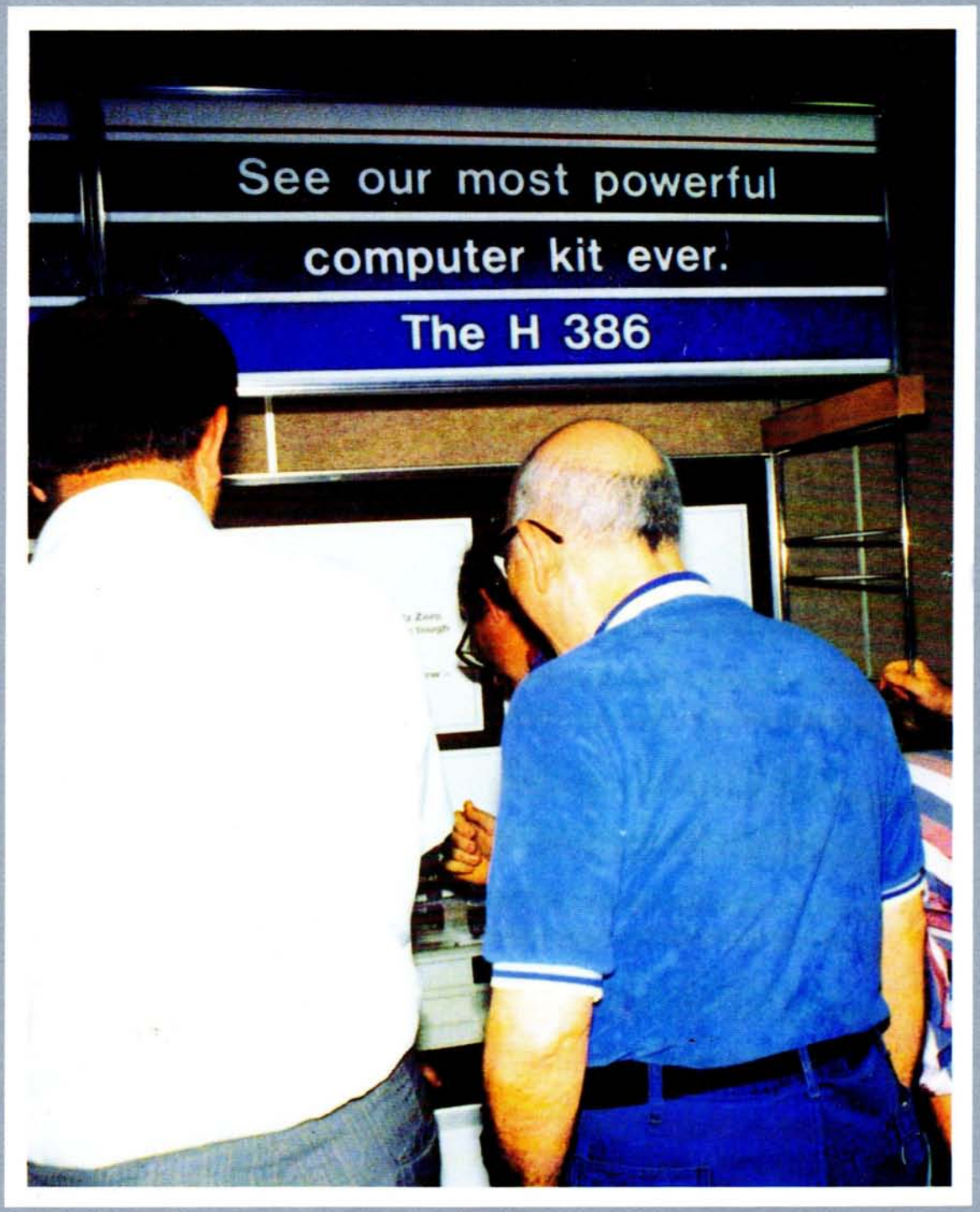
REMark®

Volume 9, Issue 3 • March 1988
P/N 885-2098 Issue 98

$2.50

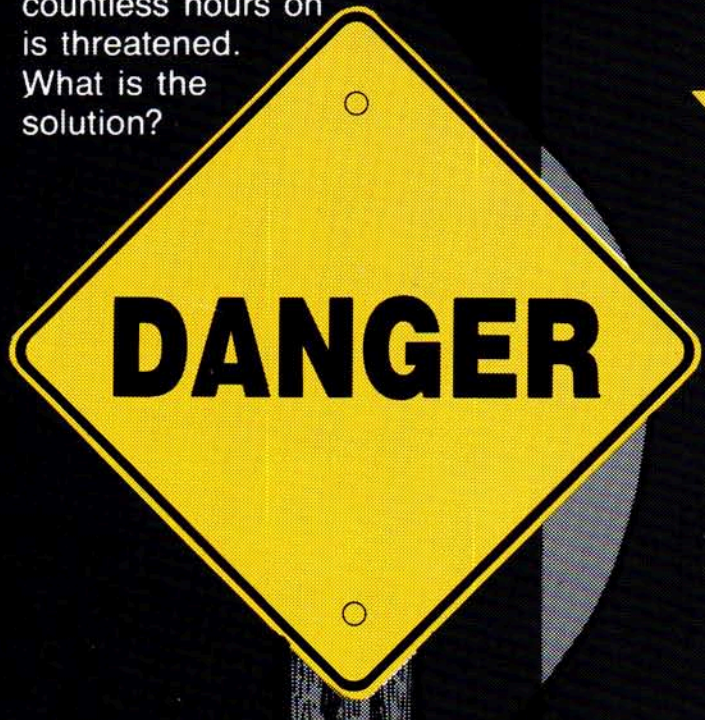
3.5" Drives For The H/Z-90
See Page 45



A Publication of Heath/Zenith

There's danger ahead if you don't backup your data!

The hidden dangers of destructive disk crashes, perilous power surges, and catastrophic keystroke errors lurk just around the corner. The safety of the precious data you've spent countless hours on is threatened. What is the solution?



The Imager tape backup system offers complete data protection. The Imager utilizes your home or business VCR and connects to any standard PC expansion slot. The Imager offers sophisticated software that allows you to backup or restore an entire disk with a single command or individual files by name or date. **NEVER LOSE DATA!**

ONLY $295.00

Dealer inquiries welcome.

Call today for more information: 800-821-7807
800-634-1967 (Outside Cal)

L-PC Lite-Pen Company
®MANUFACTURERS / PUBLISHERS

(213) 305-7616 / P.O. Box 45255 / Los Angeles, CA 90045

★★★ ZENITH SOFTWARE FOR THE ★★★ Z-100 SERIES COMPUTERS

Zenith packages with software, manuals and registration cards for the original Z-100 computer series (not for the IBM compatibles).

PART NUMBER	DESCRIPTION	LIST PRICE	SALE PRICE
MS-463-1	Z-Basic (16 bit) interpreter	$175.00	$10.00
MS-463-7	Multiplan	$195.00	$10.00
MS-253-1	BASIC-80 (8-bit)	$175.00	$10.00
CD-463-2	Condor File Manager	$299.00	$10.00

★★★ UPGRADE ACCESSORIES FOR Z-100 ★★★ SERIES COMPUTERS

HIGH DENSITY 1.2 MEG DRIVES. External floppy drive set-up complete with drive, power supply, case and cable. Ready to connect to your 8" floppy controller. $277.00. Dual Drive Unit $424.00

COLOR GRAPHICS UPGRADE. All memory chips (16 pieces 150 ns) required to update Z-100 Series computer for color. Installation instructions included. Order Memory Kit #100-64-16 ..... $36.00

MEMORY UPGRADE. All memory chips (9 pieces 150 ns) required to upgrade from 128K to 192K RAM. Installation instructions included. Order Memory Kit #100-64-9 ..... $20.25

ZMF100A by FBE Research. A modification package which allows 256K chips to be used on the old-style motherboard to reach 768K. Simple assembly with no soldering or trace cutting. Compatible with Easy PC and Gemini Emulator. $60.00 alone or $182.50 with 27 256K RAM chips included.

UCI MEMORY UPGRADE PAL CHIP SET. For Z-100's with the newer motherboard (181-4918 or greater). This set permits installation of 256K RAM chips on the motherboard. PAL set $64.00 or $187.00 with 27 256K chips required for 768K of memory on the motherboard.

SmartWatch by FBE Research. If you don't have a clock for your Z-100, get this one. More details under Z-150 upgrade listings $44.00

GEMINI EMULATOR BOARD. Makes the Z-100 compatible with the IBM PC library of programs $432.00

UCI EASY PC. IBM PC Emulator. Makes your Z-100 IBM Software Compatible. Full 8 MEG operation, color graphics and audio compatible. Retail $699.000, Payload $477.00

UCI EASY 87. Add an 8087 Numeric Coprocessor. $69.00 for the board without an 8087 Chip. With 5 MEG 8087 $197.00 or with 8 MEG 8087 installed $234.00

UCI MEMORY UPGRADE CARD. We recommend this one highly. The board has sockets for up to 2 MEG of RAM. With no RAM installed $328.00. With RAM installed and fully tested, 512K $408.00, One MEG $490.00, Two MEG $653.00 Add $35.00 for EasyDrive RAM Drive Software if desired.

UCI EASY-I/O. S-100 board that provides IBM PC communications port compatibility with your EasyPC. Easy I/O-1, One Serial Port $91.00. Easy I/O-2, Two Serial Ports, One Game Port, Clock-Calendar $127.00

UCI EasyWin. Winchester Drive Systems at reasonable prices. Complete Hard Disk Systems for mounting inside your Z-100. Systems complete with Seagate Drives, 21 MEG $578.00, 31 MEG $598.00 System without Drive and Controller $239.00

CDR Z-100 SPEED MODULE. Run your Z-100 Computer at 7.5 MHz. Installs easily with no soldering. Externally switchable between Speed and Normal mode. Payload $48.00

★★★ UPGRADE ACCESSORIES FOR H/Z-89 ★★★ COMPUTERS

Magnolia Microsystems Double Density Controller. Soft-sectored disk controller. Supports up to four each 5.25" and 8" disk drives. Complete with cables, installation instructions and CP/M ... $294.00

Magnolia Microsystems Memory Upgrade Card. Installs in your H-89 to increase memory from 48K to 64K. Payload price $50.00

INTERNAL DUAL DRIVE SETUPS. Includes two half height double sided disk drives and all hardware and connectors required to mount inside your H-89.
MITSUBISHI MF501 Setup, 48 TPI, 6 MS seek, 320K $220.00

★★★ HALF HEIGHT FLOPPY ★★★ DISK DRIVES

MITSUBISHI M501	5.25" 48 TPI DS/DD 320K/360K	$105.00
MITSUBISHI M504	5.25" 96 TPI DS/DD 360K/1.2 MEG .	$152.00
MITSUBISHI MF353	3.5" in 5.25" frame720K	$109.00
TOSHIBA ND04	5.25" 48 TPI DS/DD 320/360K	$89.50
TOSHIBA NDO4	with grey faceplate instead of black ...	$89.50
TOSHIBA NDO8	1.2 MEG for Z-200 or AT Computer ...	$119.00

★★★ UPGRADE ACCESSORIES FOR Z-148/150 ★★★ SERIES COMPUTERS

VMM 150 Video Card Eliminator for Z-150/160 computers. Allows use of EGA color card $45.00

SmartWatch from FBE Research. Installs in ROM Socket on CPU Board in Zenith computer series Z-100/148/150/158/160. This jewel of a product contains a ten year battery and keeps your computer informed of both time and date at each boot-up. Complete instructions and software included $44.00

MEMORY KIT #150-256-18. Includes a ZPAL chip which allows use of 256K RAM chips included (18 pieces 256K 150 ns RAM chips). Kit increases 128k memory to 640K or 256K memory to 704K. All chips plug into your existing Zenith Memory Board. Unbelievable but true. For Z-150/160 only $99.00

Winchester Hard Disk Drive Internal Set-up. Includes Winchester drive, controller/interface card, cables and all hardware. With 20 MEG (formated) drive $339.00 30 MEG $359.00. May be installed in Z-148 using an Expansion Card sold below.

PTZ-148 Expansion Card for Z-148. Includes 2 expansion slots plus a clock/calendar. $118.00

Winchester Hard Disk Drive External Set-up. Includes Winchester drive, controller/interface card, power supply and case with fan.
With 20 MEG (formated) drive $528.00
With 30 MEG (formated) drive $548.00

★★★ SEAGATE HARD DISK DRIVES ★★★

ST-225 20 MEG Winchester Hard Disk	$295.00
With Western Digital Controller & Cables	$339.00
ST-238 30 MEG, Requires RLL type controller	$309.00
With RLL Controller & Cables	$359.00
ST-4038 30 MEG High Speed for Z-200	$544.00
ST-4051 40 MEG High Speed for Z-200	$629.00
ST-4096 80 MEG High Speed with Software	$929.00
ST-251 40 MEG High Speed Z-150/200	$489.00

★★★ POWER SUPPLIES AND CASES FOR DISK DRIVES ★★★

Rugged steel construction with heavy duty power supplies. Purchase with drives and we will install drives in case.

Single 5.25" unit for Full Height Drive	$68.00
Dual 5.25" unit, for Half Height Drives	$92.00
Face plate for single drive in dual case	$8.00
For WINCHESTER Drive, with fan	$191.00

★★★ COMPUTERS ★★★

DTK 1000 XT TURBO Computer System. These are assembled and tested in our shop using the highest quality components - Mitsubishi floppy drives, Seagate hard disk drives, 120 ns RAM chips, 150 Watt power supply, expanded 101 key AT style keyboards, AT junior slide cases with keylock, etc. Each unit is burned in and fully tested prior to shipment. The front panel has "power on" and "hard disk access" indicator lights together with a Reset button and a lighted Turbo Speed button. In Turbo Speed the computer operates at 10MHz. We can prepare one customized to your specifications with Tape Backups, Hard Disk Drives, CGA or EGA Color, 3.5 or 5.25 inch floppy drives, Etc. Please call or write and we will prepare a quote for you. Two of our most popular configurations are quoted below. Add only $5.00 for UPS shipping within Continental U.S.

Package #1. Computer with one floppy drive, 1 Serial and 2 parallel ports, 512K RAM, Clock/Calendar and amber or green Monitor with tilt swivel base $689.00

Package #2. Computer and Monitor as above but with two floppy drives, a 30 Meg hard disk drive and 640K RAM $1096.00

ZFL-181-93 LAPTOP PORTABLE with the amazing Supertwist LCD backlit screen 640K RAM, two 720K 3.5" disk drives, clock, P & S Ports, 8 MEG. Retail $2399.00. Payload $1660.00

PAYLOAD
COMPUTER SERVICES

15718 SYLVAN LAKE, HOUSTON, TEXAS 77062
PHONE (713) 486-0687

Please **MAIL** or **PHONE** your order today and expect prompt service. **MASTERCARD** and **VISA** gladly accepted with no additional charge. All hardware carries a 90 or more day warranty. Add $5.00 to all prepaid orders for handling and shipping, we pay the balance. Texas Residents please add 8.0% sales tax. We accept purchase orders from schools, government and approved accounts.

Heath/Zenith Users' Group

Managing Editor ..... Jim Buszkiewicz
(616) 982-3837

Software Engineer ..... Pat Swayne
(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 M. Adney

Contributing Editor ..... Joseph Katz

Printer ..... Imperial Printing
St. Joseph, MI

	U.S. Domestic	APO/FPO & All Others
Initial	$22.95	$37.95*
Renewal	$19.95	$32.95*

* 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
P.O. Box 217
Benton Harbor, MI 49022
(616) 982-3838

REMark®

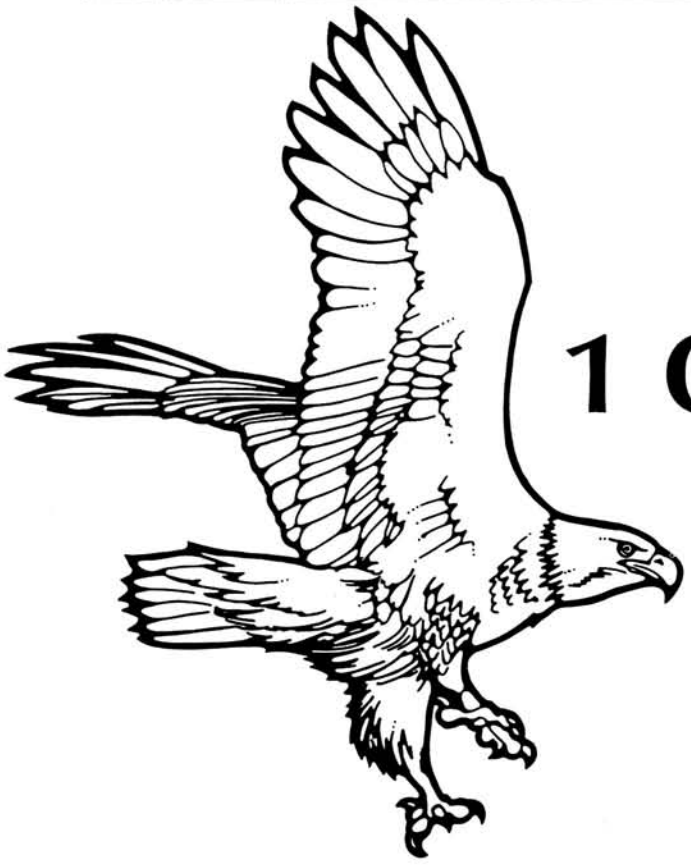
Official magazine for users of HEATH ZENITH computer equipment.

Volume 9, Issue 3 • March 1988

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: *Jim talks about the H-386 in his Editorial. See Page 6.*



1 Or 9.2 To 18.7

Jim Buskiewicz
Managing Editor

The original title was "25 Or 6 To 4" by a group called "Chicago". My title is actually by a person called Peter Norton. Back when the first real IBM PC became available, Mr. Norton wrote a disk full of utilities which became known as the infamous "Norton Utilities". One of these programs was called "System Information", or "SI.COM". When executed on **any** PC or PC Compatible, it gave you some information about the workings of your computer along with a 'speed index' number. This number appears to be relative, and a standard IBM PC running at 4.77MHz should result in a '1' when the program is executed. A Heath H-248 running at 8MHz (no wait states) will produce the second number, 9.2, and the new 16MHz H-386 will yeild the third number, 18.7. Does that mean that the '386 is almost 19 times faster than the original PC? Hard to say. Here's an example. Assembling HUGPBBS (my bulletin board program with a source size of about 150k), takes about 360 seconds on a standard 4.77MHz PC, 73 seconds on my H-248, and 43 seconds on the '386. Keep in mind that a good part of this time was spent in accessing the hard disk. This means that no matter how fast the processor is, it's going to get slowed down by hardware I/O (input-output).

The H-386 is your typical Heath Un-Kit (no soldering) and its construction is reminicent of the '241 and '248. The chassis, power supply, and drive mounting brackets are all very similar if not identical to the ones in the '200 series computer. The difference starts with the mother board. Instead of two connectors for each plug-in board, three now exist (remember, this is a 32-bit machine, so we need more address and data lines). Each board position is IBM AT compatible, however the third connector is proprietary to Zenith. Included with the kit are the following boards: CPU, I/O, Disk Controller, Memory (1meg), and Video (Z-449). A 4-meg memory and 'Cache' memory card are available as options. The only other difference is the addition of a second fan to help dissipate much of the heat.

The ROM monitor 'Setup' screen is identical to the '200' series with only one minor addition. During setup, you are allowed to select a processor speed of 'slow', 'fast', or 'smart'. The 'slow' mode actually is equivalent to the speed of an 8MHz '248. The fast mode allows the '386 to run at its full 16MHz, and the 'smart' mode apparently allows the unit to figure out what speed to run at. Unfortunately, the 'smart' mode isn't so smart. I ran the Norton "SI" utility in the smart mode, and the machine ran **slower** than a '248!

The H-386 comes with the new style Zenith keyboard which very few people seem to like. For a 'likeable' fix, see the Nov '87 issue of REMark, "Making Monkey See Monkey Do Livable".

The stock system comes with a single 1.2Mb disk drive, and sells for $3349.95. Also included is MS-DOS, '386 disk diagnostics, and Integrated 7+ software. If you can live with a slightly slower machine, then the '248 computer is probably the better buy, but, if **time** is of the essence, then the H-386 will let you soar with the eagles.

HUG members **and** non-members who have **built** the H-248 from kit, have I got an unbelievable deal for you!! I can't give you the specifics now, but be patient, and keep watching REMark for more information.

*

BUGGIN' HUG

Adding External Drives to the Z151 PC

Dear HUG:

Having just moved to MS-DOS with the purchase of a new Heath/Zenith 151 PC, I, like many others, was anxious to expand my new system. I have a S100 CP/M system which I had been home brewing over the past several years. One of the components of this system was a PC type cabinet containing a power supply and two Qume 142 5" drives.

I had read in the Heath/Zenith Z-100 PC Series Operations Manual that the system could accept up to four floppy disk drives. However, there was no information on how to connect the additional drives other than the setting of dip switch one on the CPU board.

The section on the disk controller board only said that it would support up to four floppy drives, one or two external. How to connect them was up to me.

There are two 34 pin connectors on the disk controller board, P501 and P502. The two internal drives were connected to P501. The first thing I did was set the dip switch one on the CPU board for four drives by positioning both positions 6 and 7 to the left.

I then made up a new cable with four drive connectors and set the device select jumpers on the two external drives to positions 2 and 3 (Drives C and D). I plugged this into P501 and fired it up. It did not work! This stumpted me for awhile as this is the configuration that I expected would work.

To cut a long story short, the correct procedure is as follows:

1. Set dip switch one on the CPU board for four (or three) drives.

2. Make up a separate cable for your one or two external drives and plug this into P502 on the controller board. Leave the two internal drives connected to P501 and make no changes to them at all.

3. Configur the two external drives with the device select jumper on the drive's PC boards for device 0 and 1 (A and B).

4. Put a terminating resistor pack in the drive furthest from the controller card.

That's all there is to it. Quick and easy. For some reason, the hardware software combination requires this configuration. The external drive which you selected as drive A (Device Select Jumper on 0) becomes drive C and the one you selected as drive B (Device Select Jumper on 1) becomes drive D.

I would not have got this working if I had not taken a very close look at the disk controller board and discovered that there was no connection at all to the device 2 and 3 select pins on the two connectors P501 and P502, only device 0 and 1 being connected. So what they do is select device 0 on P502 (external drives) when they want drive C and select device 1 on P502 when they want drive D.

I have had no problems whatsoever with the external drives and it is a real pleasure having four floppies in the system. At the moment, I have only 320K of memory, not enough to make an effective RAM disk, so the extra two drives come in real handy.

One example is Wordstar 2000+, which takes up all of drive A for the program and overlays and the dictionary disk takes up drive B. Having drives C and D available for program files allows me to use this program.

I hope this will save someone the hassle I went through when adding external drives to the Z-151.

Gord Wiggins
PO Box 164, St. A
Goos Bay NVLD CANAA
AOP ISO

Patch for the Peach Text 5000 Program

I have been a member for some five years now and have enjoyed reading REMark. Many of the improvements/patches published in your magazine have greeatly improved my software's performance and my satisfaction with the software. I want to thank you and all the HUGGIES for their dedication, unselfishness, and professionalism in support of the Heath User's Group.

I want to ask if there is anyone who has found a patch for the Peach Text 5000 program to add color. I have waited and watched for an article in REMark but to no avail. I know there must be a way much the same as the articles on Wordstar and Lotus 123 added color and speed to these programs, however I am not the technical type and would probably destroy the source code if I were to attempt such a delicate procedure. I know I would appreciate the help and I think there are a lot of others who would also appreciate jazzing up an old favorite.

Cordially,

David A. Dzeima, Capt., USAF
1-B Scott Circle
Bedford, MA 01731

Substituting the HD64180 for the Z80

Dear HUG:

Has anyone been able to substitute the HD64180 (Hitachi) for the Z80 in the H-89A?

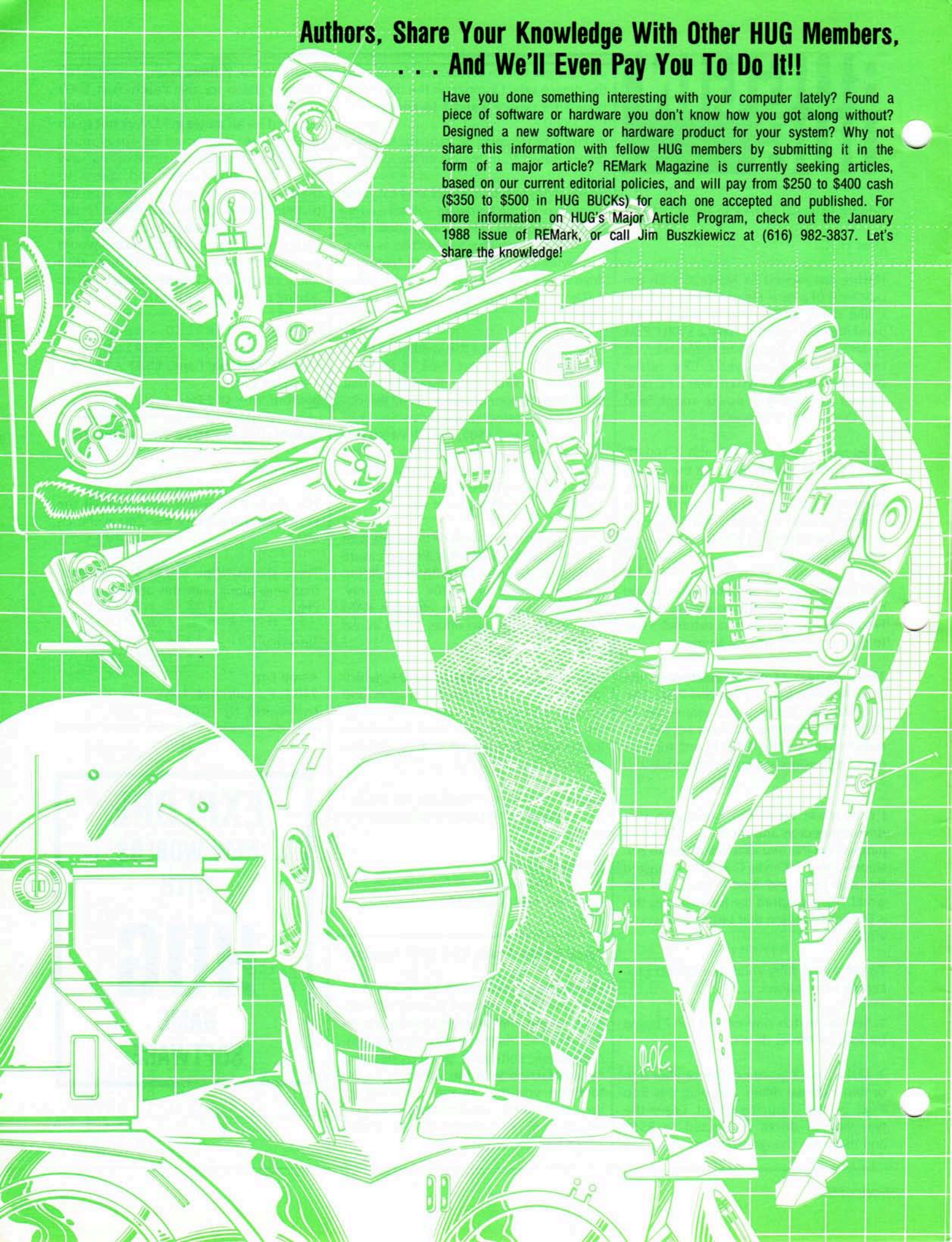
There seems to be sort of a sub-culture that goes along with this advanced 8 bit chip.

Sincerely,

Alfred Eriv
240 E. Palisade Ave, H-5
Englewood, NJ 07631

Authors, Share Your Knowledge With Other HUG Members, . . . And We'll Even Pay You To Do It!!

Have you done something interesting with your computer lately? Found a piece of software or hardware you don't know how you got along without? Designed a new software or hardware product for your system? Why not share this information with fellow HUG members by submitting it in the form of a major article? REMark Magazine is currently seeking articles, based on our current editorial policies, and will pay from $250 to $400 cash ($350 to $500 in HUG BUCKs) for each one accepted and published. For more information on HUG's Major Article Program, check out the January 1988 issue of REMark, or call Jim Buszkiewicz at (616) 982-3837. Let's share the knowledge!



CRTSAVER Revisited

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

This article was originally instigated by Joseph Katz' article in the April '87 REMark, where he makes suggestions as to how one can minimize the amount of memory wasted on the environment blocks of terminate-and-stay-resident (TSR) programs. I had learned from other sources, but never put into practice, that even more economy of memory is possible if TSR programs use MS-DOS' Free Allocated Memory function to release the environment block before they perform the code that causes them to remain resident.

Writing handy little resident spooks closely customized to my uses of the Z-100 has been a continuing hobby for me, with economy of memory a constant goal. I was at first unable to see how freeing up four or five paragraphs of memory immediately preceding my own resident TSR would do any good, since that little scrap of memory would be frozen in position ahead of my resident code, practically unuseable by any program with larger needs.

This led to a study of how MS-DOS manages memory. I put the results of the study to work as an enhancement to the CRTSAVER utility by Frank T. Clark, from his article "Advanced Assembly Language Programming" in the July '84 issue of REMark. I use his CRTSAVER routinely (from a batch file, how routine can you get), and expect that many other REMark readers do the same.

In addition to releasing its environment block, this version of CRTSAVER uses MS-DOS' chain of memory control blocks to detect if a resident copy of CRTSAVER all ready exists, to avoid inadvertently installing it twice, and it gives the user the option of defining how much inactivity time must pass before the screen will be blanked. My experience was that the original ten (plus) minutes was a little generous.

It must be pointed out that what is offered here as a description of how MS-DOS uses memory control blocks is strictly empirical. I have not had the privilege of any inside information from Microsoft or Zenith. There is some published material on the subject, apparently also empirically derived. My work suggests that much of it is wrong. If readers find some of this wrong, I apologize in advance, however, before stoning me in the public square, they should consider that Microsoft shares my guilt by keeping the information so private.

It has been suggested that perhaps Microsoft wanted to have this information closely held to discourage rash programmers who might be tempted to meddle with the memory "arena". It seems that rash programmers may do more damage operating in a realm of scanty information than in a realm of knowledge. Rash programmers are going to do their thing anyway, so here is a small rollback of the secrecy. The information should be of interest to users of the you-know-who compatibles, as well as to Z-100 users, since tests on a you-know-who's XT indicate that it is all equally applicable there.

For starters, it appears that memory control blocks are not used for any of the memory for the BIOS, or for MS-DOS itself. The first of the memory control blocks may control memory for a device driver if you load one by your CONFIG.SYS file; otherwise, it will be for COMMAND.COM or an alternate command processor, if one is in use. If VDISK or MDISK is installed, there will be a control block for all of the memory space devoted to it.

Each memory control block occupies one whole paragraph, 16 bytes, starting on a paragraph boundary. For those not used to the term "paragraph", but who use Debug, its Dump function displays one paragraph per screen line. Only the first five bytes of the control block are used for significant data in DOS2 and 3. The control

block defines the "ownership" of a number of paragraphs of contiguous memory immediately following the block.

The first byte of a control block will be either 04DH, ASCII 'M', or 05AH, ASCII 'Z'. 04DH identifies the block as either the first in the chain of control blocks, or as an intermediate member. 05AH identifies the block as the last member of the chain.

The second and third bytes are a memory word that is the segment address of the program that "owns" the memory, and the fourth and fifth bytes are a memory word that state the number of paragraphs of memory, immediately following the control block that are under its control, and are for the use of the "owner". If the memory has been released and is available for use, the owner word is zero.

When COMMAND.COM loads a .COM program, it is given control over all remaining contiguous memory. So a .COM file is necessarily loaded higher in memory than anything else, and in loading it, MS-DOS cannot take advantage of released areas of memory that exist farther down in memory. A released area could exist as the result of a released environment block, or it is possible for it to exist by the release of memory formerly devoted to a TSR program that has been evicted.

However, for an .EXE program, which is only assigned as much memory as its source code directed, if a released area of memory is of sufficient size, it can be used even though it may be embedded between other areas of reserved memory. This fact can make it worth while to have a means for evicting TSR programs that are no longer needed, even though they do not control the highest block of reserved memory.

After a .COM file is loaded into memory and receives control, it can release the memory past its own code. Then, if a released block of memory exists farther down, and the .COM file makes a request (by INT 21H, function 48H) to be allocated a block of memory that can fit within the released block, the request will be satisfied by an allocation from the released block. This is a fact that would probably seldom be of practical use, but the point is that once a program is loaded and operating, MS-DOS does not treat memory allocation requests from a .COM file any differently than those from an .EXE file.

