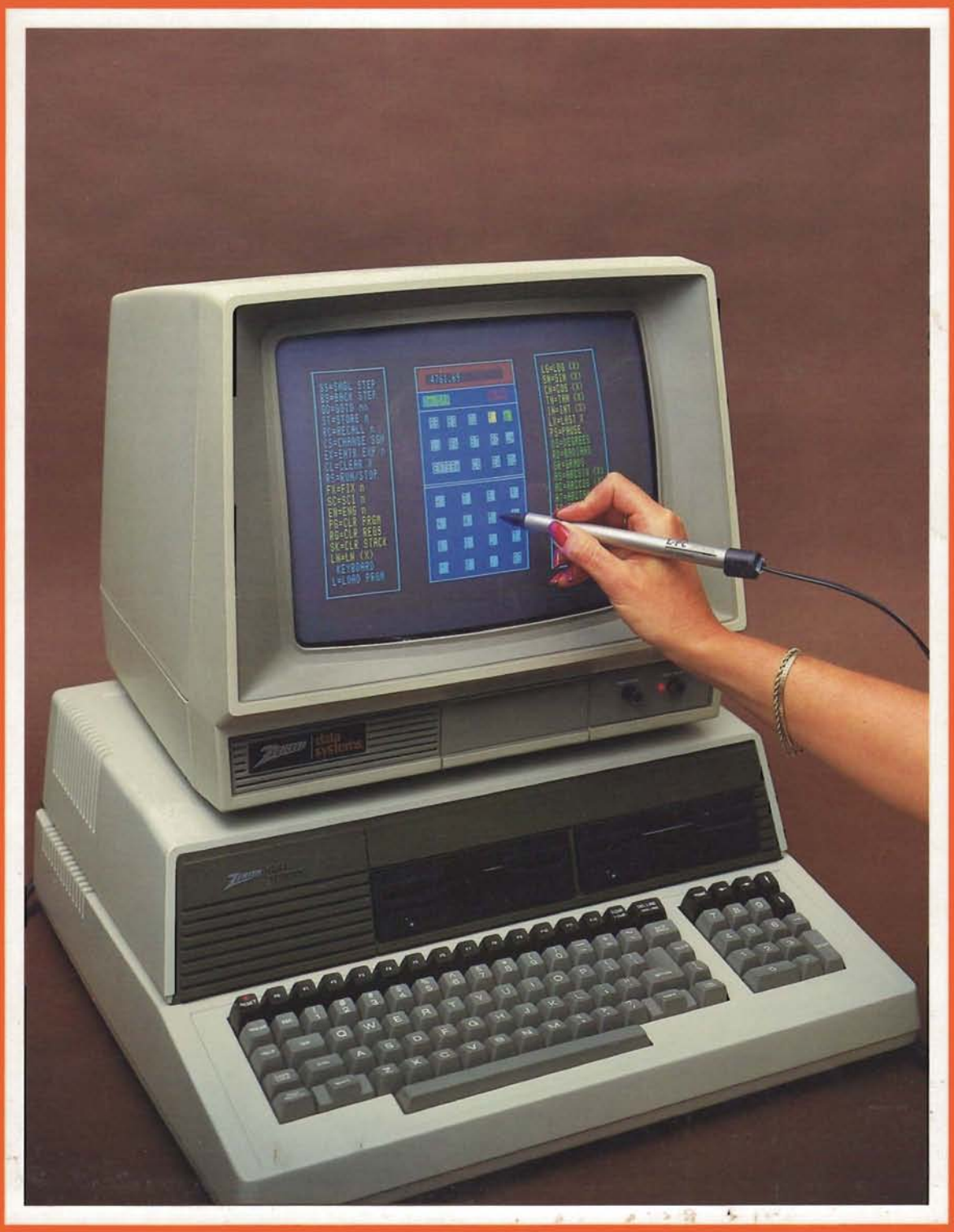
REMark®

Volume 7, Issue 9 • September 1986

P/N 885-2080 Issue 80

Official magazine for users of HEATH ZENITH computer equipment.

$2.50



The Z-171 with LiteDrive™ ... heavy duty performance in a lightweight laptop

The Zenith Data Systems Z-171 with LiteDrive™ ... heavy duty performance that goes anywhere

Zenith Data Systems and Premier Technologies give the serious PC user convenience and practicality in a completely portable full feature system for *all* your computing needs ... the Z-171 with LiteDrive™.

Now you can have the heavy duty performance of a 10 megabyte hard disk *and* the practicality of a lightweight laptop!

You no longer have to sacrifice features for portability because the Z-171 with LiteDrive™ is a truly compact portable system with full size performance.

This entire full-feature system *including the internal hard disk* weighs only 14.33 pounds. And, LiteDrive™, while adding less than one half ounce to the unit, offers 27 times the memory at up to 20 times the access speed of the standard floppy disk system.

Full Size Performance

Of course, full size performance means you can dash through virtually *any* software program with the Z-171's full MS-DOS compatability. And at the touch of an icon key you can have the added utility of features like a calculator, a world clock and calendar, an appointment schedule book, and a phone book with asynchronous communications capability.

ZENITH | data systems

Z-171 + LiteDrive™ Specifications

- 80C88 CMOS processor
- MS-DOS 2.11
- 256K RAM standard (Expands to 640K)
- Single 5.25" DSDD drive
- 10 MB internal hard disk
- Flat 10" backlit LCD
- 25 lines of 80 characters
- 640 x 200 resolution
- Standard QWERTY keyboard
- Serial & parallel port standard
- Optional: Battery pack, carrying case, memory upgrade, and internal modem
- 14.33 lbs — 9.5 x 13 x 6.6 inches

Complete Graphic Capability

Full graphic capability and excellent visibility are also provided by the full size backlit LCD screen.

Easy input with fewer errors springs from a full size sculptured keyboard. And speedy data entry is *no problem* with features like a full ASCII 128 character set, 4 cursor controls, 10 programmable function keys, and numeric keypad.

Truly Portable

Best of all, LiteDrive™ gives you internal hard disk power *to go*. For computing *anywhere*, the system runs on both AC and rechargable standard battery current, making the Z-171 *truly* portable. Even in the most severe applications safe, reliable drive integrity is assured by an automatic park function.

The LiteDrive™ is both rugged and durable thanks to a 70g shock rating. And for those who admire sheer simplicity, the entire upgrade package is field installable in about 30 minutes.

With the increased speed and storage capacity of LiteDrive™, your laptop becomes a powerful portable phone system when equipped with the optional internal modem Now you can communicate with the office, other systems, and an endless number of information networks and process data more conveniently *and even faster!*

The Zenith Data Systems Z-171 and Premier Technologies' LiteDrive™ ... a breakthrough in price, practicality, and most of all ... *performance.*

MS-DOS is a trademark of Microsoft Corporation

Premier Technologies, Inc.

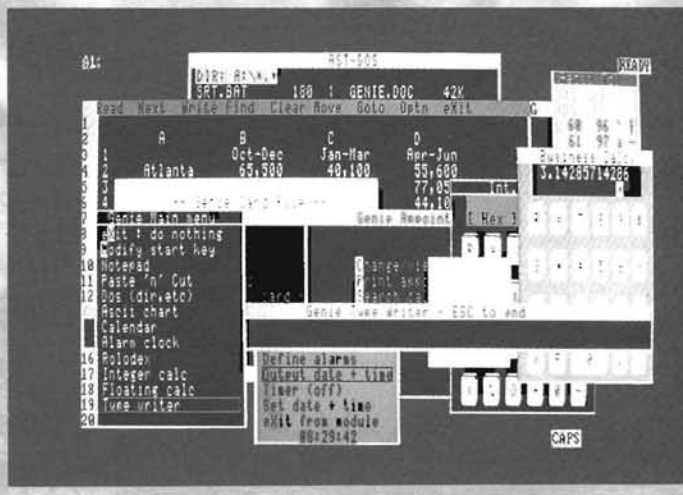
1890 McGaw Avenue • Irvine • CA 92714 • 714-261-1184 • Telex 678185

WA, OR, ID, MT, AK	CA, HI	AZ, NM, NV, UT	ND, SD, MN, IA, WI	MI	CT, MA, NY, RI, NH, VT, ME	NY, PA, NJ	VA, MD, DC	SC, NC	ALL OTHER AREAS
Microware Distributing	Premier Source	Electrical Equipment	Compar	KS Electronics	Northeastern Distributors	Peirce-Phelps	Zamoiski	Allison-Erwin	Premier Technologies
503-642-7679	714-261-2011	602-275-7801	612-944-8086	313-525-1155	617-547-8000	215-879-7079	301-644-2900	704-334-8621	714-261-1184

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 is a memory resident application. This means that once you load Genie it is always available for you to use. Just hit the magic keys and Genie will appear (Shift-Shift: No function keys lost.) You can have Genie perform various tasks, and when you finish Genie goes away and you are back where you started.

Here is what you get with Genie:

- **KEY MAPPER** - Redefine any key on the fly, store long commands in a single key, and save many maps on disk.
- **TrueDOS**™ - Run any MS-DOS program inside the window. FORMAT, SORT, BACKUP, compile, all without leaving your application.
- **CALCULATORS** - you get two calculators, one regular floating point calculator, and programmer's calculator for base conversions and bit operations.
- **NOTE PAD** - you'll never have to hunt for paper and pen again. Simply call up Genie's Note Pad and jot down what you need. Or expand the buffer to 56K: Instant editor.
- **CUT AND PASTE** - Cut text from any place on the screen and output it later. Cut long commands off the screen and into your KEY MAPPER, move data from your spread sheet to your word processor: Instant integration.
- **CALENDAR** - schedule appointments for any year up to 9999 keep track of expenses, search and print the calendar.
- **ROLODEX** - a name address and telephone number list that you can search any time. No limit on the number of cards. You can even output an address directly to your word processor.
- **ALARM CLOCK** - Have your Genie remind you of appointments, set alarms to ring at any time on any day. A window appears with a reminder of what each of the 8 alarms is for.
- **ASCII Table** - programmers never have to leaf through big books to find the ASCII value of a character.
- **TYPEWRITER** - Knock out quick memos any time, even send ESCAPE codes to your printer. Cut a portion of a spread sheet and paste into the typewriter for quick printouts.
- **SCREEN SAVER** - Automatic phosphor protector for your tube. Genie will even let you blank the screen manually to discourage peepers.
- **COMMAND STACK** - Lets you access up to the last 2K worth of commands you typed.

Best of all with Genie you only have to load what you use. If you don't need to use an ASCII chart don't load it. Need extra alarms? then load two alarm clocks. Your own personal Genie.

And now Genie user's have the advantage of Genie accessories, which can be installed and accessed like any Genie function - any time:

- a **Scientific Calculator** more powerful than most real calculators. Has a full range of math, science and statistical functions and very high accuracy.
- a **Spelling Helper**™ To help you find the correct spelling without leaving your word processor.
- a **Print Spuffer**™ (spooler/buffer) to speed and manage all your printing.
- a development pack to let you write your own pop up programs, including many routines for windowing, and source code for our Rolodex. Writing pop ups is a snap. (Please call for details)

Only Genie Gives you so much for so little.

ADVANCED SOFTWARE TECHNOLOGIES

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

ORDERS ONLY CALL

1-800-437-4400 Ext 800

Yes please rush me the following software for my
_____ Z110/120 _____ PC compat.

_____	Copies of Genie	@ $54.95	_____
_____	Spelling Helper(s)	29.95	_____
_____	Print Spuffer(s)	24.95	_____
_____	Scientific calc.	19.95	_____
_____	Genie pack (I want it all) Saving $40	$89	_____
		Sub total	_____
Check encl ☐		shipping	4.00
M/C ☐		NY Tax (8.25%)	_____
VISA ☐		TOTAL	_____

Card Number _____

Expires _____ Your Tel Num _____

Name _____

Address _____

Company _____

City/ST/Zip _____

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

Production Coordinator ..... Lori Lerch
(616) 982-3794

Secretary ..... Margaret Bacon
(616) 982-3463

HUG Bulletin Board ..... (616) 982-3956

Contributing Editor ..... William Adney

Contributing Editor ..... Joseph Katz

Printer ..... 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®

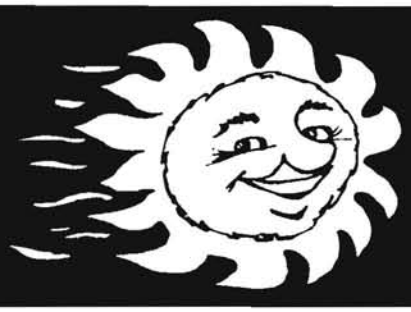
Official magazine for users of HEATH ZENITH computer equipment.

Volume 7, Issue 9 • September 1986

Index of Advertisers

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

On The Cover: *This month's cover features Calc Software, by Robert F. Doolittle, and the L-PC Lite-Pen, by the Lite-Pen Company of Los Angeles. The HUG New Product listing of Calc is on Page 42. An article about the L-PC Lite-Pen is on Page 65. Photos taken by Dan Wilson, Heath Graphic Design.*



End of Summer SALE

Say goodbye to summer with a special buy on a sensational computer!

HS-241-81
IBM AT Compatible
Computer with monitor

New IBM AT Compatible Advanced Personal Computer

Only **$2249***

• Fast one-evening kit • **MS-DOS** operating system • 512K RAM, **expandable to over 15MB** • single **1.2MB** 5.25" floppy disk drive • 8 expansion slots • start-up diagnostic • famous **Heath/Zenith** quality

HS-148-41 IBM-PC Compatible Computer Kit with ZVM-1220A Monitor

IBM PC/XT Compatible Computer with 10 Meg Drive, Free Monitor & 640K Memory

Kit Only **$1395***

• **MS-DOS** operating system • runs **IBM software** • **360K** disk storage • **640K** memory • 8 MHz CPU • both mono and **color graphics** capability • 10 Meg **Winchester** hard disk drive • controller, expansion card, clock/calendar • solderless one-evening kit

*Price includes HUG discount.

You can also enjoy great savings on quality peripherals and accessories like these!

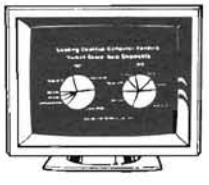
CM-1000

C.ITOH RGB Color Monitor

Only **$399**

Reg. $499

Works with IBM PC and compatibles. Plus use as a TV with your VCR.



NEC Multisync High Resolution Color Monitor

Only **$679**

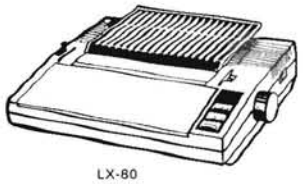
Reg. $899

• Compatible with **IBM PC/AT, PC/XT, PC** and look-alikes • compatible with all three PC color graphics boards made by IBM • color resolution superior to any comparatively priced monitor • 7 colors available

Epson LX-80 Printer

Just **$249.95**

Reg. $329.00



LX-80

Correspondence quality FRICTION FEED printer prints directly on your letterhead • Fast! 100 characters per second • Select typestyles at a touch. Over 150 combinations • Mix text and graphics on same line • One-year warranty.

NASHUA DSDD Diketttes

Lifetime guarantee.

Box of 10 Special **$9.95**

Phone orders accepted.

Prices good through October 15, 1986.

Call 616-982-3614 for store locations or consult your Heathkit Catalog.

HEWLETT PACKARD

ZENITH | data systems

AUTHORIZED DEALER

HZC-239

Your TOTAL SERVICE computer center

Service • Support • Software • Accessories • User Training • Competitive Prices

Heath® ZENITH®

Units of Veritechnology Electronics Corporation

Computers & Electronics

BUGGIN' HUG

Status Of MASM V4.0 Problems . . .

Dear HUG:

In the July issue of REMark (Volume 7, Issue 7), I stated that there were two problems with MASM V4.0 that surfaced when I tried to assemble the Z-150 version of the ANSI device driver.

The first problem had to do with the way a -1 is handled in V4.0 versus V3.0. Microsoft stated that it is a known problem, but has not sent me a fix yet.

The second problem had to do with multiple end-of-file characters (1AH) in the Z-150 BIOS source files. Two weeks after I submitted the problem to Microsoft, they sent me a patch to fix the problem. If anyone is interested in the patch, I can send it to them. At the same time, if there is enough interest, I could put it in the next issue of REMark.

Rich Mueller, Ph.D.
11890-65th Avenue N.
Maple Grove, MN 55369

Poor Man's Method For Changing Directories

Dear HUG:

For over a year I have enjoyed REMark. I have gained marvelous tips from the articles and the Letters to the Editor. I now have an opportunity to share some of my knowledge. I read Joseph Katz', "The WordStar Directory Changer" in the April 1986 issue. I have found the 'poor man's' method for changing directories in WordStar or ANY program. I have a Z-100 with MSDOS 3.0 and Pascal, Condor, Multiplan, and WordStar. I am using WS and DMBS in a RAM drive. The following batch file works with any of these programs. The batch file is called directory.bat.

```
a:
chkdsk b:/f
subst c: b:/prnlabel
subst d  b:/songs
subst e. b:/letters
subst f: b:/records
ws
a:
subst c  /d
subst d: /d
subst e: /d
subst f: /d
```

The disadvantage is that you must put a 'directory.bat' on each data disk, but usually this is a one time job. After the first one, copy it to your other disks and change the directory names. Make sure subst.exe and chkdsk.com (keeps you posted of room on disk) are on the same disk as your programs. If you use more than one program in place of 'ws', you can use

```
pause [Type 'start ws' or 'start dbms']
```

Start.bat is below:

```
echo off
if "%1" == "dbms" goto x
if "%1" == "ws" goto y
:x
echo on
a:
pause [Set date ]
dbms
goto end
:y
a:
ws
 end
```

Now you only need to log on to a letter in any of the programs I mentioned. Instead of typing the command for a program type 'b:directory'.

Mr. Katz made passing reference to having to 'reestablished special setups' in WordStar. Format files are my favorite. If I have a file with any variance on the default margins or tabs, I create a ruler line at the top of the file. Frequently used setups have their own file with an .fmt extension. For example, the setup for this letter is the following:

```
   [Crtl]P Return
----!-------------------------------------------------
 op
 mt4
 po5
 cw10
```

This format file is called let12.fmt for letter with 12 pitch. Create a file and use the [Ctrl]KR to bring forth the format file. Use [Ctrl]OF on the ruler line below the two dots and your status line is identical to your ruler line, tabs and all.

I agree with Mr. Katz, the subdirectories do help keep a large directory organized.

Keep up the good work, REMark!

Salli Brackett
1770 Adelaide Street #105
Concord, CA 94520

Interfacing The Princeton Max-12 Monitor And The H/Z-151 PC

Dear HUG:

A special monochrome board (Z-329) for the H/Z-151 PC is available that offers the advantage of sharp, well formed characters. There is barely a hint of the pixels forming each one. After months of going buggy staring at a green composite screen (ZVM-122), I got this board and hooked it up to a Princeton Graphics Max-12 monitor.

The Max-12 is unique in that it can display monochrome characters (from the Z-329 or equivalent board) or shades of amber which correspond to the 16 color palette of the color board. You'd think that the 9 pin connector could plug directly into the color card of the Zenith PC, but modifications are needed:

1. On the 151 color board, find R322 (close to the output connector) and solder a 1K 1/4-watt resistor in parallel with it.

2. Set the internal jumpers as follows:

   J303 +
   J304 V
   J306 +
   J307 +
   J308 +

DIP Switch 2 (on CPU Card) Position 7 – right (60 hz)

3. In the Max-12, place a jumper wire across D301 (which is near the input connector). This is best done on the foil side of the board.

4. Set the screen up for a steady monochrome signal, then switch the connector to the color board. If the signal is scrambled, adjust the H2-Hold pot inside the Monitor for a stable display when switching between the two modes.

5. Build a hardware switch with a three pole double throw switch. Switch pins 6, 8 and 9 between the two boards. Also connect pins 3, 4 and 5 from the monitor to the color board. Connect pin 7 from the monitor to the monochrome board. Pins 1 and 2 on both boards should be grounded.

6. Software may also require switching before use. Most programs include the switch in their preamble so all you'll have to do is manually flick the switch. If this is not the case, you can switch from mode V M 7 to mode V M 3 at the boot prompt (internal monitor). There are many public domain utilities available that will switch modes while in DOS.

The P.G.S. monitor looks great and does double duty. The total modification to my system was a fraction of the cost of a new graphics card plus color monitor and gives me the best of both worlds.

Thanks to Princeton Graphics for the technical information.

Rick Liftig
129 Green Road
Meriden, CT 06450

COBOL For The Masses

Dear HUG:

Hello! I thought it might be useful to the members to share my experience with the only (that I know of) low cost COBOL compiler available for micros. I speak of the Ellis Computing package advertised in BYTE ($40).

I'm a teacher and have been for the last 20 years, I own a Z-151-52 and have been 'into computing' for about five years. My expertise is in communication more than in the inner workings of the hardware. I use a computer for what it can do for me rather than for the fun of experimentation for which I have great affection, but little time. I offer the following article for that part of the membership that might be interested in one of the oldest languages around at a price most can afford.

"What self-respecting micro owner would be interested in one of the oldest high level languages extant? Not me! Not, at least, until I met Ralph. Ralph was a programmer when the only language was that of the machine. When COBOL came along it was a boon for the work he was doing. He tried to convince me of its simplicity, effectiveness and widespread use. He even produced charts showing that it was, by far, the most ubiquitous language in use today. Failing to get me hooked, he made the ultimate sacrifice, he would teach a course here at Salem College if I would enroll. How could I say 'no'?

It immediately became apparent that if I was going to do more than pay lip service to the language, I needed to be able to work on it at home when I had the time. What to do? I could have bought a terminal emulator and a modem to access the PDP at the college or a compiler that would run on my Z-150. Microsoft has one ... for $400. It was then that I noticed the ad for an "ANSI compatible" compiler from Ellis Computing. The text emphasized its speed, and it came with a book. Apparently, the company has been around for a while. It is behind the "NEVADA" line that has been producing low cost software for the eight-bit world for some time. What did I have to lose at one-tenth the cost? Within a week of my call to San Francisco (they've since moved to Nevada) I had my package.

If you know nothing about COBOL, I will relate, very briefly, what the language is like. It was designed to handle data. It is used by banks and probably credit card companies to update files on customer purchases, payments, billings and other routine data handling. One doesn't find it much in the micro world because of its history and the vast amounts of data it manipulates. It is unglamorous when compared to the 'latest thing' and lacks the ease of use when it comes to certain tasks that micro users have taken for granted. Since most typical applications run in batch mode, generating screens are a pain and even report writing is more difficult than one would expect. In fact, about half way through the second course I took with Ralph, I concluded that using dBase II was alot simpler and accomplished the same thing. I came to this conclusion because I have no experience in the world of mainframes and little on minis. For truly large applications that go beyond the capabilities of micros (and are really inappropriate for them), COBOL is the language of choice. If one suspects that a vocational opportunity may someday include 'data processing', it wouldn't hurt to know the language.

One quick example. To clear the screen one must send a machine specific code in a DISPLAY statement. The manual lists what these are for a number of terminals (including the ANSI standard), but search as I might, I could not discover what the code is that works with my Zenith (it's not the MS-DOS CLS nor the hex equivalent). I also cannot afford the Programmers Utility Pack which might contain it. Now, I could configure as an ANSI terminal by amending my CONFIG.SYS file, but that would mean changing DEVICE= MDISK.DVD (the RAM-disk maker). Oh, I imagine I could use different software to create the RAM-disk, but by this time I just didn't want to spend any more time with it. (Alot more difficult than entering CLS in BASIC or ClrScr in Turbo).

Well, what's UTAH COBOL like? It is not a full implementation of the language. The best analogy I can think of recalls the way I learned BASIC. On a VIC-20 and Commodore-64. For those who don't know, these machines use a subset of Microsoft BASIC known as V2. Bare bones. Not all bad though. Most of the functions left out are still available by way of "work arounds" which may take longer, but still accomplish the purpose. When I related this to Ralph he wasn't dismayed at all. "COBOL is COBOL" he was fond of saying. "It just means you'll need more ingenuity." In a nutshell, that's UTAH COBOL.

The manual supplied is also anemic. I had two other textbooks to help me, but they were only good for understanding the language, not coding. This was because so many of the commands in 'standard' (viz. ANSI) COBOL are not implemented. It (the manual) was typical of those that try to be tutorials and references at the same time ... namely, not complete as either. As with so many other aspects of this compiler package I kept saying 'what can you expect for $40'. Still, I successfully completed every assignment that Ralph and his sucessor imposed.

While UTAH COBOL (hereinafter referred to as UC) is a compiled language, it is not LINKed before being run. I believe the "run-time interpreter" allows for more generic implementations, probably

contributing to the low cost. The package is not supplied with an editor, so for those of us used to Turbo Pascal's environment . . . After some initial experimentation with the edit1-compile-edit cycle, I put everything into a RAM-disk and was much happier thereafter. Just don't forget to copy the file to disk before you quit.

Pages 133 to 140 in the manual summarize the ANSI-74 reserved words and those implemented in UTAH's version. I will try to relate my own experiences dealing with the work-arounds necessary to accomplish the task. UC lacks a COMPUTE verb, as well as an exponentiation operator. Thus, when writing a 'return-on-investment' program, I had to use the arithmetic operators ADD, SUBTRACT, MULTIPLY and DIVIDE only. A nuisance, but I survived. UC does not implement a DELETE verb either. In a file-maintenance program I had an interesting time trying to delete a record in a relative file. Briefly, I had to decrement the KEY for all successive records then rewrite them. It should be easier. UC also lacks BASIC's equivalent of APPEND (in standard COBOL, EXTEND); that is, adding to a sequential file. Thus, one must treat an appending operation in the same way as a change in a record, i.e. read the entire sequential file into memory, make the change (or add the record), then rewrite the file. This latter operation isn't helped by the inability to have an OCCURS within an OCCURS (two dimensional array); only one subscript please. Nested IFs are also impossible without a work-around (in general, no conditions within conditions). Ah yes! As I go through the alphabetical list of reserved words I am reminded of the lack of INDEX. No sane person wants to be limited to sequential file access only and UC provides RELATIVE file handling. When I was given an assignment to write an indexed sequential file program, I necessarily used this feature. A good substitute, but I wanted to learn ISAMs too. There's no MERGE or SORT (and, of course, no SORT-MERGE) function. No, can't do a PERFORM . . . VARYING, either.

The last assignment was to use a CALLed program. This 'subprogram' is compiled separately from the main one. Being naive, I thought the run-time unit could somehow know not to load the program at the starting address of the calling program (thereby obliterating it), but I was wrong. The called program must be ORGed correctly (it must be told to load beyond the calling program's end); I still haven't figured out how to do that. Those of you with assembly language facility should be able to (I guess) if you can translate the relevant section of the manual.

I'm trying not to be tedious, but these features are simply not available and I feel I should say so. There are also many other 'non-implementations' that I just didn't run into. On the final exam I was asked about how to use START. Hmmmm. Never had call to use it . . . and it's not implemented anyway.

So, what's the bottom line? If you want to learn COBOL on a microcomputer and can't afford the $400 for the 'standard', you could wait for Borland International to release Turbo-COBOL (how long are you willing to wait?) Or you can spend $40 for a compiler and book neither of which is state-of-the-art. If I got a job tomorrow that relied on writing COBOL would I be able to? Frankly, I don't know. I'd like to believe that I know enough about the structure of the language to accomplish a task, but until such time comes I can't say for sure. Ralph seemed to think so."

Lew Roberts
636 Lawman Avenue
Bridgeport, WV 26330

Using Mixed-Cases In MS-DOS 2.1

Dear HUG:

Your readers may be interested in a fix for a "feature" I've discovered in both the Z-100 and Z-100 PC versions of MS-DOS 2.1. The same "feature" probably occurs in IBM versions, although I haven't verified this. The problem occurs when using mixed-case input to the FOR command. Although essentially all "generic" MS-DOS commands are case insensitive, the syntax check applied to FOR is not. To remind your readers, the correct syntax is

```
FOR %variable IN (set) DO command
```

As with other MS-DOS commands, 'FOR' may be in mixed case, and 'command' will be properly interpreted by the shell when passed back after variable substitution irrespective of case. But 'IN' and 'DO' must not be mixed. Either 'DO' or 'do' will work, but 'Do' will not! This is not documented in the MS-DOS manual, nor is it in the spirit of MS-DOS. I believe this has been changed in MS-DOS 3, but I haven't been able to verify this because I can't boot MS-DOS 3 on my Z-100 with the UCI RAM board installed. The fix will be available from UCI "real soon". (While they were at it, however, the guys at Zenith built case-sensitivity into FC under MS-DOS 3. Although, of course, 'FC' can be 'fc' or 'Fc', the switches must be lower-case. I can find no other example of this anywhere in MS-DOS. Can anyone figure out why they did this?)

Using DEBUG, the fix to FOR is quite straightforward. I have found that COMMAND.COM actually checks the FOR command line for 'IN', and 'in', 'DO' and 'do' separately. My solution is to lower-case the word being checked before the test is made, and then check against 'in' and 'do'. The procedure for making the patch follows. As usual, one should make a copy of COMMAND.COM on a bootable disk and make all DEBUG changes to the copy, never changing the original distribution copy. One additional complication arising from patching COMMAND.COM which does not occur with other programs is that the system must be rebooted (CTRL/ALT/DEL on the PC or CTRL/RESET on the Z-100) before you can test your work. In the example below, I will reproduce the locations appropriate to my copy of COMMAND.COM on the Z-100, dated 06-Sep-84. The segment register will depend on what resident programs are loaded, and the offset will depend on your particular version of COMMAND.COM.

Assuming you have prepared a new bootable disk on drive B: and have a copy of DEBUG on drive A: (the "-" is the DEBUG prompt)

```
A>DEBUG B:COMMAND.COM
-R                          Makes list of registers and flags
```

Note the value of the CX register for use in the Search commands below. For my version of COMMAND.COM it's 4025h. The offset returned by the Search will depend on your COMMAND.COM, and its value is then used in the Enter command which follows:

```
-S 100 L 4025 3D 49 4E   Search for  CMP  AX,"IN"
xxxx:1CBB                Found at offset  1CBB
-E 1CBB 0D 20 20 90 90   Replace with OR  AX,2020h; NOP, NOP
-S 100 L 4025 3D 44 4F   Search for  CMP  AX,"DO"
xxxx:1D13                Found at offset  1D13
-E 1D13 0D 20 20 90 90   Replace with OR  AX,2020h, NOP, NOP
-W                       Write out the new COMMAND.COM
-Q                       Quit -- Return to MS-DOS
```

That's it! Now reboot your system from the "new" system disk and try the following:

```
FOR %x In (*.*) Do Dir %x
```

This isn't a particularly useful implementation of FOR, but at least it should no longer produce a 'Syntax Error'. Your new copy of COMMAND.COM can now be propagated to other disks using COPY.

Incidentally, the above bug and its fix were encountered because I have recently discovered how to make FOR do some useful work I had been stymied on. As a result of a letter in PC Magazine on another topic, I now know how to make FOR run a batch file. For example, suppose you have a batch file named DOFOR.BAT to do a Microsoft Fortran compile, containing the lines:

```
FOR1 %1
PAS2
```

You might expect you could invoke this batch file for all Fortran source modules in a directory with:

```
FOR %x In (*.FOR) Do DOFOR %x
```

However, this does not work — DOFOR is invoked only once. What does work is:

```
FOR %x In (*.FOR) Do COMMAND/C DOFOR %x
```

Knowing that this works, and that the first try above doesn't, you can probably figure out the hows and whys by re-reading the MS-DOS manual descriptions of the MS-DOS components and the command COMMAND. It's too bad Zenith didn't include an example like this in the manual!

Sincerely,

Howard Rubin
229 N. Taylor Avenue
Oak Park, IL 60302

Configure Your Printer From The Keyboard

Dear HUG:

Some time ago I read in some publication about the possibility of making a program to configure the printer from the keyboard. After some time for gestation, I finally wrote such a program for my printer (Mannesmann-Tally) in MBASIC. I found that it was generally just too much trouble to use the program because one always had to load the MBASIC and then run the program. It would have been fine if I had been able to compile the program. Finally, I decided to write a similar program in the C-language and the accompanying program is the result.

A single case structure of nine cases is adequate for the selection of the various possibilities, with the possibility of selecting emphasized print appended at the end of the program.

An attempt has been made to make the program friendly so that it will accept either an upper case or lower case answer to any query.

The compiled program has been thoroughly tested and is used regularly when any printing is desired in a single format. It is still necessary to place codes directly in the copy when changes in format are desired in the course of the printing.

A similar program should be possible for any printer, only the codes neccessary would change.

Sincerely,

Randall E. Hamm
NE 1640 Upper Drive
Pullman, WA 99163

```
#include "fprintf.c"
#include    "scanf.c"
#define SCOUT  6

main( )
{
        char a,
        int fout;

        fout=fopen("LP.","w");
        printf("This program is to alter printer configuration");
        printf("of a Mannesmann-Tally printer   First turn on the\n");
        printf("printer and follow the program given.\n\n");
        printf("Would you alter the configuration\n");
                scanf("%s\n",a);
                if(!stromp(a,"Y") || !stromp(a,"y")) {
                        printf("1 draft quality 10 CPI\n");
                        printf("2 draft quality 12 CPI\n");
                        printf("3 draft quality 16.7 CPI\n");
                        printf("4 draft quality 20 CPI\n");
                        printf("5 Double width 5CPI\n");
                        printf("6 Double width 6CPI\n");
                        printf("7 Double width 8 3CPI\n");
                        printf("8 Double width 10CPI\n");
                        printf("9 Correspondence quality\n\n");
                        printf("ENTER SELECTION\n");

                        switch(cget())
                        {
                        case '1':  break; /*M-T is in 10CPI at turn
                                            on*/
                        case '2': putc(27,fout);
                                  putc(91,fout);
                                  putc(53,fout);
                                  putc(119,fout);
                                  break;
                        case '3': putc(27,fout); /* to get 12CPI */
                                  putc(91,fout);
                                  putc(54,fout);
                                  putc(119,fout);
                                  break;
                        case '4': putc(27,fout); /* to get 20 CPI */
                                  putc(91,fout);
                                  putc(55,fout);
                                  putc(119,fout);
                                  break;
                        case '5': putc(27,fout); /* to get DW 5CPI */
                                  putc(91,fout);
                                  putc(48,fout);
                                  putc(119,fout);
                                  break;
                        case '6'  putc(27,fout), /* to get DW 6CPI */
                                  putc(91,fout),
                                  putc(49,fout);
                                  putc(119,fout),
                                  break;
                        case '7': putc(27,fout); /* to get DW
                                                    8.3CPI */
                                  putc(91,fout);
                                  putc(50,fout);
                                  putc(119,fout);
                                  break;
                        case '8': putc(27,fout), /* to get DW 10CPI */
                                  putc(91,fout);
                                  putc(51,fout),
                                  putc(119,fout),
                                  break;
                        case '9':
                                  putc(27,fout); /* to get correspon-
                                                    dence quality */
                                  putc(91,fout);
                                  putc(49,fout);
                                  putc(121,fout);
                                  printf("Would you like 10 CPI?\n"),
                                  scanf("%s",a);
                                 if(!stromp(a,"Y") || !stromp(a,"y")) {
                                        putc(27,fout), /* for 10CPI */
                                          putc(91,fout);
                                          putc(52,fout),
                                          putc(121,fout);

                                  if(stromp(a,"Y") || stromp(a,"y")) {
                                       putc(27,fout); /* for 12 CPI */
                                          putc(91,fout),
                                          putc(53,fout);
                                          putc(121,fout);
```

Continued on Page 82

SEEK.COM

A CP/M Assembly Language Learning Adventure

Part 3

M.D. Zapolski, Sr.
226 N. West Avenue
Bridgeton, NJ 08302

Introduction

Last month, the development of SEEK.ASM made significant progress. We added routines for loading disk files to RAM, error trapping, and displaying messages. In this part, we'll work on the routines that will accomplish our primary objective: counting a text file's lines, words, and sentences. But before starting Module 3, I'd like to provide some insight on my program methodology. You will soon discover that SEEK.ASM goes through the text file 3 times to extract the information it is looking for. You may ask, why not just once? That was my original plan. However, I quickly found that approach impractical as it resulted in routines that were complex, and difficult to manage.

Step 2 — Module 3 (Lines)

Counting the number of lines in a text file is the simplest of the three routines. This is a direct result of the file's internal structure. If you'll recall, each line ends with a CR/LF combination. So, why not count the CRs in the file? Well, that is exactly what the routine does. Chart #4 shows the activities performed by the LINES routine. Upon entry, the routine checks for an EOF marker (1AH). If not found, it then checks for a CR (0DH). Each time a CR is located, a counter is incremented. The routine looks at every RAM location until the EOF is detected.

As noted, a counter is used to keep track of the number of CRs. In brief, a counter is a register that holds a value representing some parameter. But, there is a limited number of registers, and they are frequently needed for other purposes. Consequently, the "count value" is stored in RAM, and recalled when needed. How large should the storage area be? For our purpose 2 bytes is reasonable. This will allow a number up to 65,535 to be saved. Few text files will approach this size. Besides, the TPA is restricted to about 57k. A Data Word (2 byte area) will be satisfactory to cause the assembler to provide this space. So, add the line CTR DW 0 in the DB/BW Area. But thinking ahead, there are 3 routines requiring DW spaces. To satisfy this need, the DW statement should now read CTR DW 0,0,0. The storage locations can then be accessed by the labels: CTR, CTR+2, and CTR+4.

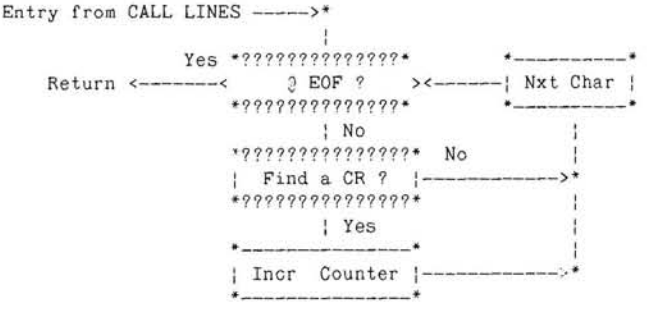


Chart 4
Count Lines Routine

Before entry to the LINES routine, the RAM pointer must indicate the starting address of the text. Therefore, insert the line LXI H,TXTB before the CALL LINES instruction. Now let's analyze the ASM version of Chart #4:

```
NXLN    INX     H
LINES   MOV     A,M
        CPI     1AH
        RZ
        CPI     ØDH
        JNZ     NXLN
FNLN    PUSH    H
        LHLD    CTR
        INX     H
        SHLD    CTR
        POP     H
        JMP     NXLN
```

Having set the RAM pointer to TXTB, the first character is MOVed to the A register. Next, a check for the EOF marker is

made (CPI 1AH). Then LINES tests for a CR. If found, the routine "falls" into a counter incrementing routine (FNLN) which is used 3 times in SEEK.ASM. These routines differ only in their labels, and the operands for the SHLD and LHLD instructions. In the incrementing routine, the first step is to save the RAM pointer's value on the stack since it represents the current search address of the text file. Now, load HL with the value of the counter. Then add 1 to its value, save the new value, reload the RAM pointer into HL, and move to the next text file character location. From this step, the routine continues and is completed at the EOF.

Step 3 — Module 5 (SNTCS)

Although the next module on Chart #1 is Count Words, it is a little too much to handle at this time. To set the stage for it, I'll have to cover some new concepts which conveniently are used in the Count Sentences routine (SNTCS). To aid your understanding of these concepts, Table 2 was developed. It shows the relationship between certain ASCII hex values and the corresponding characters.

ASCII	ITEM
00–19	Control Codes
20	Space (blank)
21–2F	! " # $ % & ' ( ) * + , - . /
30–39	Numbers (0–9)
3A–40	: ; < = > ? @
41–5A	Capital letters
5B–60	[\ ] ^ _ `
61–7A	Lowercase letters
7B–7E	{ | } ~

Table 2
ASCII Hexadecimal Equivalents

Most often, the end of a sentence is identified by the period. However, it can also be identified by an !, ", or ?. So far, it seems clear what to look for to count a sentence. Yet, if these were all we used, the results would be in error. Consider the effect of quoted remarks within a sentence. This could cause 3 sentences to be counted as a result of 2 quotation marks and the period. Not to mention the period which also serves as a decimal point. Therefore, counting just periods, quotes, etc. is not an accurate method. What is needed is to establish "boundary conditions" for the routine such that it will produce an accurate sentence count. To this end, look at some typed sentences. After the end punctuation, the typist inserts 2 spaces before starting the next sentence. This is a key aspect of the SNTCS routine. In addition, sentences in computer text files that end close to the 80th character are terminated with a CR and LF; and thus, may not include the 2 spaces. Using this information, Chart #5 was developed. As before, an EOF check is performed. Next, a "filter" is used to strip away undesirable ASCII values. In this case, only ASCII values >20H are allowed to reach the next test. This action removes the control codes and spaces. Following this, all ASCII values >3FH (letters) are stripped away. Now we can test for the end-of-sentence punctuation. If found, the RAM pointer is moved ahead 2 bytes. That location is then checked for a CR, LF, or a space. If one of these 3 is detected, then count 1 sentence. If the rationale does not seem clear, test some examples. Remember to consider the CR and LFs inserted by the editor program.

