

# The COMPUTER JOURNAL

J Programming - User Support  
Applications

Issue Number 53

November/December 1991

US\$3.95

The CPU280

Local Area Networks

An Arbitrary Waveform Generator

Real Computing

Zed Fest '91

Z-System Corner

Getting Started in Assembly Language

The NZCOM IOP

Z-Best Software

The Computer Corner

# Now \$4.<sup>95</sup> Stops The Clock On Over 100 GENie Services

For the first time ever, enjoy unlimited non-prime time\* usage of many popular GENieSM<sup>1</sup> Service features. For just \$4.95 a month. Choose from over 100 valuable services including everything from electronic mail and stock closings to exciting games and bulletin boards. Nobody else gives you so much for so little.

You can also enjoy access to a wide variety of features like software libraries, computer bulletin boards, multi-player games, Newsbytes, and the Computer Assisted Learning Center (CALC) for just \$6.00 per non-prime hour for all baud rates including 2400. That's less than half of what some other services charge. Plus with GENie there's no

sign-up fee.

Now GENie not only gives you the information and fun you're looking for. But the time to enjoy them, too.

Follow these simple steps.

1. Set your modem for half duplex (local echo), at 300, 1200 or 2400 baud.

2. Dial toll free 1-800-638-8369.

Upon connection, enter HHH.

3. At the U#=prompt, enter XTX99486, GENIE then press RETURN

4. Have a major credit card or your checking account number ready.

For more information in the U.S. or Canada, call us voice at 1-800-638-9636.

*TCJ* readers are invited to join us in the CP/M SIG on page 685 and the Forth Interest Group SIG on page 710. Meet the authors and editors of *The Computer Journal!* Enter "M 710" to join the FIG group and "M 685" to join the CP/M and Z-System group.

We'll meet you there!

**JUST \$4.95**

**Moneyback**

**Guarantee**

**Sign up now. If you're not satisfied after using GENie for one month we'll refund your \$4.95.**

## The Computer Journal

### Founder

Art Carlson

### Editor/Publisher

Chris McEwen

### Technical Consultant

William P. Woodall

### Contributing Editors

Bill Kibler

Matt Mercaldo

Tim McDonough

Frank Sergeant

Clem Pepper

Richard Rodman

Jay Sage

*The Computer Journal* is published six times a year by Socrates Press, P.O. Box 12, S. Plainfield, NJ 07080. (908) 755-6186

Opinions expressed in *The Computer Journal* are those of the respective authors and do not necessarily reflect those of the editorial staff or publisher.

Entire contents copyright © 1991 by *The Computer Journal* and respective authors. All rights reserved. Reproduction in any form prohibited without express written permission of the publisher.

**Subscription rates\*** Within US: \$18 one year (6 issues), \$32 two years (12 issues). Foreign (surface rate): \$24 one year, \$44 two years. Foreign (airmail): \$38 one year, \$72 two years. All funds must be in U.S. dollars drawn on a U.S. bank.

Send subscription, renewals, address changes, or advertising inquiries to: *The Computer Journal*, P.O. Box 12, S. Plainfield, NJ 07080, telephone (908) 755-6186.

### Registered Trademarks

It is easy to get in the habit of using company trademarks as generic terms, but these trademarks are the property of the respective companies. It is important to acknowledge these trademarks as their property to avoid their losing the rights and the term becoming public property. The following frequently used trademarks are acknowledged, and we apologize for any we have overlooked.

Apple II, II+, Iie, Iie, Usa, Macintosh, DOS 3.3, ProDoe; Apple Computer Company. CP/M, DDT, ASM, STAT, PIP; Digital Research. DateStamper, Sack-Grounder II, Dos Disk; PluPerfect Systems. Clipper, Nantucket; Nantucket, Inc. dBase, dBASE II, dBASE III, dBASE III Plus, dBASE IV; Ashton-Tate, Inc. MBASIC, MS-DOS, Windows, Wad; Microsoft WordStar; Micro-Pro International. IBM-PC, XT, and AT, PC-DOS; IBM Corporation. Z80, Z280; Zilog Corporation. Turbo Pascal, Turbo C, Paradox; Borland International. HD64180; Hitachi America, Ltd. SB180; Micromint, Inc.

Where these and other terms are used in *The Computer Journal*, they are acknowledged to be the property of the respective companies even if not specifically acknowledged in each occurrence.

# TO / The Computer Journal

11th Issue Number 53 November / December 1991

Editor's Desk.....2

Reader-to-Reader .....2

The CPU280..... 3

When 8 Bits Aren't Enough

By Tilmann Reh.

Local Area Networks..... 6

Broadband Cabling

By Wayne Sung.

An Arbitrary Waveform Generator..... 9

Using the Harris RTX2001A.. Part Two

By Jan Hofland.

Real Computing..... 17

Mach, Desqview/X., RBOCs, C Sickness, HST Modems,

Metal and More on Minix

By Rick Rodman.

Zed Fest'91 .....20

Where Beer and BIOS Meet

By Lee Bradley with Commentary by David Cottrell.

Z-System Corner. .... 21

User Groups in Germany, Using the NZCOM Virtual

BIOS, More Programming for Compatibility

By Jay Sage.

Getting Started in Assembly Language ..... 27

Implementing Functions with Structured Programming

By A.E. Hawley.

The NZCOM IOP.....31

A General-purpose IOP Loader Module

By Terry Hazen.

Z-Best Software..... 38

Spotlight on Gene Pizzetta:

By Bill Tishey.

The Computer Corner..... 48

By Bill Kibler.

---

---

# Editor's Desk

By Chris McEwen

---

---

## Don't You Love Euphemisms?

These are lean times. You have heard that companies need to do more with less. This is a euphemism for giving more work to fewer people and aptly describes the office I occupy during my "other life." I spent the months of July through October out of town, with just a few weeks interspersed here and there..

This is both an explanation for my delay in getting this issue to you and its smaller size. I had grown fond of the sixty-four page format, enough so that I intend to return to it. But for now, we must suffice at forty-eight pages. We can hardly do more with fewer people; I am the entire production staff! I am back now, and things are returning to normal — or what passes for such in these parts..

## No One Called the Client

I spent the last few months working on everything but computers — at least what the world thinks of as computers. I spent a week at the National Automatic Merchants' Association trade show in Chicago in early October. That's vending machines, nickel-robbers for the uninitiated. These days, the machines will gladly relieve you of your dollar bills. Five-spot robbers are just around the corner.

In looking at the new wonders of the vending machine world, I found myself speaking to the vice president of engineering of a leading manufacturer. Curiosity got the best of me and I had to ask, "What processor do you use to control these things?" The answer was no surprise: Motorola 68HCH's. And what do they program in? "C, of course."

Of course.

Here we have folks producing equipment that is made-to-order for Forth and they are using C. This is not unusual. You hear it everyday. But the reason Forth was not considered floored me. A Forth vendor had never approached them, and they had no idea what it could offer.

If manufacturers of classic embed-

ded control applications as vending machines have never been approached, what other markets has the Forth world ignored?

There is an ongoing discussion over the Internet about Forth's saleability. Emotions run high over the value of ANSI standards, the lack of function libraries and incompatibilities between different Forth systems. Shucks, all this is meaningless drivel if no one calls the client!

## Other Worlds

I have mentioned the Internet, Fidonet, GENie and other forms of digital telecommunications several times over the

*See Editor, page 26*

---

---

## Reader-to-Reader

I have a suggestion. To help me locate the proper issue number when the *TCJ's* are in a bookshelf, I've taken to marking the issue number on the spine in quite large numbers centered 3 1/2" from the top. The orientation is sideways (if the issue was lying flat on the table with the spine toward you, front cover up, the issue number would be right side up). The digits are about 3/8" high and, since they wrap around both sides of the spine, they visually reduce to an apparent and still readable 3/16" high or so. You might consider adding an issue number to your cover in a similar format for easier reference.