As to how little scraps of memory freed by releasing the environment blocks of TSR programs can be utilized, the answer is obvious (after you find it). If the first TSR loaded frees its environment block, the next TSR loaded can use up that freed region for its own environment block rather than to reserve a new region. If the second TSR frees it, it can be used again by a third, and so on. If the environment block was initially over minimum size, and you load three or four TSR utilities, the saving can become quite significant.

It takes some devious special programming to demonstrate this, but it too is obvious by hindsight. When no foreground program is operating, and COMMAND .COM is patiently awaiting your beck and call, it owns all of the remaining available memory. The implication is that all programs that operate by user direction from the command line are child processes of COMMAND.COM and are passed control of appropriate amounts of memory, which then reverts to COMMAND.COM when they end.

Now, on to the modified version of CRTSAVER. Although the part of CRTSAVER that remains resident is all, but identical to the original, there are differences. When I first attempted to assemble Clark's original source code, the version of MASM that had kept giving me error messages until I moved the memory assignment directives to a position earlier in the source code than the instructions that referred to them. They have such a position here.

Another difference is that all the symbolic names for numbers, which required reference to include files, have been replaced by the explicit values. The DOS2 versions of the originally named include files do not seem to define all of the symbolic names, resulting in a flood of error messages. This CRTSAVER version will assemble with MASM versions from 1.07 through 4.0.

A third difference is at the label ON where the value 04650H is placed in the variable TIMER, rather than 0FFFFH (-1), as in Clark's original. This gives an inactivity time of three minutes if the user does not specify a choice of other timing, whereas 0FFFFH would give an inactivity time close to eleven minutes. I hope that readers will be patient with my use of Hexadecimal notation where decimal would serve just as well. When I first set out to learn Assembly Language programming, I had an understanding of Hex notation, but no fluency with it, and have made a habit of using it just to become more fluent.

The part of CRTSAVER that does the installation is where the enhancements are. CRTSAVER provides a useful TSR installation framework that, with a little adaptation, could serve very well to emplace other resident routines, so for this part the differences from Clark's original version will be explained in detail.

After the label OFF, starting with THIS__SITE and ending with ENDIF, is a group of assembler directives that will cause the assembler to start the INSMSG string on a paragraph boundary. This is part of the continuing strategy to impose work on the assembler if it can result in a reduction of executable code. The INT 21H function 31H, recommended for use with DOS2 and 3 in lieu of INT 27H, requires input of the number of paragraphs to be held resident. By having the offset of INSMSG on a paragraph boundary, the number of paragraphs is easily obtainable by shifting the offset of INSMSG right by four bits.

The Z-DOS manual, on page 10.115, and the Programmer's Utility Pack, Version 2, on page 10.76, give another example of how to use assembler directives for setting to a paragraph boundary. These would require putting a new assembler directive after the CRTSAVER SEGMENT directive. I preferred this arrangement as introducing less new material in the part of the source code that was supposed to mimic Clark's original, however, be warned that this only works if the starting point of SPAN is already on a paragraph boundary (BEGIN, at 0100H in this case).

Following the ENDIF directive, the EQU statement defining ENV__SEG is part of the process for releasing the program's environment block. MS-DOS places the segment of the environment block at offset 02CH in the program header. The Free Allocated Memory function, 049H, requires the segment address in the ES register of the memory to be freed. The 8088 MOV instruction allows us to move a value directly from memory into a segment register. If we give a symbolic name to the content of offset 02CH, the segment value there can be moved to ES with a mini-

mum of executable code. MASM will not let us assign an immediate value by the EQU directive, but it will allow us to define a memory item as being at an offset relative to some other memory location that it "knows" about, so we can define ENV__SEG as at some negative offset relative to BEGIN.

Having established an easy means for placing the right value in ES, the release of the environment block then takes only the three lines of source code at label INS020, resulting in just eight bytes of binary code. In this instance, since the .COM file will usurp at least 1024 bytes of space on a double-sided floppy, and since this installation code does not remain resident, using a few more bytes of code to put the value in ES some other way would be of little consequence. However, if one is to make an illustration, then let it be a neat clean one.

Similar reasoning about disk storage space and the fact of non-resident status is responsible for two of the error messages that are provided for very improbable causes of abortive termination. Finding memory allocation in error, or finding the currently executing CRTSAVER defective, would probably never happen to most users, but if one of those should occur, there is no economy-of-space reason to leave the user wondering what went wrong.

The definition of a memory location named DELAY is part of the means by which the user may set the amount of inactivity time that will elapse before the screen is blanked. The location defined as DELAY is the part of the instruction statement at the label ON that establishes the inactivity time. When the program is invoked, the delay time can be input as a decimal number on the command line, after the program name, as in:

```
A:CRTSAVER 180 or B:CRTSAVER/180
```

The number is the seconds of delay. It may be separated from the file name by either a space or a slash. If the user gives a number less than 20 seconds or over 600 seconds, the default time of 180 seconds will be used.

The Assembly Language to set a user-selectable delay starts at label INS140 and continues to label INS180. The conversion of the ASCII characters on the command line to a single equivalent number is done in the instructions from label INS150 to INS160. The algorithm takes the ASCII representations of a decimal number one digit at a time, starting at the most significant digit, and incorporating it into the final result in its properly weighted value. BX is used to accumulate the final number. As each digit is taken from the command line, the prior value of BX (initially 0) is multiplied by 10, then the current digit is added and the result returned to BX for another entry to the top of the loop. A "hundreds" digit is thus multiplied by 10 two times, a "tens" digit multiplied by 10 once, and a units digit not multiplied at all.

As pointed out in Clark's original article, the timer interrupt, which counts down the value placed in DELAY, is effectively at .01 second intervals, so the number of seconds must be multiplied by 100. The three instruction statements at INS170 do this and put the final number in DELAY.

The first instructions after the label INSTALL determine the MS-DOS version number, and terminate with an error message if it is not version 2 or above. Our use of the chained memory control blocks to avoid dual installations of CRTSAVER starts with the fourth instruction line after the label INS 020. Here, we acquire the number of the first non-existent paragraph of memory. MS-DOS puts the number at offset 2 in the program header and we hold it in DX to define when we are all through with the search for memory control blocks.

We set out to find the chain of memory control blocks in quite a brute force way, just examining the first byte of each paragraph from the very beginning of memory on, looking for an 04DH character. Each time that one is found in that position, we use the procedure REACH to see if the data word 3 bytes higher will enable us to locate another 04DH character. If it does, there is a reasonable probability that the first 04DH character is the identifying byte of a memory control block. To give an acceptably low probability of error, REACH is called a second time, using the information in the presumed second control block to see if it can be used to locate the identifying byte of yet a third (probable) control block.

By recourse to probability theory, an argument could be made that the odds of failure in this dual test routine are not worse than 1 in 32768. They would be that bad only if the tests were made in a completely random field of memory, where our actual field of memory is the highly organized MS-DOS and BIOS. The odds are improved further by the fact that once we conclude that we have located the chain of memory control blocks, it is still treated as a tentative conclusion until we have used the chain to locate our own currently operating code segment.

The instructions that make this initial finding of the chain are those from label INS030 down to INS060. We can be assured that there are enough memory control blocks to do what those instructions intend, because in the absolute minimal situation there will be one block set up for COMMAND.COM, or its equivalent, plus one for the environment block of CRTSAVER.COM and one for the executable code of CRTSAVER.COM.

The instructions from INS070 down to INS140 follow the chain all the way to the top of memory, testing the "owner" of each block of memory to see if it is a previously installed CRTSAVER, or if it is the code segment of the currently executing CRTSAVER. Each of those findings is recorded in the state of the NEW__CPY and OLD__CPY flags. When the top of memory is reached, to proceed with the installation of this CRTSAVER requires that this one, and no other, was found. It sounds like a lot of testing, but it actually executes very fast, because there are not all that many memory control blocks and the signature tests of non-CRTSAVER code generally fail, and end, at the first byte.

The 12 lines of assembly source following the label INS180 and the called procedure SET__VEC are comparable to the means shown in Clark's article for placing an interrupt vector to direct the interrupt to our code, and for storing the existing interrupt vector as an exit address for our code. Using this as an exit address ensures that the functions of the previous interrupt handler are also executed. Clark's method would work here, but since this version will only run under MS-DOS2 or 3, we might as well illustrate the DOS functions, which are just slightly simpler for the user and make DOS responsible for the proper use of CLI and STI instructions.

With a reliable way to find and make use of the chain of memory control blocks, and thus to locate the executable code of a TSR program, it becomes feasible to control and/or modify the TSR by means

Listing 1

```
CRTSAVER        SEGMENT

        ASSUME  CS:CRTSAVER,DS:CRTSAVER,ES:CRTSAVER,SS:CRTSAVER

        ORG     0100H

BEGIN:  JMP     INSTALL

CRT_OFF DB      0
TIMER   DW      ?
NXT_CRT DD      ?
NXT_KB  DD      ?
NXT_TM  DD      ?

INT_KB:
        CALL    TESTIT
        JMP     DWORD PTR CS:NXT_KB

INT_CRT:
        CALL    TESTIT
        JMP     DWORD PTR CS:NXT_CRT

TESTIT  PROC    NEAR
        TEST    CS:CRT_OFF, -1
        JE      ON

        PUSH    AX
        IN      AL, 0D8H
        AND     AL, 0F8H
        OR      AL, 8
        OUT     0D8H, AL
        MOV     CS:CRT_OFF, 0
        POP     AX

ON:     MOV     CS:TIMER, 04650H  ;3 minute delay
        RET
TESTIT  ENDP

INT_TM:
        SUB     CS:TIMER, AX
        JC      OFF_CRT
        JMP     DWORD PTR CS:NXT_TM

OFF_CRT:
        TEST    CS:CRT_OFF, -1
        JNE     OFF
        PUSH    AX
        IN      AL, 0D8H
        AND     AL, 0F7H
        OR      AL, 7
        OUT     0D8H, AL
        MOV     CS:CRT_OFF, -1
        POP     AX

OFF:
        JMP     DWORD PTR CS:NXT_TM
```

of transient programs. It would be possible to create a TSR program with a special section of code that would evict its own resident code if the program was run a second time with a command line switch that activated the eviction routine. Another possibility would be a keyboard macro substitution program where the table of macro substitutions was replaced by another table, useful for different purposes, under control of a transient program. A TSR program could be made that did nothing else, but reserve a region of memory into which other resident routines could be installed and removed at will, using a transient program. I am sure the readers of REMark can come up with many more variations on this theme.

```
THIS_SITE       LABEL NEAR              ;Or, LABEL BYTE, but
SPAN    EQU     THIS_SITE - BEGIN       ;must have an attribute

IF      SPAN MOD 16
        ORG     (THIS_SITE + 16) - (SPAN MOD 16)
ENDIF

ENV_SEG EQU     WORD PTR BEGIN - 0D4H   ;Set up symbolic names
DELAY   EQU     WORD PTR ON + 5         ;for these memory items

INSMSG  DB      0DH,0AH,'CRTSAVER routines installed.'
        DB      0DH,0AH,024H
NOTMSG  DB      0DH,0AH,'CRTSAVER not installed ...'
        DB      0DH,0AH,0AH,024H
DOSMSG  DB      'Wrong version of MS-DOS.',0DH,0AH,024H
EXIMSG  DB      'Resident copy of CRTSAVER already exists.'
        DB      0DH,0AH,024H
MEMMSG  DB      'Memory allocation is in error.',0DH,0AH,024H
FAIMSG  DB      'This copy of CRTSAVER may be defective.'
        DB      0DH,0AH,024H

MULT1   DW      0AH
MULT2   DW      064H

OLD_CPY DB      0
NEW_CPY DB      0

INSTALL:

; Determine MS-DOS version number.  Must be 2.0 or over

        MOV     AH, 030H
        INT     21H
        CMP     AL, 2
        JAE     INS020
        MOV     DX, OFFSET NOTMSG       ;If not 2 or over
        MOV     AH, 9
        INT     21H
```

```
        MOV     DX, OFFSET DOSMSG
        MOV     AH, 9
        INT     21H
        INT     20H                     ;For Z-DOS termination

INS010: PUSH    DX
        MOV     DX, OFFSET NOTMSG       ;Show common message
        MOV     AH, 9                   ;first,
        INT     21H
        POP     DX                      ;Then specific message
        MOV     AH, 9
        INT     21H
        MOV     AH, 04CH                ;Terminate function
        INT     21H                     ;for MS-DOS2 or 3

; Set ES to segment of environment block, release it

INS020: MOV     ES, ENV_SEG
        MOV     AH, 049H
        INT     21H

; Determine if CRTSAVER is already resident in memory.  Do not
; retain this copy in memory if so.  Locate any existing copy
; by locating DOS's chained memory control blocks and working
; our way up the chain.

        MOV     DX, WORD PTR DS:[2]     ;Top of memory, from PSP
        XOR     BX, BX

INS030: CMP     BX, DX                  ;Are we at the top?
        JB      INS040

        MOV     DX, OFFSET FAIMSG       ;Should never be at top
        JMP     INS010                  ;while in this loop

INS040: MOV     ES, BX
        CMP     BYTE PTR ES:[0], 04DH   ;Marks start of a block
        JNE     INS050

; When the first genuine memory control block is found, it will
; be validated by the fact that, in two successive calls to the
; procedure REACH, the value of ES, plus one, plus the word at
; ES:[3] points to another paragraph whose first byte is either
; 04DH or 05AH. An 04DH character may exist as the first in a
; paragraph just as a random event, in executable code. 04DH is
; the instruction to "DEC     BP", or could be a data byte.

        CALL    REACH
        CMP     CL, 04DH                ;If not 04DH, pass over
        JNE     INS050                  ;paragraph held in BX
        CALL    REACH
        CMP     CL, 04DH                ;If second test succeeds
        JE      INS060                  ;break out of this loop
        CMP     CL, 05AH
        JE      INS060

INS050: INC     BX
        JMP     INS030
```

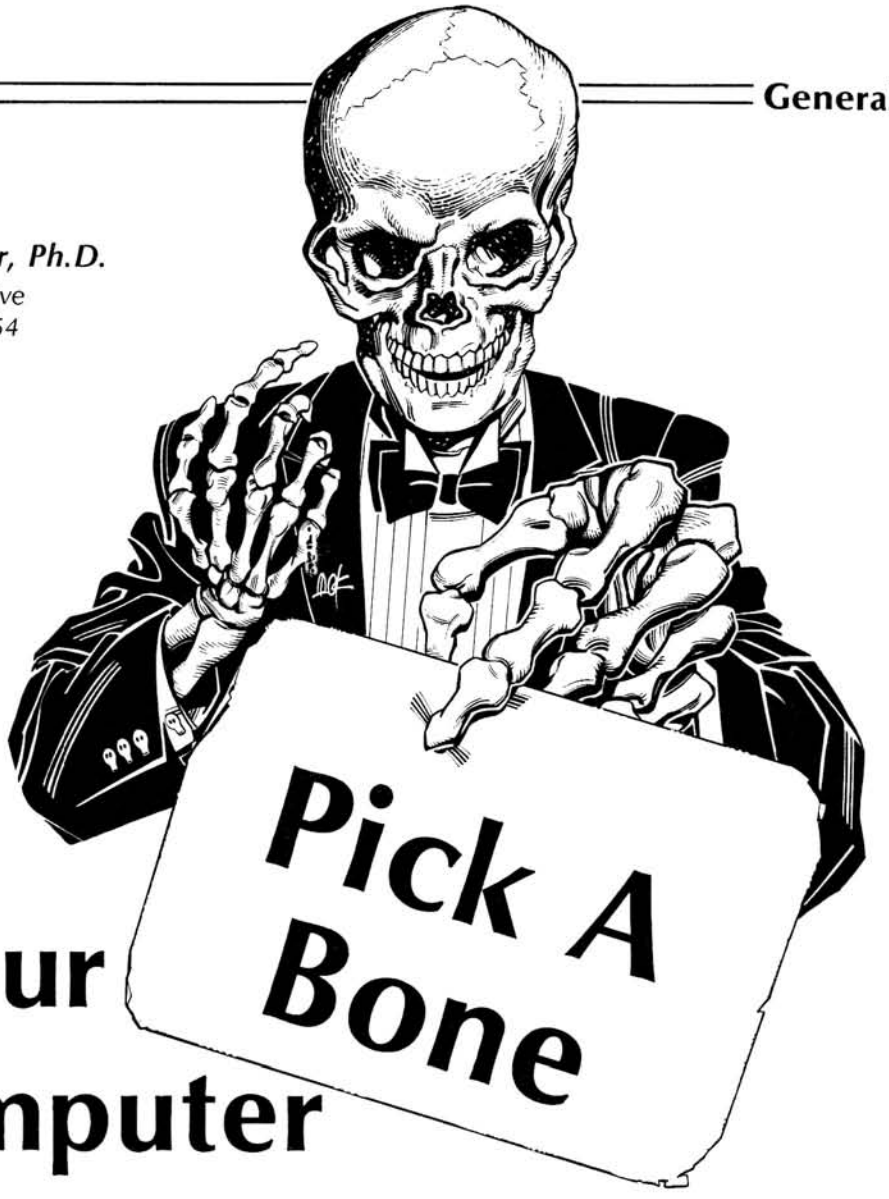
```
INS060: MOV     ES, BX

; When we arrive at INS070, we are presumably into the chain
; of memory control blocks.  This continues to be verifiable by
; success in using one element of the chain to locate another.

INS070: CALL    REACH                   ;For next in the chain
        CMP     AX, DX                  ;AX has ES value
        JB      INS100

; When at the top of memory, decide what to do based on flags
; we set on the way

        CMP     OLD_CPY, 0              ;If old copy found
        JE      INS080

        MOV     DX, OFFSET EXIMSG       ;write error message
        JMP     INS010                  ;and bail out

INS080: CMP     NEW_CPY, 0              ;If this copy verified
        JNE     INS140                  ;go to next task

INS090: MOV     DX, OFFSET MEMMSG       ;Else show error message
        JMP     INS010                  ;and bail out

; If CL has 04DH or 05AH, we should perform a signature check
; except that if the block is our code segment, it is not
; needed. If not one of those, we have an error situation

INS100: CMP     CL, 04DH
        JE      INS110
        CMP     CL, 05AH
        JNE     INS090

INS110: MOV     AX, WORD PTR ES:[1]
        MOV     CX, CS
        CMP     AX, CX
        JNE     INS120
        INC     NEW_CPY
        JMP     INS070

INS120: PUSH    ES
        MOV     ES, AX
        MOV     SI, OFFSET INT_KB       ;Compare this part of
        MOV     DI, SI                  ;code area as signature
        MOV     CX, 020H                ;Compare 32 bytes
        CLD
        REPE    CMPSB
        POP     ES
        JCXZ    INS130                  ;If both areas same
        JMP     INS070                  ;else continue search

INS130: INC     OLD_CPY                 ;Set "in place" flag
        JMP     INS070                  ;and continue search

; Get user's input from command line for inactive time

INS140: XOR     DX, DX
```

```
        MOV     CX, DX
        MOV     BX, DX
        MOV     SI, 080H
        LODSB
        OR      AL, AL                  ;If nothing there, skip
        JZ      INS180                  ;this number input
        LODSB                           ;Get delimiter

; Permit only space or slash as first character after the
; filename.

        CMP    AL, 020H
        JE      INS150
        CMP     AL, '/'
        JNE     INS180

; The INS150 loop acquires the inactive time number from the
; command line. Only Ascii characters 0 through 9 are allowed,
; else the default inactive time will be used.

INS150:  LODSB
        CMP     AL, 0DH                 ;0DH = all done
        JE      INS160
        CMP     AL, 030H
        JB      INS180
        CMP     AL, 039H
        JA      INS180
        AND     AL, 00001111B           ;Strip off Ascii part
        MOV     CL, AL                  ;Hold till BX multiplied
        MOV     AX, BX                  ;BX has prior result
        MUL     MULT1                   ;Multiply it by 10
        ADD     AX, CX                  ;Add this new digit
        MOV     BX, AX                  ;Put new result in BX
        JMP     INS150                  ;Do until 0DH found

; Require that time be at least 20 seconds, not more than 600.
; Otherwise, use the existing 04650H value for DELAY.

INS160: OR      DX, DX                  ;DX must remain zero,
        JNZ     INS180                  ;else delay is too large
        CMP     BX, 0258H
        JA      INS180
        CMP     BX, 014H
        JB      INS180

INS170: MOV     AX, BX
        MUL     MULT2                   ;Make .01 second units
        MOV     DELAY, AX               ;Move to executable code

INS180: MOV     AL, 055H                ;Interrupt number
        MOV     DX, OFFSET INT_CRT      ;Our interrupt handler
        MOV     DI, OFFSET NXT_CRT      ;Where exit address goes
        CALL    SET_VEC

        MOV     AL, 050H
        MOV     DX, OFFSET INT_KB
        MOV     DI, OFFSET NXT_KB
        CALL    SET_VEC
```

```
        MOV     AL, 051H
        MOV     DX, OFFSET INT_TM
        MOV     DI, OFFSET NXT_TM
        CALL    SET_VEC

        MOV     DX, OFFSET INSMSG
        MOV     AH, 9
        INT     21H
        MOV     DX, OFFSET INSMSG

; DX has size of the stay-resident area. Convert to paragraphs

        MOV     CL, 4
        SHR     DX, CL
        MOV     AH, 031H                ;Keep process function
        INT     21H                     ;End of this phase

SET_VEC PROC    NEAR
        PUSH    AX                      ;Save interrupt number

; Get exit address in place before we direct the interrupt here.

        MOV     AH, 035H                ;Function to get prior
        INT     21H                     ;interrupt vector
        MOV     [DI], BX                ;Store its offset
        MOV     [DI + 2], ES            ;and its segment

        POP     AX                      ;Recover AL
        MOV     AH, 025H                ;Function places vector
        INT     21H                     ;to our code
        RET
SET_VEC ENDP

REACH   PROC    NEAR

        MOV     AX, ES
        ADD     AX, WORD PTR ES:[3]
        INC     AX
        MOV     ES, AX
        MOV     CL, BYTE PTR ES:[0]
        RET

REACH   ENDP

CRTSAVER        ENDS

        END     BEGIN
```

*

Paul M. Spannbauer, Ph.D.
One Brockley Drive
Delmar, NY 12054



With Your Computer

Introduction

Have you ever tried to learn the names of bones in the human body? The computer program described below will provide the correct anatomical name of a skeletal bone when questions are answered about the bone's location in the body. The program has been used to assist students in learning the names of bones in a biology course, but the program could also be useful in anatomy, zoology, health, and other related courses.

The program will run without modification on any Zenith IBM-compatible ('148, '159, etc.) that runs GW-BASIC (BASICA). The statement format can be easily modified for computers that use other versions of BASIC.

Background Information

The human skeleton is broken down into axial and appendicular divisions. The axial division includes the bones that make up the main body axis: skull, spinal column, and thoracic (chest) cage. The appendicular division includes the bones that make up the limbs: shoulders, arms, hips, and legs. The individual bones included in these two divisions are listed in Table 1.

In order to answer some of the program's questions about a bone's location, knowledge of the following directional terms is required:

Medial — Near the middle of the body.
Lateral — Near the side of the body.
Anterior — Near the front of the body.
Posterior — Near the back of the body.
Anatomical Position — The body is facing forward, with the arms at the sides and the palms of the hands facing forward.

Picking A Bone

After beginning the program, the location of a bone to be identified must be decided upon. This can be done by observing an articulated human skeleton by looking at a diagram or X-ray photograph of a skeleton, or by picturing the skeleton in your mind. The computer will ask questions about the location of the bone in an effort to narrow down the possibilities until the bone can be identified. The program will then loop back to the beginning and ask if another bone is to be identified.

Out On A Limb

The runstream shown below demonstrates how the program works. The ulna in the arm has been used as an example.

This program will provide you with the anatomical names of skeletal bones. Look at a human skeleton and select a bone that you would like identified. Then answer each question by typing the NUMBER that applies to the bone. Then press the RETURN key. Is the bone:

1. AXIAL
2. APPENDICULAR

?2

Is the bone part of the:

1. SHOULDER
2. ARM
3. HIP
4. LEG

?2

Is it in the:

1. UPPER ARM
2. LOWER ARM
3. WRIST
4. HAND

?2

When the arm is in anatomical position, is the bone:

1. MEDIAL
2. LATERAL

?1

The bone you have selected is an ULNA. WOULD YOU LIKE TO TRY ANOTHER BONE?

1. YES
2. NO

?2

OK

When the bone has been identified, its name will blink on the screen ("an ULNA" blinks in the above example). The use of numbered options eliminates the need for entering exact spellings and dealing with capital vs. lower case letters.

Summary

All 206 skeletal bones can be named by the program. However, no provision has been made for identifying "left" or "right" paired bones or individual bone markings.

The program allows students to work at their own pace and to assume control over their own learning. The program format would lend itself to use in other allied health courses where classification and terminology are of importance.

Have fun picking a bone with your computer!

Table 1
Skeletal Bones

Axial Bones		Appendicular Bones	
Skull	Ethmoid	Shoulders	Clavicle
	Frontal		Scapula
	Lacrimal	Arms	Carpals
	Mandible		Humerus
	Maxillae		Metacarpals
	Nasal		Phalanges
	Nasal conchae		Radius
	Occipital		Ulna
	Ossicles	Hips	Os coxae
	Parietal	Legs	Femur
	Sphenoid		Fibula
	Temporal		Metatarsals
	Vomer		Patella
	Zygomatic		Phalanges
Neck	Hyoid		Tarsals
Thoracic cage	Ribs		Tibia
	Sternum		
Spinal column	Cervical vertebrae		
	Coccyx		
	Lumbar vertebrae		
	Sacrum		
	Thoracic vertebrae		

```
10 REM "BONES" - By PAUL SPANNBAUER, Ph.D.
20 REM CREATED 05/17/87
30 REM
40 CLS
50 COLOR 14
60 PRINT TAB(22);"----------------------------------------"
70 PRINT TAB(22);"¦             B O N E S              ¦"
80 PRINT TAB(22);"¦                                    ¦"
90 PRINT TAB(22);"¦                by                  ¦"
100 PRINT TAB(22);"¦                                   ¦"
110 PRINT TAB(22);"¦      Paul M. Spannbauer, Ph.D.      ¦"
120 PRINT TAB(22);"¦                                   ¦"
130 PRINT TAB(22);"¦   Dept. of Microcomputer Services   ¦"
140 PRINT TAB(22);"¦   Hudson Valley Community College   ¦"
150 PRINT TAB(22);"¦            Troy, N.Y.             ¦"
160 PRINT TAB(22);"----------------------------------------"
170 COLOR 2
180 PRINT:PRINT TAB(10);"This program will provide you with the anatomical
    names of"
190 PRINT TAB(10);"skeletal bones. Look at a human skeleton and select a bone
    that"
200 PRINT TAB(10);"you would like identified. Then answer each question by
    typing"
210 PRINT TAB(10);"the NUMBER that applies to the bone. Then press the RETURN
    key."
220 REM
230 PHRASE$ = "The bone you have selected is "
240 REM
250 COLOR 14
260 PRINT:PRINT "Is the bone:"
270 PRINT:PRINT TAB(5);"1 -- AXIAL"
280 PRINT TAB(5);"2 -- APPENDICULAR"
290 INPUT A
300 IF A = 1 GOTO 1180
310 PRINT:PRINT "Is the bone part of the:"
320 PRINT:PRINT TAB(5);"1 -- SHOULDER"
330 PRINT TAB(5);"2 -- ARM"
340 PRINT TAB(5);"3 -- HIP"
350 PRINT TAB(5);"4 -- LEG"
```

```
360 INPUT B
370 IF B = 1 GOTO 410
380 IF B = 2 GOTO 500
390 IF B = 3 GOTO 820
400 IF B = 4 GOTO 840
410 PRINT:PRINT "Is the bone:"
420 PRINT:PRINT TAB(5);"1 -- ANTERIOR"
430 PRINT TAB(5);"2 -- POSTERIOR"
440 INPUT C
450 IF C = 2 GOTO 480
460 PRINT:PRINT PHRASE$;:COLOR 31:PRINT"a CLAVICLE.";:COLOR 14
470 GOTO 2420
480 PRINT:PRINT PHRASE$;:COLOR 31:PRINT"a SCAPULA.";:COLOR 14
490 GOTO 2420
500 PRINT:PRINT "Is it in the:"
510 PRINT:PRINT TAB(5);"1 -- UPPER ARM"
520 PRINT TAB(5);"2 -- LOWER ARM"
530 PRINT TAB(5);"3 -- WRIST"
540 PRINT TAB(5);"4 -- HAND"
550 INPUT D
560 IF D = 1 GOTO 600
570 IF D = 2 GOTO 620
580 IF D = 3 GOTO 710
590 IF D = 4 GOTO 730
600 PRINT:PRINT PHRASE$;:COLOR 31:PRINT"a HUMERUS.";:COLOR 14
610 GOTO 2420
620 PRINT:PRINT "When the arm is in anatomical position, is the bone:"
630 PRINT:PRINT TAB(5);"1 -- MEDIAL"
640 PRINT TAB(5);"2 -- LATERAL"
650 INPUT E
660 IF E = 1 GOTO 690
670 PRINT:PRINT PHRASE$;:COLOR 31:PRINT"a RADIUS.";:COLOR 14
680 GOTO 2420
690 PRINT:PRINT PHRASE$;:COLOR 31:PRINT"an ULNA.";:COLOR 14
700 GOTO 2420
710 PRINT:PRINT PHRASE$;:COLOR 31:PRINT"a CARPAL.";:COLOR 14
720 GOTO 2420
730 PRINT:PRINT "Is the bone in:"
740 PRINT:PRINT TAB(5);"1 -- THE PALM OF THE HAND"
750 PRINT TAB(5);"2 -- A FINGER"
760 INPUT F
770 IF F = 1 GOTO 800
780 PRINT:PRINT PHRASE$;:COLOR 31:PRINT"a PHALANX.";:COLOR 14
790 GOTO 2420
800 PRINT:PRINT PHRASE$;:COLOR 31:PRINT"a METACARPAL.";:COLOR 14
810 GOTO 2420
820 PRINT:PRINT PHRASE$;:COLOR 31:PRINT"an OS COXA.";:COLOR 14
830 GOTO 2420
840 PRINT:PRINT "Is it in the:
850 PRINT:PRINT TAB(5);"1 -- UPPER LEG"
860 PRINT TAB(5);"2 -- KNEE"
870 PRINT TAB(5);"3 -- LOWER LEG"
880 PRINT TAB(5);"4 -- ANKLE"
890 PRINT TAB(5);"5 -- FOOT"
900 INPUT G
910 IF G = 1 GOTO 960
920 IF G = 2 GOTO 980
930 IF G = 3 GOTO 1000
940 IF G = 4 GOTO 1090
950 IF G = 5 GOTO 1110
960 PRINT:PRINT PHRASE$;:COLOR 31:PRINT"a FEMUR.";:COLOR 14
970 GOTO 2420
980 PRINT:PRINT PHRASE$;:COLOR 31:PRINT"a PATELLA.";:COLOR 14
990 GOTO 2420
1000 PRINT:PRINT "When the leg is in anatomical position, is the bone:"
1010 PRINT:PRINT TAB(5);"1 -- MEDIAL"
1020 PRINT TAB(5);"2 -- LATERAL"
1030 INPUT H
1040 IF H = 1 GOTO 1070
1050 PRINT:PRINT PHRASE$;:COLOR 31:PRINT"a FIBULA.";:COLOR 14
1060 GOTO 2420
1070 PRINT:PRINT PHRASE$;:COLOR 31:PRINT"a TIBIA.";:COLOR 14
1080 GOTO 2420
1090 PRINT:PRINT PHRASE$;:COLOR 31:PRINT"a TARSAL.";:COLOR 14
1100 GOTO 2420
1110 PRINT:PRINT "Is the bone in:"
1120 PRINT:PRINT TAB(5);"1 -- THE SOLE OF THE FOOT"
1130 PRINT TAB(5);"2 -- A TOE"
1140 INPUT I
1150 IF I = 2 GOTO 780
1160 PRINT:PRINT PHRASE$;:COLOR 31:PRINT"a METATARSAL.";:COLOR 14
1170 GOTO 2420
1180 PRINT:PRINT "Is the bone in the:"
1190 PRINT:PRINT TAB(5);"1 -- SKULL"
1200 PRINT TAB(5);"2 -- NECK"
1210 PRINT TAB(5);"3 -- CHEST"
1220 PRINT TAB(5);"4 -- LOWER BACK"
1230 PRINT TAB(5);"5 -- BASE OF THE SPINE"
1240 INPUT J
1250 IF J = 1 GOTO 1660
1260 IF J = 2 GOTO 1300
1270 IF J = 3 GOTO 1390
1280 IF J = 4 GOTO 1550
1290 IF J = 5 GOTO 1570
1300 PRINT:PRINT "Is the bone part of the spinal column?"
1310 PRINT:PRINT TAB(5);"1 -- YES"
1320 PRINT TAB(5);"2 -- NO"
1330 INPUT K
1340 IF K = 1 GOTO 1370
1350 PRINT:PRINT PHRASE$;:COLOR 31:PRINT"the HYOID bone.";:COLOR 14
1360 GOTO 2420
1370 PRINT:PRINT PHRASE$;:COLOR 31:PRINT"a CERVICAL VERTEBRA.";:COLOR 14
1380 GOTO 2420
1390 PRINT:PRINT "Is it part of the SPINAL COLUMN?"
1400 PRINT:PRINT TAB(5);"1 -- YES"
1410 PRINT TAB(5);"2 -- NO"
1420 INPUT L
1430 IF L = 2 GOTO 1460
1440 PRINT:PRINT PHRASE$;:COLOR 31:PRINT"a THORACIC VERTEBRA.";:COLOR 14
1450 GOTO 2420
1460 PRINT:PRINT "Is it in the middle of the chest?"
1470 PRINT:PRINT TAB(5);"1 -- YES"
1480 PRINT TAB(5);"2 -- NO"
1490 INPUT M
1500 IF M = 1 GOTO 1530
1510 PRINT:PRINT PHRASE$;:COLOR 31:PRINT"a RIB.";:COLOR 14
```

```
1520 GOTO 2420
1530 PRINT:PRINT PHRASE$;:COLOR 31:PRINT"the STERNUM.";:COLOR 14
1540 GOTO 2420
1550 PRINT:PRINT PHRASE$;:COLOR 31:PRINT"a LUMBAR VERTEBRA.";:COLOR 14
1560 GOTO 2420
1570 PRINT:PRINT "Does the bone have holes (foramina)?"
1580 PRINT:PRINT TAB(5);"1 -- YES"
1590 PRINT TAB(5);"2 -- NO"
1600 INPUT N
1610 IF N = 1 GOTO 1640
1620 PRINT:PRINT PHRASE$;:COLOR 31:PRINT"the COCCYX.";:COLOR 14
1630 GOTO 2420
1640 PRINT:PRINT PHRASE$;:COLOR 31:PRINT"the SACRUM.";:COLOR 14
1650 GOTO 2420
1660 PRINT:PRINT "Is most of the bone:"
1670 PRINT:PRINT TAB(5);"1 -- OUTSIDE THE SKULL"
1680 PRINT TAB(5);"2 -- INSIDE THE SKULL"
1690 INPUT O
1700 IF O = 2 GOTO 2130
1710 PRINT:PRINT "Is the bone on the:"
1720 PRINT:PRINT TAB(5);"1 -- FOREHEAD"
1730 PRINT TAB(5);"2 -- TOP"
1740 PRINT TAB(5);"3 -- SIDE"
1750 PRINT TAB(5);"4 -- BACK"
1760 PRINT TAB(5);"5 -- FACE"
1770 INPUT P
1780 IF P = 1 GOTO 1830
1790 IF P = 2 GOTO 1850
1800 IF P = 3 GOTO 1870
1810 IF P = 4 GOTO 1890
1820 IF P = 5 GOTO 1910
1830 PRINT:PRINT PHRASE$;:COLOR 31:PRINT"the FRONTAL bone.";:COLOR 14
1840 GOTO 2420
1850 PRINT:PRINT PHRASE$;:COLOR 31:PRINT"a PARIETAL bone.";:COLOR 14
1860 GOTO 2420
1870 PRINT:PRINT PHRASE$;:COLOR 31:PRINT"a TEMPORAL bone.";:COLOR 14
1880 GOTO 2420
1890 PRINT:PRINT PHRASE$;:COLOR 31:PRINT"the OCCIPITAL bone.";:COLOR 14
1900 GOTO 2420
1910 PRINT:PRINT "Is it part of the:"
1920 PRINT:PRINT TAB(5);"1 -- NOSE"
1930 PRINT TAB(5);"2 -- UPPER JAW"
1940 PRINT TAB(5);"3 -- LOWER JAW"
1950 PRINT TAB(5);"4 -- CHEEK"
1960 PRINT TAB(5);"5 -- MEDIAL EYE SOCKET"
1970 INPUT Q
1980 IF Q = 1 GOTO 2030
1990 IF Q = 2 GOTO 2050
2000 IF Q = 3 GOTO 2070
2010 IF Q = 4 GOTO 2090
2020 IF Q = 5 GOTO 2110
2030 PRINT:PRINT PHRASE$;:COLOR 31:PRINT"a NASAL bone.";:COLOR 14
2040 GOTO 2420
2050 PRINT:PRINT PHRASE$;:COLOR 31:PRINT"a MAXILLA.";:COLOR 14
2060 GOTO 2420
2070 PRINT:PRINT PHRASE$;:COLOR 31:PRINT"the MANDIBLE.";:COLOR 14
2080 GOTO 2420
2090 PRINT:PRINT PHRASE$;:COLOR 31:PRINT"a ZYGOMATIC bone.";:COLOR 14
2100 GOTO 2420
2110 PRINT:PRINT PHRASE$;:COLOR 31:PRINT"a LACRIMAL bone.";:COLOR 14
2120 GOTO 2420
2130 PRINT:PRINT "Does the bone make up the nasal septum?"
2140 PRINT:PRINT TAB(5);"1 -- YES"
2150 PRINT TAB(5);"2 -- NO"
2160 INPUT R
2170 IF R = 1 GOTO 2250
2180 PRINT:PRINT "Does the bone have wings?"
2190 PRINT:PRINT TAB(5);"1 -- YES"
2200 PRINT TAB(5);"2 -- NO"
2210 INPUT S
2220 IF S = 2 GOTO 2340
2230 PRINT:PRINT PHRASE$;:COLOR 31:PRINT"the SPHENOID bone.";:COLOR 14
2240 GOTO 2420
2250 PRINT:PRINT "Is it part of the TOP or BOTTOM of the nasal septum?"
2260 PRINT:PRINT TAB(5);"1 -- TOP"
2270 PRINT TAB(5);"2 -- BOTTOM"
2280 INPUT T
2290 IF T = 1 GOTO 2320
2300 PRINT:PRINT PHRASE$;:COLOR 31:PRINT"the VOMER.";:COLOR 14
2310 GOTO 2420
2320 PRINT:PRINT PHRASE$;:COLOR 31:PRINT"the ETHMOID bone.";:COLOR 14
2330 GOTO 2420
2340 PRINT:PRINT "Is the bone inside the ears?"
2350 PRINT:PRINT TAB(5);"1 -- YES"
2360 PRINT TAB(5);"2 -- NO"
2370 INPUT U
2380 IF U = 2 GOTO 2410
2390 PRINT:PRINT PHRASE$;:COLOR 31:PRINT"an AUDITORY OSSICLE.";:COLOR 14
2400 GOTO 2420
2410 PRINT:PRINT PHRASE$;:COLOR 31:PRINT"a NASAL CONCHA.";:COLOR 14
2420 PRINT:PRINT:BEEP
2430 PRINT:PRINT"WOULD YOU LIKE TO PICK ANOTHER BONE?"
2440 PRINT:PRINT TAB(5);"1 -- YES"
2450 PRINT TAB(5);"2 -- NO"
2460 INPUT V
2470 IF V = 1 GOTO 40
2480 CLS
2490 END
```

