GET_DATE << 8);

   sysint (INT21,&inregs,&outregs),

   outportb(CLK_YAR,(outregs.cx - 198Ø)),
   outportb(CLK_DOM,bin2bcd(outregs.dx & ØxØØff));
   outportb(CLK_MON,bin2bcd(outregs.dx >> 8));

   printf ("\n The clock is now set!\n\n");

}
```

```
bin2bcd(arg)
  int arg;
{
   return  ((arg/1Ø * 16) + (arg % 1Ø)) ;
}

system(cmd)
char *cmd;
{
        register char *prog;
        char *getenv();
        char buffer|13Ø|;

        if ((prog = getenv("COMSPEC")) == Ø)
                prog = "/command.com";
        sprintf(buffer+1, "/C %.123s\r", cmd);
        buffer|Ø| = strlen(buffer+1) - 1;
        if (fexec(prog,Ø,buffer,Ø,Ø) == -1)
                return -1,
        return wait();
}
```









SOLDER SIDE

768K Z-100™ WITH OUR ZMF100™ PACKAGE

$65 EACH

Our ZMF100™ modification package lets you plug 256K RAM chips into your "old" motherboard Z-100™ computer. Up to 768K (27 RAM's) on the part number 85-2653-1 motherboard.

Simple, mostly plug-in installation requires no trace cutting or funny assembly tricks. Fully reversible. No permanent changes to the motherboard.

Requires disassembly of the computer and some easy soldering to connect the pin-1's of the memory sockets together. All materials (except memory chips and solder) are supplied along with complete, detailed instructions.

Price: $65 each.

Quantity Discounts. UPS Shipping Free.
Order Direct or See Your Local Heath/Zenith Dealer.
Literature Available. VISA or Mastercard Accepted.

FBE

FBE Research Company, Inc.
P.O. Box 68234
Seattle, Washington 98168
(206) 246-9815

INTELLIBURNER

EPROM-EEPROM-MICROCOMPUTER PROGRAMMER

UNIVERSAL PROGRAMMING CAPABILITIES AT AN AFFORDABLE PRICE

- **Ultra Fast Programming – 2716's in 16 Seconds**
- **Programs & Verifies 8K thru 256K Single Voltage EPROMs**
- **Erases, Programs & Verifies 2815, 2816 & 2817 EEPROMs**
- **Programs & Verifies 8748 and 8751 Series MICROCOMPUTERS***
- **Programming Characteristics Selected by Convenient Personality Jumper Plug (DIP Header)**
- **Program, Verify, Status, & Diagnostic Display with Tricolor LED**
- **Serial Interface — 3, 4, or 5 wire — 1200 to 19200 Baud**
- **Supports ACK/NAK, XON/XOFF and READY/BUSY Protocols**

NO SPECIAL SOFTWARE REQUIRED. Transfer disk files (Intel Hex or Motorola 'S' Format) to EPROM with your system's line printer or modem software. Transfer EPROM contents to disk file with your system's modem software. Or use the supplied software ** to transfer any binary or ASCII file to/from EPROM.

PROGRAMS:

2758	2716	27128	2815	8741*	8748*
2516	2732	27128A	2816	8742*	8749*
2532	2732A	27256	2817	X2212*	8751*
2564	2764	68764	27C Series		8755*

*Requires low cost personality adapter

SOFTWARE AVAILABLE FOR: **

CP/M systems on 8" SSSD – many 5¼ formats
TRS-80 Model I & III TRSDOS – NEW! TRSDOS 6.X
Heath H8/H89 HDOS & CP/M
ZENITH Z90 & Z100 CP/M – Z100 ZDOS – Z150
KAYPRO II/IV/10/16/286/2000
IBM PC. PC/XT. AT – many MS-DOS systems

IntelliBurner Programmer with Software	$ 299.00
Stand-Alone RAM OPTION with 64K Bit RAM (Expandable to 256K Bit)	89.00
RS-232 Interconnect Cable	12.00
IntelliBurner PC Boards, EPROM, Plans & Software	99.00

Low Cost "DumBurner" serial programmers harness the power of your personal computer with the supplied software for full programming capabilities:

DumBurner II Programmer for 28 Pin and 24 Pin EPROMs and EEPROMs (with Software)	$ 199.00
DumBurner II Bare PC Board, Plans & Software	59.00
16K/32K DumBurner for 24 Pin EPROMs (with Software)	149.00
16K/32K DumBurner PC Board, Plans & Software	39.00

ROSS CUSTOM ELECTRONICS
INCORPORATED
1551 Sandra Drive
Boulder City, Nevada 89005
PHONE (702) 293-7426

! ! NOW SHIPPING with ALL 'BURNERS ! !
TOOLKIT Software for Editing EPROM Information, Plus HEX/MOTOROLA 'S'/Binary File Conversion Utilities

Ultra Violet Products EPROM Erasers
Model DE-4 - Holds 8 EPROMS - Special Prices

Add $3 Shipping & handling ($2 Bare Boards). C.O.D.'s accepted. Foreign Orders add required postage. Specify Environment and Media Requirements.
H8, H89, Z100 are ™ Heath/Zenith. CP/M is ™ Digital Research. TRS-80 is ™ Tandy Corp.. IBM-PC, PC/XT, AT are ™ IBM. MS-DOS is ™ Microsoft.

Z-100 STORAGE SOLUTIONS

1.2 MEGABYTES 5 ¼" FLOPPY DRIVES

Internal	
• single drive	$240.00*
• dual drives	$440.00*
External (with case & PS)	
• single drive	$365.00*
• dual drives	$550.00*
Bare Drive	$190.00*
High Density Diskettes	$ 45.00*

*Applicable taxes and shipping extra

All of the above work with your standard Z-100 floppy controller and operating systems as is, along with your standard 5 ¼" drives.

HARD DISK SYSTEMS - Under development

DIVERSE SYSTEMS

7916 S. Logan Dr., Littleton, Co 80122
(303) 798-3302, after 5:30 pm & weekends

A Print Processor For Magic Wand

Randall Stokes
401 S. Silver
Centralia, WA 98531

I have been following, with some interest, the Buggin' HUG discussion of Magic Wand and its lack of support for dot matrix printers. This problem has long frustrated me, but I find the use of the \OUT command extremely tedious and time consuming. Not only must I continually remember what numbers do what, but the finished document with commands imbedded is very difficult to read.

My solution to this problem is a quick preprocessor program. You create your document in Magic Wand using all of their default printer codes ('@' for boldface, '_' for underline, etc.). Then you run PROCESS, using either of the available command syntaxes. This program converts each special character into its corresponding \OUT sequence, keeping particular track of those characters that act as toggles. Then you PRINT your file normally.

I find the inconvenience of having to run PROCESS before printing a small price to pay for the ease of using Magic Wand's control codes. Further, by modifying the program slightly, you can designate more than the four codes MW offers you, allowing you instantly to take advantage of more of your printer's capabilities. It would be easy to add italics, compressed print, etc.

Another possibility would be accessing your printer's character graphics set. While editing a document in Magic Wand, try this trick: depress the 'off line' key on your H89 or H19. Press 'ESC', then 'F'. Press 'off line' again. Now start scrolling the page with the function keys. All the lower case letters are displayed in graphics! To reset to normal display, follow the same procedure but press 'ESC','G'.

Here's where PROCESS comes in: the character graphics set on your printer probably does not match character for character the one on the terminal; that is, the symbols may be there, but in a different order, represented by different ASCII codes. You could, however, design a special version of PROCESS that would translate each character for your terminal into its equivalent character for your printer. Now you can EDIT the graphics on the screen, PROCESS it, then PRINT it! A poor man's graphics package! (Note: If your printer's graphic characters fall between ASCII 128 and 255, Magic Wand's PRINT will not work properly. You can still print your processed file with PIP (PIP LST:=<filename>) or use control-P and the TYPE command.)

I have recently added a new feature to PROCESS which allows the actual printing of all the command characters, a capability usually sacrificed with Magic Wand. In order to print a character literally ('&', for example), simply enter the character twice in your source ('&&'). PROCESS keeps track of these "doublings", and substitutes the appropriate \OUT sequence to print the character. See the TEST file reproduced here for an example. (Note: the BASIC version of PROCESS does NOT provide this feature.)

The original program was written in assembly language, source enclosed. I ran a 10K test file of random ASCII characters; it took one minute, 24 seconds to process (H89 2MHz). Incidentally, for those of you who hate assembly language, I enclosed an MBASIC program that ran the same test file in about five minutes. It's easier to type in, but you pay with speed and ease of execution.

The assembly source itself is rather bare-bones, and could be beefed up. In particular, it does not provide for changing the special character values. You will have to assemble different versions for different uses (or use DDT to change special characters). Still, it does the job, and is a good starting point for more ambitious programmers.

For those of you unfamiliar with assembly language, I provide the following assembly instructions. The code can be assembled using ASM and LOAD, both provided on your CP/M distribution disk. I'll assume that both ASM and LOAD are on the default drive, and that you are creating the source file on B:.

Enter the code with an editor (Magic Wand does fine), then assemble it with ASM. Type:

```
ASM B:PROCESS
```

A correct assembly will print two hex numbers and the words "USE FACTOR". Anything else and you should examine the source (it must be EXACTLY like the one you are copying from), then reassemble. Once assembled, use LOAD:

```
LOAD B:PROCESS
```

You now have a runnable (.COM) file. See the source for command syntax. For those of you who hate typing in anything, I can download the whole thing to a hard sector disk. Contact me at the listed address.

Try PROCESS out; I think you'll agree it beats using \OUT any day!

✱

BASIC Listing

```
10 'PROCESS.BAS — file preprocessor for MAGIC WAND
20 '    processes an MW text file, substituting appropriate
        printer codes for
30 '    MW's special codes
40 '                              by Randall Stokes
50 '                                 401 S  Silver
60 '                                 Centralia, WA  98531
70 '
80 '
90 DIM D$(20),FLAG(20)              'Room for expansion
100 X=1
110 READ D$(X)                      'fill expansion table
120 WHILE D$(X)<>"$END$"
130     X=X+1
140     READ D$(X)
150 WEND
160 KEY$=CHR$(95)+"@<>&"            'list of special characters
170 ON ERROR GOTO 400
180 INPUT "source file: ",FILE1$
190 OPEN "I",1,FILE1$
200 INPUT "destination file: ",FILE2$
210 OPEN "I",2,FILE2$               'See if destination exists
220 PRINT "Destination exists already!":GOTO 310
230 OPEN "O",2,FILE2$
240 WHILE NOT EOF(1)
250     A$=INPUT$(1,1)
260     A=INSTR(KEY$,A$)
270     IF A=0 THEN PRINT #2,A$;:GOTO 300
                                  'not special character
280     FLAG(A)=NOT FLAG(A)         'process special character
290     PRINT #2,D$(A*2+FLAG(A)),
300 WEND
310 CLOSE
320 END
330 '
340 DATA "\OUT27,45,1\","\OUT27,45,0\","\OUT27,69\",
     "\OUT27,70\"
350 DATA "\OUT27,83,0\","\OUT27,84,27,72\",
     "\OUT27,83,1\","\OUT27,84,27,72\"
360 DATA "\OUT27,52\","\OUT27,53\",$END$
370 '
380 '   error handling routines
390 '
400 IF ERR=53 AND ERL=190 THEN PRINT
     "Source file not found!":RESUME 310
410 IF ERR=53 AND ERL=210 THEN RESUME 230
420 ON ERROR GOTO 0
```

Assembly Listing

```
;  PROCESS.ASM  --   9/14/84      Randall Stokes
;
;  File pre-processor for Magic Wand with unsupported
     printer.
;
, This program reads a file, searches for any MW default
, printer codes, and expands them into '\OUT . . .' syntax.
; You can add any other special characters you want
    recognized
; as well by changing the scanning routine and the flag
    table.
;
; Two possible command lines from CCP:
;
;               PROCESS D:FILENAME.EXT
;
;                       will process D:FILENAME.EXT and create
;                       D:FILENAME.PRC on the same drive
;
;               PROCESS D:FILENAME.EXT E:FILENAM2.EX2
;
;                       will process D:FILENAME.EXT and create
;                       E:FILENAM2.EX2 on specified drive
;
; Program does warm boot when finished
;
;SYSTEM EQUATES:
,
BOOT    EQU     0000H
BDOS    EQU     0005H
FCB1    EQU     005CH   ;first file name
FCB2    EQU     006CH   ;second file name
SFCB    EQU     FCB1    ;first file control block
SBUFF   EQU     0080H   ;default source buffer
TPA     EQU     0100H   ;beginning of TPA
;
PRINTS  EQU     9       ;print string function
OPENF   EQU     15
CLOSEF  EQU     16
DELETEF EQU     19
READF   EQU     20
WRITEF  EQU     21
MAKEF   EQU     22
SETDMA  EQU     26
;
LF      EQU     0AH
CR      EQU     0DH
CTLZ    EQU     1AH