T.H. Los Gatos, CA

*Most magazines employ "perfect binding" where the pages are glued to a backplane to accomplish what you want. I decided against this for two reasons: the*

*pages will not lay flat with the book open, making it more difficult to copy listings and the binding cost is much higher. TCJ is a hold-out in publishing source code, and we run on a very tight budget, so perfect binding was ruled out.—Ed.*

—

I am a long-distance member of Morrow Atlanta Users Group and receive their Mor-Atlanta News, where I read your introduction to *TCJ*. I am not sure if your "free issue" offer will extend to chaps like me sitting under palm trees in the South Pacific! I have enclosed my last remaining US \$1.00 note to cover reply postage.

P.D., Rotorua New Zealand

*I spent your dollar at the post office mailing off your first copy for a trial subscription. When you get the invoice, just deduct the dollar you already sent.*

*See Reader, page 30*

---

---

*Letters to the editor and other readers are welcome. Submit to The Computer Journal, Post Office Box 12, South Plainfield, NJ 07080-0012. Letters may also be electronically submitted via Internet to "cmcewen@gnat.rent.com," via GENie™ to "TCJ\$" or to Socrates Z-Nodeat (908) 754-9067. Submission implies permission to publish your letter unless otherwise stated. Letters may be edited as necessary.*

---

---

# The CPU280

## When 8 Bits Aren't Enough

By Tilmann Reh

---

---

### The History

When Zilog first introduced the Z800 MPU in their 1983/84 data book, I was working with a homebrew Z80 system based on ECB-bus EuroCards (a well-established standard here in Europe these days). My system was running with the usual 4 MHz clock, but my CP/M-Plus BIOS already had some very nice advantages. For example, I had developed the AutoFormat system, a technique which supports a wide variety of disk formats automatically. The basic principle is the same as what MS-DOS uses: the disk contains a parameter block which holds all information necessary to process it. Another feature was the automatic installation of and adaptation to various peripherals. Since several some people around here had similar machines, none of them had to configure hardware parameters (as long as they didn't need very special things).

When I had a look at the 'preliminary' specifications of the Z800, I began thinking about some performance improvements for my system. So I waited for this chip to become available. I was still waiting in 1985 when Hitachi's HD 64180 suddenly appeared. This processor had much of what the Z800 was supposed to offer, but it was actually available! So I gave up waiting any longer and developed a 64180-based single board computer. It had the 64180 MPU (with both serial channels used), 32 KB EPROM and 512 KB DRAM, a 765-type floppy controller for up to four drives, and an ECB bus interface on a EuroCard. The clock frequency was 9.216 MHz, which made an effective Z80 speed of about 11 to 12 MHz. Since the ECB bus cannot handle these frequencies, the bus clock was only half the CPU clock (4.6 MHz), with timing signals stretched to meet standard Z80 peripheral requirements. My new 'CPU180' system did the work of three of the older boards: the CPU, the memory, and the floppy controller board.

The new single-board design brought the advantage of direct access to all basic I/O, so the BIOS could rely on these facilities and use them. The CP/M-Plus implementation I wrote for that system was more comfortable than the former Z80 multi-board version. For example, it introduced automatic disk exchange recognition. The high processor performance was amazing compared to the 4 MHz Z80.

Then came the IBM-PCs. At that time CP/M was systematically forced to death here in Europe. The special German computer syndrome became, "it has to be always the newest, biggest, and fastest machine," and the computer magazines followed that slogan. So, after a while I was the only one around here who was still active with CP/M. That was the reason why the CPU180 was built exactly once: I was working with the prototype board, and no one else seemed to be interested. (By the way, Wayne and Paul, if you had known that back in 1985/86, you wouldn't have had to design the YASBEC!)

At the end of 1988 Uwe Herczeg, another 'Tone CP/M user' in Germany, placed an ad in a major computer magazine asking for other CP/M users to come forward. This brought the remaining activists together. But most of them were still using 4 or 6 MHz Z80 systems, often also based on ECB-bus EuroCards, and some were very interested in my CPU180 system. So we began thinking about a redesign of that board with state-of-the-art technology. One of those new friends suggested using the Z280 instead of the Z180. Z280? I ordered the data sheet and what did I find but the late incarnation of the old Z800 idea. Unbelievable but true, the Z280 was really available! So we decided to use the Z280, for this CPU has a far more powerful instruction set than the Z180.

### The Hardware Design

#### Basics and Memory

As the hardware and software concepts of the CPU180 had proven themselves very well, all basic principles were carried over to the CPU280 design. As it was with the CPU180, the highest priority design rule was to get the absolute maximum performance out of the MPU with the absolute minimum circuit expense and parts cost. For the printed circuit board that meant using a standard double-sided PCB (no multi-layer) with normal wire thickness (0.25 mm) and no SMD parts at all. Just the good old sort of computer boards: reliable and easy to maintain.

In order to get the full power of the Z280, it must be run in the 16-bit Z-bus mode, with cache and burst-mode enabled, with no RAM wait states, and with the highest possible clock frequency (12.5 MHz). The on-board memory interface is 16 bits wide.

Main memory is built with dynamic RAM. There are eight sockets for 1 M-bit or 4 M-bit chips (with 4 bit data width). Chips must be used in groups of four to fit the 16-bit data bus. So the possible memory capacities are 512 KB

---

*Tilmann Reh is an electronics engineer at the Center for Sensor Systems of the University of Siegen, Germany. In addition, he owns a small company which develops custom specific problem solutions, often using microcontrollers or microprocessors. He has been active with CP/M since 1983, changed to CP/M-Plus in 1985, and has developed a number of ECB-bus boards. Tilmann can be reached by regular mail at 'In der Grossenbach 46, W-5900 Siegen, Germany' or by e-mail (Internet/bitnet) at 'tilmann.reh@hrz.uni-siegen.dbp.de'.*

and 1, 2, or 4 MB. This directly accessible memory should be enough for just about any circumstances. The standard configuration is with eight 1 M-bit chips (1 MB total). To save space and because of the availability of exactly pin-compatible 1 and 4 M-bit memory chips, ZIP-cased RAM is used.

Having had very good experience with synchronous DRAM timing (the CPU180 memory worked absolutely error-free for many years), I built a fully synchronous timing chain for the DRAMs again. This uses an HCT175 as a four-bit shift register and a GAL (gate array logic) containing the CAS decoding and selection logic. The DRAM interface also supports the processor's burst mode, using a GAL to sweep the lowest address bits during the burst (we can't use nibble-mode RAM's, as these are not available with 4-bit width!). The RAM timing is designed to meet all specifications of the Z280 at 12.5 MHz and RAMs with 100 ns access time. However, as the prices for 100 ns and 80 ns types are identical, we normally take the latter for safety.

Since the memory interface is far too fast for the ECB bus, external memory is not supported. Additional reasons for this decision were the data bus width (16 bit vs. 8) and the different timing and status signals (Z-bus vs. Z80). Support for external memory with burst-mode is absolutely impossible, as the ECB bus doesn't have strobing signals which could do that job. Interfacing the fast 16-bit internal databus to the slow 8-bit ECB bus would require a large amount of circuitry, and that would violate our main design rule. So, if someone really needs more than 4 MB RAM, please connect an external I/O-accessed RAM-disk via ECB (DMA support is no problem).

The boot software is placed in two 27C256 or 27C512 EPROMs, so the boot capacity is 64 KB or 128 KB. For easy handling, ordinary 28-pin DIP sockets carry the EPROMs. As this memory usually is needed only during boot-up, burst-mode is not supported, and to accommodate slower memory chips up to three wait states may be added.

The Z280 is able to use different memory timings when accessing the lower or upper half of its 16 MB address space. The EPROM is decoded into the lower half (8 MB, what a waste!) and the RAM into the upper half. This way, it is possible to use the RAM with zero wait-states and burst mode while the EPROM uses up to three wait-states and doesn't support burst mode. Mapping any desired memory configuration to the CPU's logical 64k address space is done with the PMMU (Paged Memory Managing Unit) internal to the Z280.