*

Computers, Music And MIDI

Part 3

T. E. Thompson
61 Sample Road
Ottawa, Ontario
CANADA K1V 9T9

PC Compatible Owners Take Note: *Even though the original computer used for the completion of this article was a Heath H-89, the Asynchronous Communications Element (8350 ACE) is the same device used for the 'COM' serial ports in all PC compatibles. Further, a good portion of the support software is written in a higher level language, making it easily transportable. The Z-80 assembly language code could easily be hand translated to 8088/86 code.* -ed.

In the first two articles of this series, we covered the Musical Instrument Digital Interface (MIDI) standard and a hardware MIDI interface for the H-89 computer. This month, I'll present three software packages to make use of the interface. The H-89 MIDI port I described could be adapted to other computers, if you are familiar with the computer bus, and concepts in the software that follows could also be used to produce a useful package for another machine. I am staying close to the H-89 because that is the machine that needs the support, but I hope the ideas are of assistance to owners of other computers, as well.

Writing useful software for the H-89 was a challange for two reasons. First, the 2MHz clock speed of the H-89 is slow when one considers all the work that has to be done between individual data bytes coming in at about one every 300 microseconds. There is time for only about 200 assembly language instructions depending on the instruction mix before the next byte is ready for processing.

Second, the code in the operating system for dealing with the 2 millisecond TICCOUNT system clock is long and very complex to maintain compatibility with the H8 control panel operation. The clock is used for disk interface timing and cannot be disabled entirely, so the next best thing is to disable all interrupts during a MIDI input sequence and to re-enable the interrupts at the end. This has the side effect of disabling the computer keyboard, as well, so we have a rather serious trade-off problem. The result of these two problems is that there is no real-time recording capability in the software presented here. I am still working on it, though.

There are five basic software functions to be performed in carrying out any useful MIDI capability. They can be broken down as follows:

1. Enter step time note and command information.
2. Compile the note data into a playable data file.
3. Play the data file on a synthesizer.
4. Save and restore patch data.
5. Record and play back from the synthesizer in real time.

I'll talk about each of these functions in turn, and I'll discuss the programs included in Listings 1 to 3. I used some commercial software, but I had to write the three examples shown here, which I use for functions 2, 3 and 4. These programs use a variety of languages and MIDI techniques, as the idea was to experiment with the hardware and to educate myself.

I learned a great deal about C programming and MIDI, and I had a lot of fun doing it.

The first function is the entry of data in "step time". Step time is used to enter notes individually with no need to be a keyboard expert to get the data correct. I used a commercial editor (PIE from Software Toolworks) to do this job. The important point is to adhere to the proper format for the note data. Notice that the

first line in the .SCR file is ignored by the data compiler, so it can be used to put a title on the file. Figure 1 is an illustration of the format with some application notes, and Figure 2 is a sample of the first 4 bars of Erik Satie's "Gymnopedie Number 2". In this sample, the synthesizer voices I used were my 77 for piano and 69, a flute-like sound. The basic format was borrowed from the 885-1031 HUG Music disk for the H8 and rather heavily modified. Credit for the concept, however, belongs to James Grogan who wrote the original H8 and H-89 software.

The second function necessary is to compile the raw note data into a data file that can be used by the music player program.

This is the C language program called MUSIC6 that is shown in Listing 1. This program will read the note data file (which must have an .SCR extension) and produce a music data file with a .DAT extension. Notice that there are several functions that are specific to the MIDI operations. I have since pulled these functions out into a MIDI library that I link into my C-based MIDI programs to avoid re-inventing them every time.

For the third function, I use the PLAY6 program in Listing 2. This program is written in Macro80 assembler for speed and efficiency sake. I could possibly have succeeded in making a C version run fast enough to keep 6 voices going, but once I had the assembler version working I went on to other things! The MIDI set up sequence is specific to the 8250 UART used in the MIDI interface and the first operation is to use this code to initialize the UART for the correct baud rate and data format. PLAY6 will then read the entire song into memory and search through the data from the start for the first note (or rest) to play in each of the six voices. The note data is then loaded into the next note buffer for temporary storage, and finally, into the current note area when the computer is ready to play it. Note timing and all note-on and note-off commands are controlled by the internal clock, so timing resolution is limited to 2 mS. When all voices reach the end of the playable data, the program pauses. You must hit a control-C to get back to HDOS at that point. Control-C will stop the program at any time, but be warned that this will probably result in the dreaded stuck synthesizer notes that can be cured only by turning the power off! I have also found some interesting results from playing random data, such as text or .ABS files.

The fourth useful function is the ability to save and restore the parameters for the voice patches in the synthesizer. The program in Listing 3, VOICE2, is much more specific to the SixTrak synthesizer I own than the other programs are. PLAY6 will play music on any synthesizer, but patch transfer is a system exclusive message type that is closely tied to the hardware. The design concepts are the same though; just check your owner's manual for the machine specific MIDI message details. This program is written in C and uses some of the same subroutines as MUSIC6. The logical next step is to build a capability to edit the patch parameters and to reload them to the synthesizer.

Some commercial programs also have the ability to use random numbers to set the parameters and to generate new random voice sounds for further experimentation. This is an area that I have not yet had an opportunity to try, but it is next on my list.

Figure 1

Score data for the MUSIC6 compiler is to be entered into a file with the extension .SCR. When MUSIC6 is run, it will ask for a file name to process and will assume an extension of .SCR, so no extension is required on the entry. The output will go to a file on the same drive with the extension .DAT which will be in the correct format for PLAY6. Data in the .SCR file should be in the following format:

1. Each note or rest is defined by a 5 or 6 character group as follows:

```
                                                                    vnaodd

v - voice number 0 to 5_____________________________________________¦ ¦ ¦ ¦ ¦ ¦
                                                                    ¦ ¦ ¦ ¦ ¦ ¦
n - note name A to G________________________________________________¦ ¦ ¦ ¦ ¦
           R - rest                                                 ¦ ¦ ¦ ¦ ¦
                                                                    ¦ ¦ ¦ ¦ ¦
a - accidental # sharp______________________________________________¦ ¦ ¦ ¦
             ! flat                                                 ¦ ¦ ¦ ¦
          space none                                                ¦ ¦ ¦ ¦
                                                                    ¦ ¦ ¦ ¦
o - octave number L,0 to 6__________________________________________¦ ¦ ¦
                                                                    ¦ ¦
d - duration W whole________________________________________________¦ ¦
           H half                                                     ¦
           Q quarter                                                  ¦
           D triplet half                                             ¦
           E eighth                                                   ¦
           A triplet quarter                                          ¦
           S sixteenth                                                ¦
           B triplet eighth                                           ¦
           T thirty second                                            ¦
           C triplet sixteenth                                        ¦
                                                                      ¦
d - duration . dotted_______________________________________________¦
           : double dotted
```

For example a typical notation would read as:

```
3E 3Q.,3F 3Q:,3G 3H,3A#4H

voice 3, note E, no accidentals, octave 3, dotted quarter note
voice 3, note F, no accidentals, octave 3, double dotted quarter
voice 3, note G, no accidentals, octave 3, half
voice 3, note A sharp, octave 4, half     -
NOTE: Middle C is C 3.
```

2. For readability, several notes may be entered on one line with separating commas, but no extra spaces.
3. Any line with a '/' in the first position will be treated as a comment line and will not be processed, except for printing on the screen during compilation of the score data. Such lines are useful for marking bar numbers, etc.
4. Any line beginning with an '*' is a voice change command and has the following format:

```
                                                                *vnn
                                                               ¦ ¦ ¦ ¦
command indicator-----------------------------------------------¦ ¦ ¦ ¦
                                                                 ¦ ¦ ¦
voice number 0 to 5 ---------------------------------------------- ¦ ¦
                                                                   ¦ ¦
patch number 00 to 99 ------------------------------------------------ ¦
```

5. Any line starting with an '=' is a voice parameter change command and has the following format. This is a SixTrak machine specific function.

```
                                                          =vppnnn
                                                         ¦ ¦ ¦ ¦ ¦ ¦
parameter change indicator ------------------------------- ¦ ¦ ¦ ¦ ¦ ¦
                                                           ¦ ¦ ¦ ¦ ¦ ¦
voice number 0 to 5 ------------------------------------------ ¦ ¦ ¦ ¦ ¦
                                                             ¦ ¦ ¦ ¦ ¦
parameter number 00 to 35 ---------------------------------------- ¦ ¦ ¦ ¦
                                                               ¦ ¦ ¦ ¦
new value 000 - 001 to 000 - 127 ------------------------------------ ¦ ¦ ¦
  (depends on parameter range)
```

6. The end of the score is marked with an 'X' in the first position of a line.
7. A rest must be shown as octave L.
8. Note errors detected are reported in the following format:

```
e.g.
 Error number -4 in line number 123
 3E 3Q.,3F 3Q:,3G 7H,3A#4H
```

The error numbers relate to the character position in each note group, but there is no indication of which note group on the line is in error.

```
error number                position of unrecognized
                              character
                              voice number
    1                       note name
    2                       accidental
    3                       octave number
    4                       duration
    5                       duration dots
    6                       separation comma
    7
```

in a command line:

```
    1                       voic
e number
    2                       pro
gram number
```

in a parameter line:

```
    1                       voic
e number
    2                       para
meter number
    3                       para
meter value
```

I should point out, in particular, the subroutine called "itoax" in the listing for VOICE2. This is a routine to convert an integer value to an ASCII string and is usually called "itoa" in a normal C library. This function in my copy of the Software Toolworks C80 library is incorrect, so this new function is included here to make the conversions work properly.

The fifth function is the real time recording from the synthesizer. This is something I have experimented with on the H-89, but as yet I have not been able to make it work properly. The timing constraints and the clock interference have made it quite impossible, so far. Maybe someone will think of some bright new idea that will help. I look forward to hearing of any progress.

There are several enhancements that could be made to these programs to make them easier to use and to improve the functionality. For example, the format of the data file could be modified to include some sort of time code of the measure/beat form to allow for much more flexibility in defining note-on and -off command times. The music now has a somewhat "machine-like" quality that could be eliminated by slight adjustments in note timing. There has been proposed recently a standard format for MIDI sequencer files. If such a standard is adopted as part of the MIDI standard, these programs should be modified to match. In any case, they are offered as a starting point for fun and creation on the H-89 and on other machines with MIDI interfaces, as well.

One thought that has occurred and that I have played with is a method of combining fractals with music. Visual fractal patterns are now quite common in the computer literature and it seems that music produced using the same sort of semi-random processes could be very interesting indeed.

Figure 2

```
SATIE GYMNOPEDIE NO. 2
/SATIE GYMNOPEDIE NO. 2
/set voices 0 - 69,1to5 - 77
*069
*177
*277
*377
*477
*577
/1
0G 4H.
1R LQ,1G 3H
2R LQ,2E 3H
3R LQ,3B 3H
4R LH.
5G 1H.
/2
0A 5Q,0G 4Q,0F 4Q
1R LQ,1A 3H
2R LQ,2F 3H
3R LQ,3C 3H
4R LQ,4A 3H
5D 1H.
/3
0E 4Q,0F 4Q,0G 4Q
1R LQ,1G 3H
2R LQ,2E 3H
3R LQ,3B 3H
4R LH.
5G 1H.
/4
0D 4H.
1R LQ,1A 3H
2R LQ,2F 3H
3R LQ,3C 3H
4R LQ,4A 3H
5D 1H.
/END
XX
```

MS-DOS Machines

I have not said much about the use of MIDI with MS-DOS machines for the

main reason that I don't own one. There is a lot of commercial hardware and software available for these machines. For example, Roland makes what has become thus far the de facto standard interface, the MPU-401 (1), but there are other devices available. The circuitry described here could be easily adapted to operate in a standard 8088 computer. IBM has recently announced a device called the Music Feature that includes a MIDI interface and a multi-timbral synthesizer on a single plug-in board for their PCs. They may just have created another de facto standard! For more information on available hardware and software for these computers, see the June 1986 issue of Byte Magazine (2) or your local music store.

Well, that's it. There is a lot of detail in these three articles, and I will be the first to admit that it may take a fair amount of both hardware and software experience to duplicate my set up from the information in these pages. My hope is, however, that I have generated some interest and made it a little easier for the experimenters in the group to become involved with computers, MIDI and music.

I will be happy to send a copy of all the software listed here on a single-sided HDOS hard sectored, pre-formatted disk for $10 (give it a volume number over 150) or for $12 if I am to supply the disk. Just send your money order and/or disk to T. E. Thompson, 61 Sample Road, Ottawa, Ontario, CANADA, K1V 9T9.

References

(1) Byte Special Issue, Inside The IBM PCs, Vol.10, No.11, November 1985.

(2) Byte, Computers and Music issue, Vol.11, No.6, June 1986.

Listing 1

```
/* source for MUSIC6*/
#include stdio.h
main() /*MUSIC6 DATA COMPILER*/
{
/*main loop*/

  int f1,f2,lineno,ernote;
  char infile[16],outfile[16],s[80],c;
  cls();

  getname(infile);
  strcpy(outfile,infile);
  strcat(infile,".SCR");

  if (!(f1 = fopen(infile,"r"))) {
        printf("%s does not exist.\n",infile);
        return;
        }

  strcat(outfile,".DAT");
  if (!(f2 = fopen(outfile,"wb"))) {
        printf("%s cannot be opened.\n",outfile);
        return;
        }

  c='\377';          /*put fake HDOS header on data file*/
  putc(c,f2);
  c='\000';          /*377000 - abs file*/
  putc(c,f2);
  c='\200';          /*042200 - file start */
  putc(c,f2);
  c='\042';
  putc(c,f2);
  c='\000';          /*000000 - file length */
  putc(c,f2);
  c='\000';
  putc(c,f2);
  c='\200';          /*042200 - entry point */
  putc(c,f2);
  c='\042';
  putc(c,f2);

  readline(f1,s,80);
  lineno = 1;
  while (readline(f1,s,80) != -1)
    {
      lineno++;
      if (s[0] == '/'){          /*for an input comment line print it*/
          printf("%s\n",s);
          continue;
      }

      if (s[0] == 'X')          /* end of the input data*/
      break;

      if (s[0] == '='){         /*do a voice parameter change*/
        ernote = param(s,f2);
        if (ernote != 0) {
          printf("%d %s %d\n",lineno,s,ernote);
          errmes(lineno,s,ernote);
          break;
        }
        continue;
      }

      if (s[0] == '*'){         /*do a voice patch select change*/
        ernote = comand(s,f2);
        if (ernote != 0) {
          printf("%d %s %d\n",lineno,s,ernote);
          errmes(lineno,s,ernote);
          break;
        }
        continue;
      }
```

```
    ernote = decomp(s,f2);   /*otherwise do a note*/
      if (ernote != 0) {
         errmes(lineno,s,ernote);
         break;
      }
   }

 fclose(f1);
 fclose(f2);
}


cls() /*clear the screen*/
{
printf("\033E");
}


getname(f) /* get a file name*/
char f[16];
{
int l;
printf("\nEnter the name of the file to work with - ");
l=getline(f,17);
return(f);
}


readline(filenum,str,num) /*read a line from filenum to max of num chars*/
int filenum,num;         /*return -1 at EOF else return nmbr of chars*/
char str[];              /*read by end-of-line char*/
{

int i;
i=0;
while (i<num) {
    if ((str[i++]=getc(filenum)) == EOF)
       return(-1);
    if (str[i-1] == '\n'){
      str[i] = '\0';
      return(i-1);}
          }
return(i);
}


errmes(linenum,string,ernmbr)     /*print an error message for bad data*/
int linenum,ernmbr;
char string[];
{
   printf("Error number %d in line number %d.\n",ernmbr,linenum);
   printf("%s\n",string);
}


decomp(s,fnum)      /* decompose a note string and write the data*/
                 /* to fnum   - return 0 for OK, -n for error */
char s[];
int fnum;
{
int i,erf,another,done;
char m,time,voice,note;
erf = 0;
i=0;
another=1;
while ((s[i] != '\n') && (another == 1)) {

   switch (s[i++]){
      case '0': voice=1;break;
      case '1': voice=2;break;
      case '2': voice=4;break;
      case '3': voice=8;break;
      case '4': voice=16;break;
      case '5': voice=32;break;
      default: erf= -1;
   }
   switch (s[i++]){
      case 'A':note=1;break;
      case 'B':note=3;break;
      case 'C':note=4;break;
      case 'D':note=6;break;
      case 'E':note=8;break;
      case 'F':note=9;break;
      case 'G':note=11;break;
      case 'R':note=0;break;
      default: erf= -2;
   }
   switch (s[i++]){
      case ' ': m=note;break;
      case '#': m=note+1;break;
      case '!': m=note-1;break;
      default: erf= -3;
   }
   switch (s[i++]){
      case'L': m+=8;break;
      case'0': m+=20;break;
      case'1': m+=32;break;
      case'2': m+=44;break;
      case'3': m+=56;break;
      case'4': m+=68;break;
      case'5': m+=80;break;
      case'6': m+=92;break;
      default: erf= -4;
   }
   if(m==8) m=0;
   switch (s[i++]){
      case'C': time=2;break;/*REM TRIPLET SIXTEENTHS*/
      case'T': time=3;break;
      case'B': time=4;break;/*REM TRIPLET EIGHTHS*/
      case'S': time=6;break;
      case'A': time=8;break;/*REM TRIPLET QUARTERS*/
      case'E': time=12;break;
      case'D': time=16;break;/*REM TRIPLET HALVES*/
      case'Q': time=24;break;
      case'H': time=48;break;
      case'W': time=96;break;
      default: erf= -5;
   }
```

```
    another=0;
    done=0;
    switch (s[i++]){
       case',':another=1;
              done=1;
              break;
       case'.':time=(3*time/2);
              break;
       case':':time=(7*time/4);
              break;
       case'\n':i-- ;
              done=1;
              break;
       default: erf= -6;
    }

    if (done == 0){
     switch (s[i++]){
        case',':another=1;
               done=1;
               break;
        case'\n':i-- ;
               done=1;
               break;
        default: erf= -7;
     }
    }
    putc(voice,fnum);
    putc(time,fnum);
    putc(m,fnum);
  }
  return(erf);
}


comand(s,fnum)     /* enter a program change command and write the data*/
                 /* to fnum   - return 0 for OK, -n for error */
char s[];
int fnum;
{
  int erf,prognm;
  char t[3],prog,voice;
  erf = 0;

     switch (s[1]){
        case '0': voice=1;break;
        case '1': voice=2;break;
        case '2': voice=4;break;
        case '3': voice=8;break;
        case '4': voice=16;break;
        case '5': voice=32;break;
        default: erf= -1;
     }
  voice = voice+128;

  t[0] = s[2];
  t[1] = s[3];
  t[2] = '\0';
  prognm = atoi(t);
  if ( prognm < 0 || prognm > 99)
    erf = -2;
  prog = prognm;

  putc(voice,fnum);
  putc(prog,fnum);
  putc(0,fnum);
  return(erf);
}

param(s,fnum)      /* Enter a parameter change command and write the data*/
                 /* to fnum   - return 0 for OK, -n for error */
char s[];        /* This is very machine specific to the SixTrak*/
int fnum;
{
  int erf,paranm,parval;
  char t[3],num,voice,val;
  erf = 0;

     switch (s[1]){
        case '0': voice=1;break;
        case '1': voice=2;break;
        case '2': voice=4;break;
        case '3': voice=8;break;
        case '4': voice=16;break;
        case '5': voice=32;break;
        default: erf= -1;
     }
  voice = voice+192;

  t[0] = s[2];
  t[1] = s[3];
  t[2] = '\0';
  paranm = atoi(t);
  if ( paranm < 0 || paranm > 35)
    erf = -2;
  num = paranm + 2;    /*offset to SixTrax controller #*/

  t[0] = s[4];
  t[1] = s[5];
  t[2] = s[6];
  t[3] = '\0';
  parval = atoi(t);
  if ( parval < 0 || parval > 127)
    erf = -3;
  val = parval;

  putc(voice,fnum);
  putc(num,fnum);
  putc(val,fnum);
  return(erf);
}
```