        ORG     TPA

;
;               *** set local stack ***
;

START   LXI     SP,STACK

;
;               *** check syntax ***
;

        LDA     FCB1+1
        CPI     ' '
        JZ      SYNERR          ;print correct syntax
        LDA     FCB2+1
        CPI     ' '
        JNZ     MOVE2           ;jump if 2nd form

,
,               *** process first command form ***
,

        MVI     C,9             ;file drive + name --
                                 no extension
        LXI     D,FCB1
        LXI     H,DFCB          ;destination FCB
MOVE1   LDAX    D               ;get character
        INX     D               ;ready next
        MOV     M,A             ;move it
        INX     H               ;ready next
        DCR     C
        JNZ     MOVE1           ;loop until done
        INX     H               ;skip extension
        INX     H
        INX     H
        MVI     C,4
        XRA     A
M1      MOV     M,A
        INX     H
        DCR     C
        JNZ     M1              ;move nulls into new FCB
        JMP     READY           ;ready to go

;
;               *** process second command form ***
;

MOVE2   MVI     C,16
        LXI     D,FCB2
        LXI     H,DFCB
M2      LDAX    D
        INX     D
        MOV     M,A
        INX     H
        DCR     C
        JNZ     M2


;
;               *** open source file ***
;

READY   XRA     A
        STA     DFCBCR          ;zero cr in destination
                                 FCB
        LXI     D,SFCB          ;open source
        CALL    OPEN
        CPI     ØFFH
        JZ      NOFILE          ;source file doesn't exist
        LXI     D,DFCB
        CALL    OPEN            ;try to open destination
        CPI     ØFFH
        JNZ     DEXISTS         ;jump if destination
                                 already exists
        LXI     D,DFCB
        CALL    MAKE
        CPI     ØFFH
        JZ      NODIR           ;no directory space

;
;               *** both files open -- now process ***
;
;               B = source buffer pointer
;               C = dest buffer pointer
;

        LXI     D,DBUFF         ;destination buffer
        MVI     C,128
        MVI     B,1             ;force write
LOOP    CALL    GETCHAR
        JZ      DONE
        PUSH    D
        PUSH    H
L1      CALL    ANALYZE         ;is it a special character?
        JZ      SPECIAL         ;yes
        POP     H
        POP     D
        CALL    STORECHAR
        JZ      DSKFULL
        JMP     LOOP

;
;               *** process special character
;               entry: a = char
;                      hl = flag table entry
;                      stack = SBUFF location (top),
;                              DBUFF location
;                      bc = counters
;

SPECIAL MOV     D,A             ;save character
        XTHL                    ;flag table ptr on stack,
                                 hl = SBUFF location
        CALL    GETCHAR
        JZ      DONESP          ;done, but write char in
                                 D first
        CMP     D               ;same as previous special
                                 char?
        POP     D
        XTHL
        XCHG                    ;hl = flag, de = DBUFF,
                                 stack = SBUFF
        JNZ     EXPAND          ;characters different
        PUSH    D
        LXI     D,5
        DAD     D               ;point to expand code
                                 address
        POP     D
        CALL    SENDEXP         ;send expanded code
        JZ      DSKFULL
        POP     H
        JMP     LOOP

;
;               *** process if both characters different
;               first expand char1, then return to L1 to
;               analyze char2
;               entry: a = char2
;                      bc = counts
;                      de = DBUFF (current)
;                      hl = flag
;                      stack top = SBUFF (current)
;

EXPAND  PUSH    PSW             ;save char2
        MOV     A,M             ;get flag
        CMA                     ;flip it
        MOV     M,A
        INX     H
        INR     A               ;should it be second
                                 entry?
        JZ      E1              ;no
        INX     H
        INX     H
E1      CALL    SENDEXP
        JZ      DSKFULL
        POP     PSW             ;retrieve char2
        XCHG                    ;a = char2, hl = DBUFF,
                                 stack = SBUFF
        XTHL                    ;hl = SBUFF, stack = DBUFF
        PUSH    H
        JMP     L1

;
;               *** done, but send char in D first
;               entry: bc = counts
;                      d = char1
;                      hl = SBUFF
;                      stack = DBUFF
;


DONESP  MOV     A,D
        POP     D
        CALL    STORECHAR
        JZ      DSKFULL

DONE    MVI     A,CTLZ          ;write EOF
        MVI     C,1             ;force write
        CALL    STORECHAR
        JZ      DSKFULL
        JMP     GOOD

;
;               *** get character -- reads next sector
;                   if necessary
;               entry: bc = counts
;                      hl = previous SBUFF location
;               exit:  a = char
;                      b, hl adjusted
;                      z set if EOF
;               uses:  a,bc,hl
;

GETCHAR INX     H
```

```
        DCR     B               ;finished with this
                                 sector?
        JNZ     G1              ;no
        CALL    READ
        ORA     A
        JNZ     G2              ,if EOF
        LXI     H,SBUFF
        MVI     B,128
G1      MOV     A,M             ;get next char
        CPI     CTLZ            ;EOF?
        RET                     ;z set if EOF
G2      XRA     A               ;set z
        RET                     ;z set if EOF

;
;               *** analyze character to see if special
;                   point hl to appropriate flag table
                    entry if so
;               entry: a = char
;               exit:  a = char
;                      hl = flag table if special
;                      z set if special
;               uses:  a,de,hl
;

ANALYZE LXI     D,7             ;offset to next table entry
        LXI     H,FLAG1         ;point to first flag
        CPI     '_'
        RZ                      ;z set if special
        DAD     D
        CPI     '@'
        RZ
        DAD     D
        CPI     '<'
        RZ
        DAD     D
        CPI     '>'
        RZ
        DAD     D
        CPI     '&'
        RZ
        DAD     D
        CPI     '\'
        RET                     ,z set only if special

;
;               *** store character in DBUFF, write if
                    necessary
;               entry: a = char
;                      bc = counts
;                      de = DBUFF (next available location)
;               exit:  z set if disk full
;                      c,de adjusted
;               uses:  a,bc,de
;

STORECHAR       EQU     $
        STAX    D
        INX     D
        DCR     C
        CZ      WRITE           ;if necessary
        RET

;
;       *** send expanded code to destination
;       entry: bc = counts
;              de = DBUFF (current)
;              hl = address for start of proper
;                   expanded code
;

SENDEXP MOV     A,M             ;get address in hl
        INX     H
        MOV     H,M
        MOV     L,A
S1      MOV     A,M             ,get a character
        CPI     0               ;end yet?
        JZ      S2              ,yes
        INX     H
        CALL    STORECHAR
        RZ                      ;return if disk full
        JMP     S1
S2      INR     A               ;reset z flag
        RET

;
;               ***     system calls    ***
;

OPEN    MVI     C,OPENF
        JMP     BDOS
CLOSE   MVI     C,CLOSEF
        JMP     BDOS
MAKE    MVI     C,MAKEF
        JMP     BDOS
READ    PUSH    B
        PUSH    D
        LXI     D,SFCB
        MVI     C,READF
        CALL    BDOS
        POP     D
        POP     B
        RET
WRITE   PUSH    H               ;save registers
        PUSH    B
        MVI     C,SETDMA        ;point to dest buffer
        LXI     D,DBUFF
        CALL    BDOS
        LXI     D,DFCB          ,write the sector
        MVI     C,WRITEF
        CALL    BDOS
        PUSH    PSW             ;save error code, if any
        MVI     C,SETDMA        ;point back to source
                                 buffer
        LXI     D,SBUFF
        CALL    BDOS
        POP     PSW             ;restore registers
        POP     B
        POP     H
        ORA     A
        JNZ     W1              ;disk full error
        INR     A               ;reset z flag
        LXI     D,DBUFF
        MVI     C,128
        RET
W1      XRA     A               ;set z if full
        RET

;
;               ***     error routines  ***
;

SYNERR  LXI     D,MSG1
        JMP     FINISH
NOFILE  LXI     D,MSG2
        JMP     FINISH
DEXISTS LXI     D,MSG3
        JMP     FINISH
NODIR   LXI     D,MSG4
        JMP     FINISH
DSKFULL LXI     D,DFCB
        MVI     C,DELETEF
        CALL    BDOS
        LXI     D,MSG5
        JMP     FINISH
WRTPROT LXI     D,DFCB
        MVI     C,DELETEF
        CALL    BDOS
        LXI     D,MSG6
        JMP     FINISH

;
;               ***     normal finish   ***
;
```

```
GOOD    LXI     D,DFCB
        CALL    CLOSE           ;close dest file
        CPI     ØFFH
        JZ      WRTPROT         ;write protected?
        LXI     D,NORMAL
FINISH  MVI     C,PRINTS
        CALL    BDOS            ;write normal exit message
        JMP     BOOT            ;return to CP/M

;
;               ***     flag table      ***
;

FLAG1   DB      Ø               ;first flag -- underline
        DW      CODE1
        DW      CODE2
        DW      UNCODE
        DB      Ø               ;second flag -- boldface
        DW      CODE3
        DW      CODE4
        DW      BFCODE
        DB      Ø               ;third flag -- superscript
        DW      CODE5
        DW      CODE6
        DW      SUCODE
        DB      Ø               ;fourth flag -- subscript
        DW      CODE7
        DW      CODE6
        DW      SBCODE

;
;               ***  insert any additional flags here ***
;               the term 'CODEn' refers to the proper code from the
;               table below  For each flag, the first code sets that
;               option, the second code resets it
;

        DB      Ø               ;fifth flag -- italics
        DW      CODE8
        DW      CODE9
        DW      ITCODE
        DB      Ø               ;sixth flag -- command marker
        DW      CODE1Ø
        DW      CODE1Ø
        DW      SLSHCDE

;
;               *** expanded codes ***
;               the following codes are for the GEMINI 1Ø and 15
;               printers. Change them to match those needed for your
;               printer  (Epson codes are similar, if not identical)
;

CODE1   DB      '\OUT27,45,1\',Ø        ;set underline
CODE2   DB      '\OUT27,45,Ø\',Ø        ;reset underline
CODE3   DB      '\OUT27,69\',Ø          ;set boldface
CODE4   DB      '\OUT27,7Ø\',Ø          ;reset boldface
CODE5   DB      '\OUT27,83,Ø\',Ø        ;set superscript
CODE6   DB      '\OUT27,84,27,72\',Ø    ;reset superscript & subscript
CODE7   DB      '\OUT27,83,1\',Ø        ;set subscript

UNCODE  DB      '\OUT95\',Ø             ;'-'
BFCODE  DB      '\OUT64\',Ø             ;'@'
SUCODE  DB      '\OUT6Ø\',Ø             ;'<'
SBCODE  DB      '\OUT62\',Ø             ;'>'
ITCODE  DB      '\OUT38\',Ø             ;'&'
SLSHCDE DB      '\OUT92\',Ø             ;'\'

;
;               *** add additional codes here  Follow the same format
;               as above, making sure to follow the OUT sequence with
;               a comma zero
;               Like this:

CODE8   DB      '\OUT27,52\',Ø          ;set italics
CODE9   DB      '\OUT27,53\',Ø          ;reset italics
CODE1Ø  DB      '\',Ø                   ;send literal command marker

;
;               *** messages ***
;