### **I/O Basics and Bus Interface**

In order to take advantage of widely available Z80 peripherals, the ECB bus clock and corresponding timing signals really should not exceed 6 MHz. As the CPU clock frequency is 12.288 MHz (for easy baud rate generation, it should be an integer multiple of 2.4576 MHz), it is convenient to use half the CPU clock frequency. This results in a bus clock of 6.144 MHz, which requires Z80B or Z80-6 components. With the internal wait-state generator of the Z280 programmed to insert four wait-states every I/O transaction (the maximum value), the duration of the I/O cycles exactly matches the divided clock. The clock divider is implemented by a single flip-flop (HCT74) and is reset at the beginning of each transfer to produce the correct phase relationship between bus clock and timing signals. Since the Z-bus control and timing signals cannot be used for the ECB bus, they are converted

into the appropriate Z80-type signals with a GAL.

Since the external peripherals use Z80-type vectored interrupts, the bus interface must be able to generate the correct interrupt acknowledge and RETI timings. Interrupt-acknowledge is treated as an I/O transaction and stretched by the Z280, but the RETI instruction consists of two memory cycles, which are too fast for the peripherals (besides the fact that the memory control signals do not appear on the ECB bus). Last but not least, the Z280 has a new interrupt mode (mode 3) which uses another return instruction (RETIL). So a slow RETI timing can be simulated using special accesses to on-board I/O locations, with a GAL generating the correct signals for the ECB bus. This way, vectored external interrupts are supported, although the clock speeds are very different.

The Z280 supports 24-bit I/O addresses as opposed to the 8/16-bit addresses of the Z80. The upper eight I/O address bits are accessed via the 'I/O Page Register' within the MPU. For full access to the 256 I/O addresses which are specified for the ECB bus, the bus interface is decoded to have an I/O page of its own. Another page is used for the on-board I/O, and two pages are reserved for the internal I/O registers of the Z280. Decoding of the I/O pages and addresses is done within GALs.

### **Internal and On-Board I/O**

The Z280 comes with a serial interface (UART), three 16-bit counter/timer circuits, and four DMA channels on-chip. To avoid an external baud rate generator for the UART, one of the timers is used. The 12.288 MHz clock allows standard baud rates up to 38400 to be generated. Higher, but non-standard, baud rates are also possible.

The internal UART is completed with two handshake signals which are not supplied by the MPU. A second serial interface is provided by a Twenty-Pin-UART (TPUART) COM81C17 by SMC (it includes its own on-chip baud rate generator). Both interfaces are buffered and shifted to RS-232C levels with a 5V-only line driver and receiver, the LT1134.

The CPU280 contains a real-time-clock (RTC) with 50 bytes of nonvolatile RAM, the Dallas DS 1287. Since this part already contains the lithium battery, no external circuitry is required. The RTC is able to generate interrupts at a specified date and/or time (alarm) or periodically. The NVRAM is ideal for storing configuration parameters.

Floppy disk I/O is handled by a WD/SMC 37C65 floppy disk controller (FDC). This neat chip, cased in a 44-pin PLCC, contains the complete controller. Just connect the CPU bus to some pins and the disk drives to some other (and don't forget the quartz crystal), and everything is running. Floppy-related data transfer may be handled by one of the Z280's DMA channels.

Simple TTL chips (HCT367, HCT259) are used to implement one 6-bit input port and one 8-bit output port. One bit of each port is used for the handshake signals for the first RS-232C interface. The other outputs drive some FDC control lines and three LED's for status display. Some of the inputs are connected to jumpers so that they can be used for configuration purposes.

There are four 16V8 type GALs on the CPU280 board. They contain memory address and CPU state decoding, I/O decoding, RAM timing and CAS decoding, and some 'glue' logic. Since nearly all signals are processed with only one

logic stage, the standard 'slow' 25 ns GALs are fast enough.

The entire CPU280 circuit is designed using CMOS technology. The internal logic is made with the 74HCT series, which is fast enough for nearly every signal. The bus interface and one timing-critical function in the DRAM interface use 74ACT chips. All LSI also are CMOS, and the GALs should be taken from the 'Q' series (quarter-power). As a result, the complete CPU280 board—fully operating at maximum clock speed—draws only about 350 mA from a single 5V power supply. No other voltages are required. The 32 chips nearly exactly fill the board space of the EuroCard, with just enough space left to avoid a multilayer PCB. (Hey, Wayne and Paul, why did you need SMD for just 23 chips?)

As with every good single-boarder, you just need to connect a power supply, at least one disk drive of any kind and size, and a terminal to complete the CP/M-Plus workstation. However, by connecting the CPU280 to a standard ECB backplane, you are free to use nearly any available ECB I/O board.

### The Software

The best hardware doesn't produce anything without the right software. With this in mind, I adapted my CPU180 BIOS to the CPU280, now using the powerful Z280 instruction set, of course. As with the circuit design, the basic principles and structures of the BIOS were taken directly from the CPU180. However, I completely redefined and enhanced the AutoFormat system and added a menu-driven hardware configuration program to the boot loader (this was impossible with the CPU180, as it had no NVRAM). Of course, some further improvements were made based on the experience of four years of CPU 180 operation.

Normally, the complete system (boot loader, CCP, BDOS, and BIOS) is booted directly out of the EPROM. As a result, you can boot up your machine in two seconds without any noise or mechanical action. (For testing new system versions, of course, booting from disk is also possible by pressing a button during RAM-test.)

With all functional enhancements fully compliant with CP/M-Plus definitions, all CP/M-Plus and most CP/M-2.2 software can be run on the CPU280 without any problems. By the way, the operating system runs in the Z280's system mode, while all user programs run in user mode. This is done mainly to achieve easy bank switching (which the MMU does automatically in this case), but it also increases system security. Unfortunately, CP/M-Plus must have the BIOS entry vector and some data structures in common memory, so you cannot absolutely prevent user programs from damaging the system software. But since we don't have any better (and still compatible!) operating system, we have to live with that fact.

### The Power

What is the real performance of a Z280 running with every booster switched on? The answer, unfortunately, is not as clear as you would probably like it to be. First, the Z280 is a pipelined CPU. So you really can't say how many clock cycles any instruction will take; it depends on the last few previous instructions. In addition, some instructions (jumps and calls, for example) flush the pipeline and thus are relatively 'slow'. Second, effective CPU speed depends on the 'hit ratio' of the cache controller. Small loops will run much faster than 'spaghetti code'. Third, the 16-bit arithmetic unit of the Z280 (opposed to the 4-bit one of the Z80) processes

indexing and math operations with greater speed gain than with other instruction types. So the more a program makes use of these instructions, the greater the effective speed.

Although it is impossible to specify exactly the power of the Z280, you can say that with normal 8080 or Z80 software it will have the power of a Z80 running at 16 to 20 MHz. Of course, using the new instructions (there are more than 600!) further increases the performance (while losing Z80 compatibility, of course). As the Z280 is used more and more, I hope we will soon see the first real Z280 programs or the first real Z280 operating system (which could get rid of the annoying 62 KB TPA limit of CP/M-Plus...).

### The Development History

After the first version of the CPU280 ran stably in March, 1990, I made a redesign of the PCB layout with slight changes in parts of the circuit. In June, 1990, I ordered the first run of PCBs, and in November, 1990, I sent them along with all semiconductors and special parts to about 25 people here in Germany. A few other people around the world got a PCB without the part set. I had to wait until November because that was the time when the 12.5 MHz version of the Z280 became available, and we didn't want to take the slower version first and upgrade a few months later. As of early 1991, many of the machines were running very well, as far as I have been informed by the users. The CPU280 has proved itself to be very fast and very reliable. Our 'PD and ZCPR Man', Helmut Jungkunz, likes the machine for its flexible method of processing different disk formats; that's why he uses it for nearly all disk distribution he has to do. Of course, he likes the raw power, too.

But the CPU280 is not our only project here in Germany. In a future column I would like to introduce my next board, an IDE controller which connects standard (PC) AT-Bus harddisks to any ECB-bus based CP/M system. The board also contains an active termination of all bus signals, a Centronics printer interface, a four-LED power monitor, and two system control buttons for reset and NMI (which, with the CPU280, forces a warm system reboot). This board nicely matches the CPU280 and allows one to build a really high power CP/M workstation.