The ASM version of Chart #5 is:

```
NXSN  INX   H              |FND   INX   H
SNTCS MOV   A,M            |      INX   H
      CPI   1AH            |      CPI   20H  ;a space ?
      RZ                   |      JZ    FDSN
      ADI   0DFH           |      CPI   0DH  ;a CR ?
      JNC   NXSN           |      JZ    FDSN
      MOV   A,M            |      CPI   0AH  ;a LF ?
      ADI   0C0H           |      JNZ   NXSN
      JC    NXSN           |FDSN  PUSH  H
      MOV   A,M            |      LHLD  CTR+4
      CPI   21H   ;ck for ! |      INX   H
      JZ    FND            |      SHLD  CTR+4
      CPI   22H   ;ck for " |      POP   H
      JZ    FND            |      JMP   NXSN
      CPI   2EH   ;ck for  |
      JZ    FND            |
      CPI   3FH   ;ck for ? |
      JNZ   NXSN           |
```

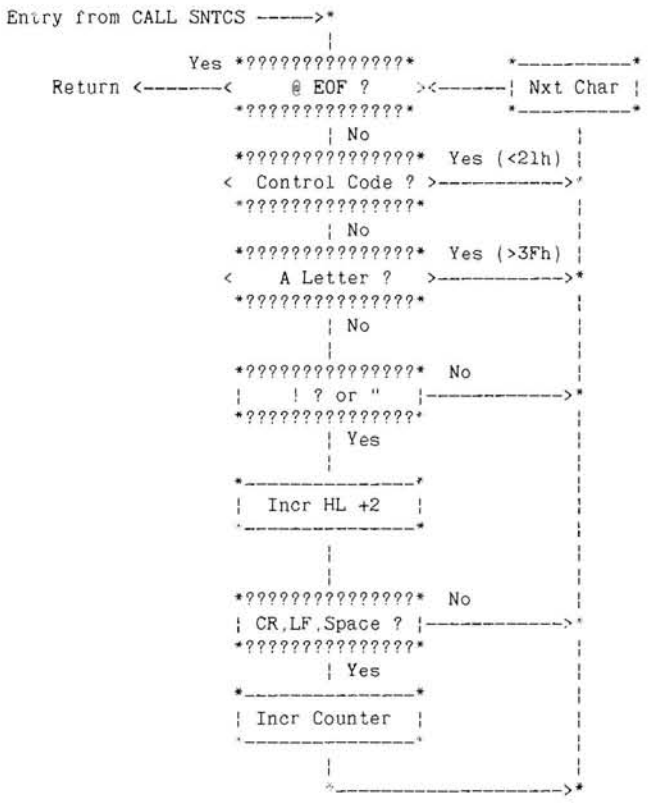


Chart 5
Count Sentences Routine

After passing the EOF check, SNTCS filters out all characters <21H. This is accomplished by ADDing the contents of the A register to a value that sets the Carry Flag (CY) when their sum exceeds a limiting value. In this instance, the limiting value is 0DFH (0FFH–020H). Although this method satisfies the program's need, it also changes the value in the A register. Consequently, the A register may require reloading before the program continues. The next test operates in a similar manner, except it filters out characters >3FH. So, its limiting value is 0C0H (0FFH–03FH). Also take note that the first filter operates (allows access to the next filter/test) by setting CY, whereas the second operates when CY is not set. The action of these filters creates the mathematical equivalent of 20H< X <=3FH. Where x is the ASCII value of the character. Now SNTCS can test for the key punctuation (!."?). If detected, the RAM pointer is advanced 2 bytes and a test for the CR, LF, or space is conducted. When all tests are satisfied, the counter is advanced, and the routine continues to the EOF. At this point, you may wonder why it is necessary to filter out any characters prior to the key punctuation test. Good ques-

tion! In actuality, the routine would work without the filters. But, there are considerably more characters in a text file than key punctuation. Couple this with the fact that each ASM instruction takes some finite amount of time, and the result is wasted time. Hence, the net result of the filters in this routine is to reduce program execution time. This is always a programming goal.

Step 3 — Module 4 (Words)

The final counting routine is Words. The most difficult aspect of its development was analyzing the many written variations of a word. This resulted from one of my boundary conditions: certain numbers should be counted as a word. For example,

1. #45
2. 10%
3. $1.00
4. $1,000.00
5. 1/2

In addition, the routine should accurately count hyphenated words, and words at the end of a sentence. If it were not for these conditions, Words would be simple. It could look for the spaces between words, and the punctuation at the sentence's end. Although Words counts words with a reasonable degree of accuracy, it will not correctly count letter/number combinations like PNJ20A/51Z. Words would see this string as 2 words: PNJ2 and 02A/51Z. These conglomerations are often found in part or model numbers, and infrequently in the average text file. Lastly, you'll find that referring to Table 2 for the remainder of this discussion will be quite helpful.

Chart #6 is the flow chart for Words. As with the previous routines, Words first checks for the EOF marker. Next, like in SNTCS, all control codes and spaces are filtered out. From this point, Words branches to one of 2 smaller routines. Essentially, it decides whether the character is a letter or a number. And if you think about it, when the first character of a "word" is a letter or number, then the remainder of it consists of letters or numbers. Of course, this is not an absolute rule. But, it is close enough. For example, examine the string "I live at 20547 Grandville, Detroit, Michigan". In each word and number sequence, the observation made above holds true. And, the end of a word is identified by a space, or punctuation. A hyphenated word is an exception to this "rule". As a result, some special treatment will be necessary.

Assume the letter string "Grandville" has been encountered. Since ASCII for "G" is >39H, Words branches to the LETTERS side. The RAM pointer is advanced 1 byte, and a test for a hyphen is performed. Since no hyphen exists, the routine now determines whether this second character is another letter, a space, or some punctuation. When the next character's ASCII value is >40H, the RAM pointer is incremented, and the routine continues until the comma (2CH) is found. At this time, the routine counts "Grandville" as a word.

The NUMBERS side of Words contains a punctuation filter which removes these characters. This is necessary since only numbers should get to this side of the routine. Assume the number string "$1,000.00" is being examined. The first character to get past the punctuation filter is the "1". Then the next character is checked. If its ASCII value is >20H (a space), the RAM pointer is moved forward 1 byte, and the next character checked. This process continues until the space is found. The WORDS counter is then incremented.

To summarize, Words decides whether a letter, or number string has been detected. After this decision, it enters a loop which inspects RAM addresses until a delimiter (space, CR, LF, etc.) is found. Thereupon, it counts 1 word and continues through the text file until the EOF.

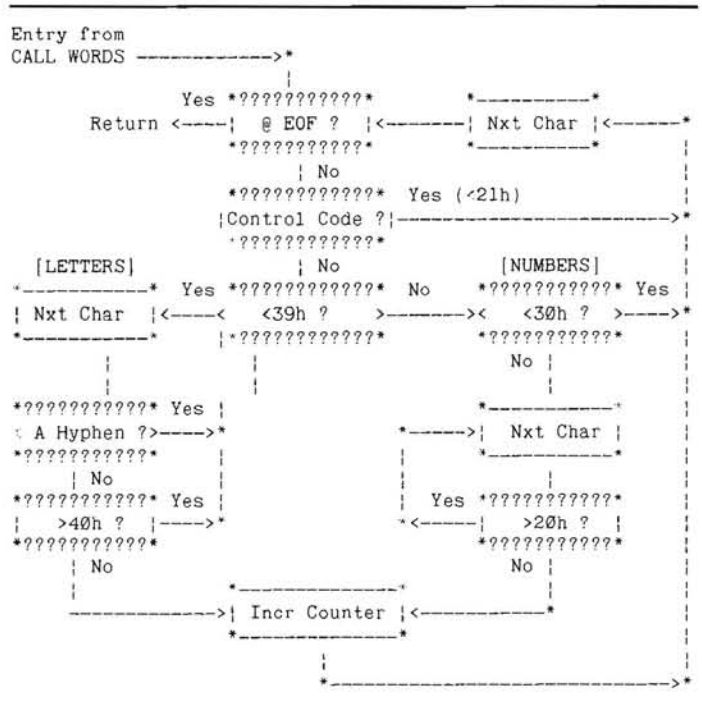


Chart 6
Count Words Routine

Now, let's look at the ASM version of Chart #6:

```
NXWD  INX   H                 |NXCR  INX   H
WORDS MOV   A,M               |      MOV   A,M
      CPI   1AH  ; @ EOF ?    |      CPI   2DH  ;a hyphen ?
      RZ                      |      JZ    NXCR
      ADI   0DFH ; >20H ?     |      ADI   0BFH ; >40H ?
      JNC   NXWD              |      JC    NXCR
      MOV   A,M               |FDWD  PUSH  H
      ADI   0C6H ; >39H ?     |      LHLD  CTR+2
      JC    NXCR              |      INX   H
NXNM  ADI   0AH  ; <30H ?     |      SHLD  CTR+2
      JNC   NXWD              |      POP   H
      INX   H                 |      JMP   NXWD
      MOV   A,M               |
      ADI   0DFH ; >20H ?
      JC    NXNM
      JMP   FDWD
```

Considering the last two routines (Lines & SNTCS), there is only one new item in the Words routine. That item occurs between the ADI 0C6H and the JNC NXWD after the NXNM label. Before this routine, anytime the ADI was used, the A register was restored by a MOV A,M. But in this instance, you'll see A isn't restored. Although the values in the A register are changed by the ADI 0C6H instruction, they are changed by a specific amount. Since we know that value, we can calculate a new number which represents the A register's original value. An example will help you understand my meaning. First, look at the following table:

	A Register Values	
Orig	ADI 0C6H	ADI 0AH
00	C6	CF
2F	F5	FF
30	F6	00
39	FF	
3A	00	

If A initially contains 00H, it will be C6H after the ADI 0C6H instruction. Then, if you add 0AH to A, it will contain CFH. Since the limiting value needed is 30H (to filter punctuation <30H), we seek a value when added to F6 that will set CY. The value is 0AH

(FFH–F6H). From this table you can also see how the other limiting values set CY.

All other concepts in the Words routine have been covered in previous sections. So there doesn't seem to be much value in restating these processes. Suffice it to say that Words steps through the text file counting words, and certain number strings, until it reaches the EOF.

Summary

With the 3 text counting routines added to SEEK.ASM, we are close to completing the program. It will now open a disk file, load it into RAM, and count the text file's attributes. Further, it will trap file opening and loading errors, and display the appropriate messages. If it were not for one other small need we would be through. Of course, the program hasn't told us what the count values are. And it won't until we make a routine that takes the hex values, converts them to decimal numbers, and displays them. With this object in mind, you should refresh your memory on how to convert numbers from hex to decimal. How might the program do it?

Finally, integrate these routines into SEEK.ASM after the EPGM label, but before PRNSEQ. As before, assemble, load, and test the program. Use DDT.COM to follow the program steps, and look at the count values. The more DDT is used, the more useful it becomes. It is a valuable tool for testing ASM programs. Until next month . . .

*

A Hi-Tech File Handler

Robert Pearce
Pearce Farm Software
2325 West Cabana Circle
Mesa, AZ 85202

Over the past decade, we have read several articles on sequential and random access file handling. This file handler is both.

I call it DISAM. That is Dynamic Indexed Sequential Access Method. I took the logic from the IBM main-frame world. IBM calls it VSAM, Virtual Storage Access Method. That is the access method used when virtual storage was introduced by IBM. The access method itself has nothing to do with virtual storage.

DISAM is a modified form of VSAM. You may retrieve records either sequentially or by using a predefined key. If you read sequentially, the records are read in ascending key order. If you read by using the record key, you get that record. You may even use a shortened form of the key if you wish.

What is this key? The DISAM record key is a character key located somewhere within the record. It may start at the beginning of the record or in the tenth position (offset) or at the end of the record. DISAM doesn't care as long as the key is somewhere in the record. The key may be up to 32 characters in length. I have found that a length between 9 and 11 characters is enough to find a name in a mailing list file. If the character string is longer, DISAM doesn't care either. DISAM only uses what you define to be the key length. The rest becomes part of the record. If the actual key is shorter, then you must pad out the rest of the key with spaces.

Some of you may remember HEXAPAWN. I rewrote it for a 4 X 4 board. The original 3 X 3 board used an incore array. This array contained a 9 byte board map and possible moves. In 8 to 10 games, HEXAPAWN was winning all the games. In the 4 X 4 version, it took many more games to get smart so I used DISAM and defined a 16 character key. Again, the board image followed by possible moves. There are currently 163 records in the DISAM file and the game is getting smarter, but there are so many more possible moves that I get tired of playing before it gets in a winning mood.

What about the record? The record length is limited to what BASIC can handle. (DISAM was written for BASIC access.) 255 byte records. Actually, two characters are needed to tell DISAM what you want to do so the real limit is a 253 byte record preceded by a one character DISAM instruction and a comma, 255 total. However, the record may be shorter. As short as the record key plus the key offset. Basically, if you define the key at the start of the record, offset 0, then the record may be as short as the key. If you define the key at offset 20, then the shortest record will be 20 plus the key length.

There is NO space wasted. Records are packed together the same way as sequential files. Even though this is a keyed access file handler.

To recap for a minute. DISAM reads records by key or sequential access. The key may be from 1 to 32 bytes in length. The record may be from the key length plus key offset to 255 (253 in BASIC) bytes in length. And DISAM was written for BASIC.

Since DISAM is a smart file handler, the file must be defined before it can be used. There is a utility called DISAM.ABS that does file definition.

Let's look at the DISAM file structure. DISAM is really two files in one. First, there is an index file. This file locates the data record. Then, there is a data file. This is where the data record is. The data records are stored in key sequence order in the data file. The index file contains the key of the highest record value in a data file block and the address of that block. In a nut shell, you can locate one record in 150 in two disk reads using a key length of 11 bytes and an average data record length of 79 bytes. Or 1 in 300 with three disk reads max. This makes file access by character key very fast.

In detail, DISAM works this way. When you initially define a DISAM file, a dummy record of high values is stored in the data file. This dummy record of hex 'FF' values is key length plus key offset long. A key of hex 'FF' key length long is placed in the index file.

When you add a record to the file, the file handler looks for an index record equal or greater than the record you are adding. The Index record points to a data block. That data block contains records that are greater than your record key. Your record will be inserted in that data block. If this is the first record to be added to the initialized DISAM file, the only key in the file is the hex 'FF' key and your record key will be less than that. So your record will be inserted below the 'FF' data record.

If you add a second record, it will be inserted below your first if the record key is less that your first, or below the 'FF' record if the record key is greater that your first record. This goes on until the data file block is full. Then, the block is split. More on that later.

Now let's read the DISAM file. When the DISAM file is opened it is positioned at the start of the data file. If you request a record without specifying a key, DISAM will return the first data record and reset the pointers to the next record. You may continue to

issue unqualified reads and DISAM will continue to return records in key sequential order. You may use a generic key to position the data file. Suppose you issue a read for record 'C'. Assuming that the key length is greater that 1 and that there are some records whose keys start with 'C', you will get the first record with a key that starts with 'C'. From there, you may issue unqualified reads and sequentially read all of the 'C' keyed records and into the Ds, Es, etc. Suppose you want a specific record. Then, you issue a read with a fully qualified key and get that record.

Now let's delete a record. You must use a fully qualified key with the delete request. DISAM will find the data record and slide all of the records in the data block beyond the deleted record down over it removing any knowledge that the record ever existed. This allows other inserted records in the data block to reuse the vacated space. What if a new record insert is longer than the available space? The data block splits.

How about replacing an existing record that is NOT the same length as the current record. DISAM combines the delete and add functions to replace existing records. Replacement records do not have to be the same length. DISAM is dynamic.

So far we have covered the insert, read, delete and replace of a data record in DISAM. Now for some internals.

When a file block becomes full, be it an index block or a data block, it splits. Half of the information on the full block is moved to a new block. The file is expanded. In the case of an index block only the block chaining is effected. In the case of a data block, the block chaining is effected and a new index record is created for the index block.

It is entirely possible that a record add finds too little space in the data block. The file handler checks for key record space in the index block and finds not enough. The index block splits first creating space for a new index record. Then, the data block splits creating space for the new data record and adding an index record. Lastly, the data record is inserted and control is returned to the user. This is a worst case happening. It costs user time and it can be minimized.

Now let's assume that we have added many records to the DISAM file. While we were adding these records, the file split many times. This is a normal occurrence. However, some of the free space created with the splits was never used and now it should be removed. DISAM.ABS is the utility for this purpose.

The DISAM utility provides four major functions. It allows you to 'dump' the file to a sequential device; a printer if you want to scan the file data; or to the terminal; or to a disk device for file reorganization. This is what we want to do now, reorganize the DISAM file, remove the extra free space, and condense the file.

So, using 'dump' and the prompts, we write the file to a sequential file in key sequence order. In the process of defining how to dump the DISAM file you are asked how much to dump. The dump function allows you to dump from a key, from a count, for a count, or even one record. Normally, you would specify all of the records starting at the beginning of the file. The default sequential extention is .DMP so you can use the same file name.

After the dump is complete, you are returned to the menu and we select the 'purge' function. This deletes the DISAM file. Protection was added so that you cannot define an existing file. The purge function returns you to the menu and we select 'define'.

We redefine the DISAM file as we originally did and now we are down to 'load'. In the load function, we are asked "How much free space should be left in each data block?". This is load free space. The intent here is to provide for record length changes through updates without causing block splits. If your average record length is 80 bytes and you are going to add more records, then allow enough space for another record in each data block – 80 bytes. If you specify anything other than zero, then space is left in the index blocks, as well.

If you specify zero load free space, then the records will be packed as tightly as possible and use the least amount of file sectors. This runs about 3 to 5 sectors over a sequential file depending on the key length and the number of records in the file. After the load, you may purge the .DMP file.

You may initially load the DISAM file via the 'load' function if the sequential file has been sorted into key sequence order through a sort program. There is also a DISAM load utility that will load an unsorted file, but since it uses the insert logic, you must reorganize the file after the load to reduce the space used. It splits all over the place during the load and the file could end up using twice the normal space required.

I mentioned that DISAM was written for BASIC. There are only three (3) required BASIC lines of code needed to run DISAM under BASIC. In reality more should be used for error checking.

The three are:

1. Define the entry point

```
nnn DEF USR=((127 AND PEEK(8535))*256+PEEK(8534)) OR &H8000
```

   The entry point to the DISAM file handler is placed at 41.126 when it was loaded. (LOAD FH:)

2. The open of the DISAM file

```
nnn R$=USR("0,[CHANNEL],FILENAME"+"")
```

   The 'O' is the open command. The channel may be specified or allowed to default to channel 5. BASIC will never know. The '+""" is to force BASIC to create a new string descriptor so your program won't be modified if there is an error.

3. The file access

```
nnn R$=USR("X,"+REC$)
```

   X may be any of the DISAM file access functions: A = add, D = delete, G = get, P = put, C = close file. Get and Delete require record keys. (REC$) Add and Put require the data record. (REC$) DISAM will get the key. Close requires nothing.

When the file is opened, R$ will be 9 bytes long. The format is RC,KEYLEN,KEYOFF. RC is the return code. KEYOFF is key offset from the start of the data record and KEYLEN is the key length. Your BASIC program does not need to know this when it is written. Therefore, one program may access many DISAM files one at a time. In all other functions, R$ either contains a return-code or the data record. 1,2,3 that's all there is to accessing a DISAM file from BASIC.

If you are a HUG member and have access to a modem, you can download the hex files for the DISAM utility and file handler. They are: DISAM.DOC, DISAM.HEX, and FHDVD.HEX. If you want the source code and all the in's and out's of DISAM, that's $20.00 from Pearce Farm Software, 2325 West Cabana Circle, Mesa, AZ 85202.

I hope that you have learned a little more about what is currently in use in the world of the big machines. There is no need for micro users to be stuck with sequential and random access only. DISAM is available for HDOS 2.0 only at this time. ✱

CAD On A Shoestring

Pierre D. Olivier, III
2019 Marengo Street
New Orleans, LA 70115

How would you like to design printed circuit boards on your H/Z-100 PC? How about blueprints for a house or a machine part? Possibly you want to design business logos. All of these are available with ProDesign II from American Small Business Computers. ProDesign II is a budget-priced ($300) CAD program with lots of power. If you think $300 isn't budget-priced, compare that with smARTWORK from Wintek Corporation ($895, advertised for printed circuit work ONLY on the IBM ONLY) or AutoCAD ($1500 or $2000, depending on how many extensions are purchased). ProDesign II features optional 8087 support (included at no extra charge), layering capabilities, and the ability to define and store shapes for later incorporation in a drawing.

ProDesign II is a "word processor for graphics". It has all the capability and complexity of WordStar but lacks the on-screen menu. It is not yet easy for me to use, although basic operations such as marking points and connecting those points with vectors (lines) aren't difficult to perform. It's when you get away from the basics that the problems arise. This is apparently inescapable in a program of this scope. If a menu were used such as in WordStar, there wouldn't be any room for graphics. It took me six months to learn how to get by without the WordStar menu and I expect a similar time frame to achieve full competency as a ProDesign user.

The User's Manual contains 3 pages of contents listing and 5 pages of indexing. I have found it convenient to use the manual as a reference, rather than as a tutorial. This is a book one READS on those nights when sleep doesn't come easily. Browsing through it prior to using the program is helpful, but don't feel that you have to memorize it. ProDesign uses all aspects of the keyboard, but an 8-1/2" x 11" reference card is provided. Using this card in conjunction with the index is the method I prefer for learning this program. Commands include: Arc, Backward Arc, Curve, Section Delete, Erase, Fillet, Color, Status Line Format, Overlay, Layer, Section Move, Section Copy, Oval, Paint, System Status, Rotate, Text, Units, Vector, Line Width, Section Save, Clear Screen, Zoom, Crosshair, Rubber Band Line, Rubber Band Box, Box, Arrow, Semi-Circle, Quarter-Circle, Set Point, Gravity Move, Gravity Point, Change Point, Auto Dimension, Attribute, Repeat Section, Line Type, Help, Screen Window, Print, Plot, Directory, Stop, Retrieve, Save, Undo, and Oops. It should become clear from this list that this is no piker of a program. Examples are given on how to use each command. There are four lessons provided, using one of the sample pictures provided on Disk II. There is also a section on using a digitizing tablet and one devoted to such special activities as creating mirror images.

ProDesign comes on three disks. Disk I is the basic ProDesign program and its supporting files, overlays, and drivers. It has 0K left on it, so in order to make a bootable disk, I had to split the files into two disks when copying the programs. I use one for editing and the other for printing/plotting. The second disk contains sample drawings created with ProDesign, such as a Space Shuttle and a B-1 Bomber. Disk III is the 8087 version of ProDesign. Not having an 8087 installed, I can't offer speed comparisons, but when I do get one (it's high on my priority list), I'll submit some benchmarks.

The first thing to do with ProDesign II (after making backup disks and working disks) is run the PDSETUP program. This configures ProDesign to recognize the peripheral(s) you may have connected to your computer. ProDesign supports a wide variety of printers, plotters, mice, digitizers, and graphics boards. A comprehensive list is provided at the end of this article, for your convenience if you feel that ProDesign would be of benefit to you. You may already have some or all the devices supported. ProDesign also has an internal configuration option that lets you set default conditions such as how far to step the cursor, "snap" function, grid display, and grid size if implemented. ProDesign is NOT copy protected, so you can backup to your hearts content. It may be low-cost in comparison to other CAD programs, but $300 is definitely not low-cost to my wallet. The ability to protect my investment with backups means a lot to me.

ProDesign requires 512K RAM (as compared to 256K for smARTWORK and 328K for AutoCAD) and two disk drives. The extra ram requirement still yields a less expensive total package when compared with smARTWORK or AutoCAD even if you use a second 320K board instead of 256K chips such as I have installed. On the subject of disk drives, a hard disk would be most desireable. I already mentioned the need to split my working disks into two separate floppies.

With the Z-309 video card, you get four colors on screen: black (background), light cyan, light magenta, and white. You can

change colors at any time, but regardless of which color you select from the available palette of 0 - 15, those three are the only ones shown on screen. Not having access to a Z-319 for evaluation, I can't even guess how well that would work with this program.

Cursor control can be accomplished with either the arrow keys or by the use of a mouse or digitizer. I have a Logitech Logimouse, which is supported, but I find I have more precise, if somewhat slower, control with the keyboard.

Hardcopy output is available from plotter and/or graphics-capable printer. A test output to my Epson RX-80 printer resulted in slight misspacing (the test subject was a foil pattern for a 40-pin dip; Figure 1). The pads are not all the same size. This is due to the inherent resolution of the printer rather than the accuracy of the program. The printer has a vertical resolution of 72 dots/inch, which doesn't "divide out" evenly for the 1/10th inch spacing used on dips. My plotter is in New Orleans at this writing and was unavailable for use in the test. I would recommend printer hardcopy only for those subjects not requiring extreme accuracy in dimensioning. Circuit boards would require printing five times their actual size in order to get a correct relationship between horizontal and vertical resolutions. This is possible, but would severely limit the size of the board on a standard sized printer, the maximum print width is 8". One fifth of that is a circuit board 1.6 wide. Even the wide carriage printers have only about 13" of print width. The largest circuit board size would be around 2.6" wide. In applications where an absolute relationship between vertical and horizontal dimensioning is not required, printer output should be satisfactory. Other printers may not have this problem — the only ones I'm familiar with are the Epson RX-80 and a couple that aren't supported by ProDesign.

Following is a list of devices supported by ProDesign. I have omitted some redundant entries — those that had more than one "mode", for instance. The list is presented by manufacturer and then by model. If the model number sequencing seems a bit illogical, it is because PDSETUP presents the selections grouped with other models using the same control commands.

Printers

ANADEX	6500, 9000, 9001, 9500, 9501
BROTHER	1509
CITIZEN	MSP-10, MSP-20, 120-D, MSP-15, MSP-25
C. ITOH	7500, 8510, 1550, PROWRITER JR., PROWRITER, 8510 AP, PROWRITER II, 1550 AP
COMREX	CR-420
DATA PRODUCTS	8010, 8020, 8050, 8070
EPSON	FX-80, FX-85, FX-100, FX-185, FX-286, LQ-800, LQ-1000, LQ-1500, SQ-2000, LX-80, LX-90, MX-80, MX-100, RX-80, RX-80 FT, RX-100
FUJITSU	DOTMAX 24
HP	LASER JET, THINK JET
IBM	COLOR PRINTER, GRAPHIC PRINTER, PROPRINTER
IDS	PRISM 80, 480, PRISM 132
JDL	750
JUKI	5510
MANNESMAN TALLY	85, 86, 160, 180, 290, 460, 490
MPI	150G
NEC	8023, P2, 8025, P3, P5
OKIDATA	84 STEP-2, 2410, 92, 182, 192, 93, 193
PANASONIC	1090, 1091, 1092, 1093
PRINTEK	930, 935
PRINTRONIX	4160
RADIO SHACK	CGP-220, DMP-130, DMP-430, DMP-2100
RITEMAN	10, 15
STAR	GEMINI 10, DELTA 10, GEMINI 15, DELTA 15, RADIX 10, RADIX 15, SG-10, SD-10, SR-10, SG-15, SR-15
TI	855, 865
TOSHIBA	1340, 1341, 341, 351, 1350, 1351, 1352

Plotters

ALPHAPLOT	I, II
AMDEK	AMPLOT II, DXY 100
CDS	2300 LASER PRINTER
C. ITOH	CX-4800
ENTER COMPUTER	SWEET-P 100, SWEET-P 600, 1200
EPSON	HI-80
FACIT	4550, 4551
GOULD	6310, 6120, 6320
HP	7220, 7440 COLOR, 7470, 7475 (A- & B-SIZE), 7550 (A- & B-SIZE), 7580B (B-, C-, & D-SIZE), 7585B (B-, D-, & E-SIZE), 7586B (B-, D-, & E-SIZE)
HOUSTON INSTRU.	595, 695, 29, 40, DMP-41, DMP-42, DMP-51, DMP-52
IBM	7371, 7372 (A- & B-SIZE), XY 749, XY 750
IOLINE	LP 3700
NICOLET	ZETA-8
PANASONIC	VP-6801
PENMAN	
RADIO SHACK	FP-215
ROLAND	DXY 101, DXY 800, DXY 880, DXY 980
TAXAN	KPL 710
TEKTRONIX	4662
WESTERN GRAPHTEC	MP-1000

Digitizers/Mice

ALPHAPLOT	
CALCOMP	2000/2100 (ASCII or BINARY)
GEOGRAPHICS	DIGITIZER
GTCO	DIGI-PAD, MICRO DIGI-PAD
HITACHI	TIGER (ASCII or BINARY)
HOUSTON INSTRU.	HIPAD
IMSI	MOUSE
KRAFT	MOUSE
KURTA	PEN MOUSE, SERIES ONE, SERIES TWO MODEL 400, SERIES TWO MODEL 1000
LOGITECH	LOGIMOUSE
MICROSOFT	MOUSE
MOUSE SYSTEMS	PC MOUSE
NUMONICS	2200
SAC	GP-7, GP-8
SUMMA	MM-1201 (ASCII or BINARY), MM-961 BINARY, BIT PAD ONE
TANDY	GT-2000

Graphics Boards

IBM	STANDARD GRAPHICS (320 X 200, 4 COLOR)

IBM	STANDARD GRAPHICS (640 X 200, 2 COLOR)
A T & T	6300 (MONOCHROME)
HERCULES	GRAPHIC BOARD (MONOCHROME)
IBM	ENHANCED GRAPHICS ADAPTER (4 COLOR, 64K)
IBM	ENHANCED GRAPHICS ADAPTER (16 COLOR, 128K+)
IBM	ENHANCED GRAPHICS ADAPTER (MONOCHROME)
PERSYST	BoB BOARD, EXTENDED COLOR GRAPHICS
PERSYST	BoB BOARD, EXTENDED MONOCHROME GRAPHICS
SIGMA	400 BOARD (640 X 400 COLOR)
TANDY	2000 (COLOR)
TANDY	2000 (MONOCHROME)
TANDY	DLX GRAPHICS DISPLAY ADAPTER
TECMAR	GRAPHICS MASTER (COLOR)
TECMAR	GRAPHIC MASTER (MONOCHROME)
*	3270 PC (COLOR)

* - NO MANUFACTURER NAMED IN MENU - IBM?

Inclusion in these lists is not intended as a recommendation, or as an indication that the named device is compatible with the Z-100 PC system. Items listed are presented as an indication of those compatible with ProDesign II. For example, I doubt highly that any of the Tandy graphics boards would work in the Z-151, since they are designed for a different buss. It is a fact that some Zenith users could also be Tandy or IBM or Compaq users, and it is for their benefit that the entire list is provided. Also, of those that do use the same buss as the Zenith and IBM, some are known to be compatible while others are questionable. I don't have the facilities and/or resources ($$$) to test them.

I am quite happy with the price of ProDesign. I am happy with the ability to protect my investment through unlimited copying. I am less than satisfied with the printer output, but I knew the Epson wouldn't handle that degree of precision before I bought ProDesign. If your needs include a CAD program for small-to-medium scale projects, I think you'll find it an excellent value for the money.

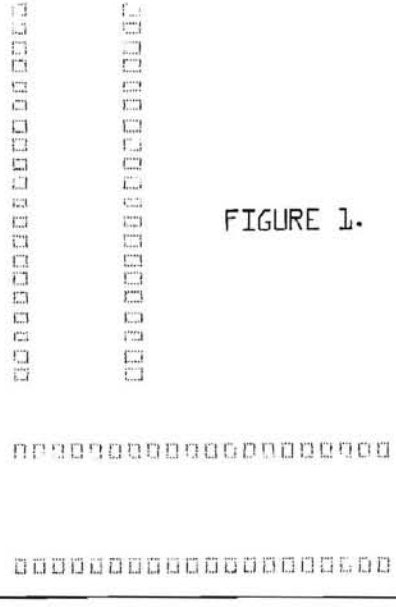


FIGURE 1.

Controlled Data Recording Systems Inc.

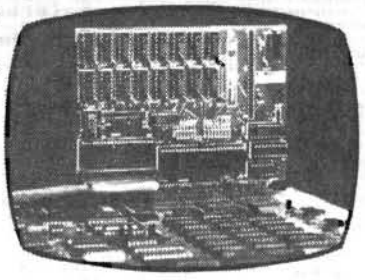
Quality Products and Support for the Heath/ Zenith Community

New for the H/ Z89-90 Computer Users:

The Super RAM 89 Package

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.



Now get RAM 89

HDOS DRIVER Software Only **$35**

RAM 89 DVD supports RAM SY0: and SY1:, added autoload features, plus more!

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.

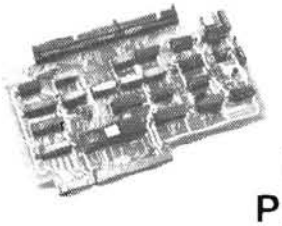
The Ram Drive Software (SRAM) allows one or two logical ram drives. The ram drive(s) can be located starting anywhere from logical A: to O: (standard drives get relocated). SRAM can be set to start at logical A: and warm boot with ram (no floppy disk accesses needed). Ram drive attatches to any of the 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 256 k 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.

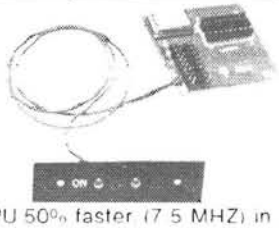
Z100 HARDISK CONTROLLER

The CDR317 is the ULTIMATE IN HARD DISK controllers for the H/Z110- H/Z120 computers

- Direct replacement for the Heath/Zenith Z217 hard disk controller package.
- Able to work with standard ST506/412 type Winchester Drives and the cartridge removable Syquest Drives.
- Works with all standard Heath/Zenith drivers, operating systems, and Z217 Boot Roms.
- Is an intelligent SCSI host adaptor for future network expansion.
- Less expensive than the Z217 controller.
- Available now through your local Heath/Zenith dealers.

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.

SUPER DRIVES 85

A powerful floppy drive package for the Z100 or any computer that can use 8 floppy drives.

Two extended technology 5.25 drives in a package with interface cables, case and power supply, that look exactly like standard 8 floppy drives to the computer. Now have **2.4 megabytes of floppy file storage capacity** in one 5.25 package.

ONLY **$495** LIST

Dealer inquiries invited

For information about these products contact
Controlled Data Recording Systems Inc.
7216 Clairemont Mesa Blvd.
San Diego, Ca. 92111 Phone (619) 560-7727 or contact a Heath/Zenith dealer that carries the C.D.R. Systems Inc. line of products.
Now... C.D.R. Systems Bulletin Board (619) 560-8929

A Winchester For The '89

Part 8

Peter Ruber
P.O. Box 502
Oakdale, NY 11769

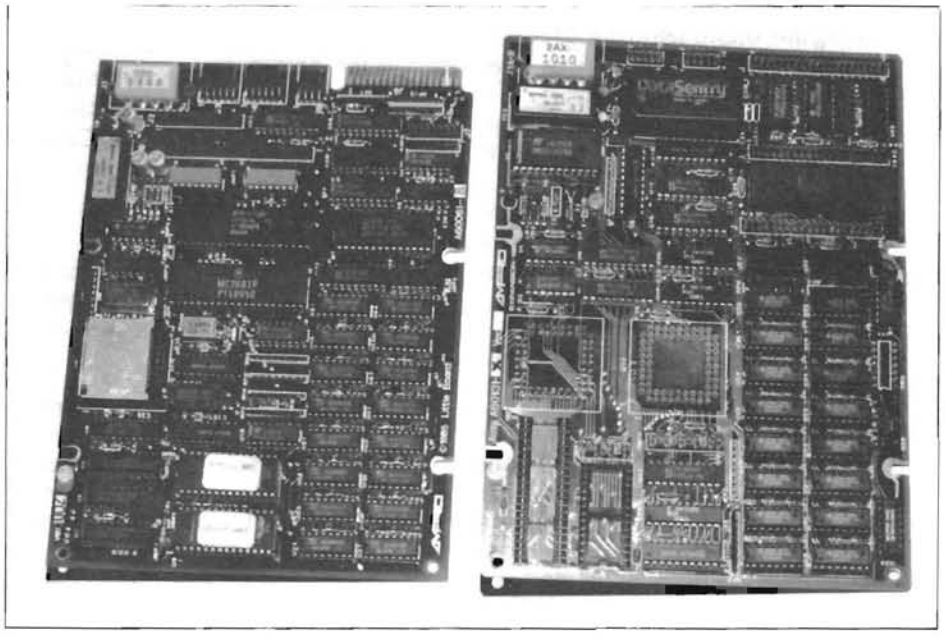


Photo 1

When the Ampro 186 PC-DOS board (left) is mated with the EXPANSION/186 (right), you have a very formidable IBM compatible computer for your '89 or your H-19 or H-29 terminals. The features include an 8 MHz 80186 CPU, 4 serial ports (2 of which are programmable for RS232 or RS422 Sync/Async), a Centronics parallel port, floppy disk control for 4 40- and/or 80-track drives, an SCSI hard disk controller and interface for up to 64 devices, a non-volatile programmable clock, 1024k RAM and a provision for a 10 MHz 8087 Math co-processor and an 82188 bus controller. New software now allows the networking of 4 186 PC-DOS boards under CONCURRENT DOS, and a simple serial interface connection allows it to use an IBM compatible computer as a terminal or to act as a co-processor for multi-tasking.

When you were left hanging with the previous installment in this series, I promised you a full review of the Ampro 186 PC-DOS Little Board computer. Even though I had this board installed in my '89 when I wrote about the Ampro CP/M Little Board, I hadn't as yet had the time to do a thorough check of the hardware and test out an armload of software. This process has taken more than two months, during which time I made a pest of myself at Ampro's offices and a variety of software houses.

Of all the co-processors available for the '89 (or your H/Z-19 or H/Z-29 terminal), this is by far the fastest, cheapest, and most sophisticated piece of hardware on the market. So, if you're planning to sell your '89 because you've just purchased a new Heath/Zenith PC computer, the best advise I can give you is — DON'T.

The Ampro 186 PC-DOS single board computer system offers all the features of the Ampro CP/M Little Board/PLUS computer: 2 Serial Ports, a Parallel Port, an LED Monitor/Reset switch, a connector for 4 Floppy disk drives, an SCSI hard disk connector, and a user ID header for networking applications. All sockets and headers are in the same location, even the physical size is approximately the same. It was also designed to mount directly to a standard 5.25" floppy drive, but that's where the relationship ends.

What we have here are nearly all the components of a full-blown IBM compatible desktop system that you can hold in the palm of your hand. The 186 board has even fewer components than the CP/M Little Board, and it is an exciting example of Ampro's technical wizardry. However, the aspect I liked best of all is that Ampro continues to provide support for the durable Heath/Zenith terminals.

Where Are The ICs?

At first evaluation, stuffing an 8-MHz 80186 CPU onto the 186 Little Board is like trying to cram the engine of a Sherman Tank into a Volkswagon. The 80186 CPU is a highly advanced integrated microprocessor (in a 64-pin quad in-line package) that can eliminate some 15-20 support chips found in most standard 8088 IBM-compatible systems. It is a "true" 16-bit CPU, that includes 2 independent high-speed DMA channels, three 16-bit timers, a programmable interrupt controller, as well as programmable memory, I/O chip-select logic and system bus control. It offers twice the performance of the 8086; it has a 4 MByte/Sec bus bandwidth interface, and can directly address 1 MByte of memory. It is completely Object Code compatible with all existing 8086 and 8088 software through an extended instruction set. It is mounted in an exotic Textool Socket with a metal heat-shield, because it gets so hot that it will burn your fingertips.

The remainder of the prime ICs number only three. A Western Digital 1772 Floppy Disk Controller, a Signetics 2681 Dual Asynchronous Receiver/Transmitter (DUART) provides a programmable interface for the 2 Serial Ports and the Parallel Printer Port. Lastly, there is the NCR 5380 SCSI Controller for hard disk interfacing and networking applications.

Two 2764 programmed ROMS contain a highly compatible emulation of the IBM BIOS, a couple of custom PAL and logic chips round out the circuitry. My board arrived with 512k memory in the form of two banks of 41256 ICs. Ampro also offers a 128k version for $50 less, but it isn't worth considering for such a small price spread. Besides, most of the important software is so memory-intensive that 512k is the minimum configuration to have.

Interfacing the Ampro 186 to the '89 computer or the H/Z-19 and '29 terminals is exactly the same as I described in the Ampro CP/M Little Board/PLUS article (Part 7 in this series). All connector pin designations are the same on both boards. The cables I constructed out of spare parts for the CP/M board worked for the 186. Over protracted use, however, I noticed that my makeshift assembly had some loose contacts which caused oxidation to form on some of the pins due to the damp coastal climate on Long Island. This was rectified when I ordered a neat harness assembly from Ampro for $36. This had the proper plugs and connectors for the 2 Serial Ports and the LED Monitor and Reset Switch. I was not aware of this when I reviewed Ampro's CP/M board and hasten to mention it now.

The current Ampro ROM BIOS was enhanced to be compatible with IBM'S PC-DOS 3.1, allowing for more than the standard two floppy drives on a system. Additional floppy drives and the redefining of hard disk drive letters is supported by the PC-DOS 3.1 "SUBSTITUTE" command. Floppy drives on the 186 are jumpered according to conventional designations of drive 0-1-2-3 rather than the IBM method of selecting drives through the connecting cable.

While working on this evaluation, I received the new floppy disk software driver called AMPRODSK.EXE. This allows the user to mix any combination of 40- and 80-track drives on his system. 80-track drives cannot only be formatted, but the system DOS can be placed on them with the FORMAT B:/S commands. This has a very distinct advantage, as you can now Boot from 80-track drives, include your program files and still have ample storage available on the disk for saving data files. There is no loss of flexibility because an 80-track drive can read and copy data from a 40-track disk.

Is It, Or Isn't It, IBM Compatible?

Answer: Yes, No — and Almost! I'm sorry if this sounds ambiguous or confusing, but this is a complex aspect of the 186 Little Board that will require a detailed explanation.

You will have to purchase a copy of PC-DOS ($95) in order to Boot the system. I tried a "generic" version of MS-DOS and the DOS of several compatible systems but I got funny results. I have been advised that the Compaq version of MS-DOS will work. The 186 will work with Versions 2.1 and 3.1. Ampro recommends the latter for its enhanced features and because it also allows direct booting from a hard disk and larger hard disk partitions.

On the other hand, though, you can only use "GENERIC" MS-DOS programs on the system at the present time. This is because the 186 doesn't support the IBM's screen graphics commands, nor the IBM's hardware I/O due to the reassignment of these ports to conform with the 2681 DUART.

The "generic" software is also a requirement because the 186 hardware has to be able to function with a variety of terminals, each of which has its own protocol structure. This isn't an inhibiting factor as there are literally thousands of MS-DOS programs on the market that will run on the Ampro 186 and a Heath/Zenith terminal, so you would not be wanting for development or application software.

All software development tools available for the 8088 will work on the Ampro 186. Here is a partial list:

Turbo Pascal
Fortran 77
Microsoft C
Lattice C
IBM Macro Assembler
GW Basic
Intel Compilers & Tools
Palasm
Abel
Masm

Programs that are developed on the 186 will run on an IBM or compatible. When you perform assembly or compiling tasks on the 186, however, there is a definite speed increase of about 300%.