MSG1    DB      CR,LF,LF,'The correct syntax for this command is:'
        DB      CR,LF,LF,'       PROCESS d:FILENAME.EXT'
        DB      CR,LF,LF,'which will produce the new file'
        DB      'd:FILENAME.PRC, or'
        DB      CR,LF,LF,'       PROCESS d:OLDFILE.EX1 e:NEWFILE.EX2'
        DB      CR,LF,LF,'which will process d:OLDFILE.EX1 to '
        DB      'produce e:NEWFILE.EX2 '
        DB      CR,LF,'Drive references (d. or e:) are optional '
        DB      CR,LF,'$'
MSG2    DB      CR,LF,'Source file not found!',CR,LF,'$'
MSG3    DB      CR,LF,'Destination file already exists!',CR,LF,'$'
MSG4    DB      CR,LF,'No directory space on disk!',CR,LF,'$'
MSG5    DB      CR,LF,'Disk is full!',CR,LF,'$'
MSG6    DB      CR,LF,'Write protected?',CR,LF,'$'
NORMAL  DB      CR,LF,'Done. No Problems.',CR,LF,'$'

DFCB    DS      9               ;destination FCB
        DB      'PRC'           ;default extension
        DS      21              ;rest of FCB
DFCBCR  EQU     DFCB+32         ;current record
DBUFF   DS      128             ;destination buffer
;
        DS      32              ;local stack
STACK   DW      Ø
;
        END
```

Text Before PROCESSing

```
This is a test. This line is normal.
The second part of this line is @in boldface@.
This line will end __underlined__.
This line is <superscripted<.
This line is >subscripted>.
&This entire line is in italics.&
This line is normal.
Now we will test the new features of our PROCESSor.
This is an ampersand: '&&'
This is an underline: '____'
This is a left arrow: '<<'
This is a right arrow: '>>'
This is an 'at' sign: '@@'
This is a backslash: '\\'
This is normal command which won't appear when printed: \RM80\
What happens when I put three special characters in a row?
&&&This should give me an ampersand followed by two lines in italics,
ending with an  ampersand.&&&
```

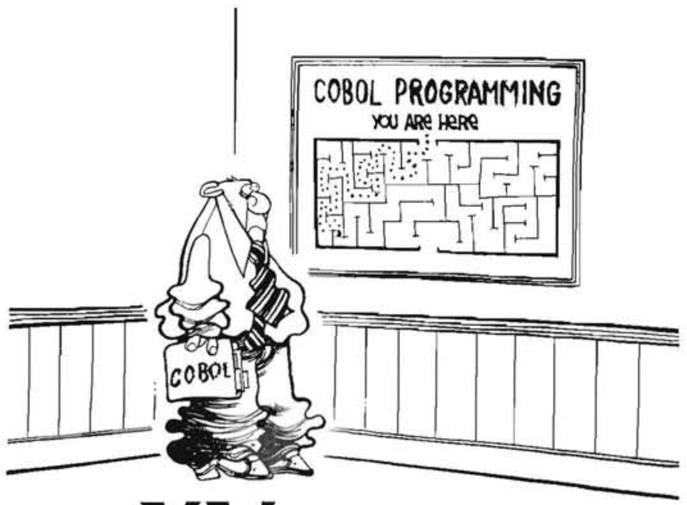
PROCESSor Test

```
This is a test. This line is normal.
The second part of this line is \OUT27,69\in boldface\OUT27,70\.
This line will end \OUT27,45,1\underlined\OUT27,45,0\.
This line is \OUT27,83,0\superscripted\OUT27,84,27,72\.
\OUT27,52\This entire line is in italics.\OUT27,53\
This line is normal.
Now we will test the new features of our PROCESSor.
This is an ampersand: '\OUT38\'
This is an underline: '\OUT95\'
This is a left arrow: '\OUT60\'
This is a right arrow: '\OUT62\'
This is an 'at' sign: '\OUT64\'
This is a backslash: '\OUT92\'
This is normal command which won't appear when printed: \RM80\
What happens when I put three special characters in a row?
\OUT38\\OUT27,52\This should give me an ampersand followed by two lines in italics,
ending with an ampersand.\OUT38\\OUT27,53\
```

Results Of Printing PROCESSor Test

```
This is a test. This line is normal.
The second part of this line is in boldface.
This line will end underlined.
This line is superscripted.
This line is subscripted.
This entire line is in italics.
This line is normal.
Now we will test the new features of our PROCESSor.
This is an ampersand: '&'
This is an underline: '_'
This is a left arrow: '<'
This is a right arrow: '>'
This is an 'at' sign: '@'
This is a backslash: '\'
This is normal command which won't appear when printed:
What happens when I put three special characters in a row?
&This should give me an ampersand followed by two lines in italics, ending
with an ampersand.&
```

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



COBOL Corner XV

Introduction

I hope that you have successfully completed your "single-level control break report" — PRGM07! This month we will work on a "multiple-level control break report" — PRGM08. However, I wish you would not start this program until you have a "good" program #7. You must know how the Program #7 logic works if you are going to follow this program design!

I find that the flowcharts are getting a little out of hand with these longer programs. Have you? If you have, I would like to suggest an alternate for the flowcharts. Many programmers prefer "PSEUDOCODE". I like flowcharts for short programs, but we have about reached their limit. Pseudocode is a "generic" (sounds like a pharmacy) name for an English-like program documentation language. The use of pseudocode allows the programmer to write the program logic in a language that:

1. Programmer can understand as English (no COBOL language is used).
2. Allow the COBOL programming language statements to be written directly from the documentation.
3. Imposes few, if any, syntactical rules on the programmer.

I have put this in the Introduction for two reasons:

1. So that you can look-up "Pseudocode" before we get into the use of it in this article.
2. So that you can be thinking about whether you want to consider pseudocode in place of flowcharts.

Our "multiple-level control break report" will be a reformatted Program #6 Earnings Report. So be sure you have the completed Program #6.

Multiple-Level Control Breaks

When there is more than one control level, the control break logic complexity increases in two areas:

1. The test for control breaks.
2. The need to provide additional control break line printing modules.

Much of the programming specifications, documentation, and coding will be similar to what you used in Program #6 and Program #7. However, rather than just printing a detail-line for each employee as in Program #6, we will print a Department Total Earnings, a Plant Total Earnings, and a Report Total Earnings. This should give you a hint of what the control breaks will be in this program. We will also add a second date to this report. It will be a "period-ending date". Many reports apply to a different date period than the "run-date" but both are important and should be on the report.

As we have done in previous articles, I will supply the Program Specifications as your Program Analysis.

Program #8 Specifications

```
-------------------------------------------------------------
                   PROGRAM SPECIFICATIONS
-------------------------------------------------------------
PROGRAM NAME: DEPARTMENT EARNINGS        PROGRAM ID: PRGM08
              REPORT
-------------------------------------------------------------
```

Program Description:

A Department Earnings Report will be printed from the input file called "PAYROLL". The report will provide control break totals for each department within each plant. The report will also print a report total.

Input File:

Payroll File — FILEL4. (The records will be presorted in ascending sequence according to the following sort keys:

Major Field	Plant Code
Intermediate Field	Department Number
Minor Field	Social Security Number.)

Output File:

Department Earnings Report.

List Of Program Operations:

A. Read each input payroll record.

B. For each record, the program should do the following processing:

   1. Print a detail-line in accordance with the format shown on the Print-Chart.
   2. Accumulate the department earnings this period and the department year to date earnings.

C. When the Department Number within the Plant Code changes, the program will print a control break Department Total Line in accordance with the format shown on the Print-Chart containing the following:

   1. Plant Code.
   2. Department Number.
   3. Department Earnings This Period.
   4. Department Earnings Year To Date.

D. When the Plant-Code changes, the program will print a control break Plant Total Line in accordance with the format shown on the Print Chart containing the following:

   1. Plant-Code.
   2. Plant Earnings This Period.
   3. Plant Year-To-Date Earnings.

E. After all of the input records have been processed, the program should print a Report Total Line in accordance with the format shown on the Print Chart containing the following:

   1. Total Earnings This Period For All Departments.
   2. Year-To-Date Earnings Of All Departments.

F. Headings will be printed at the top of each report page in accordance with the format shown on the Print Chart. After 50 lines have been used on a report page, the program should skip to the top of the next page and print the headings again. Do not allow a control break line to be printed on a report page without at least one detail-line for the control group to which it applies appears.

   1. The Page Number will be incremented each time the headings are reprinted and will be displayed on the second heading line in accordance with the format shown on the Print Chart.
   2. The Period-Ending Date and the Run Date should be obtained from the Working Storage Section. The Period Ending Date will be printed on the first heading line and the Run Date will be printed on the second heading line in accordance with the format shown on the Print Chart.

G. Line-spacing shall be as follows:

   1. The first heading line shall be one inch below the top of the page.
   2. The second heading line shall be single-spaced from the first heading line.
   3. The first column header line shall be double-spaced from the second heading line and the second column header line shall be single-spaced from the first column header line.
   4. The first detail-line after the column headers shall be double-spaced.
   5. The second and successive detail-lines for the same department number will be single-spaced from one another.
   6. Each control break total-line will be double-spaced from the previous line.
   7. The first detail-line following a control break total-line shall be double-spaced from the previous control break total-line.

H. COBOL will be the programming language.

Output Report Line Format

```
 PRINT
POSITIONS   FIELD NAME             COMMENTS
---------   ----------             --------
                      DETAIL LINE
                      ***********
  1-2       FILLER                 PROVIDE LEFT MARGIN.
  3-5       PLANT CODE             3 DIGITS
  6-8       FILLER                 PROVIDE SPACING.
  9-12      DEPT. NBR              4 DIGITS.
 13-15      FILLER                 PROVIDE SPACING.
 16-18      SOC-SEC-NBR            FIRST 3 DIGITS
   19       FILLER                 PROVIDE HYPHEN (-)
 2Ø-21      SOC-SEC-NBR            SECOND 2 DIGITS.
   22       FILLER                 PROVIDE HYPHEN (-).
 23-26      SOC-SEC-NBR            LAST 4 DIGITS
 27-29      FILLER                 PROVIDE SPACING.
 3Ø-47      EMPLOYEE NAME          LAST NAME FIRST
 48-49      FILLER                 PROVIDE SPACING.
 5Ø-59      PERIOD EARNINGS        ZERO-SUPPRESS NON
                                   SIGNIFICANT DOLLAR
                                   POSITION ZEROS.
 6Ø-65      FILLER                 PROVIDE SPACING.
 66-77      YTD EARNINGS           ZERO-SUPPRESS NON
                                   SIGNIFICANT DOLLAR
                                   POSITION ZEROS
 78-132     FILLER                 PROVIDE RIGHT MARGIN

                 DEPARTMENT TOTAL LINE
                 *********************
  1-3       FILLER                 PROVIDE LEFT MARGIN
  4-6       PLANT CODE             3 DIGITS
  7-9       FILLER                 PROVIDE SPACING.
 1Ø-13      DEPT.NBR               4 DIGITS.
 14-18      FILLER                 PROVIDE SPACING.
 19-34      FILLER                 PRINT "DEPARTMENT TOTAL"
 35-46      FILLER                 PROVIDE SPACING.
 47-56      DEPT.PERIOD            ZERO-SUPPRESS NON
            EARNINGS               SIGNIFICANT DOLLAR
                                   POSITION ZEROS INSERT
                                   COMMAS & DECIMAL POINT
 57-58      FILLER                 PRINT " *"
 59-63      FILLER                 PROVIDE SPACING
 64-73      DEPT.YTD EARNINGS      ZERO-SUPPRESS NON
                                   SIGNIFICANT DOLLAR
                                   POSITION ZEROS INSERT
                                   COMMAS & DECIMAL POINT.
 74-75      FILLER                 PRINT " *"
 76-132     FILLER                 PROVIDE RIGHT MARGIN