A friend of mine is developing an LCD terminal. Together with a low-power Z180 single-boarder (based on my CPU180) it makes a powerful CP/M-Plus laptop! My newest project, which is coming along very well, is a high-performance CRT terminal for text and graphics at an unbeatable price. Wait and see! Needless to say, all three of these boards are EuroCard size. My friends and I look forward to describing these projects in future issues of *TCJ*.

### How to Get a CPU280

Some of you may be wondering how you can get a CPU280 of your own. Well, overseas shipment costs would be too high for sending complete kits as we did here in Germany. I think the easiest way to make the CPU280 available in the USA (and elsewhere outside Germany) is for a local dealer to provide my PCB together with locally obtained components. For the US market, who could do that job better than Sage Microsystems East? So, if you are interested in the board, the semiconductors, or both, please contact Jay Sage. Kits will include a disk with the complete system software, a hardware manual describing the circuit details, and a brief software description (I still haven't found the time to complete a real software manual).®

# Local Area Networks

## Broadband Cabling

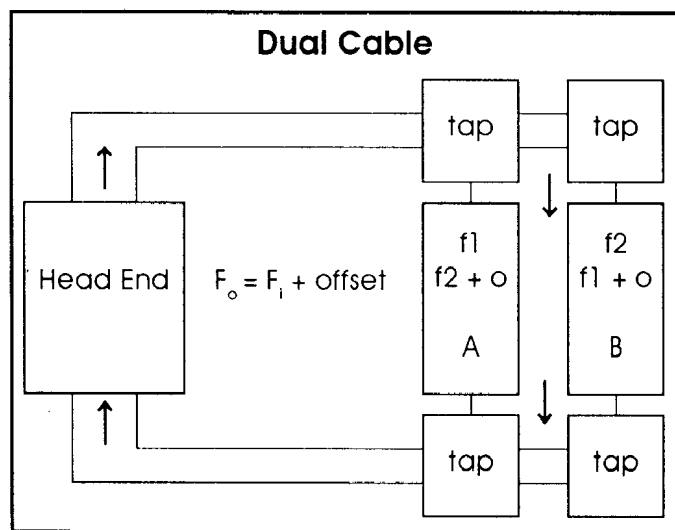
By Wayne Sung

Ethernet, along with other 'local' area networks (or as the British prefer, 'small' area networks), seldom stay small. When people start realizing the benefits available from being connected to a network, they will quickly want to join.

An Ethernet can cover quite a lot of area using only base-band coax, but there are several drawbacks in trying to extend coax beyond one building. The biggest problem is with ground differences among buildings. Another problem is that the coax does not withstand well the stresses required for long pulls in conduit.

### An Alternative to Coax

There are a number of ways to solve these problems. One of these involves the use of broadband cable, that is to say



cable television technology. This type of system often already exists on campuses and works quite well for extending Ethernets.

One would normally think of a broadband cable as something that only delivers services one way, which is the case in cable television. Data services generally need two directions. How would we modify an existing cable system to carry data services?

The most straightforward way would be to add another cable. This way each cable would send signals in one direction only. There would be a place where some electronic equipment would receive the signals from one cable, process them, and send them down the other cable. This place is called a 'head end', and in the simplest case would be an amplifier.

Let's look at two modems made to constitute a point-to-point circuit. Although the two are shown adjacent in the figure, they could literally be miles apart. The only requirements are that modem A generate a signal on frequency  $f_1$  and modem B generate one at  $f_2$ . Modem A would receive at  $f_2$ , and modem B at  $f_1$ . The offset is generally zero in this case. These frequencies can be chosen at random, but in practice will correspond to television channel frequencies. This allows the best fit if this cable is to carry multiple services.

A double cable system is simple enough, but the expense of installing the second cable is quite high. In fact, in most cases it will be less expensive to modify the first cable to make it bi-directional. This is where broadband seems like magic sometimes, and this is the process I want to explain more fully here.

### Splitting the Bandwidth

Consider that a broadband cable can handle a rather large frequency range, typically in the hundreds of megahertz. We can divide this frequency range in half, so that one band is used for each direction. If we take a piece of coax (up to several thousand feet long) and put a number of signals into the coax they will stay distinct.

A piece of wire is basically passive, and so will not cause the signals to interfere with each other. Although every signal exists everywhere along the wire, this is of no concern. We need one more item to make this scheme work. This is called a translator.

A translator takes an input signal and changes its frequency by some amount. This is exactly the case where a television station broadcasts on a certain channel but on cable it is in a different channel. There are two common translation amounts for data work, 156.25 MHz (called mid-split) and 192.25 MHz (called high-split).

Using a mid-split case, and returning to the modem example, modem A will transmit at  $f_1$  into the coax. The translator will receive this, and change it to  $f_1 + 156.25$  MHz. Similarly modem B's signal will be changed to  $f_2 + 156.25$  MHz. Thus A will now have to receive  $f_2 + 156.25$  MHz and B will have to receive  $f_1 + 156.25$  MHz.

A complication arises when the cable must be extended over much larger distances. There is loss in the cable, so in a very long cable signals will start to weaken. It is simple

---

*Wayne Sung has been working with microprocessor hardware and software for over ten years. His job involves pushing the limits of networking hardware in attempting to gain as much performance as possible. In the last three years he has developed the Gag-a-matic series of testers, which are meant to see if manufacturers meet their specs.*

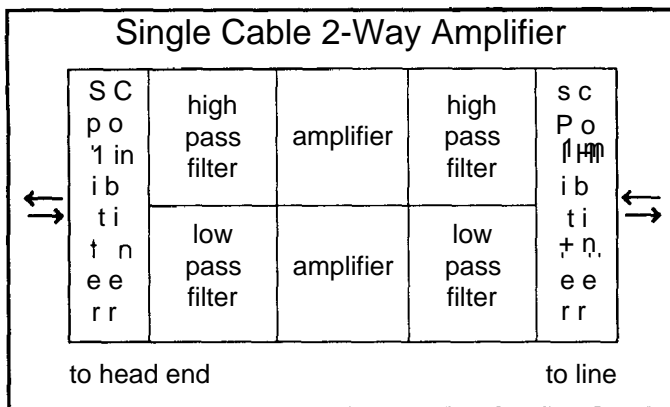
enough to install amplifiers to bring the signals back up, but amplifiers normally work in one direction only.

Fortunately, since we have already divided the frequency range, we can use low- and high-pass filters to split the signals and have two amplifiers, one for each direction. One reason it is cheaper to modify a cable system for bi-directional use is that most likely there is already space for the filters and additional amplifiers. The amplifier enclosures are designed for bi-directional use, and adding the second direction is fairly simple.

Typically the low frequency band flows toward the head end and the high frequency band away from it. This is not an absolute requirement, but in those cases where a cable already exists to provide one-way services these will be in the high range. For example, if there are television channels, they will start at channel 2 (54 MHz). Thus it becomes natural to design the two-way system so as not to have to disturb existing services.

In fact, there is a low-split system where all television channels are kept intact. However, this leaves very little space for one direction. Also, at lower frequencies it is more difficult to get good performance from wide-bandwidth devices (which we will need when we start extending LANs).

To add devices, we install taps along the main trunk. These taps isolate devices from each other. Remember that all



devices are to send toward the head end and receive from it. Individual devices do not need to hear each other, and in fact do not benefit from being able to do so.

RF circuits are easily affected by other circuits operating at nearby frequencies, so it pays to keep them isolated. In practice, each modem output will be isolated from the trunk by 10 dB or more, providing 20 dB of isolation from one modem to another. Taps are available with multiple branches per unit.