Listing 2

```
        TITLE PLAY6 - A 6 VOICE MIDI MUSIC PLAYER PROGRAM
;       This program is written for Microsoft M80 in Z80 format
        .Z80
;
;       PLAYS FROM DATA FILES IN THE FOLLOWING FORMAT:
;       00ffffff        00=note + voice # flag
;       0ttttttt        note duration in MIDI tics
;       0nnnnnnn        note number 0-127
;  or
;       10ffffff        10=voice change + voice # flag
;       0nnnnnnn        new voice number 00-99
;       00000000        dummy byte
;
;  or   11ffffff        11=parameter change + voice # flag
;       0nnnnnnn        parameter controller # 2-37
;       0vvvvvvv        new value 0-1 to 0-127 as per parameter
;
A.NL    EQU     012Q
EC.EOF  EQU     001Q
UIVEC   EQU     201FH
MIDPRT  EQU     320Q        ;BASE ADDRESS OF MIDI I/F
;
USERFWA EQU     2280H       ;start address
;
EXT1    EQU     20A2H
EXT2    EQU     20A3H
;
SCALL   MACRO   TYPE        ;define HDOS System calls macro
        DEFB    377Q,TYPE
        ENDM
;
        ASEG
        ORG     USERFWA
;
BEGIN:  LD      A,0         ;INIT UART
        OUT     (MIDPRT+1),A
        LD      A,16
        OUT     (MIDPRT+4),A
        LD      A,128
        OUT     (MIDPRT+3),A
        LD      A,4
        OUT     (MIDPRT),A
        LD      A,0
        OUT     (MIDPRT+1),A
        LD      A,3
        OUT     (MIDPRT+3),A
        IN      A,(MIDPRT)
        LD      A,11
        OUT     (MIDPRT+4),A
;
        LD      HL,QUIT     ;SET UP TO GO TO QUIT SUBROUTINE ON CTL C
        LD      A,3
        SCALL   .CTLC
        XOR     A           ;RESET THE DISK INTERFACE
        OUT     (177Q),A
        LD      (EXT1),A
        LD      (EXT2),A
;
;
;
;
READER: LD      HL,PROMPT   ;GET FILE NAME TO PLAY
        SCALL   .PRINT
        LD      HL,CONBUF
READR1: SCALL   .SCIN       ;GET FILE NAME
        JP      C,READR1
        LD      (HL),A
        INC     HL
        CP      A.NL
        JP      NZ,READR1
        DEC     HL
        LD      M,0
        LD      HL,CONBUF
        LD      DE,DEFALT
        LD      A,1
        SCALL   .OPENR      ;OPEN THE FILE FOR READ
        JP      C,QUITERR
        LD      DE,BUFR
READR2: LD      A,1
        LD      BC,256
        SCALL   .READ       ;READ NEXT SECTOR
        JP      NC,READR2 ; IF NO ERROR CARRY ON
        CP      EC.EOF      ;WAS IT EOF?
        JP      NZ,QUITERR ; NO - BAD ERROR SO QUIT
        LD      A,1         ;YES - CLOSE THE FILE
        SCALL   .CLOSE
        JP      C,QUITERR
;
;       Main routine to play the data after it has been loaded into RAM
;
;
;
PLAYIT: LD      HL,(UIVEC+1)    ;GET RETURN ADDRESS OF CLOCK INTERRUPT
        LD      (CLKRET),HL     ;SAVE IT
        LD      HL,TICKER       ;SET UP TIC INTERRUPT
        LD      (UIVEC+1),HL
        LD      A,(.MFLAG)      ;SET CLOCK ENABLE BIT
        OR      00000001B
        LD      (.MFLAG),A
        LD      A,0F0H          ;SET UP DOUBLE MODE (a SixTrak Sysex message)
        CALL    PUTM            ; SysEx message F0 01 7B 00 F7
        LD      A,01H
        CALL    PUTM
        LD      A,7BH
        CALL    PUTM
        LD      A,00H
        CALL    PUTM
        LD      A,0F7H
        CALL    PUTM
PLAY1:  LD      A,0             ;start of play loop
        LD      (CHAN),A        ;checks each channel in turn for
        LD      IX,BASE0        ;a note to play
        CALL    CHECK
        LD      A,1
        LD      (CHAN),A
        LD      IX,BASE1
```

```
        CALL    CHECK
        LD      A,2
        LD      (CHAN),A
        LD      IX,BASE2
        CALL    CHECK
        LD      A,3
        LD      (CHAN),A
        LD      IX,BASE3
        CALL    CHECK
        LD      A,4
        LD      (CHAN),A
        LD      IX,BASE4
        CALL    CHECK
        LD      A,5
        LD      (CHAN),A
        LD      IX,BASE5
        CALL    CHECK
        JP      PLAY1
;
;
;
;CHECK  SUBROUTINE
;
;       ENTRY:  IX = BASEn
;               IX+0=VOICEn
;               IX+1=TIMEn
;               IX+2=DURATIONn
;               IX+3=NOTE NUMBERn
;               IX+4=NEXT DURATIONn
;               IX+5=NEXT NOTE NUMBERn
;               IX+6=POINTER TO NEXT LOW
;               IX+7=POINTER TO NEXT HI
;               IX+8=VOICE FLAG
;
CHECK:  LD      A,0
        CP      (IX+1)          ;IS BASIC TIMER DONE?
        JP      M,CHECK3        ;NO - GET OUT
        LD      A,(SPEED)       ;YES SET IT UP AGAIN
        LD      (IX+1),A
        DEC     (IX+2)          ;DECREMENT NOTE TIMER
        JP      NZ,CHECK3       ;NOT DONE YET
        CALL    NOTOFF          ;DONE - TURN NOTE OFF
        LD      A,(IX+4)        ;PUT THE NEXT NOTE IN PLACE
        LD      (IX+2),A
        LD      A,(IX+5)
        LD      (IX+3),A
        LD      A,0
        CP      (IX+3)          ;IS THE NEXT NOTE A 0?
        JP      Z,CHECK1        ;YES - IS A REST
        CALL    NOTEON          ;NO - TURN THE NOTE ON
CHECK1: CALL    GETNXT          ;GET ANOTHER FROM THE DATA
        JP      C,CHECK2        ;NONE AVAILABLE
        LD      L,(IX+6)        ;GET DATA POINTER
        LD      H,(IX+7)
        INC     HL
        LD      A,(HL)          ;GET NEW DURATION
        LD      (IX+4),A        ;SAVE IT
        INC     HL
        LD      A,(HL)          ;GET NEW NOTE NUMBER
        LD      (IX+5),A        ;SAVE IT
        INC     HL              ;POINT TO NEXT DATA
        LD      (IX+6),L        ;AND SAVE POINTER
        LD      (IX+7),H
        JP      CHECK3
CHECK2: LD      (IX+4),0        ;CLEAR OUT NEXT NOTE
        LD      (IX+5),0
CHECK3: RET
;
;
;
;GETNXT SUBROUTINE
;
;       ENTRY:IX = BASE POINTER
;
;       RETURN : CLEAR CARRY = FOUND NEXT NOTE
;                IX+6=POINTER TO THE NEW NOTE
;                IX+8=VOICE POINTER FLAG
;
;                SET CARRY = END OF DATA
;
;
GETNXT: LD      L,(IX+6)
        LD      H,(IX+7)
NXT1:   LD      B,(HL)          ;GET NEXT VOICE BYTE
        LD      A,B
        AND     (IX+8)          ;CHECK FOR CURRENT VOICE
        JP      Z,NXT2          ;NOPE
        SCF                     ;YES FOUND ONE
        CCF                     ;CLEAR CARRY
        JP      NXT4
NXT2:   LD      A,(HL)
        CP      0               ;ARE ALL FLAGS 0?
        JP      Z,NXT3          ;YES - GIVE UP
        INC     HL              ;NO - MOVE ON TO NEXT NOTE
        INC     HL
        INC     HL
        JP      NXT1
NXT4:   BIT     7,B             ;IS IT A COMMAND BYTE?
        JP      Z,NXT5          ;NO,GO PLAY IT
        LD      A,(CHAN)        ;YES, GET CHAN NO.
        ADD     A,176           ;ADD CONTROLLER OPCODE
        BIT     6,B             ;IS CONTROLLER BIT SET?
        JP      NZ,NXT6         ;YES,CARRY ON
        ADD     A,16            ;NO, MAKE A VOICE CHG CMD
NXT6:   CALL    PUTM
        INC     HL
        LD      A,(HL)          ;GET NEW PROG # OR CONTR #
        CALL    PUTM
        INC     HL
        LD      A,(HL)          ;GET NEW CONTR VALUE
        BIT     6,B             ;IS IT A CONTROLLER?
        JP      Z,NXT7          ;NO, CARRY ON
        CALL    PUTM            ;YES, SEND VALUE
NXT7:   INC     HL
        JP      NXT1            ;GET NEXT NOTE
NXT3:   SCF                     ;SET CARRY
NXT5:   LD      (IX+6),L        ;SAVE POINTER
```

```
        LD      (IX+7),H
        RET
;
;
;
;NOTEON SUBROUTINE
;
;       ENTER:IX+3=NOTE NUMBER
;
;
NOTEON: LD      A,(CHAN)        ;GET CHAN #
        ADD     A,144           ;ADD TO NOTE ON OPCODE
        CALL    PUTM
        LD      A,(IX+3)
        CALL    PUTM
        LD      A,64
        CALL    PUTM
        RET
;
;
;
;NOTOFF SUBROUTINE
;
;       ENTER:IX+3=NOTE NUMBER
;
;
NOTOFF: LD      A,(CHAN)        ;GET CHAN #
        ADD     A,144           ;ADD TO NOTE ON OPCODE
        CALL    PUTM
        LD      A,(IX+3)
        CALL    PUTM
        LD      A,0
        CALL    PUTM
        RET
;
;
;
;TICKER SUBROUTINE
;
;       ENTER : VIA 2mS CLOCK INTERRUPT
;
;
TICKER  EQU     $
        LD      HL,TIME0
        DEC     (HL)
        LD      HL,TIME1
        DEC     (HL)
        LD      HL,TIME2
        DEC     (HL)
        LD      HL,TIME3
        DEC     (HL)
        LD      HL,TIME4
        DEC     (HL)
        LD      HL,TIME5
        DEC     (HL)
        RET
;
;
```

```
;
;PUTM SUBROUTINE
;       PUT A BYTE TO THE UART
;
;       ENTRY   A=BYTE TO BE SENT
;
;
;
PUTM:   PUSH    AF              ;SAVE BYTE TO BE SENT
PUTM1:  IN      A,(MIDPRT+5)        ;   CHECK IF UART IS BUSY
        AND     040Q
        JP      Z,PUTM1         ;  YES, CHECK AGAIN
        POP     AF              ;NO, SEND BYTE
        OUT     (MIDPRT),A
        RET
;
;
;
;QUIT SUBROUTINE
;
;
;
;
;
QUITERR:SCALL    .ERROR
QUIT:   LD      HL,(CLKRET)     ;REMOVE CLOCK VECTOR
        LD      (UIVEC+1),HL
        LD      A,(.MFLAG)      ;DISABLE CLOCK SERVICE
        AND     11111110B
        LD      (.MFLAG),A
        XOR     A               ;GRACEFUL EXIT
        SCALL    .EXIT
;
;
;
;DATA AREA
;
;
;
SPEED:  DEFB    12              ;BASIC TIMER(2 mS TICS) AT 24/QTR NOTE
CHAN:   DEFB    0               ;CURRENT CHANNEL #
CLKRET: DEFW    0               ;CLOCK SERVICE RETURN VECTOR SAVE
;
;
;VOICEn DATA :  IX = BASEn
;              IX+0=VOICEn
;              IX+1=TIMEn
;              IX+2=DURATIONn
;              IX+3=NOTE NUMBERn
;              IX+4=NEXT DURATIONn
;              IX+5=NEXT NOTE NUMBERn
;              IX+6=POINTER TO NEXT LOW
;              IX+7=POINTER TO NEXT HI
;              IX+8=VOICE FLAG
;
;
BASE0   EQU     $
VOICE0: DEFB    0
```

```
TIME0:  DEFB   1
DUR0:   DEFB   1                        ;VOICE 0 PARAMETERS
NOTE0:  DEFB   0
NDUR0:  DEFB   1
NNOTE0: DEFB   0
POINT0: DEFW   BUFR+8
VFLAG0: DEFB   1
;
BASE1   EQU    $
VOICE1: DEFB   0
TIME1:  DEFB   1
DUR1:   DEFB   1                        ;VOICE 1 PARAMETERS
NOTE1:  DEFB   0
NDUR1:  DEFB   1
NNOTE1: DEFB   0
POINT1: DEFW   BUFR+8
VFLAG1: DEFB   2
;
BASE2   EQU    $
VOICE2: DEFB   0
TIME2:  DEFB   1
DUR2:   DEFB   1                        ;VOICE 2 PARAMETERS
NOTE2:  DEFB   0
NDUR2:  DEFB   1
NNOTE2: DEFB   0
POINT2: DEFW   BUFR+8
VFLAG2: DEFB   4
;
BASE3   EQU    $
VOICE3: DEFB   0
TIME3:  DEFB   1
DUR3:   DEFB   1                        ;VOICE 3 PARAMETERS
NOTE3:  DEFB   0
NDUR3:  DEFB   1
NNOTE3: DEFB   0
POINT3: DEFW   BUFR+8
VFLAG3: DEFB   8
;
BASE4   EQU    $
VOICE4: DEFB   0
TIME4:  DEFB   1
DUR4:   DEFB   1                        ;VOICE 4 PARAMETERS
NOTE4:  DEFB   0
NDUR4:  DEFB   1
NNOTE4: DEFB   0
POINT4: DEFW   BUFR+8
VFLAG4: DEFB   16
;
BASE5   EQU    $
VOICE5: DEFB   0
TIME5:  DEFB   1
DUR5:   DEFB   1                        ;VOICE 5 PARAMETERS
NOTE5:  DEFB   0
NDUR5:  DEFB   1
NNOTE5: DEFB   0
POINT5: DEFW   BUFR+8
VFLAG5: DEFB   32
;
PROMPT: DEFB   'What file would you like to play','?'+200Q
DEFALT: DEFB   'SYODAT'
CONBUF: DEFS   80
BUFR:   DEFB   0,0,0
        DEFB   0,0,0
        DEFB   0,0,0

        END    BEGIN
```

Listing 3

```
/* VOICE2.C source for the Sequential SixTrak voice patch librarian*/

#include stdio.h
#define port 208        /*base address for MIDI port*/
#define stat 213        /*MIDI port status register address*/


main() /*VOICE Sixtrak voice editor*/
{
/* set up synth port for 32.5 kbps 8 bit no parity*/
       int byte;
       char c;
       cout(port+1,0);
       cout(port+4,16);
       cout(port+3,128);
       cout(port,4);
       cout(port+1,0);
       cout(port+3,3);
       cinp(port);
       cout(port+4,11);
/*main loop*/
       domenu();
       while (( c = tolower(getchar())) != EOF)
       {
         switch (c) {
         case 's': save();
                   break;
         case 'l': load();
                   break;
         case '\n': break;
         default: printf(''Legal choices are S L or ^D\n'');
                  domenu();
                  break;
         }
       }
}

cinp(portno)
{#asm
       POP    B
       POP    D
       LXI    H,$+5
       MOV    M,E
       IN     0
       MOV    L,A
       PUSH   D
       PUSH   B
```

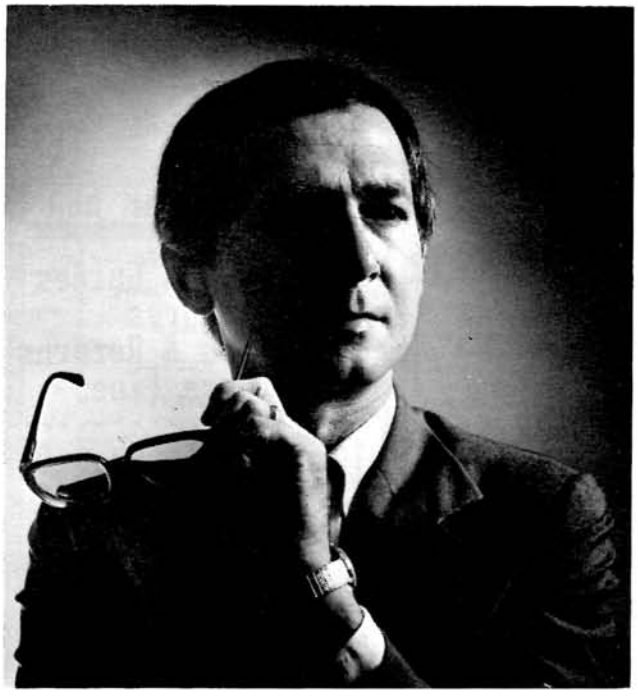
```
#endasm
}

cout(portno,byte)
{
#asm
        POP     B
        POP     D
        MOV     A,E
        POP     D
        LXI     H,$+5
        MOV     M,E
        OUT     0
        PUSH    D
        PUSH    D
        PUSH    B
#endasm
}

save()
{
int startn,endn,fl,i,n;
char filname[16],s[7],c,*itoax(),v[33],*getvoice();
c=getchar();
cls();
printf(''\nSave voice parameters from synth to disk\n'');

do{
  printf(''Enter the numbers of the first and last voices to be saved\n'');
  printf(''Range is 00 to 99\n'');
  (scanf(''%d,%d'',&startn,&endn));
  } while (startn < 0 || endn > 99 || startn > endn);

getname(filname);
if (fl = fopen(filname,''u'')) {
        fclose(fl);
        printf(''That file already exists.\n'');
        return;
        }
if (!(fl = fopen(filname,''w''))) {
        printf(''That file cannot be opened.\n'');
        return;
        }

itoax(startn,s);
fprintf(fl,''%2s\n'',s);

itoax(endn,s);
fprintf(fl,''%2s\n'',s);

for (i=startn; i <= endn; i++)
   {
   getvoice(i,v);
   wrtvoice(v,fl);
   }

fclose(fl);
domenu();
}
```

```
load()
{
int fl,i;
int startn,endn,lstartn,lendn,voicen,maxn;
char filname[16],s[7],c,v[32],*rdvoice();
c=getchar();
cls();
printf(''\nLoad voice parameters from disk to synth\n'');

getname(filname);

if (!(fl = fopen(filname,''r''))) {
        printf(''That file does not exist.\n'');
        return;
        }

startn = readnm(fl);
endn = readnm(fl);

printf(''File %s has voices from %d to %d.\n'',filname,startn,endn);

do{
  printf(''Enter the numbers of the first and last voices to be loaded\n'');
  printf(''Range is %d to %d\n'',startn,endn);
  (scanf(''%d,%d'',&lstartn,&lendn));
  } while (lstartn < startn || lendn > endn || lstartn > lendn);

printf(''Load range is %d to %d.\n'',lstartn,lendn);
seek(fl,(((lstartn-startn) * 32 )+6),0);
maxn = 99-(lendn-lstartn);

do{
  printf(''Enter the destination voice number of the first voice\n'');
  printf(''Range is 0 to %d\n'',maxn);
  (scanf(''%d'',&voicen));
  }while ( voicen > maxn );

for (i=lstartn; i <= lendn; i++)
   {
   rdvoice(v,fl);
   putvoice(voicen++,v);
   }

fclose(fl);
domenu();
}


readnm(fl)  /*read the next number from the file fl*/
int fl;
{

int i,num;
char s[3];
i = 0;
do  s[i++] = getc(fl);
    while (s[i-1] != '\n');
s[i-1] = '\0';
num = atoi(s);
```

```
return num;
}

domenu()     /*print the main menu*/
{
cls();
printf(''SIXTRAK VOICE EDITOR\n\n'');
printf(''[S]ave voices from synth to disk\n'');
printf(''[L]oad voices from disk to synth\n'');
printf(''\n\nEnter your choice or ^D to quit\n'');
}


cls() /*clear the screen*/
{
printf(''\033E'');
}


getname(f) /* get a file name*/
char f[16];
{
int l;
printf(''\nEnter the name of the file to work with - '');
l=getline(f,17);
return(f);
}


char *itoax(n,s) /* convrt n to chars in s */
char s[];
int n;
{
      static int c, k;
      static char *p, *q;

      if ((k = n) < 0)     /*record sign*/
            n = -n;
      q = p = s;
      do {              /* generate digits in reverse order*/
            *p++ = n % 10 + '0';
      } while ((n /= 10) > 0);
      if (k < 0) *p++ = '-';
      *p = 0;

      while (q < --p) {  /*reverse string in place*/
            c = *q; *q++ = *p; *p = c; }
      return (s);
}


char *rdvoice(v,filenum)  /*read a set of 32 voice bytes from filenum*/
char v[];
int filenum;
{
int n;
      for ( n=0; n<=31; n++)
        v[n] = getc(filenum);
      return(v);
}


char *getvoice(nmbr,v)   /*get 32 voice bytes from synth voice nmbr*/
char v[];              /*and save in string v*/
int nmbr;
{
char c;
int n;
      midio(0xf0);     /*sysex message F0 01 00 nmbr f7*/
      midio(0x01);
      midio(0x00);
      midio(nmbr);
      midio(0xf7);
#asm
      DI
#endasm
      c = midii();     /*dump sysex header & read data*/
      c = midii();
      c = midii();
      c = midii();
      for ( n=0; n <= 31; n++)
         {
         v[n] = (midii() | '\100' ) ;
         }
      c = midii();
#asm
      EI
#endasm
      return (v);
}


midii()  /*get a byte from the midi port - wait status*/
{#asm
MIDII1: IN     stat
      ANI    001Q
      JZ     MIDII1
      IN     port
      MOV    L,A
#endasm
}


midio(data) /* output a byte to midi port when ready*/
{#asm
MIDI01: IN     stat
      ANI    040Q
      JZ     MIDI01
      POP    B
      POP    D
      MOV    A,E
      PUSH   D
      PUSH   B
      OUT    port
#endasm
}

wrtvoice(v,filenum)   /*write 32 voice bytes to filenum*/
```

```
char v[];
int filenum;
{
int n;
       for ( n=0; n <= 31; n++)
          putc(v[n],filenum);
}


putvoice(nmbr,v)  /*put 32 voice bytes from string v to voice nmbr*/
char v[];
int nmbr;
{
char c;
int n;
       midio(0xf0);       /*sysex message F0 01 05 nmbr data F7*/
       midio(0x01);
       midio(0x05);
       midio(nmbr);

       for ( n=0; n <= 31; n++)

           midio(v[n] & '\017' ) ;

       midio(0xf7);
}
```

*



Mainstream Computing

by Joseph Katz

What you see here is a glimpse of what you can get from Xerox Ventura Publisher 1.1, desktop publishing software that's a dream come true for preparing document oriented publications on your Heath or Zenith computer. I did this column on my Z-248.

Xerox Ventura Publisher 1.1 is the reigning monarch of document-oriented page layout programs for desktop publishing on IBM compatible computers.

Although Ventura Publisher most certainly will do single sheet advertisements and other page-oriented publications, it is peerless in the realm of books, magazines, manuals, and other long, complex, multi-part publications, especially those that need revision after typesetting. There, in that realm where the rigidity of other desktop publishing programs can make you want to join the Luddites and do bad things to machines, Ventura Publisher lets you get the work out and make you think good thoughts about computers.

The controlling metaphors in Ventura Publisher relate to the area of book design and production. For example every "publication" consists of at least one "chapter." In the world of Ventura Publisher a chapter is a discrete entity composed of one or more text files imported from word processing programs, picture files imported from paint programs, draw programs, or scanners, and control files generated by Ventura Publisher when you tell it to associate those other files into a chapter. It's the relationship among all the source files you supply, recorded in the control files Ventura Publisher supplies, that determines the meaning of "chapter" within a Ventura Publisher "publication."

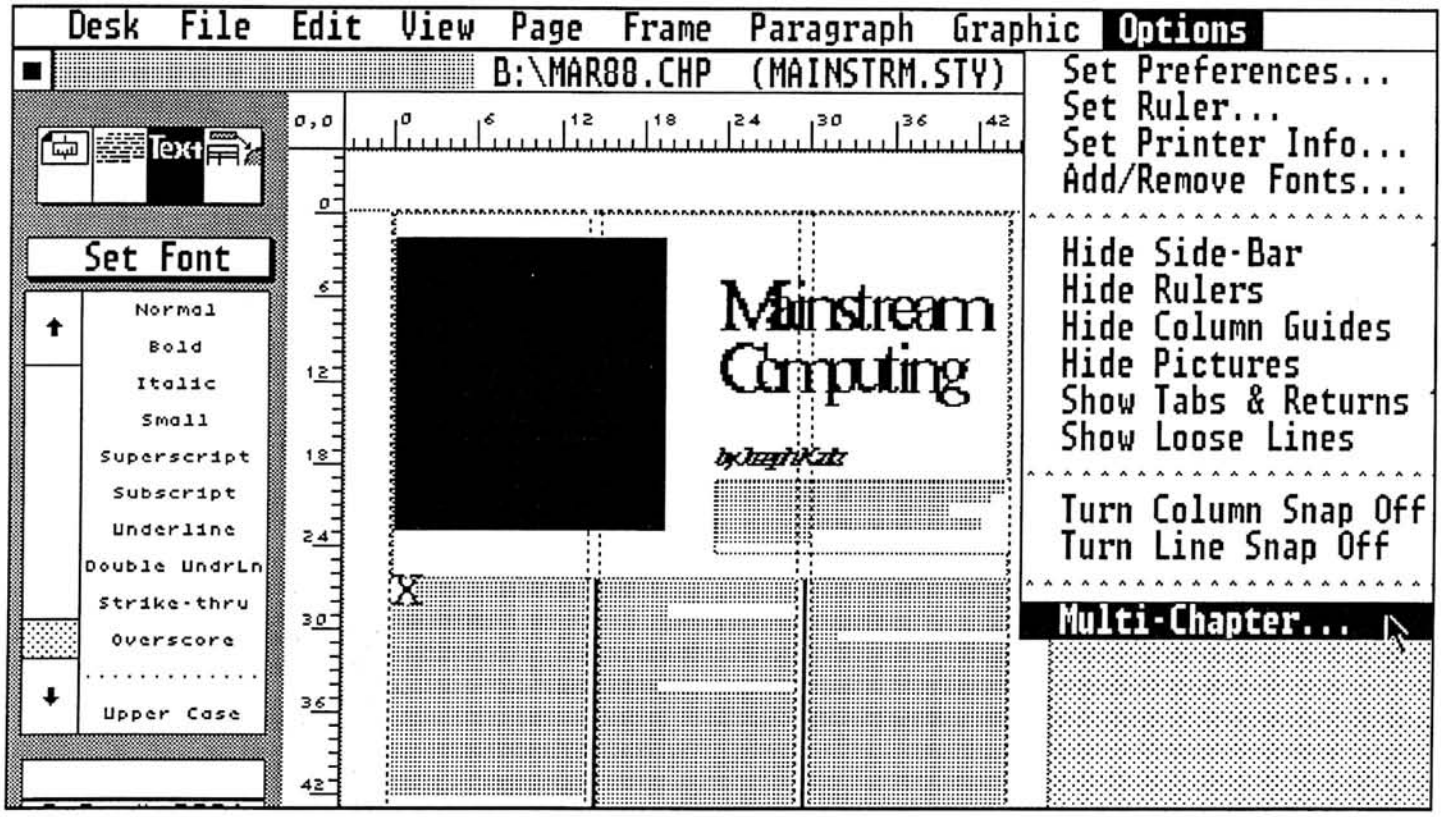
Publications may be from 1 through 9,999 individual pages apportioned among from 1 through 128 individual chapters. Actual chapters are linked together when you instruct Ventura Publisher to print the publication.

That's a computer simulation of the bookmaking process in real life. The manuscript of a book or magazine is a messy stack of heterogeneous papers first given an appearance of homogeneity when it's set in type according to the publisher's "style sheet" specifying the design for that publication. The chapters (magazine articles and any other subdivisions of a "publication" are Ventura Publisher "chapters" too) can be redistributed and restyled for reuse in other publications. Ventura Publisher lets you do the same thing and more easily. Just begin another chapter, most likely but not necessarily for another publication, and import the text and picture files you want in it.

The essential point is that the relationship among elements in a chapter and publication exists only as pointers maintained in Ventura Publisher's control files. Those elements—the text and picture files—therefore remain outside Ventura Publisher where they lead a double life. They are independent files and components of Ventura Publisher chapters, which in turn are linked into Ventura Publisher publications.

One advantage of this approach is that Ventura Publisher requires less space for files than other desktop publishing programs: it doesn't duplicate internally the original content of each imported text file and it doesn't have to use real

Please address all correspondence to me at 103 South Edisto Avenue, Columbia SC 29205. I'll try to answer letters accompanied by a self-addressed stamped envelope, but my volume of mail is too heavy for me to promise. I'll assume the right to publish your letter (edited, if I think that appropriate).



The Multi-Chapter option defines publications.

or virtual memory to keep track of enormous publication files produced by those other programs. So Ventura Publisher is fast. It's fast enough to be usable on an XT compatible, such as a Zenith Z-158. Although you'll prefer an AT compatible, such as a Zenith Z-248, Ventura Publisher really doesn't require it. What it does require is a graphics card and monitor (CGA works, but you'll want Hercules Graphics, EGA, or VGA) but not in color (Ventura Publisher doesn't use the color display except if you have a color printer), a hard disk drive, a mouse, and 640 KB of memory. Ventura Publisher—like other professional desktop publishing programs—eats memory, so forget Ventura Publisher's stated "minimum requirement" of 512 KB RAM. You want the maximum.

Another advantage to Ventura Publisher's approach is that any text editing performed within the program is recorded in the text file itself. Since other programs soak in the original content of each imported text file, it's the import that's edited and not the source text file. When text must be revised, therefore, your choice is to make the changes twice or allow the publication to depart from the text files used in its creation. That's Hobson's Choice. Ventura Publisher doesn't force you to make it.

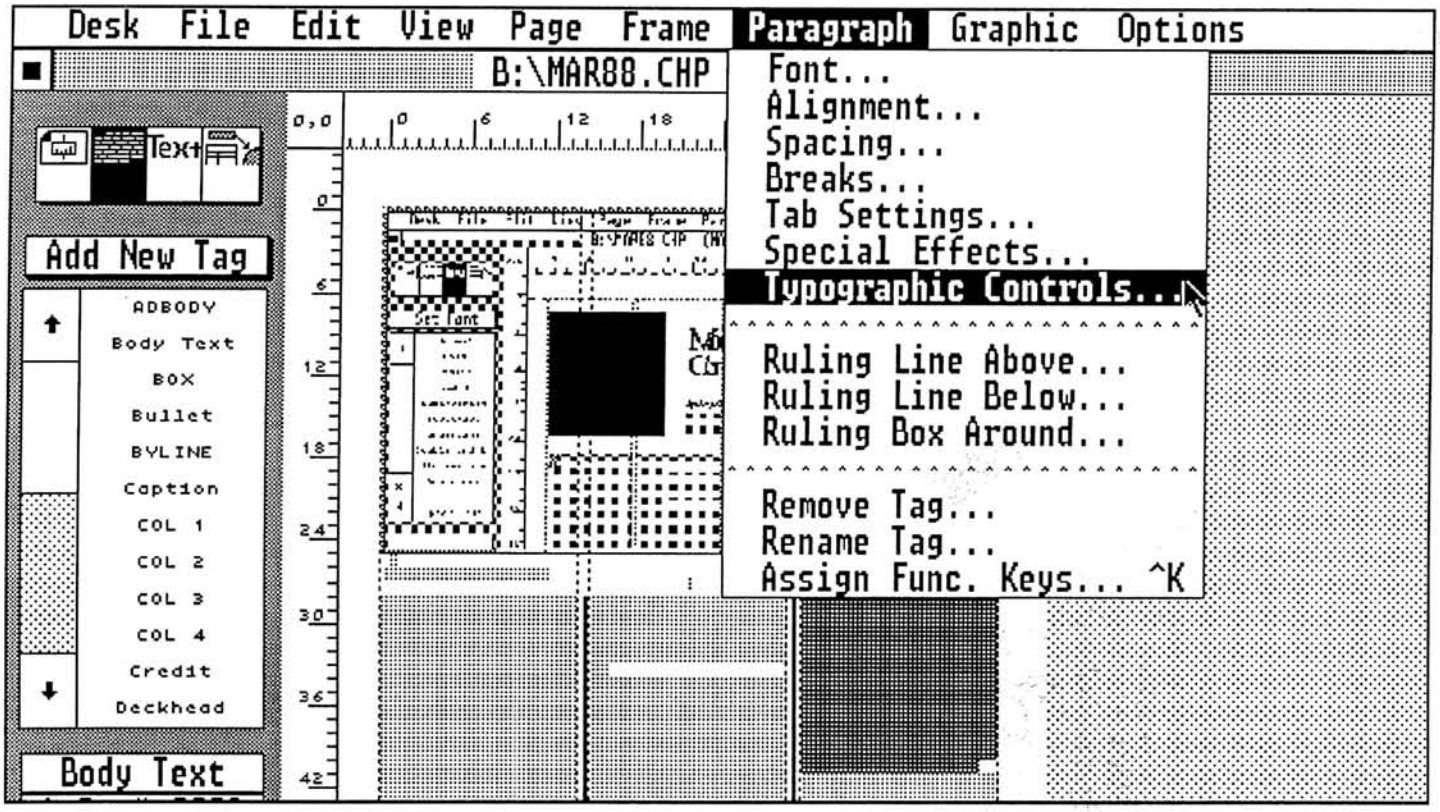
Still another advantage to Ventura Publisher's way of handling things is ease of editing and revision. If changes must be made to the text, you do it with a word processing program in the text file itself. Indeed you can do it from with within Ventura Publisher by going into text mode, placing the cursor where changes go, then typing your little heart out. And that's a useful practice for small changes. But big changes and wholesale revisions can be made more easily with a word processing program in the text file itself. Ventura Publisher itself doesn't care which way you do things.

Let's pause for a moment to look at the component files Ventura Publisher can import. Ventura Publisher 1.1 directly supports many of the major programs that produce text and pictures.

Text can be in the form of ASCII files, but Ventura Publisher does understand text attributes (such as boldface and underlining) in files from the following word processing programs: DisplayWrite III and DisplayWrite IV (and any other program that produces DCA files), Microsoft Word, Microsoft Windows Write, Multimate, WordPerfect, WordStar, Xerox Writer, and XyWrite. We're talking about supported text attributes, not styling: you can't expect to import a text file from Microsoft Word, for example, and have its style sheet translated into Ventura Publisher's.

Ventura Publisher directly imports "line art" (the program's perfectly reasonable designation of object-oriented graphics from draw programs) from AutoCAD (and any other program that produces SLD slide files), GEM Draw and GRAPH, Lotus 1-2-3 (and any other program that produces Lotus PIC files), MacDraw (and any other program that produces Macintosh PICT files), VideoShow, CGM, Mentor Graphics, HPGL, and any program that produces Encapsulated PostScript Files ("EPSF").

"Image" formats (the program's designation for bitmapped graphics from paint



At the left is the list of tags in the style sheet.

programs) that can be directly imported into Ventura Publisher are those from GEM Paint and Dr. HALO 2.0, MacPaint, and PC Paintbrush (and any other program that produces PCX files).

What Ventura Publisher 1.1 doesn't import directly by a menu option it can support indirectly with utility conversion programs. Two presently come with the package. DXFTOGEM converts a DXF ("Drawing Interchange Format") file to a GEM file that can be directly imported into Ventura Publisher. TXTOPCX converts an ASCII text file to PC Paintbrush PCX format. There's also supposed to be a third program, to convert Microsoft Windows object-oriented files, but it's not yet available.

Ventura Publisher's metaphor of book production in the real world continues. Just as in that real world, the relationship among elements in a Ventura Publisher chapter is determined in large part by a "style sheet," which specifies how each element is to be treated typographically.

A primitive book, for example, might have only chapter numbers, chapter titles, chapter subtitles, and body text as its major elements. A designer working in traditional ways would write a list of specifications for the typeface, style, size, leading, alignment, and other features to be used for each of those four elements. Assuming the specifications for body text as the default, the other three elements would be assigned some kind of "tag"—perhaps "A" for chapter numbers, "B" for chapter titles, and "C" for chapter subtitles. Then the conscientious designer would go through the book's manuscript, tagging each element that should be set differently from the default body text, and the typesetter would follow the designer's style sheet to produce what was specified.

You'll immediately recognize three advantages in this method of translating a design to type. One advantage is that tags save a great deal of time: there's much less labor involved in putting "B" next to the chapter headings than in marking each of them with something like "Bodoni 10/12 X 24-1/2 centered visually" or, worse, the gibberish used in contemporary electronic composition systems. Another advantage is that tags save mistakes: it's hard to miswrite or misread "A," "B," and "C," but easy to think of one face or size instead of another. The biggest advantage, though, is that new decisions about the book's design call for nothing more than a change of style sheet and not a remarking of the entire manuscript.

That's the way Ventura Publisher works. Any "paragraph"—by which Ventura Publisher means any text separated from any other text by a carriage return and a line feed—can be "tagged." You can use tags such as "A," "B," and "C," but Ventura Publisher allows much more meaningful identifications such as "Number," "Chapter Title," "Subtitle," and "Body Text." If you work with an existing style sheet, you simply read it into the chapter before or after you import the text files. (It's associated with them at the time you save the chapter to disk.) Then you go through the file tagging each element except the body text: as in the real world, Ventura Publisher assumes that any text element that isn't anything else (because it isn't tagged as

anything else) is body text. As you work along, the display changes to show you the text conforming to the style for each tag.