Communication between the 186 and an H/Z terminal is handled with a file called TERM.SYS which is provided on Ampro's Support Software diskette. This is a Terminal Device Driver that transforms IBM PC compatible terminal control sequences into the control codes required by a specific ASCII terminal. It provides cursor position, clear screen, etc, and allows the use of many programs which lack "MS-DOS Generic" terminal installation options.

Having experimented with a variety of programs that are labelled as "generic", I have been able to classify them into three distinct groups:

1. Programs which include the H/Z terminal in their installation file.
2. Programs which include an ANSI installation mode.
3. Programs which have no installation mode.

The H/Z-19 (and '89) terminal fares a lot better than most others because it can also be configured for ANSI mode by moving Pin #5 of Switch 401 on the Terminal Logic Board to "1". The H/Z-29, being more sophisticated, can be set up for ANSI mode through a keyboard routine and stored in non-volatile memory.

I did some checking and found out that the latest crop of Zenith Terminals — the Z-39 and Z-49 — can also be used with the 186 in not only the H/Z-19/29 or ANSI mode, but in DEC VT101 or Hazletine 1500 mode, which permits access to certain "generic" MS-DOS software that includes installation routines for these terminals.

I guess I was a bit spoiled by the vast documentation that Ampro provided with the CP/M board, because the data on the 186 board is skimpy by comparison, and intimidating by its brevity. But somewhere along the 4th dedicated reading I realized that my recent introduction to MS-DOS was lacking and that I'd better do some quick boning up. Microsoft's MS-DOS manual doesn't exactly inspire confidence. It's almost a journey backward to the original cryptic documentation that came with early versions of CP/M.

Fortunately, Microsoft Press published a decent book by Van Wolverton, called "Running MSDOS", which attempts to demystify the operating system's manual and to compensate for Microsoft's inability to produce documentation with a reasonable degree of clarity. (Wait till you tackle Microsoft's "Word" manual: it's grimace and groan from first page to last!)

There are two basic setups to consider with the Ampro 186 integrations with the H/Z-89/19. One is a floppy disk system; the other a hard disk system. We'll cover the floppy disk version first.

Setting Up A Floppy System

Ampro provides four software drivers to get a basic system functional:

TERM.SYS — This is the Terminal Driver. The H/Z-19 configuration designation is T4.
SETCON.SYS — This initializes the Terminal Port (Serial Port A) which is the DOS "CON" Port. By including this driver as the first entry in the system CONFIG.SYS file, and establishing a console baud rate. No characters are sent to the console prior to this function.
SETCOM1.COM — This sets up Serial Port "B" as the DOS "COM1" port for connecting either a modem or serial printer. You can set (as with Port A) the baud rate, parity, data length, stop bits and the hardware handshaking option, and borrow under software control additional handshaking signals from Port A.
AMPRODSK.EXE — As mentioned above, this driver can be used to format and set up a DOS system on 80-track drives.

When you first power up your Ampro 186/Heath system, the light in drive A: comes on to signal that the system is ready. Load your PC-DOS 3.1 system disk into drive A: The system will now automatically boot itself and display the DATE/TIME sequence. Place a blank disk in drive B:. Type FORMAT B:/S to format the blank disk and place the system COMMAND.COM file on it. Then COPY the three Ampro files described above to the newly sysgened disk. Also COPY the EDLIN and ANSI.SYS programs to it.

Invoke EDLIN and enter filename: CONFIG.SYS. At the prompt, type I (for insert data) and hit RETURN. Then create the command lines for the CONFIG.SYS program:

```
DEVICE = \SETCON.SYS 89600 D8 S1 PN H
DEVICE = \TERM.SYS T4
DEVICE = \ANSI.SYS
```

Type a ^C to end the program, then E to exit, and your CONFIG.SYS file is ready.

What we have done here is to initialize Terminal Port A with a Baud Rate of 9600, 8 Data Bits, 1 Stop Bit, No Parity, and Hardware Handshaking. The Heath system doesn't require hardware handshaking, but setting it up won't hurt.

Now, RESET the system and boot up on the new disk to check it out. Just prior to the screen display of the DATE/TIME boot sequence you should see a message saying HEATH TERMINAL DRIVER INSTALLED.

In order to test out a variety of programs, I made a half-dozen copies of this master working system volume and placed major programs on each disk.

Two key word processing programs I worked with were T/MAKER Integrated Software from T/Maker Software, Inc., and SPELLBINDER from Lexisoft. Ampro has licensed T/MAKER for both their CP/M and 186 systems. I did not have it available when I worked with the CP/M board, but I have become so impressed with it on the 186 that I shall spend some time on a formal review in the near future. It sells for $159 and offers sophisticated Word Processing, File Management, Spell Checking, Spreadsheet, Database Management, List Processing, Data Transfer, Graphics (Bar Charts), and Programming.

It provides a good contrast to SPELLBINDER because it doesn't contain a Heath/Zenith configuration file, while SPELLBINDER does. Copy the main T/MAKER program files to one of your system volumes, put the second T/MAKER disk in drive B: and type B:TMODIFY. When the modification program is on screen, type 1 to configure your terminal. Select ANSI by typing YES after the prompt. Then type 99 to write the modification to your system disk.

Lexisoft was kind enough to provide me with a "generic" MS-DOS version of SPELLBINDER to evaluate with this system. After placing the essential program files to a system volume, type CONFIGSB to call up the configuration program. Just follow the prompts by entering the appropriate numbers for the H-19 terminal, your printer, etc.

As I mentioned earlier, any program that doesn't contain a configuration file that includes Heath terminals will generally include an ANSI mode. This will do just fine if you remember to set the proper switch on the '19's TLB. You can do this on the keyboard with the '29.

Programs that have Heath terminal configuration files are dBASE, WORDSTAR, SUPERCALC 2, and others that were converted from 8-bit systems. Utility programs like DSPOOL, SIDEWAYS, RAMDISK, FORMAT are generally designed to work on a variety of MS-DOS systems and will work on the 186. It is advisable (at least for the moment) to check out any program at your dealer to see if it has a configuration program that includes either a HEATH or ANSI mode. (All the programs mentioned in this article have been tested, including the NORTON UTILITIES.)

You can also use the powerful PC-DOS batch processing utility to create an AUTOEXEC.BAT file to automatically load during the boot process not only the terminal configuration operation but your program file as well. (I have to confess that by the time you read this article, Ampro will have an enhancement available that will allow the 186 greater latitude. I will explain this shortly.)

Setting Up A Hard Disk System

Ampro's 186 Little Board currently supports the following SCSI hard disk controllers: The Adaptec ACB-4000 and -5000, the Shugart 1610-4 and the Xebec "Owl". The latter combines their 1410 controller and the drive electronics all on one card mounted on a 10-MByte drive. Any ST506 compatible hard disk can be used as long as you have a manual or data sheet on the drive's characteristics:

- Number of cylinders
- Number of heads
- Starting cylinder for reduced write current (RWC), if needed
- Starting cylinder for write precompensation (WPC), if needed
- Landing zone cylinder
- Drive step rate

The only cautionary note issued by Ampro is that not all Seagate model ST-506 (5-MByte) Winchester drives will work reliably with the Adaptec ACB-4000 hard disk controller. The Adaptec controller uses a 5 microsecond step pulse width. Although this pulse width technically meets the ST-506 specifications, some Seagate ST-506 drives will not seek properly with such a short pulse. The Adaptec's short pulse width allows the controller to "slew" high performance drives (such as Maxtor) at their fastest possible rate. The Seagate ST-406, and most ST-506 compatible drives, do not exhibit this problem.

The Ampro HARD.SYS device driver supports one or more hard disk drives connected to one or more SCSI (SASI) compatible hard disk controllers. It is only required if you add additional drives and controllers to your system.

As with other device drivers accessing I/O ports on the 186 board, HARD.SYS must be added to your CONFIG.SYS file if you ultimately elect to install large capacity drives and multi-partitions. An enhancement being developed by Ampro, as I write this article, will allow partitions as large as 32-MBytes on hard disk drives capable of storing 100-MBytes or more of data.

But for general user purposes of a single drive you merely need to include HFORMAT.EXE and HPARK.EXE on your System Disk. HFORMAT.EXE is the hard disk formatter utility, and it is quite simple to use.

At the system command prompt, type HFORMAT [R]. The program will then prompt you for the information it requires to format your hard disk. You will be prompted for:

Controller SCSI ID (this depends on the jumper setting on the controller).
Controller make and model.
Drive "logical unit number" (this depends on which connector on the controller the drive's data cable is connected. Drive must also be correctly jumpered according to this number.
Drive characteristics (these were mentioned above).
Format interleave (Ampro advises an interleave factor of 2. The Adaptec controller will function with an interleave factor of 1 for increased speed).

HFORMAT will format and initialize your drive(s) by writing a "clean" set of data to the drive. It also stores information about the drive needed by the SCSI controller, and it maps out any bad regions on the drive's storage surface. HFORMAT will, in the event of a problem with either the drive or the controller, issue an error message suggesting that you check out cable connections, jumper configurations and drive parameter information.

When this procedure is finished, RESET the computer and boot again. PC-DOS will now recognize the presence of the hard disk drive as DOS letter "C". As a test, you can use the PC-DOS DIR command.

Now run the PC-DOS FORMAT utility on the hard disk drive:

```
A>FORMAT C: /S/V [RETURN]
```

Remove the system disk from drive A and RESET your computer again. This time it should "boot" from the hard disk. From this point on, if you put a floppy disk in drive A, the system will boot from the floppy disk. IF there is no disk in drive A, the system will boot from the hard disk.

As a safety feature, Ampro provides a utility called HPARK.EXE which positions the read/write heads of the hard disk drive to a predefined landing zone on the disk surface. This guards against the possibility of data loss due to power on/off glitches. You should always run HPARK prior to turning off the AC power.

If you have parked your drive's read/write heads and decide not to turn off the computer, you must perform a RESET of the system before you access the drive again.

Installing The 186 Little Board

My own 186 Little Board is mounted inside my H/Z-89 computer in the cavity where the internal disk drive used to be. The floppy drive connector provides the +5 and +12. The -12 volts required by the RS232C signals is derived from an on-board voltage converter. This is not a very permanent housing, because too many external cables are required to connect the board to floppy disk drives, a printer, and a hard disk system.

If you are planning to use it with an H/Z-19 or -29 terminal, a separate case should be considered. A company called Integrand Research Corp. has designed a variety of full-height and low profile "main frame" cases that have been designed especially for the Little Board series and contain powerful switching power supplies to handle up to 2 floppy drives, 2 hard disk drives, as well as an area to mount the Little Board and a hard disk controller card. The prices range from $125 to $195. All connectors terminate on the backplane with cutouts for all connectors, so that you can easily plug in the terminal. This gives you a more transportable system if you plan to move it around. The low profile cases are quite durable and will support the H/Z-29 terminal monitor on top and allow you to position the keyboard more comfortably on your desk.

Ampro has also integrated the 186 Little Board into a complete system with assorted floppy and hard disk drives. It is called the Bookshelf 200 series and it is worth investigating. A number of system components are available separately (cables, cases, drives, controller boards, etc.). There are just too many options to list within the limitations of this article.

Documentation

Two manuals are available with the 186 Little Board: one covers the system's support software; the other all technical aspects of interfacing, system set-up, schematics, diagrams of system functions, plus specification sheets on the major component ICs.

They are well-written and easy to follow, but require the user to have a working knowledge of PC-DOS. Some minor problems you might encounter in setting up and using "generic" MS-DOS software have been covered above. The solutions are simple, of course, once you understand the configuration conventions of individual software packages.

The hardware functions superbly. The data on my '89 screen zips around at incredible speeds. It is, as I stated at the beginning, the fastest, cheapest and most sophisticated enhancement available for the '89.

Ampro has the Source Code available for all the drivers and utilities on the Support Software package for $79. For programmers and system developers, Ampro offers a 2-ROM Monitor EPROM set for $65. It includes not only standard monitor functions, but advanced debugging and hexdump features. The Monitor EPROM set is a substitute for the system ROMS and all hardware I/O is accessible. The Monitor EPROM can also be used to check out the system RAM and other operations in trouble-shooting situations.

What's Next?

Ampro has apparently decided to stretch the 186 Little Board to the limits. It already has a plug-in daughter board that contains an additional 512k RAM, 2 more Serial Ports and a socket for an 8087 Math co-processor.

Another software driver that will remap the '89s (and '19) keyboard to emulate IBM function keys is on its way.

And then, there's SUPER DUO, a terminal-type program that will permit the 186 board to hook into an IBM or compatible com-

puter via the Serial Ports, with a one-key toggle to switch back and forth Thus, programs developed on the 186 can be tested and run on an IBM moments later. Other applications that come to mind include loading different portions of large database or spreadsheet programs into each system and then being able to transfer sections of data from one computer to the other.

I have also received an interesting program that Ampro has licensed from Microtec Research, Inc. called PLUS80 CP/M & 8080 EMULATOR. This attaches an 8080 CPU instruction set and the CP/M operating system to PC-DOS. Programs are then transferred via a terminal program from the CP/M computer to the 186. Although the PLUS80 emulator does not provide 100% equivalence to an 8080-CP/M environment, most commercial CP/M applications programs are properly emulated. There are a few restrictions: no support for direct BIOS calls; programs may not access I/O ports directly; programs must not use the Z80 instructions not common to the 8080; certain public domain CP/M disk utilities violate the PLUS80 restrictions and are not properly emulated. The only testing I have had time for involved WORDSTAR and MBASIC, which ran on the 186 with PC-DOS as though it was a high-speed CP/M computer.

I've saved the best part for last. Yes, there will be IBM graphics compatibility for the '89 through a plug-in VIDEO RAM EMULATOR. This daughter-board will intercept all IBM graphic calls and displays and convert them to their ASCII equivalents, and then translate them to the terminal screen.

This means LOTUS 1-2-3, SYMPHONY and other graphic intensive programs will run on the '89. In black and white, or green or amber only. I assume that the combination of the VIDEO RAM EMULATOR and the IBM KEYBOARD EMULATOR will also permit the installation of SIDEKICK and other Borland utilities which make specific use of the function keys.

For additional information, please contact:

AMPRO COMPUTERS, INC.
67 East Evelyn Avenue
Mountain View, CA 04041
(415) 962-0230

Other references made in this article:

T/Maker Company
2115 Landings Drive
Mountain View, CA 94043

Lexisoft, Inc.
P. O. Box 1950
Davis, CA 95317

Integrand Research Corp.
8620 Roosevelt Avenue
Visalia, CA 93291

*

Western Regional

Heath/Zenith Users' Group

Conference

- LATEST TECHNOLOGY
- NEW SOFTWARE
- HEATH/ZENITH VENDORS
- GUEST SPEAKERS
- PANEL DISCUSSIONS
- DINNER/PRIZE DRAWING

NOVEMBER 14 - 16, 1986

AT... Disneyland® Hotel ANAHEIM, CA.

The Official Hotel At The Magic Kingdom

1986 WESTERN REGIONAL HEATH/ZENITH USERS' GROUP CONFERENCE

A CONFERENCE OF HEATH/ZENITH COMPUTER USERS AND LOCAL USER GROUPS.

SPECIAL GUEST SPEAKERS - PANEL DISCUSSIONS - HEATH/ZENITH COMPUTERS - ELECTRONICS - AND HEATH/ZENITH RELATED VENDORS.

CONFERENCE OPENS FRIDAY AFTERNOON ON NOVEMBER 14TH AND CLOSES AT NOON ON SUNDAY, NOVEMBER 16TH.

☆ [*Advance Registration Rates Good Thru October 15th*] ☆

NOVEMBER 14 - 16, 1986

DINNER/PRIZE DRAWING $25.00
WITHOUT DINNER $7.50

THE DISNEYLAND HOTEL IN ANAHEIM, CALIFORNIA

— WIN A COMPUTER —

THOSE WHO ATTEND THE DINNER* ON SATURDAY, NOVEMBER 16TH ARE ELIGIBLE FOR NUMEROUS PRIZES INCLUDING

(1) HS 241-81 (Extended Drive) With Video Card and Monitor

(2) ZP150 **(1) Hero Jr.**

— SPECIAL DISNEYLAND HOTEL RATES —

Single — $76.00
Twin/Double — $76.00
Triple/Quad — $86.00

*You must be present to WIN!!

PLEASE COMPLETE FOR EACH REGISTRANT:

MAKE CHECKS PAYABLE TO: **WESTERN REGIONAL HUG** CONFERENCE
Mail To: ● 1555 N. Orange Grove Avenue, Pomona, CA 91767

NAME _____
ADDRESS _____
CITY,STATE,ZIP _____

CHECK ONE
__ Admission includes Dinner/Prize Drawing $25.00 ($28 after 10/15)
__ Admission WITHOUT Dinner $7.50 ($10 after 10/15)

NAME _____
ADDRESS _____
CITY,STATE,ZIP _____

CHECK ONE
__ Admission includes Dinner/Prize Drawing $25.00 ($28 after 10/15)
__ Admission without Dinner $7.50 ($10 after 10/15)

NAME _____
ADDRESS _____
CITY,STATE,ZIP _____

CHECK ONE
__ Admission includes Dinner/Prize Drawing $25.00 ($28 after 10/15)
__ Admission WITHOUT Dinner $7.50 ($10 after 10/15)

NAME _____
ADDRESS _____
CITY,STATE,ZIP _____

CHECK ONE
__ Admission includes Dinner/Prize Drawing $25.00 ($28 after 10/15)
__ Admission WITHOUT Dinner $7.50 ($10 after 10/15)

Calendar Algorithms Unraveled

Robert G. Brasfield
303 N. 175th Street
Seattle, WA 98133

Writing for publication is not my ruling passion, however, like Mr. Winkler, author of "Keep It Simple Stupid — KISS Dating" in the February '85 REMark, I have become impatient with most of the published algorithms for computing calendar data. Many of them either (1) Don't work, except over a very limited range of time, or (2) Are so obscure, that not understanding how they work, one can only have faith in the result by having faith in the (often unknown) author. Mr. Winkler's routines will no doubt be easier to understand, at the cost of a little more memory usage than is really necessary.

What I want to offer is another point of view: there is nothing wrong with being other than brute force simple, if what you are doing is right, and it can be shown so. Further, the type of routine that is truly useful is one that is short and practical to include as a called subroutine in application programs rather than being just a demonstration of method. Many application programs that use dates and calculations of elapsed time are going to concern that area of universal interest, MONEY, so they do need to be used with a sense of confidence. I want to present a routine for calculating elapsed time between calendar dates, and a basis for understanding how and why it works, so that users can feel at ease with its results.

Also, I think some definitions of terms are needed. The Encyclopedia Britanica informs me that the "Julian Day" is, to quote, "a device of chronological reckoning often used by astronomers in order to avoid the complication due to months and years of unequal length. The days are numbered consecutively starting from the Julian Era, Jan. 1, 4713 B.C. For example, Jan. 1, 1930 was Julian day 2,425,978."

Since Jan. 1, 1930 by Mr. Winkler's algorithm would have been day 704,933, the algorithm does not accomplish conversion to a Julian date, but merely to a similar system of plain consecutive numbers that is referenced to 01-01-00 A.D. Even that reference date is just a convenient fiction, since the counting of days has been made non-continuous by acts of popes and politicians. The routine that I will show here also does a conversion to a format of simple consecutive numbers. To give it a name that will not get me in trouble with the astronomers, I will refer to it as "Semi-Julian".

The Britannica also says that dates, as established in England and other localities that followed their assignment of dates, are only continuous since Sept. 14, 1752, due to an act of the Parliament to "catch up" with the rest of the nations that use our present system. The routine that I offer here is valid for any dates from 9-14-1752 until someone alters the system again.

The primary utility of Semi-Julian dating is in the finding of elapsed time between dates. In such use, each date is converted to Semi-Julian, with the elapsed time then being just the difference of the Semi-Julian numbers. There seems to be little need for the conversion from Semi-Julian to the conventional calendar representation. It could be done though, with an inversion of this routine, similar to Winkler's inversion of his routine. The most appropriate conversion from Semi-Julian time is to a day of the week, which is most simple, just Semi-Julian Mod 7, from an appropriate reference point, which is also illustrated here.

Now to the nitty-gritty. The conversion from a calendar mode of expression to Semi-Julian is just a determination of a total sum of days from some convenient reference point. Unless one is an archeologist or historian interested in some event on 1/1/0000, there is no reason not to choose another reference if that can help in our computations. Also, since the months just roll along in a continuous sequence regardless of when we hold our New Year celebrations, we can feel free to consider the year as starting at any point that eases our computation problems. The method that I will explain here treats the year as starting on March 1 and incorporates a constant offset of +122 days when deriving the number of days due to the months number.

With the reservation that we must later make some adjustment for the omission or inclusion of extra leap year days in the first February of a new century, it should be clear that we can get the total days represented by the "year number" part of the calendar date just by the expression INT(Years * 365.25). The adding of the day number into the Semi-Julian sum of days is surely in no need of explanation.

What does need clarification is how we deal with the months, particularly that nasty February, with its strange number of days. Wouldn't it be great if some wiz had arranged it so we could just use an average number for the days in the month? We can come pretty close to doing just that. The Britannica says we can thank one Luigi Ghiraldi, advisor to Pope Gregory XIII for the arrangement of months that makes our method work.

At the heart of arriving at a number for the days represented by the months is the expression in Listing 1, INT(M(N) * 30.6), where M(N) is the number assigned to a particular month. This somewhat more terse expression serves the same purpose as Mr. Winkler's "Month table", and I hope to remove any mystery as to how the expression works, so you can have as much confidence in it as you would have in the table.

Listing 1

```
100 CLS: PRINT "Dates must be input as MM, DD, YYYY in ";
110 PRINT "that order, for example, 6,19,1959"
120 INPUT "Enter the first date   ", M(1), D(1), Y(1)
130 PRINT
140 INPUT "Enter the second date   ", M(2), D(2), Y(2)
150 '
160 'The foregoing lines are just the way that we provide"
170 'input to the core algorithm of lines 230 to 290.  In"
180 'use of the algorithm in an application program it will"
190 'often be that one of the two dates is encoded in the"
200 'program as a reference date, while the other is"
210 'provided interactively by the user, as above."
220 '
230 FOR N = 1 TO 2
240   M(N) = M(N) + 1
250   IF M(N) < 4 THEN M(N) = M(N) + 12: Y(N) = Y(N) - 1
260   DAYS(N) = INT(Y(N) * 365.25) + D(N) + INT(M(N) * 30.6)
270   DAYS(N) = DAYS(N) - INT(Y(N)/100) + INT(Y(N)/400)
280   W(N) = DAYS(N) - 7 * INT(DAYS(N)/7)
290 NEXT N
300 '
310 'What follows is the means for displaying the output of"
320 'the algorithm for demonstration here.  In application"
330 'programs the output could take many other forms."
340 '
350 DATA Saturday, Sunday, Monday, Tuesday, Wednesday
360 DATA Thursday, Friday: FOR N = 0 TO 6: READ W$(N):
    NEXT
370 ELAPSED = DAYS(2) - DAYS(1): PRINT
380 PRINT "First date, ";W$(W(1))
390 PRINT "Second date, "W$(W(2))
400 PRINT "Elapsed time was";ELAPSED;"days.": END
```

When we redefine the year as starting on March 1st, you will see we make one giant step forward for suffering mankind! It now turns out we have two easy alternatives for avoiding February. Either, (1) we are dealing with only a fraction of the month, in which case the February problem goes away when we add in the day number, or (2) we are dealing with the whole month, in which case the February problem disappears because it is included in the INT(Years * 365.25) element.

With February tucked away out of mischief, at the end of our newly defined year, we can start to construct a table that will help show how the expression INT(M(N) * 30.6) works. The first line of numbers in Table 1 are the numbers that would be given to those months by lines 240 and 250 of Listing 1. The next line is just the number of days that the month would contribute to a Semi-Julian number, as a month preceding the month in which the calendar date occurs. Thus, for a date occurring in April, March would contribute 31 days, whereas the contribution by April would go in by adding in its day number. In the line entitled SUM, the number for each month is the sum of a fixed 122 day offset plus the sum of all the days of the preceding months, with the "synthetic" year starting March 1.

The line entitled CALC is what we get if we solve the expression INT(M(N) * 30.6) with the values M(N) of 4 to 15. As you see, we generate a sequence identical to the SUM sequence. Thus, you can also see that if we generate such a number and subtract out the 122 day constant offset, we would have the number of days that should be contributed to a Semi-Julian number by the months preceding. In practical fact, for most uses of the number, we don't even have to bother removing the offset.

The lines of Table 1 below the dashed boundary are not entirely necessary for understanding how Listing 1 operates, but they may help you to understand the operation of the routine that Winkler referred to in his article as from HUG Volume II, 885-1103. The article by Jennifer McGraw in the REMark for November 1983 has the same routine. The line entitled XS is the excess of the SUM line over what it would be if all months had 30 days, while the line entitled (.6) is the result of evaluating INT(M(N) * .6). In the routine that Winkler and McGraw quoted, there is a term INT(Z1 * 2.6) that computes the excess over a constant month of 28 days — more about that later. The real issue is that in all of these various expressions, it is the pattern of retained integers due to multiples of the .6 part that makes them work.

Listing 2, if you choose to key it in and play around with it, will let you explore how far the constant can deviate from 30.6 before the method fails. This exercise might serve to give you some respect for Mr. Ghiraldi's ingenuity, as well as make you more at ease with this method. It will probably also give you serious doubt about the number 30.57, as Mr. Winkler credited to Ashton-Tate's dBASE II, in his article.

Listing 2

```
100 CLS: DIM DESIRED(15): FMT$ = "###": SP$ = "   "
110 REV$ = CHR$(27) + "p": NORM$ = CHR$(27) + "q"
120 DATA 122,153,183,214,244,275,306,336,367,397,428,459
130 FOR I = 4 TO 15: READ DESIRED(I): NEXT I
140 PRINT "This program will allow you to see the result"
150 PRINT "for yourself, if other values are used in place"
160 PRINT "of 30 6 in the expression INT(M(N) * 30.6)"
170 PRINT "NOTE: It is pointless to deviate very far!!"
180 INPUT "Enter a value  "; C: CLS
B190 PRINT "The SUM line is the sum of 122 + the total of"
200 PRINT "all the days in the preceding months of the year"
210 PRINT "assuming the year to start on Mar. 1"
220 PRINT
230 PRINT "       MAR   APR   MAY   JUN   JUL   AUG   SEP",
240 PRINT "   OCT   NOV   DEC   JAN   FEB"
250 PRINT "SUM    ";
260 FOR N = 4 TO 15
270   PRINT USING FMT$; DESIRED(N);
280   PRINT SP$;
290 NEXT N
300 PRINT
310 PRINT "CALC   ",
320 FOR N = 4 TO 15
330   DAYS = INT(N * C)
340   IF DAYS <> DESIRED(N) THEN PRINT REV$;
350   PRINT USING FMT$; DAYS;
360   PRINT NORM$,
370   PRINT SP$;
380 NEXT N
390 PRINT: PRINT
400 PRINT "The CALC line follows from evaluating the term"
410 PRINT "of Listing 1, INT(M(N) * C), with C =";C;"and"
420 PRINT "with M(N) having values from 4 to 15.": END
```

Table 1

	MAR	APR	MAY	JUN	JUL	AUG	SEP	OCT	NOV	DEC	JAN	FEB
# --	4	5	6	7	8	9	10	11	12	13	14	15
DAYS	31	30	31	30	31	31	30	31	30	31	31	
SUM	122	153	183	214	244	275	306	336	367	397	428	459
CALC	122	153	183	214	244	275	306	336	367	397	428	459
XS	2	3	3	4	4	5	6	6	7	7	8	9
(.6)	2	3	3	4	4	5	6	6	7	7	8	9

When we use this method, the Semi-Julian number corresponding to a calendar date will consist of 3 parts, (before leap year adjustments):

1. A number of days contributed by the year's number. This has an offset of -60 days due to our using as a zero reference the day that would have actually been 3-1-0000. Year 0000 would have been a leap year, with 29 days in February.
2. A number of days contributed by the month's number. This includes an additional offset of +122 days due to our use of an excess of 4 in our assignment of month numbers, relative to our "redefined" year structure.
3. A number contributed by adding in the day number.

Since in determining elapsed time between two dates, both of the Semi-Julian numbers will contain the same offsets, the offsets cancel each other. In using a Semi-Julian number to find the day of the week, we can cancel these offsets simply by positioning the names of the days in our weekday names array (Listing 1) so that when we use DAYS(N) MOD 7 as a subscript into the array, February 22, 1985 is indicated as Friday. In my version of BASIC, the MOD operator complains overflow if it is applied to numbers greater than 32767, so I had to do it the long way (Line 280), but the result is the same as a MOD 7 operation.

In Listing 1, lines 240 and 250 are where the effective start of the year is redefined. If the month is February, for example, we add 13, which gives it the number 15, and we compensate by subtracting 1 from the number of years. Line 270 makes a correction for the fact that the extra leap year day is omitted in a leap year evenly divisible by 100, except when it is also evenly divisible by 400. With those corrections the routine is good all the way back to Sept. 14, 1752 and on into the future for as long as we retain the present calendar system. Or, for another 253 centuries, after which a 7 digit single precision number will no longer contain DAYS(N). Oh well, they will probably have something better than my Z-100 by then anyway.

Now that you thoroughly understand how Listing 1 works, I can introduce you to Listing 3. This does even more violence to the "Keep it Simple Stupid" precept. When I fire up my Z-100, enter the date and time as requested by the operating system, and then go on to some application program that lets me analyze my most recent financial disaster, I find myself exasperated with any need to enter a date by a different format. The operating system demands the form 9-12-1984, where a simple basic INPUT statement fires back error messages unless I use 9,12,1984. A LINE INPUT would let me use the 9-12-1984 format, but would have to be converted into numerical values before it was useful as input to the Semi-Julian routines. Being an admirer of "lean and mean" code, I hate to use a lengthy conversion routine that does no more than cater to my longing for consistency of formats.

Listing 3 lets me live with my guilt feelings over the lengthy conversion, because the same FOR-NEXT loops that convert the LINE INPUT string to numerical values also evaluates each number as a constituent of the Semi-Julian sum, as it is converted. This routine runs sort of like a Junior Pac-man, starting at the left end of the LINE INPUT string, gobbling up the string, spitting out the delimiters and digesting the numbers. It uses essentially the same methods as Listing 1. Line 190 is where we put February at the tail end of the year, while in Line 250, we let the presently accumulated size of DAYS(I) tell us whether it is necessary to compensate by reducing the year number by 1. Each time the Junior Pac-man comes to a delimiter, known as such in Line 170 by an ASCII value of 47 or less, it fails to skip over lines 180 to 210. In line 180, it lets the value of DAYS(I) determine whether the current value for V should be evaluated as a day number or a month number. When it comes to the end of an input string, it falls through from the NEXT A, Line 230, and then interprets the value V as representing a year number.

Listing 3

```
100 CLS
110 PRINT "Enter the dates as Month, Day and Year in that"
120 PRINT "order  6-12-1954 or 1/12/57 are useable formats "
130 LINE INPUT "Enter the first date   "; D$(1)
140 LINE INPUT "Enter the second date  ", D$(2): CLS
150 FOR I = 1 TO 2
160    FOR A = 1 TO LEN(D$(I))
170       IF ASC(MID$(D$(I),A,1)) > 47 GOTO 220
180          IF DAYS(I) > 0 GOTO 210
190          IF V > 2 THEN V = V + 1 ELSE V = V + 13
200          DAYS(I) = INT(V * 30.6): V = 0: GOTO 230
210          DAYS(I) = DAYS(I) + V: V = 0: GOTO 230
220       V = (V * 10) + VAL(MID$(D$(I),A,1))
230    NEXT A
240    IF V < 100 THEN V = V + 1900
250    IF DAYS(I) > 428 THEN V = V - 1
260    DAYS(I) = DAYS(I) + INT(V * 365.25)
270    DAYS(I) = DAYS(I) - INT(V/100) + INT(V/400)  V = 0
280 NEXT I
290 ELAPSED = DAYS(2) - DAYS(1)
300 W(1) = DAYS(1) - 7 * INT(DAYS(1)/7)
310 W(2) = DAYS(2) - 7 * INT(DAYS(2)/7)
320 DATA Saturday, Sunday, Monday, Tuesday, Wednesday
330 DATA Thursday, Friday: FOR N=0 TO 6: READ W$(N): NEXT
340 PRINT "From ";W$(W(1));", ";D$(1);", to ";W$(W(2));
350 PRINT "  ",D$(2);", was";ELAPSED;"days": END
```

Listing 3 allows the use of either the form 9-12-1984 or the form 9/12/1984. (And some nonsense alternatives, too.) It will permit the shorthand form, 9-12-84, as long as the year is within the 20th century. Lines 150 - 280 could be used in an applications program as a called subroutine, similar to the Lines 230 - 290 of Listing 1, with the advantage of accepting a LINE INPUT string for the date.

The algorithm referred to by Mr. Winkler and Ms. McGraw is put into an executable form as Listing 4. I am not going to go into it in

detail, just point out similarities to Listing 1 methods. Since it only attempts to give the day of the week, it only has to be concerned with the excess days over an even number of weeks. Line 110 puts February at the end of the year, and subtracts 1 from the year number, if needed. The expression Z2 = INT(Z0/100) in Line 120 becomes part of the centuries vs. leap years adjustment when Z2 is subtracted in Line 130, where + INT(Z2/4) is the 400 year adjustment. In Line 130, INT(Z0*1.25) yields the excess of days over years of an even 52 weeks and INT(Z1*2.6) yields the excess of days over months of an even 4 weeks. The -1 at the end of Line 130 is just a "fudge factor" that permits Sunday to be represented as 0. Listing 1 gets its fudge factor by loading Sunday into the 2nd element of the array of weekday names.

Listing 4

```
100 INPUT "Enter the date by the format MM,DD,YYYY ", M,D,Y
110 Z0 = Y : Z1 = M + 1 : IF M < 3 THEN Z0 = Y - 1 : Z1 = M + 13
120 Z2 = INT(Z0/100)
130 Z7 = INT(Z0*1.25) + INT(Z1*2.6) + INT(Z2/4) - Z2 + D - 1
140 Z7 = Z7 - (7 * INT(Z7/7))  PRINT "Z7 =";Z7
150 PRINT "0 implies SUNDAY - - - - 6 implies SATURDAY."
```

Listing 5 is a throw-in that has nothing to do with dates, but it may qualify as trick code and a flouting of "Keep it Simple Stupid". What I hope for with it is that I can get in the last word on how to turn off the Z-100's key click. For a time, there was hardly an issue of REMark that did not have some new method touted. If you key in Listing 5 as a BASIC program, and run it, you will have created a COM file named OPEN.COM. When it is run under a DOS of almost any description, it will give you keyboard access to all of the Escape Sequences, as listed in Appendix O of the ZDOS (Version 1.nn) manual. I have no doubt that it will work on any MSDOS system that uses escape sequences and provides a key for inputting the escape character, but I have only actually tried it out on the Z-100 and one non-Zenith computer running an MSDOS 2.nn version.

Listing 5

```
10 DATA 180,1,205,33,60,27,117,251,205,33,60,27,116,2,235
20 DATA 240,205,32
30 OPEN "R", #1, "OPEN.COM", 18: FIELD #1, 18 AS W$
40 FOR I = 1 TO 18
50   READ C: A$ = A$ + CHR$(C)
60 NEXT I
70 LSET W$ = A$: PUT #1, 1: CLOSE: END
```

If you do use OPEN.COM, one word of caution. Don't type the Escape Sequence for "Disable keyboard", because you would then have to shut down the machine, or do Control-Reset to recover control. In using OPEN.COM, the way you terminate is to type the ESC character twice in succession. Many of you are probably so used to your ESC key sitting there seemingly doing nothing at all, it may be a shock to find you can actually type ESCx2 and turn off the key click. I find that ESCx; to give a non-blinking cursor results in a less "nagging" atmosphere too.

The only other practical method that I saw published for using escape sequences from the keyboard under the operating system, involved batch files with REMARK commands in which the escape character was input using the F8 key, in EDLIN. (See letter from Richard Hole, page 64 of the May 1984 REMark. ZDOS's manual text for EDLIN avoids exposing the F8 key's uses.) Using that scheme, you use up 128 bytes and one directory entry for each escape sequence that you make separately accessible. OPEN.COM gives you all of the sequences for 128 bytes and one directory entry. It can also prove useful if you have a screen display that you would like to save with the PSC screen dump utility, and you would like to add some explanatory note before invoking PSC. I like it as a demonstration of the power and versatility of random access files.

The cursor addressing sequence can use a little explanation. To send the cursor to line x, column y, you type ESCYnm, where n is the keyboard character found listed in the ASCII equivalence table as (31 + decimal line number), and m is the keyboard character that would correspond to (31 + decimal column number). This description is in terms of lines numbered 1 to 25 and columns numbered 1 to 80. For example, to put the cursor at line 24, column 1, the sequence would be ESCY7spacebar. You will find that the Home key, cursor direction keys, the D CHR/I CHR key, and the DEL LINE/INS LINE key that were inert under the operating system alone, will be functional. Have fun with all of those things you couldn't do before. ✱

On The Leading Edge

MS-DOS I/O Redirection Part 2

William M. Adney
P.O. Box 531655
Grand Prairie, TX 75053

One of the unfortunate facts of writing for any publication is the lead time required to get an article in print. Some magazines have a lead time that is on the order of six months. REMark has a shorter lead time that is about three months. Why is this important? It's tough to write an article that alleges to be "On the Leading Edge" when you have to deal with these lead times — the article will always be about three months "late" because of that. Even worse, new product announcements and current news will be stale information by the time the article actually appears in print, although it may have been the major news topic at the time the article was actually written.

Although no writer likes these lead times, there are few ways that most publishers can improve on them. REMark has tried to minimize these lead times by essentially requiring that all articles be submitted electronically. That eliminates the time required to keyboard the entire article and also makes it much easier to reduce typos, since most of the errors will be the author's fault. But there is more to it than that.

Depending on length and complexity, typesetting can take a significant amount of time. For example, the MS-DOS COPY command that appears in my FlipFast book took over three hours to typeset on my publisher's Itek typesetter because of the number of font changes and length of the command. And that's just one command. Although photo- typesetting produces excellent copy, it can take a long time and require a lot of manual intervention depending on the material. That's one of the specific reasons for the current popularity of "desktop publishing" software.

The next step in the publishing process is the tedious chore of page layout. This has been a strictly manual job requiring the use of a drawing board and a sharp knife. The typeset galley is cut to fit the page and is usually pasted down on a stiff piece of cardboard called a flat. If more than text is involved on a page, the process is more complex. Page layout can be difficult when photographs, listings, advertisements, and other items must be added. Since this process has been nearly all hand work in the past, that explains another reason for the popularity of the desktop publishing software. Page layout is done electronically. When used with a laser printer, the desktop publishing software can produce camera-ready copy in a matter of minutes.

The bad news is that even the best laser printers can only produce copy with a print density of about 300 dots per inch. Phototypesetters typically provide a resolution of at least twice that. In short, desktop publishing has found a niche, but it cannot compete with the quality of a photo-typesetter . . . yet. That's one reason that most magazines, including REMark, have not gone to the use of the desktop publishing software and peripherals. There is a distinct and noticeable difference in the quality of the typeset material, and high-quality publications, like REMark, will simply not accept a lesser quality.

Why is all of that important to you? I probably would never have thought of it except that Bob Ellerton told me about a letter from a reader about my review of the Easy PC that appeared in the June issue. The gist of it was that the reader thought that UCI had corrected most of the problems (e.g. the capacitor and the ROM problem) that I mentioned in the article. This Huggie knew that the lead times made up-to-the-minute information impossible, but he thought that UCI had already corrected most of the problems by the time my article appeared in print.

I agree and my most current information is that UCI has indeed corrected the problems that I told you about. I even suggested in the article that UCI is a highly respected and responsive company in the Heath/Zenith world, and I thought that they would take positive and effective action to correct the problems based on their past history. UCI sent me a new system board and an updated ROM (mentioned in my article), but both were received too late for me to go through the entire testing process again. My Easy PC was provided direct from Heath Company. It was a production unit that was shipped straight from stock. I don't like to report on things that are still in the "testing" stage no matter how close it may be to a production version.

Perhaps there is an important lesson to be learned from this. It is not unusual for new pieces of hardware or software to have some production problems in the first few months. Witness the recall of a number of new car models as an example. Computer hardware and software are no different. As a matter of policy, I never recommend new hardware or software to a friend or client until I am satisfied that it is highly reliable. I admit to being conservative in that respect simply because I believe that most users require a high degree of reliability above all else. I know that I do since I use my computers as business tools.

If you believe that you have an absolute need for a new piece of hardware or software, be cautious about buying something during

the first few months of production. It may take a while for the manufacturer to work out the production bugs which inevitably occur during early stages. Above all, be sure that you are satisfied with the reliability of a new product before you entrust it with valuable data. And if you have any questions about hardware or software changes, don't assume ANYTHING. Check with the dealer or manufacturer. That goes for new cars, as well as new computer hardware and software.

Verifying Changes In Software

As mentioned last month, one major business user has reported hard disk "problems" with his Z-241 using MS-DOS version 3.1. Reports are that he purchased the Z-241 with MS-DOS 3.1 before Zenith made hard disks available with capacities of larger than 32 megabytes. The 10 MB and 20 MB hard disks were available in the catalog, but the 40 MB (ZD-400) hard disk was not available from Heath or Zenith when the Z-241 was introduced.

This user had a "PC compatible" hard disk with a capacity of something like 38 megabytes. Using the Zenith hard disk utilities, he had reportedly allocated a DOS partition of 32 megabytes which has been the standard MS-DOS (and PC-DOS) limit for a physical hard disk. That is, you can buy a 43 megabyte hard disk, but the standard DOS cannot use anything above the 32 megabyte limit. And the 32 megabyte limit for the physical hard disk has been a standard "feature" of all MS-DOS systems and their derivatives (i.e. PC-DOS) ever since Microsoft provided hard disk support in their operating systems. All versions of PC-DOS, including the latest version 3.2, still have the 32 megabyte physical limitation. Early versions of the Zenith MS-DOS 3.1 (IO.SYS versions 3.03 and below) still had that 32 megabyte limitation. The latest Zenith MS-DOS versions (IO.SYS 3.04 and above) provide an enhancement to use 32 megabyte partitions for a physical hard disk of 128 megabytes. We will take a look at the reasons for that next month.