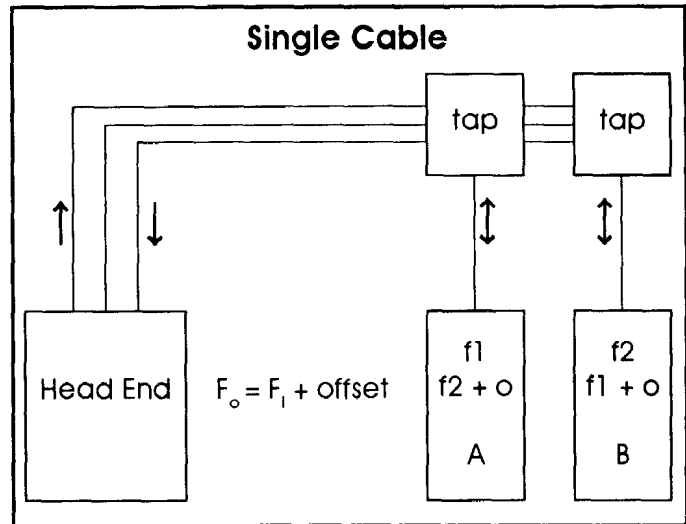
Note that splitters and taps are not the same. A splitter is meant to divide the total incoming signal several ways equally. A tap provides minimal loss in the through direction and a higher loss in the branch. Thus a two way splitter has at least 3 dB loss from the input to each output whereas the through direction of a tap should have less than 1.5 dB loss.

If a splitter gets used instead of a tap, the trunk level will suffer needlessly. This extra loss then sometimes gets (incorrectly) compensated for by reducing the values of otherwise correctly installed taps. Or, the outputs of modems have to be run at maximum to get enough working signal. This means that normal gain changes over temperature might cause intermittent operation.

Since the two frequency bands are handled separately, we

can set the receive levels on the trunk fairly high so that we can have some isolation on the receive side as well. It is also common to have 10 - 20 dB of receive isolation, but again this factor is often compromised due to low levels in the trunk.

With digital signals, there is a lot of leeway and many



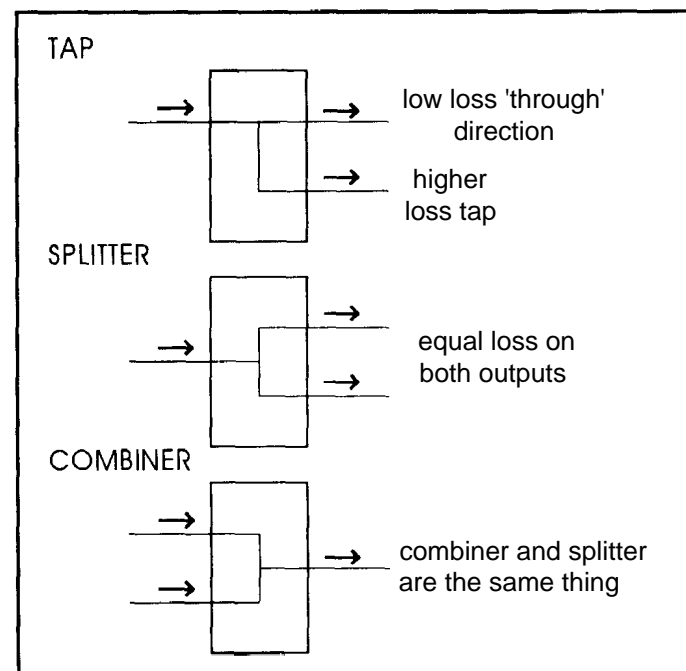
times the system works despite poor practices. If there is video being carried, though, the effects will be a lot more obvious.

With proper engineering, a broadband system can do a very good job of data distribution. For the most part, it does take some vigilance to keep one running well. Even temperature differences from winter to summer can cause problems. Still, one good technician can keep a fairly large broadband system running smoothly (until vacation time).

#### Why Not Fiber Optics?

Given that the cost of fiber optic systems is declining, why bother with broadband at all? If we are talking about a totally new installation, this is a compelling argument. However, although fiber solves the ground difference problem nicely (since it does not conduct), it is much more difficult to carry multiple services on fiber.

Individual services, particularly very high bit rate services,



are ideal for fiber. However, frequency multiplexing is much more difficult (what you would be multiplexing is the color of light). Telephone companies are already using fiber optic systems carrying gigabits per second, but these are point-to-point.

In a broadband system, any service can be tapped at any point using a device which costs only a few dollars. The same mechanism does not exist (yet?) for fiber. An ideal situation, perhaps, would be to use fiber between buildings and broadband inside the buildings. It is possible to take a whole broadband cable's content and modulate it onto a fiber.

### Getting Practical

So much for the general theory of broadband systems. How do we use them to extend local area networks? There are several different methods available.

The most direct way would be to design a system where the transceiver becomes an rf device, being able to take the baseband Ethernet signal and convert to and from an rf signal. A computer connected to this kind of transceiver would not know it was not baseband in most cases.

One consideration needed is whether the signal transmitted by the computer goes to the head end and back before being received by this computer as a confirmation. If the signal does make the round trip, this is a delay that the computer is not expecting, because a normal transceiver has no time delay between transmit and receive.

If the round-trip delay, which is of course related to the size of the plant, exceeds a certain amount, the computer might think that it is not able to generate a signal on the cable, and not complete the transmission.

On the other hand, if the rf transceiver is made to echo the transmitted signal to the computer, then there is no confirmation that the signal actually went onto the cable. With a baseband system, inability to actually generate a carrier should cause an error.

In either case, the total size of the plant will not greatly exceed the size of a baseband plant, because the collision timing windows don't change. It is possible to build a slightly larger plant because collision detect is done at the head end, which allows some of the timing to be reallocated.

By the way, any method of transmitting Ethernet over two paths has these considerations. This is true, for example, of

fiber optic and microwave systems.

One other disadvantage of the rf transceiver is the wide bandwidth required in the rf spectrum. Since the signaling rate is not changed, the rf side must still handle a 10 Mb/s signal. This usually ends up requiring two to three TV channels (12 - 18 MHz) in each direction and that is a lot to ask for.

One would normally want to be able to use the full size of a broadband plant. To be able to do this, the Ethernet timing window must be bypassed. This is often done by having the rf transceiver receive the entire packet before sending it onto the cable. Once the packet is in the transceiver, the timing window is satisfied, and a different method can be used in the cable.

This method is called a buffered repeater, although strictly speaking a repeater operates in a bit-by-bit fashion rather than by packets. Having the extra circuitry increases the cost, but fortunately it is not necessary to have one of these for each machine. A number of machines can be attached normally and then use one buffered repeater to leave the building.

It is then possible to change the signaling rate on the rf side to lower spectrum requirements. Note that simply decreasing the signaling rate also extends the distance, although this is not usually a good enough tradeoff.

To get a factor of 10 distance increase requires the same factor decrease in speed, and a 1 Mb/s system would feel significantly different from a 10 Mb/s system. Thus the rf distance gain is usually obtained with a proprietary signaling method, or at least one that is not based on CSMA/CD.

One possibility would be to use a token bus on the rf side. The transceiver would convert between the two formats. A token bus normally is able to span large distances, because access is not mediated using collisions. It happens that some token bus systems actually have collision detect requirements, but only for the purpose of adding new stations.

Most of the time, the conversion from baseband to another transmission medium is accompanied by some method to separate traffic between machines on the same side of the converter (local traffic) from traffic that occurs between machines in different locations (remote traffic). Two of these methods are bridging and routing, to be more fully described next time.®

---

*Real/52, from page 19*

The major improvements are supposed to be support for larger disk partitions and Posix-compliant utilities.

As I mentioned once before, UC Berkeley is working to remove AT&T code from their operating system. Later this year, a BSD release—but not a complete operating system—should be available from them for about \$500.

### Desqview/X

In August, Desqview/X should be released. This package will be more than an X terminal on a PC; it will allow programs to be written to run on the PC using Xt and Motif. I predict that this will be an earth-shaking product—the essential bridge between the workstation and the PC worlds. X window has already overturned the mini and workstation worlds. To have complete transparency of execution across a network of heterogeneous machines, and software portability, is an extremely compelling concept. And you don't lose any of your old programs either.

OS/2 2.0 should be available from IBM sometime in the fall. This should be an amazing product as well. Supposedly it will run Windows code, PM code, X window code, and DOS code, all at the same time.