Tags, as you realize, are really macros, similar to those in spreadsheets like Lotus 1-2-3, in keyboard redefinition programs like Alpha Software's Keyworks, and in word processing programs like XyWrite III Plus, only infinitely more sophisticated. For example you can develop a library of separate but coordinated style sheets that give different typographical treatment to the same tags, defining "Number," "Chapter Title," and "Subtitle" in widely divergent ways. Then all you do to convert an existing publication to a completely different format is associate it with a different style sheet. The change is instantaneous and you see it on the computer screen. Or, to switch directions for a moment, you can shuffle tags in the text file and wind up with a different format from the same style sheet. Because the tags are inserted in the text file itself, it's an easy matter to use the global search and replace feature of a word processor for making the changes. Or, to go in yet another direction, you can insert the tags in the text file yourself instead of having Ventura Publisher do it for you. That way the document takes shape instantly as soon as the text file is imported into Ventura Publisher. It's a useful procedure because one inconvenience in Ventura Publisher is that it does not automatically translate typewriter characters such as "--" into typographical characters such as "—": within Ventura Publisher you must change each character manually, one by one. It's easier to make the translation with a word processing program.

Ventura Publisher runs in a subset of Digital Research's GEM environment. There are tradeoffs. Ventura Publisher is faster than it would be if it ran under Microsoft Windows and can use the excellent libraries supplied with Digital Research's own GEM applications (including snazzy borders and symbols from GEM Draw). Unfortunately GEM does not have the popularity of Windows and so Digital Research is the only major vendor I know that supplies GEM application software. One major difference between GEM and Ventura Publisher's adaptation is in the vastly superior handling of everything related to typography that Ventura Publisher has contributed. Instead of the two choices GEM gives you for typefaces—"Dutch" and "Swiss" as they're called—on a PostScript printer, for example, Ventura Publisher lets you use any font installed in or downloadable to your laser printer. Ventura Publisher 1.1 supports the Apple LaserWriters and other PostScript printers, Hewlett-Packard LaserJets, Hewlett-Packard LaserJet Plus, JLaser, AST TurboLaser, Cordata, Xerox 4045, and Interpress. There's support too for the Xerox 4020 color ink jet printer and Epson dot matrix printers. There's also supposed to be a driver to support Hewlett-Packard's DDL, but it's not yet available.

Ventura Publisher 1.1 has the best control of typography I've seen so far in a desktop publishing program. Its default handling of such typographical elements as kerning, tracking, and spacing is nice and tight, which suits my taste. If your tastes are different, a menu option allows you to adjust any typographical element. Taste is not the only consideration. Ventura Publisher allows subtle typographical adjustments that can make shrink or stretch the available copy to fill the available space. Ventura Publisher's typographical controls ought to be a standard feature in all desktop publishing programs.

Ventura Publisher 1.1 is a professional production tool for serious desktop publishers. It is one of the pinnacles in desktop publishing today. It's therefore also a complex program. Expect a steep learning curve, especially if you haven't already had experience in book design and production. Expect too to spend time working through the excellent *Training Guide* and example publications (which have style sheets you can adapt to your own jobs). The documentation for Ventura Publisher 1.1 is nearly superb. It needs to correct frequent misspellings that matter (such as "Logitek LogiMouse"), correct occasional misinformation that matters (such as the flat instruction to install a Logitech LogiMouse as a PC Mouse compatible when you should install the C7 mouse as one needing MOUSE.COM, which it does), and add detailed technical information that matters (such as which filenames contain each driver and which options can be used in command lines to avoid having to redo the installation process when installing a new monitor or mouse). But notice the level of my criticisms. Those are the kinds of minor things one picks at in a giant of a product.

Product Information

Xerox Desktop Publishing Series: Ventura Publisher Edition 1.1. $895.
Xerox Corp.
1301 Ridgeview Drive
Lewisville, TX 75067
214/436-2616

*

A Tip: Ventura Publisher 1.1 and The LaserWriter Connection or *Blitz*

If you use the printer interface programs in The LaserWriter Connection or ***Blitz***, you can simplify installation of Ventura Publisher 1.1 by telling it that your printer is on LPT1 instead of COM1 or COM2.

In fact that procedure is essential if you're using ***Blitz***. The reason is that Ventura Publisher 1.1 tries to be helpful by monitoring output to serial printers. Unfortunately it doesn't know that there's software like ***Blitz*** to make LaserJets and LaserWriters run at 19200 Baud, so Ventura Publisher can't figure out what's happening. When you install Ventura Publisher for LPT1—the standard printer port—responsibility for controlling printer traffic shifts to ***Blitz*** and Ventura Publisher feels confident about sending printer output as fast as possible. — *Joseph Katz*

On The Leading Edge

William M. Adney
P. O. Box 531655
Grand Prairie, TX 75053-1655

Disk Access Time, Z-100 Hard Disks, Zenith OS/2, DSBACKUP+, DS Manager, DS Tutor

If the letters I receive are any indication, many of you have a significant interest in hard disks of one kind or another. I appreciate those letters since it gives me a good idea of your interests, and much of what I include in this column is based on your questions. I have mentioned high-speed hard disks (e.g. 30 ms or so) before, but I haven't explained what that really means. And these high-speed hard disks are more expensive to the tune of usually a $100 premium or so, and one wonders whether they are worth it. You'll see how the various factors combine to determine the disk access time in this article. I'll also mention specific hard disks that some Huggies have successfully implemented on their Z-100 systems.

I have also hidden some other goodies in this article that I think nearly everyone will find helpful. One of these is the fact that nearly every DOS book in the world tells you what a file name (i.e. filename.typ) is, but there is seldom any mention of how to name files on your system. At this point, I have worked with computers for over 20 years, and I have found that naming files is one of the more difficult chores that everyone has to do. You'll see some of the "tricks of the trade" that I think you will find helpful.

Disk Access Time

Many of you are trying to figure out whether or not to buy a hard disk and have discovered the quagmire of hard disk technology. There are many flavors of hard disks that include physical size, capacity, access time, and other considerations; but there is usually one thing you will see in many advertisements. It will generally be listed something like "30 MB (65 ms)" or "30 MB (30 ms), and the 30 ms disk is more expensive than the 65 ms disk. Is it worth it? Maybe -but let's take a look at the factors that influence the speed of a disk drive. And you will see how the speed of a disk drive can affect the overall performance of your system.

These factors are the same for both hard disks and floppy disks, but with one exception, there is usually little, or nothing, you can do about a floppy disk. Aside from the keyboard, the disk I/O is usually the slowest thing in your system, so you may want to see how each factor influences your system's performance. The values of these factors affect the overall speed of data movement (i.e. data transfer rate) from a disk. The factors are: rotational delay, data transfer time, and seek time.

ROTATIONAL DELAY is defined as the time it takes for the beginning sector of the data to be positioned under the read/write head assuming that the head is already positioned over the correct track or cylinder. If you know that a hard disk rotates at 3600 RPM, you can easily calculate the time for one rotation of the hard disk as follows:

$$\frac{60 \text{ seconds}}{\text{minute}} \times \frac{\text{minute}}{3600 \text{ revolutions}} = \frac{0.01666 \text{ seconds}}{\text{revolution}}$$

When you work with numbers in this order of magnitude, it is easier to convert them to milliseconds (ms -- thousandths of a second), and the above calculation shows that it takes about 16.7 ms for a single rotation of a hard disk. A standard 48 TPI (Tracks Per Inch) drive rotates at 300 RPM, and the high density drives used in the Z-200 and IBM AT rotate at 360 RPM. Most 8" disks also use 360 RPM too. Since a floppy disk spins at roughly one-tenth of the speed of a hard disk, you can

see that the time required for one rotation of a floppy is about TEN times longer than a hard disk. Now you know one reason why floppies are considerably more than a tad slower than a hard disk.

But rotational delay is an average time. Therefore, it is safe to assume that, on the average, it takes one-half of a revolution for the beginning sector to be positioned under the read/write head. As a result, one-half of the calculated value for one revolution is the rotational delay for any hard disk or about 8.4 ms. There is no way to change this unless the rotation speed of a floppy or hard disk is also changed. The rotation speed of a hard disk is also one reason that the interleave factor (see February 1988 REMark) is required. On a slower floppy disk, you will find that the interleave is 1, and technology has not sufficiently progressed to have an interleave higher than 2 on a hard disk.

The second factor is data transfer rate. DATA TRANSFER RATE is generally defined as the speed at which data moves between devices. It is usually measured in bits per second. In this discussion, data transfer rate is the speed that data are transferred to and from the system. The data transfer rate for your system is affected by all kinds of things. It includes the clock speed of the computer, the hardware in the disk drive and its internal electronics, and the electronics of the disk controller.

In most PC systems, the data transfer rate for a typical hard disk is 5 megabits per second. A standard 5.25" DS/DD 48 TPI floppy and controller has a data transfer rate of 250 kilobits (0.25 megabits) per second, and the 5.25" high density disks for the Z-200 have a data transfer rate of 500 kilobits (0.5 megabits) per second. The data transfer rate is a fixed value and generally depends on the capabilities of the disk controller as well as the hardware characteristics of the drive itself. A hard disk has a higher data transfer rate than a floppy drive, and this is one of the specific differences between a floppy disk controller and a hard disk controller, not to mention the physical drive characteristics.

Perhaps the most important is the last one. SEEK TIME is defined as the average time required to move the read/write heads to a specified track or cylinder. The actual value of the seek time depends on the starting track and the specified track for the seek. For example, it is obviously much faster to move the heads from cylinder 8 to cylinder 9 than it is to move the heads from cylinder 3 to cylinder 300 because of the physical distance involved.

A hard disk may be advertised with a track-to-track seek time that is a measure of movement from one track to another, such as from track 8 to track 9 which may be on the order of 3 ms or so. That isn't particularly helpful because the real measure of disk performance is the average seek time and this may range from 25 ms to 85 ms or so. Hard disks with faster seek times (i.e. smaller numbers) are more expensive because of the technology, and you will generally pay at least a $100 premium for the faster disks. These faster hard disks generally use the latest voice coil technology instead of the old stepper motors to improve the average seek time.

When you talk about disk access time, the total data movement time for a disk is based on the "sum" of these factors: the rotational delay, the data transfer rate, and the average seek time. You can't do much about the data transfer rate since that is essentially fixed by the hardware. The rotational delay time is also fixed at about 8.4 ms for a hard disk, so you can't do much about that either. Consider that the general range of seek times for a hard disk goes from 25-85 ms or some three to ten times greater than the rotational delay. That can make a considerable difference in the performance of your system, but the next part of the question is: "Will the performance improvement be worth the extra cost?"

Will a High-Speed Hard Disk Really Help?

The answer is a definite maybe. Although a high speed hard disk clearly is able to get to data faster than a slower one, the real improvement in your system is a matter of opinion. In other words: "How much faster will you see a screen display with a word processor using a fast or slow hard disk?" The answer is that you probably won't SEE much improvement with a high speed hard disk as compared to a slower one. And if you can't actually see the improvement, is the difference worth the extra cost? I don't think it is.

Part of this answer also depends on what specific application programs you are using. If you use your system primarily for spreadsheets and most word processing, I don't think the cost is worth it. For the most part, these programs work with data in memory, so you won't see any improvement except in possibly the time required to load and save a file. A fast hard disk might save as much as a second or two depending on the size of the file, but I don't think most people will see much difference.

On the other hand, if you use a program that has a lot of disk I/O--like a large data base application in dBase III-- you might find some significant time savings for your application that you can really see. Since the actual time savings is so dependent on the data base file size and the specific program you are using, it is not realistic to give a definite "yes" or "no", but data base applications will generally show the most improvement.

But there are even exceptions to that. For example, Borland's Reflex loads the entire file in memory, so there is little to be gained with a high speed hard disk except for the initial file loads and saves that I previously mentioned.

Whether or not you really need a fast hard disk is something that you will need to spend some time looking at based on the software that you use. In most cases, I would say that the speed advantage probably isn't worth the cost unless you have a specific data base problem to solve. If you have some extra money to spend when you get a hard disk, I think it is better to get some good utilities and backup software instead of getting a faster drive.

The Mace Utilities provides just about everything you'll need to maintain your hard disk, and DSBACKUP+ is excellent for taking hard disk backups. I use both packages a lot, and both are highly recommended. For most users, these packages will give you a much better return on your money than the speed will. The Mace Utilities includes the DEFRAG program to optimize your hard disk, and a utility to "unerase" deleted files. DSBACKUP+ is a much faster and more reliable way to backup your hard disk compared to the BACKUP/RESTORE commands provided with DOS.

Speed Up Z-100 Disk Access

If you have a Z-100, there is one little-known way you can improve the performance of your floppy disks. Use the CONFIGUR command to lower the values of the disk step rates for each of your floppy disk drives. For example, most

5.25" drives work just fine at a step rate of 6 ms which is the minimum value in the CONFIGUR command. Interestingly enough, all Z-100 MS-DOS versions that I have seen are shipped with this value set at the maximum of 30 ms. You can dramatically improve the performance of your system by setting this at the minimum value, and you can do the same for 8" drives too.

I guess this is a carry-over from the old H-89 systems because some of the early disk drives tended to be temperamental at low step rates. As far as I can recall, all versions of the Zenith operating systems for the Z-100 (including CP/M-85) were shipped at the maximum step rate, and CONFIGUR is about the same for all of the Zenith OS versions for the Z-100. This feature is not available, or necessary, in PC compatibles since the operating system is already set up for maximum performance by nearly all vendors of DOS.

Hard Disks on the Z-100

I received a lot of letters about the wide variety of hard disks that have been implemented on the Z-100. In the July 1986 REMark, I mentioned that I could not think of a reason why the high speed or "AT type" hard disks would not work on a Z-100. Since the real difference is just the seek time, my information indicated that any standard hard disk should work just fine on a Z-100 as long as you knew its characteristics (e.g. heads, cylinders, etc. for PREP) and had the appropriate mounting hardware. In any case, it appears that a lot of Huggies have successfully implemented these hard disks on their Z-100s.

Charles Hill (Tucson, AZ) successfully added a Seagate 4038 36 ms drive to a couple of Z-100s over two years ago. He notes that he has a Fujitsu 85 ms drive on another Z-100, and he can actually see a considerable performance difference because of the drive speed. He uses the standard Zenith Z-217 hard disk controller on these systems, but he mentions that a friend also successfully installed a 4038 with the CDR controller. Dr. Tim Oldfather (Hilo, HI) has also successfully installed a 4038 drive in his Z- 100 with the Z-217 controller.

Chuck Taylor (Glendale, AZ) installed a 30 ms Hiatachi with a 40 MB capacity with no problems. He said that everything works just fine with the standard Z-217 hard disk controller.

P. M. Fenton (MGM, AL) notes that he has a Seagate 6640 installed on his Z-100 with the CDR controller, but he could not get it to run with the Gemini Board. Although the hard disk runs fine in the normal Z-100 mode, it cannot be accessed in the IBM mode. Without going into all of the details, it appears that Gemini Technology developed a special ROM that partially solves the problem, but the Gemini Board still cannot talk to the IBM mode without writing a device driver. I am not surprised at the problem since the Gemini Board was designed to work only with Zenith upgrades such as the Z-217 controller, and the manual specifically states that. In addition, I believe that the Gemini Board was released before the CDR hard disk controller, so there wasn't much of a chance for compatibility in the design of the Gemini Board. Perhaps the most important point here is to be sure that you check for compatibility before you buy any hardware for your system. In many ways, the CDR controller is much better than the Z-217, but you need to be careful when mixing hardware. That is also important on the PC Series systems too.

Fred Smith (Fayetteville, GA) implemented a CMI 6640 hard disk that has a 40 MB capacity on his Z-100. He reports that the seek time is less than 40 ms, and he has found that the drive is reliable despite all of the problems that CMI drives have had in the IBM AT.

Dan Scott (Austin, TX) installed a 43 MB Miniscribe 6053 hard disk that has a 28 ms seek time in his Z-100. He mentions that PREP gave him a nasty error message because the drive was larger than the 32 MB Z-100 limit when the /K switch is not used. Many of these people mentioned some problems in bringing up the hard disk because they usually did not have the hard disk characteristics that are required by the Z-100 PREP. That brings up another point worth mentioning.

Hard Disks and ROM Upgrades

I recently received a letter from Nathan Crane (Tempe, AZ) asking about ROM upgrades and an MS-DOS upgrade from version 3.20 to 3.21 for his 248. In particular, he wanted to know if there was any particular advantage in replacing the ROM set for hard disk support, and new features of the latest Zenith release of MS-DOS 3.21. Both are interesting questions, so let's take a look at each one.

As far as replacing a ROM in a computer is concerned, I think it is a waste of money unless you know that a new ROM will fix a specific problem that you have. For example, my recollection is that you must have at least Z-100 ROM version 2.5 or greater to use a hard disk. In the PC Series computers, the only good reason to replace a Zenith ROM set is when you KNOW that a new ROM will fix some defined problem such as a PC compatibility problem with a specific program. For the Z-200 in particular, you need to have at least ROM version 1.9 to run Zenith's new OS/2 operating system. In most cases, it is not at all cost-effective to replace a ROM, and you really don't get any additional features that are something you can see. In short, if it ain't broke, don't fix it. By the way, the latest Z-200 ROMs are available as ZCA-10 for $24.80 as shown on page 104 of the Heath catalog #209.

I just received the latest Zenith release of MS-DOS 3.21 (BIOS 3.34), and there are certainly no obvious differences in terms of hard disk support. There are two additional device drivers: EMM.SYS for the Zenith expanded memory board only, and ZCACHE.SYS to create a cache for floppy/hard disk drives in memory. The other significant difference is that 3.21 also supports the 1.4 MB 3.50" drives in addition to the 720 KB 3.50" drive. Other than that, there is hardly any noticeable difference between the 3.20 version and the current 3.21.

For a hard disk, both versions support the PREP/Q command that supports all standard PC-type hard disks up to a physical maximum of 128 megabytes. The PREP/Q command (page 4.11 in my 3.20 manual) will prompt for all necessary parameters which will override the standard ones contained in the ROM. As a result, there is no need to update the ROMs to expand the hard disk capability. If you already have version 3.20, I think the best (and perhaps only) good reason to spend the money for 3.21 is to be able to use the new 1.4 MB 3.50" drives.

DSBACKUP and MS-DOS on the Z-100

There are an incredible number of tricky problems that can occur when you use a computer, and I finally was able to specifically identify one with some help from (and special thanks to) Hugh Kenner. The problem is that I've received reports that the ANSI version of DSBACKUP did not seem to work on some Z-100 computers, and all of the problems that I have been able to identify have been specifically the

result of using the PC version on a Z-100 as I reported in January. But there is another problem. I was surprised to learn that there are apparently a lot of people who still use a version 2 release of MS-DOS on the Z-100. Hugh reminded me of one of the "forgotten" facts which is that the version 2 ANSI driver is "crippled" and just about worthless for anything. The problem is that Zenith did not implement all of the usual ANSI escape sequences in that driver, so it does not work as one would expect. It appears, for example, that the cursor positioning sequence (e.g. ESC line;column H) was not implemented, so there is no standard way to position the cursor with the version 2 ANSI driver. Since all software designed to use an ANSI driver probably assumed that all of the usual sequences were implemented, you will most certainly have problems with that software. When I talk about the "usual" ANSI sequences, I mean the same ones that IBM has implemented in their ANSI driver supplied with PC-DOS.

Design Software's DSBACKUP program assumed that the usual ANSI sequences were implemented, and I found that it does not display the screen correctly with any Z-100 version 2 release that I have. In particular, the menu display is overwritten on the left side of the screen which is the reason I assume Zenith left out the cursor positioning sequence I mentioned. As you may recall from the January article, this is totally unrelated to the "Wild Interrupt" problem that I discussed.

It was interesting to check my computer log and find that even the original DSBACKUP testing was done nearly two years ago with MS-DOS version 3.10 on the Z-100. In any case, I strongly recommend that you get version 3.10 for your Z-100 while it is still available. Since all rules have their exceptions, this is one to the "If it ain't broke, don't fix it" approach.

New from Design Software

I just received the latest version of DSBACKUP+ (version 1.10A) from Design Software, and it keeps getting better. It is not, and never has been, copy protected which is one of the reasons that I like it. One of the new features is that you can interrupt a backup or restore at some point and then resume it later when you have more time -- a very nice feature. There is also a graphics display of the subdirectories on your hard disk which makes it extremely easy to selectively backup and restore them.

Is it fast? You bet! I was able to backup 12,106 kilobytes on drive C in 3 minutes and 44 seconds which was the total time required for the complete backup. That works out to just over 3 MB per minute which isn't too shabby. I did that primarily to compare the DSBACKUP+ capability with another product that I have seen that claims to be able to backup 10 megabytes in a few minutes, and I wondered if that was possible. Anything is possible (just about), so I performed the above backup to another hard disk partition which is the only way to attain that speed. There is no way to backup that much data in such a short elapsed time because you will normally be changing disks. I got around that problem by using a second hard disk partition which is one way to achieve that kind of speed. I hope Design Software will not advertise that because I think it is misleading. If you use DSBACKUP+ in the usual way to backup to floppies, you can backup the 12,106 kilobytes in just a tad over six minutes plus the time required to change floppies.

Design Software has a couple of other outstanding programs that are worth mentioning. If you're a newcomer to DOS, DS Tutor is an excellent way to learn about your computer system. It covers both hardware and software with excellent graphics displays as appropriate. It even has a glossary of computer terms that are used in the tutorial. DS Tutor is completely menu-driven, so it is quite easy to use.

The tutorial is detailed and I highly recommend it. For example, it even has a discussion of the interleave factor that I discussed last month. I think that the discussion of printers and ports will be quite helpful to a lot of people. And of course, there is a discussion of the basic DOS-related things that you really need to know to successfully operate your system.

Another new program is DS Manager which is actually a rather complete system management tool that has two primary features. First, you can create a multi-level menu set for all your applications software. This allows you to select Word Processing followed by a menu with WatchWord and WordStar.

The second feature is a DOS shell that allows you to copy, rename, move, and erase files. It is easy to use and the shell is probably worth the cost of DS Manager by itself. But there's more. DS Manager also provides some system security by using a password system that you can customize. In addition, the software also creates a log of system usage and provides reports so that you can see what is being used. All of these features are rather unique, and I found them quite easy to use.

If you are looking for some excellent software to help you with your system, these products are highly recommended.

In the Zenith News -- OS/2

Zenith began shipping their version of the new Microsoft OS/2 operating system last month (December 1987 actually) which was just shortly after IBM. IBM has called their version the OS/2 Standard Edition 1.0 which sells for $325 and does not include the Presentation Manager (an updated Windows). The Zenith OS/2 version is $26 cheaper ($299) than IBM's, and I'm glad to see that Zenith is pricing products a little more realistically. Perhaps this will continue for other products.

I have seen both the Zenith and IBM OS/2 versions, and I have to admit that Zenith did a dynamite job on theirs. The Zenith installation program is absolutely spectacular (and easy), and I think ZDS did an exceptional job on it. I have fooled around with the Zenith OS/2 a little bit, and I'll tell you some of my impressions that essentially apply to all OS/2 versions unless specifically noted.

OS/2 has two modes: the real mode and the protected mode. The real mode is the so-called DOS compatibility mode which means that you should be able to run all current DOS programs in that mode. The protected mode is used for special programs that have been written to run in the multitasking mode. That is, you cannot generally run today's current programs in the protected mode, and it will probably be some time before a reasonable selection of protected mode programs are available. All versions of OS/2 have some interesting requirements, and I have found some details about the Zenith OS/2 requirements.

Perhaps the most significant requirement is that you must have at least an 80286-based system (e.g. Z-200 or Z-386) in order to run OS/2. It simply will not run on a 150 series computer (or IBM PC) because it requires a minimum of the 80286 CPU. As far as I know, all OS/2 versions also re-

quire a hard disk. If you are using a Z-200 computer, you must also have at least ROM version 1.9.

The December 28, 1987 issue of Computer & Software News (page 17) also mentions that you must have at least 1.5 MB of memory, and I have learned that it must be extended, not expanded (i.e. EMS), memory. I predicted a couple of years ago in this column that new operating systems would require more memory and a hard disk, and it looks like that time has arrived.

For the most part, OS/2 looks quite similar to DOS as far as the user interface is concerned. In many cases, the commands are identical, although there are some differences in the way the commands work in the real mode versus the protected mode. Most of the differences are not a problem since most of the DOS command syntax has been preserved. I suspect that most DOS users will have little difficulty with the OS/2 commands.

All in all, I think Zenith did a very nice job on the implementation of OS/2, and I am looking forward to working with it.

Other News

As some of you may have noticed, I have started writing articles for PC Resource magazine on Zenith computers. My first article appears in the February 1988 issue. This is part of a column called the "Clone Panel Update" that includes columns from specialists in various brands of computer hardware. The purpose of this column is to discuss compatibility problems that are unique to each specific computer. If you own a Zenith PC series computer, you may find some of these topics helpful.

In the Mail

Bill Nabor wrote to me about a comment that I made in the December column about metal daisy wheels for a printer. He noted that sometimes the metal daisy wheels are hazardous to the health of your printer even if they are available. The problem is that the metal ones are heavier, and the print head unit may not be able to withstand the constant use of the heavier unit even though it's available for your printer. His advice is to be sure to check with the manufacturer of your printer to verify that the metal daisy wheel can be successfully used on your printer. I agree that you should always check with the hardware manufacturer to be sure that an accessory can be properly supported.

Powering Down

We have just gotten over the ice storms in Texas that hit in the last week. It's amazing what a little ice will do on the streets, but the good news is that there isn't much snow. I was raised in Indiana, and I walked to school in the snow. With a little luck, it will get warmer soon.

If you have any questions about anything in this column, be sure to include a self-addressed, stamped envelope (business size preferred) if you would like a personal reply to your question, suggestion or comment.

Products Discussed

DSBACKUP+	$79.95
DS Tutor	39.95
DS Manager	49.95
Design Software	
1275 W. Roosevelt Road	
Chicago, IL 60185	
(800) 231-3088 (Orders only)	
IBM OS/2	$325.00
IBM Product Centers	
Zenith Software	
OS/2	$299.00
Z-100 MS-DOS (OS-63-30)	150.00
MS-DOS 3.21 (OS-64-61)	149.00
MACE Utilities version 4.10	$99.00
Paul Mace Software, Inc.	
400 Williamson Way	
Ashland, OR 97520	
(800) 523-0258 (Orders only)	

*

Data Dating In A Spreadsheet

John B. Ellery
Flat 2, Mudeford Green
10 Mudeford Green Close
Christchurch, Dorset
ENGLAND BH23-359

For arranging dated events within a spreadsheet in chronological order, there are a number of simple devices which can be used. However, the measurement of time lapse events needs a finer routine.

Assume three cells hold the day, month and year in any order.

B6 = day of the month "dd"
C6 = month of the year "mm"
D6 = the year "yyyy"

Then, the formula D6*10000+C6*100+B6 will produce the number "yyyymmdd" against which comparisons can be made to test whether another similarly arranged date is earlier, the same or later. This is the process used in the following example.

Once the above has been entered, changing the Current Date will cause Cells C9 and B15 to alternate between the bonus figure and zero.

Julian Dates

When it comes to measuring a time lapse between dates, however, the process is a little more complicated. A system, used by banks, insurance companies, and so on employs the Julian Date. This converts the standard date into a total number of days. The difference between two such dates is a specific period in days.

To convert a standard date (mm dd yyyy), the Julian System calculates the number of days to the end of the previous year (yyyy-1), allowing for leap years, then the number of days to the end of the previous month (mm-1), adding a day if it's a leap year and the month is March or later, and finally, the days (dd) are added to the total.

The above process is demonstrated in the following example for both the Current Date and the Test Date.

Cells B11 and B12 calculate the days to the end of the previous year adding one day every fourth to allow for leap years.

```
   ¦  A  ¦¦  B  ¦¦  C  ¦¦  D  ¦¦  E  ¦¦  F  ¦¦  G  ¦
 1¦
 2¦          PERSONAL  ACCOUNTS
 3¦          1987    TO    1988
 4¦
 5¦           Day      Month      Year
 6¦            28        1      1988 = CURRENT DATE
 7¦
 8¦                    Bonus
 9¦       TEST 1:      335.50  i.e. If today;s date is
10¦                           the same as or later than
11¦                           the test date, credit bonus.
12¦       Test Code           Detail
13¦        19880128 TODAY;S DATE i.e. B5 TO D5
14¦        19871205 TEST DATE i.e. 5 DEC 1987
15¦             .00 OPPPOSITE OF TEST 1

Cell    Contents
B3  =  1987
C9  =  IF(D6*10000+C6*100+B6>=B3*10000+1200+05,335.50,0)
B13 =  D6*10000+C6*100+B6
B14 =  B3*10000+1200+05
B15 =  IF(D6*10000+C6*100+B6>=B3*10000+1200+05,0,335.50)
```

```
   ¦  A  ¦¦  B  ¦¦  C  ¦¦  D  ¦¦  E  ¦¦  F  ¦¦  G  ¦¦  H  ¦
 1¦
 2¦          PERSONAL  ACCOUNTS    Payment
 3¦          1987    TO    1988    Due:    $  168.52
 4¦          --------------------------    -------------------------
 5¦           Day    Month      Year
 6¦             3      1      1988 = CURRENT DATE
 7¦             5     12      1987 = TEST DATE
 8¦        Days to  Days in
 9¦        previous    given     Adjust for
10¦       year end:    year:      leap year
11¦         725751       3       0 = FOR CURRENT JULIAN DATE
12¦         725386     339       0 = FOR TEST JULIAN DATE
13¦
14¦       TEST 1:     168.52   i.e. If today;s date is 30
15¦                         days or more later than the
16¦                        test date, add 1.5% to payment.
17¦       Test Code          Detail
18¦          725754 CURRENT JULIAN DATE B11 to D11
19¦          725725 TEST DATE i.e. 5 DEC 1987 (B12 to D12)
20¦          171.05 OPPOSITE OF TEST 1
```

```
B11 = 365*(D6-1)+INT((D6-1)/4)
B12 = 365*(D7-1)+INT((D7-1)/4)
```

The formula in Cell C11 (and C12) calculates the days to the end of the previous month and then adds the days (dd). Multiplying the month number by INT(365/12) produces 30 for month 1 and one day too many for months 2 to 6 inclusive. The formula allows for this. It also produces a total of 364 for month 12, but December is not used in this process.

```
C11 = (IF(AND(C6-1>1,C6-1<7),-1,0))+(IF(C6-1
  =1,1,0))+INT(365/12*(C6-1))+B6
```

```
C12 = (IF(AND(C7-1>1,C7-1<7),-1,0))+(IF(C7-1
  =1,1,0))+INT(365/12*(C7-1))+B7
```

To allow for the quoted year being a leap year, D11 and D12 add the extra day, if necessary.

```
D11 = IF(AND(C6-1>=2,INT(D6/4)=D6/4),
  1,0)
```

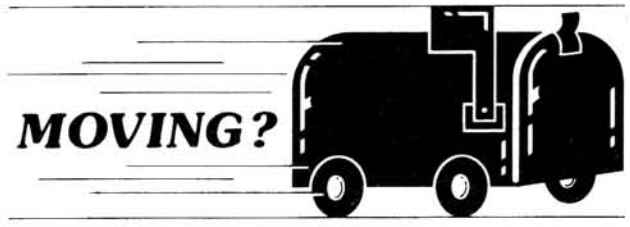
B18 and B19 add the three groups of "days" at B11, C11, D11 and B12, C12, D12 to give the Julian Date.

```
B18 = SUM(B11:D11)
```

C14 tests to see if the Current Date (B18) is 30 days or more after the Test Date (B19) and, if so, adds 1.5% to the amount due in F3. The payable amount is then displayed. B20 is for demonstration purposes only.

```
C14 = IF(B18-B19>=30,F3*1.015,F3)
B20 = IF(B18-B19>=30,F3,F3*1.015)
```

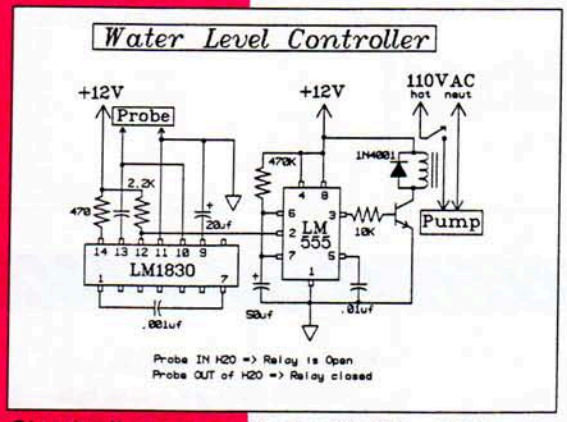
*



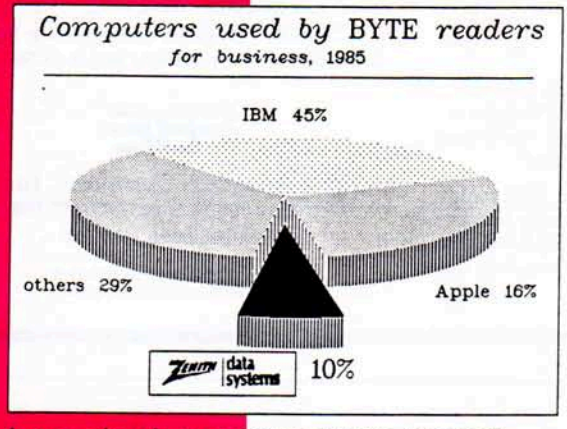
Please let us know 8 weeks in advance so you won't miss a single issue of REMark!