In any case, this user foolishly assumed that his Zenith MS-DOS version would handle a hard disk larger than the 32 megabyte limit since the 40 megabyte hard disks were being sold in the catalog. He evidently was able to define the remaining six megabytes as a DOS partition and then ran the FORMAT program on that partition. In addition to that six megabytes, FORMAT performed a "wrap around" and formatted the first 26 megabytes of partition one thus wiping out all of the directories and File Allocation Tables (FAT) in addition to a lot of business data. He reportedly called Zenith Software Consultation and was incorrectly told that was a bug in MS-DOS. Not true! It was a built-in limitation by Microsoft. When Zenith introduced the 40 megabyte (ZD-400) hard disks, the MS-DOS for the PC series was enhanced to provide additional support. IBM still does not have it.

This story goes from the ridiculous to the sublime in that this user now blames Zenith for the loss of a significant amount of his business data. He has ignored the fact that all prudent users of computer systems have current backups of business data. That notwithstanding, he also failed to verify that the DOS would handle the larger hard disk. I guess he assumed that since Zenith was selling the hard disks, his version of MS-DOS would also support it.

Although Zenith is not perfect, they do take a considerable amount of care in developing and testing software. I guess that there is no way to please everyone. But . . . onward and upward.

At long last, here is the final installment for the article on I/O redirection.

Some Definitions And Terminology — A Review

Recall that there are three definitions that we have used to understand I/O redirection: standard input (called STDIN), standard output (called STDOUT), and standard error output (called STDERR). We can dynamically redefine the STDIN and STDOUT using appropriate symbols and command syntax in a standard DOS command.

The default STANDARD INPUT (STDIN) device is defined as the standard source of input characters. The default is the keyboard or terminal. Remember that a keyboard is one part of a terminal, such as the Zenith Z-49 terminal. Unless otherwise specified, MS-DOS will always expect input from the keyboard. That is the normal case when you enter a DOS command.

The default STANDARD OUTPUT (STDOUT) device is defined as the standard destination for output characters. The default is the CRT or terminal. A CRT is the other main part of a terminal. Unless otherwise specified in one of several ways, MS-DOS will always send all output to the CRT.

The default STANDARD ERROR OUTPUT (STDERR) is defined as the standard destination for error messages. Like the STDOUT, the default is the CRT or terminal. Unless otherwise specified, MS-DOS will always send all error messages to the CRT.

One of the advanced features of MS-DOS Version 2 and later is that you can dynamically redefine STDIN and STDOUT. Dynamic redefinition means that you can change the STDIN (from the default keyboard) and/or the STDOUT (from the default CRT) by using the appropriate command syntax and several special symbols.

In the first article, we discussed the two distinct ways to redefine STDIN and STDOUT: Command Piping and I/O Redirection. The first article discussed Command Piping in some detail and will not be repeated here.

In I/O REDIRECTION, the STDIN can be redefined so that the system obtains input from a specified DEVICE other than the default keyboard. In addition, STDOUT can be redefined so that the output is sent to a device other than the default CRT.

MS-DOS I/O Redirection

I/O redirection permits the redefinition of STDIN and STDOUT so that the system may obtain data from other than the default keyboard or send data to other than the default CRT. Like the command piping, special command line characters are used to indicate input and/or output redirection. These characters are:

`<` The "less than" symbol causes input redirection. The input to a command or program is redirected to obtain data from a source (i.e. a file or device) other than the default keyboard. The general format for the command line is:

```
command < source
```

Note that the "less than" symbol points toward the command indicating that the source is read for input to the command.

`>` The "greater than" symbol causes output redirection. The output from a command or program is redirected to send output to a destination (i.e. a file or device) other than the default CRT. The general format of the command line is:

```
command > destination
```

Note that the "greater than" symbol points away from the command indicating that the destination is output from the command. It is particularly critical to understand that if the destination is an existing file, the file contents will be OVERWRITTEN with the new output. If the destination is a file that does not exist, it will be created with the new output.

>> The double "greater than" symbol causes DOS to append the redirected output to the destination which is normally an existing file. The general form of the command line is:

```
command >> destination
```

Like the output redirection, the double "greater than" symbols point away from the command indicating that the destination is output from the command.

Input From A File Or Device

The < symbol may be used to redirect input for any command, function or program that uses STDIN for input. The input may be defined to be a file or any valid input device. To access a file for input, the general form of the command line would be:

```
command < [d:][path]filename.typ
```

The optional drive — d: — and the optional path may be used in the command line. Input redirection has a wide variety of uses, but one way to use the feature is to use it with DEBUG to initiate a system patch. For example, a file called PATCH.DAT could be created that contains all of the DEBUG subcommands required for the patch. DEBUG would then update the COMMAND.COM file when the following command line is input:

```
DEBUG COMMAND.COM < PATCH.DAT
```

This example assumes that all DEBUG subcommands, including Quit (Q), were included in the PATCH.DAT file. When the input is a device, the general command form is:

```
command < device
```

The specified device must be a valid input device like a modem connected at a serial port such as AUX or COM1. A printer (e.g. PRN) is not a valid input device.

Using I/O Redirection

One of the things that I particularly miss about MS-DOS is the lack of a STAT command that allows me to dynamically redefine printers on the Z-100. But I/O redirection allows me to run a batch file that will reconfigure the printer ports with no problems. One batch file, called P2H25.BAT (Print to H-25), sets up the system to print on my H-25 printer. The file contains the following line:

```
CONFIGUR < P2H25.DAT
```

That batch file is obviously supported by P2H25.DAT that contains the following lines which are the appropriate responses to set up the H-25 printer and save the configuration to both disk and memory. All I did was go through the CONFIGUR command manually and make a list of the responses as follows:

```
A
C
(RETURN -- shows as a blank line)
H
E
E
```

Since I also have a DTC Stylewriter letter quality printer, I have another batch file which configures the system for parallel port operation. It contains the following line:

```
CONFIGUR < P2DTC.DAT
```

That batch file is obviously supported by P2DTC.DAT that contains the following lines which are the appropriate responses to set up the parallel port for the DTC and save the configuration to both disk and memory.

```
A
A
(RETURN -- Shows as a blank line)
H
E
E
```

Why would I be interested in changing output routing in this way. The best answer is that it is very helpful when using the PRINT command since I can easily choose the output printer. It is also interesting to watch the prompts and screens fly if you're interested in that.

On To Color Monitors

When I bought my H-100 over three years ago, I purposely got the Low Profile with the idea that I might want to add a color monitor. Although I liked the idea of the All-in-One computer, because it was similar to my old H-89, it was not easy or convenient to add color because of the built-in CRT. I figured that I would "someday" want to upgrade to a color monitor.

It didn't seem financially wise to buy a color monitor simply because I wanted one. However, the idea of getting an HS-241 provided the necessity for an additional monitor unless I wanted to be in the business of continually swapping monitors between computers. I didn't. So I began looking for a color monitor that would work with both my H-100 and HS-241.

Although I looked at the Zenith color monitors, I decided that I simply could not afford the high price even with the HUG 20% discount for assembled hardware. I am willing to pay a modest premium for the Zenith name and support, but when it looks like that premium is on the order of $100.00 or more, I look for other alternatives.

My personal requirements for the color monitor included full color support and connectivity to both the Z-100 and PC series computers. Since I obviously use a computer for a lot of text, I also wanted the capability to change to a "green only" display at the press of a switch. The resolution was important, but I was looking for a medium resolution capability. And, of course, the monitor had to fully support ALL of the capabilities of both the Z-100 and the HS-241. Finding a monitor to fit those requirements proved to be a more difficult task than I had originally thought.

It's a good thing that the Heathkit stores are allowed to carry products other than Heath and Zenith brands. Both of my local Heathkit stores (Dallas and Fort Worth) carry the C. Itoh CM-1000 color monitor that met all of my requirements at a very reasonable price. Even though the list price of the monitor is $499.95, both stores sell the CM-1000 for $429.00. In addition to my requirements, it also supports the audio feature with an internal speaker and a headphone jack for private listening. It also has a slightly larger 13" screen than my 12" ZVM-122A which is another advantage. For those of you interested in the technical details, I have provided that information as Listing 1.

From what I can tell, the CM-1000 seems to be compatible with just about everything including Apple and IBM. The monitor can be adapted to various systems through the use of slide switches (and appropriate cables) that are on the back of the monitor. That

means that my monitor will likely be compatible with just about any computer that I might purchase during the life of the CRT.

Listing 1
C. Itoh CM-100 Color Monitor Specifications

CRT:	13" , 0.42 mm pitch, 26.4 lbs.
Power:	115 VAC, 60 Hz, 85 Watts
Video Input:	RGB, Composite, RGB1, XRGB, Separate Video
Audio Output:	1.0 Watt
Bandwidth:	15 Mhz (approx.)
Display:	5 x 7 dot array (80 x 25 or 40 x 25)
Resolution:	640 dots x 240 lines (80 x 25) 320 dots x 240 lines (40 x 25)
Dimensions:	14.96" (W) x 13.38" (H) x 16.73" (D)

The power switch is located on the front. Usual controls for contrast, position, size, volume, and others are conveniently located on the right side of the monitor under a push-in panel that opens to display the control legends.

If you are considering an upgrade to a color monitor, I highly recommend the CM-1000. The only thing that I don't like about the monitor is the 8 pin DIN plug that is used for the RGB input to the CM-1000. However, both Dave Koslowsky (Fort Worth) and Don Murray (Dallas) have thoughtfully included an additional cable that is plug compatible with both the Z-100 and the HS-241, as well as the entire Heath/Zenith PC series. And since the IBM systems are Heath/Zenith compatible, the monitor will also work with those computers.

In the event that your local Heathkit store does not carry the C. Itoh monitors, you can order one from either the Fort Worth or Dallas Heathkit stores. Both stores accept MasterCard and Visa, and the phone numbers are listed at the end of this article.

Kit Building

For those of you who are confirmed kit builders like me, there is nothing more frustrating than trying to do something without the proper tools. This subject came to mind during a recent kit building spree involving the IR-5208 plotter and the HS-241 computer. Although it is reasonably easy to build a Heathkit using the tools recommended in the manual, I have a few additions that I have found make some assembly much easier. Some of these are special purpose tools that were not originally intended for electronics assembly.

For those Huggies who do not care to build kits, you still may find that some of these tools will be useful for other tasks, such as home repair. Even if you live in an apartment, one of the most frustrating things is to try to get apartment management to fix something simple on a Sunday night. That could make the investment in a few tools worthwhile if for no other reason than to save some frustration.

My personal experience is that I really don't like to read an article about good tools unless I can find the tool that was discussed in the article. Most of the tools that I discuss here can be found in the 1986 Radio Shack catalog or the Sears 1985/86 Power and Hand Tools Specialog. For those tools that are not listed in either catalog, I will note where I got the tool.

I'll assume that you have all of the basic tools that you require, such as screwdrivers (flat blade and phillips), pliers, and a soldering iron. Since I have found that there are some special tools that I use for building almost every kit (except for the '148 and '241), I'll talk about the "mandatory" tools first.

Super Helpful Tools

One of the most helpful tools that I have ever found is a good wire stripper. Although most wire strippers work fairly well, the biggest problem that I have found is trying to hold a very small wire with one hand and "pull" the insulation off with the wire stripper. That sounds simple, but try that after you have been working with a hot soldering iron for a couple of hours. Then try to use one of those wire strippers to strip 1/4" of insulation while holding the slick insulation of a #22 wire with one sweating (and slippery) hand. My solution was to find a wire stripper that automatically holds the wire before it removes the insulation. Radio Shack has an Automatic Wire Stripper (64-1919) that sells for $8.95 that holds the wire and strips it without pulling or cutting any of the stranded wires. Since it has a range of 8-22 gauge of wire, it is surprisingly useful for a number of other household tasks, too.

Do You Have A Screw Loose?

Maybe you don't right at the moment, but you probably have if you've ever built a kit, especially a computer kit. The phillips screws for the card cage on my H-100 were especially difficult to start because there is not enough room for adult hands and fingers in the nooks and crannies. The hold-down screws for the cards in the PC series computers can also be difficult to start.

Perhaps the most "fun" I've ever had is to be at the final assembly step in a kit and drop one of the screws into the unit. Adney's Rule of Kitbuilding says that "You will drop the last assembly screw into your kit and it will become lodged in an inaccessible place so that complete disassembly is REQUIRED to get it out". And, of course, you don't DARE turn the power on until you get that little jewel out of there.

My solution to that problem is to use some special screwholding starter tools that I bought from Snap-On Tools quite a few years ago. Although these tools are primarily used when working on an automotive ignition, they are incredibly useful for kitbuilding, particularly when working in tight locations. The first tool is for a normal slotted screw. The second is for regular phillips screws. Sears has a similar tool for slotted screws only, but it has a magnetic tip on one end. I do not like to have magnetic items around my computer, so I'm not particularly in favor of that particular item. Snap-On is the only place that I have found that sells both the flat and phillips screw starters without the magnets. The Snap-On starters do not have magnets, so I highly recommend them.

Since most of the Snap-On dealers that I have known are independent, you will need to look up your local dealer in the Yellow Pages under "Tools". My guess is that both the flat and phillips screw starters are still under $5.00 each, but local prices may vary. You can also ask your local service station dealer when the Snap-On truck comes and make it a point to be there.

Working With ICs

All of the Heathkit manuals provide warnings about static electricity and integrated circuits. Some ICs are quite susceptible to damage from static electricity. You can always tell those since they are packed in small pads of conductive foam. ROMs and CPU chips are usually especially sensitive to static discharges.

It always pays to be careful with this kind of IC. Radio Shack sells a little gadget called a Static-Draining Wrist Strap which has a hook-

and loop wrist strap connected to an alligator clip with a two foot wire. The theory is that you hook the clip to the board or chassis to eliminate static discharge so that you can handle sensitive ICs.

When I assembled an H-150 for a friend, I found a rather large number of ICs that were packed in envelopes. Radio Shack also sells 5" × 5" Conductive Foam Mats, so I bought a couple of those to organize all of the ICs. They work great for storage too — I had a lot of 64k memory chips left over from my H-100 upgrade.

One Huggie wrote to tell me of his woes with the Easy PC that were similar to mine as described in the June issue. He strongly suggests that a flat blade screwdriver is not the proper tool to remove ICs with, since he broke one of the pins on the 8088 chip. I can't really disagree with that since you really should use an IC remover for that, but it is a little difficult to find one that will accommodate the physical size of the 8088. However, the key point is that you must be careful with ICs regardless of what type of tool you are using to remove it.

When I talk about using a small flat blade screwdriver, I mean something about four inches long with a 1/8" or 3/16" wide blade such as the Sears 9HT41421. It's also useful for removing the screws on the DB-25 connectors for printers and modems.

Perhaps the best idea for IC tools is the set from Radio Shack that includes both an IC Inserter and Remover. The IC Inserter handles chips the size of normal RAM ICs and also has a pin straightener. I used it to install the 256K chips on my H-100 and found that it was a great help. The IC Remover is a tong-like tool that is helpful in removing ICs. If you work much with ICs, I think that you will find all of these tools to be extremely helpful.

Soldering And Desoldering

One very useful gadget for soldering is the Radio Shack "Helping Hands" which consists of a 4" arm with two alligator clips on a weighted base. Ever try to tin some very small wires in a cable harness — they move all over the place when you touch them with a soldering iron. I cut a small piece of the conductive pad used for ICs to pad the jaws of the clips so that they don't bite into the wire.

Other useful items for soldering include the Radio Shack Soldering Aid Set and 6" Locking Forceps. The Soldering Aid Set includes a pointed probe, a slotted probe, a brush/scraper, and a small heat sink. The forceps are handy for working with small parts and can also be used as a heat sink.

In the event that you need to desolder a connection, I like the Radio Shack "One Hand" Desoldering Iron. It has a 45 watt tip (be careful with that) and a squeeze bulb to remove solder. Other desoldering tools that I also use include a Desoldering Bulb and the Vacuum Desoldering Tool. By the way, you should not use these tools to attempt to remove a component from a multi-layer printer circuit board that is common to most computer kits. I use these tools for experimental things that do not involve printed circuit boards.

Nuts And Bolts

Almost every kit has some tight corners that make the use of pliers difficult and perhaps not advisable. If the pliers slip, you may scratch the cabinet. In many cases, a small set of nutdrivers will help considerably, such as the 8-Piece Nutdriver Set sold by Radio Shack.

But I have found that both sets (standard and metric) of Sears Ignition Wrenches are extremely helpful during kit assembly in several ways. A small open-end or box-end wrench is quite handy for tight places with all of the advantages that each type of wrench has to offer. Did you ever try to start a nut on a bolt that was in a place just a little too small for your hand? If so, the trick is to place a small piece of tape over the wrench end and use that to hold the nut.

The standard Sears set includes sizes from 7/32" to 7/16" and the metric set runs from 5 to 11 mm. This is one of those things that will usually work when nothing else will.

Other Helpful Tools

A small set of 5" Ignition Pliers are also useful for tight places. And an Ignition File or emery board is also useful for getting things to fit "exactly right". And for those times when a cabinet will not align precisely with the bolts, a tapered reamer can be used to enlarge the hole.

Since I don't work with electronic components that often any more, I have found that a Color Code Guide to resistor, capacitor, and inductor values is real helpful when I'm building other things besides Heathkits.

One item that is a particular favorite in my tool box is the SM-77 digital voltmeter from Heath. It is sensitive enough to be used on computer equipment and it has both an analog and digital display. It has a neat continuity checker that provides a beep so you don't have to look at the meter. Although it is not inexpensive, it is one of those tools that I use for all kinds of electrical measurements.

One of the hazards of getting older is that it is sometimes difficult to see very small items. I particularly noticed that when I built an H-150 for a friend and was trying to solder all of the IC sockets on the printed circuit boards. Heath has a neat little Headband Magnifier that I found to be quite necessary. Unfortunately, I didn't find a use for it when I built my '241 since there was no soldering required.

That's about it for the tool list, but there is one other item that I have found helpful in working with computers.

Cable Management

One of the hazards in working with computers is that there always seems to be cables all over the place. It is difficult to walk around the cables and my wife does not appreciate cables on the floor of my study since she can't run the vacuum. She also thinks it looks tacky and I agree.

My solution was to get some of the Radio Shack Cable Clips so that I could keep all of the cables in one bundle. These clips have an adhesive that works on smooth surfaces, but I found that one of the hazards of using them is that they will also remove the paint from baseboards, etc. I think it is worth it even though I may have to do a little touch-up painting once in a while. And I found some cable markers that provided an easy way to identify each cable when I moved to Texas.

More On IBM's "New" Keyboard

One of my friends owns an "old" IBM XT that has the "old" keyboard. Since mechanical things like disk drives and keyboards do wear out, his keyboard was beyond all hope, so he went to a local IBM Product Center to get a replacement keyboard. He was told that the "old" keyboards are not available, and the salesman tried to sell him a brand "new" XT system. I find it difficult to believe that IBM does not have replacement keyboards for the older systems, so I suspect that my friend's case was an isolated one

I did not recognize the significance of this until I learned that the "new" keyboards are NOT compatible with the "old" systems. That is, if you purchased an "old" PC or XT before the enhanced keyboard was available, the new keyboard will not work with your old system, although it appears that most software will not be affected.

I have talked to a number of software vendors, and I find that they are ignoring the function keys (F11 and F12) on the new keyboard since that would make their software incompatible with the six million plus PCs and compatibles that already exist. And no vendor wants to support two versions of the same software. About the only exception that I have found so far are vendors who sell keymapping software like SmartKey, ProKey, and Superkey.

The Advanced PC

Building the '241 kit was a snap. I spent about three hours on it, but I guess it took a little longer because of the hard disk installation. This is one of those kits that having a phillips screw starter makes the assembly go much easier.

I have been using my new '241 for testing of MS-DOS 3.1 with the hard disk and have been quite pleased with the system response. It is noticeably faster than the '100 (at 8 Mhz), and it can be slowed down to run like an IBM AT. The no wait states technology used in the '241 is both good news and bad news.

The good news is that the '241 is faster than the AT. The AT runs at about 0.75 MIPS (Millions of Instructions Per Second) and the '241 runs at about 0.94 MIPS. It's also good news that the '241 can be slowed down to the AT speed by just removing a jumper on the CPU board.

The bad news is that the '241 does not work well (or at all) with some of the add-on memory boards because of the high speed. It appears that one of the few reliable add-on boards is the obvious one available from Heath and Zenith. If you are planning to add a memory board, such as the Intel Above Board, be prepared to slow down the '241 by pulling that jumper.

There are inevitably some who will point to Zenith and say that the '241 is not compatible, but it can be slowed down to work properly with these boards by pulling the previously mentioned jumper. I have also seen some initial reports that these same boards will not work with IBM's new 8 Mhz AT, so that should cause an interesting flurry of activity among some of these hardware manufacturers. In a few cases, it appears that fixing the problem is a simple matter of using faster RAM chips. If you are considering adding memory to your '241, you may want to wait a few months until the dust settles on this issue. Most important, be sure that you can test a potential memory board purchase before you buy it or get a money-back guarantee in the event that it will not work at high speed. I suspect that we'll see a few manufacturers with close-out specials on the slower memory boards in the near future

Next Month

I have been working with the '241 for some time now, and the system is a pleasure to use. The keyboard is MUCH better than the AT. I'm glad to see that people at Heath and Zenith are flexible enough to recognize that perhaps IBM really can't design the best keyboards for computers. Perhaps that is the reason for the new AT compatible keyboard kit that appears in the latest catalog.

We will take a look at the '241 system with a hard disk and MS-DOS 3.1 in the next article. Although I will talk about a few technical points, most of the article is written in non-technical terms with the idea of understanding how MS-DOS works with a hard disk. We will also spend some time looking at each hard disk command with the idea of understanding the whys and wherefores.

If you have any questions about the information in this article, please let me know. Please enclose a stamped, self-addressed envelope (as usual) if you would like to have a personal reply.

Products Discussed

Software

MS-DOS Version 3	
PC only (OS-63-31)	$ 150.00
Z-100 only (OS-63-30)	150.00
Programmer's Utility Pack (CB-3163-30)	150.00

Hardware

Advanced Personal Computer (HS-241)	$2499.00
Monochrome/Color Video Card (Z-409)	239.00
20MB Winchester (ZD-200)	1499.00
40MB Winchester (ZD-400)	2499.00
Z-200/AT Keyboard (HK-200)	124.95
H-100 Desktop Computer (HS-1108-41)	999.00
Easy PC Emulator (PC-250)	699.00
Z-150 MS-DOS (OS-63-31) (with PC-250 only)	90.00
Gemini Emulator Board (PC-251)	599.00
Z-150 MS-DOS (OS-63-31) (with PC-251 only)	90.00

Heath/Zenith Computer Centers
Heath Company Parts Department
Hilltop Road
St. Joseph, MI 49085
(800) 253-7057 (Heath Catalog orders only)

C. Itoh Color Monitor (CM-1000) $ 429.00

Mr. Don Murray
Heath/Zenith Computer Center
12022-C Garland Road
Dallas, TX 75218
(214) 327-4835 (VISA and MasterCard accepted)

Mr. Dave Koslowsky
Heath/Zenith Computer Center
6825A Green Oaks Road
Fort Worth, TX 76116
(817) 737-8822 (VISA and MasterCard accepted)

Listing 2
Kit Building Tools List

Heathkit Fall 1986 Catalog No. 203

Headband Magnifier (GDP-201 - p. 52)	$ 24.95
Digital Multimeter (SM-77 - p. 57)	119.95

Radio Shack 1986 Catalog

Page 120	
Conductive Foam Mat (276-2400)	$ .89
Static-Draining Wrist Strap (276-2399)	1.99
IC Inserter and Extractor (276-1574)	6.95
Cable Management Accessories (p. 127)	
Color Code Guide (271-1210 - p.133)	$ .59
Page 136	
Desoldering Bulb (64-2086)	$ 1.99
Vacuum Desoldering Tool (64-2098)	4.95
"One-Hand" Desoldering Iron (64-2060)	6.95

Continued on Page 83

Z-System To Boot!

Analytical Products Offers Z-System On Bootable Disk . . . Ready To Run

Rick Swenton
19 Allen Street
Bristol, CT 06010

If this is the first time you have ever heard of Z-System and you own a Z-80 based computer, you just don't know how to live! Z-System is an enhanced CP/M 2.2 compatible operating system. BE VERY CAREFUL. Once you use Z-System, you may loose the desire to upgrade (?) your H-89 to an MS-DOS compatible machine. Z-System features are typical of those found in MS-DOS and UNIX. Enhancements like shells, paths, multiple commands on a single line, named directories, flow commands, I/O redirection, error handling and memory-resident commands have elevated the 8-bit computer to heights never before considered attainable. If the terminology gets confusing, please refer to my two previous REMark articles on ZCPR3 and Z-System found in the November 1985 and January 1986 issues, respectively.

In this article I will describe a bootable Z-System for the H-89 offered by Analytical Products. If you have been following my articles on ZCPR3 and Z-System, you may be having a tough time deciding which way to implement Z-System. Until now, Heath users had only two ways to implement Z-System: Manual-Install and Auto-Install. The manual installation procedure required the user to be proficient in the use of assembly language and be able to make changes to the Basic Input/Output System (BIOS). It also required skill in the use of a debugger (DDT or equivalent) and the MOVCPM/SYSGEN functions. The Auto-Install version overlays the user's version of CP/M currently residing in memory. The end-result is the same. However, under the Auto-Install version, the system is booted from standard CP/M. Then Z-System is brought into memory via a COM file. The Manual-Install version results in a Z-System bootable disk.

Now there is a third way to obtain Z-System: Analytical Product's bootable Z-System. This is just like having the manual install version except that someone else did all the work for you and then some. This takes all the guesswork out of installation and allows the novice to get down to the "nuts-and-bolts" of Z-System right out of the box.

Just a quick note: Throughout this article I will refer to ZCPR3, the Replacement Console Command Processor which replaces CP/M's standard CCP, and ZRDOS, the Replacement Disk Operating System which replaces CP/M's standard BDOS. Together they form Z-System. ZCPR3 is the part of Z-System which processes commands and ZRDOS is the part of Z-System which coordinates disk activity. They are two separate programs which were designed to work together. The Basic Input/Output System (BIOS) is the final part of Z-System (or CP/M). The BIOS is written specifically for the computer hardware by the manufacturer.

What You Get

The disk set includes a bootable system disk and a number of disks containing the support files. The exact number of disks depends on the type of media desired. The source code to the essential files like the BIOS and ZCPR3 are present. Due to the massive size of the source code to the utilities, it is not provided on the disk set but is available for a copying charge. The supplied documentation includes "ZCPR3 - The Manual" by Richard Conn, the ZRDOS Programmers Manual and an instruction booklet by Analytical Products. "ZCPR3 - The Manual" is a 351 page book covering every aspect of ZCPR3, the Console Command Processor.

Implemented Features

When performing a manual install of ZCPR3 you will find it hard to select from the many options available. Analytical Products selected those options for its Z-System which it felt made the most sense while keeping memory space for user programs (TPA) as large as possible.

A large TPA memory was maintained by optimizing code in the BIOS. The BIOS is a version of Heath's 2.2.04 implementation re-

written in Z80 code and shortened as much as possible. All of the H-8 code was removed from the BIOS in the shortening process so even if your H-8 has a Z-80 CPU board, you cannot boot this Z-System disk. More about this later. The BIOS source code was broken down into modules for ease of modification. The modules consist of the separate disk device drivers for H17, H37, H47 and H67, the CRT and printer device drivers, the Cold Boot routine, and finally the system equates and other definitions. At this time, Analytical Products supplies a bootable disk for the combined support of H17 and H37. If you currently own and use Livingston Logic Labs BIOS-80 which supports double-sided 48 or 96 TPI hard-sectored disks, you can obtain a version of Z-System from Analytical Products which includes BIOS-80 code. You must provide proof of ownership of BIOS-80. The BIOS modules for H47 and H67 are included for those brave souls who want to try their hand at bringing up the Z-System on those disk types. Contact Analytical Products if you need to operate on H47 or H67 disks. All of the standard features of Heath CP/M 2.2.04 have been retained. If you do wish to customize any of the BIOS modules, you will need a Z-80 compatible assembler such as Microsoft's M80.

Analytical Products also provides a bootable Z-System for the CDR soft-sectored controller board and a version for the Heath H-8 is planned for release soon.

The ZCPR3 command processor was also optimized. This was done so that Analytical Products could pack as many features as possible into the 2K maximum size permitted for the command processor. The original version of ZCPR3 was able to be assembled for operation (with reduced features) on an 8080 CPU. In order for the ZCPR3.ASM file to assemble for both the Z80 and 8080, macros were used to translate the Z80 opcodes into 8080 equivalents. Since there were no 8080 equivalents to the Z80 block move instructions like LDIR, they were not used in the original version. Analytical Products optimized ZCPR3 wherever possible by using Z80 instructions. They also added clear screen and printer form-feed commands. All possible internal commands in ZCPR3 have been implemented. All of these improvements result in improved speed and efficiency while ensuring the largest TPA size for user programs.

In order to configure the special options of Analytical Products' Z-System, they have provided several reworked versions of familiar Heath programs: CONFIGZ.COM, MOVZR17.COM and MOVZR37.COM.

The CONFIGZ.COM program is similar to Heath's CONFIGUR.COM. It allows you to tailor the software to meet the needs of your hardware. In addition, it allows you to automatically run a program called STARTUP.COM on cold or warm boots. The inner details of auto running a program on cold or warm boot under CP/M are dramatically different under Z-System. With CONFIGZ.COM, the difference is transparent to the user. CONFIGZ correctly sets-up Z-System to automatically run the STARTUP .COM file on cold or warm boot if desired.

The MOVZ17 and MOVZ37 programs are similar to MOVCPM17 and MOVCPM37 except that in the MOVZnn programs, the standard CCP was replaced with ZCPR3 and the standard BDOS was replaced with ZRDOS. Integrating ZCPR3 and ZRDOS into the MOVZnn programs means that Z-System is fully relocatable and that you can create a new Z-System of any size you require. This may need to be done when installing other hardware accessories such as RAM boards.

In the CDR version of the bootable Z-System, the names of the modified system utilities are consistent with the familiar CDR utilities (i.e. I/OMOD instead of CONFIGUR).

Standard Features

Analytical Products implementation of Z-System includes the following standard features:

- Named Directories — 5 available
- Flow Control Package (memory resident or disk-based)
- External File Control Block — 36 bytes
- Shell Stack — two 32 byte stacks
- External Stack — 48 bytes
- External Environment Descriptor
- External Path — 5 pairs of bytes
- Wheel Byte
- External Command Line Buffer — 200 characters

I will now explain the standard features provided with the Analytical Products' Z-System disk in more detail.

Named Directories

These are physical names assigned to drive/user areas. An example of this would be a pair of disks which have programs in different user areas. Text files are in user area 3 of drive A, your checkbook program is on drive A in user area 12 and your BASIC programs are in user 5 of drive B. Using Named Directories, you could then have the following:

```
A3:TEXT      A12:CHECKS     B5:BASIC
```

Under Z-System, the A> prompt also shows the current user area. If you were logged into the first directory, the system prompt would be:

```
A3:TEXT>
```

If you wanted to move to your checkbook program directory, you could do this either by typing "A12:" or by using the CD.COM utility (Change Directory) and type "CD CHECKS". Once you set up your Named Directories, you don't have to remember what the drive/user areas were. You can log into them using the name of the directory. As part of the memory conservation effort, logging directly into a named directory, such as by typing "CHECKS:" is not permitted under this version of Z-System. The CD utility must be used to change directories my name. Of course, you can still change the directory using the DU (Drive/User) form, such as "A12:".

Flow Control Package

Two types of Flow Control Packages (FCP) are provided. Both are memory resident. One version (called the "Resident" version) processes flow commands solely from its own memory-resident command set. The other version (called "IF.COM version) also attempts to process Flow Commands from its own memory-resident command set. However, with this version, if the desired Flow Command can not be found in the memory-resident command set, the FCP will invoke the more powerful IF.COM file to resolve the command. With this version, IF.COM must be located on the disk in the "ROOT" directory and it will take a bit more time to resolve the command since the FCP must be checked and then the IF.COM file loaded and executed. (The Root directory is the last drive/user area specified in the search path. See "External Path".) Under the first FCP version, you cannot take advantage of the extended power of IF.COM processing. If the desired flow command is not memory-resident, the command processing

stops quickly. This is nice if you only use a few Flow Commands and you know which ones are memory resident.

The following flow commands are available from the memory-resident FCP:

IF EMPTY — if a file is empty
IF EXIST — if a file exists
IF INPUT — ask the user for a yes or no
IF EQ — if afn1=afn2 (filenames are the same)
IF ~ — a leading tilda (~) negates any of the above, such as IF ~EMPTY means "if the file is NOT empty"

All of the standard ZCPR3 flow commands are available with IF.COM processing.

Future releases of Z-System will probably have only the resident IF commands. The IF.COM commands will still be available, but to again increase available user memory, the automatic scanning of resident IF commands will be disabled and if the commands are not found, the attempt to process it will halt. The more powerful IF.COM will be renamed so as not to conflict with the resident IF processing and the invocation of IF.COM will be my manual user selection.

External File Control Block And External Stack

These are features which are transparent to the user and were implemented under ZCPR3 to remove code from the CCP which really did not have to be inside the CCP and place it elsewhere in memory. Having both of these external freed-up valuable space to add more commands and features to the CCP. In addition, the External FCB and External Stack are now available to Z-System utilities which need to access them. One example of the use of the External FCB is in shell processing. The ZCPR3 command processor stores the name of the command it just processed into the external FCB buffer. A shell would use this feature to determine the name it was invoked under so it can become ready to re-execute itself.

Shell Stack

Two 32 byte Shell Stacks are provided. The Shell Stack allows ZCPR3 to process shells. A shell is a "front-end" which provides a user-interface instead of the system prompt. Among the standard shell programs provided are MENU/VMENU and VFILER. MENU/VMENU are menu-oriented command preprocessors where the user can create custom menus for inexperienced persons to operate the computer. Their interface to the system is a menu instead of the A0> system prompt. VFILER is a very powerful full-screen video oriented file handler. It performs disk operations like copying, renaming, erasing, etc. It performs the same basic functions like NSWEEP, DISK7, WASH and others, but it takes into account the ZCPR3 environment to address the video attributes of the user's CRT terminal.

Z3 Message Buffer

The Z3 Message Buffer is another transparent feature. Without it, ZCPR3 could not process flow commands, shells, error handling, or ZEX, the memory-resident batch processor similar in function to CP/M's SUBMIT.

External Environment Descriptor

This area in memory is where the pointers to the Z-System segments and buffers are maintained. Each Z-System utility has a pointer to this Environment Descriptor. Since the utilities know where the Environment Descriptor is located, they can also easily know everything about the configuration of the Z-System without having all that information present within each utility. If you alter the size of the Z-System and relocate the address of the Environment Descriptor, you must also change the pointer to the Environment Descriptor in each of the Z-System utility files. This is what is known as "installing" a Z-System file. You merely change the pointer to the Environment Descriptor. A special utility is provided to a "mass" installation of the files. This would only be necessary if you alter the size of Z-System. All of the files on the distribution disk are pre-installed.

External Path

A five level Path is provided to specify the order of drives and user areas (directories) to search for COM files to be executed. The Path could be specified in the drive/user format, such as A3 A12 B5 or in the named directory format, such as TEXT CHECKS BASIC where, in this example, the order of searching for a requested COM file would be first to look on A3:TEXT, then on A12:CHECKS and finally on B5:BASIC. The Path, as well as the assignments of the named directories, are easily changed by the user.

Wheel Byte

The Wheel Byte is nothing more than a software switch that can enable or disable certain Z-System commands. This provides the user with a means of system security. The Wheel Byte could be reset to disable the use of COM files which check that byte before running.

External Command Line

A Multiple Command Line Buffer is provided which will allow you to enter more than one command at the system prompt. For example, you could type "DIR;TYPE MYFILE.DOC;B2:;ERA LETTER.TXT <return>". Each command is separated with a semicolon. This command would display the directory of the current drive, type the file MYFILE.DOC on the screen, log into drive B: user 2, and finally erase the file LETTER.TXT. Up to 200 characters can be entered on a single command line.

Z-System Utilities

The distribution disk set contains the standard ZCPR3 and ZRDOS utility files, as well as some of the newer video-oriented utilities written by members of the Z-System user community. In addition, a full complement of on-line help files are provided.

I used to think that help files were a waste of disk space and that the hard copy was all I needed. But using the computer's power to electronically "flip" through the pages of the on-line "book" while I was learning Z-System was much easier.

If you desire the source code to the Z-System utilities, it is available for a copying charge ($5 per disk). There is about 912K of files which will require 5 SSDD soft-sectored disks or 11 hard-sectored disks to contain the files.

For The Future

Analytical Products has plans to make Z-System bootable disks available for Magnolia and the H47/H67 controllers for the H-89, as well as the Heath controllers for the H-8. The current version only supports H17/H37 (and BIOS-80) or CDR operation on the H-89. Call them for information on availability of these products.

Conclusion

Analytical Products has done a fine job in producing a bootable Z-System disk for the H-89. They have implemented as many

features as possible while conserving available memory. No installation is required. Personal customization and relocatability (for add-on accessory boards) was simplified. If you have an H-89, you should not be without Z-System. If you were wondering about your skills to attempt a manual installation, consider Analytical Products bootable disk as the fastest way to get you into the heart of the power of Z-System!

Ordering Information

Z-System Bootable Disk Set for Heath/Zenith 89/90 computers with H17/H37 disk controllers or the CDR soft-sectored disk controller $98.00, specify disk format. Add $3.00 for shipping and handling. California residents add 6% sales tax. Pay by check, money order, Visa/Master Card or COD.

Analytical Products
20663 Avenue 352
Woodlake, California 93286
(209) 564-3687

Acknowledgement

I would like to thank Mr. Peter Shkabara of Analytical Products for providing me with an evaluation copy of Z-System and for the time he spent answering my questions and supplying the latest product update information.

*

HUG Price List

The following HUG Price List contains a list of all products in the HUG Software Catalog. For a detailed abstract of these products, refer to the issue of REMark specified.

Part Number	Description of Product	Selling Price	Vol. Issue
HDOS HARDCOPY SOFTWARE			
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			
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	
885-1135-[37]	HDOS Variety Pkg.	20.00	76
GAMES			
HDOS			
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	45
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			
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			
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	20.00	58
885-8042-37	ZDOS/MSDOS Poker Party	20.00	77
UTILITIES			
HDOS			
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			
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/MSDOS			
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
885-3031-37	ZDOS/MSDOS Graphics	20.00	69
885-3037-37	MSDOS ZPC II	60.00	76
885-8029-37	ZDOS Fast Eddy	20.00	53
885-8035-37	MSDOS DOCUMAT and DOCULIST	20.00	70
885-8041-37	ZDOS/MSDOS Orbits	25.00	75
H/Z100 ZDOS/MSDOS - H/Z150 PC MSDOS			
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
885-3032-37§	MSDOS Halley's Comet Locater	20.00	70
885-3035-37§§	MSDOS SPELL5 & SPELL5F	20.00	72
885-3036-37§	ZDOS/MSDOS DEBUG Support Util	20.00	77
885-8039-37§§	MSDOS DPATH	20.00	74
885-8040-37§§	MSDOS HELP Programs	20.00	74

§ All program files run on both

§§ Program files run partially on both

Part Number	Description of Product	Selling Price	Vol. Issue
PC/IBM COMPATIBLE			
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
885-6007-37	MSDOS DND (Dung. & Dragons)	20.00	70
885-6009-37	MSDOS Screen Saver Plus	20.00	76
885-8033-37	MSDOS Fast Edit	20.00	62
885-8037-37	MSDOS Grade	20.00	70
885-8044-37	MSDOS TCSpell	20.00	79
PROGRAMMING LANGUAGES			
HDOS			
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			
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			
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			
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
885-8036-[37]	CP/M Grade	20.00	70
ZDOS			
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

Continued on Page 83

HUG NEW PRODUCTS

HUG P/N 885-8043-37

CALC $20.00

Introduction: CALC is a faithful, working, full-function simulation of the Hewlett-Packard HP-25 pocket calculator. There are only two differences. These are, all ten registers may be used as storage registers rather than the first eight, and the number of program steps has been arbitrarily set to 100 rather than 49. User programs may be stored on disk for subsequent loading. The calculator uses RPN (Reverse Polish Notation) like most HP calculators. Input to the calculator is via a light pen.

Requirements: To properly use this program you need an H/Z-100 (not PC) series computer with at least 128k of system RAM. It is in full color so all video RAM planes should be installed (192k). Finally, a light pen must be connected to J4 on the rear panel. This program requires MS-DOS version 2.0 or greater; it will not work with ZDOS.

The following programs or files are included on the HUG P/N 885-8043-37 CALC disk.

CALC	.COM
FACT	.PRG
README	.DOC
ABSTRACT	.DOC

For those of you who don't have a light pen for your H/Z-100, HUG has made special arrangements with the Lite-Pen Company of Los Angeles, California to enable you to purchase a very high quality lite pen at a distributor's price. Included with this software is a coupon allowing $114.98 off the standard retail price for this lite pen. This single pixel resolution pen normally sells for $204.95

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.

Note: All special update offers announced in REMark (i.e. ZPC II update) must be paid by check or money order, payable to the Heath Users' Group. **NO CREDIT CARDS ACCEPTED.** ZPC II contains only one disk. It is a combination of ZPC I and the ZPC Support disk plus added improvements. Thank you.

but can be purchased with this coupon for only $89.97. Included with this lite pen are two cables (one for the H/Z-100, and one for a standard IBM PC), operations manual, and demo software (for the IBM PC). Even if you never use CALC, the coupon alone is worth the price of this HUG package. A more detailed review of this pen can be found in the September 1986 issue of REMark.

Author: Robert F. Doolittle

Comments: Each key is labeled with a 2 character label. These labels change dynamically in both color and content when a calculator 'f' or 'g' function key is pushed. An on-line HELP facility is included to further define these key labels. It is toggled on and off by the keyboard HELP key. More extensive instructions and explanations are provided in the README.DOC file.