### Next time

Wow, look at all the X's. And I didn't even mention X.400 or X.25. Next time I'm going to get away from all this grandiose complexity and discuss Kotekan, a "small is beautiful" operating system for the NS32 by Don Rowe. Meanwhile, don't trust anything with capital letters in the middle!®

Where to write or call:

Home Control Concepts  
9353-C Activity Road  
San Diego, CA 92126  
1-619-693-8887 or 1-800-CONTROL

---

# An Arbitrary Waveform Generator

## Using the Harris RTX2001A

By Jan Hofland

---

*{This article is second of a series on the building and programming of a waveform generator for mechanical structural testing. The hardware, with schematics, was given in our last issue, where we also began discussion of the software. In this installment, we continue with the software and Forth extensions. The series concludes in our next issue with the source code.—Ed.}*

### Commands

The following 16 specific commands have been implemented:

<u>command</u>	<u>description</u>
0 NOP	no operation
1 fastSine	set up to operate in fast sine mode. The next value input is the frequency.
2 accuSine	set up to operate in the accurate sine mode. The next value input is the frequency.
3 setAmpl	set output amplitude, in millivolts, for accurate sine output mode. The value following this command is used as the output amplitude. This value will be truncated to a multiple of 50 mV.
4 bufLoopO	set up to output from buffer 0 in circular buffer mode
5 bufLoopI	set up to output from buffer 1 in circular buffer mode
6 pingPongO	set up to output in ping pong mode starting with buffer 0
7 pingPongI	set up to output in ping pong mode starting with buffer 1
8 bufOutO	set up to output from buffer 0 until it is empty
9 bufOutI	set up to output from buffer 1 until it is empty
10 IdBuf	used to select a buffer and load it with data values. The next parameter must be the number of values to be loaded, up to 2048, and the msb set if it is for buffer 1. Then the data values are input in sequence.
11 setoffset	set the offset to be subtracted from the sinewave value in accurate sinewave mode, in 100's of microvolts. The next value following this command is the offset.
12 setFilt	used to set the filter cutoff point. The next value input is used for the filter cutoff, range 1 to 40000 Hz.
13 getPeriod	used to input the desired sample rate. The next value input is used for the timer 0 period. If in either sinewave operating mode, a check is performed to readjust the phase increment to be consistent with this period and the output frequency.
14 stop	disable timer 0 interrupt to stop outputting points
15 start	enable the timer 0 interrupts to start outputting data and then go into a loop to repeatedly check the stale data flag in

register RX and update the value in scratchpad  
RH if the stale data flag is set. Continue this  
loop until the timer 0 interrupt is masked.

The execution vectors for each of the 16 commands are contained in a table called *parse* that is used for command execution.

### Main Operating Loop

Here is the overall operating loop:

```
: main ( — )
  initialize
  BEGIN begin an infinite loop
  readFifo read a command from the input stream
  doCmd. perform the command
  AGAIN ; go do it again
```

and here is the command execution word:

```
: doCmd (n —) perform the command represented by n
  DUP cmdMsk AND make sure the upper 12 bits are zeroes
  IF if the result of the logical AND is non-
  DROP zero then it is an unrecognized command
  . Unrecognized Command " and discarded an error
  message to the user
  ELSE parse EXECUTE otherwise get an execution vector
  THEN ; from command parse table and execute it.
```

### Demonstration Software

Part of the software is for demonstrating some of the capabilities of the Waveform Generator. As designed, the Waveform Generator expects its commands and arbitrary data to be input from the input FIFO. Rather than design and debug some hardware for controlling it in this manner I chose to simulate the input FIFO in software. This has some impact on the performance realized in the breadboard, but it does demonstrate that the operating modes do, in fact, work as designed. The basic scheme is to construct a queue of commands and data exactly as if they were being input from the input FIFO. I then use a vectored input routine to accept input from the software queue rather than the input FIFO.

### Input Stream Queue

A 512 word input queue is constructed using the word *buildCQ*. This queue is called *cmdQ*. It is used for setting up the various demonstrations. The way it gets used is to load it with

---

*Jan Hofland is employed with Hewlett Packard as a hardware design engineer, working primarily on 68K based systems. His personal interests include woodworking, and playing with electronics for over 20 years. His current favorite system is the F68HC11 from New Micros. Jan can be contacted at 2419 123rd Ave. SE, Everett WA 98205 or by telephone (206) 334-0738 during the evenings.*

the sequence of command codes each followed by the required parameter, if any. The last command entered is the start command. Then the queue end pointer and tail pointer are adjusted to conform with the actual size of the command queue.

### Demonstration Sequences

There are 6 sequences implemented:

- demoCircSq implements a 1 Khz square wave using circular buffer mode and a 40 Khz filter cutoff. The square wave amplitude is a nominal -1V to +1V.
- demoFsine20 implements a 20 Khz sinewave using fast sine mode and a filter cutoff of 20 Khz.
- demoFsine1 implements a 1 Khz sinewave using fast sine mode and a filter cutoff of 1 Khz.
- demoAsine1 implements a 1 Khz sinewave using the accurate sine mode with a 1 volt RMS amplitude, filter cutoff of 1 Khz and an offset of 90 millivolts.
- demoPp Uses the ping pong output mode to output a half sine of 1 Khz, pause for 500 microseconds, output another half sine at half amplitude, pause for another 500 microseconds, and then output a full cycle of 2 Khz sine. Then stop. There isn't anything particularly useful about this waveform choice. It just shows some of the versatility in output modes.
- demoOnce Uses the once out mode to output one cycle of a 100 Hz sine that is linearly attenuated to zero.

### The Demonstration Loop

The word *choose* invokes a menu structure for the user to choose which demo to be run and then it runs it. The main operating loop for the demonstration is patterned after the main high level word of the normal operating environment with one difference. Rather than being an infinite loop, it is repeated until the command queue is empty.

The word *demo* kicks it all off. Here is its definition:

```

: demo ( — )          start up a demo
  BEGIN
    choose dVect EXECUTE choose a demo and fetch its
    execution vector from the
    table dVect and execute it.
    This sets up the queue of
    commands and parameters.
    demoMain          perform the commands in the
    command queue
  AGAIN ;            and repeat it.

```

Note that four of the six *demos* do not have a terminating condition. Stop them with the red pushbutton and reinvoke *demo* to perform another one.

### Conclusions

This model has clearly demonstrated the capability to contract a waveform generator based on the RTX2001 A. Much of the hardware added can be incorporated into an ASIC, if so desired. It should be a simple matter to include all of the control logic described here into the ASIC. I would envision keeping the input FIFO off-chip, as well as the DAC and the reconstruction lowpass filter. Keeping the added data RAM off-chip as well as increasing its size also makes sense. I would make the DAC sample clock a dedicated hardware timer, as well as the filter clock toggle control. Of course, both of these timers would remain programmable.

If I were to take the next step in the design, I would probably incorporate the following capabilities:

- Provide a second DAC for range control. This would have to be a multiplying type DAC and would provide dynamic amplitude scaling of all outputs. I expect that a 12 bit DAC would be adequate for this purpose.
- Provide a means of measuring the analog output DC offset and correcting for that offset. It turns out that as good as the switched capacitor filter is for waveform reconstruction, it has rather poor DC offset characteristics. Provision of an 8 bit DAC, possibly on-chip, for correcting for offsets makes sense. Another, though less versatile, alternative would be an analog servo circuit to servo the output toward an average zero level.
- Provide a means to bypass the filter to allow random noise, wideband outputs. This kind of stimulus is useful for some kinds of testing.
- Provide a user input for shutting down the output and for gracefully ramping the output to zero when the shutdown occurs.