Want New & Interesting Software?
Check Out HUG Software

IMPROVE YOUR IMAGE WITH ShowOff™



Circuit diagram created with ShowOff



Lotus pie chart enhanced with ShowOff

ZIP video image displayed by ShowOff

Graphics to improve your image can now be yours with ShowOff for the Z100.

ShowOff is unequaled in graphic capability for the Z100.

ShowOff is a true graphic system integrator for the Z100.

Only with ShowOff do you get high resolution (640 x 480), 92 fill colors, 92 patterns, professional quality text and support for mice and digitizing tablets.

Only with ShowOff can you capture and combine any Z100 screens and create high resolution graphics.

Only with ShowOff can you capture and enhance Lotus 1-2-3, AutoCAD, and other graphics.

Only with ShowOff can you combine Macintosh pictures with Z100 graphics in high resolution. (Macintosh pictures are available in the public domain on many electronic bulletin boards and as commercial clip art in MS-DOS format.)

Only with ShowOff can you capture pictures* from PC-emulator boards and PC-compatibles using CGA graphics.

Only with ShowOff can you use your Z100 pictures with desktop publishing programs* for combining text and graphics.

Only with ShowOff do you get slide and slide show capabilities for self-running presentations and using high resolution pictures in your own programs.

Now we've added new capabilities for high resolution graphics with ZIP, Image Processing for the Z100.

ZIP brings to the Z100 a method for creating computer graphics from video sources. ZIP displays video images on the Z100 with support for the ImageWise video digitizer. ZIP has full image processing functions and creates high resolution pictures for use with ShowOff.
*Requires ShowOff Art/Utility Disk

ShowOff $79, with Logitech mouse $174
ShowOff Art/Utility Disk $15, ZIP $49
VISA . MC . personal checks
CALL TODAY: (314) 962-7833

HOGWARE COMPANY
470 BELLEVIEW
ST LOUIS MO 63119
(314) 962-7833

HOGWARE

ShowOff Demo Disk $3. ShowOff minimum requirements: H/Z100, 384K, full color memory, MS-DOS 2.0

Add 3.5″ Drives To The H/Z-90

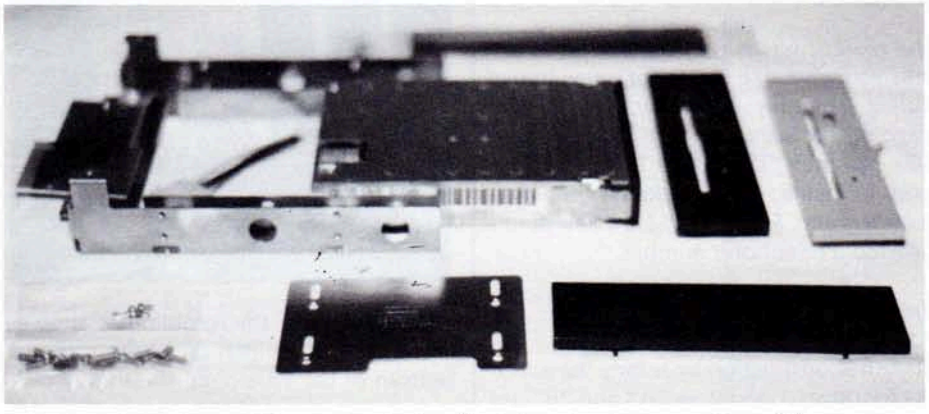
Don Powers
P.O. Box 969
Marion, MA 02738

With the advent of the Zenith Z-181 and MS-DOS 3.2, the 3.5″ disk drive becomes a potentially important peripheral. It offers potentially tremendous advantages to other Heath MS-DOS users, as well. These including: high density (720k bytes), low media profile, relatively safe transportation of software (hard, shuttered case), and a relatively low drive cost.

Shouldn't these advantages be available to all Heath users? Certainly. Consider the Z80-based H/Z-90 microcomputer. (The same procedure should work for an H-8 with an H8-37 controller.) I have two H-90s that have been nursed along from H-19 terminals to H-88s (tape drive) to H-89s (hard-sectored disk), and ultimately to H-90s using soft-sectored disks. With the latest versions of CP/M-80 and HDOS, these computers run 5.25″, 80-track drives (96-tpi). The initial Tandon drives (TM100-4 and TM101-4) were tremendously expensive; About $400 a piece. The half-height TEAC FD55F goes for about $150 today.

3.5″ Drives

A number of mail-order houses sell the Toshiba ND352 3.5″ floppy drive with a 5.25″ mounting kit for less than $150 (plus shipping and handling). I'll include the names of two companies of which I'm aware. However, there should be a number more by the time you read this: prices should drop with the competition.



I purchased two kits and they arrived a little over a week later from the opposite coast via UPS Air. Each drive and kit is packed in separate boxes. I was impressed by the quality of the drives and mounting kits. (The drives are model FDD 4202, which I have been told may already be obsolete.) I was disappointed that there were no technical specifications enclosed, but I assume from the measurements taken later that there is some savings in power over the larger drives.

You should only need three tools to assemble the kits. These consist of a small phillips head screw driver (all screws are phillips head), a pair of tweezers (essential for changing jumper plugs and holding screws), and a knife or scissors to cut the thick plastic pages holding the parts. You may also need an additional screw driver to remove any existing drives.

The 3.5″ drive sits eventually in a sturdy

metal bracket, so that it takes on the size and mounting hole positions of a half-height 5.25" drive. There is a PC card in the back of the bracket that mates to the data plug on the drive. The fingers on the other end of the card accept the standard 34-pin edge connector. A power cable adapter is also included. (The 3.5" drives have much smaller connectors.)

There is one jumper on the PC board that must be shifted to one position: "A", if you are mounting the drive in a new style AT and shifted to the other; "B", for the old style AT, PC or XT. The only other jumpeer to set is the drive selection plug in the drive. Here tweezers are essential. The access hole is too small for fingers.

In addition to the standard plastic 3.5" drive front plate (easily removable), the kit contains 5.25" face plates for both the regular PC (black) and AT (grey) type drives. In addition, you get a blank, snap in, half-height panel if your system previously used full-size drives. The blank panel will only fit a relatively narrow rim enclosure and will not fit the drive cutout in either the H/Z-90 computer or H/Z-77 disk drive case.

Other goodies included AT plastic side rails, one side mount plate with assorted screws and washers, and a 12-page manual. Ordering two kits gets you two side plates, which allow you to mount two drives together (one on top of the other), if necessary. These were important since by first mounting the drives together, you can use the existing holes in the H/Z-77 to mount the drives on one side.

The manual has two faults. First, it never tells you what size or type of screw to use during assembly. Secondly, on the last page of the manual, it tells you if all else fails, call their "Disk Drive Technical Support in Irvine, CA, weekdays between 9:00 am and 5:00 pm PST." They didn't provide a telephone number.

Putting It Together

Start the construction by setting the jumpers for drives 1 and 2, or "A" and "B" under CP/M-80. (HDOS should work, as well.) Next, attach the correct face plate, in this case, the black 5.25" one. After connecting the drive to the board in the bracket, it is necessary to firmly mount the drive by attaching four 3/16th inch long, 5-48 thread screws through the metal kit frame to the drive. (I'm guessing at the thread size; it is metric and somewhere between a four and a six with more threads per inch than a 4-40.) There is a separate bag of six screws all of this same size. Use four of these. If you hold the screws with the tweezers, this should be a relatively easy task, otherwise, it's impossible. Next, be certain that two screws holding the PC board to the frame are tight.

The last hardware step in the two kit assembly process is to connect the side mount plates. Again, there are plenty of screws from which to select. Use the two smaller thread 5/16-inch long flat head screws included with each plate. (These and all the other screws used have the same metric thread previously mentioned.) The manual also contains a two page description on the software configuration required for PC/MS-DOS 3.2.

The next step was to connect the drives to an H/Z-90. My configuration has an internal single-sided, 48 tpi drive which is configured as "C". The H/Z-77 has two Tandon 96 tpi drives as "A" and "B". I removed "B" and hooked up one of the 3.5" drives in its place. (They both fit in one side/half of the case.) Note that because of the side mounting plates, that are now vertical, the drives will not fit through the plastic front cutout.

The solution to this problem is simple. File 1/16th of an inch off of the top and bottom of the lip around the floppy disk cutout on the H/Z-90. Mounting the drives in the H/Z-77 drive case is easily remedied without any case modifications by sliding the drives in from the rear.

At this point, I ran into my only problem with the hardware supplied. My mounting requirements called for four pan head, metric screws and flat washers, two sets for mounting the top of the drive package and two for the bottom. There were only three screws. (There was an extra flat head of the size used for mounting the plates. Fortunately, each pair of AT plastic side rails came packaged with four screws, which attached to the same holes, and there were plenty to spare.

Next, connect the internal power and data cables to the 3.5" drive designated "B". Neither of the small drives has a removable resistor pack, so removing a pack is not a requirement. (The larger "A" drive was on the end of the data line and kept its resistor pack.)

The next step is to turn on the disk drive power supply and computer. BOOT the Z-90 from drive "A" under CP/M-80. Since both "A" and "B" were configured for 6msec track-to-track, 96 tpi drives, no installation of the high density disks were necessary. However, for anyone not already using high density drives, this is the first step. Call CONFIGFUR and say no to the standard configuration. Select "B" to change the drive specifications and save the changes in memory and on disk.

I next converted all my frequently used program disks over to the 3.5" format by first "FORMAT"ing my smaller disks. My directory command showed each to possess the same formatted, but empty 624k byte capacity that my 96 tpi, 5.25" disks have. You can get the full transfer of files by using DUP. Prices are dropping for 3.5" floppies, and if you shop around you should be able to pick up some quality brand names at bargain prices. The lowest price I've found is slightly under a dollar a piece for double-sided floppies.

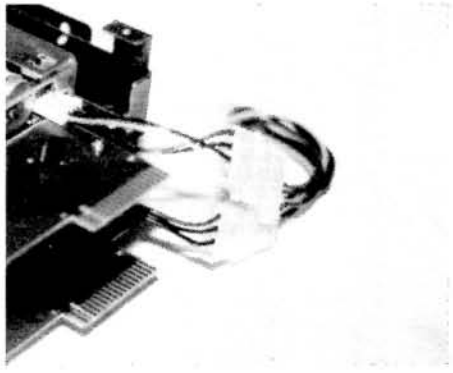
The next step is to turn the system "Off" and do away with the 5.25", 96 tpi disk drives. In my case, I left initially "C" as the internal drive (a single-sided, 48 tpi, 5.25" drive) and mounted the 3.5" drives externally. This means that "C" is still compatible with the Heath soft-sectored standard for easy transfer of software from original master disks, as well as to and from friends' machines.

Mounting Internally

At this point, the system is operational. However, I believe that a number of H/Z-90 users would prefer to use the 3.5" drives internally, and the prospect of doing it intrigued me. Assuming the H/Z-90 case drive opening has been enlarged, mount the drives. Use the same screws used to mount the drives in the case that the H/Z-77 case requires.

When using the H-88-9 metal case don't use the flat metal washers supplied with the drives. Do use the original black plastic washers to isolate the drives from the case. The screws are not long enough to fit through both and attach to the drives.

The other steps are explained below. They include adding a Y power cable extension, adding another 34-pin data connector to the data cable, and "possibly" reversing the cables to P3 and P4 on the Z-89-37 soft-sector controller board.



The first step is relatively easy. Your local Heath/Zenith Computers & Electronics Center should have a power supply Y adapter (Part No. MG-6083). I bought one for about $8.00. If they aren't available, check some of the electronic hardware magazines. The price may be cheaper, but the delivery is slower.

The second part you'll need is a 34-pin data edge connector. These are also fairly common. I had one left over from another project, which I attached to the data cable about 2-1/2 inches from the present drive connector.

To do the assembly, I put the connector on a work bench. Place the cable on the connector. Check the connector position carefully to see that pin numbers, key and orientation agree with the other drive edge connector. Next, put on the retaining clip as far as possible by hand. Check the position of the connector to the cable again. Put a flat plate (metal or wood) on top of the clip, and holding it in place, strike the plate with a hammer several times until the clip clicks into place to secure the connector to the cable. The sources for this connector are the same as for the Y power connector.

For the last step, there are two ways of designating "A" and "B" as the internal drives. One way calls for the reversal of the data cables on the soft-sector controller end. The internal drives must be connected to P4 "not P3". Be certain that jumper J6 is made and not J4, J5 or J7. The alternative method calls for leaving the internal drives on P3, but J4 "and J5" must be jumpered and "not J6 or J7". This is the route I took. (You'll need another .1 inch spaced jumper. Buy extras, they're easy to lose.)

The purpose for the switch in cables and/or jumpers is the logic controlling the flow of data "from" the drives. All signals to the drives are brought out to both connectors. However, data from the drives, the WRPT, INDEX, TRK0 and RDATA lines are not. They are fed into a quad tri-state multiplexer, a 74LS257. Which set of inputs are read from P3 or P4 is a function of the positioning of the jumper(s).

No Noise

At this point, you're ready to go. I checked my system for noise generation prior to making the conversion. (Do this by operating radios and televisions near the H/Z-90 before the disk conversion.) I couldn't detect any noticeable differences from the interference previously experienced when there was a 48 tpi drive in the high density H-88-9 enclosure.

Finally, eliminating noise is important because the system must meet FCC specs. The original H-89 upgrade (and disk-based computer) was not designed for high density disk drives. Thus, came the H-88-9 upgrade kit. However, when I tried the same test using the original H/Z-88 to H/Z-89 disk mounting case, the noise results were about the same. (Also, there did not appear to be any failure in computer operation.) If you get noise check your grounding.

Toshiba 3.5-inch drive suppliers:

Jade Computer Products
4901 W. Rosecrans Avenue
Box 5046
Hawthorne, CA 90250-5046
(800) 421-5500 (Continental U.S. and Hawaii)
(800) 262-1710 (Inside California)
(213) 973-7077 (In Los Angeles)

JDR Microdevices
110 Knowles Drive
Los Gatos, CA 95030
(800) 538-5000

*

The following HUG Price List contains a list of all products in the HUG Software Catalog and Software Catalog Update #1. For a detailed abstract of these products, refer to the HUG Software Catalog, Software Catalog Update #1, or previous issues of REMark.

Make the no-hassle connection with your modem today! ***HUGMCP*** doesn't give you long menus to sift through like some modem packages do. With *HUGMCP, YOU'RE* always in control, not the software. Order ***HUG P/N 885-3033-37*** today, and see if it isn't the easiest-to-use modem software available. Joe Katz says it was so easy to use, he didn't even need to look at the manual. "It's the only modem software that I use, and I'm in charge of both HUG bulletin boards!" says Jim Buszkiewicz. *HUGMCP* runs on ANY Heath/Zenith computer that's capable of running MS-DOS!

HEPCAT is here! *HEPCAT* is here! *HEPCAT* is here! So what is *HEPCAT*, you may ask? Why it's just another Pat Swayne SUPER-UTILITY. *HEPCAT* is an acronym for *HUG Engineer's and Programmer's CAlculation Tool*. Just what we don't need, another memory resident calculator, right? Wrong! With *HEPCAT*, you can throw away the rest and use the best. *HEPCAT* only uses two partial lines on your screen, and best of all, does NOT cause existing programs to stop executing! That means, while your computer is grinding numbers internally, you can be grinding them externally. Order ***HUG P/N 885-3045-37***.

Can't remember how to use the MS-DOS 'COPY' command? Forget the exact command line format for 'ASGNPART'. Too far to go for the MS-DOS manuals on the shelf on the other side of the room? Why not just type 'HELP' on the keyboard? You say it comes back with "Bad command or file name"? It wouldn't if you had HUG's ***HELP*** program. With *HELP* installed on your hard disk, all you need to do is type 'HELP' for a complete list of MS-DOS commands and transients along with a brief explanation of how each command works, as well as the format for its use. *HELP*, ***HUG P/N 885-8040-37***, works on *ALL* Heath/Zenith computers that run MS-DOS!

HUG Price List

PRODUCT NAME	PART NUMBER	OPERATING SYSTEM	DESCRIPTION	PRICE
H8 — H/Z-89/90				
ACCOUNTING SYSTEM	885-8047-37	CPM	BUSINESS	20.00
ACTION GAMES	885-1220-[37]	CPM	GAME	20.00
ADVENTURE	885-1010	HDOS	GAME	10.00
ASCRITY	885-1238-[37]	CPM	AMATEUR RADIO	20.00
AUTOFILE (Z80 ONLY)	885-1110	HDOS	DBMS	30.00
BHBASIC SUPPORT PACKAGE	885-1119-[37]	HDOS	UTILITY	20.00
CASTLE	885-8032-[37]	HDOS	ENTERTAINMENT	20.00
CHEAPCALC	885-1131-[37]	HDOS	SPREADSHEET	20.00
CHECKOFF	885-8010	HDOS	CHECKBOOK SOFTWARE	25.00
DEVICE DRIVERS	885-1105	HDOS	UTILITY	20.00
DISK UTILITIES	885-1213-[37]	CPM	UTILITY	20.00
DUNGEONS & DRAGONS	885-1093-[37]	HDOS	GAME	20.00
FLOATING POINT PACKAGE	885-1063	HDOS	UTILITY	18.00
GALACTIC WARRIORS	885-8009-[37]	HDOS	GAME	20.00
GALACTIC WARRIORS	885-8009-[37]	CPM	GAME	20.00
GAMES 1	885-1029-[37]	HDOS	GAMES	18.00
HARD SECTOR SUPPORT PACKAGE	885-1121	HDOS	UTILITY	20.00
HDOS PROGRAMMERS HELPER	885-8017	HDOS	UTILITY	16.00
HOME FINANCE	885-1070	HDOS	BUSINESS	18.00
HUG DISK DUPLICATION UTILITIES	885-1217-[37]	CPM	UTILITY	20.00
HUG SOFTWARE CATALOG	885-4500	VARIOUS	PRODUCTS THRU 1982	9.75
HUGMAN & MOVIE ANIMATION	885-1124	HDOS	ENTERTAINMENT	20.00
INFO. SYSTEM AND TEL. & MAIL SYSTEM	885-1108-[37]	HDOS	DBMS	30.00
LOGBOOK	885-1107-[37]	HDOS	AMATEUR RADIO	30.00
MAPLE	885-8005	HDOS	COMMUNICATION	35.00
MAPLE	885-8012-[37]	CPM	COMMUNICATION	35.00
MICRONET CONNECTION	885-1122-[37]	HDOS	COMMUNICATION	20.00
MISCELLANEOUS UTILITIES	885-1089-[37]	HDOS	UTILITY	20.00
MORSE CODE TRANSCEIVER	885-8016	HDOS	AMATEUR RADIO	20.00
MORSE CODE TRANSCEIVER	885-8031-[37]	CPM	AMATEUR RADIO	20.00
PAGE EDITOR	885-1079-[37]	HDOS	UTILITY	25.00
PROGRAMS FOR PRINTERS	885-1082	HDOS	UTILITY	20.00
REMARK VOL 1 ISSUES 1-13	885-4001	N/A	1978 TO DECEMBER 1980	20.00
RUNOFF	885-1025	HDOS	TEXT PROCESSOR	35.00
SCICALC	885-8027	HDOS	UTILITY	20.00
SMALL BUSINESS PACKAGE	885-1071-[37]	HDOS	BUSINESS	75.00
SMALL-C COMPILER	885-1134	HDOS	LANGUAGE	30.00
SOFT SECTOR SUPPORT PACKAGE	885-1127-[37]	HDOS	UTILITY	20.00
STUDENT'S STATISTICS PACKAGE	885-8021	HDOS	EDUCATION	20.00
SUBMIT (Z80 ONLY)	885-8006	HDOS	UTILITY	20.00
TERM & HTOC	885-1207-[37]	CPM	COMMUNICATION & UTILITY	20.00
TINY BASIC COMPILER	885-1132-[37]	HDOS	LANGUAGE	25.00
TINY PASCAL	885-1086-[37]	HDOS	LANGUAGE	20.00
UDUMP	885-8004	HDOS	UTILITY	35.00
UTILITIES	885-1212-[37]	CPM	UTILITY	20.00
UTILITIES BY PS	885-1126	HDOS	UTILITY	20.00
VARIETY PACKAGE	885-1135-[37]	HDOS	UTILITY & GAMES	20.00
VOLUME I	885-1008	N/A	SOFTWARE LISTINGS	9.00
VOLUME II	885-1013	N/A	SOFTWARE LISTINGS	12.00
VOLUME III	885-1015	N/A	SOFTWARE LISTINGS	9.00
VOLUME IV	885-1037	N/A	SOFTWARE LISTINGS	12.00
WATZMAN ROM SOURCE & DOC	885-1221-[37]	CPM	H19 FIRMWARE	30.00
WATZMAN ROM	885-4600	N/A	H19 FIRMWARE	45.00
WHEW UTILITIES	885-1120-[37]	HDOS	UTILITY	20.00
XMET ROBOT X-ASSEMBLER	885-1229-[37]	CPM	UTILITY	20.00
Z80 ASSEMBLER	885-1078-[37]	HDOS	UTILITY	25.00
Z80 DEBUGGING TOOL (ALDT)	885-1116	HDOS	UTILITY	20.00
H8 — H/Z-89/90 — H/Z-100 (Not PC)				
ADVENTURE	885-1222-[37]	CPM	GAME	10.00
BASIC-E	885-1215-[37]	CPM	LANGUAGE	20.00
CASSINO GAMES	885-1227-[37]	CPM	GAME	20.00
CHEAPCALC	885-1233-[37]	CPM	SPREADSHEET	20.00
CHECKOFF	885-8011-[37]	CPM	CHECKBOOK SOFTWARE	25.00
COPYDOS	885-1235-37	CPM	UTILITY	20.00
DISK DUMP & EDIT UTILITY	885-1225-[37]	CPM	UTILITY	30.00
DUNGEONS & DRAGONS	885-1209-[37]	CPM	GAMES	20.00
FAST ACTION GAMES	885-1228-[37]	CPM	GAME	20.00
FUN DISK I	885-1236-[37]	CPM	GAMES	20.00
FUN DISK II	885-1248-[37]	CPM	GAMES	35.00
GAMES DISK	885-1206-[37]	CPM	GAMES	20.00
GRADE	885-8036-[37]	CPM	GRADE BOOK	20.00
HRUN	885-1223-[37]	CPM	HDOS EMULATOR	40.00
HUG FILE MANAGER & UTILITIES	885-1246-[37]	CPM	UTILITY	20.00
HUG SOFTWARE CATALOG UPDATE #1	885-4501	VARIOUS	PRODUCTS 1983 THRU 1985	9.75
KEYMAP CPM-80	885-1230-[37]	CPM	UTILITY	20.00
MBASIC PAYROLL	885-1218-[37]	CPM	BUSINESS	60.00
MICRONET CONNECTION	885-1224-[37]	CPM	COMMUNICATION	16.00
NAVPROGSEVEN	885-1219-[37]	CPM	FLIGHT UTILITY	20.00
REMARK VOL 3 ISSUES 24-35	885-4003	N/A	1982	20.00
REMARK VOL 4 ISSUES 36-47	885-4004	N/A	1983	20.00
REMARK VOL 5 ISSUES 48-59	885-4005	N/A	1984	25.00
REMARK VOL 6 ISSUES 60-71	885-4006	N/A	1985	25.00
REMARK VOL 7 ISSUES 72-83	885-4007	N/A	1986	25.00

PRODUCT NAME	PART NUMBER	OPERATING SYSTEM	DESCRIPTION	PRICE
SEA BATTLE	885-1211-[37]	CPM	GAME	20.00
UTILITIES BY PS	885-1226-[37]	CPM	UTILITY	20.00
UTILITIES	885-1237-[37]	CPM	UTILITY	20.00
X-REFERENCE UTILITIES FOR MBASIC	885-1231-[37]	CPM	UTILITY	20.00
ZTERM	885-3003-[37]	CPM	COMMUNICATION	20.00
H/Z-100 (Not PC) Only				
ACCOUNTING SYSTEM	885-8048-37	MSDOS	BUSINESS	20.00
CALC	885-8043-37	MSDOS	UTILITY	20.00
CARDCAT	885-3021-37	MSDOS	BUSINESS	20.00
CHEAPCALC	885-3006-37	MSDOS	SPREADSHEET	20.00
CHECKBOOK MANAGER	885-3013-37	MSDOS	BUSINESS	20.00
CP/EMULATOR	885-3007-37	MSDOS	CPM EMULATOR	20.00
DBZ	885-8034-37	MSDOS	DBMS	25.00
ETCHDUMP	885-3005-37	MSDOS	UTILITY	20.00
EZPLOT	885-3023-37	MSDOS	PRINTER PLOTTING UTILITY	20.00
GAMES CONTEST PACKAGE	885-3017-37	MSDOS	GAMES	25.00
GAMES PACKAGE II	885-3044-37	MSDOS	GAMES	25.00
GRAPHICS	885-3031-37	MSDOS	ENTERTAINMENT	20.00
HELPSCREEN	885-3039-37	MSDOS	UTILITY	20.00
HUG BACKGROUND PRINT SPOOLER	885-1247-37	CPM	UTILITY	20.00
HUG BACKGROUND PRINT SPOOLER	885-5009-37	CPM86	UTILITY	20.00
HUGPBBS	885-5006-37	CPM86	COMMUNICATION	40.00
HUGPBBS SOURCE LISTING	885-5007-37	CPM86	COMMUNICATION	60.00
ICT 8080 TO 8088 TRANSLATOR & HFM	885-5008-37	CPM86	UTILITY	20.00
KEYMAC	885-3046-37	MSDOS	UTILITY	20.00
KEYMAP	885-3010-37	MSDOS	UTILITY	20.00
KEYMAP	885-5001-37	CPM86	UTILITY	20.00
KEYMAP CPM-85	885-1245-37	CPM	UTILITY	20.00
MAPLE	885-8023-37	CPM	COMMUNICATION	35.00
MATHFLASH	885-8030-37	MSDOS	EDUCATION	20.00
ORBITS	885-8041-37	MSDOS	EDUCATION	25.00
POKER PARTY	885-8042-37	MSDOS	ENTERTAINMENT	20.00
SCICALC	885-8028-37	MSDOS	UTILITY	20.00
SKYVIEWS	885-3015-37	MSDOS	ASTRONOMY UTILITY	20.00
SMALL-C COMPILER	885-3026-37	MSDOS	LANGUAGE	30.00
SPELL5	885-3035-37	MSDOS	SPELLING CHECKER	20.00
SPREADSHEET CONTEST PACKAGE	885-3017-37	MSDOS	VARIOUS SPREADSHEETS	25.00
TERM86 & DSKED	885-5004-37	CPM86	COMMUNICATION & UTILITIES	20.00
TREE-ID	885-3036-37	MSDOS	TREE INDENTIFIER	20.00
USEFUL PROGRAMS I	885-3022-37	MSDOS	UTILITIES	30.00
UTILITIES BY PS	885-5003-37	CPM86	UTILITY	20.00
UTILITIES	885-3008-37	MSDOS	UTILITY	20.00
ZBASIC DUNGEONS & DRAGONS	885-3009-37	MSDOS	GAME	20.00
ZBASIC GRAPHIC GAMES	885-3004-37	MSDOS	GAMES	20.00
ZBASIC GAMES	885-3011-37	MSDOS	GAMES	20.00
ZPC II	885-3037-37	MSDOS	PC EMULATOR	60.00
ZPC UPGRADE DISK	885-3042-37	MSDOS	UTILITY	20.00
H/Z-100 — PC Compatibles				
ADVENTURE	885-3016-37	MSDOS	GAME	10.00
ASSEMBLY LANGUAGE UTILITIES	885-8046-37	MSDOS	UTILITY	20.00
DEBUG SUPPORT UTILITIES	885-3038-37	MSDOS	UTILITY	20.00
DPATH	885-8039-37	MSDOS	UTILITY	20.00
HADES	885-3040-37	MSDOS	UTILITY	40.00
HELP	885-8040-37	MSDOS	CAI	20.00
HEPCAT	885-3045-37	MSDOS	UTILITY	35.00
HUG BACKGROUND PRINT SPOOLER	885-3029-37	MSDOS	UTILITY	20.00
HUG EDITOR	885-3012-37	MSDOS	TEXT PROCESSOR	20.00
HUG MENU SYSTEM	885-3020-37	MSDOS	UTILITY	20.00
HUG SOFTWARE CATALOG UPDATE #1	885-4501	VARIOUS	PROD 1983 THRU 1985	9.75
HUGMCP	885-3033-37	MSDOS	COMMUNICATION	40.00
HUGPBBS SOURCE LISTING	885-3028-37	MSDOS	COMMUNICATION	60.00
HUGPBBS	885-3027-37	MSDOS	COMMUNICATION	40.00
ICT 8080 TO 8088 TRANSLATOR	885-3024-37	MSDOS	UTILITY	20.00
MATT	885-8045-37	MSDOS	MATRIX UTILITY	20.00
MISCELLANEOUS UTILITIES	885-3025-37	MSDOS	UTILITIES	20.00
REMARK VOL 5 ISSUES 48-59	885-4005	N/A	1984	25.00
REMARK VOL 6 ISSUES 60-71	885-4006	N/A	1985	25.00
REMARK VOL 7 ISSUES 72-83	885-4007	N/A	1986	25.00
SCREEN DUMP	885-3043-37	MSDOS	UTILITY	30.00
UTILITIES II	885-3014-37	MSDOS	UTILITY	20.00
PC Compatibles				
ACCOUNTING SYSTEM	885-8049-37	MSDOS	BUSINESS	20.00
CARDCAT	885-6006-37	MSDOS	CATALOGING SYSTEM	20.00
CHEAPCALC	885-6004-37	MSDOS	SPREADSHEET	20.00
CP/EMULATOR II & ZEMULATOR	885-6002-37	MSDOS	CPM & Z100 EMULATORS	20.00
DUNGEONS & DRAGONS	885-6007-37	MSDOS	GAME	20.00
EZPLOT	885-6003-37	MSDOS	PRINTER PLOTTING UTILITY	20.00
GRADE	885-8037-37	MSDOS	GRADE BOOK	20.00
HAM HELP	885-6010-37	MSDOS	AMATEUR RADIO	20.00
KEYMAP	885-6001-37	MSDOS	UTILITY	20.00
LASERWRITER CONNECTION	885-8050-37	MSDOS	PRINTER UTILITY	40.00
SCREEN SAVER PLUS	885-6009-37	MSDOS	UTILITIES	20.00
SKYVIEWS	885-6005-37	MSDOS	ASTRONOMY UTILITY	20.00
TCSPELL	885-8044-37	MSDOS	SPELLING CHECKER	20.00

You've got a screen full of important technical data that would be nearly impossible to memorize, and you already have writer's cramps from the last screen full. With ***SCREENDUMP*** from HUG, you can reproduce a complete video screen on a dot matrix printer, including both text and graphics without having to exit the current program. *SCREENDUMP* supports most of the more popular dot matrix printers, including the newer 24-pin and laser jet models. The latest version of *SCREENDUMP* is ***HUG P/N 885-3043-37***.

"Thank Heaven for *HADES*!" That's what a lot of MS-DOS users are saying when ***HADES*** rescues a file that just got accidentally erased. Erased file recovery is only a small part of the capabilities of this program. *HADES* is *HUG's Absolute Disk Editing System*. Within the realms of MS-DOS, *HADES* allows you to directly edit any part of any disk. Directories, files, file attributes. FATS: nothing can hide from you when you use *HADES*. *HADES* works on *ANY* computer that can run MS-DOS version 2 or greater. Order ***HUG P/N 885-3040-37*** today!

Want to keep your H/Z-100? Want to run a lot of that good PC compatible software out there? Don't want to buy a PC compatible though? Then get ***ZPC II, HUG P/N 885-3037-37***, and the ***ZPC II upgrade disk, HUG P/N 885-3042-37***.