TABLE C Rating: (5), (10)

HUG P/N 885-3007-37 Z-DOS/MS-DOS CP/EMulator Update

CP/EMulator is a program that allows you to run CP/M programs under Z-DOS or MS-DOS on an H/Z-100 (dual processor) computer. It has been improved considerably and re-released under the same HUG part number. For a description of the original program, see page 27 of your HUG Software Catalog Update #1.

CP/EMulator has been improved to run faster and handle "tough" CP/M programs that it could not run before, including CP/M WordStar (all functions work), PIP (including all switches), and the CP/M editor, ED. Screen I/O is faster, so that word processors and action game programs are more useful and enjoyable. With the new CP/EMulator, you can run nearly all of the CP/M programs in the HUG library under MS-DOS or Z-DOS. (However, CP/M Microsoft BASIC is required for some programs, and is not supplied.)

If you have the original CP/EMulator and would like to upgrade, send in your original HUG disk and $5.00 to Heath/Zenith Users' Group, Attn: Nancy Strunk, Hilltop Road, St. Joseph, MI 49085. Make checks payable to: Heath/Zenith Users' Group.

Note: Since CP/EMulator now runs the CP/M editor, the HUG CP/M editor is no longer included on the disk.

HUG P/N 885-6002-37 MS-DOS CP/EMulator II and ZEMulator Update

CP/EMulator II is a program that allows you to run CP/M programs under MS-DOS on an IBM-PC compatible computer, such as the H/Z-150 series, etc. ZEMulator is a program that emulates the Z-100's function keys and escape codes. They have been improved considerably and re-released under the same HUG part number. For a description of the original program, see page 42 of your HUG Software Catalog Update #1.

CP/EMulator II has been improved in the following ways:

- Support of the V20. If you have a V20 processor in your computer, CP/EMulator II will use it to execute the 8-bit code in CP/M programs, which will cause them to run much faster. If you do not have a V20, CP/EMulator II will emulate the 8-bit code itself. It automatically detects the V20 and uses it, if one is installed.
- Faster screen output. Whether you have a V20 or not, CP/EMulator II will be faster than before, because screen output has been speeded up.
- Runs more programs. CP/EMulator II has been improved to run more CP/M programs, including WordStar (all functions), PIP (all switches), and ED.

ZEMulator has been improved to include a special graphic mode that provides all H19 graphic characters for use on the newer Heath/Zenith computers, that do not have the graphic characters built in. For older H/Z PCs, such as the H/Z-150 series, the built-in H19 characters can still be used.

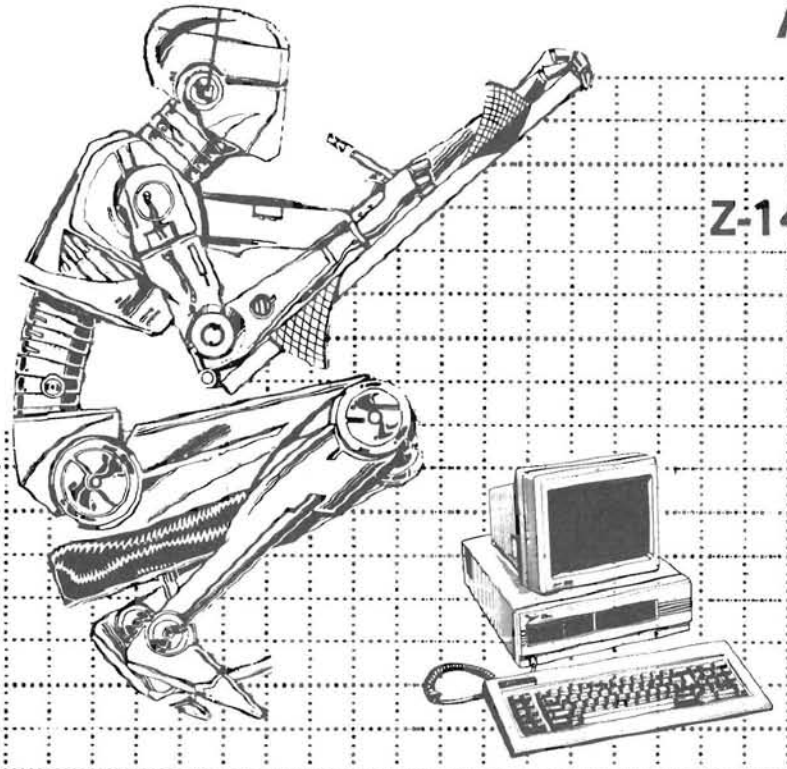
In addition to the above improvements, the CP/EMulator II and ZEMulator disk comes with two new programs that were not on the original disk. These programs were designed for the H-8 or H-89 CP/M user who also has a PC-compatible computer.

HRDCPM — This program works like the RDCPM program included with your Heath/Zenith MS-DOS, but it can read H-89 or H-8 format double density 48 tpi single- or double-sided, soft-sector disks, as well as Z-100 CP/M disks. The Heath/Zenith RDPCM program for PC compatibles can read Z-100 disks and IBM CP/M-86 disks, but not H-89 disks. Note: HRDCPM cannot read SINGLE DENSITY soft-sector disks or hard-sector disks, because the PC disk controller is incapable of reading such disks.

TF89 — This program allows you to connect your Heath/Zenith PC and your H-8 or H-89 together with a serial cable, and transfer CP/M files from the H-8 or H-89 to the PC. PIP is used on the H-8 or H-89 to send the file, and TF89 receives it and saves it on your MS-DOS disk. The transfer rate is 9600 baud, and any file (text or machine code) can be transferred. If your H-8 or H-89 cannot produce a disk format that HRDCPM can read, you can still transfer files using TF89.

If you have the original CP/EMulator II and ZEMulator disk, and would like to upgrade, send in your original HUG disk and $5.00 to Heath/Zenith Users' Group, Attn: Nancy Strunk, Hilltop Road, St. Joseph, MI 49085. Make checks payable to: Heath/Zenith Users' Group. *

FIRST CAPITOL DESIGNS THE BEST VALUES IN SYSTEMS AND SOFTWARE



Z-148 38 Mb System

$1999

- ✲ blazingly fast
- ✲ 8 mHz with no wait states
- ✲ new high speed disk controller
- ✲ 640 K RAM
- ✲ amber graphics monitor
- ✲ 38 Mb disk and 1 floppy
- ✲ serial and parallel ports
- ✲ color graphics adaptor

CAD Package Special

Get into CAD for a low cost ... our CAD package special includes Generic CADD, a program amazingly similar to AUTOCAD (TM), and the Microsoft mouse. Special price only $199, through September 30, 1986. If purchased with Z-148 38 Mb system, only $149.

Options: DotPlot — will allow you to output Generic CADD plot files to your dot matrix printer, if you don't have a plotter, only $24.95.

AutoConvert — convert your Generic CADD files to AUTOCAD DFX format and vice versa, only $24.95.

ZENITH | data systems

AUTHORIZED SALES AND SERVICE

First Capitol Computer

First Capitol Computer
1106 First Capitol Dr.
St. Charles, MO 63301
(314) 724-2336

A Buffer Stuffer

Bruce Noblick
185 Crestview Road
Columbus, OH 43202

If you have ever needed the ability to capture interrupt controlled input data, pass strings of characters to other sub-routines, or read disk files one character at a time, you may wish to utilize the code presented in this article to simplify your programming efforts.

This article describes a set of routines used to handle a first-in first-out queue (FIFO). Consider the following introductory notes (each is expressed as a question):

What Is A Queue?

A queue is a means of storing information that may not be immediately processed.

What Is A Last-in First-out (LIFO) Queue?

Most micros implement a LIFO queue. You might be more familiar with it if you refer to it as a stack. A stack only requires one pointer. This pointer moves in one direction when you store data and in the other when you retrieve it. In many cases, there is no counter for items on the stack. This must be handled by software.

What Is A First-in First-out Queue?

A FIFO queue inserts and removes data independently requiring separate pointers. This structure does not directly provide an item count, but it makes it easier for software to implement one. A typical application for a FIFO queue would be a type-ahead buffer in an interrupt environment. An example of this routine is shown in Chart 1.

Chart 1
Using the queue handling routines

```
; Chart 1 -- Using the queue handling routines
;
, This program example includes the basic components of
; an interrupt driven console input handler:
;
; Replacement code for the console input and status BIOS
; routines: an interrupt handler that queues data for the
, console input routine, and an initialization routine
; that patches them into CP/M
;
; This initialization routine is not complete  It only in-
, cludes the portion required to tie in the supplied routines
;
. The purpose of this example is to show you how to use the
; routines in Chart 3
;
; Lines with         will be used to indicate that other
, code should be present
;
, This program will not assemble properly unless you
; include the source code from Listing 1
;
. Note that rule 3 of the requirements, restrictions and
, remedies applies to this example.
,
CON   EQU    350Q      ; console base port
DATP  EQU    CON       , console data port
CTLP  EQU    CON+5     , console status port
RCVB  EQU      1       , data ready bit
RCVR  EQU      1       ; value of data ready bit when true
IJMP  EQU     24       ; address of JMP to interrupt handler
;
, INITIALIZATION ROUTINE
,     .
;     .
;
      LXI    H,IBUF    , get queue address
      SHLD   TQUE+1    , and notify queue handling routines
      LXI    H,IVEC    , get new interrupt handler address
      SHLD   IJMP+1    ; and replace original interrupt address
      LHLD   1         , get BIOS base address
      INX    H         , increment to address of console status
      INX    H         ; routine
      INX    H
      INX    H
      LXI    D,GCSTAT  , new console status routine address
      MOV    M,E       ; to be patched into BIOS
      INX    H
      MOV    M,D
      INX    H         , increment to address of console
                         input routine
      INX    H
      LXI    D,GCDATA  ;new console input routine address
      MOV    M,E       ; to be patched into BIOS
      INX    H
      MOV    M,D
;
;     .
;     .
,
; Interrupt processor
;
IVEC: PUSH   PSW
      IN     CTLP      , is data ready?
```

"Have Your Computer Talk To Ours!"

"HUGPBBS is on line, 24 hours a day, with over 10 megabytes of free software available for downloading. There's software for every Heath/Zenith operating system, with the majority being for MSDOS, and specifically the Heath/Zenith PC compatible computer systems. Also included is software for HDOS, CP/M, and MSDOS for the H/Z-100 computer system. In addition to this software is a message base through which you can exchange information with other HUG members. Have your computer call (616) 982-3956, 'The Heath Users' Group Personal Bulletin Board System' and make connection at 300, 1200, or 240 baud. Type a carriage return several times to get my attention. Registration requires that you supply your human's first name, last name, HUG ID number, and some sort of secret password (up to 16 characters). Alternatively, your human can call Jim Buszkiewicz at HUG, and register via voice connection at (616) 982-3837. Call today! All it takes is a computer, modem, and a phone call for your computer to talk to ours!"

```
        ANI     RCVB
        SUI     RCVR
        JNZ     IRET        , no, return to user
        IN      DATP        ; otherwise, get data
        CALL    SCQUE       ; and queue it
IRET:   POP     PSW         ; return to non-interrupt code
        EI
        RET
;
; Console status routine
;
GCSTAT  DI                  , see RRR rule 3
        CALL    TQUE        , data available?
        EI
        RZ                  ; return if not
        MVI     A,0FFH      , A=0FFH if data available
        RET
;
; Console input routine
;
GCDATA  DI                  , see RRR rule 3
        CALL    FQUE        ; get data from queue if available
        EI
        JC      GCDATA      ; repeat until data is found
        RET
;
;       ...     ...
;
;
IBUF:   DB      0           , queue character count
        DS      257         , space for get pointer and
                            data buffer
;
;
        END
```

What Are "Get" And "Put" Pointers?

A "get" pointer shows where the next available data item is located in the queue buffer. A "put" pointer shows where the next data item may be placed in the queue buffer. As stated above, a LIFO queue uses a single pointer for both, but a FIFO queue implements them separately.

The queue structure, as presented in the article, is similar to a FIFO queue as described above. Each queue buffer has two parts:

1. A pointer area — Two bytes that describe buffer utilization.
2. A data area — A 256 byte auto-wrap around or ring buffer.

The queue format is described by the following chart:

Byte	Description
0	Character count — 0–255
1	Get pointer — 0–255
2–257	Data buffer

Notice that only 255 characters may be stored in a buffer, not 256. Note also that there is no reference to a 'put' pointer. The 'put' pointer is derived by adding the 'get' pointer and character count. The character count may be thought of as a "put displacement". This implies that you may clear the queue by storing a zero in the character count. This causes the 'get' and 'put' pointers to refer to the same location.

As an example, let's make the following assumptions. The character count (first byte) contains 000H. (It should be defined or initialized that way). The 'get' pointer (second byte) was not initialized, but it contains an arbitrary value of 0FFH (it does not need to be defined or initialized with a specific value unless you intend to use other mechanisms to store data in the queue). See Chart 2 for an example. Notice that the pointers after the CLRQUE call are set the same as after the second FQUE call. This is because no change was made to the 'get' pointer after that point in the chart and the CLRQUE call doesn't change it either.

Chart 2
Pointer Handling Example

BEFORE:				AFTER:		
PUT DSP	GET PTR	PUT PTR	SUBR	PUT DSP	GET PTR	PUT PTR
---	---	---	---	---	---	---
0	255	255	SCQUE	1	255	0
1	255	0	FQUE	0	0	0
0	0	0	SCQUE	1	0	1
1	0	1	FQUE	0	1	1
0	1	1	SCQUE	1	1	2
1	1	2	SCQUE	2	1	3
2	1	3	CLRQUE	0	1	1

The assembly language source code listed in Listing 1 provides the following functions: SCQUE — send character to queue, FQUE — fetch character from queue, TQUE — test for data in queue, CLRQUE — clear queue.

When you send a character to the queue, the character count is incremented to update the 'put' pointer. When you fetch a character from the queue, the character count is decremented and the 'get' pointer is incremented. This causes no net change in the 'put' pointer. When you test for data in the queue, the buffer address and character count are returned to you. It is also possible to determine if no buffer has been assigned. (See the requirements, restrictions and remedies for information about assigning a buffer). When you clear the queue, only the character count is altered: it is set to zero. This is good enough for most applications to recognize that the buffer has been cleared. These routines can handle multiple queues simultaneously. This feature is discussed in the following paragraph:

This section of the article describes the requirements, restrictions and remedies (the three R's) that apply to this code.

1. The first byte of a buffer should initially be set to zero. If not, the buffer could be thought to have information in it. You could call CLRQUE to do this for you.

2. These routines will do nothing until you store a buffer address at TQUE+1. This address must NOT be zero.

3. Though it is possible for interrupt and non-interrupt code to share a single copy of this code, that raises the possibility of conflicts. To avoid this problem, use the following general procedure in your interrupt code:

   3A. Call TQUE to get the current queue address and save it for later.
   3B. Store your new queue address at TQUE+1.
   3C. Do whatever your interrupt code was intended to do.
   3D. Retrieve the original queue address and store it at TQUE+1.

4. Though it is possible for interrupt and non-interrupt code to share a single buffer, that can also cause problems. To avoid the problem, don't call any of these routines from non-interrupt code with interrupts enabled.

5. If interrupt routines have their own copies of this code and their own queue items 3 and 4 above do not apply.

6. It is possible to handle blocks of text, such as a block of data from a disk sector read or to be written by your program. The following general rules should clarify this point.

   6A. To store a block of data in the buffer for retrieval by the FQUE routine:

      6A1. Set 'get' pointer and character count to zero.
      6A2. Place the block of text in the first portion of the 256 byte buffer.
      6A3. Set the character count to the length of the text in the buffer.

   6B. To process blocks of text placed in the buffer by the SCQUE routine.

      6B1. When the character count reaches the number of characters you wish to extract from the buffer, clear both pointers and extract the data. Remember that only 255 characters may be stored in each buffer.

I have used these routines in many of my own programs, such as a replacement program for PREL.COM, a program for the Type-'n-Talk, and others. A good friend of mine has used them in his own very powerful full screen text editor for CP/M. I hope they will be useful to you, as well.

Listing 1

```
; Listing 1 -- FIFO queue handler sub-routines
;
; SCQUE -- send character to queue
; ENTRY. A=character
; FLAGS: C=1 buffer overrun would have occured if char-
; acter had been sent or if no buffer had been selected
; USES:  F
.
SCQUE:  PUSH    H
        PUSH    PSW
        CALL    TQUE    ; get queue buffer address
                          and length
        INR     A       , check character count
        JZ      CQ1     ; jump if buffer full
        MOV     M,A     ; update character count
        DCR     A       , find the place to "put" the
                          next character
        INX     H
        ADD     M
        INX     H
        CALL    DADA
        POP     PSW     ; get character back and
        MOV     M,A     ; store it
        POP     H       ;; return to caller
        ORA     A
        RET
CQ1:    POP     PSW     ; return to user
        POP     H
        STC             ; with carry bit set as error
                          indicator
        RET
;
; FQUE -- fetch character from queue
; ENTRY: no parameters
, EXIT.  A=character
, FLAGS: see TQUE FLAGS
; USES:  A F
;
FQUE    PUSH    H
        CALL    TQUE    , get buffer address and length
        JC      FQ1     ; jump if no data or no buffer
                          assigned
        DCR     M       ; update character coung
        INX     H
        MOV     A,M     , retrieve get pointer
        INR     M       , update get pointer
        INX     H       , find the place to "get" the
                          next character
        CALL    DADA
        MOV     A,M     , get the character
        ORA     A
FQ1:    POP     H       ; return to user
        RET
;
, TQUE -- test for data in queue
, EXIT.  A=character count
;        HL=buffer address
; USES:  A F H L
: FLAGS: C=1 Z=Ø no queue assigned
,                (word at TQUE+1=Ø)
;        C=1 Z=1 no data available
;        C=Ø Z=Ø data available
; NOTES: If no queue buffer has been assigned, the char-
, acter count will be returned to the caller as Øffh.
.
TQUE:   LXI     H,Ø     , get queue buffer address
;
; ** The above instruction is modified by the user to
, specify the queue buffer address  A value of zero means
; that no queue buffer has been allocated
;
        MOV     A,H     ; test buffer address for zero
        ORA     L
        JNZ     TQ1     , if not zero, continue
        DCR     A       ; otherwise indicate error and exit
        STC
        RET
TQ1     MOV     A,M     , get character count
        ORA     A
        RNZ             ; return if data available
        STC             ; set carry with zero to indicate
                          no data
        RET
;
; CLRQUE -- clear queue buffer
; USES:  F
, FLAGS  C=1 no queue buffer assigned
,
CLRQUE: PUSH    H
        PUSH    PSW
        CALL    TQUE    , get queue buffer address and length
        JNC     CLQ1    , no possible error
        JNZ     CQ1     ; JUMP IF NO BUFFER ASSIGNED
CLQ1:   MVI     M,Ø     ; set character count to zero
        POP     PSW     ; return to user
        POP     H
        ORA     A       , clear carry
        RET
;
; DADA -- add A to HL (unsigned value in A)
, USES: A F
;
DADA:   PUSH    D
        MOV     E,A
        SUB     A
        MOV     D,A
        DAD     D
       POP     D
        RET
.
        END
```

The Best Gets Better

A Review Of The Perks (Version 2) Desktop Utility

Pat Swayne
HUG Software Engineer

When the Sidekick desktop utility program was first released, those of us who use H/Z-100s complained quietly that yet another terrific program was out for PCs, clones, and others of that ilk, that we had to do without. By now, however, there may be as many desktop utilities for the H/Z-100 as there are words in this sentence, and the problem is which one to select. Well, if my vote counts for anything, I say the best one is and always has been Perks, written and sold by Barry Watzman.

A "desktop utility" like Perks is a program that loads itself into memory and remains resident there while you run other programs. It is actually a combination of several programs that can be called up and used whenever you need them by entering a special key sequence on your keyboard. When you call up one of the programs in a desktop utility, it usually appears on the screen as a "window" that overlays the background text or graphics. The entire background is usually not overlayed, so that you can refer to things on the screen from your foreground program while you are using the utility. In Perks, the windows can be positioned anywhere on the screen, and their positions can be permanently saved so that the next time you use your computer, the windows of your utility are where you put them. In some of the other desktop utilities that I have seen, only limited positioning of the windows is available, and some do not allow you to save the new positions permanently.

The utilities available in the original version of Perks were a calculator, an ASCII character table, a calendar with appointment file, a filer (for executing certain DOS commands), and a notepad editor. The new Perks Version 2 adds a Rolodex-type card file, a modem program, a telephone dialer, a typewriter emulator, and a cut-and-paste function. Since most of you are probably familiar with desktop utilities by now (there have been a number of articles in REMark about them), I will not describe all of the functions in full, but I will point out the features of some of them that set Perks apart from the other programs of this type.

The calculator in Perks is a four function calculator that works in hexadecimal and decimal numbers. In the hexadecimal mode, you can also perform logical (AND, OR, XOR) and shift operations. In the decimal mode, it is a floating point calculator with 9 digit accuracy, and it automatically puts commas between thousands. You can perform all normal functions of the decimal mode from the numerical keypad of your computer. With other desktop utility programs, you may have to use the main keyboard for some functions.

The Notepad editor in Perks uses WordStar-style commands, and also uses function and keypad keys for some commands. If you are familiar with WordStar, the Perks notepad will be a familiar friend. Scrolling in the Perks notepad window seems to be a bit faster and smoother than with other desktop notepads that I have seen, especially when you consider the size of the window, which is larger than the others. The size of the window is adjustable, and can be made to fill about two-thirds of the screen if you wish.

The modem utility in Perks is a complete modem program in itself, with full upload and download capability, and it supports XMODEM protocol. Unless you want the bells and whistles of a big fancy modem program, it may be all you need.

The telephone dialer may be the most fun part of Perks. It searches the current text on the screen for anything resembling a phone number. If it finds one, it marks it in reverse video, and gives you the option of dialing that number on a Hayes-compatible modem or moving to another number on the screen. If you are, for example, looking at the README.DOC file from a piece of software, and it gives a phone number for consultation, you can use Perks to dial the number for you when you see it on the screen.

Probably the best thing about Perks is that it works, and works well. I have never experienced a crash because I popped up Perks while running another program, even if the other program is an IBM PC program running under ZPC. I have experienced crashes with other desktop utilities, and most of them will not run correctly under ZPC. Perks is also more efficient in memory usage than other desktop utilities. Some of them allow you to vary the number of functions loaded to conserve memory, but when you load in as many features as Perks has, you wind up using much more memory than Perks.

One feature of Perks that no other desktop utility I know of has is that it works under Z-DOS, as well as MS-DOS. If you are still using Z-DOS on your H/Z-100, Perks is your only choice.

Perks is one of those programs that I call "clean" software. If you are thinking of getting a desktop utility for your H/Z-100, you probably couldn't do better than Perks. Perks is available from Barry A. Watzman Microcomputer Systems and Consulting, 560 Sunset Road, Benton Harbor, MI 49022, (616) 925-3136. Now, if only I had a Perks for my Z-158 . . . *

HUG Members only!

HUGE HUGGER DISCOUNTS now available on these selected products...exclusively for members only!

That's right. We're offering HUG members these close-out, last call and limited quantity specials from Heath Company.

The items listed are not necessarily representative of complete systems. please call your nearest Heath/Zenith Computers & Electronics Center or Heath Company toll-free at 1-800-253-7057 for more information before ordering. Alaska and Michigan residents call: 616-982-3411.

Watch for additional MEMBER ONLY discounts in upcoming issues of REMark®!

REMark Discount List

	Item	Last Adv'd Price	HUG Member Price	Qty Left
ADP-1501	Stereo Synthesizer	99.95	74.95	65
GDP-1433	Digital Golf Trainer	199.95	149.95	36
H-19-3	H-19 to H-88 Mod Kit	220.00	199.95	11
H-88-1	Floppy Disk Accessory	150.00	99.95	42
H-88-9	Drive Mounting & Inst	25.00	14.95	3
H-89A	W/O CRT, Ser I/O, Disk Cont	----	399.95	4
HCA-9	Print Stand For H-125	99.00	59.95	7
NEP-1210	Assem Set-back Thermostat	79.95	39.99	83
PC-131	USR PCM-5 1200B Modem (PC)	349.95	239.95	31
PC-200	USR 1200B Modem Card (S100)	399.00	279.00	35
PCA-120-1	6-PAK Plus Game Acc'y	39.95	19.95	16
PCA-120-1	6-PAK Plus Game Acc'y	39.95	19 95	16

	Item	Last Adv'd Price	HUG Member Price	Qty Left
PM-111	Stat-Attack Anti-Static Kit	24 95	9.95	68
PP-215	EPSON JX-80 Color Printer	699.95	449.95	23
PP-240	EPSON LQ-1500	1299.00	999.00	4
PP-400	Koalapad Touch Tablet	99.95	69.95	125
PPA-240-3	LQ-1500 Interface: Serial	149 95	129.95	8
PPA-240-4	LQ-1500 Interface: Parallel	94.95	79.95	18
PPA-400-1	Koala Speedkey Software	59.95	29.95	144
ZA-100-4	S-100 Extender Board	59.00	19.95	207
ZA-219	Amber CRT/H-121 (Tube only)	75 00	14.95	537
ZG-219	Green CRT/H-121 (Tube only)	75.00	14.95	84
ZW-219	White CRT/H-121 (Tube only)	75.00	9.95	142

* New additions this month

Heathkit®

Heath Company

A subsidiary of Zenith Electronics Corporation.
Some items may not be available at all store locations.
Prices, product availability and specifications subject to change without notice.

CP-236

Developing MS-DOS Batch Files An Overview

Eric L. Pang
1530 Nehoa Street
Honolulu, HI 96822-2008

Introduction

After using MS-DOS for a while, one will find that certain procedures or operations require that you type in the same commands over and over again. An example of this might be creating a working diskette, i.e. a bootable diskette containing several frequently used utilities and applications.

To create this working diskette, the diskette is first formatted and the MS-DOS system placed on it. Certain files, such as CHKDSK .COM, a screen clock, an extended directory program, a print spooler and a word processor, are then copied to this diskette. For each working disk created, the commands entered are the same time after time. MS-DOS provides the "batch file" capability where all of these commands can be placed into an ASCII file, and this file "executed" as a regular MS-DOS "command" by typing the name of it at the MS-DOS prompt. (This "batch file" feature is also available under other operating systems with various degrees of flexibility, e.g. CP/M submit files, UNIX script files and VAX/VMS command files.)

By convention, these MS-DOS files are given the file name extension of ".BAT", e.g. "WORKDISK.BAT". You must not have a ".COM" or ".EXE" file of the same name as MS-DOS will first search for the ".COM" or ".EXE" file, and execute that file if found. The MS-DOS "PATH" feature (which is used to search different drives and subdirectories for a program) will also work with batch files.

Creating The WORKDISK.BAT File

As an example, we will develop a batch file "WORKDISK.BAT" for creating our working disk. Assume the working disk to be created is on drive B: and the necessary files to be copied to the working disk reside on drive A:.

Start up your favorite editor, creating "WORKDISK.BAT", and type in the commands that you would normally execute in order to create the working diskette,

```
FORMAT B: /S/9
VERIFY ON
COPY   A:CHKDSK.COM   B:
COPY   A:SCRNCLK.COM  B:
COPY   A:SDIR.COM     B
COPY   A:PRINT.COM    B:
COPY   A:WS* *        B:
```

You should now have a file WORKDISK.BAT on drive A:. To test the batch file, place an empty disk into drive B: and type "WORKDISK<RETURN>".

The commands contained in the file will be displayed as they are executed. (You will be asked by FORMAT.COM to enter various information about a disk label and whether other disks are to be formatted. Although I could have used the /N flag, at least for MS-DOS v2.x for the Z-100, to suppress the prompts, I will take care of the input requests to FORMAT using another technique later on in this article.) If an error occurs during execution, e.g. a file to be copied is not present on drive A: or a command was entered incorrectly, an error message will be displayed on the screen and execution continued.

From this basic batch file, other commands and features can be added to improve usability and readability, for example, adding messages to the user and programmer's comments. A brief summary of the commands to be added in this example are:

Command Description

REM [remark] Batch file remark or comment.
ECHO ON Turns on ECHO command; display batch file commands during their execution.
ECHO OFF Turns off ECHO command; suppress display of batch file commands.
ECHO [string] Display string on output device (usually the console).
PAUSE [prompt] Suspends batch file execution until any key, except <CTRL-C>, is pressed; with display of an optional prompt.

These commands can be incorporated into the batch file as follows,

```
ECHO OFF
REM
REM  WORKDISK BAT - this batch files creates a working
REM        diskette   Format and transfer system, copy
REM        various utilities and editor
REM
```

```
ECHO Place working disk to be created into drive B:
PAUSE
ECHO Formatting and transferring system
FORMAT B  /S/9
ECHO  ...
ECHO Copying utilities to working disk
VERIFY ON
COPY    A:CHKDSK.COM    B
COPY    A:SCRNCLK.COM   B:
COPY    A:SDIR.COM      B:
COPY    A:PRINT.COM     B:
COPY    A:WS*.*         B:
ECHO Done. Working disk created
ECHO ON
```

The first command, ECHO OFF, prevents the commands and REM statements from showing up on the screen. The second ECHO displays a message for the user to place a diskette into drive B:. PAUSE prompts the user to press <RETURN> when ready. FORMAT.COM is run (and will then ask you to enter information about a disk label and whether more diskettes are to be formatted). The following ECHO is used to add a line feed to the last message issued by FORMAT.COM. The user is then informed that the utilities will be copied to the working disk. The VERIFY option for COPY is set to ON. The files are copied and a short message is displayed to the user indicating that the working disk has been created. ECHO is then turned back on.

Although more aesthetic than the original version, you will still be required to answer questions from the format program and messages from the MS-DOS commands, example COPY, will still show up on the screen. To remedy the message problem, the output from these commands can be sent or "redirected" to the null device, essentially throwing the messages away,

```
ECHO OFF
REM
REM  WORKDISK.BAT - this batch files creates a working
REM        diskette.  Format and transfer system, copy
REM        various utilities and editor
REM
ECHO Place working disk to be created into drive B:
PAUSE
ECHO Formatting and transferring system
FORMAT B  /S/9
ECHO  ...
ECHO Copying utilities to working disk
VERIFY ON
COPY    A:CHKDSK.COM    B: > NUL:
COPY    A:SCRNCLK.COM   B: > NUL:
COPY    A:SDIR.COM      B: > NUL:
COPY    A:PRINT.COM     B: > \dev\nul
COPY    A:WS*.*         B: > \dev\nul
ECHO Done  Working disk created
ECHO ON
```

where "NUL:" and "\dev\nul" are two different names that can be used for the "NULL device". (See: Holub, A. April 1986. "C-Chest. Redirection-The /dev Directory, SWITCHAR, and Touch." Dr.Dobb's Journal of Software Tools. vol. 11(4).)

To take care of the requests required by FORMAT.COM, input redirection can be used. An ASCII file is created with the data needed by FORMAT.COM. Input to FORMAT.COM is then redirected from this data file. Create a file, "FORMAT.DAT", that contains three lines. The first two lines contain a <RETURN> while the third line contains "N",

```
<RETURN>
<RETURN>
N
```

Edit WORKDISK.BAT changing the line containing FORMAT.COM to

```
FORMAT B: /S/9  < FORMAT.DAT  > \dev\nul
```

Except for the displaying of the

```
ECHO OFF   and
ECHO ON
```

commands on the screen, WORKDISK.BAT is now quite a "professional" looking program.

A Special Batch File — AUTOEXEC.BAT

The file named "AUTOEXEC.BAT" is a special file. This file is executed automatically each time the system is booted up. This file can contain programs to set the system date and time from a real-time clock, to set PATH, etc. Note that if AUTOEXEC.BAT is executed, you are not asked to enter the date and time. Thus, these commands must be placed within AUTOEXEC.BAT if you need to enter the date and time manually.

Calling Other Batch Files

One can include the name of a batch file as the last command of the currently executing batch file. Control does not return to the first batch file, however. One exception to this is to call the second batch file with "COMMAND.COM", for example:

```
REM - presently in the first batch file
REM - the next line will execute a second batch file
COMMAND.COM /C BATCH2.BAT
REM - return here and continue with first batch file
```

This technique will also work with other "resident" or "transient" MS-DOS commands.

Other Commands Used In Batch Files

Although I will not use the following commands, they are included here for completeness. They may come in handy when writing other batch files.

Command Description

EXIT Exit command and return control to a previous command level.
FOR Loop processing; repeat an MS-DOS command for a given set of conditions.
GOTO Branch to a label within batch file.
IF Conditional execution during batch file processing.
SET Set a variable within the system environment.
SHIFT Decrement position of batch file parameters.

Check your MS-DOS manual for a more detailed explanation of these commands.

Setting Up A Word Processing Application Diskette

As another example, I will discuss a batch file that can be used to set up a word processing application diskette. This diskette should be bootable, and contain all the necessary word processing files and utilities on this one diskette, if possible. Assume for our example a 2 drive diskette-based system with 768K. The word processor used in this example will be WordStar (obviously a hold-over from the "real" 8-bit days, not unlike the writer of this article).

What we want to do for this application is to 1) boot up the diskette, 2) set up a RAM disk, 3) copy the word processor and its overlays, along with several other utilities and files (e.g. spelling checker, key remappers, etc.) to the RAM disk (to increase performance), make the diskette containing the documents the default drive (drive B:), and 4) start up WordStar. Note that WordStar may have to be patched in order for it to find its

overlays correctly. (See note at the end of article). As an alternative to patching, I will use an "extended PATH program called "SuperPath" which will help WordStar find its overlays.

For step 1, create a bootable diskette with "FORMAT /9/S" (syntax for the Z-100). The second step requires that a file "CONFIG.SYS" be present on this diskette. Step 3 will be accomplished with a batch file called "WPSTART.BAT". The last step will be performed with a batch file named "WP.BAT".

First, we need to set up a RAM disk. On my system, I create it by telling MS-DOS that a device driver called "RAM.COM" will be made into a "device". This command is placed in the file CONFIG.SYS (which is an ASCII file containing subcommands used to set system characteristics just before the end of the boot sequence). If available to you, a device driver called MDISK.DVD could be used instead. The CONFIG.SYS file should look something like this,

```
BUFFERS = 20
FILES = 12
DEVICE = RAM.COM 225
BREAK = ON
```

The number of buffers and files have been increased from the default values in order to increase performance (though I have not done any actual tests to prove this). The size of the RAM disk will be just about 225K. Of course, the size you make your RAM disk will be dependent upon the amount of memory available on your system and the number and size of the files that are to be copied there. You should leave about 200K of system memory for WordStar though. On my system, the RAM disk is automatically assigned to drive I:, the "next available" drive on my system. For an interesting article on CONFIG.SYS, check: S. Libes' article on page 38 in the May/June 1986 issue of Micro/Systems Journal.

Next create an AUTOEXEC.BAT file. Upon boot up, this file will set the system date and time, and startup the word processing batch file.

```
REM - AUTOEXEC.BAT
DATE
TIME
WPSTART
```

The file WPSTART.BAT, listed next, is used to set up the word processing system. It creates the "extended path" SPATH, so WordStar can find its overlay files, then copies WordStar and its overlays to the RAM disk. Other files, the spelling checker and dictionary, the key remapper files and the batch file that actually calls the WordStar, "WP.BAT", are also copied to the RAM disk. Drive B: is then made the default disk (the disk containing the documents) and the word processor batch file executed.

```
REM - WPSTART.BAT
SPATH I:\;A:\
COPY  WS* *        I.
COPY  SPELSTAR.*   I:
COPY  KEYWS.COM    I:
COPY  UNMAP.COM    I
COPY  WP.BAT       I
B:
WP
```

If you do not use SPELSTAR frequently or if there is not enough room in your RAM disk, just keep the dictionary on diskette. RAM disk requirements will be reduced dramatically.

Besides starting up WordStar, WP.BAT is also used to install the key remapper.

```
REM - WP.BAT
KEYWS
I:WS %1
UNMAP
CLS
```

Upon exit from WordStar, the key remapper is removed and the screen cleared. Control is returned to the operating system. Drive I: is used explicitly to prevent an inadvertent execution of the diskette copy of WordStar.

Of course, the word processor can be reinvoked anytime by typing "WP" or "WP filename" at the MS-DOS prompt. As can be seen in WP.BAT, any file name specified after WP is substituted into the "%1" parameter and is passed to WordStar for further processing.

Though not always advisable (because of power failure for example), I had an occasion where a user wanted to copy the working document to RAM disk and edit it there. In this situation, two RAM disks can be created, one for the documents and the other for WordStar and utilities. The basic batch files presented above will be modified only slightly in order to take into account the different drives being used, the copying of the working documents to the RAM disk and the rewriting of the documents back from RAM disk to diskette.

CONFIG.SYS is edited so that two RAM disks, I: and J:, can be created,

```
BUFFERS = 20
FILES = 12
DEVICE = RAM.COM  60
DEVICE = RAM.COM 180
BREAK = ON
```

Drive I: will contain the documents and drive J: WordStar and utilities.

WPSTART.BAT file is modified to take into account the new extended PATH, "J:\;A:\", and the copy commands edited so that drive J: is now the destination drive (i.e. "COPY filename J:"). Drive I: is then made the default drive.

WP.BAT is updated to reflect the different location of the word processor and to copy the document back to diskette upon exit from the word processor,

```
REM - WP.BAT (with documents on I., ws on J.)
COPY document_name  I.
KEYWS
J:WS %1
ERASE *.BAK
COPY  *.*       B:
UNMAP
CLS
```

Here, we are assuming that drive I: was already made the default drive and that we want to copy the edited document back to B:.

The only "trick" with this set-up is getting the working document copied to RAM disk. MS-DOS has no elegant way to interactively copy a desired file to RAM disk; you must already have in mind the file you want to copy. For this application, I had to create my own program, BATCOPY.COM, for interactive file copying. Assuming the source drive for the documents is B:, WP.BAT finally becomes,

```
REM - WP.BAT (with interactive copying to RAM disk)
DIR B: /W
BATCOPY B: I  A.
I
KEYWS
J:WS %1
ERASE I:*.BAK
COPY  I:* *     B:
```

```
UNMAP
CLS
```

where the syntax for BATCOPY.EXE is

```
BATCOPY source_drive: destination_drive: command.com_drive:
```

and "command.com_drive"is the drive name where COMMAND.COM can be found, usually drive A:. The source for BATCOPY is presented at the end of the article. Notice that WP.BAT now makes explicit references to drive I:. This was done to reduce drive name confusion for the novice user.

Summary

I hope this motivates you to create your own batch files which can be quite useful for taking care of many everyday chores, e.g. creating working disks, setting up the word processing environment, spreadsheet environment, for compiling programs, etc.

To create complicated batch files, however, additional software may be helpful and even required. An "extended" PATH utility is quite useful and will remedy many of the shortcomings of MS-DOS and the numerous MS-DOS applications that do not have the ability to use PATH to find its overlays and files. For even more complicated batch files, one may be forced to write customized utilities, in assembler or C (or Pascal?), to overcome the limitations of MS-DOS batch files.

Note 1: WordStar Overlay Problems

WordStar is destined to look for its overlays in either the currently logged drive or in the root directory of drive A: (i.e. A:\). This can be overcome by, 1) patching WordStar so that it will look in the root directory of another drive for its overlays or 2) using an "extended PATH" utility program.

To patch WordStar on Z-150 type computers so that it will find its overlays on drive I:, use DEBUG on WS.COM and change the byte at 2DC from 01 to 09 (03 for drive C:, etc.). Next, change the two bytes at 1E04 and 1E05 to 90s.

The above patch will not work with the WordStar for the Z-100 though. A patch for WordStar Professional can be found in "Customizing MS-DOS WordStar Professional for the H-100 Computer" by F.T. Clark on pages 47-51 in the November 1985 issue of REMark.

One can also use an MS-DOS utility that "extends" the search PATH. The program I use, SuperPath, intercepts the DOS open file, file size and exec calls, and searches the subdirectory paths set up by the first call to SuperPath, e.g.

```
SPATH I:\, A:\
```

As you can see, SuperPath's syntax is quite similar to the MS-DOS PATH command. ("SuperPath Directory Performance Utilities", Martin Scot Development / 4515 Purdue N.E. / Seattle, WA 98105 / phone (206) 527-9605.

An equivalent HUG program, DPATH (HUG P/N 885-8039-37), could also be used.

Note 2: KeyMapper

The key remapper referred to in this article was written by Patrick Swayne and can be obtained from HUG (P/N 885-3010-37) for Z-100s and P/N 885-6001-37 for Z-150s.

Note 3: RAM Disk

The device driver to create the RAM disk, RAM.COM, came with the Desmet C development system. MDISK.DVD, a program that comes with MS-DOS supplied by Zenith can also be used. Though it may have to be patched or modified in order to create a RAM disk of "non-default" size, this being dependent on your particular version of MDISK.DVD.

Software to create RAM disks by "non-device" means, such as those programs that may be included with PC add-on RAM cards, are generally difficult to use and set up.

Note 4: Screen Clock

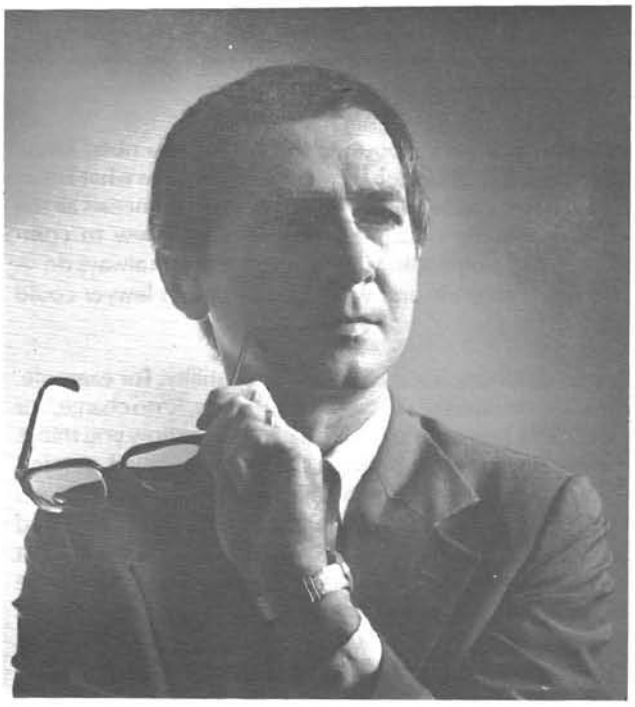
The screen clock mentioned here was developed by Patrick Swayne. It maintains a 24 hour time display in the upper right corner of the screen. Versions are available for the Z-100s and Z-150s. Refer to REMark Nov. 1984 page 61, REMark Aug. 1984 page 43 and REMark May 1984 page 37.