Finally I must comment on the use of Forth for the prototyping environment. In retrospect, I find it amazing that I was able to write and debug all the software necessary to control this hardware in the matter of about 7 weeks of spare time effort. This is the first programming I have done in anything other than an academic environment. The language lends itself to rapidly start doing something related to the application rather than spending all of your time learning syntax or trying to relate some obscure command behavior to what the hardware is supposed to do.®

Listing 1. GAL 26CV12 ABEL Source Code

```

MODULE mem_addr_decode ;
  flag '-r4'

  TITLE 'memory address RAM select and filter clock toggle
  control for RTX2001 waveform generator
  Jan Hofland
  900404
  900406
  900415
  .

  +---- U----+
  telk 1 | . 28| tcLo
  GAO 2 | : 27| cntrNabLo

  "          GAI  13  26| cntLatch
  "          GA2  14  25| cLoadLo
  "          grd_wrLo 15  24| fltClk
  "          gioLo  16  23| reset (input)
  "          Vcc   17  22| resetLo
  "          MAI 9  18  21| gnd
  "          MAI 8  19  20| no_out20
  "          MAI 7  10  19| RAMSEL
  "          MAI 6  11  18| LRAMSEL
  "          MAI 5  12  17| uds (input)
  "          MAI 4  13  16| Ids (input)
  "          MAI 3  14  15| MAI2 (input)
14          +----+
  " The purpose of this device is twofold. It decodes a memory
  " address to enable the external 8K x 16 RAM. This memory is

```

```

" addressable as either bytes or words. The other purpose of
" the device is to control the programmable counter used to
" provide a clock source for the switched capacitor filter IC.
" The filter requires a clock source 100 times its cutoff
" frequency. For its highest bandwidths (up to 40 KHz), using
" a timer interrupt would require interrupt servicing at a 4
" MHz rate. Instead, what is done is to use a programmable
" down counter to toggle filtClk. The down counter is used for
" divide down of up to 256. For higher divisions, an internal
" timer in the RTX2001 is used to generate an interrupt, and
" the filtClk is toggled by writing to an ASIC bus address.

" modified to change from a 22V10 to a 26V12 device to allow
" full memory decoding of RAM contiguous with the evaluation
" board MAP chip RAM.

" device declaration
" U05 DEVICE 'P22V10';
U3 DEVICE 'P26CV12';

" pin declarations
" inputs
tcik          pin 1;    "8 MHz clock input
GA0,GA1,GA2 pin 2,3,4; " ASIC bus addresses
grd_wrLo     pin 5;    "ASIC bus read/write
gioLo        pin 6;    "ASIC bus external address      active low
tcLo         pin 28;   "counter terminal count      active low

MA19,MA18,MA17,MA16 pin 8,9,10,11; "memory bank address
MA15,MA14,MA13,MA12 pin 12,13,14,15; "upper 4 memory bits
uds,Ids pin 17,16; "memory bus upper and lower data strobe
reset pin 2 3; "system reset

" outputs
cntrNabLo pin 27; "enable filter clock counter active low
cntLatch pin 26; "ASIC bus write data to counter data
               "in latch
cLoadLo pin 25; "counter load control active low
filtClk pin 24; "filter clock
URAMSEL,LRAMSEL pin 19,18; "upper and lower byte RAM select
resetLo      pin 22; "active low reset output
no_out20     pin 20; "unused output

" constant declarations
x,z = .X.,.Z.;
k = .K.; "high-low-high clock pulse

mAddr = (MA19,MA18,MA17,MA16,MA15,MA14,MA13,MA12);
        "upper memory address
gAddr = [GA2,GA1,GA0]; "ASIC bus address

" pin attributes
cntLatch,cLoadLo ISTYPE 'com,neg';
cntrNabLo,filtClk ISTYPE 'reg,pos';
URAMSEL,LRAMSEL ISTYPE 'com,pos';
resetLo ISTYPE 'com,neg';

EQUATIONS

" disable outputs of I/O pins used as inputs

reset.EN = 0;
MA12.EN = 0;
uds.EN = 0;
Ids.EN = 0;

filtClk.RE = reset; "filter clock asynchronous reset

IresetLo = reset;

" counter data latch control
l cntLatch = (gAddr == 6) & !grd_wrLo & IgioLo & Itcik
& !reset;

" counter load control asynchronous set-reset flip-flop
!cLoadLo = reset "force low if reset active
# !tcLo & Itcik "set (low) during low half of
"clock if counter terminal count asserted
# !cLoadLo & !tcik; "hold asserted during low half
"of clock reset during high half of clock
"if reset isn't asserted

" external RAM selects
URAMSEL = ((mAddr == 5) # (mAddr == 6) # (mAddr == 7)

```

```

# (mAddr == 8) & uds;
LRAMSEL = ((mAddr == 5) # (mAddr == 6) #
# (mAddr == 8) & ids;

" cntrNabLo is a registered signal used to enable the external
" filter clock down counter. It is set by writing to ASIC bus
" address 5 and cleared by writing to external address 4. The
" counter is enabled when cntrNabLo is cleared. The data
" written to these address is 'don't care'.

STATE_DIAGRAM cntrNabLo
state 0: if ((gAddr == 5) & !grd_wrLo & IgioLo)
then I
else 0;
state 1: if (((gAddr == 4) & igrd_wrLo & IgioLo) # reset)
then 0
else 1;

" filtClk is implemented as a toggle flip-flop. If cntrNabLo
" is low, then the hardware counter is used to toggle filtClk.
" Since cLoadLo gets asserted whenever the counter reaches
" terminal count, and the fact that cLoadLo changes the
" counter control from count down to load, thereby deasserting
" terminal count out, cLoadLo is used for toggle control. If
" cntrNabLo is high, then filtClk is toggled by writing to an
" ASIC bus address. The data is a 'don't care'.

EQUATIONS
filtClk.EN = Ireset;

STATE_DIAGRAM filtClk
state 0: if (!cntrNabLo & !cLoadLo
# cntrNabLo & (gAddr == 7) & tgrd_wrLo & IgioLo)
then I
else 0;
state 1: if (!cntrNabLo & !cLoadLo
# cntrNabLo & (gAddr == 7) & lgrd_wrLo & IgioLo)
then 0
else 1;

TEST_VECTORS 'memory select'
([mAddr,uds,Ids] -> [URAMSEL,LRAMSEL])
[ 0, 0, 0 ] -> [ 0, 0 ];
[ 1, 1, 1 ] -> [ 0, 0 ];
[ 2, 1, 1 ] -> [ 0, 0 ];
[ 3, 1, 1 ] -> [ 0, 0 ];
[ 4, 1, 1 ] -> [ 0, 0 ]; "MAP chip RAM active
[ 5, 0, 1 ] -> [ 0, 1 ];
[ 5, 1, 1 ] -> [!, 1 ];
[ 5, 1, 0 ] -> [!, 0 ];
[ 5, 0, 0 ] -> [ 0, 0 ];
[ 6, 0, 1 ] -> [ 0, 1 ];
[ 6, 1, 1 ] -> [!, 1 ];
[ 6, 1, 0 ] -> [!, 0 ];
[ 6, 0, 0 ] -> [ 0, 0 ];
[ 7, 0, 1 ] -> [ 0, 1 ];
[ 7, 1, 1 ] -> [!, 1 ];
[ 7, 1, 0 ] -> [!, 0 ];
[ 7, 0, 0 ] -> [ 0, 0 ];
[ 8, 0, 1 ] -> [ 0, 1 ];
[ 8, 1, 1 ] -> [!, 1 ];
[ 8, 1, 0 ] -> [!, 0 ];
[ 8, 0, 0 ] -> [ 0, 0 ];
[16, 1, 1 ] -> [ 0, 0 ];
[32, 1, 1 ] -> [ 0, 0 ];
[64, 1, 1 ] -> [ 0, 0 ];
[128, 1, 1 ] -> [ 0, 0 ];
[255, 1, 1 ] -> [ 0, 0 ];

TEST_VECTORS 'filter clock control'
([tcik,gAddr,grd_wrLo,gioLo,tcLo,reset] ->
[ cntLatch,cLoadLo,cntrNabLo,filtClk,resetLo ])
[ 1, 6, 0, 0, x, 1 ] -> [ 1, 0, 0, z, 0 ]; "reset
[ 1, 6, 0, 0, 1, 0 ] -> [ 1, 1, 0, 0, 1 ];
[ 0, 6, 0, 0, 1, 0 ] -> [ 0, 1, 0, 0, 1 ]; "counter
" latch enable
[ 1, 6, 0, 0, 1, 0 ] -> [ 1, 1, 0, 0, 1 ];

[ 1, 5, 0, 0, 1, 0 ] -> [!, 1, 0, 0, 1 ];
"32" ( k, 5, 0, 0, 1, 0 ) -> ( 1, 1, 1, 0, 1 ); "set