ORDERING INFORMATION

For VISA and MasterCard phone orders, telephone the Heath Users' Group directly at (616) 982-3838. Have the part number(s), descriptions, and quantity ready for quick processing. By mail, send your order, plus 10% postage and handling ($1.00 minimum charge, up to a maximum of $5.00) to: Heath Users' Group, P.O. Box 217, Benton Harbor, MI 49022-0217. VISA and MasterCard require minimum $10.00 order. No C.O.D.s accepted.

Questions regarding your subscription? Call Margaret Bacon at (616)982-3463.

10 – Very Good
9 – Good
8 – Average

TABLE C
Product Rating

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

7 – 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/Z–89) 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., 885–1103 or 885–1211[–37] SEA BATTLE.

ORDERING INFORMATION

For VISA and MasterCard phone; telephone Heath/Zenith Users' Group directly at (616) 982-3838. Have the part number(s), description, and quantity ready for quick processing. VISA and MasterCard require minimum $10.00 order. By mail, send your order, plus 10% postage/handling ($1.00 minimum, $5.00 maximum) to: Heath/Zenith Users' Group, P.O. Box 217, Benton Harbor, MI 49022-0217. Orders may be placed, by mail only, using your Heath Revolving Charge account. Purchase orders are also accepted by phone or mail. No C.O.D.s accepted.

Questions or problems regarding HUG software or REMark magazine should be directed to HUG at (616) 982-3463.

NOTES

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).

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.

HUG P/N 885-3047-37
Z-100 WordStar Connection
$20.00

The Z-100 WordStar Connection is a set of utilities that allows you to purchase the PC-compatible version of WordStar(R) Release 4.0 and run it on a Z-100 (not PC) series computer without any PC emulation required. The Word Finder thesaurus program is also supported by this package, making it the most complete solution to using WordStar 4.0 on the Z-100. Since the PC-compatible version of WordStar 4.0 can be obtained for as little as $89 by upgrading a previous version, the Z-100 WordStar Connection is also your best value.

Requirements: To use this package, you need a Z-100 series computer with at least 256k of memory (320k if you want to use Word Finder), and the Z-100 version of MS-DOS 2 or above (for example, Heath Catalog no. OS-63-30). WordStar 4.0 will not run under Z-DOS (which is actually MS-DOS version 1).

Program Author: Patrick Swayne, HUG Software Engineer

The Z-100 WordStar Connection disk contains these files:

README	.DOC	ZWFREMOV	.COM
ZWS	.DOC	FIXWF	.BAT
FIXWS	.COM	FIXWF	.DAT
FIXWSN	.COM	FIXWS	.ASM
ZINSTALL	.COM	ZINSTALL	.ASM
ZCHANGE	.COM	ZCHANGE	.ASM
FIXWSPC	.COM	FIXWSPC	.ASM
ZWF	.DOC	ZWF	.ASM
ZWF	.COM	ZWFINS	.ASM
ZWFINS	.COM	ZWFREMOV	.ASM
ZWFINSER	.COM		

Here is an explanation of the files:

ZWS.DOC -- Instructions for using the PC-compatible version of WordStar 4.0 on a Z-100.

FIXWS.COM, ZINSTALL.COM, ZCHANGE .COM -- The programs in this package that make it possible to run WordStar 4.0.

FIXWSPC.COM -- If you also use WordStar 4.0 on a PC-compatible computer, this program can be used to patch it so that it uses similar keypad and function key assignments as the Z-100 version produced by FIXWS. It puts more power in the keypad keys than you have with the default layout.

ZWF.DOC -- Instructions for using Word Finder on a Z-100.

ZWF.COM, ZWFINS.COM, ZWFINSER .COM, ZWFREMOV.COM, FIXWF.BAT, FIXWF.DAT -- The programs in this package that make it possible to run Word Finder on a Z-100.

FIXWS.ASM, etc. -- The source code for all programs is supplied.

Table C rating: 10.

HUG P/N 885-4008
REMark Volume 8..... $25.00

Now available from HUG is REMark Volume 8. This volume set differs from all previous volumes in that it consists of all 12 original issues of REMark from 1987. If you're a new member to HUG, and just purchased your PC compatible computer, this volume as well as Volume 7 from 1986 is for you! Both volumes are chock full of articles pertaining specifically to your system. Order yours today before supplies run out!

IMAGER BACKUP

by Joseph Katz

This VCR backup system for IBM compatible computers deserves more than one look.

Because its business end is an ordinary video casette recorder, Autofax Corporation's Imager disk-to-tape backup system for IBM compatible computers is so inexpensive that it's worth considering on the basis of price alone. But price alone is not all that Imager has to offer. Because it uses ordinary video cassettes in that ordinary VCR, Imager offers some extraordinary capabilities to recommend it above FastBack or other floppy disk backup software at one end of the scale and dedicated streaming tape backup systems at the other.

Costs

Compare each system's price of admission and the cost of its munchies once you get in. You'll probably be astonished.

FastBack lists for $179. A good 40 MB streaming tape system lists for about $700. Imager lists for $295 but is available to members of the Heath/Zenith User's Group for the special price of $195—a few dollars more than FastBack and about $500 less than a streaming tape system. (The deal is good enough for users of other computers to spend $20 on a year's membership in HUG. I've therefore included information on joining.)

FastBack requires first-quality floppy diskettes at a list price of $2-4 each, depending on whether you back up to an XT or AT's 360 KB drive or an AT's 1.2 MB drive. Streaming tape cartridges list for about $25 each. Imager requires first-quality video cassettes at a list price of $6.95 for the 120-minute size. Look at the cost-per-megabyte for backups on each system: FastBack has a cost-per-megabyte of about $4 on a 1.2 MB drive and a cost-per-megabyte of about $6 on a 360 KB drive; a 40 MB streaming tape system has a cost-per-megabyte of about 62.5 cents; Imager with a 120-minute cassette has an astonishingly tiny cost-per-megabyte of about 6.2 cents. Of course you can get discounts on media for each system, so the actual cost-per-megabyte will depend on your ability to wheel and deal. But the hierarchy will stand, I think. And I think you'll find it easy to lower Imager's cost-per-megabyte because first-quality video cassettes are consumer—not computer—products that are widely discounted.

Of course you must not discount the labor involved in using each system. FastBack requires a human's presence throughout backups and restores because someone has to swap diskettes in and out of the disk drive. Streaming tape backup systems can be used in conjunction with timing devices for unattended backups. Imager can use a programmable VCR, or a non-programmable VCR connected to a timer, for unattended backups.

The bottom line, by every standard, is that Imager is relatively the least expensive backup system on the market. In fact Imager is downright cheap if you already own a VCR. Autofax supplies the add-in board, software, cables, and manual that comprise Imager itself. You supply the VCR. Whatever you have probably will do nicely: Imager isn't fussy about video for-

Address all correspondence to Joseph Katz, 103 South Edisto Avenue, Columbia SC 29205. I'll try to answer letters accompanied by a self-addressed stamped envelope, but my volume of mail is too heavy for me to promise. Unless your letter specifically says otherwise, I'll assume the right to publish it (edited, if I think that appropriate).

For Further Information

Imager. $295 ($195 to HUG members).
The Light Pen Company
Box 45255
Los Angeles, CA 90045-0255
800/634-1967

HUG membership. $22.95 new.
$19.95 renewal.
Heath/Zenith User's Group
Box 217
Benton Harbor, MI 49022
616/982-3838

mats and is supposed to work with NTSC, SECAM, or PAL non-interlaced formats. If those abbreviations dribble off into alphabet soup for you, you're in the home VCR league. In that case you'll be happy to know that Imager likes both VHS and Betamax machines.

Should you be without a VCR, you might want to use Imager as the excuse to buy one. An inexpensive unit with two heads, SP ("Standard Play") speed, and RCA jacks for video input and video output is perfectly adequate, but programmability and the ability to search by cue or tape counter are desirable. Keep in mind, though, that you needn't keep Imager's needs exclusively in mind. Imager doesn't modify the VCR in any way. When you want to use it for backups or restores you simply plug two cables from the board into the VCR's video input and output jacks. You therefore can use the VCR to watch your favorite Fred Astaire movie whenever you're not actually using it as a backup system. Imager may be one of the closest things to a free lunch today. It's certainly a backup bargain.

Installation

Installing Imager is a relatively easy three-part job. First you install the Imager board in your computer, then you cable the board to your VCR, and finally you install the Imager software. Do things in that order.

The board goes into any full length slot in your computer. Power consumption is a maximum of 4 Watts so any but the puniest or most overburdened of computers ought to take the Imager board in stride: Heath and Zenith computers will have no trouble satisfying the power requirements. (My Z-248 is loaded with add-in boards and had no trouble.) Cabling the board to your VCR is easy if you ignore the instructions in Imager's current manual: they talk about red and white plugs on the cables, but the cables supplied with mine had red and black plugs. Not to worry. Imager comes with a pair of cables that terminate in RCA jacks ("phono jacks") of the kind used to connect audio equipment. The Imager board has only two jacks and the video input jack is above the video output jack. Plug one cable from the video output jack on the Imager board to the video input jack on the VCR: that's the backup connection. Plug the other cable from the video input jack on the Imager board to the video output jack on the VCR: that's the restore connection. Installing the software is simply a matter of running an installation program. It copies required files to your desired hard disk, then automatically tests to see that the board is alive and well and communicating with your VCR. You can't run Imager until that automatic check is passed. When it is, your system is ready to go.

Operation

Everything goes the same way: turn on the VCR, pop in a video cassette, rewind it to the start, and—if you're new to Imager—run Imager's menu-operated software (Figure 1). The "Backup," "Verify/Compare," "List Files," and "Restore" options invoke a series of questions that determine the directories and files to be used in each operation (Figure 2). Either unambiguous filenames or wildcards are acceptable, so you can tackle a complete hard disk or any part of it. The current software version (2.7b) handles up to 126 directories at a time, with up to 579 files in each.

Since the software is modular and the menu is really only a shell for separate Imager programs that do the actual work, an experienced user can skip the menu and run the separate programs themselves. You can run them from the DOS command line or—better yet—from batch files. The reason why it's often better to run the programs from batch files is that they can be used to thoroughly customize and automate your backup systems.

I've switched to the plural "systems" now because Imager's modular software lets you take such good advantage of the flexibility designed into home VCRs and their video cassettes that you can create batch files to make a variety of systems for your own special needs. See the sidebars for some Imager tips and tricks.

If you have either a programmable VCR or an external timer on a non-programmable VCR, for example, batch files can make Imager's MRBACK program back up your disks automatically after your normal working hours. That way no one has to waste time watching the machines. For another example, batch files can make Imager's MRDIR program write directory listings of your backup video cassettes to a disk file. It becomes a rough-and-ready catalog of your backup video cassettes. Should only a few disk files become corrupt and need replacing with backups, you can use your word processor to search that catalog for the cassettes containing the specific files you need. Of course if you have a database management program and some knowledge of how to program it you can convert that rough-and-ready catalog into a sophisticated index of your video cassette backups and employ them as an archive for your software and data library.

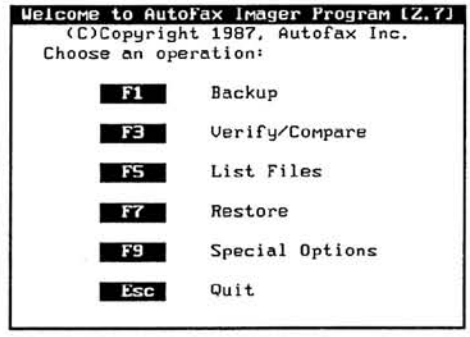


Figure 1. Imager's main menu.

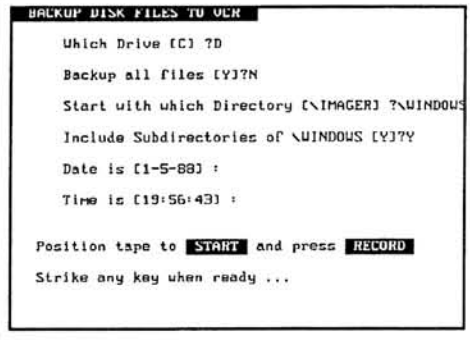


Figure 2. A typical Imager sub-menu.

Applications

That kind of flexibility, coupled with the low cost of setting up and running the system, makes Imager particularly well suited to a variety of special needs. Desktop publishing can serve as the source for extended examples that might stimulate your creativity in thinking about what Imager can do for your own major application.

Desktop publishing really demands frequent mirror image backups of every hard disk. That way when—not "if"—a hard disk crashes every file can be restored to its original place. File locations, an inconsequential matter to programs used in most other applications, matter a great deal to software like PageMaker and Ventura

Publisher. They map the trail to each component of a publication and falter or die when the terrain shifts unexpectedly. Imager will do mirror image backups of two 30 MB hard disk drives on one 90-minute VHS cassette in about an hour. You therefore can start Imager running and go out to lunch without worrying about days of lost work when your hard disk drives go out to lunch too. Reformat them if you're lucky, or replace them if you're not, and use Imager to restore their contents in one sustained spurt. You'll be back in business immediately and won't even have to reinstall PageMaker, Ventura Publisher, or other software.

Desktop publishing also creates the need for periodic hard disk housekeeping to clear the crumbs left behind at the end of each project. They're substantial crumbs. A publication usually consumes at least a few hundred kilobytes for its main files and scatters around countless more kilobytes of graphics and text files converted or edited during the course of preparing the main files. You can't live with all those files on your hard disks forever. Of course you can't live without them either: what happens if you need to revise the entire publication or discover occasions to reuse some of its components? One way out from between that rock and its attendant hard place is to revise your procedures with Imager's capabilities in mind. Segregate each publication and associated files into its own subdirectory, then when the job is finished use Imager to archive that subdirectory. Using a VHS machine Imager stores 26 MB on a 30-minute cassette, 55 MB on a 60-minute cassette, 83 MB on a 90-minute cassette, and 110 MB on a 120-minute cassette. Once the material is safely archived you can erase it from your hard disk, remove the subdirectory, and reuse that space for a new job. Imager can recreate the entire subdirectory on demand, immediately ready to meet the requirements of PageMaker, Ventura Publisher, and other desktop publishing software.

A variation of this Imager capability helps meet desktop publishing's need for temporary offline storage. There's at least some dead time in every publishing project, such as the time spent waiting for proofs to be read, corrected, and returned. During that time the publication merely languishes on a hard disk, eating expensive space for no good reason except the greater cost of time to move the publication offline temporarily. Removable hard disks would be ideal because they can be swapped online for work and offline for storage, but at a minimum of $100 each they are prohibitively expensive when a stalled publication must be stored until the cows come home. Imager looks like the next best solution because video cassettes now cost around $6 each and Imager has a data transfer rate of 12,000 bytes per second. It therefore can back up or restore 10 MB in 11 minutes, fast enough so individual subdirectories can be held offline, on inexpensive video cassettes, while one publication is between times and another publication makes more profitable use of precious online hard disk space.

Imager Tip #1:. Make Both Backups and Archives

Distinguish between "backups" (kept against the need to restore a crashed hard disk) and "archives" (software and data stored offline).

The distinction is useful because VCRs access the contents of video cassettes sequentially. Sequential access won't matter to you if want to restore your entire system from the Imager backup cassette: Imager will read and write every file in order and that is just what you want. But if you want to restore only the last file from a big backup you'll have to wait until the cassette winds slowly past everything you don't need before Imager picks out just what you do need. Remember that the last file on a 120-minute video cassette can take nearly two hours to reach.

So you might want to reserve a few cassettes for regular mirror image backups of your entire hard disks and use other cassettes for archiving at least your individual software packages. If you install a new Microsoft Windows application, for example, and later discover that it has modified the WIN.INI file in some undesirable way you can recreate the original much more quickly if you've archived Windows than if it's only somewhere on your last mirror image backup. Besides, if you're making frequent mirror image backups, you may have overwritten your favorite installation of Windows with the one it turns out you really don't want.

There's one tip that will prove useful whether you backup only or backup and archive too. Keep your subdirectories small—not only under the 579 files Imager's software can handle currently, but as small as possible—by reserving each subdirectory according to some logical principle. At least keep each major application in its own subdirectory, and if you have more than one software package for an application consider keeping each in its own subdirectory. That way if a subdirectory is corrupted you might be able to use the remedy of last restort while minimizing your losses: first erase the subdirectory's contents, then remove the subdirectory itself, and finally have Imager restore the entire subdirectory with the original contents.

Conclusions

The most important blessing Imager bestows is the economies achieved by substituting mass market products for specialized computer products. You must recognize the implications of that substitution or else you will turn that blessing into a curse.

VCR technology is analog, recording

Imager Tip #2: Cue or Count to Store Multiple Backups on One Cassette

If your VCR has a cuing feature, it's easy to put more than one backup or archive on a high capacity video cassette. Just leave about 10-15 seconds between backups and use the cue search to look for those spaces.

If your VCR has no cuing feature but does have a counter, zero it every time you put a cassette into the machine. Then note the count when you begin a backup or archive and the count when you end it. Leave enough space between the end of one backup or archive and the beginning of the next so that you don't accidentally overwrite something old with something new.

Cuing, however, is automatic and therefore probably better than counting.

the continuous flow of information that make up images intended to blend into one other on playback. It therefore doesn't matter if part of an image is lost because of weak spots in the video cassette's magnetic coating. Such "dropouts" are common on video cassettes, more so on those of poor quality, but who really cares about the loss of an occasional pixel in Fred Astaire's flashing feet? Every data bit matters in computer backups, though, because digital data is discrete and discontinuous. Therefore the requirements of digital technology are quite strenuous.

Imager compensates for the limitations of quality video cassettes by an automatic process, calculated into the backup times, of "quadruple redundant recording." Each data block is recorded in four different places to the odds that at least one of those four recordings will be good on a quality video cassette. All bets are off with poor quality video cassettes. They have enough weak spots to defeat the aim of redundant recording.

So does recording at slow speeds in an attempt to save money by cramming data onto each video cassette. Sure you can back up 330 MB onto a 120-minute video cassette by recording at a so-called "six hour speed." You just can't restore the data you've backed up, which sort of misses the point. It's easy to see the difference between the higher quality of a movie recorded at the VCR's fastest speed and the lower quality of that same movie recorded at its slowest speed. Slow speeds produce recordings with weak signal density. The result is that marginal spots on the tape behave as if they were weak spots and the number of dropouts increases dramatically. Even if you tolerate added snow and flicker in "Shall We Dance?" set the VCR up to SP when three-quarter time turns into backup time.

You should find that Imager performs well with first quality video cassettes recorded at fast speed.

Imager Tips 3, 4, 5: Three Routes to Unattended Backups

You don't want to tie up yourself or anyone else the computer is being backed up. Imager's WAKEUP utility and a programmable VCR can back up your system automatically while the humans are elsewhere. All you do is write some batch files to control the operation.

Run this one just before you leave every day and it will start Imager backing up your Drive C at 11:30 PM every night:

```
WAKEUP 23:30:00
MRBACK C:\ /S
:end
```

The first line delays execution of the second line until the right moment (WAKEUP requires 24 hour time), at which point the second line runs Imager's backup program to do all of Drive C including its subdirectories.

If you have more than one drive to back up and your VCR has cuing capability, use Imager's WAIT utility to punctuate the cassette with a little space between backups:

```
WAKEUP 03:00:00
MRBACK C:\ /S
WAIT 15
MRBACK D:\ /S
:end
```

The VCR continues to roll while the computer waits for 15 seconds after Drive C has been backed up, with the result that there's 15 seconds of blank space to mark the beginning of the Drive D backup. If only Drive D fails later, you can use the VCR's cuing capability to search for that blank space before you tell Imager to restore.

Of course you can make your batch file take the wakeup time from the DOS command line if your working days shift. Here's a relatively fancy batch program:

```
echo off
echo **              DO NOT DISTURB
echo ** THIS COMPUTER IS AWAITING BACKUP AT %1
WAKEUP %1
echo ** Now backing up Drive C
MRBACK C:\ /S
WAIT 15
echo ** Now backing up Drive D
MRBACK D:\ /S
cls
echo ** THE VCR CONTAINS AN IMAGER BACKUP CASSETTE.
echo ** Remove the cassette, store it properly, and
switch off the VCR.
:end
```

Make certain that the clocks in your computer and VCR are synchronized and set the VCR to start about a minute or two before the WAKEUP time so you're sure the tape is rolling when the computer begins pumping data to it.

ENABLE Part 4

George P. Elwood
1670 N. Laddie Court
Beavercreek, OH 45432

A Tutorial Database Management — A Beginning

In this, the fourth in the ENABLE series, we will discuss and provide an introduction to the last of the major modules in the program, database management. Of all of the parts of ENABLE, the database management module is the best. As a long time user of dBase II, both in CPM and MS-DOS, I found ENABLE's database management far easier to learn and use. This does not count the integration which I will cover in a later article. This article is based on ENABLE version 2.0, which is the Z-100 version. I will indicate both Z-100 and PC keystrokes, as necessary.

As I said, I have been a long time user of dBase II. I used this basic program to develop many application programs. These were menu driven and some where fairly complex. I had been working on an application in dBase II that involved several linked databases, menus, input forms, and report forms for two plus months and could never get it to work exactly the way I thought it should. I received ENABLE and used this application as a learning experience for the database management module. Even learning how to use the program, I completed the application in two weeks, and it worked. This program includes six input screens, ten reports, three databases and menus.

In this article on the dbms module of ENABLE, I will develop a database as a learning example. This database will show how to order Zenith computer systems that are on the DoD standards requirements contract (the Z-248 contract). The first thing to do in developing a database is to think through the problem. You must decide on fields, what you want the input screens to look like, and actually the first consideration, the reports you need. Time spent in this phase will save you time later and make the development of the database easier. Consider possible growth, as grow it will. ENABLE will let you add additional capability in the future without losing anything you have done to that point.

ENABLE writes two files with each database, a definition and the actual database information, unlike other database programs. The definition is a detailed structure of the database. All of the selections you make in the define definition section are stored here. This gives the database a tremendous capability, as will be displayed later. The two files have the same name, but different extensions. The database will have a *.DBF extension, while the definition has a *.$BF extension. Both are required. The other dbms file extensions are:

.DBF	Database file
.$BF	Database definitions
.$IF	Input form
.$RF	Report form
.SS	Select set
.SSn	Internal record processing file
.ERR	Error select set
.NDX	Index file
.@@@	Backup file
.DEF	Database def. used as DB
DBM	dbms macro
DB.MNU	dbms menu

Entry into ENABLE's dbms feature is done in the same manner as the other modules; (U)se System, (D)ata/Graphics, (D)esign, and (D)atabdata Definitions. You will then be prompted for a name. Enter "ORDER" and <RETURN>. If you enter a name that is not in the file structure, ENABLE will prompt with a "New File", "List Directory", or "Retype". <RETURN> will permit you to enter the database definition

for a (N)ew File. ENABLE then requests additional default information for the database. You can input a short description of the database that will be printed if you request it. Next comes the default input and report forms. You can leave these blank or fill them in. If you have several input or report forms, it is better to leave them blank. For this example, put in "ORDER" for both of these requests. The next screen tells you that if you change the definition, you must copy the database into the new definition. If you change the definition, ENABLE will lead you through the procedure necessary to change the database to fit the definition with a built-in macro. More about this later.

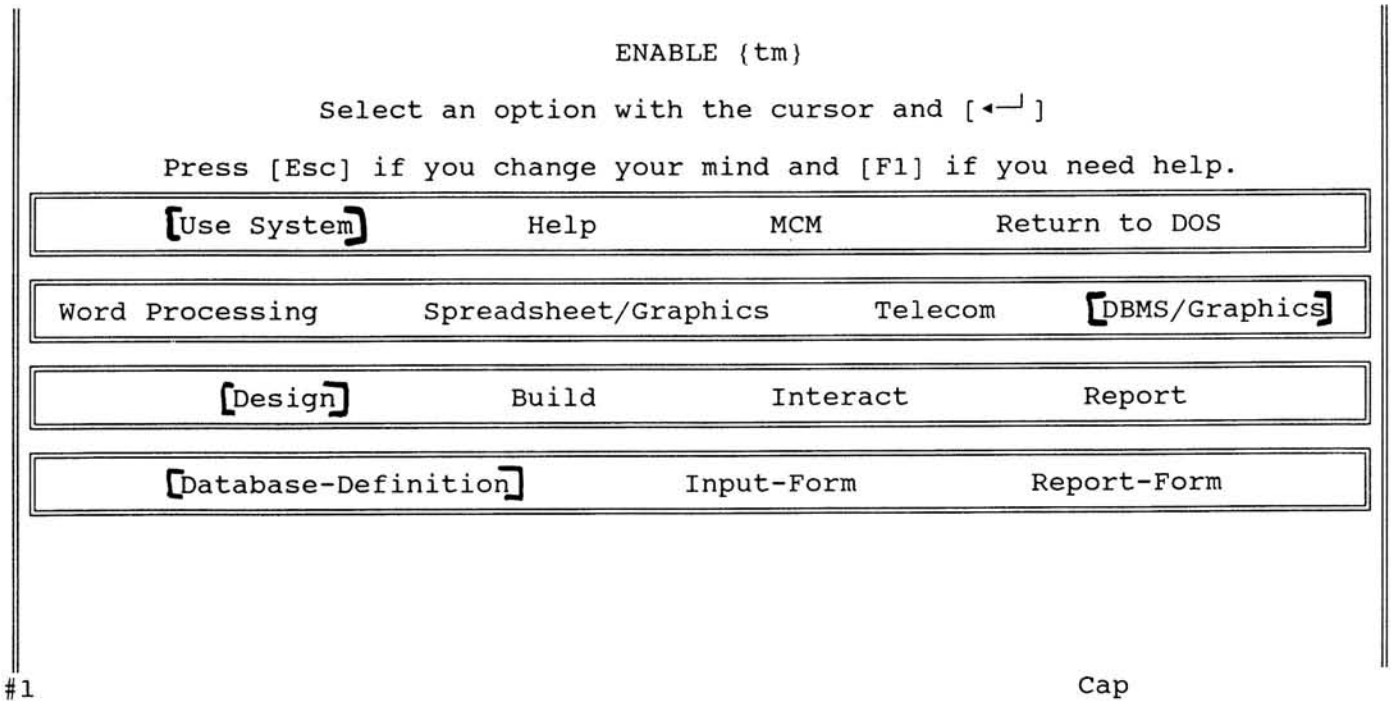


Figure 1
ENABLE Main Menu to Database Creation

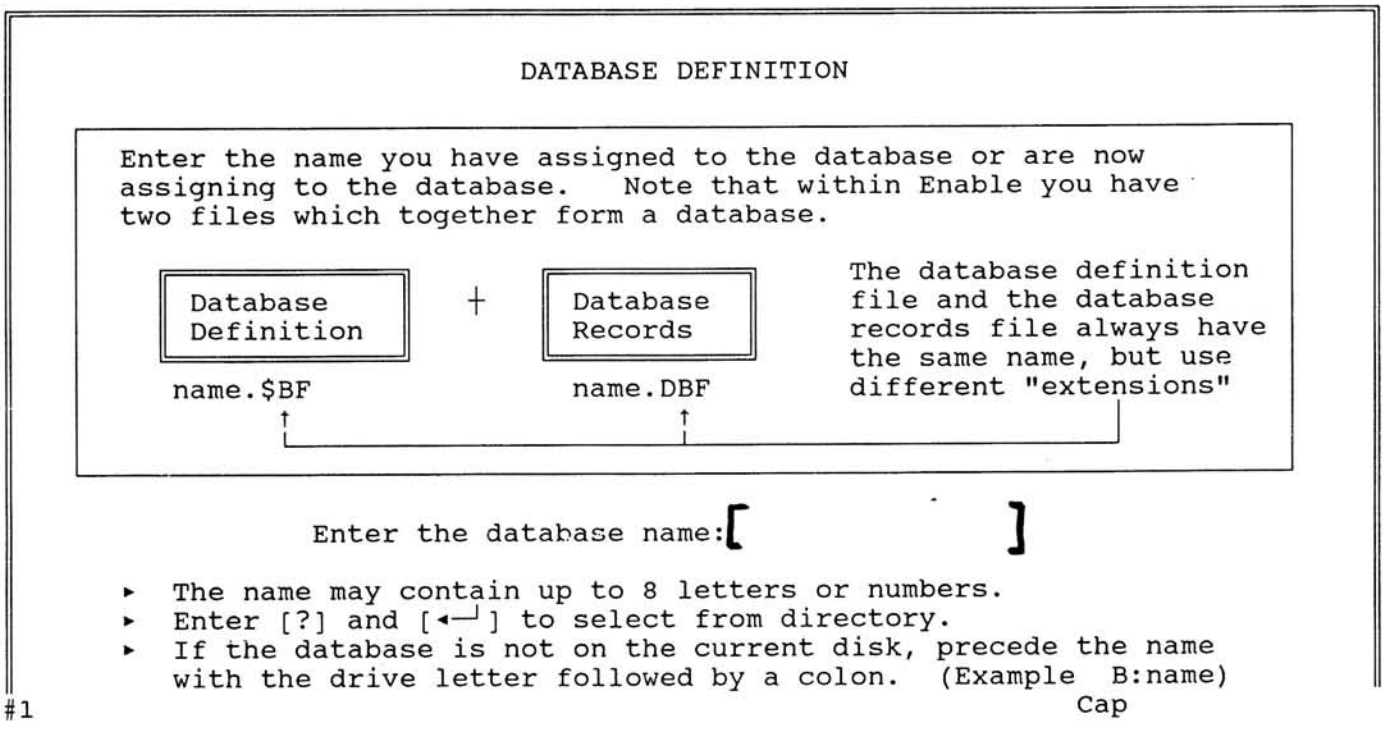


Figure 2
Database Entry Screen

```
                        DATABASE DEFINITION


     Enter database description: [Z-248 Order Form                ]

     Enter the name of the default input form:    [ORDER           ]


     ▸ The database description is for your reference and appears only
       when the database definition is displayed or printed.


#1   Definition of database B:\ORDER                    Cap       F10=Menu
```

Figure 3
Database Second Entry Screen

You are now at the ENABLE definition screen. The first field is NSN, national stock number. Type in NSN for the field name and <RETURN>. The first question on the first screen is the method of definition, (Q)uick or (D)etailed. Each question has a different path and the amount of detail varies greatly. Once you start into the definition questions and answers for the field, you must complete the entire sequence. You should not terminate in the middle by using the END key or the definition may be corrupted. If you make a mistake, you can backspace to correct it or use the ESC to back one screen. You can also complete the sequence and return and edit it.

```
 Field Name     Data     Data   Min   Max   Dec   Index   Dup   Fld   Pictures
                Source   Type   Len   Len   Pt.   File    Keys  Req   Rept Edit
   NSN          Keyboard Text     1    18          N             Y     N
   CLIN         Keyboard Text     4     6          Y       N     Y     Y
   DESCRIPT     Keyboard Text     0    25
   MODEL        Keyboard Text     0    12
   COST         Keyboard Num      1     8     0    N             Y     Y
   TYPE         Keyboard Text     0     1          Y       Y     N     Y
   DATE         System            8     8          N             N     Y      Y
   QTY          Keyboard Num      0     3

Total flds:9      Real flds:9          Total Len:85       Real Len:85

        Enter the field name:

   ▸ Enter the name of each field in your database, one at a time,
     pressing [◂┘] after each one.
   ▸ Next you will be asked for specific information about the field.
   ▸ Your responses will be summarized in the table above.
   ▸ To review, change or delete a field, press [↑] for additional prompts.
   ▸ Press [Esc] to return to page one.

#1   Definition of database B:\ORDER                    Cap       F10=Menu
```

Figure 4
Definition Screen

The (Q)uick definition is exactly that, quick. If selected, the next question is type data, (I)nteger, (D)ecimal and number of places, (T)ext or (L)ogical. It then asks for the maximum field length. These are the only questions, but for simple fields, it is sufficient.