Note 5: An Interactive "COPY" Program Used In Batch Files

Since MS-DOS does not provide a way for true interactive file copying from within a batch file, the following program was written. After a directory of the source drive is displayed, "BATCOPY" is executed. The user can then enter the name of the desired file(s) that are to be transferred to a pre-designated destination drive.

Listing

```
/* batcopy.c - program to allow user to enter name of several
     files from the keyboard from w/in a batch file  files are
     then copied from pre-designated source to pre-designated
     destination drive  Source for Desmet C

     program operates by building a MS-DOS command,

                     drive.\command.com /c

     an argument or "command tail",

               copy source:filename destination

     and passes these 2 strings to MS-DOS via exec()

                  exec(command, command_tail),

     Unlike UNIX, control is returned to the calling program
     after exec() is executed
*/

#include <stdio.h>
main(argc,argv)
int   argc,
char  *argv[];
{
    char cmmd_tail[30],     /* arg to pass to command.com     */
         cmmd_com[20],      /* drive + string "command.com"   */
         fname[14];         /* name of file to copy           */

    if (argc < 3) {         /* not enough arguments    */
        puts                /*     display usage */
        ("usage  batcopy src_drv  dest_drv  [cmmnd_spec]\n"),
        exit(0);
    }

    if (strlen(argv[3]) == 0)    /* build command.com string */
        sprintf(cmmd_com, "%s\\command.com", "a:"),
    else
        sprintf(cmmd_com, "%s\\command.com", argv[3]),

    puts("Enter filename(s), press <RETURN> to end \n"),
    for (;;) {
        scanf("%s", fname);         /* get file(s) from user  */
        if (strlen(fname) == NULL)
            exit(0),                /* no name, so exit  */
        sprintf                     /* form command string */
        (cmmd_tail, "/ccopy %s%s  %s", argv[1],fname,argv[2]),
        exec(cmmd_com, cmmd_tail); /* execute command string */
    }
}
```



Mainstream Computing

Joseph Katz
103 South Edisto Avenue
Columbia, SC 29205

It's hot. I know the heat wave this July has been hard on many parts of the country, but it has absolutely devastated South Carolina. The *average* temperature in Columbia has been 100 degrees this month. It's dry, too. We've had some storms, mostly thunder and lightning and not enough rain to do any good. Janet and I actually look forward to HUGCON V as an opportunity for a change in weather. (Matthew has a new screwdriver and wants to visit all the exhibitors' booths.) You know things are bad when anyone cools off with dreams of visiting Chicago in August.

During one of those electrical storms something happened to the power service in an old residential area downtown revitalized as professional offices. They're well gentrified, but apparently not all well electrified. Nevertheless, many of them have microcomputers. You know what's coming now, don't you? I got some calls for help, one from someone with Zenith equipment. His surge protectors had gone bad overnight, he said. Gad. They weren't surge protectors at all, just Radio Shack power strips — the kind of thing that has a sensor on one of six outlets to turn them all on when the computer is powered. That power strip had been fried. I'm no technician, as you well know, but I've ducked into a chicken house once or twice. What my friend had on the floor looked like something dropped from a bucket. We tested the "grounded" outlet from which it really had come: no ground. If there's any moral in this story, I guess it's something like "Don't fool around with Mother Nature." Don't cheat on the electrical code, and get surge protectors that really are.

I've been lucky to have my air conditioned office for a cave and oodles of interesting things to occupy my time. Periodically I squint through the venetian blinds, then — like a mole — burrow back. As for a few of those interesting things . . .

Timeslips, A Time Accounting Program For Consultants

I enjoy the *doing* of consulting. It's the bookkeeping and billing I hate. Those scraps of paper on which I keep track of time and expenses hide whenever I need to total them. Of course I also loathe totaling them. So I often wind up underbilling, sometimes by considerable amounts, and I usually send my bills late. I did have the ultimate billing system once, but she quit. When she did, I began looking for a software package that would take over the work. That was a few years ago. Although I've looked hard as well as long, I never found anything really suitable for me.

All the time accounting and billing systems I tried were variations on a theme. They were aimed specifically at law firms that bill seven or more "timekeepers" (anyone whose time is billable). Billing formats in those packages were inflexible, Procrustean even for some of my lawyer friends. All the packages used lots of disk space for their many program modules, and their multitudes of data files gradually swelled like a new sponge in water. Huge. Slow. Clumsy. They were designed for professionals with more extensive accounting needs than mine. Within a short time after I learned each new package, I returned to my old, slothful ways. And I lost money.

Timeslips is so different from the others I've tried that it repaid more than its selling price the very first week I used it. It's a real find. This package is aimed generally — not only at lawyers, but also at ad agencies, graphic artists, accountants, and writers working to time. It's aimed at "consultants" in the broadest sense. Billing formats, therefore, can be customized to a reasonable extent, so you're not forced into unseemly invoices with items that read "In the matter of": you have them say what you want. The total storage capacity can be apportioned according to your own mix of timekeepers, clients, and transactions, and it therefore seems a remarkable tool for sole proprietorships and small firms in which everyone has some responsibility for keeping track of time and expenses. Keeping track is still not as much fun as the consulting itself, but I haven't seen anything else that makes it as easy for me as *Timeslips* does.

This package uses an imaginative approach that separates the functions of time recording and time billing. There's a separate module for each: *TSTIMER* is used to record time and expenses; *TSREPORT* is used to bill them. Each module does some nifty other things too, but each is an independent program that shares the same data files. The physical triad — *TSTIMER, TSREPORT,* and the data files — reflect a logic that makes *Timeslips* extraordinary and give it extraordinary capabilities.

Each timekeeper in a firm can keep track of his own time and expenses with a copy of *TSTIMER* and his own data files. Then, at billing or reporting time, those individual data files can be combined. Simply copy them to floppy diskettes, take the diskettes to any computer that has *TSREPORT,* and tell it to combine the files. Of course any computer can have its own complete copy of *Timeslips,* including *TSREPORT,* but that's not necessary. The package's modular construction makes redundancy avoidable.

TSTIMER on a computer near each timekeeper's desk avoids other kinds of redundancy. Its metaphor is, as the package name implies, a "timeslip." Timeslips are contemporaneous records kept by lawyers of their work for a client. Instead of buying pads of timeslips, you use *TSTIMER* to create them as you go. With the program running, you press F2 to create a new slip. What you are doing, of course, is creating a new record in the data files. You determine when you create that record whether it is for billable time or billable expenses. Depending on your choice, *TSTIMER* shows you one of two screens that differ only in details. You're not forced to reorient yourself to an entirely different set of demands.

If it's billable time you're recording, *TSTIMER* displays a timeslip. It has blanks for such things as timekeeper (the lawyer or consultant), account (perhaps the client or a project), and activity (a billing category such as "phone call from client" or "alterations in galley proofs"). There's one block in which you can describe the work or make other comments (take up to four lines, and the program will do automatic word wrapping); a second block for an optional reference number (you determine its significance and *TSREPORT* can use it for sorting); a third block for the beginning and ending dates of an activity; and two blocks for time. One is for an activity's projected time, the other is for time actually spent on the activity.

You can enter the latter time manually, or you can have *TSTIMER* time an activity and enter the elapsed time automatically. (Although a clock/calendar board is not a prerequisite to that function, it's about the best way to keep from cheating yourself or your client. In that connection, *TSTIMER* will not allow you to time two or more activities at once. Take that, Perry Mason!) So if the activity takes place in your office, *TSTIMER* all but fills out the computer timeslip for you and there's no need for a paper timeslip. You'll need paper to record an activity outside the office: North Edge Software's brochure says *Timeslips* comes with a pad of paper timeslips, but I didn't get any. An illustration in the brochure shows that the paper timeslips are a nice, simplified version of the computer timeslip. You — or an aide — should be able to transfer information from paper to the computer easily.

If it's billable expenses you are recording, *TSTIMER* displays an expense slip instead of a timeslip. The major difference between the two is that instead of blanks for projected and actual times, the blanks are for quantity (by units of whatever you bought) and amount (per unit). Use them as appropriate to your expense. For example, you might have taken a reimbursable trip by plane to the famed Sextant Towers. You create the slip and choose "travel" as the expense activity to record. Immediately the slip becomes an expense slip. You put "Roundtrip ticket (Studio Airlines), Columbia to Washington, July 2–5" as the description. The quantity is "1," the amount is "$365.79" (first class and lavish tips, right?). The rest of the slip is filled out as usual.

What's a really nice thing about all this is that you need know absolutely nothing about database managers, which is what *Timeslips* is, or even that you're using one. The program conceals all the tedious and confusing details that scare people new to computers. All you're doing is filling out slips, the way you always do — if your system is sensible and you are too. Even a lawyer could do it.

Beneath that simplicity is a great deal of capability. For example, you can mark either kind of slip "work in progress," "no charge," or "do not bill." The program itself will mark slips to show you things about their status, such as whether a slip has been billed, deleted by you, or exported for a combined bill. For another example, you can run *TSTIMER* as a normal program (from the DOS command line whenever you need it) or as a RAM RESIDENT program that leaps to the ready at the press of some keys. The latter mode makes it easy to do things like time an incoming phone call; the former keeps *Timeslips* out of the way if you're an occasional user. I haven't figured out yet which is the better mode for me.

TSREPORT is used only when you want a product from the data files — for example, bills. Whoever prepares the bills makes the following decisions, in order, by menu selections: which data file (RAM resident, single disk file, or combined file); which slips in that file (perhaps by work-in-progress and consultant, so you know who to prod a little); and in which order to sort them (maybe by outstanding amounts, so you know who to prod first). Then you tell the program to do the selection and make the sorts. It's fast. Next, you choose the report format from one of seven: detail, summary/graphs, IRS, billing worksheet, timeslips, bills, or user-defined. Within those seven categories are every kind of report I can imagine, more than I can summarize here. My hunch is that if you can't find the right one for you, you really need a custom program — the kind you can't afford anyway, so why bother mentioning it? And next you choose the device to which you want the report sent: parallel, serial, display, file, or graph. Those terms include just about the only computerese in *Timeslips,* but they seem unavoidable. If you can hook a device — printer, plotter, storage, cable, or whatever — to your computer, *TSREPORT* can output to it.

I thought at first that some of the report formats and devices were little more than froth and window dressing. I was not right. My first graph, for example, astonished me with what it revealed about how I should be spending my time most profitably. Would that I had *Timeslips* before I made some unprofitable commitments. Damn. File output makes available some fascinating possibilities. You'll have thought about those that start with incorporating a report image into a document with your word processing program for further massaging. What really interests me, though, are the possibilities created by *TSREPORT's* options for output in other formats. One, called "ASCII delimited," produces a file with the data comma-delineated, carriage-return terminated. That gobbledygook means you can pull those files into *dBASE, DataStar,* or any other database manager; into Lotus' *1-2-3, SuperCalc,* or any other spreadsheet; and into *MailMerge* or, in fact, any other pro-

gram that lets you import data. It's the *lingua franca* of data exchange. In short, you can use *Timeslips* to produce data that is absolutely portable.

Installation of *Timeslips* takes a few minutes. There are so few decisions to make that it's hard for even a timewaster to waste time. You can, however, customize several features of the program during installation. If you decide on different ways to handle those features later, you can reinstall easily enough. Just be careful during reinstallation to not overwrite any existing data files with new files of the same name: to be really safe, work on a backup of the program and the data files. I can't think of any other errors you could make that would be fatal. *Timeslips'* requirements are modest: a true compatible with a minimum of 256KB of RAM, some kind of monitor, and either two floppy diskette drives or a hard disk drive. You'll have no problems with this program on any Heath or Zenith compatible. To print bills you'll need some kind of printer too: *Timeslips* doesn't care which. You can print bills and other reports on fanfold paper or single sheets, on billheads or on blanks that need your company name and address: *Timeslips* can handle anything. The manual is clear, direct, and well written. Smart help is available within the program, automatically giving you the explanation appropriate to the option on which the cursor rests. If you need more help than there is on the screen, look at the left-hand corner of the message and you'll see a reference to the applicable page in the manual. The only thing I don't like about *Timeslips* is that *TSREPORT* always centers your company name and address on reports. I prefer them flush left. My way suits my taste, but I can live with *Timeslips'* way. You should know that I have only skimmed the surface in describing *Timeslips:* it does much, much more than I've said. Although I could say more about this remarkable program, I think I've said enough to give you the picture. If you're in a situation similar to mine, buy *Timeslips.* It costs $99. You'll love it. *Timeslips* is not copy protected, of course.

Mirror Reflects *Crosstalk*

Remarkable in its own way is SoftKlone Distributing's *Mirror.* I was curious about a communications program advertised as a $49.95 clone of the $195 *Crosstalk XVI,* only better. Impulse finally made me bite. Well, *Mirror* is a $49.95 clone of the $195 *Crosstalk XVI,* only better.

If you're at all familiar with *Crosstalk XVI,* your jaw will drop the first time you run *Mirror.* It looks like *Crosstalk,* talks like *Crosstalk,* and acts like *Crosstalk,* but it's not *Crosstalk* purloined. For one thing it's bigger, by a hefty 94KB. For another thing it's really a different program made to look like *Crosstalk:* missing, for example and thank goodness, are the irritating wisecracks displayed when you run *Crosstalk* without the name of a script. For another thing it's more capable: it emulates a few more terminals, supports a few more file transfer protocols, and supports the crucial XMODEM protocol (crucial because it's the standard for bulletin board systems) more thoroughly.

Some of the added capabilities seem minor or cosmetic, but others are really significant improvements. My absolute favorite is the ability to run *Mirror* as a background program. With *Crosstalk* my computer was tied up during a long file transfer and I twiddled my thumbs or read. With *Mirror* I initiate the transfer, press both shift keys to drop out of the program, and run my word processing program. Nope, *Mirror* hasn't lost a byte yet. Pretty close to the background capability in my estimation is the *Mirror* manual. It's a whopper, a thick paperback. There's nothing wrong with *Crosstalk's* manual: it explains clearly how to use the program. *Mirror's* just goes much further. It starts with a nice introduction to the theory and practice of computer communications. It ends with a set of appendices on the details of cabling and modem installation. They're the kind of details a casual user discounts until the need arises. Then they become priceless details. On them *Mirror's* manual is about as good as the manual for *MITE,* another communications program sold from a company in Tallahassee, Florida, with similarities that increase my already high belief in the value of having good neighbors.

After I shut my jaws, I transferred my *Crosstalk* script files to *Mirror* and began running with only one slight hitch: the January, 1986, release of *Crosstalk XVI* introduced a command to set the length of the Break signal. *Mirror* doesn't use it, so it objects when encountering the command in a script file. All you do is edit out the line that begins "LBreak" and from then on everything runs nicely. *Mirror* introduces a few more commands, which may be reset in script files but which are not missed if you use a *Crosstalk* script. *Mirror* is a clever clone that costs $145 less than the original. Neither *Crosstalk* nor *Mirror* is copy protected, of course.

DOG Ends The Nagging Heartbreak Of Fragmentation

It's a good idea to "defragment" a hard disk periodically. I do it to mine at least two or three times a month. This past month I was so bored by being locked in my office that I did it much more often. "How did you spend your summer?" "I defragmented my hard disks." "Ah. Do you often have that problem?"

Here's the problem. Every file you erase leaves a chink of deallocated space on a disk. MS-DOS fills those chinks with pieces of files you next create or expand. So when you copy a new program from its distribution disk to your well-used hard disk, for example, parts will be scattered helter skelter. Each time you run the program, therefore, the drive heads move all over the place to gather the program's non-contiguous parts. Those are the fragments. You need to reassemble them periodically.

Don't shrug off that problem because you seldom or never add programs. Data files really are the worse problem. You know you create a file whenever you start a new document with your word processing program. You may not know, though, that your word processing program routinely creates and erases files on its own. The next time you use *WordStar,* type *CTRL K F* when you're editing a document. You'll see a file directory appear. Scroll through it for EDBACKUP.$$$: it's a temporary file opened every time you use *WordStar* and erased every time you exit it. Any program that can handle files bigger than about 32KB probably does so via temporary files, and most of them open the temporaries immediately, as a matter of course. They're not always written in the same place either. Where they go depends upon the location of space available at the time. They continually produce chinks MS-DOS keeps filling and creating. I don't have to say that programs like spreadsheets and database managers do the same. Imagine the trips read/write heads must take through a big database whenever you search for information.

A highly-fragmented disk wastes time and causes added wear on the mechanical parts. I can't be sure of how many drive failures have been traced to prolonged, excessive fragmentation, but I am positive about the loss of time. Every so often, when I've been negligent about hard disk housekeeping, things drag — sometimes to a crawl. It takes apparent eons for my word processing program to open an existing file or save a new one.

You're probably wondering why, if the problem is *so* serious, MS-DOS doesn't provide a diagnostic tool. Well it does, you silly: CHKDSK.COM. The Microsoft MS-DOS manuals for Version 2 and 3 (the two versions so far that support hard disks) even explain the problem and give a solution. The trouble is that all these explanations are too half-baked to be much good. So foggy are they that it's easy to miss the existence of the problem altogether. Take a look at "Advanced Concepts" in the "command description" of *CHKDSK* and you'll see what I mean. Impenetrable, right?

The way to find out if *any* files on a disk are fragmented is to issue this command: *CHKDSK *.**. The *Manual* says you have to specify the filename, as if any relatively sane person will run *CHKDSK* once for each file on a hard disk. The *Manual* neglects to mention you can use wildcards in place of filenames. You can and should, just as I've shown you. I guess it's a trick, of sorts. What you want to see is the *CHKDSK* report conclude with, "All specified file(s) are contiguous." If it's not there, they aren't.

That's the reverse of what I expect in a diagnostic. I want quiet until there's a problem, at which time the alarm should sound. *CHKDSK* works the other way unless you know another little trick: *CHKDSK *.*/V*. It combines the wildcards with a switch that tells CHKDSK to be verbose, in which case you would get warnings about specific problem files. Here's one: "D:\OPTIMIZE.NOT Contains 2 non-contiguous blocks." If you own a hard disk and it's nearby, fire it up right now and give that command a try.

Aha! You have the problem, don't you? So what do you do about it?

The *Manual* says "You might want to copy a noncontiguous file to another disk because copying records the files contiguously." That is techie talk disguised as English. What someone is trying to say is this: copy the files to another disk, erase the original files, then copy the files back. If "another disk" is a floppy diskette, moreover, use *COPY* instead of *DISKCOPY*: the latter makes a mirror image of the disk, which is what you *don't* want; the former works on a file-by-file basis, which means it assembles the file's pieces in order to make the copy, which is what you do want. Remember that the mere act of making the copy does nothing at all for the originals. Remember too that you have to erase the originals before copying the copies back to the hard disk, otherwise, everything will wind up in exactly the same condition you were trying to repair.

CHKDSK, COPY, and a bunch of floppy diskettes are all you need to keep your hard disk in shape if you're a conscientious housekeeper. In fact that's all you need if you do regular file-by-file (not mirror-image) backups. (That's techie talk too, I guess. It's contagious. What I'm trying to say is, don't use *DISKCOPY*.) Just add two more steps to your complete backup routine, when you back up the entire hard disk: first, when the backups are finished, erase your entire hard disk; next, restore the files you've just backed up. All the files will be contiguous.

Do make absolutely sure the backups are good before you delete the contents of your hard disk, though. I have been done in by the MS-DOS *BACKUP* and *RESTORE* utilities often enough so I don't trust them. (There's a bug in the DOS 3.1 *BACKUP*, by the way, which affects files that span backup disks.) And keep in mind that you have an all-or-nothing proposition here: the smallest part of a hard disk you can do at one time is a logical drive — Drive C or Drive D, for example — because fragmentation knows not the bounds of subdirectories.

Neither does it know the difference between hard disk and floppy diskettes, by the way. If you have a diskette that has seen hard use, run *CHKDSK *.*/V* on it and see what you learn. If it's horribly fragmented, use *COPY* to make a backup and see if the backup works faster and with less drive activity (listen to the drive and watch how long the red light stays on). I'd be particularly careful if you use one of the quad density floppy diskette drives in a Z-241 for working disks instead of for archiving or transferring files.

If you suspect me of technobabbling the past few paragraphs, you probably shouldn't be trying to delete the contents of your hard disk. You might become bloodthirsty if you make a mistake. Instead, you need a program that will defragment the hard disk "in place," without copying the programs to floppies, or tape, or cartridges. They also work by making copies and erasing the originals, because the only way to defragment a file is to move it together physically, but they shuffle the fragments in and out of whatever space is available on the hard disk drive. It's like a demon playing The Towers of Hanoi.

DOG, an acronym for "Disk OrGanizer," is — despite its incredibly awful name — no dog at all. It not only will defragment a disk, but also can reorganize it for maximum speed. I guess I'm not enough of a speed demon to do the required tinkering with script files: I use *DOG* just for defragmentation, which it does quickly and effectively.

DOG has been available since June 1, 1986. You can download it from CompuServe's IBM Hardware SIG. If I remember to take a disk with me to HUGCON V, give it to Jim Buszkiewicz, and Jim is willing, you might also be able to download it from HUGPBBS. But you'll probably spend less money and gain the advantage of a pipeline to support if you order it directly from the author, G. Allen Morris III. The documentation says to send him whatever you think the program is worth, but that's if you get it from a bulletin board system. Since he asks for a $20 contribution from businesses, a check for that amount sounds right for a mail order too. *DOG* is not copy protected, of course.

SoftLogic's *Disk Optimizer*

If you're a constant reader, you know that I normally don't call attention to the *absence* of copy protection in software. I'm doing it now to make a point. The packages I've discussed so far are extraordinary and obviously the result of extraordinary labors. None of them are copy protected. *Optimize*, the defragmentation program in SoftLogic's *Disk Organizer* package, *is* copy protected. The scheme is Superlok from Softguard, and it's set up so you can install *Optimize* on a hard disk only twice. For that you pay $50 (plus $5 for shipping and handling if you order from SoftLogic by mail). If you want "a not copy protected version," you request it from the service department, sign a license agreement, and pay *another* $25. That's $75 for a utility program.

Optimize is indeed a nice little utility program. It defragments a disk and it's truly easy to use, although neither better nor easier than *DOG* if all you want is defragmentation. The program just needs to know which drive you want defragmented: it defaults to the logged drive, but you can tell it to defragment any drive on your system. That's all you have to do.

The time *Optimize* spends in its work depends on factors such as total drive capacity, amount of free space, and amount of free RAM. *Optimize* takes about forty-five minutes for the cramfilled 10MB of the Drive C on my H-158, which has 640KB of RAM. It's slower than *DOG*, but I don't mind that. SoftLogic says the process

takes less time if the drive has been optimized before, but the time is about the same even if I rerun the program after it has just finished a defragmentation. I don't mind that either, although *DOC* is smart enough to know when a disk doesn't need to be defragmented. If I need to use my computer before the program has finished, a tap of the spacebar brings everything to an orderly halt. It's not a good idea to do that, because interrupting *Optimize* has resulted in more fragmentation than before I started. I don't even mind that very much.

What I do mind is the concept I have of the way SoftLogic does business. Copy protection on a program like *Optimize,* which changes the structure of a hard disk, works against the buyer's interests even more than copy protection on other kinds of programs. The price for the copy protected version seems too high for an unprotected version, and the price for the unprotected version seems flatly outrageous. I wouldn't be so disturbed, though, if SoftLogic gave you the facts before you bought so you could make an informed decision about what you were getting for your money.

It doesn't. SoftLogic doesn't say anything about copy protection in its advertising, so it doesn't say anything about the extra-cost unprotected version either. You, therefore, enter the transaction with the seller having withheld material facts you discover only after you buy the goods.

SoftLogic apparently does so intentionally. The company's policy is to see what a reviewer says about one of SoftLogic's products before sending a review copy of the next. I've been reviewing for almost twenty-five years now and I've never had any other company even hint at such a policy. It taints the review process. So while I like *Optimize,* I most certainly recommend *against* buying it. I just don't like the way SoftLogic does things.

MonoGrafx And *MACE*

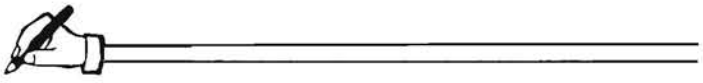
Nope, not copy protected, and no one demanded to vet my opinions as a condition of anything. Analytx International, which had advertised both a copy protected and unprotected version of *MonoGrafx,* dropped the copy protected version around the time I was praising the program and damning the protection. As you could gather from my last column, I think *MonoGrafx* an absolute gem. The program does quick, straightline graphics just right for batting out elegant forms. I did some timeslips, for *Timeslips,* with it and had one of Janet's favorite quickcopy places run off a batch. Just right.

MACE is the "MonoGrafx ASCII Character Editor." It's a tool for editing or creating characters in a *MonoGrafx* printer driver. You do it by setting a bit map for each character. First you get an existing *MonoGrafx* character file by selecting it from a menu. Then you select the first character to edit. You cannot change the standard alphanumerics (ASCII 32–126), but the so-called "extended ASCII character set" (128–255) are yours to do with as you will. (ASCII 127 is DELete, which for various reasons cannot be used to produce a character.) The editor properly displays a grid representing the pins on the printer's print head. You mark the pins to be fired whenever *MonoGrafx* encounters that character. Since it's an extended ASCII character (an IBM graphics character), you produce it in *MonoGrafx* by typing the character code on the keypad while you depress the ALT and SHIFT keys: 128, for example, is the cedilla. What you see in *MonoGrafx* is the cedilla when you type the character: remember, it's the printer driver, not a display driver, you've modified. But what gets printed instead of the cedilla is the character you've created.

MACE is very nice, very easy to use, and performs just as it should. Unfortunately, Analytx International does not supply graphics design talent with *MACE.* If you have some, you'll be able to use *MACE* to create any special characters you need printed. They'll become part of the *MonoGrafx* driver for your printer, so you only need to do them once. Even with my lack of talent, I confess, I did a few symbols that I really have wanted for some time. One is the copyright symbol: it's boxy and sort of squiggly at the same time (maybe I do have hidden art talent after all). Nevertheless, it's a veritable copyright symbol. Trust me.

By the way, *MicroGrafx* has a new manual. It mainly is a conflation of its predecessor's parts, to keep you from having to jump all over the place when you need information. I wasn't at all troubled by the predecessor, mostly because *MonoGrafx* is so well designed that I'm not sure it even needs a manual.

Hmm. That was not a terribly bright thing for a writer to say, was it? It's the heat.



Mainstream Mailbag

Z-148 Performance And Disk Drives

Dear Dr. Katz:

I thought the Z-148 PC and Z-158 PC were essentially identical except for the Z-158 having expansion slots and a larger power supply. If so, why does The Norton Utilities' SI program calculate a PC performance index of 1.7 on my Z-148 (set at 8 MHz, of course) and only a 1.5 on your Z-158 (*REMark,* July 1986, p. 19)?

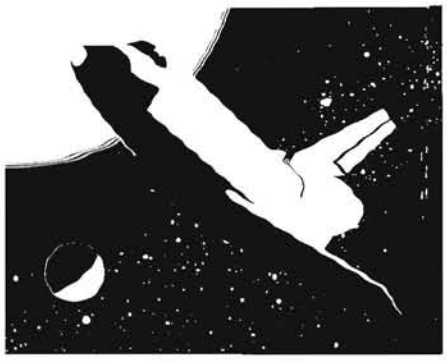
Barry Hindin
Reynoldsburg, OH

The Z-148 at 8 MHz is faster than the Z-158 at 8 MHz. The 148 doesn't use wait states; the 158 does. I haven't even seen a 148, by the way, because I haven't been able to crack the two companies involved. I surely can't afford to buy everything they produce just so I can write about them in magazines that support their customers. But what I hear about the 148 indicates you bought one heck of a fine entry level machine. The next time you hear somebody brag about their Leading Edge or no name clone, you are entitled to one condescending smile.

Because I've never even seen a 148, I can't really answer your question about adding a hard disk drive without losing one floppy. I'd guess that you could add an external hard drive. If someone writes me about a different way, I'll try to publish the information.

Shareware Communications Programs

I'm grateful to Bob Montante of Bloomington, Indiana, for a good, long letter suggesting a look at two shareware communications programs, PibTerm 3.2.5 *and* ProComm 2.3. *Mr. Montante likes them — especially* ProComm *— for "good terminal emulation and flexible file transfer capabilities." The letter is too long to publish here, but I pass along Mr. Montante's recommendations with my gratitude for them. As a small token of that gratitude, I'll pass along two tips to Mr. Montante: don't drink from the WCTU fountain downtown, and don't shout "I'm a college boy in the stonecutters' bars.*



EXPLORE
NEW WORLDS WITH
HUG
GAME SOFTWARE

HOME FINANCE SYSTEM VERSION 2

—An extensive Home Finance System that keeps track of checking, asset accounts (cash, savings, IRAs, CDs), and regular bill payments. Let your printer write your checks for you on **any** business-sized check (design your own check format).
—Checks have user defined codes and a separate flag for tax deductible items.
—Many reports, including listing all checks, or checks by codes or tax flag.
—System consists of 130 page users manual with 5 program disks (5-¼") and a sample data disk.

Hardware: H8/HZ89 (64K) or HZ100 with 2 disk drives. Any Heath®, Zenith® or other printer.
Software: CP/M or CP/M-85/86 (Ver. 2.2) and MBASIC 5.21 for CP/M.
Order: Complete System $89† (specify hard or soft sector 5¼", HZ89 or HZ100). Manual alone $21.†
MasterCard/Visa accepted, please include your phone number.
†Prices include shipping.

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

S & K Technology, Inc.
Quality Software for Heath/Zenith Microcomputers

For the Z100...

WatchWord® **$100.00**
The ultimate in word processing with speed and power. See subscripts, superscripts, underlining, and boldface directly on the screen. Create your own fonts and special characters. Other features include centering, formatting, automatic horizontal scrolling with long lines, large file capability, split screen, macros, color, and an extensive configuration facility. See reviews in Remark (July 1985) and Sextant (Jan-Feb 1985, Sep-Oct 1985). Requires 192K RAM.

The Resident Speller™ **$100.00**
Spelling checker for use with WatchWord.
Checks as you type from inside WatchWord or checks a file.
Includes a 50,000 word expandable dictionary.
Requires 192K RAM to check a file.
Requires 300K RAM to check as you type.

Demo disk for both **$ 3.00**

For IBM compatibles including the Z150 and Z200 series...

PC WatchWord® (New) **$ 99.95**
The ultimate in word processing for the sophisticated user. Most of the features of the Z100 version except for screen fonts. Requires 256K RAM.

PC Resident Speller™ **$ 99.95**
Spelling checker for ASCII files such as those created with WatchWord, WordStar, WordPerfect, PeachText, and VolksWriter. Includes Strike. Requires 256K RAM.

Strike™ **$ 49.95**
Adds as-you-type spelling checking to your word processor. Works with the word processors above and also with DisplayWrite, MultiMate and PFS:Write. Requires 100K RAM in addition to that used by your word processor.

Demo disk for Strike and The PC Resident Speller **$ 2.00**

Demo disk for PC WatchWord **$ 2.00**

Texas residents please add state sales tax.

S & K Technology, Inc., 4610 Spotted Oak Woods, San Antonio, Texas 78249, (512) 492-3384

ZPC Update #9

Pat Swayne
HUG Software Engineer

This is the ninth in a series of articles in support of ZPC (HUG part no. 885-3037-37), a program that allows you to run IBM PC software on H/Z-100 (dual processor) computers. This is also the third in the series dealing with ZPC Version 2. If you missed the last two REMark issues and would like the last two ZPC Updates, send me a self-addressed stamped envelope (c/o HUG) and indicate that you want ZPC Update #7 and/or #8.

In this ZPC Update, I will present an improved version of the ZPC Hardware Support circuit. On the software side, I will present a correction to the SETZPC patch in ZPC Update #7, instructions for running Lotus Symphony and Microsoft Word version 3 under ZPC, a patch for a new release of Word Perfect version 4.1, and a utility that can cause ZPC to display a steady (not blinking) cursor.

ZPC Hardware Support Circuit Improvement

There are two problems with the original design of the keyboard interrupt section of the ZPC Hardware Support circuit (ZHS) that was presented in the April 1986 issue of REMark (and reprinted in the ZPC documentation). One problem is that the interrupt pulse generated when port 60 or 61 (hex) is accessed is not long enough. The other problem is that the 8259 interrupt controllers in your computer may not be fast enough to respond to the interrupt in time for the software to work properly. A HUG member called in and discussed these problems with me, and he was to send in a schematic of his changes to take care of the problems, but it did not arrive in time for this article. However, based on what he told me, I have come up with some changes to the board that probably duplicate his work, even though I used a different IC than what he suggested.

I have written a short program that can test your ZHS board and tell whether the interrupt generates is too short or too fast. If you type in and run the following BASIC program, it will create ZHSTST .COM, which can then be run to test your board. The source code for ZHSTST.COM is at the end of this article.

```
10 REM THIS PROGRAM CREATES ZHSTST.COM
20 DEFINT A-I:OPEN "O",1,"ZHSTST.COM
30 S=0:S1 = 24313 :FOR I=1 TO 243
40 READ B:S=S+B:PRINT #1,CHR$(B),
50 NEXT I:IF S<>S1 THEN PRINT "TYPING ERROR!":END
60 CLOSE #1:LOCATE 23,1:PRINT "DONE!":SYSTEM
70 DATA 51,192,142,216,190,60,1,86,30,196
80 DATA 60,199,4,120,1,140,76,2,14,31
90 DATA 228,243,36,247,230,243,228,241,36,127
100 DATA 230,241,50,219,186,96,0,236,60,85
110 DATA 116,10,179,1,236,236,60,85,116,2
120 DATA 179,2,186,218,3,185,100,0,236,168
130 DATA 1,116,7,226,249,128,203,4,235,13
140 DATA 185,100,0,236,168,1,117,5,226,249
150 DATA 128,203,4,186,175,1,10,219,116,8
160 DATA 129,194,17,0,208,235,115,248,180,9
170 DATA 205,33,10,219,117,240,31,94,137,60
180 DATA 140,68,2,205,32,0,0,0,0,0
190 DATA 46,162,115,1,176,32,230,242,230,240
200 DATA 46,137,54,116,1,46,140,30,118,1
210 DATA 94,31,251,138,68,255,30,86,46,142
220 DATA 30,118,1,46,139,54,116,1,60,236
230 DATA 117,8,131,250,96,117,3,176,85,207
240 DATA 46,160,115,1,207,13,10,90,72,83
250 DATA 32,66,111,97,114,100,32,79,75,36
260 DATA 32,32,13,10,56,50,53,57,39,115
270 DATA 32,115,108,111,119,36,32,32,32,13
280 DATA 10,80,111,114,116,32,54,48,32,105
290 DATA 115,32,98,97,100,36,13,10,86,105
300 DATA 100,101,111,32,112,111,114,116,32,98
310 DATA 97,100,36
```

After you create ZHSTST.COM, copy it to your system disk and enter

```
ZHSTST
```

at the system prompt, and hit RETURN. If ZHSTST prints "ZHS Board OK", your board has passed all the tests. If it prints "8259's slow", it means that your 8259 interrupt controller chips are not fast enough to respond to the interrupt as generated by your board. If it reports "Port 60 is bad", or if the program "hangs up", or if a "WILD INTERRUPT" message appears, it could mean that the interrupt pulse generated by your ZHS board is not long enough, or that the port 60 section of your board is not working (U1, U2, and U3). If ZHSTST prints "Video port bad", it means that the video port section of your ZHS board is not working (U4, U5, and U6).

Figure 1 shows the change that will lengthen the interrupt pulse from your ZHS board. It is a good idea to make this change even if ZHSTST does not report any problems. U1 on the ZHS board is replaced with a 74S74 (Heath p/n 443-900) or a 74AS74 (443-1209) and wired as shown in the drawing. You should also replace U2 with a 74S30 (no Heath p/n) or a 74AS30 (443-1289), and U3 with a 74S04 (443-897) or a 74AS04 (443-1205). Notice that the sINTA line (S-100 pin 96) is no longer used, and instead the INT line (pin 73) is used to time the pulse. The pulse generated by this circuit is actually held active from the time the port is accessed until the trailing edge of the INT pulse, which gives the 8259 plenty of time to process the interrupt.

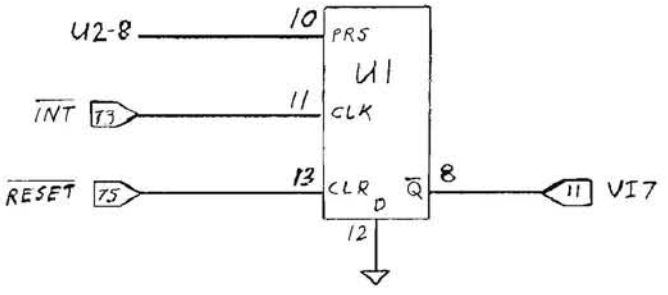


Figure 1

If you have 8259s in your computer that are too slow, the above change may not take care of the problem. The member who called me suggested that if the RDY line on the S-100 bus was held low from the time the port was accessed until the time the INT line became active, it might take care of the problem. I wired up a gate on my ZHS board to do that, and it did not make any difference on the one Z-100 we have here at HUG that had slow interrupt controllers. So I opened up that Z-100 to see what brand of 8259s it had, and found them to be NEC chips. I replaced them with Intel 8259s and that corrected the problem. The 8259s are ICs U208 and U209 on the main board, and are under the video board.

Figure 2 shows an additional change that can be made to the board to improve the video section. In this change, the gate of U5 that uses pins 9 and 10 is taken out, along with the .001 microfarad capacitor, and the output of U5 pin 7 is connected directly to U6 pin 14.

ZPC Update #7 Correction

There is an error in the patch for SETZPC that was given in ZPC Update #7. Near the bottom of page 59 (in the July issue), change the line with E7FC to E7CF. Near the top of page 60, change the line with 6FC,12 to 6CF,12.

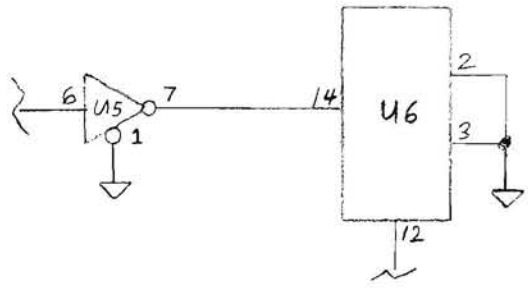


Figure 2

Lotus Symphony

The ZPC documentation is partly in error in stating that Symphony will run under ZPC without being patched (after copy protection is removed). It will only run in video mode 7 (PC 7), and you cannot create graphs in that mode. To allow Symphony to run in a graphics mode, you can patch it using the FIXLTS program that was presented in the last ZPC Update for patching Lotus 1-2-3 release 2. However, the screen driver file in Symphony has a different name, so you must create a batch file that will rename the driver file, run FIXLTS to patch it, and rename it back to the original name. The batch file should be called FIXSYM.BAT, and should contain these lines:

```
REN %1LOTUS.SET 123.SET
FIXLTS %1
REN %1123.SET LOTUS.SET
```

Copy this batch file and FIXLTS.COM to your ZPC System disk, and enter

```
FIXSYM d:
```

and hit RETURN, where d: is the drive containing the Symphony program disk to be patched. As with Lotus 1-2-3, you should run "Install" BEFORE you make the patch. Set ZPC to mode 7 to run "Install", and select first time installation from the main menu. Answer Y to the question "Can your computer display graphs?". When asked "How many monitors?", select one. When asked "Do you want graphs and text together?", you can select Yes or No, but you will get a faster display if you select No. Select "IBM Color Card" from the list of graphic cards, with either "color monitor" or "single color monitor". After you have finished the installation procedure, patch the driver on the Symphony program disk with FIXSYM, and put ZPC in mode 3 to run Symphony.

Microsoft Word Version 3

To patch Microsoft Word version 3 for use with ZPC, add these lines to your PATCHER.DAT file:

```
Microsoft Word version 3
Insert the disk containing WORD.COM.
WORD.COM
674A,0,0
6751,B0
z
```

Then use PATCHER to make the patch as instructed in your ZPC Version 2 manual.

Note: ZPC Update #7 (July 1986) contained a patch to correct a bug in the PATCHER program. To ensure that all patches are applied correctly, be sure to patch your PATCHER.COM if it is dated 3-25-86.

Word Perfect

There is a release of Word Perfect version 4.1 that is different from the release that I used for the patches in PATCHER.DAT. This release is dated 2-18-86 (the date on WP.EXE). If you have this release, add these lines to your PATCHER.DAT file:

```
WORD PERFECT version 4.1, 2-18-86
Insert the disk containing WP.EXE
WP.EXE
2957,90,90,90,90,90,90,90,90,90,90,90
116B6,90,90,90,90,90,90,90,90,90,90,90
121B4,90,90,90,90,90,90,90,90,90,90,90
1FD63,EB
z
```

Then use PATCHER to make the patch as instructed in your ZPC Version 2 manual.

A Steady Cursor

Some people prefer a steady (not blinking) cursor at times, especially when they run word processing software. On a real PC computer, there is no way to make the cursor stop blinking (at least, I do not know of a way), but with ZPC Version 2, a patch can be applied to make the cursor remain steady. I have written a program called BLINK.COM that can apply or remove the patch, so you can have a non-blinking or blinking cursor. If you type in and run the following BASIC program, it will create BLINK.COM.

```
10 REM THIS PROGRAM CREATES BLINK.COM
20 DEFINT A-I:OPEN "O",1,"BLINK.COM
30 S=0:S1 = 8935 :FOR I=1 TO 67
40 READ B:S=S+B:PRINT #1,CHR$(B);
50 NEXT I:IF S<>S1 THEN PRINT "TYPING ERROR!":END
60 CLOSE #1:LOCATE 23,1:PRINT "DONE!":SYSTEM
70 DATA 30,51,192,142,216,190,70,0,142,4
80 DATA 31,190,47,1,179,14,160,93,0,60
90 DATA 79,117,5,190,67,1,179,3,191,238
100 DATA 47,185,10,0,252,250,243,165,251,190
110 DATA 235,47,38,136,28,205,32,198,6,91
120 DATA 1,0,128,54,89,1,1,251,232,130
130 DATA 6,250,198,6,91,1,1,128,62,89
140 DATA 1,1,116,13,198,6,89,1,1,251
150 DATA 232,128,6,250,144,144,144
```

If you would like to assemble BLINK.COM, the source code is at the end of this article (following the source for ZHSTST.COM). To use BLINK.COM, copy it to your ZPC system disk, and enter

```
BLINK OFF
```

at the system prompt and hit RETURN, to turn blinking off. To restore blinking, enter

```
BLINK
```

at the system prompt and hit RETURN.