                    PLANT TOTAL LINE
                    ****************
  1-3       FILLER                 PROVIDE LEFT MARGIN
  4-6       PLANT CODE             3 DIGITS.
  7-18      FILLER                 PROVIDE SPACING
 19-29      FILLER                 PRINT "PLANT TOTAL"
 3Ø-46      FILLER                 PROVIDE SPACING.
 47-56      PLANT PERIOD           ZERO-SUPPRESS NON
            EARNINGS               SIGNIFICANT DOLLAR
                                   POSITION ZEROS INSERT
                                   COMMAS & DECIMAL POINT
 57-59      FILLER                 PRINT " **"
 6Ø-63      FILLER                 PROVIDE SPACING
 64-73      PLANT YTD EARNINGS     ZERO-SUPPRESS NON
                                   SIGNIFICANT DOLLAR
                                   POSITION ZEROS INSERT
                                   COMMAS & DECIMAL POINT
 74-76      FILLER                 PRINT " **"
 77-132     FILLER                 PROVIDE RIGHT MARGIN
```

REPORT TOTAL LINE
****

1-18	FILLER	PROVIDE LEFT MARGIN
19-3Ø	FILLER	PRINT "REPORT TOTAL"
31-46	FILLER	PROVIDE SPACING
47-56	RPT PERIOD EARNINGS	ZERO-SUPPRESS NON SIGNIFICANT DOLLAR POSITION ZEROS INSERT COMMAS & DECIMAL POINT
57-6Ø	FILLER	PRINT " ***".
61-63	FILLER	PROVIDE SPACING
64-73	RPT YTD EARNINGS	ZERO-SUPPRESS NON SIGNIFICANT DOLLAR POSITION ZEROS INSERT COMMAS & DECIMAL POINT
74-77	FILLER	PRINT " ***"
78-132	FILLER	PROVIDE RIGHT MARGIN.

PROGRAM HEADER LINE #1
****

1-2	FILLER	PROVIDE LEFT MARGIN
3-28	PROGRAM NAME	PRINT "DEPARTMENT EARNINGS REPORT"
29-5Ø	FILLER	PROVIDE SPACING
51-7Ø	FILLER	PRINT "PERIOD ENDING DATE:-"
71-72	PER-DATE-MONTH	1 OR 2 DIGITS.
73	FILLER	PRINT SLASH (/)
74-75	PER-DATE-DAY	2 DIGITS.
76	FILLER	PRINT SLASH (/)
77-78	PER-DATE-YEAR	2 DIGITS
79-132	FILLER	PROVIDE RIGHT MARGIN

PROGRAM HEADER LINE #2
****

1-2	FILLER	PROVIDE LEFT MARGIN
3-7	FILLER	PRINT "PAGE:".
8-1Ø	PAGE NBR	ZERO-SUPPRESS NON SIGNIFICANT POSITION ZEROS.
11-6Ø	FILLER	PROVIDE SPACING
61-7Ø	FILLER	PRINT "RUN DATE:-".
71-72	RUN DATE MONTH	1 OR 2 DIGITS.
73	FILLER	PRINT SLASH (/)
74-75	RUN DATE DAY	2 DIGITS.
76	FILLER	PRINT SLASH (/)
77-78	RUN DATE YEAR	2 DIGITS.
79-132	FILLER	PROVIDE RIGHT MARGIN.

COLUMN HEADER LINE #1
****

1-2	FILLER	PROVIDE LEFT MARGIN
3-7	FILLER	PRINT "PLANT"
8-9	FILLER	PROVIDE SPACING.
1Ø-14	FILLER	PRINT "DEPT "
15-16	FILLER	PROVIDE SPACING.
17-27	FILLER	PRINT "SOCIAL SEC."
28-49	FILLER	PROVIDE SPACING
5Ø-57	FILLER	PRINT "EARNINGS".
58-63	FILLER	PROVIDE SPACING
64-75	FILLER	PRINT "YEAR TO DATE"
76-132	FILLER	PROVIDE RIGHT MARGIN

COLUMN HEADER LINE #2
****

1-3	FILLER	PROVIDE LEFT MARGIN.
4-7	FILLER	PRINT "CODE"
8-1Ø	FILLER	PROVIDE SPACING
11-13	FILLER	PRINT "NBR"
14-18	FILLER	PROVIDE SPACING.
19-24	FILLER	PRINT "NUMBER".
25-32	FILLER	PROVIDE SPACING.
33-45	FILLER	PRINT "EMPLOYEE NAME".
46-51	FILLER	PROVIDE SPACING.
52-6Ø	FILLER	PRINT "THIS PER."
61-67	FILLER	PROVIDE SPACING.
68-75	FILLER	PRINT "EARNINGS"
76-132	FILLER	PROVIDE RIGHT MARGIN.

Input Record Line

FIELD POSITION	FIELD NAME	DATA CLASS	COMMENTS
1-2	FILLER		SKIP CODE "L4".
3-11	SOC-SEC- NBR	NUMERIC	9 DIGITS
12-29	EMPLOYEE NAME	ALPHANUMERIC	LAST NAME FIRST
3Ø-49	FILLER		SKIP NON-USED DATA
5Ø-56	PERIOD EARNINGS	NUMERIC	7 DIGITS.
57-59	PLANT CODE	ALPHANUMERIC	3 DIGITS.
6Ø-63	DEPT.NBR	ALPHANUMERIC	4 DIGITS.
64-71	FILLER		SKIP NON-USED DATA.
72-79	YTD EARNINGS	NUMERIC	8 DIGITS.
8Ø	FILLER		NOT USED.

Documentation

With the Program Specifications, along with the Output and Input Layouts I have provided above, you readers are now ready to prepare your Print Chart, Record Chart, and Structure Chart. Please do these now. Notice that the charts are similar to the ones that you did for Program #7. I would like to have you notice that the Department Total, Plant Total, and Report Total Lines are arranged in major to minor sequence for your print chart. This usually makes reports easier to read.

If you are going to do a Flowchart for this program, it would require additional modules and additional logic to test for the multiple control breaks.

The program coding, except for the comment paragraph, for the IDENTIFICATION DIVISION will be nearly the same as you have just completed and I will let you code these. Also, the FILE SECTION of the DATA DIVISION will be very similar to your Program #6 and #7. I would like you to code these. I would like you to try coding the Working Storage Section. You will need to add to the control-fields, work area, accumulator area, header lines, and total lines. Try it! You will not find it difficult with the Program Instructions I have provided above.

Pseudocode Vs Flowchart

I am going to do the pseudocode for this program so that you will know how to do it for future programs and to help you decide if you like pseudocode better than flowcharts. Flowcharts are getting large and complex. You really need a large sheet of paper or a lot of "continues" to handle all of the flowchart "boxes". I think they are getting out of hand! So, that is the "why" of pseudocode. Remember, you should and MUST do one or the other if you are going to understand and be able to create a Logical Program!

The following Pseudocode is ONE of many ways. I will show you another method in a later program:

```
-----------------------------------------------------------
                         PSEUDOCODE
-----------------------------------------------------------
PROGRAM NAME: DEPARTMENT EARNINGS        PROGRAM ID: PRGMØ8
              REPORT
-----------------------------------------------------------

ØØØ-PRINT-EARNINGS-REPORT MODULE
--------------------------

1. Open the two files.
2  Perform 1ØØ-INITIALIZE-VARIABLE-FIELDS.
3  Perform 2ØØ-PROCESS-PAYROLL RECORDS until no more records
4. Close the two files.
5. Stop the run

1ØØ-INITIALIZE-VARIABLE-FIELDS MODULE
-------------------------------

1. Set the End-of-File indicator to NO
```

```
2  Set the First Record indicator to YES
3. Set the Total Accumulator fields to ZERO.
4. MOVE Period Date and Run Date to headings

200-PROCESS-PAYROLL-RECORDS MODULE
----------------------------
1  Perform 800-READ-PAYROLL-RECORD
2  IF this is the first record
      MOVE the input Plant-Code to the
      Previous Plant-Code field
      MOVE the input Dept-NBR to the
      Previous Dept-NBR field
      Set the First Record indicator to NO
3. IF input Plant-Code is not the same as Previous
      Plant-Code
      Perform 220-PRINT-DEPT-TOTAL-LINE
      Perform 230-PRINT-PLANT-TOTAL-LINE
   ELSE
     IF input Dept-NBR is not the same as Previous Dept-NBR
        Perform 220-PRINT-DEPT-TOTAL-LINE.
4  IF there is an input Payroll Record
   Perform 210-PRINT-DETAIL-LINE

210-PRINT-DETAIL-LINE MODULE
----------------------
1. IF this is the FIRST PAGE OR FULL PAGE of the report
   Perform 870-PRINT-REPORT-HEADING.
2  MOVE input Plant-Code field to detail-line
   Plant-Code field
3. MOVE input Dept-NBR field to detail-line Dept-NBR field
4  MOVE input Social Security NBR field to the detail-line
   Social Security NBR field   (Use three components
   separated by hyphens on detail-line.)
5  MOVE input EMPLOYEE NAME field to the detail-line
   EMPLOYEE NAME field
6  MOVE input Period Earnings field to the detail-line
   Period Earnings field.
7  MOVE input Year To Date Earnings field to the
   detail-line Year To Date Earnings field
8  Move the detail-line to the output Earnings Report Line.
9  Perform 890-PRINT-REPORT-LINE.
10  Add the input Period Earnings to the Department Period
    Earnings accumulator total
11  Add the input Year To Date Earnings to the Department
    YTD Earnings accumulator total
12  Set the line-spacing indicator for single-spacing

230-PRINT-PLANT-TOTAL-LINE MODULE
--------------------------
1  MOVE the Previous Plant-Code to the control break
   Department Total Line Plant-Code field.
2  MOVE the Previous Dept-NBR to the control break
   Department Total Line Dept-NBR field
3  MOVE the Department Period Earnings accumulator total
   to the control break Department Total Line
   Dept-Period-Earnings field.
4. MOVE the Department YTD Earnings accumulator total
   to the control break Department Total Line
   Dept-YTD-Earnings field
5  MOVE the control break Department Total Line to
   the output Earnings Report Line.
6. Set the line-spacing indicator for double-spacing.
7  Perform 890-PRINT-REPORT-LINE
8. Add the Department Period Earnings accumulator total
   to the Plant Period Earnings accumulator total
9. Add the Department YTD Earnings accumulator total to
   the Plant YTD Earnings accumulator total
10. MOVE ZEROS into the Department Period Earnings
    accumulator
11  MOVE ZEROS into the Department YTD Earnings
    accumulator
12  MOVE the "new" input Dept-NBR into the Previous
    Dept-NBR field

230-PRINT-PLANT-TOTAL-LINE MODULE
-------------------------
1. MOVE Previous Plant-Code to the control break Plant
   Total Line Plant-Code field.
2  MOVE Plant Period Earnings accumulator total to control
   break Plant Total Line Plant Period Earnings field.
3  MOVE Plant YTD Earnings accumulator total to control
   break Plant Total Line Plant YTD Earnings field
4  MOVE the control break Plant Total Line to the output
   Earnings Report Line
5  Perform 890-PRINT-REPORT-LINE
6  Add the Plant Period Earnings accumulator total to
   the Report Period Earnings accumulator total
7  Add the Plant YTD Earnings accumulator total to the
   Report YTD Earnings accumulator total
8  MOVE ZEROS into Plant Period Earnings accumulator total
9  MOVE ZEROS into Plant YTD Earnings accumulator total
10. MOVE the "new" input Plant-Code into the Previous
    Plant-Code field.

700-PRINT-REPORT-TOTAL-LINE MODULE
---------------------------
1  MOVE the Report Period Earnings accumulator total to
   the Report Total Line Report Period Earnings field.
2. MOVE the Report YTD Earnings accumulator total to
   the Report Total Line Report YTD Earnings field
3. MOVE the Report Total Line to the output Earnings
   Report Line
4. Perform 890-PRINT-REPORT-LINE.