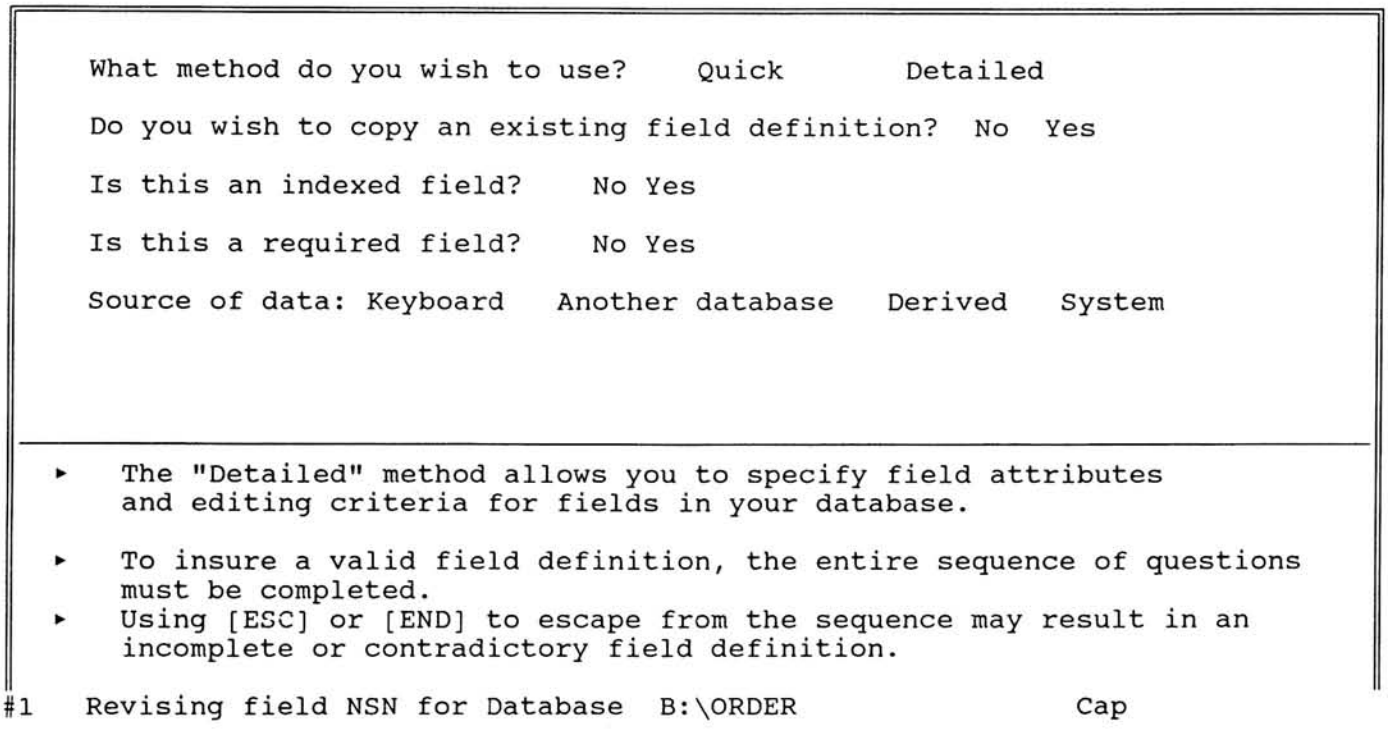


Figure 5
First Input Screen

The NSN field requires the (D)etailed definition and many more questions need to be answered. The first question permits you to copy field definitions from other fields and other databases. You are prompted for the field and database, if this is selected. This saves time if you have to enter several definitions that are the same in several different databases. The next question asks if the field is indexed. If selected, it prompts for the name of the index file, and if duplicate entries are permitted. These names must be different for the different indexes or ENABLE will get confused when it tries to use the index field. For our example, do not select index. If this field was indexed, we would select (N)o Duplicate Entries as NSN is unique to each line item. The next question asks if the field is a required field. Select (Y)es for this prompt. This answer will make ENABLE prompt you for an input. If you try and bypass this request during the data input, ENABLE will beep and say that an input is required. This can be bypassed during data entry if the TAB key is used. The last question on this first input screen is source of data. You can select (K)eyboard, (A)nother, (D)erived, or (S)ystem for this question. Select Keyboard by hitting a <RETURN>, which is the default.

The other answers to this question provide additional capability for the ENABLE database. (A)nother permits you to select a field from another database. If this is selected, another menu appears and you have more questions to answer. You are prompted for the name of the database, the look-up field, the index linking field, the linking field between the two databases, the Columnar Heading for the field, and the Error message. ENABLE does not copy the data from the other database, but uses it so that data is not duplicated in two databases saving space. In order for this to work, both databases must have an indexed field, any field. The index link is the link between the two databases and this must be added in the next question. The heading listing is not required, but if entered, will be displayed when the columnar format is selected. The Error message will be displayed on the 25th line or bottom line if the input is not correct.

(D)erived is a means to input data based on the results of other fields in the database. This can be the result of a mathematical formula or a concatenation of text fields. In one database, I use the catenation of two fields by using the formula (lname)-&(address). This puts the last name and address together as a unique field. If selected, you are prompted for the formula. You have 136 spaces in which to put this formula. Next, you can select the type of data that will result, (N)umeric, (T)ext, or (L)ogical. If numeric is selected, you can specify the number of decimal places in the result. (L)ogical permits you to select the statement for the true condition, (Y)es or (T)rue. Numeric or text selection permits you to specify the length of the field, enter the report picture, and the columnar heading.

(S)ystem can be used to input the system date or time. The System inputs can be selected by date or time of (A)ddition-of-record or (E)dit-of-record. The Addition-of-record will only show the date/time the record was added. The Edit selection will update every time the record is edited. You can select data or time and specify the report picture using these formats. For the date, you can select using DD for days, MM for months, and YY for years along with separators, if desired. If time is selected, you can use HH for hours, MM for minutes, SS for seconds and x to indicate AM or PM. Additionally, you can select HM which is the military 24 hour clock.

The response to these questions move you on to the next menu screen. This screen adds additional parameters to the field. The first question is for the type of data; (N)umeric, (T)ext, (L)ogical, or (O)ther. For our example, select (T)ext even though NSN is numbers, it does

have special characters in it (-) as separators. Numeric is for number only. Logical fields can be further defined with the type answer you want, (Y)es or (T)rue for the type response to the question.

If you select (N)umeric, you can select (I)nteger, (D)ecimal, or (C)urreny. Decimal prompts you for number of places. Currency indicates that "$" and commas are permitted.

(O)ther brings up five additional responses and further highlights the capability of the ENABLE database. This option provides a means to error check inputs to the database. The first choice is (D)ate where you can select a date format for the input. The inputs permitted are the standard ENABLE date formats. ENABLE suggests that you use YY/MM/DD so that you can sort on the field, if necessary. You can put in an error message, so if the data is entered wrong, ENABLE will beep and your message will be displayed on the 25th line for the Z-100 or the bottom line for the PC.

The next choice is (T)ime. Again, you can select the format for the input using the standard format. These are the same type inputs as in the spreadsheet. Again, you can insert an error message if you so desire and ENABLE will display it if the data is input wrong.

The next choice is (S)tate-Code. This is the post office two alpha designator for the states. If you select a wrong State code, ENABLE will beep. This is used in conjunction with the next option, (Z)ip Codes. If you input a State code and then try to input a ZIP code that does not exist in that State, ENABLE will beep and so indicate. You have the option of either a 5- or 9-digit ZIP code.

The last option is the (P)hone-No. You can select numbers to include Country codes and extensions. You select the format you want by using CC for country code, AAA for area code, NNN-NNNN for the number and XXX for the extension. This can be depicted as CCC-AAA-NNN-NNNN/XXX. Again ENABLE does error checking using the State code and Area code. If you insert a wrong area code in the data field, ENABLE will beep and so indicate.

```
Select the data type:     Numeric      Text     Logical     Other

Your choices:     Date    Time    State-Code    Zip-Code    Phone-No.

Enter the edit picture for the phone number:

Enter the minimum length of the phone edit picture:  1
```

```
▸  Enter the format Enable should use to display telephone numbers.
          Example:  CC-AAA-NNN-NNNN/XXX

       Use "CC" for country code,       "AAA"  for the area code,
           "N"  for each digit of the phone number and
           "X"  for each digit of the extension number.
       Use  ( ) :  / \ -  .   to separate elements.

#2   Revising field NSN for Database  B:\ORDER                    Cap
```

Figure 6
Definition Screen Two

```
What text is permitted?    Anything   Letters   Numbers+Letters   Special

Does field have minimum and maximum values?      No    Yes
Does field have a list of acceptable values?     No    Yes

Enter the minimum length of this field:  1
Enter the maximum length of this field:  18
```

```
▸ This field may contain letters and numbers, or letters only, plus
  the punctuation you specify.

#1   Revising field NSN for Database  B:\ORDER                    Cap
```

Figure 6a
Definition Screen Three

```
Enter report picture:

Columnar report heading:

Error msg:

▸ Enter the format Enable should use to display data from this field.
  Type an "X" to indicate an alphanumeric character,
          "U" for upper case letters,
          "L" for lower case letters,
  and punctuation where you want it. See the documentation for more choices

#1   Revising field NSN for Database  B:\ORDER                    Cap
```

Figure 6b
Definition Screen Four

The next choice is minimum and maximum value for the field. This option permits you to establish minimum and/or maximum values for the input. You can also select minimum and maximum reasonable values for the input. This can be used for numeric, date, and times. An example would be a request for a date. During the data definition, you indicated the earliest possible date and the probable earliest date. Entries before this time will cause ENABLE to beep and display either your error message or the ENABLE default message.

The next choice permits you to enter acceptable values. This can be used for any inputted data. You type in the acceptable responses separated with a comma.

Next, you select the minimum and maximum length of the field. If you indicated that the field was required, a minimum length of one (1) is necessary.

The last selection screen asks for the report picture. You can select the format for the report using X for alphanumeric, U for upper case, L for lower case, and various punctuations. You can provide another header for the columnar report format if you want something more than the field name. The last choice is Error message. This is the message that will appear on the bottom line of the screen if a wrong entry is made.

To review, for the NSN field definition you should have selected (D)etail, (N)o, (N)o, (Y)es, (K)eyboard, (T)ext, (A)nything, 1, 18, <R>, <R>, and <R>.

The remainder of the fields are completed with the following responses:

```
NSN      D N N N K T A N N 1<> 15<>
         <> <> <>
CLIN     D N Y CLINX<> N Y K T A N N
         4<> 6<> NNNNUU<> <> <>
DESCRIP  Q T 25<>
MODEL    Q T 12<>
COST     D N N Y K N C 0<> N N 1<>
         8<> $NN,NNN<> <> <>
TYPE     D N Y TYPEX<> Y N K T L <>
         B N N Y H,S,h,s<> 0<> 1<> U <>
         "Must be either "H" or "S". <> <>
DATE     D N N N S ADD D DD MON YY<>
         <> <>QTY Q I 3<>
ORDERNO  Q T 4<>

"<>" indicates a RETURN or ENTER.
```

Before leaving the definition area, let's go back into one of the fields and review it to simulate an edit. This procedure is easy in ENABLE. From the definition screen, push

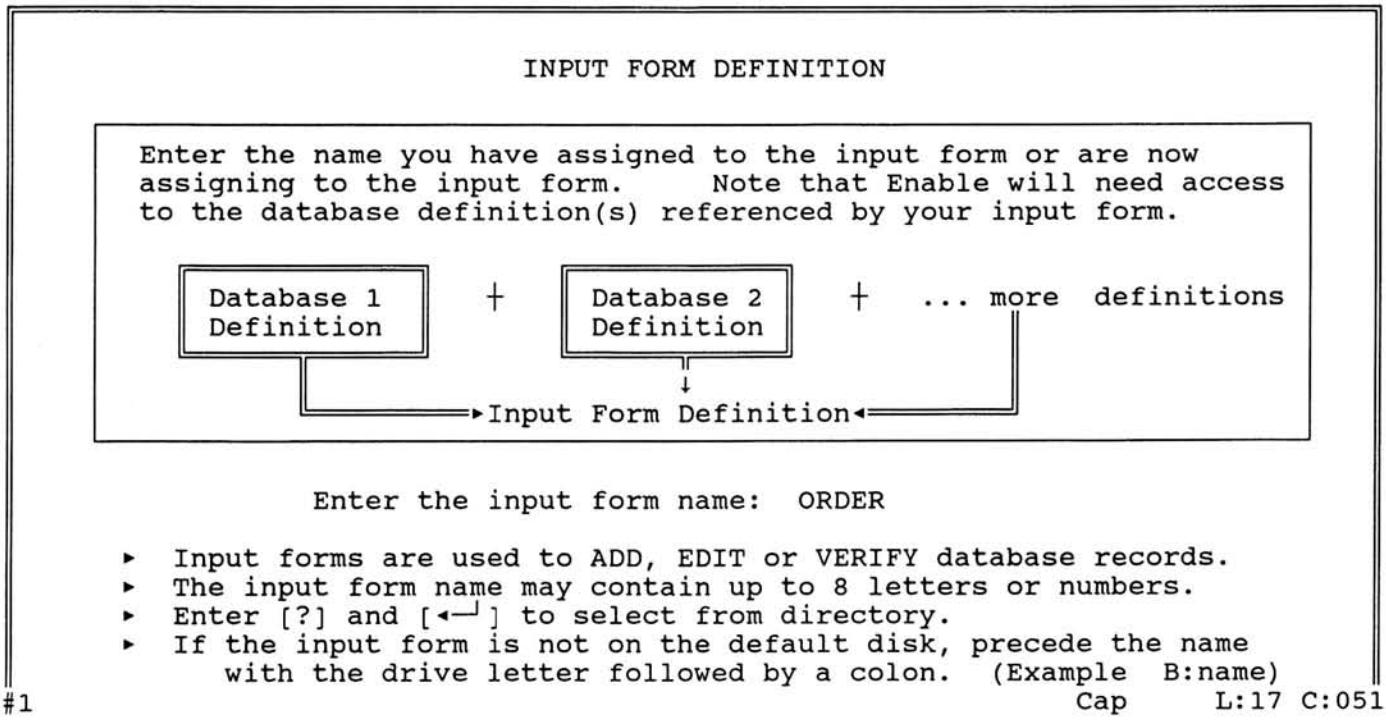


Figure 7
Definition Screen with Inputs

the up arrow. The highlighted area will move to the first field on the list. By using the arrow keys, you can move up or down the list of fields. By pressing <RETURN>, the definition for the selected field will be brought back on the screen. You can change any of the selections at this time. Again, you must continue through the entire sequence. All of these responses seem like a lot of work, and at the time it is. There is no way to bypass all of these questions. These questions give ENABLE dbms its power and make it easy to use.

This completes the database definition for "ORDER". Like all of ENABLE functions, the F10 key brings up the top line menu. You select (S)ave and the definition is saved to the disk selected in the PROFILE unless you have selected another drive and/or directory. The other options are (P)rint, (D)elete, or (Q)uit. Print is only available with version two and is quite detailed. If you have many fields, this will take a lot of paper. The print out details all of the options you have selected, either yes or no and responses if required.

(D)elete will remove the definition so ENABLE will prompt to be sure that's what you want. (Q)uit will take you back to the main menu. You will be prompted if you are sure. If you select Yes, you will return to the main menu and the definition will be removed. Make sure you save before you quit.

It is now possible to enter data into the database. ENABLE will use a default input form for this input. To use this form, type (U)se System, (D)atabase/Graphics, (B)uild. ENABLE will present an add screen. It prompts you for the the name of the database. As in all of ENABLE, if you do not remember the name of the file, type in a "?" and ENABLE will present all of the available *.DBF files on the disk, even if it is only a *.$BF definition. Select "ORDER" as the file to use. ENABLE will then prompt you for a "C"reate to make a default input form. This form is a basic list-

ORDER DATABASE DEFINITION

```
DATABASE DESCRIPTION: Z-248 Contract listing
DEFAULT INPUT FORM:   ORDER
DEFAULT REPORT FORM:  ORDER
-------------------------------------------------------------------------------
FIELD: NSN              METHOD: Detailed    SOURCE OF DATA: Keyboard
REQUIRED?: No  INDEXED?: No  ALLOW DUPLICATES?: No  INDEX FILE:
DATA TYPE   NUMERIC TYPE   TEXT TYPE        LOGICAL TYPE   OTHER TYPE
Text                       Anything
MINIMUM LENGTH: 1  MAXIMUM LENGTH: 15  DECIMAL PLACES:
MINIMUM AND MAXIMUM VALUES?: No
LIST OF ACCEPTABLE VALUES?: No
EDIT PICTURE                           REPORT PICTURE
```

```
REPORT HEADING:
ERROR MESSAGE:
-------------------------------------------------------------------------------
FIELD: CLIN              METHOD: Detailed     SOURCE OF DATA: Keyboard
REQUIRED?: Yes  INDEXED?: Yes  ALLOW DUPLICATES?: No  INDEX FILE: CLINX
DATA TYPE   NUMERIC TYPE   TEXT TYPE          LOGICAL TYPE   OTHER TYPE
Text                       Anything
MINIMUM LENGTH:  4  MAXIMUM LENGTH:  6  DECIMAL PLACES:
MINIMUM AND MAXIMUM VALUES?:  No
LIST OF ACCEPTABLE VALUES?:  No
EDIT PICTURE                           REPORT PICTURE
                                       NNNNUU

REPORT HEADING:
ERROR MESSAGE:
-------------------------------------------------------------------------------
FIELD: DESCRIP           METHOD: Quick        SOURCE OF DATA: Keyboard
REQUIRED?: No  INDEXED?: No  ALLOW DUPLICATES?: No  INDEX FILE:
DATA TYPE   NUMERIC TYPE   TEXT TYPE          LOGICAL TYPE   OTHER TYPE
Text
MINIMUM LENGTH:  0  MAXIMUM LENGTH:  25  DECIMAL PLACES:
MINIMUM AND MAXIMUM VALUES?:  No
LIST OF ACCEPTABLE VALUES?:  No
EDIT PICTURE                           REPORT PICTURE

REPORT HEADING:
ERROR MESSAGE:
-------------------------------------------------------------------------------
FIELD: MODEL             METHOD: Quick        SOURCE OF DATA: Keyboard
REQUIRED?: No  INDEXED?: No  ALLOW DUPLICATES?: No  INDEX FILE:
DATA TYPE   NUMERIC TYPE   TEXT TYPE          LOGICAL TYPE   OTHER TYPE
Text
MINIMUM LENGTH:  0  MAXIMUM LENGTH:  12  DECIMAL PLACES:
MINIMUM AND MAXIMUM VALUES?:  No
LIST OF ACCEPTABLE VALUES?:  No
EDIT PICTURE                           REPORT PICTURE

REPORT HEADING:
ERROR MESSAGE:
-------------------------------------------------------------------------------
FIELD: COST              METHOD: Detailed     SOURCE OF DATA: Keyboard
REQUIRED?: Yes  INDEXED?: No  ALLOW DUPLICATES?: No  INDEX FILE:
DATA TYPE   NUMERIC TYPE   TEXT TYPE          LOGICAL TYPE   OTHER TYPE
Numeric     Currency
MINIMUM LENGTH:  1  MAXIMUM LENGTH:  8  DECIMAL PLACES:  0
MINIMUM AND MAXIMUM VALUES?:  No
LIST OF ACCEPTABLE VALUES?:  No
EDIT PICTURE                           REPORT PICTURE
                                       $NN,NNN

REPORT HEADING:
ERROR MESSAGE:
-------------------------------------------------------------------------------
FIELD: TYPE              METHOD: Detailed     SOURCE OF DATA: Keyboard
REQUIRED?: No  INDEXED?: Yes  ALLOW DUPLICATES?: Yes  INDEX FILE: TYPEX
DATA TYPE   NUMERIC TYPE   TEXT TYPE          LOGICAL TYPE   OTHER TYPE
Text                       Letters
                           Blanks OK?: No
                           Case Allowed: Both
                           Permitted Punctuation:
MINIMUM LENGTH:  0  MAXIMUM LENGTH:  1  DECIMAL PLACES:
MINIMUM AND MAXIMUM VALUES?:  No
LIST OF ACCEPTABLE VALUES?:  Yes
LIST:  H,S,h,s
EDIT PICTURE                           REPORT PICTURE
```

```
REPORT HEADING:  Must be either "H" or "S".
ERROR MESSAGE:
--------------------------------------------------------------------------------
FIELD:  DATE              METHOD:  Detailed     SOURCE OF DATA:  System Date
                                                System data supplied during ADD
REQUIRED?:  No  INDEXED?:  No  ALLOW DUPLICATES?:  No  INDEX FILE:
DATA TYPE    NUMERIC TYPE    TEXT TYPE          LOGICAL TYPE    OTHER TYPE

MINIMUM LENGTH:  8  MAXIMUM LENGTH:  8  DECIMAL PLACES:
MINIMUM AND MAXIMUM VALUES?:  No
LIST OF ACCEPTABLE VALUES?:  No
EDIT PICTURE                            REPORT PICTURE
YY/MM/DD                                DD MON YY
REPORT HEADING:
ERROR MESSAGE:
--------------------------------------------------------------------------------
FIELD:  QTY               METHOD:  Quick        SOURCE OF DATA:  Keyboard
REQUIRED?:  No  INDEXED?:  No  ALLOW DUPLICATES?:  No  INDEX FILE:
DATA TYPE    NUMERIC TYPE    TEXT TYPE          LOGICAL TYPE    OTHER TYPE
Integer
MINIMUM LENGTH:  0  MAXIMUM LENGTH:  3  DECIMAL PLACES:
MINIMUM AND MAXIMUM VALUES?:  No
LIST OF ACCEPTABLE VALUES?:  No
EDIT PICTURE                            REPORT PICTURE

REPORT HEADING:
ERROR MESSAGE:
--------------------------------------------------------------------------------
FIELD:  ORDERNO           METHOD:  Quick        SOURCE OF DATA:  Keyboard
REQUIRED?:  No  INDEXED?:  No  ALLOW DUPLICATES?:  No  INDEX FILE:
DATA TYPE    NUMERIC TYPE    TEXT TYPE          LOGICAL TYPE    OTHER TYPE
Text
MINIMUM LENGTH:  0  MAXIMUM LENGTH:  4  DECIMAL PLACES:
MINIMUM AND MAXIMUM VALUES?:  No
LIST OF ACCEPTABLE VALUES?:  No
EDIT PICTURE                            REPORT PICTURE

REPORT HEADING:
ERROR MESSAGE:
--------------------------------------------------------------------------------

REAL FIELDS:  9  TOTAL FIELDS:  9  REAL LENGTH:  83  TOTAL LENGTH:  83
 (Total length does not include the length of any external fields.)
```

Figure 6c
Part of Definition Printout

ing with the field name and a highlighted area the size of the field. This is about as basic as you can get. This input form is very similar to the dBase II basic input form. Note that three fields already have data in them. Date, which is a system input, should present the current date, and the two number fields, cost and quantity should reflect 0 as there is no data. To get out of this form, press the F10 key to get the top line menu.

The top line menu displays the following: (N)ext, (P)revious, (S)ave, (I)gnore, (1)-Query, (2)-Index, (W)here, and (Q)uit. Note that as you move across the top line, the function key expert commands are displayed. Input data at this time. If you don't have actual data, put in anything. If you try and put in data that is not compatible, ENABLE will let you know. After you fill in the data for the first record, press F10 and (N)ext or just a <RETURN>, as Next is the default value for the next record. Press Quit to get out of the input form without saving anything that you put in the form, if you have not added any records. Note that on the status line on the bottom of the screen, the following items are displayed. Starting on the left side of the screen, first comes the Window number, then the total number of records in the database, record number you are on, the number of selected records, and current number of records in the group you are working on. As you enter records, the numbers all should count up. If you enter using edit or a where clause, the numbers will reflect different numbers.

From the top line menu, (P)revious will move to the previous record in the database. (S)ave will save the records that have been entered on the disk. This should be accomplished occasionally to protect the data you have entered. Now that you have entered some data from the

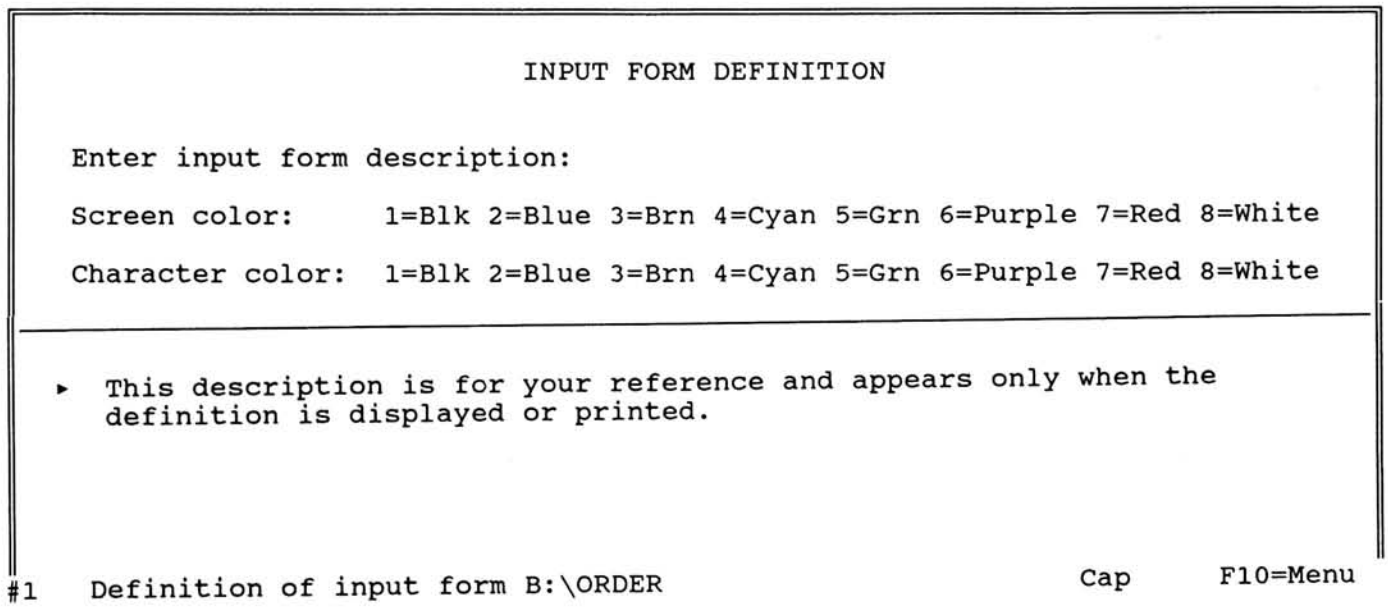


Figure 8
Basic Input Form Definition Screen

top line menu, press (Q)uit. You are returned to the Add menu. Press ESC and you are back to the database selection menu. This is the menu you use to perform most database functions. Select (Q)uit to return to the main menu.

Even though the basic input form meets some needs, ENABLE has a very neat capability to generate custom input forms. This capability makes ENABLE very easy to use and makes the input form look professional. You don't have to tell anybody that you created it in a few minutes. Most input forms can be created in less than five minutes, but then I have spent over 45 minutes on one, but then it had over two hundred entries and block graphics in it.

To get to the input form design, press (U)se System, (D)atabase/Graphics, (D)esign, and (I)nput form. ENABLE will prompt for a form name. Type in ORDER, the next screen will prompt for the name of the database to be used with the form. Again, type in "ORDER". The next screen will prompt for a brief description of the input form. You also get to select the color of the screen and character color. The next screen is the input form information screen. This screen provides information on how to use the input form generation capability. Pressing <RETURN> will get you to the actual form. Since you used the basic input form in the previous section, this form will be displayed. You are now in the word processor, so all those functions are available.

Since we want to develop an entirely new input screen, we will block the entire screen and delete it. ENABLE's integration permits these multiple uses of the functions which makes everything you learn helpful. To delete the default form, mark the entire screen by placing the cursor in the top left corner of the screen and press the F7 key. Move the cursor to the bottom right corner and press the F7 key again. The text part of the screen should become highlighted. Press F9 DEL CHR B (F9 Del B for the PC) and the screen should be cleared. Now, for this example, type in "Z-248 Contract Input Form" on line two. Press F0 F4 (ALT/F4 for the PC) and the text will be centered. Now to add a little to the screen, we will use the block character set. F10 will bring the top menu up. This menu is slightly different in that it has only two basic inputs, Database Options and Word Processing Options. Select (W)ord Processing, (L)ayout, (7) Use Special characters, (1) Box character set. The character set will be displayed on the bottom of the screen. Use whatever line characters you like. When you are done, press F10, (W)ord processing, (L)ayout and then the cursor keys to select (7) and (6) to turn off the character set.

At a location that you deem appropriate, type in "National Stock Number." Now type SHIFT/F9 after the above entry. ENABLE prompts for the field name. If you do not remember the fields in the database, again ENABLE's standard format only requires a "?" and all fields are displayed. Use the cursor keys to select NSN and a <RETURN>. Several more screens are now available. The first request if special processing is required. If the answer is (N)o, you are back to the input form. If the answer is yes, three more screens are presented. The first asks for screen color, character color (these are different than the basic database definition colors), should the field be verified (do you have to retype the data before it is accepted), and should the field be copied to the next add screen (if the data remains the same this will save time). The next screen permits you to protect the field from update, require the field have a <RETURN> to move to the next field, and to provide for default data. The last screen provides a means to input a special message if invalid data is entered, provides a means for the input form to go to a different field rather than the next one in line, and it finally asks if you have macros to processes.

Continue this process for the CLIN, DESCRIP, MODEL, COST, and TYPE. I used two fields per row and centered them. Because the word processor is being used, additional information can be added, if necessary, during this process. As an example, add "(Hardware/Software)" as a help for type. Any of the word processing attributes can be used in this form. This whole input form takes about two minutes to complete. We did not add date because it is input by the system, and quantity and Orderno because they are not needed. To edit any of the

```
                You are now ready to revise or create your input form.

    1.  Place a field label on the screen using ANY of Enable's
        word processing features.  The label identifies where the data
        is to be keyed.

    2.  Position the cursor where you want the data in the field to
        begin.  Press [Shift] and [F9] to tell Enable to "Put It Here".

    3.  Answer the questions Enable will ask about the field.

    4.  Repeat the process for each field in your input form.  If you
        wish to delete a field after you have positioned it on the screen,
        or if you wish to revise a field's options, press [F10] for
        instructions.

    5.  When you're finished, press [F10] to save the form using
        the Save options or to return to the Main Menu.

    6.  Press any key when you're ready to begin and Enable will
        display a blank screen or your previously-created input form.

#1  Definition of input form B:\ORDER                    Cap      F10=Menu
```

Figure 9
Input Form Help Screen. Shows Before Entry into Actual Screen.

```
                          Z-248 Input Form

          National Stock Number _________________  CLIN _____

        Description _____________________  Model ___________

              Cost _______  Type (Hardware/Software)




#1  Definition of input form B:\ORDER                    Cap      L:01 C:001
```

Figure 10
Designed Input Form

fields, move the cursor over the diamond located in the first position of the field. You can then edit the field with the same response as listed above.

To save the form, press F10, (F)orm Design Option, and (S)ave. To leave the form, again press F10, (F)orm Design Option, and (Q)uit. To use the designed input form; (U)se System, (D)atabase/Graphics, and (B)uild. Type in the "ORDER" for the database, and "ORDER" as the input form. The designed form is now presented on the screen. Fill in the data as required using the F10 key and (N)ext at the end of each record. When you have finished the data input, (S)ave the records and (Q)uit to the main database screen, and (Q)uit again back to the main screen.

This completes the introduction to ENA-

BLE's database. We have completed a database definition and built an input screen. From these two exercises, some of the power of ENABLE has been presented. The definition phase showed the level of error checking that is available within ENABLE. The questions asked may seem like a lot to do, but then it is worth it when it comes to data entry by people not familiar with computer operations and databases. We also built a simple input screen using the word processor and "point and shoot" capability of ENABLE. This one feature will save many hours of time when it comes to developing input screens when compared to dBase.

The next article will return to the word processing for an intermediate level presentation. If you have questions on any parts of ENABLE that I have covered, please write it down and mail it to me, and I will provide a response as soon as possible.

*


"Why Haven't You Called?"
"I've stayed up 24 hours a day waiting to hear from you. I know that for most of you, this may be your first time, but I promise, I'm very gentle! If you're still unsure, let me tell you what I have to offer. First, there's my message base. Here's where my users exchange ideas, receive assistance, and sell things. Next, is my database. Most of my users will tell you, it's second to none! With over 30 megabytes of hand-picked software it caters to all Heath/Zenith computers, but mostly PC compatibles. Finally, I have something new for you, my 'Bargain Centre'. Here you can buy surplus software and hardware, at unheard of prices. Interested? I hope so! Set your modem to either 300, 1200, or 2400 baud, and call me right now at (616) 982-3956, and register today. If you're still a bit shy, you can still call my human at (616) 982-3837 and register with him. Although he talks at 150 baud, he's gentle, too!"
M&C

Heath/Zenith Users' Group

P.O. Box 217
Benton Harbor, MI 49022-0217

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

POSTMASTER: If undeliverable, please do not return.

P/N 885-2098