Upgrade Your ZPC

If you are still using ZPC version 1, you're missing the power and versatility of ZPC Version 2. You can upgrade by sending your original HUG ZPC disk and $20.00 to Heath User's Group, Attn: Nancy Strunk, Hilltop Road, St. Joseph, MI 49085. If you have both the old ZPC and the ZPC Support Disk, send both disks in and $15.00 for the upgrade. Make checks payable to: Heath/Zenith Users' Group.

Source Code For ZHSTST.COM

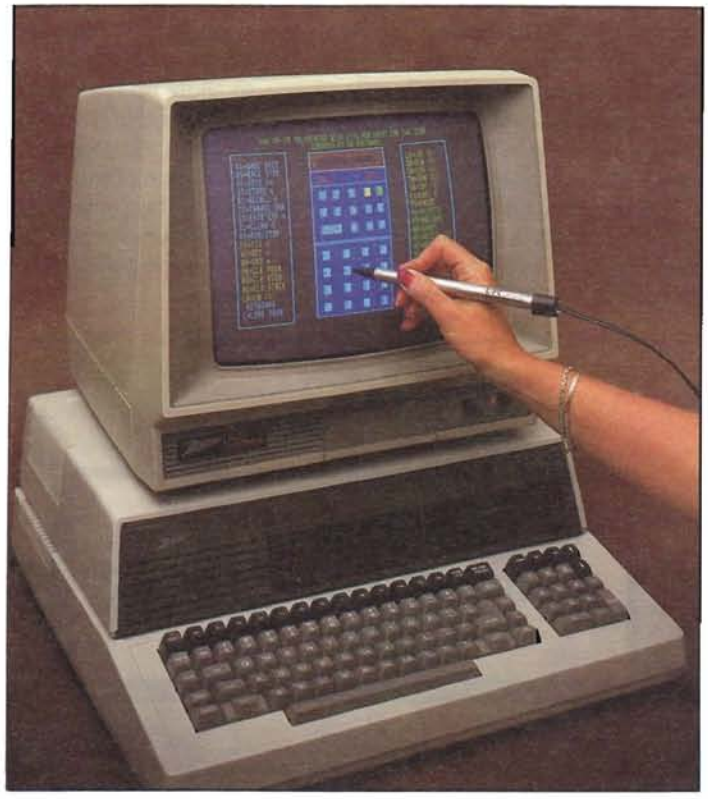
```
        PAGE    ,132
;       ZHS TEST ROUTINE
;       BY P. SWAYNE, HUG  04-AUG-86

MASTER  EQU     0F2H
SLAVE   EQU     0F0H
ZHSINT  EQU     4FH

JMPF    MACRO
        DB      0EAH
        ENDM

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

START   XOR     AX,AX
        MOV     DS,AX           ;POINT TO INT. PAGE
        MOV     SI,OFFSET ZHSINT*4
        PUSH    SI
        PUSH    DS              ;SAVE THESE POINTERS
        LES     DI,DWORD PTR [SI] ;GET OLD VECTOR
        MOV     WORD PTR [SI],OFFSET INT60
        MOV     2[SI],CS        ;SET UP ZHS INTERRUPT
        PUSH    CS
        POP     DS
        IN      AL,MASTER+1     ;SET UP 8259'S
        AND     AL,0FFH-8
        OUT     MASTER+1,AL
        IN      AL,SLAVE+1
        AND     AL,7FH
        OUT     SLAVE+1,AL
        XOR     BL,BL           ;CLEAR FLAG REGISTER
        MOV     DX,60H          ;GET PORT 60
        IN      AL,DX           ;READ IT
        CMP     AL,55H          ;ZHS WORKED?
        JZ      TSTVID          ;IF SO, TEST VIDEO
        MOV     BL,1            ,ASSUME SLOW 8259'S
        IN      AL,DX
        IN      AL,DX           ;TEST FOR SLOW 8259'S
        CMP     AL,55H
        JZ      TSTVID          ;THEY'RE SLOW
        MOV     BL,2            ,ELSE, MARK PORT 60 BAD
TSTVID: MOV     DX,3DAH         ;VIDEO PORT
        MOV     CX,100          ;SET A COUNTER
TVLP1:  IN      AL,DX           ;READ VIDEO PORT
        TEST    AL,1            ;TEST RESULT
        JZ      VIDOK           ,PASSED FIRST PART
        LOOP    TVLP1           ;GIVE HIM A CHANCE
        OR      BL,4            ;MARK VIDEO BAD
        JMP     SHORT REPORT    ;AND REPORT IT
VIDOK:  MOV     CX,100          ,FIX COUNTER
TVLP2   IN      AL,DX           ,READ VIDEO PORT
        TEST    AL,1            ;TEST RESULT
        JNZ     REPORT          ;PASSED SECOND PART
        LOOP    TVLP2
        OR      BL,4            ;MARK VIDEO BAD
REPORT: MOV     DX,OFFSET MSG1  ,POINT TO GOOD MESSAGE
        OR      BL,BL           ;IS ZHS GOOD?
        JZ      REPORT1         ;IF SO, REPORT IT
FNDBAD: ADD     DX,MSGSIZ       ,MOVE TO NEXT MESSAGE
        SHR     BL,1            ,TEST FOR A FAILURE
        JNC     FNDBAD          ;THIS IS NOT IT
REPORT1:MOV     AH,9
        INT     21H             ;REPORT CONDITION
        OR      BL,BL           ,DONE?
        JNZ     FNDBAD          ,IF NOT, CONTINUE
        POP     DS              ;ELSE, GET OLD POINTERS
        POP     SI
        MOV     [SI],DI         ;RESTORE OLD VECTOR
        MOV     2[SI],ES
        INT     20H             ;EXIT

ALSAV   DB      0
SISAV   DW      0
DSSAV   DW      0

;       INPUT 60 INTERRUPT PROCESSOR

INT60.  MOV     CS:ALSAV,AL     ;SAVE AX
        MOV     AL,20H
        OUT     MASTER,AL       ;CLEAR 8259'S
        OUT     SLAVE,AL
        MOV     CS:SISAV,SI     ,SAVE SI,DS
```

```
        MOV     CS:DSSAV,DS
        POP     SI              ;GET USER'S IP
        POP     DS              ,AND HIS CS
        STI
        MOV     AL,-1[SI]       ;GET LAST INSTRUCTION
        PUSH    DS
        PUSH    SI
        MOV     DS,CS:DSSAV     ;RESTORE DS,SI
        MOV     SI,CS:SISAV
        CMP     AL,0ECH         :LAST INS. = IN AL,DX?
        JNZ     NOZHS
        CMP     DX,60H          ;PORT = 60H?
        JNZ     NOZHS
ZHSOK:  MOV     AL,55H          ;MARK TEST PASSED
        IRET
NOZHS.  MOV     AL,CS:ALSAV     ,RETURN ORIGINAL AX
        IRET

MSG1    DB      13,10,"ZHS Board OK$  "
MSG2    DB      13,10,"8259's slow$   "
MSG3    DB      13,10,"Port 60 is bad$"
MSG4    DB      13,10,"Video port bad$"
MSGSIZ  EQU     MSG2-MSG1

CODE    ENDS
        END     START
```

Source Code For BLINK.COM

```
        PAGE    ,132
        CONTROL ZPC CURSOR BLINKING
.       TO USE THIS PROGRAM, ENTER
;
;       BLINK OFF               TO TURN CURSOR BLINK OFF
;       BLINK                   TO TURN CURSOR BLINK ON
,
,       BY P. SWAYNE, HUG SOFTWARE ENGINEER 17-JUN-86

CODE    SEGMENT
        ASSUME  CS:CODE,DS:CODE,ES:CODE,SS:CODE
        ORG     5CH
DFCB    LABEL   BYTE            ,DEFAULT FCB LOCATION
        ORG     100H

START:  PUSH    DS
        XOR     AX,AX
        MOV     DS,AX           ;POINT TO PAGE 0
        MOV     SI,11H*4+2      ;INT 11 VECTOR SEGMENT
        MOV     ES,[SI]         ;GET ZPC SEGMENT
        POP     DS              ,GET THIS SEGMENT
        MOV     SI,OFFSET BLONC ,ASSUME BLINK ON
        MOV     BL,14           ,REGULAR BLINK TIMER
        MOV     AL,DFCB+1       ;GET ARGUMENT
        CMP     AL,'0'          ;BLINK OFF?
        JNZ     BLON            ;NO, ON
        MOV     SI,OFFSET BLOFFC;ELSE, GET BLINK OFF CODE
        MOV     BL,3            ;SHORT BLINK TIMER
BLON:   MOV     DI,2FEEH        ;BLINK CODE IS HERE
        MOV     CX,10           ;20 BYTES OF CODE
        CLD                     ,MOVE FORWARD
        CLI                     ,KILL INTERRUPTS
        REP     MOVSW           ;MOVE IN NEW CODE
        STI
        MOV     SI,2FEBH        ;BLINK TIMER IS HERE
        MOV     ES:[SI],BL      ,PUT NEW VALUE IN
        INT     20H             ;EXIT

;       CURSOR BLINK CODE (NORMAL ZPC CODE) AT 2FEEH

BLONC   DB      0C6H,6,5BH,1,0  ;MOV BYTE PTR [15B],0
        DB      80H,36H,59H,1,1 ;XOR BYTE PTR [159],1
        DB      0FBH            ;STI
        DB      0E8H,82H,6      ;CALL 367E
        DB      0FAH            ,CLI
        DB      0C6H,6,5BH,1,1  ;MOV BYTE PTR [15B],1

;       CURSOR STEADY CODE (REPLACES ABOVE CODE)

BLOFFC  DB      80H,3EH,59H,1,1 ;CMP BYTE PTR [159],1
        DB      74H,0DH         ;JZ 3002
        DB      0C6H,6,59H,1,1  ;MOV BYTE PTR [159],1
        DB      0FBH            ,STI
        DB      0E8H,80H,6      ,CALL 367E
        DB      0FAH            ;CLI
        DB      90H             ;NOP
        DB      90H             ;NOP
        DB      90H             ,NOP

CODE    ENDS
        END     START
```

*



It IS Polite To Point

Hardware Review Of The L-PC Lite-Pen

Jim Buszkiewicz
HUG Software Developer

On, off, on, on, off, on, off, on, deposit. That used to be the sequence of events (actually switch settings) one had to make to enter a single byte of data into his or her microcomputer. After a few hundred or so of those switch settings you could then use your keyboard to enter data because some sort of monitor or operating system was then loaded into the system. Micros have come a long way since those times and several new methods of entering data have become very popular. Some of these methods include the mouse, bit pads, touch-sensitive screens, and finally the lite-pen. While each of these data entry methods are excellent for certain applications, the lite-pen is suitable for just about any situation. All you do is point, and touch, and the software does the rest.

A lite-pen consists of a tubular barrel focused to a point much like a ball point pen. At the tip (point) of the barrel is a photo-sensitive device connected to other electronics within the barrel. There is also some method of triggering the pen, such as a push-button on the side of the barrel (activated with the finger), or a pressure sensitive switch at the tip of the pen (activated by pressing the pen against the CRT). Exiting the pen at the opposite end of the barrel is a small multi-conductor cable with some sort of connector at the opposite end. This cable is usually connected to the computer, specifically to the CRT controller chip. When the application software and lite-pen hardware are working correctly, the CRT controller can detect a 'coincidence' or 'hit' from an illuminated area on the screen while the trigger button is depressed. This is accomplished as each line is scanned. This 'coincidence' information is returned to the software usually as X - Y coordinates. Depending upon the quality of optics, and electronics in the pen, single pixel (the smallest addressable location on the screen) resolution can be obtained.

The L-PC Lite Pen, manufactured by the Lite-Pen Company of Los Angeles, California is just such a pen. Its non-glare stainless steel body measures just 7-1/2" in length, and 1/2" in diameter. It is a professional instrument capable of single pixel resolution. The back end of the pen barrel contains a standard modular type phone jack so that the pen can be used with a variety of different computers just by changing the cable. The cable itself is a lightweight retractile cord, and is fitted with standard telephone connectors at both ends. The tip of the pen contains a pressure sensitive type switch, although side barrel switch models are available.

The L-PC comes with a cable for the IBM PC which consists of a retractile cable with modular phone type connectors on each end. Also included, is an adaptor cable which consists of a multi-colored cable with a phone type jack on one end, and a 5-pin Molex female connector on the other. Hardware is also included for mounting the phone jack end of this cable to the back of the computer. In one of the pin holes of the Molex female connector is a polarizing pin. If this hardware is to be used with any of the Heath/Zenith PC computers, this plastic pin must be pulled out.

Although this hardware is for the IBM PC series of computers, it works equally well for the Heath/Zenith PC series of computers. These systems include the H/Z-150/160, H/Z-148/158, and H/Z-200. In all three cases, the lite-pen connector consists of a single 6-pin Molex circuit board mounted, male connector. On the H/Z-150/160 and H/Z-200 series computer, the lite-pen connector is on the video board towards the top, and is clearly labeled as such. In the H/Z-158, the lite pen connector is on the I/O board, and is also clearly labeled. Although the I/O board in the H/Z-148 has a place for a lite-pen connector, no connector has been installed. The reason for this may possibly be that, it could interfere with the installation of the ZA-141-1 expansion board. It may be possible to install the connector, however, and bend the pins outward, so they will not interfere. In any case, the female Molex connector should be plugged onto the male pins so

that pin #1 of the male pins goes into the female socket containing the brown wire. This will leave male pin #6 empty. Male pin #1 is clearly marked on some of the Heath/Zenith circuit boards, and in all cases, except for the H/Z-148, will always be the pin closest to the front of the computer.

The sample software disk included with the lite-pen contains many different samples of ways a lite-pen can be used. All of the programs are selectable from a lite-pen driven menu. Some of these programs include, text editing, order entry, directory programs, games, and graphics. These programs are written in BASICA and can be studied, should you want to write your own application programs.

The manual included with the L-PC lite-pen is very thorough, beginning with lite-pen basics, and ending with how to write your own lite-pen software. Also covered is the sample software which is included on disk.

During my evaluation of this lite-pen, I had the opportunity to talk with Mr. Bill Lewis, the president of the Lite-Pen Company, and was informed that the pen electronics were designed to be almost burn-out proof. I, of course, wanted to say "sure sure", but I'm glad I didn't. A foil problem around the lite-pen connector in my H-110 caused the lite-pen I had for evaluation to get hot enough to cook a small omlet. I figured that the excessive heat wasn't normal, and turned things off after about a minute or so (I was a little slower than normal that day). After discovering and correcting the problem, I didn't have much hope for the now supposedly defunct lite-pen. Boy, was I in for a surprise! It once again operated as if nothing had happened.

The L-PC lite-pen comes with a two-year conditional warranty, and normally sells for $179.95. This price includes the single IBM PC style cable. Additional cables are available for $25 each for other systems, such as the H/Z-100. HUG has made special arrangements with the Lite-Pen Company, to allow purchasers of the HUG CALC program (P/N 885-8043-37), to obtain this high quality lite-pen with PC cables AND H/Z-100 cable for only $89.97!! This is a savings of $114.98! For a description of the HUG CALC program, see the New Products section of the September issue of REMark magazine. For more information on the L-PC lite-pen, contact Lite-Pen Company, 12500 Beatrice Street, Los Angeles, CA 90066, or call (800) 634-1967. *

The Ultimate Technological Achievement

Jim Lytle
Heath Company

Adaptability. That's one of the characteristics which we as human beings share to some degree with all the living things on this earth. While adaptability is not unique to humans, certainly the degree to which we humans can adapt is unique. Scientists tell us that it is the degree to which we have been able to adapt to the changing environment that has enabled us to survive as a species as long as we have.

This same characteristic of adaptability is also responsible for mankind's sometimes spiritless reaction to the wonders that are taking place around us. We have become so adapted to technological changes that we fail to notice how quickly technological change is taking place and how dramatically these changes are affecting our lives. Let's see if we can put some things in a new perspective and reawaken our sense of awe.

Consider this: The entire recorded history of mankind goes back about 60 thousand years. Let's see if we can look at the progress we've made in those years in a new way. For example, if we assume an average lifetime of 70 years, man's total recorded history spans about 800 average lifetimes.

For the first 600 of these lifetimes, man lived in caves. Virtually every waking moment was spent in the never ending quest for food. In other words, the basic requirements of survival consumed all of his time, energy and whatever creative ability he may have had.

It was only during the last 70 lifetimes that a written language was developed enabling knowledge to be recorded. However, the process was a tedious one as facts were painstakingly recorded using handwritten symbols. Now, a very few privileged individuals who knew how to read these symbols could spend some of their time learning about themselves and their world. They could benefit from knowledge gained by others. This privileged few within each generation no longer had to relearn the often bitter lessons learned by those who had gone before.

The printing press came on the scene about eight lifetimes ago. This enabled large masses to share the knowledge of those that had gone before. People now learned not only from their own experiences, but also from the experiences of others.

Not until the last two lifetimes did anyone anywhere use an electric motor. No longer bounded by what he could do with his own muscle power or that of domesticated animals, man underwent an industrial revolution. His untiring machines produced goods in an abundance undreamed of in the previous lifetimes.

But here's the really thought provoking part. It has only been during this last lifetime, the past 70 odd years when we and our parents were on the scene, that virtually everything we use in our daily lives has come into being!

Think about that as you look around you. How many of the things you can see existed for more than a lifetime? The clothes you're wearing are undoubtedly made of fibers developed within this period. Is there a television set in the room? An electric light? Furniture with durable plastic finishes resembling woodgrain? How about the watch on your wrist? The calculator in your pocket or on your desk? The magazine you are reading uses inks, photographic processes and production techniques which didn't exist in previous lifetimes. You'll probably have difficulty finding a single item around you of any consequence which existed in a previous lifetime.

Think about that. Virtually everything we use today in our daily lives has been developed during this last of the 800 lifetimes of man's recorded history! Can anyone even begin to imagine what changes the next lifetime will bring?

To Continue Putting Things In Perspective . . .

On December 18, 1903 the Wright brothers launched the first successful powered aircraft from a location near their bicycle shop in Kitty Hawk, North Carolina. The machine was built entirely by these two men with the help of a single assistant. The distance traveled by the machine was measured in feet and the time it remained aloft was measured in seconds. The only known report of the event on the day it happened was a telegram Wilbur sent to his father telling him of their success.

Sixty six years later (less than one lifetime), more than 400 thousand people working together made it possible for Neil Armstrong to travel a quarter million miles from earth and walk on the surface of the Moon. Six hundred million people watched the event on television as it was happening!

Continuing Further . . .

Back in 1948, two enterprising gentlemen working at Bell Labs demonstrated the first transistor. Within ten years, this new technology brought us complete radio receivers which could be held in the palm of ones hand. Today, less than 40 years since the introduction of the transistor, complete computers utilizing more than 300,000 transistors are routinely produced on silicon chips measuring less than 1/4 inch on a side!

And Continuing Further . . .

Consider the following hypothetical advertisement in the CLASSIFIED section of your local newspaper:

> **HELP WANTED**
>
> Reliable worker to perform routine, dangerous, demeaning and monotonous tasks in an automotive assembly plant. The successful candidate must be willing to work 24 hour shifts in an atmosphere of noxious fumes, deafening noise levels and ambient temperatures of 150 degrees farenheit or more. A demonstrated ability to maneuver 80 pound red-hot castings with dexterity and precision is essential.

Such an ad, of course, does not describe a human being. This is a job for a robot. The development and refinement of the integrated circuit and computer technology has brought about the beginning of the second industrial revolution. Today, some products go from the imagination of the designer sitting at a computer console to finished goods ready for shipment at the loading dock without any human intervention in between. In the totally automated factory of the very near future, computer aided design (CAD) and computer integrated manufacturing (CIM) will make it possible to produce nearly all our goods in this same manner.

With wonders such as these taking place all around us every day, let me again ask the question "Can anyone begin to imagine the wonders the next lifetime will bring?" Is there anything left to discover? Are there any improvements yet to be made?

To be sure there are. Let me illustrate.

The other day I was sitting at my desk in my office and the telephone rang. I picked up the receiver and identified myself. The voice at the other end said "Hello Jim, how are you doing?" Immediately I recognized the voice as that of my brother-in-law. He had never called me at the office before, I had not seen or heard from him in over a year, I had no reason to expect a call from him. Yet I was able to instantly distinguish that certain uniqueness in his voice which differentiated it from the thousands of others I know.

This is not a skill that I alone possess. Anyone could have done the same thing. However, no computer could have done so. We're very proud of the fact that electronic voice recognition capability of sorts does exist. Even the best of these systems will recognize only the literal meaning of the words it recognizes. Meanings which are based on context completely escape the computer just as do the subtleties which enable us to identify the speaker by the unique characteristics of his or her voice. So computers have a long way to go in the area of voice recognition.

How about vision? There's another area where we have much to learn. I recently heard of an experiment performed at Carnegie Mellon University whereby a robot controlled vehicle was able to traverse a furniture filled room. It avoided collisions with objects in the room by using an electronic vision system. It took the vehicle seven hours to negotiate the 15 feet from one end of the room to the other!

What took the vision guided vehicle so long to move such a short distance was this — every time the vehicle moved, it's perspective changed. So, the computer had to complete an enormous number of calculations to determine that it was still looking at exactly the same scene from a slightly different point of view.

What we sometimes fail to realize is that most of what we call vision is done with our brains rather than our eyes. Hold a coin directly in front of you with one of the sides perpendicular to your line of sight. What do you see? A circle, right? Now move the coin away from you. It gets smaller but it's still a circle. Now turn the coin slightly. It's no longer a circle, but an ellipse.

So we see how a simple shape, such as a circle, can appear to be an infinite number of differing sized circles, depending on its distance from the observer, or an infinite variety of ellipses depending on the point of view and the distance. We have much to learn in the way of vision also, particularly the part where an image must be interpreted. We do this all the time without even being consciously aware that we are doing so.

How about raw muscle power? Surely our technology has exceeded our puny human capabilities there? Right? Wrong! First of all, how many machines do you know of that can function for about 70 consecutive years with virtually no maintenance? More specifically, consider the muscle in your arm which enables you to raise your forearm — the bicept. An average human being can easily lift a 25 pound weight placed in the palm of the hand. Recalling your high school physics, the human forearm is a classic example of a third class lever. The fulcrum is at the elbow with the muscle attached about an inch beyond the fulcrum. The 25 pound load in the hand would be about 15 inches from the fulcrum. This gives us a mechanical disadvantage of about 15 to 1. In other words, the bicept must pull with a force of 375 pounds (15 × 25) to heft the 25 pound weight in the hand. Think about that. The bicept which weighs only a few ounces easily pulls with a force of nearly 400 pounds in this situation. There is no electromechanical equivalent in our world of technology.

We can be justifiably proud of the achievements we've accomplished over a relatively short period of time. We can look forward to indescribable wonders within our own remaining lifetimes. The human machine, however, still provides us with many examples of how far we have to go and how much we have to learn. I'm sure you can think of many more examples than the few we've presented here. Clearly, the human machine — man — is still the ultimate technological achievement.

Multifunction Resident Desktop Utilities

A Review Of Genie, The Informer And Whiz For The H/Z-100

Herb Friedman
Western Regional HUG
Pomona, CA

Do you really NEED a "desktop utility"? What do they offer that you don't already have? The large number of users of Borland International's SIDEKICK seems to indicate a need for this type of utility.

Of course, Barry Watzman's PERKS is available for the H/Z-100 and will be reviewed elsewhere in REMark. This is a review of three similar products specifically designed for use on H/Z-100 computers. They are (in alphabetical order): Genie ($49.95 from Advanced Software Technologies), The Informer ($69.95 from Sunflower Software), and Whiz ($49.95 from Software Wizardry).

These programs have more similarities than differences, and it is those differences that will help you decide which is the one for you . . . assuming, of course, you've decided to get one. When making that decision, bear in mind that many current users have stated they can't imagine how they ever got along without their desktop utilities!

All of these programs are partly or wholly resident in memory and may be called into action by a short sequence of keystrokes. They either appear as "windows" on the current screen display, or they temporarily displace what's on the screen. When you are done using the utility, the windows close and your original screen display is returned intact.

This means that these utilities may be used while you are running another program. They simply interrupt the current program, allow you to perform a function, and then return to the program. For example, you might be writing a letter using your favorite word processor when the telephone rings. The caller wishes to leave a message. Simply invoke the "notepad" function of your "desktop utility" and enter and save the message. You may then resume your letter writing exactly where you left off.

Some other useful functions provided by these programs are alarm clocks, appointment calendars, programmers calculators, scientific calculators, ASCII tables, address books, keymappers, auto-dialers, cut-and-paste, screen saver and MS-DOS functions.

Since all of these programs occupy a fair-sized chunk of memory, it is advisable to have a minimum of 192k of system memory. These programs generally require MS-DOS version 2.13 or higher and a monitor ROM version 2.5 or higher.

Installation

Genie is supplied as individual modules which may be loaded as needed. A single BASE module must be loaded first, and each of the functions comes in its own module. Genie offers many options. The base module may be loaded with or without a keymapping facility. The latter is useful if one is running Lotus 1-2-3, which attempts to read directly from the keyboard. With that module, Genie may be invoked with the <HELP>, <BREAK>, or <DELETE> key. With the keymapper in place, Genie comes up by hitting BOTH <SHIFT> KEYS simultaneously. Genie uses multiple windows, which can be opened sequentially, and are movable.

The Informer is supplied with an installation module which sets up various data files on your disk. In addition, a configuration program is provided which permits many options to be specified by the user. A file may be set up for the Calendar module to insert regularly recurring events, such as meetings, birthdays, anniversaries, etc. There is one main module that is loaded (preferably at boot-up by an AUTOEXEC.BAT file), and you are ready to go. The Informer is invoked by SHIFT-<BREAK> and exited by <BREAK>. It does not use windows, but rather the entire screen.

Whiz is easily invoked by typing WHIZ, however parameters may be added to change the calendar file name and the amount of allocated screen save space. Whiz uses windows, which are invoked by <BREAK>-n, where n is a digit from 0 to 4. The windows may be moved anywhere on the screen and have HELP messages in the borders which are toggled by the <HELP> key. The <ESC> key closes the windows.

FUNCTIONS

Alarm Clock

Genie supports eight alarms which may be set for any time on any day. When they go off, they may be set to either beep and open a window on the screen, or they may send a string of characters to the keyboard input buffer.

The Informer has two built-in alarm clocks that can be set independently. When an alarm goes off, a continuous beep is emitted for about a minute, followed by intermittent beeps for several minutes. They are difficult to ignore.

Whiz has a single alarm that beeps ten times and then stops.

ASCII Table

All three programs support ASCII conversion charts.

Notepad

Genie has a notepad (or scratchpad) for creating, editing and saving text. It can edit files of any size, merge files together and move blocks of text. It can support a buffer as large as 56K. It also supports a cut-and-paste function, which can transfer text from the screen to its buffer and then later, back to the screen.

The Informer supports personalized notepads. It saves them in files with your initials in the file name. It allows screen editing with the editing and cursor-movement keys. The buffer size is considerably smaller than Genie's.

Whiz also has a notepad facility, with limited editing capabilities. It supports copying data from the screen into the notepad. The buffer holds about 40 lines of text with 80 characters per line. This makes editing of a large file rather awkward.

Calendar

Genie provides a yearly calendar for as many years as you wish. Genie can search the calendar for a specific appointment and can print out a calendar chart.

The Informer's calendar has a unique feature. It will automatically insert recurrent events in the appropriate month. Regular monthly meetings, birthdays and other occasions will be hard to forget with this calendar. A limited search function is provided to search for a string in the current month's data file.

Whiz supports a calendar which is easy to scan. It uses the function keys to control some of the calendar functions. It lacks some of the other features just described.

Calculator

Genie provides several calculator modules. There is an integer calculator for doing simple arithmetic, a floating point calculator for greater precision, and a floating point calculator with 8087 support (if you have that co-processor on board). Each of these comes in a fancy window or a small window version, since the latter conserves lots of memory space.

The Informer provides an accurate calculator with floating point precision. (Internal accuracy of 9 decimal places). It supports base conversions in decimal, binary, octal and hexadecimal. It may be configured to store eight different constants. Entries must be given in RPN (Reverse Polish Notation). If you are not comfortable with this type of data entry, you may not wish to use this function.

Whiz also supports a full floating point algebraic expression calculator. It does not require RPN entry. It supports many arithmetic functions including SIN, COS, TAN, LOG10, Natural LOG, etc.

Address Book

Genie calls its address book "Rolodex". It can store an infinite number of names, addresses and phone numbers. There is a built-in formatting facility which allows one to change the formatted output, rather than having to reconfigure the entire program. The Genie Rolodex can type a name and address directly into a word processor, or output a phone number to a modem program to be dialed.

The Informer supports the "Address Book", a system that can store up to 256 'index cards', containing name, address, phone number and other data. They are alphabetized, easy to scan through and easy to change. This stored information can be used to print address labels, or to automatically dial a phone number.

ADDITIONAL FUNCTIONS

Genie

The Keymapper allows you to redefine most of the H/Z-100 keys. For example, you could define a key as follows: "DIR B:*.TXT <RETURN>". Each time that key was pressed, that command would be placed on the screen and executed. A file of key changes could be saved for future use.

Genie's Cut-and-Paste module is quite remarkable. Using this facility one can move text from one program to another, move text from a screen into the notepad, or save screen text in a buffer and add it to another screen later.

Genie also supports MSDOS functions within the program. It supports MSDOS functions such as DIR, TIME, DATE, ERASE, RENAME, COPY, CD, MD and RD. Besides this, it can search a disk directory tree structure for a file name and TYPE out a file of any length. The DEL command queries the user prior to file deletion.

Genie has a Typewriter module which lets you send a line typed on the screen directly to your printer. This line may contain printer control or escape sequences. Genie also supports a screen saver to prolong the life of your CRT's phosphors.

The Informer

The Informer supports automatic telephone dialing if you have an auto-dial modem. It can dial the phone number on an "Address Book" index card. This program also includes an on-screen clock that is similar to Pat Swayne's SCRNCLK module, and a screen-saver that will blank your screen if there is no activity within a stated time period. The Informer also supports some MSDOS functions such as DIR, TIME, DATE, ERASE, RENAME, COPY, CD, MD and RD.

The Informer also provides the H/Z-100 with some additional command line editing keys. The keypad left and right arrow keys will move the cursor on the command line. The up-arrow does the same thing as the <F3> key, and the <HOME> key moves the cursor to the beginning of the command line. The <D CHAR/I CHAR> key will, respectively, delete a character or add a space to the command line.

Whiz

Whiz has the fewest functions, but supports the most useful ones. Whiz provides a very useful function which resets the video display in case the display goes haywire. I have seen some programs result in the screen becoming "wrapped" around half-way and the cursor out of sight. Using this reset function is more convenient than rebooting.

Documentation And Support

All three programs provide sufficient documentation to get them installed and running. One must actually use the program for a while in order to learn how to get the most out of it, and to discover its limitations. Each program comes with a phone number to call for support.

Conclusions

These three programs have each taken slightly different approaches to becoming a multifunction resident desktop utility. Choosing among them is not easy. The choice ultimately depends on which functions are more important to you, and which of the three implements those functions better. Heath/Zenith users are indeed fortunate to be able to choose among three pieces of quality software. ✱

Taking Graphic Drawings Across A Number Of Microcomputers And A Minicomputer

Eric L. Pang
1530 Nehoa Street
Honolulu, HI 96822

Introduction

At my previous job, I was involved with developing Computer Aided Instruction (CAI) material that would be compatible with a number of different microcomputers and with our minicomputer. One of my objectives was to use line drawings of insects in various educational software and in insect identification programs. (The advantage of using line drawings over photographs is that you can direct the attention of the viewer towards specific features of the drawing. A photograph, on the other hand, is usually crowded with extraneous material that can distract the viewer.)

As you can guess, the university is full of different types of computers. In our department alone, we had, Zenith 100s, IBM PCs, Corona PCs, TI Professionals, Radio Shack 2000s, DEC PC350s and of course, our DEC VAX 11/750. How could graphic drawings be transported across this wide range of hardware running different languages and operating systems?

The displayed image of the drawing could not be stored and recalled as each computer had different display sizes and aspect ratios (640×200, 640×225, etc.), graphics modes (e.g. medium, high, etc.) and the video memory for each machine resides at different locations. The graphics commands used to draw the image could not be saved either (e.g. "LINE (10,10)-(24,120)"), as each computer used different (however slight) commands and this would require translation for each machine that the drawing would be displayed on.

This article will discuss a number of factors that were considered in developing the methods to display graphic drawings over a wide range of computers. The major factors considered were the language to use, the type of data needed to create the drawings, the format and contents of the picture data file and the actual drawing program itself.

Language Considerations

Despite the availability of more structured languages, BASIC is ubiquitous. Remember, I wanted to be able to display my drawings on the widest range of computers possible, and BASIC runs on all of the target computers, and with the exception of the VAX, all of the BASICs have built-in graphics statements.

The BASICs available for the MS-DOS machines, Zenith, IBM, TI and Corona are very similar. (Time did not permit me to work with the Radio Shack.) The only difference is that the IBM and Corona have different graphics "modes" that have to be set and lack the colors available on the Zenith and TI (in high resolution mode).

Although a popular PCDOS Pascal has built-in graphics statements, and libraries of graphics subroutines are available for C, these procedures are available only for the IBM PC. I have yet to find a comprehensive package that would run on all of the different machines I wanted to target.

Data File Structure And Contents For The Graphics Picture

The pictures could be stored in a number of different ways. The x-y coordinates used to draw the picture could be saved on disk. The commands (LINE, POINT, MOVE, etc.) used to draw the picture could be read from a disk file. The video memory containing the picture could be saved and written to a disk file.

After a little thought, I finally decided to save the drawing as an ASCII file which contained a simple "command language" for describing a graphics operation (such as lines, dots, circles, boxes, etc.) along with the x-y coordinates needed to draw the picture. The necessary file format to hold the information required by these graphics operations (center of circle, radius, color, etc.) was also developed.

The picture file consists of three types of "records", 1) a line containing a graphics command operation, enclosed by double quotes, followed by the data required by the operation, 2) lines containing x-y coordinates, and 3) a line containing text enclosed by double quotes. Except for the line containing text, fields are of fixed length. The fields are also separated by commas. This will allow the file to be read quite easily by a BASIC, C, Pascal or FORTRAN program without the need of special data parsing and data type conversion routines.

The graphics command string line consists of a graphics operation followed by ten parameters. Valid operations are: 1) "LINE" which is used to draw a series of lines for a sequence of x-y coordinates, 2) "POIN" for drawing points or dots for a sequence of x-y coordinates, 3) "CIRC" which can be used to draw circles and arcs (but not ovals), 4) "VECT" which is used for drawing a single vector or line between two given points, 5) "BOX" for drawing a box or rectangle (unfilled), 6) "FILL" which is used to fill a solid object with a particular color, and 7) "TEXT" which can be used to place text material anywhere in the drawing.

The ten parameters following each command are: 1) the color used to draw the line, point or text, 2) the x coordinate for the circle's center, one end of a vector or box, placement of the text, or a point in the object to be filled, 3) the y coordinate for the circle's center, one end of a vector, box, text, or object to be filled, 4) an x coordinate falling on the circle's circumference to be used to calculate its radius, or the other end of the vector or box 5) the y coordinate falling on the circle's circumference and used to calculate its radius, or the other end of the vector or box 6) the color to draw the circle, box or vector, 7) the radian to start the circle or arc (for a circle, this is 0.00), 8) the radian to stop the drawing of the circle or arc (for a circle, this is 6.28), 9) the aspect ratio of the circle (so that the circle will come out circular), and 10) the color of the border where filling is to cease.

Depending on the graphics command, some of the succeeding parameters may not be applicable. For example, the aspect ratio is not used for drawing a vector. Values for these unused parameters will still be present (i.e. occupy space) on the command line, however. If the command string is "LINE" or "POIN", x and y data are expected in the succeeding lines. The end of the x-y data is signaled (-1,-1). If the command "TEXT" is encountered, the next line will contain a string of text enclosed by double quotes.

An example of the data file follows. Note that the x-y data refers to coordinates generated by the digitizer with origin (0,0) at the lower left corner, and that the data will have to be transformed and rescaled for proper display on the monitor.

```
"BOX ", 7 ,  73 ,  81 , 761 , 793 , 7 , 0.00 , 6.28 ,
     0.00 , 7
"LINE", 1 ,  73 ,  81 , 761 , 793 , 1 , 0.00 , 6.28 ,
     0.00 , 7
  110 ,  540
  163 ,  546
  340 ,  548
  302 ,  210
  171 ,  315
  232 ,  600
  249 ,  660
   -1 ,   -1
"CIRC", 1 , 559 , 632 , 518 , 542 , 4 , 0.00 , 6.28 ,
     0.48 , 7
"TEXT", 1 , 400 , 162 , 518 , 542 , 4 , 0.00 , 6.28 ,
     0.48 , 7
"this is a test"
```

The first "command" indicates that a box is to be drawn with corners at (73,81) and (761,793) using color number 7. The next command uses color number 1 and draws lines connecting the succeeding coordinates. A circle centered at (559,632) is drawn next. A point on the circumference, (518,542), is used to calculate the radius. The color 4 is used as the drawing color. Finally, text material is placed at approximately (400,162) of the screen. (The coordinates are transformed to the closest row-column position.)

Displaying The Drawing — The Drawing Program

Now that we know the language to be used, the type of data needed and the format of the data, writing the drawing program is straight forward. The drawing program is actually quite simple, especially when the graphics statements are already built into BASIC.

The program first sets the various scaling parameters (generally, variables beginning with "Z" are real numbers, all others are integers). For the Z-100,

```
ZX = 1/2.15 : ZY = 1/4.75
ZYSIZE = 225 : ZXSIZE = 640 : ZASFAC = 1
ZXDOT = ZXSIZE/80 : ZYDAOT = ZYSIZE/25
```

where ZX is the x scaling factor between the Z-100 and the digitizer with ZY being the corresponding y scaling factor. ZYSIZE is used to transform the y origin (0,0) from the bottom left corner (which the digitizer uses) to the top left corner of the screen. ZASFAC is the "aspect" ratio used to ensure that circles come out circular. For comparison, the corresponding values for the IBM PC are shown below,

```
ZX = 1/2.28 : ZY = 1/5.28
ZYSIZE = 200   ZXSIZE = 640   ZASFAC = 0.86
ZXDOT = ZXSIZE/80   ZYDAOT = ZYSIZE/25
```

The code used to define the color palette can be found in the program.

The disk file containing the graphics commands and coordinates is then opened. The screen mode is set to high resolution if necessary. A loop is entered that will read each line of the picture file. The first line should contain a command and is read by the statement,

```
INPUT #1, CMMD$,ATR,CX,CY,R1,R2,CULR,ZSTRT,ZFIN,ZASP,BRDR
```

The colors are remapped (i.e. "color 2" for the Zenith corresponds to a TI's "color 4") and the color attributes set. The x-y coordinates for the command are scaled to the dimensions of the screen (these are defined as functions in the actual program) as they may be needed later,

```
CX = CX * ZX : CY = ZYSIZE - CY * ZY
R1 = R1 * ZX : R2 = ZYSIZE - R2 * ZY
```

(these statements are defined as functions in the program).

The command string is compared with the available commands and the appropriate subroutine is then called if a match occurs.

```
IF CMMD$ = "CIRC" THEN GOSUB 1580 : GOTO end-of-loop
IF CMMD$ = "VECT" THEN GOSUB 1700 : GOTO end-of-loop
IF CMMD$ = "BOX " THEN GOSUB 1650  GOTO end-of-loop
IF CMMD$ = "FILL" THEN GOSUB 1750 : GOTO end-of-loop
IF CMMD$ = "TEXT" THEN GOSUB 1850 : GOTO end-of-loop
IF CMMD$ = "LINE" THEN LMODE = 1 ELSE LMODE = 2
```

The appropriate machine specific graphics statements are executed to perform the tasks that are called. The subroutines for MSDOS-type BASICS are shown below.

To draw a circle, the radius of the circle is calculated by taking the x-y coordinates falling on the circumference of the circle and finding the distance between these points and the center of the circle. The aspect ratio is calculated and the circle drawn.

```
1580 YY = (ZYSIZE-CY)/ZY : Y2 = (ZYSIZE-R2)/ZY
1590 Y = ZX*ABS(YY-Y2) : X = ABS(CX-R1) :
     ZRAD = SQR(X*X+Y*Y)
1600 CIRCLE (CX,CY),ZRAD,CULR,ZSTRT,ZFIN,ZASP*ZASFAC
1610 RETURN
```

Drawing a box is similar to drawing a vector,

```
1650 LINE (CX,CY) - (R1,R2), CULR, B . RETURN
```

Drawing a vector is simply calling the LINE command,

```
1700 LINE (CX,CY) - (R1,R2), CULR : RETURN
```

To fill an object, a point within the outline is defined, the solid object is chosen and the PAINT command called.

```
1750 PAINT (CX,CY), CULR, BRDR . RETURN
```

To draw a series of lines or dots, the next line in the data file following the command line is read for the initial x-y coordinates. The graphics "cursor" is "moved" to this point. Subsequent INPUT calls are used to retrieve the remaining x-y coordinates that make up the figure. Encountering (-1,-1) signifies the end of the line or point data.

```
1360   INPUT #1, X,Y
1370   X2 = X * ZX : Y2 = ZYSIZE - Y * ZY
1380   PSET (X2,Y2)
1400       INPUT #1, X,Y
1410       X2 = X * ZX : Y2 = ZYSIZE - Y * ZY
1420   IF X = -1 THEN end-of-loop ELSE GOSUB 1800 :
       GOTO 1400


1800   IF LMODE=1 THEN LINE -(X2,Y2),ATR
       ELSE PSET (X2,Y2),ATR
1820   RETURN
```

If the command is TEXT, then the next line following the graphics command line is read to retrieve the text material,

```
1850 INPUT #1, TXT$
1860 COL = FIX(CX/ZXDOT+.5) + 1   IF COL > 80
     THEN COL = 80
1870 ROW = FIX(CY/ZYDOT+.5) + 1 : IF ROW > 25
     THEN ROW = 25
1880 COLOR ATR
1890 LOCATE ROW, COL   PRINT TXT$
1900 RETURN
```

After the drawing command is processed, the next drawing command is read. This is repeated until the end of the picture file.