800-READ-PAYROLL-RECORD MODULE
----------------------
1  Read the input Payroll Record file for a record
2  IF there are no more records
   MOVE YES to the End Of File indicator
   MOVE HIGH-VALUE to the input record Plant-Code
   and Dept-NBR

870-PRINT-REPORT-HEADING MODULE
-----------------------
1  MOVE SPACES to the output Earnings Report Line
2  Perform 880-PRINT-TOP-LINE
3. Set line-spacing indicator to 6 for a 1 inch top margin
4. MOVE the Program First Heading Line to the output
   Earnings Report Line
5. Perform 890-PRINT-REPORT-LINE.
6. MOVE the Page Count to the second Heading Line Page
   Number field.
7  Add 1 to the Page Count
8. MOVE the Program Second Heading Line to the output
   Earnings Report Line
9. Perform 890-PRINT-REPORT-LINE
10. Set the line-spacing indicator for double-spacing.
11  MOVE the First Column Header Line to the output
    Earnings Report Line
12  Perform 890-PRINT-REPORT-LINE
13  Set the line-spacing indicator for single-spacing
14  MOVE the Second Column Header Line to the output
    Earnings Report Line
15  Perform 890-PRINT-REPORT-LINE
16. Set the line-spacing indicator for double-spacing

880-PRINT-TOP-LINE MODULE
-----------------
1. Advance to the top of the next report page and
   write the output Earnings Report Line
2  Set Lines Used to ZERO

890-PRINT-REPORT-LINE MODULE
--------------------
1  Display the output Earnings Report Line on the video
   screen for a visual check of the program
2  Advance in accordance with the setting of the line
   spacing indicator and write the output Earnings
   Report Line
3. Add the line-spacing indicator value to the Lines Used
```

Procedure Division Coding

With pseudocode you will find that you can code the Procedure Division very easily. COBOL is an English-Like language and pseudocode is English-Like oriented, so you need to translate the pseudocode to COBOL syntax. I am sure that you will find it easy.

Did you find the "nested IF" in this program? It is in Module 200. Did you find it to be different from the "nested IF" in Program #7? I used a "linear" IF statement because it is easier to understand. I tried to show some proper indentation in the pseudocode so that it would be easier to find and understand. For a "multiple-level control break report" a proper "nested IF" must be established and the control totals must be performed in the proper sequence!

The tricky part of establishing the control break test is the determination of which control-field should be tested for first. The rule is simple. Always test the major field first and then proceed by testing each field in descending significance down through the minor field. Although the control break is done in major to minor sequence, the printing of control breaks is performed in minor to major sequence. Each of the control breaks, starting with the minor field, should be performed up to the level of the control break detected. REFER to your Structure Chart to make this clear! For example, when the first control break has occurred, we performed the 220 Module and THEN the 230 Module. Do you see this? Your Structure Chart must be used to clearly show this!

Again in the program, you should notice that I used HIGH-VALUES in the 800 Module for each of the control fields to trigger the control breaks for the LAST record. Remember the two problem areas we stated in Program #7? This solves the second problem — force the last control break after all records have been Read!

I would like to restate for a summary two items:

1. When specifying the fields used to hold the control-fields of the Previous record, it is logical to list them in major through minor sequence.
2. When specifying control-field accumulator fields, it is logical to list them in minor through major sequence.

Closing

I have left a lot of coding for you readers to do. If you feel that I am expecting you to do more than you feel you are ready for, please let me know! I would like details as to where you feel I have not made any item in this program or previous programs clear. Remember, if you write to me and you wish a written answer from me please include a self-addressed, stamped business-size envelope with your letter! Be sure to supply plenty of details and examples of what is not clear to you.

After you have coded, COMPILED, and LINKED/EXECUTED Program #8, you can compare your program with mine by using the "HUG COBOL Corner Disk-II" as we have explained many times.

We have covered in great detail how to produce a really useful and tasteful looking Report. At least I think so! Do you? So, we are now ready to get into some really useful programming! Next month I will start Table Handling. COBOL has many ways of dealing with Tables. I am sure that you will find this programming a challenge and interesting. I will try to make it seem easy!

*

$ FINANCIAL TOOLS $
FOR YOUR HEATH/ZENITH COMPUTER

★ **LOAN-MASTER™ $49.95**
Flexible loan amortization schedules provide you with the information you need to select the best terms for home and auto loans.

★ **INVESTMENT-MASTER™ $49.95**
Provides lump sum and annuity figures that help you manage periodic savings plans, mutual funds, and IRAs.

★ **STOCK-MASTER™ $49.95**
Stock market investment aid. You benefit by knowing when to buy or sell and how much to buy/sell.

★ **DEPRECIATION-MASTER™ $249.00**
Comprehensive fixed-assets analysis system. Allows for book and tax accounting of your business assets. Requires Microsoft BASIC Interpreter.

Dealer Inquiries Invited ★ Quantity Discounts Available
ORDER NOW . . . CLIP THE COUPON AND SEND

20% **SALE COUPON** 20%
20% OFF LIST PRICE FOR
ANY PRODUCT LISTED ABOVE
VERSIONS AVAILABLE FOR CP/M-80, ZDOS, and PC-DOS

Call or Write for More Information or a FREE 1985 Product Catalogue

NAME ____________
ADDRESS ____________
CITY ________ STATE ____ ZIP ____
PHONE (____) ____________

FOR FASTER DELIVERY CALL
906-249-9801

COMPANY ____________
☐ CHECK ENCLOSED ☐ Send COD (Add $2.00)
☐ COMPANY P.O. # ____________
☐ VISA ☐ MasterCard Account #____________ Exp. Date______
SIGNATURE ____________
Add $3.00 shipping/handling for all orders and 4% sales tax for Michigan residents.
Send to: **GENERIC COMPUTER PRODUCTS, INC., POB 790, Dept. 115R Marquette, Michigan 49855**

ZENITH/HEATH 161 PORTABLE COMPUTER ACCESSORIES

WEATHERPROOF VINYL COVER **$29.95**

PROTECT YOUR PORTABLE COMPUTER FROM RAIN, SNOW, AND DIRT WITH THIS BEAUTIFULLY CUSTOM DESIGNED SLIP-OVER COVER CONSTRUCTED OF HIGH QUALITY BLACK VINYL. ALSO CAN SERVE AS A DUST COVER WHEN COMPUTER IS NOT BEING USED

PROTECTIVE BACK PANEL **$9.95**

ELIMINATE CONCERNS OF DAMAGING YOUR COMPUTER CONNECTOR PANEL WITH THIS EASY TO INSERT AND REMOVE PROTECTIVE BACK PANEL CONSTRUCTED OF A TOUGH ACRYLIC POLYVINYL MATERIAL

SPECIAL: **$34.95** BOTH VINYL COVER AND BACK PANEL

EDL PRODUCTS & SERVICES
453 S McCARRONS BLVD·ROSEVILLE·MN·55113
(612)-489-0851

PLEASE SEND CHECK OR M O
(ADD $2.50 FOR SHIPPING)

☐ WEATHERPROOF VINYL COVER - $29.95 ☐ PROTECTIVE BACK PANEL - $9.95
☐ WEATHERPROOF VINYL COVER AND PROTECTIVE BACK PANEL - $34.95

NAME ____________
ADDRESS ____________
CITY ________ STATE ________ ZIP ________

Your Choice of IBM-PC Compatibles Now at an unbelievable $1299

ZVM-122A Amber Monitor shown but not included in price.

DESKTOP $1299

Heathkit HS-151-21 IBM-PC compatible computer kit with 128K memory, expandable to 640K

TRANSPORTABLE $1299

Heath HS-161-21 IBM-PC compatible computer kit with 128K memory, expandable to 640K

Both computers feature: • MS-DOS – full IBM-PC Compatibility • Single 360K Disk Drive • Free Microsoft WORD and MULTIPLAN Software • Famous Heath/Zenith quality • Complete user training available • Full service backup • **Second disk drive only $149** (Winchester also available) • **Expand to 640 RAM for $199**

★ ***HUG special – choose either kit or assembled transportable for only $1299.*** ★

CHECK THESE SPECIAL BUYS **Lotus 1-2-3 spreadsheet, graphics, and data base. Easy-to-use business package provides you with information management, spreadsheet analysis, and business graphics – all in 1 program. Only $299.**

New Epson LX-80 PRINTER

Now only **$249.95**

including parallel cable

Correspondence quality FRICTION FEED printer prints directly on your letterhead. Fast – over 100 characters per second. Select type styles at a touch – over 150 combinations – Mix text, graphics on same line. 1-Year Warranty.

SYQUEST 10 MEG REMOVABLE WINCHESTER CARTRIDGE DRIVES

Includes drive, one 10 Meg cartridge, controller board

SPECIAL $1195.00
Reg. $1395

Free installation

HEATHKIT ADVANCED SYSTEM 3 25-INCH COLOR TV WITH REMOTE CONTROL

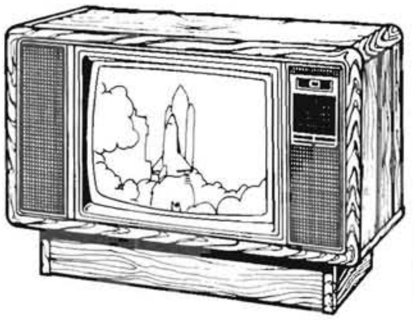
Includes choice of Mediterranean or modern cabinet. Equipped with stereo audio system, quartz-controlled electronic tuning.

Only **$499.95**

Reg. $649.95 Value

Save $150

Incredible price!



Quantities Limited.

HUG SPECIALS

U.S. ROBOTICS PASSWORD MODEM
300/1200 Baud Auto Answer Modem. Cable included.
Reg. $269 . . . Now Just **$199.00**

BORLAND SIDE KICK and SUPERKEY SOFTWARE
For IBM-PC and Compatibles. Each . Now Just **$49.95**

DISKHOLDER with 10 DSDD DISKETTES
Come and get them while they last. Special . **$12.95**

PACKAGE OF 10 NASHUA DSDD DISKETTES
Life-time warranty. . **$14.95**

PAL* LAPTOP COMPUTER

7½ lb. briefcase size unit has Microsoft WORKS, WORD, MULTIPLAN and BASIC...Modem...Calendar...Alarm...all built in. Much more, too. Act now for this unbelievable price.

$995.00

*Personal Advanced Laptop

Inquire about special corporate and government sales and service programs.

Experts in **CAD**, Networking and Communications.

Phone orders accepted.
Prices good through December 7, 1985.

ZENITH data systems Authorized Dealer

Your Total Service computer center

Service • Support • Software • Accessories • User Training • Competitive Prices

HZC-151

Heath ZENITH®

Units of Veritechnology Electronics Corporation

Computers & Electronics

The FORTRAN Formula-5

Dick Stanley
P. O. Box 9512
Alexandria, VA 22304

In the last article of this series, we discussed program control statements and mixed-mode calculations. We also looked briefly into disk file operations and arrays. This time, we will investigate how to use arrays to hold and rearrange data, and how to calculate moving averages using array techniques. To save doing everything twice, we will then write our results on a disk file that can be retrieved for the next part of this project.

The Moving Average Moves...

Remember from the earlier installments that a moving average is formed by adding together the last so-and-so many days' data and dividing by the number of days we have chosen to average. Thus, each day we add one number to the list of items to be added, and we also drop one number—the oldest one in the series.

Let's be certain we understand this concept. Here is a list of hypothetical stock prices for ten days:

June 3	33 1/4
June 4	35
June 5	34 7/8
June 6	35 1/4
June 7	36 1/2
June 10	36
June 11	36 7/8
June 12	36 3/8
June 13	37 1/8
June 14	35 7/8

To form the ten-day moving average of the stock price, add the above ten prices (the total is 357 1/8) and divide by ten. That gives a ten-day moving average for the 14th of June of 35.7125.

OK, that was easy. Now, let's say that the stock closes on June 17 at 37 1/4. What is the moving average on June 17th? We find it just as we did for the 14th: add up the prices for the last ten days and divide by ten. But now the data has changed. Our data list now looks like this:

June 3	33 1/4 <=== this entry is dropped
June 4	35
June 5	34 7/8
June 6	35 1/4
June 7	36 1/2
June 10	36
June 11	36 7/8
June 12	36 3/8
June 13	37 1/8
June 14	35 7/8
June 17	37 1/4 <=== this entry (the new day) is added

If we now add up the ten days between June 4 and June 17, we find the new total is 361 1/8, and thus the ten-day moving average is 36.1125.

That wasn't hard, was it? Now, how can we do that in our program? It is apparent that we only need to enter each data item once, so we don't want to do anything that will require repetitive entries. That is both wasteful and error-prone. We need to provide a capability to drop the oldest data item and to add in the newest, and then to add all the items and divide by ten. Because we must continually keep dropping the oldest item, we cannot merely keep a running total in memory; we need to know which item is associated with each date.

Also, we might want to calculate moving averages of more than one duration. So we don't want to drop any data except at the end of the moving average term of the longest duration. If we were calculating averages of 5, 10, and 30 days duration, we would want to drop only the 31st day's data when each new day is added.

We could just keep adding the new day to the bottom of the pile, and then counting back the required number of days, but we would find that memory is rapidly exhausted. A better approach is to define an array large enough to hold the required number of data points, and then find a way to throw out the oldest item every time a new item is added.

Let's consider the following program segment:

```
      DIMENSION PRICE(10)
             .
             .
             .
      TOTAL=Ø
      DO 1ØØ J=1,1Ø
  1ØØ TOTAL=TOTAL+PRICE(J)
      AVG1Ø=TOTAL/1Ø.Ø
```

Each time we enter the loop above, the running TOTAL is initialized to zero, the ten values held in array positions PRICE(1) to PRICE(10) are added together, and the final total is divided by ten to form the ten-day moving average, AVG10. The problem of how to throw out the oldest data remains, however; if we do this loop ten times, we will get the same answer ten times unless something changes.

There are several ways to solve this problem. Let's use a "brute-force" approach that is simple and easy to understand. First, some definitions:

```
PRICE(1) = today's price
PRICE(2) = yesterday's price
      .
      .
      .
PRICE(1Ø)= price 1Ø days ago
```

Now, if we can rearrange the array each day so that all the items move down one position, we will have succeeded in building an array that contains only the ten most recent data items, and we can then add them and divide by ten to get the current day's moving average. The program segment below will accomplish this switching.

```
    DIMENSION PRICE(1Ø)
          .
          .
          .
    TOTAL=Ø
    DO 1ØØ J=1,9
    J1=11-J
    J2=1Ø-J
1ØØ PRICE(J1)=PRICE(J2)
          .
          .     <== somewhere in here today's price is gotten
          .             and stored in PRICE(1)
    DO 1Ø1 J=1,1Ø
1Ø1 TOTAL=TOTAL+PRICE(J)
    AVG1Ø=TOTAL/1Ø.Ø
```

Look closely at the loop ending with statement 100. This loop moves all the array members up one position to make room for the new price in PRICE(1). The first time through the loop, J=1, so the replacement is made of PRICE(10)=PRICE(9). The second time, J=2, so PRICE(9)=PRICE(8). This procedure continues until the last pass, when J=9 and the swap is made of PRICE(2)= PRICE(1). When we now put today's price in PRICE(1), we have updated the array and can calculate the moving average as we did above. This time the answer will be correct every day.

There is a reason for defining J1 and J2. An array subscript can have the form of VARIABLE-CONSTANT, but it cannot have the form CONSTANT-VARIABLE. By defining the two dummy variables J1 and J2, we "fool" the system into calculating what we require, and avoid this restriction on what may be within the parentheses of an array subscript.

Why did we do the array rearrangement from the top? Wouldn't it have been simpler to say PRICE(J+1)=PRICE(J)? Well, yes, it would have been. But let's look at what would have happened if we had done that. On the first pass, we would have written the value of PRICE(1) into PRICE(2), which is just what we want to do. On the second pass, we would write PRICE(2) into PRICE(3). That isn't what we want, because the old value of PRICE(2) that we really want to move into PRICE(3) was overwritten during the first pass by the value of PRICE(1). If we continue this process, we will fill the array with the value of PRICE(1), which will not be of much help to us. By swapping data items beginning with the highest numbered ones, no data item is destroyed before it has been moved to its new location. Arrays give us powerful capabilities for data manipulation, but caution is necessary to avoid doing strange things to the data. One can readily see here the possibilities to prove the old adage, "To err is human, but to really foul things up, you need a computer."

Why didn't we then use negative increment on the DO loop counter, to count from 10 down to 2? Because that isn't an option in FORTRAN-66! That is not a serious problem, because we can always simulate a negative increment as I have shown you above. However, be aware that the older versions of FORTRAN do not like to count backwards. Some FORTRAN-77 compilers allow both negative and fractional increments, but we are limiting ourselves to FORTRAN-66. If it works there, it will work nearly anywhere!

One advantage to the way FORTRAN numbers arrays is that to form a moving average of N items, you simply add up the array items numbered 1 through N. For a 10-day moving average of FOO, the last item is FOO(10); for a 25-day moving average, the last item is FOO(25), and so on.

Getting Started

Before we can use the data stored in an array to calculate anything, we must first get it into the array. Remember that we wanted to calculate some moving averages for periods of up to 60 days, so we will need to keep that many days' data in our arrays. We will want to modify our program so that it makes it easy for us to enter more than one day's data at a time, and so that it keeps the data we have collected on a disk. We don't want to enter it each time we run the program. After all, computers are supposed to save us work, not create more of it.

The easiest way to facilitate repeated entries is to ask the operator at the end of each day's input whether or not to go around again, or to quit now. If the operator selects to enter another day's data, the program must save the data from today in a suitable fashion so that it isn't lost and so that it is still available for calculating averages, printing graphs, etc. To be on the safe side, it would be prudent to keep all of the input data for each day until we are certain that day's data is no longer needed (such as at the end of our 60-day averaging period).

Several approaches exist to gracefully present the calculated results and then give the operator a chance to continue. We could run through a timing loop and then continue automatically, or we could ask for a response from the operator after showing the results of today's calculations. For now, let's do the latter. It can be very frustrating to have the screen you are looking at disappear just as you are zeroing in on a specific item presented on it.

Data Validation

One thing that is often taken for granted is that the operator will enter the correct data every time. Trust me, he won't! If a mistake can be made, we'll make it, especially at 1:00 in the morning. Programs must be designed to be "bullet-proof" so that entry of an incorrect value will not cause catastrophic results.

Let's examine this requirement in light of our desire to query the operator as to what to do next. One easy way to do this is to offer the operator a menu with items numbered. Then we can check for the correct range of numbers. If the answer isn't in the range, we ask again and we keep asking until a correct response is given. If a correct response is given, we go and do what the operator wanted done. Figure 1 shows an example of data validation using FORTRAN.

```
C  Routine to ask operator what to do next
C  Set up formats for I/O
500       FORMAT('ØYou may now do one of the following:')
501       FORMAT('        1   Enter more data')
502       FORMAT('        2   Quit and save data to disk')
503       FORMAT('ØEnter the number of your choice, then <CR>  ')
504       FORMAT(I2)
C  Print message to operator
602       WRITE(1,500)
          WRITE(1,501)
          WRITE(1,502)
          WRITE(1,503)
C  Get operator's reply
          READ(1,504)IANS
C  If IANS=1, then go to 604, which re-validates the entry and
C       goes to the start of input routines
C  If IANS=2, then go to 601, which saves data and quits
C  If IANS is not 1 or 2, ask for input again
          IF(IANS.EQ.2) GO TO 601
          IF(IANS.EQ.1) GO TO 604
          GO TO 602
```

Figure 1. Data Validation example using FORTRAN.

The code above shows basic data validation. I have used the logical IF statement together with the logical operator .EQ. which means "equals" to test for the proper values of the response. We need not concern ourselves with values such as "Q" or "3.5", because we have set the input field to Integer type. If a value such as that is entered, the program will raise an error. Unfortunately, it will then return to the operating system prompt. Until we become more skilled at data validation, it is important to be careful!

What are those zeros that are inserted at the beginning of the text to be printed in statements 500 and 503? These are called "carriage control" characters. The name dates again from the earlier days of computing, when high-speed printers contained special control tapes to position the paper at the proper place for printing. In FORTRAN, the first character on each line is "gobbled up" by the system; whether you intend it or not, the first character is routed to an internal routine that decides where to print the next line. In the past version of STOXIN, you noticed that we used blanks and plus signs to do certain things. Now is as good a time as any to introduce you to the full set of carriage control characters. When printed as the very first character of a line, the indicated characters will do the following:

```
' ' = Write on the next line (this is a blank character)
'Ø' = Skip one line, then write (i.e. advance 2 lines)
'1' = Skip to top of next page (i.e  form feed)
'+' = Overprint next line, do not advance the paper/cursor
```

It is important to understand that FORTRAN regards the first character of every line as a carriage control character, regardless of whether you meant it that way. The following code sequence

```
1Ø     FORMAT(I1)
       J=1
       WRITE(2,1Ø)
```

will cause Logical Unit 2 to skip to the top of the next page. Nothing visible will be printed. The value of J was understood by the system as a carriage control character (which was, in this case, a form feed). Carriage control characters are never visibly printed, so errors resulting from carelessness in this area are often hard to find.

Saving Some Work

How do we go about saving all that data once we have finished entering it? Refer back to the last article, and review the parts about writing files to disks. In this example, we shall assume that the data files are to be written on drive B: (drive 2 to FORTRAN), and that the program resides on drive A:. To save the data, we loop through the arrays containing the data, and write a record for each array position, as shown in Figure 2.

```
601       CALL OPEN(6,'STOCK   DAT',2)
625       FORMAT(3I2,F8.2,I1Ø,3I5,2I1Ø,2I5,I1Ø,2F8.2,2I1Ø,F8.2,I5)
          DO 62Ø J=1,6Ø
          WRITE(6) MO(J), DAY(J), YR(J), DJIA(J),
        1   VOLUME(J),ADVNC(J),DECLN(J),UNCHNG(J),UPVOL(J),DNVOL(J),
        2   HIGHS(J),LOWS(J),ADLINE(J),DJIPCT(J),ADPCT(J),ADVVOL(J),
        3   DECVOL(J),TRIN(J),HLDIF(J)
62Ø       CONTINUE
          ENDFILE 6
```

Figure 2. Saving array data to disk.

There are two things to notice from Figure 2. First, the file name of the disk file has been padded with blanks to be exactly eleven spaces long (8 for the filename and 3 for the extension); this is important. If you don't write the filename in this fashion, strange things happen. Also, there is no FORMAT statement to show how to write the data onto the disk. The form of the WRITE statement shown above is called an unformatted WRITE, and writes a memory image of the data onto the disk. It must be read with its counterpart, the unformatted READ.

Swapping It All Around

There are still a few loose ends. If the operator chooses to go back and enter another day's data, we must update our arrays to get the right items in the right positions. Otherwise, the new inputs will simply overwrite the data just entered, and things will rapidly be fouled up. Also, our original listing of STOXIN.FOR did not dimension most of the variables, so we must do that if we are to store these things in arrays. To get the variable space we need, we can use the declaration statements in Figure 3. Because some variables that were not subscripted before now are, and some require more space than before, these declarations are more comprehensive than those in the previous version of this program.

```
C         Declare variables
          INTEGER  MO(6Ø),DAY(6Ø),YR(6Ø),ADVNC(6Ø),DECLN(6Ø),
        1   UNCHNG(6Ø),HIGHS(6Ø),LOWS(6Ø),HLDIF(6Ø),UPVOL(6Ø),
        2   DNVOL(6Ø),ESC,CLRSCN
          INTEGER*4  VOLUME(6Ø),ADVVOL(6Ø),DECVOL(6Ø),ADLINE(6Ø)
          REAL DJIA(6Ø),DJIPCT(6Ø),ADPCT(6Ø),TRIN(6Ø)
```

Figure 3. Data declarations needed for 60-position arrays.

The integer variables ESC and CLRSCN are not array variables, but were declared above so that when these statements are substituted in the original program, they will still be properly declared. Notice that we have used the continuation facility of FORTRAN here and above to make a longer logical line than we can have as a physical line. Any non-blank character would do in column 6 to denote a line as a continuation line, but it is handy to use numbers to show which level of continuation you are at. Also, indenting the continued lines a space or two improves readability of the code. FORTRAN will ignore the spaces in the code when it reads it.

Now we must deal with swapping the array elements so that the newest data is always in position 1, yesterday's is always in position 2, and so on. The way to accomplish this is to use the approach described above, and reassign the elements of each array as we loop through. The code in Figure 4 will accomplish the desired rearrangement:

```
C   This code rearranges array elements and then goes to the
C         data input routines.
604      DO 630 J=1,59
                  J1=61-J
                  J2=60-J
                  MO(J1)=MO(J2)
                  DAY(J1)=DAY(J2)
                  YR(J1)=YR(J2)
                  DJIA(J1)=DJIA(J2)
                  VOLUME(J1)=VOLUME(J2)
                  ADVNC(J1)=ADVNC(J2)
                  DECLN(J1)=DECLN(J2)
                  UNCHNG(J1)=UNCHNG(J2)
                  UPVOL(J1)=UPVOL(J2)
                  DNVOL(J1)=DNVOL(J2)
                  HIGHS(J1)=HIGHS(J2)
                  LOWS(J1)=LOWS(J2)
                  ADLINE(J1)=ADLINE(J2)
                  DJIPCT(J1)=DJIPCT(J2)
                  ADPCT(J1)=ADPCT(J2)
                  ADVVOL(J1)=ADVVOL(J2)
                  DECVOL(J1)=DECVOL(J2)
                  TRIN(J1)=TRIN(J2)
                  HLDIF(J1)=HLDIF(J2)
630      CONTINUE
```

Figure 4. Code to rearrange array data.

When you think about it, it becomes apparent that we should always rearrange the data in the array before we enter any new data in it. So, the loop above should be executed every time we set out to enter data, and not just when the operator asks for another pass through the program after entering data. If we allow data to be entered without first rearranging the data, we will overwrite the old data in the "today" position (array element number 1), and the validity of the data will be destroyed. Therefore, this code should go near the front of the program, where it will execute before any new data is entered.

You will also see that the loop in Figure 4 ends with a statement we haven't seen before, the CONTINUE statement. This is a "dummy" statement whose entire function in life is to terminate a DO loop with a "legal" statement. DO loops, you remember, can't end with a GO TO or a DO statement. The CONTINUE statement was originally devised to provide a way to end a DO loop properly when the last statement would otherwise be one not permitted. It doesn't hurt to end DO loops with CONTINUE as a general rule, however. When you do this, the limits of the loop are clearly defined. You can also indent the statements within the range of the loop, and just generally make the code more readable and structured.

Putting It All Together

We can integrate the code segments above into our STOXIN.FOR source code, and we will then be able to enter more than one day's data at a time and have our results saved to disk when we finally exit the program. The values for each day will be displayed on the screen as we enter that day's data. So far, the code to calculate moving averages in this program hasn't been presented. Since we are going to calculate quite a number of moving averages, we shall devote some attention to this whole area in the next installment.

To create the modified STOXIN program, modify the STOXIN.FOR code shown in the last article (THE FORTRAN FORMULA-4, REMark, May, 1985) as follows:

1. Replace the four lines of data declarations at the beginning of the program with the code found in Figure 3.
2. Insert the code of Figure 4 after the statement CLRSCN=69 and before the statement WRITE(1,130) ESC,CLRSCN in the Clear Screen routine.
3. Insert the code of Figure 1 immediately before the END statement.
4. Insert the code of Figure 2 immediately after the Figure 1 code you just inserted, and immediately before the END statement.
5. Since many variables have been replaced by subscripted variables, all the array variables must have subscripts added to them.
6. Remove the code that simulates the value of DJIA(2).
7. Recompile the program and run it.

If you have trouble figuring out the above instructions, an updated version of STOXIN.FOR is shown in Figure 5. A check for division by zero in calculating DJIPCT has also been added.

Figure 5. Program listing of STOXIN.FOR

```
C        PROGRAM STOXIN1
C        This program inputs stock market data for analysis
C        Updated with changes through The FORTRAN Formula - 5
C
C        Declare variables
         INTEGER  MO(60),DAY(60),YR(60),ADVNC(60),DECLN(60),
     1     UNCHNG(60),HIGHS(60),LOWS(60),HLDIF(60),UPVOL(60),
     2     DNVOL(60),ESC,CLRSCN
         INTEGER*4  VOLUME(60),ADVVOL(60),DECVOL(60),ADLINE(60)
         REAL DJIA(60),DJIPCT(60),ADPCT(60),TRIN(60)
C
C        Set up FORMAT specifications for I/O
100      FORMAT(1X)
110      FORMAT(19X,'Stock Market Technical Analysis System')
111      FORMAT(' Enter the date as MM/DD/YY:  ')
112      FORMAT(I2,1X,I2,1X,I2)
113      FORMAT(' Enter the data indicated:')
114      FORMAT(' Closing Dow Jones Industrial Average:  ')
115      FORMAT(F8.2)
116      FORMAT(' Volume of shares:  ')
117      FORMAT(I10)
118      FORMAT(' Advancing Volume:  ')
119      FORMAT(' Declining Volume:  ')
120      FORMAT(' Number of Issues Advancing:  ')
121      FORMAT(I5)
122      FORMAT(' Number of Issues Declining:  ')
123      FORMAT(' Number of Issues Unchanged:  ')
124      FORMAT(' Number of New Highs:  ')
125      FORMAT(' Number of New Lows:  ')
126      FORMAT('+',65X,I2,'/',I2,'/',I2)
127      FORMAT('+',65X,F8.2)
128      FORMAT('+',63X,I10)
129      FORMAT('+',68X,I5)
130      FORMAT('+',2A1)
C
C        Clear screen
         ESC = 27
         CLRSCN = 69
C    This code rearranges array elements and then goes to the
C         data input routines.
604      DO 630 J=1,59

                  J1=61-J
                  J2=60-J
                  MO(J1)=MO(J2)
                  DAY(J1)=DAY(J2)
                  YR(J1)=YR(J2)
                  DJIA(J1)=DJIA(J2)
                  VOLUME(J1)=VOLUME(J2)
                  ADVNC(J1)=ADVNC(J2)
                  DECLN(J1)=DECLN(J2)
                  UNCHNG(J1)=UNCHNG(J2)
                  UPVOL(J1)=UPVOL(J2)
```

```
            DNVOL(J1)=DNVOL(J2)
            HIGHS(J1)=HIGHS(J2)
            LOWS(J1)=LOWS(J2)
            ADLINE(J1)=ADLINE(J2)
            DJIPCT(J1)=DJIPCT(J2)
            ADPCT(J1)=ADPCT(J2)
            ADVVOL(J1)=ADVVOL(J2)
            DECVOL(J1)=DECVOL(J2)
            TRIN(J1)=TRIN(J2)
            HLDIF(J1)=HLDIF(J2)
630     CONTINUE
        WRITE(1,13Ø) ESC,CLRSCN
C
C       Get data, print value input at right side of screen on same line
        WRITE(1,11Ø)
        WRITE(1,111)
        READ(1,112) MO(1),DAY(1),YR(1)
        WRITE(1,126) MO(1),DAY(1),YR(1)
        WRITE(1,113)
        WRITE(1,114)
        READ(1,115) DJIA(1)
        WRITE(1,127) DJIA(1)
        WRITE(1,116)
        READ(1,117) VOLUME(1)
        WRITE(1,128) VOLUME(1)
        WRITE(1,118)
        READ(1,117) ADVVOL(1)
        WRITE(1,128) ADVVOL(1)
        WRITE(1,119)
        READ(1,117) DECVOL(1)
        WRITE(1,128) DECVOL(1)
        WRITE(1,12Ø)
        READ(1,121) ADVNC(1)
        WRITE(1,129) ADVNC(1)
        WRITE(1,122)
        READ(1,121) DECLN(1)
        WRITE(1,129) DECLN(1)
        WRITE(1,123)
        READ(1,121) UNCHNG(1)
        WRITE(1,129) UNCHNG(1)
        WRITE(1,124)
        READ(1,121) HIGHS(1)
        WRITE(1,129) HIGHS(1)
        WRITE(1,125)
        READ(1,121) LOWS(1)
        WRITE(1,129) LOWS(1)
C       End of STOXIN Original version
C
C       Program Segment CALC
C       This program segment calculates technical stock indicators
C         data input from the keyboard
C
C       Advance-Decline Line
        ADLINE(1) = ADLINE(2) + ADVNC(1) - DECLN(1)
C
C       Dow Jones Industrial Average percentage change
C         If DJIA(2)=Ø, then fake a value to avoid dividing by zero
        IF(DJIA(2).EQ.Ø.Ø) DJIA(2)=DJIA(1)
        DJIPCT(1)=(DJIA(1)-DJIA(2))*1ØØ.Ø/DJIA(2)
C
C       Advance-Decline percentage change
        ADPCT(1)=(ADVNC(1)-DECLN(1))*1ØØ.Ø/(ADVNC(1)+DECLN(1)+UNCHNG(1))
C
C       Trader's Index
        TADV = FLOAT(ADVNC(1))
        TDEC = FLOAT(DECLN(1))
        TUPV = FLOAT(UPVOL(1))
        TDNV = FLOAT(DNVOL(1))
        TRIN(1) = (TADV/TDEC)/(TUPV/TDNV)
C
C       High-Low differential
        HLDIF(1) = HIGHS(1) - LOWS(1)
C
C       Calculation finished.  Set up formats to display results
15Ø     FORMAT(' Today''s Advance-Decline Line value: ',I8)
151     FORMAT(' Percentage change in the DJIA. ',F8.1,'%')
152     FORMAT(' Percentage change in the A/D line ',F8.1,'%')
155     FORMAT(' TRader''s INdex: ',F6.2)
156     FORMAT(' High-Low Differential: ',I5)
157     FORMAT(27X,'TODAY''S  STOCK  STATISTICS')
158     FORMAT(36X,I2,'/',I2,'/',I2)
C
C       Print the results
        WRITE(1,13Ø) ESC, CLRSCN
        WRITE(1,1ØØ)
        WRITE(1,157)
        WRITE(1,158) MO(1),DAY(1),YR(1)
        WRITE(1,1ØØ)
        WRITE(1,1ØØ)
        WRITE(1,151) DJIPCT(1)
        WRITE(1,15Ø) ADLINE(1)
        WRITE(1,152) ADPCT(1)
        WRITE(1,156) HLDIF(1)
        WRITE(1,155) TRIN(1)
C
C  Routine to ask operator what to do next
C  Set up formats for I/O
5ØØ     FORMAT('ØYou may now do one of the following:')
5Ø1     FORMAT('     1   Enter more data')
5Ø2     FORMAT('     2.  Quit and save data to disk')
5Ø3     FORMAT('ØEnter the number of your choice, then <CR>: ')
5Ø4    FORMAT(I2)
C  Print message to operator
6Ø2     WRITE(1,5ØØ)
        WRITE(1,5Ø1)
        WRITE(1,5Ø2)
        WRITE(1,5Ø3)
C  Get operator's reply
        READ(1,5Ø4)IANS
C  If IANS=1, then go to 6Ø4, which re-validates the entry and
C      goes to the start of input routines
C  If IANS=2, then go to 6Ø1, which saves data and quits
C  If IANS is not 1 or 2, ask for input again
        IF(IANS.EQ.2) GO TO 6Ø1
        IF(IANS.EQ.1) GO TO 6Ø4
        GO TO 6Ø2
C
C  Routine from Figure 2 to write data to disk file
```