Because of the modularity built into this program, the transport of this program to another machine, or language, would be quite simple. The "main body" of the program will remain essentially the same. The scaling factors will have to be changed to reflect the display for the new machine. Primarily, the drawing subroutines will have to be changed to reflect the commands available to that particular language.

For example, to draw a circle using VAX-11 RGL (a graphics library written for the DEC VAX minicomputer and VT125 graphics terminal), the following commands could be used,

```
YY = (ZYSIZE-CY)/ZY   Y2 = (ZYSIZE-R2)/ZY
Y = ZX * ABS(YY-Y2) . X = ABS(CX-R1)
MOVE (CX,CY)
CIRCLE (X,Y)
```

while drawing a box is simply,

```
BOX (CX,CY,R1,R2)
```

The only major problem one will have in moving this program to another machine is in the use of the paint or fill command and the use of too many colors. For example, VAX-11 RGL will have a very difficult time filling irregularly-shaped objects. It may, in fact, be close to impossible unless the object is broken into many regularly-shaped polygons. Also, the VT125 graphics terminal has a limitation on the number of different colors that can be displayed at any one time.

Generating The Data File

The picture files are generated by a digitizer and a simple driver program I wrote called ZTABLET. It is a ZBASIC program which essentially combines the drawing program presented here and with a terminal program, along with a number of other support subroutines.

The terminal program, which can be found in the ZBASIC manual, reads the x-y data coming in on the serial port from the digitizer. The drawing part of the program takes the data and displays it as an image on the screen. Other support subroutines are used to select the drawing command, save the data, change drawing colors, etc.

Further details regarding this program can be obtained from the author.

Discussion

The only disadvantage of using this method to transport graphic pictures to other machines is that each section of the drawing will have to be created a segment at a time. The time to complete a drawing would take considerably more time than if the picture were displayed directly from video memory. Another problem associated with time spent in creating the final drawing is the time required to read the data file for the picture, which could reach more than 100K bytes in size. In many interactive applications, the time required to read this file and display the picture would be unacceptable. The user should not be expected to wait half a minute or so for a picture to be displayed.

Despite of these shortcomings, the methods presented here should go a long way in transporting graphic drawings over several types of entirely different classes of computers. Once the actual drawings are displayed on the screen, other routines can be written to save video memory to disk. The application program could then read this disk file and load the picture directly into video memory.

Listing

```
1000 REM  DRAW - Program to draw contents of picture
     file created by ZTABLET
1010 REM          Eric Pang, 1530 Nehoa St.,
                  Honolulu, HI 96822-2008
1020 REM
1030   CLS : KEY OFF : CLOSE
1040   DEFINT A-Y
1050   DIM C(7)
1058 REM
1060 REM define the functions we'll be using
1062 REM
1070   DEF FNXLATE(XPT) = XPT * ZX
1080   DEF FNYLATE(YPT) = ZYSIZE - YPT*ZY
1090   DEF FNXMOD(I,J) = I - (INT(I/J)*J)
1100 REM
1110 REM Set TT to type of machine desired
         (needed to scale figure correctly)
1120 REM 1-H/Z100  2-TI Prof  3-Corona 4-        5-
         6-IBM PC
1130 REM then go set scaling parameters
```

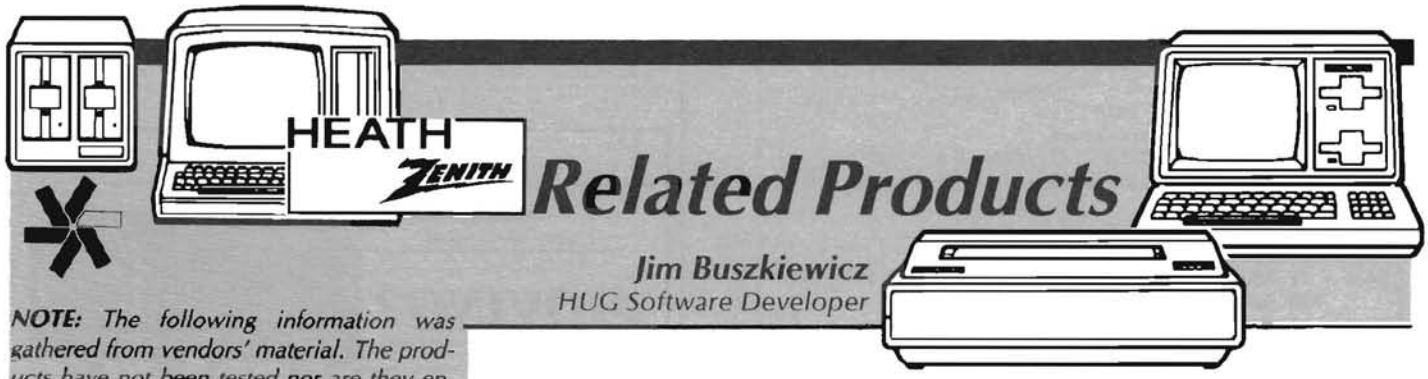
```
1132 REM
1140   TT = 1
1150   ON TT GOSUB 2110, 2170, 2230, 2300, 2300, 2300
1160 REM
1162 REM            ---  begin main program  ---
1164 REM
1170   GOSUB 1940          'REM clear screen, get file
1180   GOSUB 2020           REM open picture file
1190 REM
1200   WHILE NOT EOF (1)  :REM big loop to draw picture
1210     Y$ = INKEY$ : IF Y$ <> "" THEN 1490
     :REM in case one wants to quit
1220     INPUT #1, CMMD$, T1, CX, CY, R1, R2, T2, ZSTRT,
         ZFIN, ZASP, T3
1240     CULR = C(T2) : BRDR = C(T3)
     :REM set proper colors
1250     IF FNXMOD(TT, 3) = 0 THEN GOSUB 2390
     :REM provision for IBM
1260     IF FNXMOD(TT, 3) = 0 THEN ATR = 1 : CULR = 1 :
         BRDR = 1
1270     CX = FNXLATE(CX) ' CY = FNYLATE(CY)
     .REM translate to screen coord
1280     R1 = FNXLATE(R1) : R2 = FNYLATE(R2)
1290     IF CMMD$="CIRC" THEN GOSUB 1580 : GOTO 1430
1300     IF CMMD$="VECT" THEN GOSUB 1700   GOTO 1430
1310     IF CMMD$="BOX " THEN GOSUB 1650   GOTO 1430
1320     IF CMMD$="FILL" THEN GOSUB 1750 . GOTO 1430
1330     IF CMMD$="TEXT" THEN GOSUB 1850 : GOTO 1430
1340     IF CMMD$="LINE" THEN LMODE = 1 ELSE LMODE = 2
1350       ATR = C(T1) ' IF FNXMOD(TT, 3) = 0
           THEN GOSUB 2390
1360       INPUT #1, X,Y
1370       X2 = FNXLATE(X) : Y2 = FNYLATE(Y)
1380       PSET(X2, Y2), ATR
     'REM MOVE to 1st pnt in series
1390       Y$ = INKEY$ . IF Y$ <> "" THEN 1490
     .REM another chance to quit
1400         INPUT #1,X,Y
     :REM get remaining points
1410         X2 = FNXLATE(X)   Y2 = FNYLATE(Y)
1420       IF X = -1 THEN 1430 ELSE GOSUB 1800 .
           GOTO 1390
1430   WEND        :REM go back for more data
1440 REM
1450   CLOSE       :REM normal end of program
1460   LOCATE 25,1 : PRINT " ";
     'REM let 'em look for a while
1470   Y$ = INKEY$ . IF Y$ = "" THEN 1470 ELSE 1500
1480 REM
1490   CLOSE : CLS         :REM user quits, clean up
1500   LOCATE 23, 1        'REM move cursor up a bit
1510   END                  REM end of program
1520 REM ************************************************
1530 REM                     subroutines
1540 REM ************************************************
1550 REM
1560 REM - draw a circle or arc of color culr
1570 REM
1580   YY = FNYLATE(CY) : Y2 = FNYLATE(R2)
1590   Y = ZX*ABS(YY-Y2) : X = ABS(CX-R1)
       ZRAD = SQR(X*X+Y*Y)
1600   CIRCLE (CX,CY), ZRAD, CULR, ZSTRT, ZFIN, ZASP*ZASFAC
1610   RETURN
1620 REM
1630 REM - draw box, one end at (cx,cy) and other at
     (r1,r2) w/ color culr
1640 REM
1650   LINE (CX,CY) - (R1,R2), CULR, B
1660   RETURN
1670 REM
1680 REM - draw vector, one end at (cx,cy) and other at
     (r1,r2) w/ color culr
1690 REM
1700   LINE (CX,CY) - (R1,R2), CULR
1710   RETURN
1720 REM
1730 REM - fill solid object w/ color culr to border of
     color brdr
1740 REM
1750   PAINT (CX,CY), CULR, BRDR
1760   RETURN
1770 REM
1780 REM - draw series of lines or points, depending on
     LMODE
1790 REN
1800   IF LMODE = 1 THEN LINE -(X2,Y2),ATR
       ELSE PSET (X2,Y2),ATR
1810   RETURN
1820 REM
1830 REM - write out text material to screen with color atr
1840 REM
1850   INPUT #1, TXT$
1860   COL = FIX(CX/ZXDOT+.5) + 1    . IF COL > 80
       THEN COL = 80
1870   ROW = FIX(CY/ZYDOT+.5) + 1     IF ROW > 25
       THEN ROW = 25
1880   COLOR ATR
1890   LOCATE ROW, COL   PRINT TXT$,
1900   RETURN
1910 REM
1920 REM - display program title and get name of
     picture file
1930 REM
1940   CLS
1950   PRINT : PRINT
"ZDRAW - a program to display files created by Ztablet "
1960   PRINT
1970   INPUT "     Enter filename  ", WRKNAME$
1980   RETURN
1990 REM
2000 REM - open picture file and print a "_" to let
     user know still drawing
2010 REM
2020   OPEN "I", #1, WRKNAME$
2030   CLS
2040   IF FNXMOD(TT,3) = 0 THEN SCREEN TS
     'REM graphics mode if needed
2050   LOCATE 25,1   PRINT "_";
2060   RETURN
2070 REM
2080 REM - define scaling and transform parameters for
     a number of pc's
2090 REM
2100 REM Z100
2110       ZX = 1!/2.15    ZY = 1!/4.75
2120       ZYSIZE = 225! . ZXSIZE = 640! : ZASFAC = 1!
2130       ZXDOT = ZXSIZE/80! . ZYDOT = ZYSIZE/25!
2140       C(0)=0: C(1)=1: C(2)=2: C(3)=3: C(4)=4'
           C(5)=5' C(6)=6: C(7)=7
2150       RETURN
2160 REM TI Professional
2170       ZX = 1!/2.52  ' ZY = 1!/4.45
2180       ZYSIZE = 299! : ZXSIZE = 719! : ZASFAC = 1.1471
2190       ZXDOT = ZXSIZE/80!   ZYDOT = ZYSIZE/25!
2200       C(0)=0  C(1)=1  C(2)=4. C(3)=5: C(4)=2' C(5)=3
           C(6)=6: C(7)=7
2210       RETURN
2220 REM Corona PC
2230       TS = 105        :REM super hi-res mode
2240       ZX = 1!/2.27    ZY = 1!/3.17
2250       ZYSIZE = 325! : ZXSIZE = 640! : ZASFAC = 1.35
2260       ZXDOT = ZXSIZE/80! ' ZYDOT = ZYSIZE/25!
2270       C(0)=0' C(1)=4: C(2)=1' C(3)=4' C(4)=2  C(5)=5
           C(6)=3: C(7)=6
2280       RETURN
2290 REM IBM PC
2300       TS = 2          'REM 1=medium res  2=high res
2310       ZX = 1!/4.56 * TS : ZY = 1!/5.28
2320       ZYSIZE = 200 : ZXSIZE = 320 * TS .
           ZASFAC = 1.72069/TS
2330       ZXDOT = ZXSIZE/80! : ZYDOT = ZYSIZE/25!
2340       C(0)=0  C(1)=4: C(2)=1  C(3)=4  C(4)=2:
           C(5)=5: C(6)=3: C(7)=6
2350       RETURN
2360 REM
2370 REM - trap improper border fills due to lack of
```

```
    color on certain pc's
2380 REM
2390   IF TS = 1 THEN 2420
     REM have only monochrome
2400       ATR = 1   CULR = 1 . BRDR = 1
    :REM make then all the same
2410       RETURN
2420   IF CULR < 4 THEN COLOR .0 ELSE COLOR .1
    :REM medium resolution, remap
2430   IF BRDR < 4 THEN COLOR ,0 ELSE COLOR ,1
    :REM palette
2440   RETURN
2450 END
```

About The Author

Eric L. Pang *began this work while he was a systems analyst in the Department of Entomology at Texas A&M University. He is currently a programmer for Planning Research Corporation in Hawaii. On occasion, he does software design work with those great folks at SEACO in Kailua.*

*



Related Products

Jim Buszkiewicz
HUG Software Developer

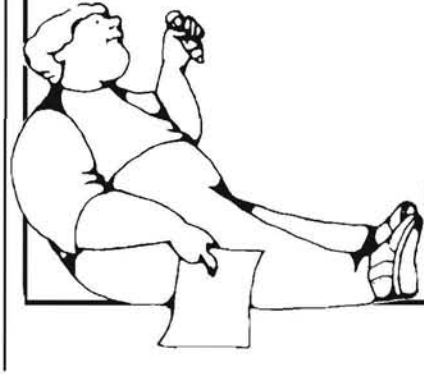
NOTE: *The following information was gathered from vendors' material. The products have not been tested nor are they endorsed by HUG. We are not responsible for errors in descriptions or prices.*

KCI Inc. of Ann Arbor, Michigan has announced the release of its LapDisk RAM Disk option for the Zenith Z-171 portable computer. The LapDisk is a small daughter board that attaches to the Z-171's memory card and allows the user to populate the memory card with four banks of 256k RAM chips for a total memory capacity of One Megabyte (memory chips are not included). The LapDisk allows DOS to access 640k of memory of application usage with 360k reserved for a RAM Disk. Installation is easy, requiring only a small screwdriver. The LapDisk offers four times faster access time than a standard floppy, longer battery life due to less access to the real diskette drive, and full memory availability provided to establish the RAM Disk and to save data to and from the RAM Disk three times faster than the COPY command under DOS. LapDisk retails for $99.95 and more information can be obtained from John Vasbinder at KCI Inc., P.O. Box 3344, Ann Arbor, MI 48106, (313) 426-2026.

Premier Technologies announced the first internal hard disk for the Zenith Z-171 Laptop computer. LiteDrive(tm) is a truly portable internal hard disk system. Engineered specifically for the Zenith Z-171, the system adds 27 times the original disk capacity at up to 20 times the access speed. LiteDrive(tm) is installed in place of the second floppy, has a capacity of 10 megabytes, and operates off battery and AC power. The entire unit adds a negligible one half ounce to the Z-171, bringing the total weight to a compact 14.33 pounds. The entire system is field installable in about 30 minutes. To minimize the worry of head crashes, an automatic park function has been implemented. Even in the most severe applications, this feature assures safe, reliable drive integrity by removing the head from the media area after 6 seconds of no access. LiteDrive(tm)'s 70g shock rating also means sound, durable operation in even the most extreme conditions. For more information, contact Premier Technologies, Inc., 1890 McGaw Avenue, Irvine, CA 92714, (714) 261-1184.

Have You Ever Asked?

1. *How can I retrieve programs from the HUG Bulletin Board?*
2. *How can I tell what programs to choose from the hundreds available?*
3. *How can I get a quick reference to the commands available on the HUG Bulletin Board?*
4. *How can I communicate with other HUG members that happen to be on-line on the HUG Bulletin Board with me?*



Get The HUG Bulletin Board Handbook (PN 885-4700)

It's $5.00 from the Heath Parts Dept.

Spreadsheets "Ain't" Just For The Bean Counters

Kenneth Mortimer, PE
352 Green Acres Drive
Valparaiso, IN 46383

Most people have the idea that spreadsheets are only used by accountants and other financial types. While they are the natural computer outgrowth of the bookkeeper's columnal pad and were originally developed for the financial users, they have uses other than in the financial area. If one looks at the LOTUS magazine or at H. W. Bauman's excellent "SPREADSHEET Corner" series in REMark you might be led to assume that spreadsheets were the exclusive property of the financial community. This is quite natural since they have become one of the most important tools of this group. One might think that the only use that an engineer would have for a spreadsheet was in preparing an estimate or in preparing a cost summary of a completed job. A look at some of the technical journals will show the spreadsheet being used for many technical calculations. I recently received a prospectus for a structural analysis textbook which mentioned that the disk that accompanied the text contained LOTUS 123 templates for certain solutions. Since I had already used LOTUS for Newmark's tabular method for the solution of the deflection of a beam, this came as no surprise to me. One of the major problems is the pride of technical types in our skills in the use of higher level languages and our tendency to look down on some of these simpler methods.

Why Spreadsheets?

Many technical calculations have been traditionally done in tabular form. Many others could be adapted to this form. Since the spreadsheet is a table that can be thought of as a matrix of numbers, it can be easily adapted to technical calculations. What are some of the apparent advantages of the use of the spreadsheet over conventional higher level language programming?

1. Many users do not view the generation of a spreadsheet template as programming and its use overcomes the "fear of coding."
2. There is no need for input statements and input formats. One just moves the cursor to the appropriate cell and enters the required label, number, or formula.
3. There is no need for output statements. The results are right in the table ready for you to print. You may have to set up the format for the various columns, but that is just a matter of answering some questions in the more friendly spreadsheets.
4. There is no worry about looping. The "repeat while" or "repeat until" conditions are automatically taken care of when you mark the range of an operation.
5. Matrix element index offsets are no problem. Once one understands the concept of relative and absolute cell addresses, it is simple to copy one line after the next.
6. Troubleshooting is easy. Since many of the partial results are displayed in various columns or rows, it is often easy to spot a "programming" error. Since the contents of each cell is computed as the cell is entered, one can observe the progress of the calculations and spot errors as one progresses. A simple editing of the particular equation immediately shows the results of the correction. However, remember that each corrected equation must be copied to the other cells in which it will be used.

How Can The Spreadsheet Be Used?

One can easily find a number of good examples of the use of the spreadsheet to solve problems in his or her own discipline. The results would be fine for one of the technical journals, but would have limited appeal to the readers of a general audience computer journal such as REMark. I have, therefore, chosen five examples that could be taken from a good quality high school or a beginning college physics course. Three examples are from mechanics, one from optics and one from circuit analysis.

Which spreadsheet should I use? Bauman's "SPREADSHEET Corner" uses LOTUS. LOTUS is the most popular spreadsheet at present. It has many subroutines that are useful in technical computations. I have a copy of LOTUS 123 ver. 3.0 at the office. So it looks like we will use LOTUS 123. I have even used HUG's CHEAPCALC for technical calculations, but it lacks subroutines and left to right arithmetic operations. Without the normal hierarchy of operations, it makes calculations very complex. If you have any other spreadsheet, it can easily be adapted to these

examples because I have carefully avoided the use of trigonometric functions in all but one example.

Example I: Resultant Of Concurrent Forces

The calculation of the resultant of several concurrent forces consists of four steps: entering the forces, resolving each into its components, adding the components of each force and finally vectorially adding the components of the resultant. I will use a different form than Bauman. I will start each entry with its cell number and then enter the contents of the cell. This will be followed by an explanation in parentheses. I will assume that you have some familiarity with spreadsheets. Only three forces are used in the example. You may expand the number if needed.

```
Enter the main title
A1  SPREAD SHEET FOR CALCULATING THE RESULTANT OF CON-
    CURRENT FORCES
Enter the column headings (centered)
A3  ^FORCE        (the magnitudes of the forces)
B3  ^X            (the x points on the line of action)
C3  ^Y            (the y points on the line of action)
D3  ^Z            (the z points on the line of action)
E3  ^R            (the direction vector length)
F3  ^FX           (the x component of the force)
G3  ^FY           (the y component of the force)
H3  ^FZ           (the z component of the force)
Enter the data
A4 to D6    Enter the data in the appropriate cells
Begin the calculations
E4  +(B3*B3+C3*C3+D3*D3) (the length of the direction
    vector)
COPY E4 to E4  E6
F4  +A4*B4/E4       (compute the x component of the force)
COPY F4 to F4  F6
G4  +A4*C4/E4       (compute the y component of the force)
COPY G4 to G4..G6
H4  +A4+D4/E4       (compute the z component of the force)
COPY H4 to H4  H6
A11 RESULTANT
F12 @SUM(F4..F6)    (sum of x components)
COPY F12 to F12 .H12  (sum of other components)
A12 +(F12*F12+G12*G12+H12*H12)^0.5 (compute the resultant)
B12 F12/$A$12       (direction cosine for the x direction)
COPY B12 to B12..D12  (other direction cosines)
```

The results of these calculations are shown in Figure 1. Note the use of the absolute cell address in Cell A12 so that the copied cell contents will always be divided by the resultant.

Example II: The Falling Body Test

A common experiment in physics is to drop a ball and measure the distance that it falls during small time increments. This may be done by using a spark gap or a strobe light. The measured results are then compared with the theoretical values.

```
Enter main title
A1   SPREAD SHEET FOR FALLING BODY TEST
Enter acceleration due to gravity
A2   ^g =
B2   9.81      (the value of g in meters/sec/sec)
Enter column headings
A7   ^TIME
B7   ^ACTUAL
C7   ^CALCULATED
D7   ^ERROR
B8   ^DISTANCE
C8   ^DISTANCE
E8   ^SQUARED
Enter data
A9 to A29   (use the fill command to enter tne times)
B9 to B29   (enter tne measured distances)
C9   +$B$2*A9*A9/2 (compute the theoretical distance fallen)
COPY C9 to C9..C29
D9   +C9-B9  (compute the error)
COPY D9 to D9 .D29
E9   +D9*D9  (compute the error squared)
A32  ^SUM
D32  @SUM(D9. D29) (compute the sum of the errors
COPY D32 to E32  (compute the sum of the error squared)
```

The results of these calculations are shown in Figure 2. If the measured distances are in feet, the appropriate value of "g" should be entered in B2. Note the relatively few cell entries that are needed for the calculations.

Example III: The Trajectory Problem

The trajectory of a projectile can be expressed in two parametric equations in terms of time. The horizontal distance is given by the equation:

$$x = v * \cos(\text{theta}) * t$$

and the elevation by the equation

$$y = v * \sin(\text{theta}) * t - g*t*t/2$$

where

v = the magnitude of the initial velocity
theta = the angle of elevation
g = the acceleration due to gravity
t = the time

This example was used not only to determine the coordinates of the path but to plot them.

```
Enter the main title
A1   SPREAD SHEET FOR TRAJECTORY PROBLEM
Enter constants
A2   GRAVITY
B2   ******** (safety code to prevent entering g in B2)
A3   INITIAL VELOCITY
```

SPREAD SHEET FOR CALCULATING THE RESULTANT OF CONCURRENT FORCES

FORCE	X	Y	Z	R	FX	FY	FZ
520	0	-10	-24	26	0	-200	-480
680	16	18	-24	34	320	360	-480
560	-12	8	-24	28	-240	160	-480

RESULTANT						
1477.294	0.054153	0.216612	-0.97475	80	320	-1440

Figure 1

```
A4   ANGLE OF ELEVATION
C2 to C4    enter appropriate data
F2   +180/@PI (constant to convert degrees to radian)
Enter column headings
A7   ^TIME
B7   ^X
C7   ^Y
Enter the equations
A8 to A38    (enter the times using the FILL command)
B8   +$C$3*@COS($C$4)*A8  (the equation for calculation x)
COPY B8 to B8..B38
C8   +$C$3*@SIN($C$4)*A8-($C$2/2)*A8^2 (the equation for
                                        calculation y)
COPY C8 to C8  C38
```

```
SPREAD SHEET FOR FALLING BODY TEST
     g =     9.8100

TIME      ACTUAL     CALCULATE    ERROR      ERROR
          DISTANCE   DISTANCE                SQUARED
0.0000     0.0000     0.0000     0.0000     0.0000
0.1000     0.0500     0.0491    -0.0009     0.0000
0.2000     0.2000     0.1962    -0.0038     0.0000
0.3000     0.4500     0.4415    -0.0085     0.0001
0.4000     0.8000     0.7848    -0.0152     0.0002
0.5000     1.2000     1.2263     0.0263     0.0007
0.6000     1.8000     1.7658    -0.0342     0.0012
0.7000     2.4500     2.4035    -0.0465     0.0022
0.8000     3.2000     3.1392    -0.0608     0.0037
0.9000     4.0000     3.9731    -0.0269     0.0007
1.0000     4.9000     4.9050     0.0050     0.0000
1.1000     6.0000     5.9351    -0.0649     0.0042
1.2000     7.1000     7.0632    -0.0368     0.0014
1.3000     8.1000     8.2895     0.1895     0.0359
1.4000     9.6000     9.6138     0.0138     0.0002
1.5000    11.0000    11.0363     0.0363     0.0013
1.6000    12.5000    12.5568     0.0568     0.0032
1.7000    14.0000    14.1755     0.1755     0.0308
1.8000    16.0000    15.8922    -0.1078     0.0116
1.9000    17.5000    17.7071     0.2071     0.0429
2.0000    20.0000    19.6200    -0.3800     0.1444

   SUM                          -0.0765     0.2847
```

Figure 2

Note the use of the absolute cell addressing to control the reading of the constants. The results of the calculations are shown in Figure 3. The plot of the results made using a Panasonic KX P1092 dot matrix printer in the high resolution mode are shown in Figure 4. The original plot occupied a whole 8-1/2 by 11 sheet.

Example IV: Optics Law

The relationship between the focal length of a lens, the object distance and the image distance is given by the equation:

$$1/f = 1/p + 1/q$$

where

f = the focal length of the lens
p = the distance from the lens to the object
q = the distance from the lens to the image (film in a camera)

The magnification of the object is given by the equation:

$$m = q/p$$

This example presents a table of image/object distances that is useful in close-up photography.

```
SPREADSHEET FOR THE TRAJECTORY PROBLEM
GRAVITY  ********     32.2000  57.2958 LAW
INITIAL VELOCITY     100.0000
ANGLE OF ELEVATION    30.0000

  TIME        X          Y
 0.0000    0.0000     0.0000
 0.1000    8.6603     4.8390
 0.2000   17.3205     9.3560
 0.3000   25.9808    13.5510
 0.4000   34.6410    17.4240
 0.5000   43.3013    20.9750
 0.6000   51.9615    24.2040
 0.7000   60.6218    27.1110
 0.8000   69.2820    29.6960
 0.9000   77.9423    31.9590
 1.0000   86.6025    33.9000
 1.1000   95.2628    35.5190
 1.2000  103.9230    36.8160
 1.3000  112.5833    37.7910
 1.4000  121.2436    38.4440
 1.5000  129.9038    38.7750
 1.6000  138.5641    38.7840
 1.7000  147.2243    38.4710
 1.8000  155.8846    37.8360
 1.9000  164.5448    36.8790
 2.0000  173.2051    35.6000
 2.1000  181.8653    33.9990
 2.2000  190.5256    32.0760
 2.3000  199.1858    29.8310
 2.4000  207.8461    27.2640
 2.5000  216.5064    24.3750
 2.6000  225.1666    21.1640
 2.7000  233.8269    17.6310
 2.8000  242.4871    13.7760
 2.9000  251.1474     9.5990
 3.0000  259.8076     5.1000
```

Figure 3

```
Enter the title
A1   Spreadsheet example---Object/Image Distance Law
     Enter the constant
A3   Focal Length
C3   50   ("standard" focal length for 35 mm camera)
Enter Column Headings
A5   ^OBJECT
B5   ^IMAGE
C5   MAGNIFICATION
A6   ^DISTANCE
B6   ^DISTANCE
Enter data and equations
A7 to A25  enter object distances   FILL makes this easy
B7   +1/(1/$C$3-1/A7)  (the equation to compute q)
COPY B7 to B7  B25
C7   +B7/A7   (the magnification equation)
COPY C7 to C7 .C25
```

The results of this calculation are shown in Figure 5.

Example V: Impedance Of A Series RC Circuit

In most physics and introductory circuits classes, one only calculates the impedance of a simple circuit for a single frequency and does not examine the results of the change in frequency. A graph is probably presented showing this effect. Sophisticated circuit analysis packages may be used to show the effect of frequency, but the spreadsheet provides a helpful tool in the presentation of the effect.

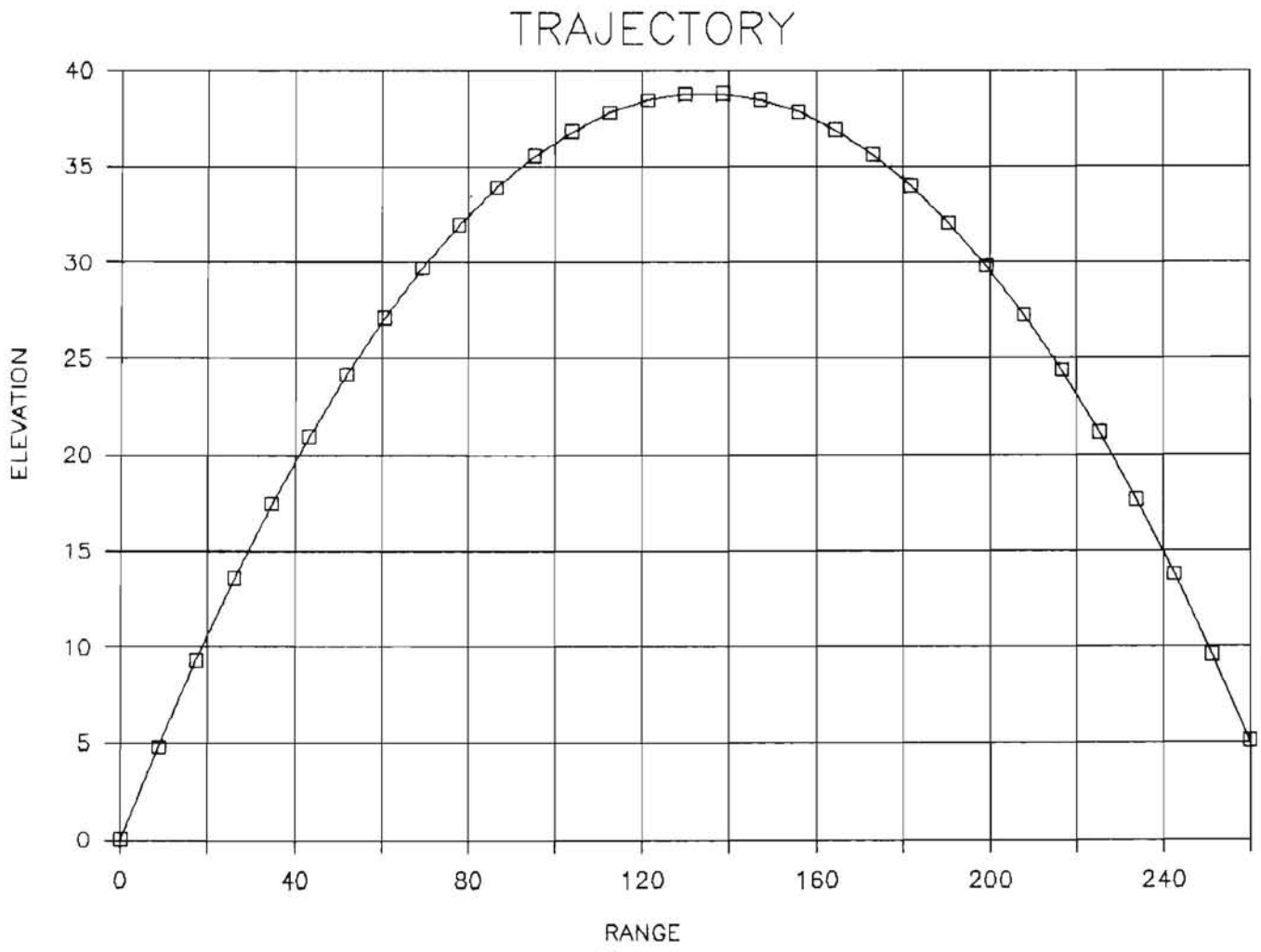


Figure 4

```
Enter the title
A1    Spreadsheet example__Impedance of a series RC circuit
Enter constants
A3    RESISTANCE-OHMS
C3    1000      (the resistance used in the example)
A4    CAPACITANCE-MFD (the capacitance in microfarads used)
Enter column heading
A6    ^FREQUENCY
B6    ^R          (the impedance of the resistance)
C6    ^ZC         (the impedance of the capacitor)
D6    ^IMPEDANCE (the series impedance of the two elements)
A7    ^HERTZ      (units)
Enter the data   and equations
A8 to A27   use FILL to enter the frequencies
COPY $C$3 to B8. B27
C8    +1/(2*?PI*$C$4*10^(-6)*A8)  (the equation for the
                                   impedance of the capacitor)
COPY C8 to C8..C27
D8    +(B8^2+C8^2)^0.5  (the series impedance equation)
COPY D8 to D8..D27
```

The results of this calculation are shown in Figure 6.

You now have five examples of the use of a spreadsheet to do some simple technical calculations. Often, as in higher language programs, the titles and column headings take more lines of code while the actual calculation take less coding. The copying of one cell's contents to a group of cells gives quite compact coding without a great deal of thought about subscript offsets. The accountants and financial planners have a good thing going in their spreadsheet systems. We technical people should put aside our pride and see what we can learn from them.

Spreadsheet example-Object/Image Distance

Focal Length 50.0

OBJECT DISTANCE	IMAGE DISTANCE	MAGNIFICATION
100.0	100.0	1.0000
150.0	75.0	0.5000
200.0	66.7	0.3333
250.0	62.5	0.2500
300.0	60.0	0.2000
350.0	58.3	0.1667
400.0	57.1	0.1429
450.0	56.3	0.1250
500.0	55.6	0.1111
550.0	55.0	0.1000
600.0	54.5	0.0909
650.0	54.2	0.0833
700.0	53.8	0.0769
750.0	53.6	0.0714
800.0	53.3	0.0667
850.0	53.1	0.0625
900.0	52.9	0.0588
950.0	52.8	0.0556
1000.0	52.6	0.0526

Figure 5

Spread sheet example----Impeadance
of a series RC circuit

RESISTANCE-OHMS 1000.00
CAPACITANCE-MFDS 8.00

FREQUENCY HERTZ	R	ZC	IMPEADANCE
10.00	1000.00	1989.44	2226.62
20.00	1000.00	994.72	1410.48
30.00	1000.00	663.15	1199.90
40.00	1000.00	497.36	1116.86
50.00	1000.00	397.89	1076.25
60.00	1000.00	331.57	1053.54
70.00	1000.00	284.21	1039.60
80.00	1000.00	248.68	1030.46
90.00	1000.00	221.05	1024.14
100.00	1000.00	198.94	1019.60
110.00	1000.00	180.86	1016.22
120.00	1000.00	165.79	1013.65
130.00	1000.00	153.03	1011.64
140.00	1000.00	142.10	1010.05
150.00	1000.00	132.63	1008.76
160.00	1000.00	124.34	1007.70
170.00	1000.00	117.03	1006.82
180.00	1000.00	110.52	1006.09
190.00	1000.00	104.71	1005.47
200.00	1000.00	99.47	1004.94

Figure 6

Continued from Page 10

```
                        }
                        printf("Proportional spacing?\n");
                        scanf("%s",a);
                    if(!strcmp(a,"Y") || !strcmp(a "y")) {
                            putc(27,fout); /* for prop
                                              space */
                            putc(91,fout);
                            putc(50,fout);
                            putc(121,fout);
                        }
                }
        }
        printf("Would you like emphasized print?\n");
        scanf("%s\n",a);
        if(!strcmp(a,"Y") || !strcmp(a,"y")) {
                putc(27,fout)  /* to get emphasized print */
                putc(91,fout);
                putc(61,fout);
                putc(122,fout);
        }
        fclose(fout);
        printf("Configuration completed\n");
}

cget()
{
int c;

#asm

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

        c = getchar()

#asm
        XRA     A
        MVI     B,000Q
        MVI     C,201Q
        SCALL   SCOUT
#endasm

        putchar(c);
        putchar('\n');
        return(c);
}

#include    "stdlib.c"
```

Error In Packet Radio — Part 1

Dear HUG:

My article, "Packet Radio Description And A Simulator — Part 1", contains a typesetting error I would like to correct. It is just the drop of a single word that leads to a misunderstanding of the Packet system.

The paragraph following the packet-frame representation at page 34 should read:

"The FLAG field is a unique 1 byte sequence 01111110. To make it unique, anytime five or more consecutive 1 bits are found in the data, a zero bit is added after the fifth 1 bit. The receiving TNC will realize that a "stuffing" was performed and delete the bit to restore the integrity of the data received."

To further clarify this concept, let me explain that frames are made up of blocks called fields. Since Packet Radio is a bit oriented protocol, the only way to discriminate the start and the end of a frame is delimiting the block with a certain bit sequence. This is the purpose of the FLAG field. This flag always consists of a zero followed by six ones and followed by one zero, or 01111110, which is also hexadecimal 7E. However, something should be done to allow five or more consecutive ones to also be part of the transmitted data. The trick is "stuffing".

Prior to packetizing, the program searches for five or more consecutive ones contained in the data field. If such sequence is found, a zero is added after the fifth one, eliminating the possibility of what seems to be a flag, to appear in the wrong place. To restore the data integrity, the protocol elminates a zero following five ones, anytime this sequence is present in the received data field.

I hope this clarifies the stuffing concept.

Best regards,

Luis E. Suarez
Apartado 66994
Caracas 1061-A
VENEZUELA

Juki 6100

Dear HUG:

Mr. Shoemaker's article on the Juki 6100 printer in the March 1986 issue of REMark was quite good. After getting my Z-100, I did a lot of serious comparisons of printers and the Juki 6100 was the best of the lot for the price and the features. It is so good, in fact, that my brother is getting one to add to his newly acquired Z-100 system.

Just to put in my two cents worth, I found I did not have to fool around with cables. I simply ordered a standard cable with all the pins connected with a DB-25 on one end and the Centronics-36 on the other. Everything works perfectly as long as you configure your hardware for a parallel printer.

Also, you can't get those low-priced printwheels from Gentry Associates anymore because they terminated their contract with the Juki line of printers as of mid 1985. So, if anyone out there knows another place where I can get printwheels less than $16, I'd sure like to know.

Also, for those who are interested, the Juki uses the IBM Selectric II ribbon with the orange leader. Depending on where you shop, the prices are nearly $3 each. However, if you purchase quantities of 12 or more, you can get these discounted to $1.55 and less from: Visible Computer Supply Corp., 3626 Stern Drive, St. Charles, IL 60174 and Smart Data Inc., 6302 Oakton Street, Morton Grove, IL 60053.

Sincerely,

1Lt Albert R. Witkins
PSC 1, Box 8236
F. E. Warren AFB, WY 82005

ZDOS To CP/M

Dear HUG:

In the October 1983 issue of REMark is a letter that describes a way to transfer data or programs from ZDOS disks to CP/M disks. I've got a Z-110 running MSDOS 2.2 and would like to have the same capability. Does anyone out there know how to do it.

Thank you,

Mike Wolfson
1125 Masters Avenue
Ashland, OH 44805

Continued from Page 41

Part Number	Description of Product	Selling Price	Vol. Issue
DATA BASE MANAGEMENT SYSTEMS			
HDOS			
885-1107-[37]	HDOS Data Base System H8/H89	30.00	23
885-1108-[37]	HDOS MBASIC Data Base Sys.	30.00	23
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			
885-6008-37	MSDOS NAVPROG	20.00	73
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
MSDOS			
885-8038-37	MSDOS RFCAD Ver. 3.50	30.00	73
COMMUNICATION			
HDOS			
885-1122-[37]	HDOS MicroNET Connection	16.00	37
885-8005	MAPLE (Modem Appl Effector)	35.00	29
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
885-8012-[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
885-3033-37	MSDOS HUG MCP	40.00	71
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	
885-4006	REMark Vol. VI Issues 60-71	25.00	
885-4500	HUG Software Catalog	9.75	
885-4501	HUG Software Catalog Update #1	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
885-3036-37	MSDOS TREE-ID	20.00	77

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 36

Page 137

Automatic Wire Stripper (64-1919)	$	8.95
"Helping Hands" (64-2093)		6.95
6" Locking Forceps (64-1866)		4.95
Soldering Aid Set (64-2227)		2.99
8-Piece Nutdriver Set (64-1800)		4.99
Combination Screwdrivers (64-1959 - p. 138)		3.49
Tapered Reamer (64-815 - p. 139)		2.49

Sears 1985/86 Power And Hand Tools Specialog

Screwholding Starter (9HT41024 - page 74) (Not recommended)	$	3.49
4-in-1 Offset Screwdriver (9HT4112 - p. 78)		6.99
5-inch Ignition File (9HT31205 - p. 135)		1.79
5-inch Ignition Pliers (9HT4513 - p. 137)		4.99
10-piece Combination Ignition Wrench Set		
(Standard, 9HT43441 - p. 141)		11.99
(Metric, 9HT43443 - p. 141)		11.99
Offset, Reversible Rachet Screwdriver (9HT4116 - p. 150)		4.99
Pocket Screwdriver (9HT41421 - p. 150)		2.99

Snap-On Tools

Flat Blade Screw Starter (See Text)
Phillips Screw Starter (See Text)

*

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.

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 885-2080