```
601     CALL OPEN(6,'STOCK   DAT',2)
        WRITE(6) MO(J), DAY(J), YR(J), DJIA(J),
     1    VOLUME(J),ADVNC(J),DECLN(J),UNCHNG(J),UPVOL(J),DNVOL(J),
     2    HIGHS(J),LOWS(J),ADLINE(J),DJIPCT(J),ADPCT(J),ADVVOL(J),
     3    DECVOL(J),TRIN(J),HLDIF(J)
620     CONTINUE
        ENDFILE 6
C
        END
```

We're Almost There

You have by now become reasonably familiar with data declarations, IF and DO statements, data array manipulation, and the fundamentals of the FORMAT statement. You have also started on the basics of working with data files stored on disks.

If you have trouble in compiling STOXIN, you probably have not declared all of the subscripted variables. This can lead to some interesting comments farther down the file, because the compiler doesn't know that the name you used is a variable, and makes some assumptions about what it finds.

If you are interested in the structure of a FORTRAN data file, use DDT or SID to look around in the data file STOCK.DAT created by running the modified version of STOXIN. If you are familiar with BASIC data files, you will see some surprises. In our next article, we will discuss data files some more, and we will begin calculating moving averages for the data elements in our arrays. Before long, we will be able to plot the data points and the derived quantities, such as TRIN, graphically on the screen, where they can serve as an aid to stock market decision-making. *

Welcome to *It's Great!*

DOODLER

Graphics Package

. . . for the Zenith Z-100 and Z-150/160

- Full feature graphics design package
- Save designs on disk for later use
- Playback mode for error correction
- Extended text capability includes. . .
  User designed character fonts
  Italic or backslant styles
  All text may be scaled

See DOODLER at your local
Heathkit Electronic Center

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

PAUL F. HERMAN
DATA SYSTEMS CONSULTANT
P.O. Box 206
Safety Harbor, FL 33572

Specification Sheet available on Request

PC EMULATOR SCOREBOARD

DEL – GEMINI BOARD

- DEL has finally provided an acceptable version of the firmware. You will probably see the very first boards shipping toward the beginning of October.

UCI – EASY PC BOARDS

- This product is still in firmware development and is expected to ship in limited quantities toward the end of October or early November.

Note: Due to the increased interest in PC emulation, the above information is intended to let you know how close to actual delivery these two important products are. Since the information is compiled approximately 30 days before you see REMark, it would be a good idea to check with your local Heath/Zenith Computers and Electronics Centers or check with Heath Mail Order (616-982-3285) to determine availability. In the future, we are requesting that users report on products that do or do not work with these two boards, so that we may publish your findings here.

!!! ATTENTION !!!

Local HUG Clubs

We are beginning to work on the information that will appear in the January 1986 Issue of REMark. Please check the Local HUG Club Listing in the January and July 1985 Issues of REMark for any possible changes in your club or possibly a new club you know of that isn't mentioned. Please send your changes as soon as possible to the Heath Users' Group, Attn: Nancy Strunk, Hilltop Road, St. Joseph, MI 49085. Thank you for your cooperation.

WINCAD

The Easy-to-Learn, Fun-to-Use Graphics Package Designed for Engineers, Architects, Programmers, Illustrators & Artists, and Students

- **Circuit Designs**
- **Floor Plans**
- **Mechanical Designs**
- **Logic Diagrams**
- **Flow Charts**
- **Organization Charts**
- **Preliminary Architectural Designs**

$249.95

Demonstration Disk is Available for $15.00 + $2.00 postage and handling
Please write for free brochure.

Dealer inquiries are invited.

REDWOOD DEVELOPMENT

7 Redwood Dr.
(406) 494-5610
Butte, MT 59701

... CHECK YOUR SPELLING AS YOU TYPE ...

The Resident Speller™

PC Version

For the H/Z 150 and compatibles

This unique program allows you to check your spelling as you enter text from your word processor. The PC Resident Speller works with most word processors including WordStar, Microsoft Word, WordPerfect, MultiMate, PeachText 5000, PFS:Write, and Volkswriter.*

* Wordstar is a trademark of Micropro Int'l; Microsoft is trademark of Microsoft Corp; MultiMate is a trademark of MultiMate Int'l; PeachText 5000 is a trademark of PeachTree Software; PFS:Write is a trademark of Software Publishing; Volkswriter is a trademark of Lifetree Software.

The PC Resident Speller sells for $99. A demo disk is available for $2. This program requires 90K of additional memory and a double sided floppy disk drive. Texas residents should add state sales tax.

S & K Technology Inc., 4610 Spotted Oak Woods, San Antonio, TX 78249 (512-492-3384)

Ultimate Communications Software

HyperACCESS™

A Dramatic Step Beyond ACCESS™†

HyperACCESS gives you tremendous power—power that lets you communicate without getting technical—power that lets you glide through applications other programs can't touch.

A New Standard for Ease of Use

A startup program sets up everything for you and a fascinating on-screen tutorial lets you try out program features in simulated use of Compuserve, MCI Mail, Easylink, and others. HyperACCESS menus are instinctively easy to use, so you can focus on your work, instead of on how to run the program.

Effortless Modem Support

Instantly call any remote system by selecting it from a menu that lets you list up to 127 systems and already lists Compuserve, The Source, MCI Mail, Easylink, and other systems (subscriptions not included). The first time you call these systems, your password and user ID are memorized, and they are entered for you automatically, thereafter.

Remarkably Flexible

HyperACCESS comes ready-to-use for the more popular applications and it lets you set and store all the details you need, to connect with main frames, minis, micros, or RS232-compatible devices that defy other programs.

Terminal Emulation

HyperACCESS emulates the most popular terminals, including DEC VT-52, VT-100, TeleVideo 900 series, IBM 3101, and H-19, to let you run programs on remote computers designed for use from terminals.

Use Your Computer's Power

From within HyperACCESS, you can use your computer's DOS commands (DIR, COPY, etc.) or run your other programs. You can even define DOS macros, so you can execute a series of DOS commands and/or programs by pressing a single key.

Use Xmodem, Kermit, or Text Transfer Methods

. . . to reliably send/receive files of any kind from/to any disk drive or directory in your computer, even through noisy telephone lines.

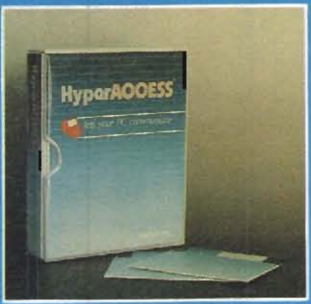
Remote Use of Your Computer

Press one key and, even when you're absent, operators of remote terminals or computers can access your computer by entering passwords you define. You can set whether each password allows a user to read or type messages, transfer files, run a given program, or use your operating system and all your programs.*

Automate Your Communications

. . . so you can perform, with a keystroke, steps that normally would take many keystrokes. HyperACCESS has the clearest, most comprehensive script language available, to make automating communications easy for beginners and exhilarating for experts. You can automate any part of a call, make unattended calls, or create custom functions, prompts, and menus.

For astonishing power and versatility, get HyperACCESS today!



For PC-compatibles and Z-100 computers with DOS and any modem, including 2400 baud and USR S100 modems. You can get HyperACCESS for $149 from computer dealers or from Hilgraeve Inc. (please specify computer and add $5 shipping).

Hilgraeve Inc.
P.O. Box 941
Monroe, MI 48161
(313) 243-0576

† Hilgraeve and its dealers will continue to sell ACCESS, as well.
HyperACCESS and ACCESS are trademarks of Hilgraeve Inc.

* Certain programs cannot be operated remotely.

Continued from Page 41

Part Number	Description of Product	Selling Price	Vol. Issue
885-1109-[37]	HDOS Retriever ASM (3 Disks)	40.00	23
885-1110	HDOS Autofile (2 Disks)	30.00	23
885-1115-[37]	HDOS Navigational Program	20.00	25
885-8008	Farm Accounting System	45.00	30
CP/M			
885-1219-[37]	CP/M Navigational Program	20.00	31
MSDOS			
885-8034-37	DBZ-A Database For The Z100	25.00	69
	AMATEUR RADIO		
HDOS			
885-8016	Morse Code Transceiver Ver 2.0 ...	20.00	42
CP/M			
885-1214-[37]	CP/M MBASIC Log Book (64k)	30.00	23
885-1234-[37]	CP/M Ham Help	20.00	49
885-1238-[37]	CP/M Ascirity	20.00	57
885-8020-[37]	CP/M RF Comp. Aided Design	30.00	44
885-8031-[37]	CP/M Morse Code Transceiver	20.00	57
	COMMUNICATION		
HDOS			
885-1122-[37]	HDOS MicroNET Connection	16.00	37
CP/M			
885-1207-[37]	CP/M TERM & HTOC	20.00	26
885-1224-[37]	CP/M MicroNET Connection	16.00	37
885-3003-[37]	CP/M ZTERM (Z100 Modem Pkg) ...	20.00	34
885-5004-37	CP/M-86 TERM86 and DSKED	20.00	56
885-5005-37	CP/M-86 16 Bit MicroNET Conn. ..	16.00	61
885-5006-37	CP/M-86 HUGPBBS	40.00	62
885-5007-37	CP/M-86 HUGPBBS Source List.	60.00	62
885-8005	MAPLE (Modem Appl. Effector)	35.00	29
885-8012-[37]	CP/M MAPLE (Modem Program)	35.00	34
885-8023-37	CP/M-85 MAPLE	35.00	45
MSDOS H/Z100 - H/Z150 PC			
885-3019-37	ZDOS 16 Bit MicroNET Connect. ..	16.00	61
885-3027-37	MSDOS HUG PBBS	40.00	66
885-3028-37	MSDOS HUG PBBS Source Listing ..	60.00	66
	MISCELLANEOUS		
885-0004	HUG Binder	5.75	
885-1221-[37]	Watzman ROM Source Code/Doc ...	30.00	33
885-4001	REMark Vol. I Issues 1-13	20.00	
885-4002	REMark Vol. II Issues 14-23	20.00	
885-4003	REMark Vol. III Issues 24-35	20.00	
885-4004	REMark Vol. IV Issues 36-47	20.00	
885-4005	REMark Vol. V Issues 48-59	25.00	
885-4500	HUG Software Catalog	9.75	
885-4600	Watzman/HUG ROM	45.00	41
885-4700	HUG Bulletin Board Handbook	5.00	50
885-3015-37	ZDOS Skyviews	20.00	55

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.

Continued from Page 7

cle. Thank you for your interest.

Instructions:

Change the number 10 to 11 in the lines 1140, 1150, 1160.

Add the line "1375 DATA X,Y,Z,VX,VY,VZ", where the X through the VZ represent the position and velocity components of the celestial object. (Example: 1375 DATA .2,1.2,-.1,-.005,.003,.008)

Add the color and name of the object to line 1840. (Example: 1840 3,URANUS,2,NEPTUNE,5,PLUTO,2,PROBE) The name of the object must be 7 letters or less.

Change the 10 in line 2380 to NP, which should now read: 2380 FOR I=1 to NP.

Sincerely,

James Tursa
3489-C Lake Austin Boulevard
Austin, TX 78703

No Support For Prowriter

Dear HUG:

Could you please help me with information on the following subject?

I own a Zenith Z-150 PC and a C.Itoh Prowriter printer 8510A. However, this popular printer is not supported by ZDS. I am using Microsoft WORD and can't find the Prowriter in the list of printers supported. Do you have any information available to solve my problem?

Thanks for your help.

Yours sincerely,

F.H. Knottenbelt
Boedapeststraat 24
2034 CZ Haarlem
The Netherlands

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

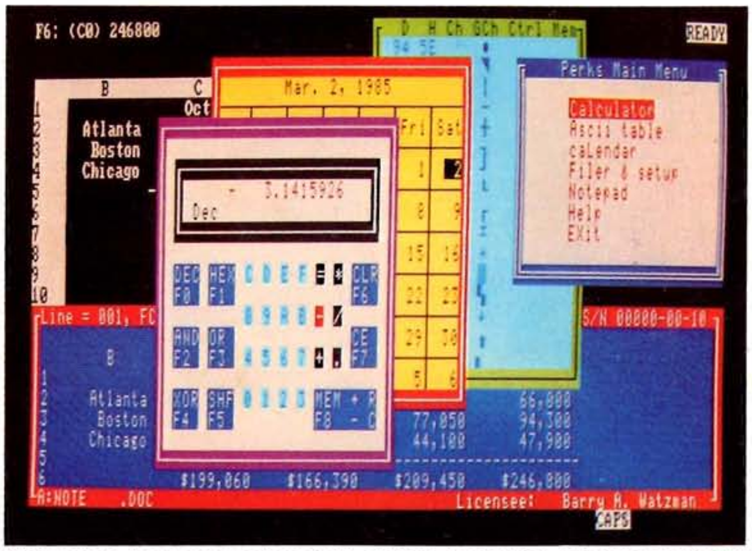
	NEW MEMBERSHIP RATES	RENEWAL RATES	
U.S. DOMESTIC	$20 ☐	$17 ☐	
FPO/APO & ALL OTHERS*	$35 ☐	$30 ☐	U.S. FUNDS

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

If you have a Z-100, treat yourself to some

PERKS™

Perks is the background desktop utility program for Z-100's featuring full-color windows, a calculator, a notepad, an ASCII table, a Filer, a screen saver, context sensitive help screens and a calendar with an appointment schedule.



Shown above is an actual screen photo of Perks in operation. The Notepad window contains data "imported" from the Lotus 1-2-3 worksheet being prepared when Perks was activated.

Perks works with all Z-100's having 192k or more of memory and using the Z-DOS/MS-DOS 1.X or 2.X operating system.

Here are some actual unsolicited comments from customers:

"My heartiest congratulations and thanks for a fine product. I expect that Perks will be the best software dollar I have yet spent, and I have spent lots."

"Anyone with a Z-100 should run to buy Perks"

"Thanks for a great program . . . Please . . . keep up the good work for us Heath/Zenith users."

"I've been playing with Perks, and having tons of fun; I think it'll be extremely useful to me . . ."

"I know I said this over the phone, however, once again I would like to thank you for a GREAT product."

Perks is priced at $99.97 and is available from:

BARRY A. WATZMAN Microcomputer Systems & Consulting

560 Sunset Road • Benton Harbor, Michigan 49022-7142 • (616) 925-3136

Heath/Zenith Users' Group

Hilltop Road
Saint Joseph, Michigan 49085

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

POSTMASTER: If undeliverable, please do not return.

P/N 885-2070