```

```

" ctrNabLo high
[ k, 4, 0, 0, 1, 0 ] -> ( 1, 1, 0, 0, 1 ]; "set
" ctrNabLo low again
[ 1, 0, 0, 0, 0, 0 ] -> [ 1, 1, 0, 0, 1 ]; "counter at
"terminal count
[ 0, 0, X, X, 0, 0 ] -> [ 1, 0, 0, 0, 1 ]; "counter
"load
[ 0, 0, X, X, 1, 0 ] -> [ 1, 0, 0, 0, 1 ];
"37" [ 1, 0, x, x, 1, 0 ] -> [ 1, 1, 0, 1, 1 ]; "toggle
"filtcik
[ 1, 0, 0, 0, 0, 0 ] -> [ 1, 1, 0, 1, 1 ]; "counter at
"terminal count
[ 0, 0, x, x, 0, 0 ] -> [ 1, 0, 0, 1, 1 ]; "counter load
"40" [ 0, 0, x, X, 1, 0 ] -> [ 1, 0, 0, 1, 1 ];
[ 1, 0, X, X, 1, 0 ] -> [ 1, 0, 0, 1 ]; "toggle
"ctrNabLo high
( k, 5, 0, 0, 1, 0 ] -> [ 1, 1, 1, 0, 1 ]; "set
"ctrNabLo high
[ 1, 0, x, x, 1, 0 ] -> [ 1, 1, 1, 0, 1 ];
"44" [ 1, 7, 0, 0, x, 0 ] -> ( 1, x, 1, 0, 1 ];
"program control toggles:
[ k, 7, 0, 0, x, 0 ] -> [ 1, x, 1, 1, 1 ];
"46" ( k, 7, 0, 0, x, 0 ] -> [ 1, x, 1, 0, 1 ];
[ k, 7, 0, 0, x, 0 ] -> [ 1, x, 1, 1, 1 ];
"48" [ k, 7, 0, 0, x, 0 ] -> [ 1, x, 1, 0, 1 ];
END mem_addr_decode

```

Listing 2. GAL 22CV10 Abel Source Code

```

MODULE asic_bus_control
" flag '-r4'

TITLE 'ASIC bus address decode and ADC strobe control
for RTX2001 waveform generator
Jan Hofland
900404
/

"
+----- U-----+
"      tclk |1      24| Vee
"      GAO |2      23| GDO tri-state
"      GA1 |3      22| GDI tri-state
"      GA2 |4      21| serDatLd
"      grd_wrLo |5 | 20| DACItch
"      gioLo |6      19| Qd local state variable
"      noUse7 |7      18| Qc local state variable
"      noUseS |8      17| Qb local state variable
"      strobLo |9 | 16| Qa local state variable
"      fifoMfLo |10 | 15| fifoRdLo
"      fifoFulLo |11 | 14| fifoWrLo
"      gnd |12 | 13| reset
"
+-----+

" The purpose of this programmable device is to load a 16
" bit shift register from the external ASIC bus and shift its
" output into the serial DAC, and strobe the DAC LE (Latch
" Enable) input after 18 data bits are shifted into the DAC.
" Data is shifted in most significant bit first at an 8 MHz
" rate. The last two bits are always zero. This device also
" controls writing to the FIFO and reading from it. If the
" FIFO is full, then writing is inhibited, and if it is
" empty, then reading is inhibited. The two FIFO status bits
" can be read on the lower two bits of the ASIC bus.

"device declaration
U8 DEVICE 'P22V10';

"pin declarations
" inputs
tclk pin 1; "8 MHz clock out of RTX2001
GAO,GA1,GA2 pin 2,3,4; "ASIC bus register address
grd_wrLo pin 5; "ASIC bus read/write
gioLo pin 6; "ASIC bus external access enable
noUse7,noUseS pin 7,8; "unused inputs pulled up
strobLo pin 9; "external FIFO data write strobe
" pulled low to request a write. Set
" High after fifoWrLo goes low.
" fifoWrLo then goes high again to
" write data into FIFO. User must
" leave data valid at least 250 ns
" after setting strobLo high.
fifoMfLo,fifoFulLo pin 10,11; "FIFO status signals
reset pin 13; "system reset inhibits FIFO read/write

"outputs
GDI,GD0 pin 22,23; "ASIC bus data lines for reading
"status
serDatLd pin 21; "shift register load control
DACItch pin 20; "DAC latch enable signal
Qd,Qc,Qb,Qa pin 19,18,17,16; "counter state variables

```

```

"used for generation of DACItch
fifoRdLo pin 15; "ASIC bus controlled FIFO read
"active low
fifoWrLo pin 14; "FIFO write signal active low

"constant declarations
x,z,c,k = X,..z,..e,..K;
gAddr = [GA2,GA1,GA0]; "ASIC bus register address
sent = [Qd,Qc,Qb,Qa]; "shift register control variables

"pin attributes
GDI,GD0 ISTYPE 'com,pos';
Qd,Qc,Qb,Qa ISTYPE 'reg,pos';
SerDatLd ISTYPE 'com,pos';
DACItch ISTYPE 'com,pos';
fifoRdLo ISTYPE 'com,neg';
fifoWrLo ISTYPE 'reg,neg';

STATE_DIAGRAM sCnt
" a 16 state gray code counter is implemented. The idle
" is state 0. If the DAC shift register is loaded, the
" counter will progress through 16 state to state 1000,
" the DACItch signal, and then reverse direction. When
" gets back to state 1010 it will then go to the idle state
" 0000. DACItch will get cleared 1/2 clock cycle later
" strobe the serial data into the DAC.
state b0000:
if ((reset & (gAddr == 0) & !grd_wrLo & !gioLo)
then b0001
else b0000;
state b0001:
if ((reset)
then b0011
else b0000;
state b0011:
if ((reset)
then ^b0010
else ^b0000;
state ^b0010:
if ((reset)
then ^b0110
else b0000;
state ^b0110:
if ((reset)
then b0100
else b0000;
state b0100:
if ((reset)
then b1000
else b0000;
state b1000:
if ((reset)
then ^b1100
else b0000;
state ^b1100:
if ((reset)
then ^b1101
else b0000;

```

```

state ^b101:
  if (!reset)
  then b101
  else b000;
state b101:
  if (!reset)
  then b100
  else b000;
state ^b110:
  if (!reset)
  then b100
  else b000;
state b100:
  if (!reset & IDACtch)
  then b101
  else b000;
state ^b1011:
  if (!reset & IDACtch)
  then ^b1001
  else if (!reset & DACtch)
  then ^b1010
  else b000;
state b1001:
  if (!reset & IDACtch)
  then b1000
  else if (!reset & DACtch)
  then ^b1011
  else b000;
state ^b1000: "turn-around state
  if (!reset)
  then ^b1001
  else b000;

EQUATIONS

fifoWrLo.RE = reset; "asynchronous reset of all flip-flops

serDatLd = (gAddr == 0) & !grd wrLo & !gioLo;

DACtch.EN = Ireset; "tri-state connection to DAC if
              "reset active
DACtch = (sCnt == 8) "set at turn-around state
          # DACtch & (tclk # (Qd & !Qc)); "not reset condition

!fifoRdLo = (gAddr == 1) & grd wrLo & !gioLo & itclk &
            fifoMtlLo & Ireset
            # !fifoRdLo & !tclk; "hold low if last read of
            "fifo asserts fifoMtlLo
IfifoWrLo := Ireset & IstrobLo & fifoFullLo
            # Ireset & IstrobLo & IfifoWrLo; "hold low if
            "writing last word in FIFO sets
            "fifoFullLo

GDI.EN = (gAddr == 3) & grd wrLo & !gioLo; "status read
        "tri-state enable

GDI = IfifoFullLo;

GDO.EN = (gAddr == 3) & grd wrLo & !gioLo; "status read
        "tri-state enable

GOO = IfifoMtlLo;

TESTVECTORS 'serial data control'
  ([tclk,gAddr,grd wrLo,gioLo,reset] ->
    [sCnt,serDatLd,DACtch])
    (0, x, x, x, 1) -> (0, x, z);

```

```

[1, X, X, 1, 0] -> [0, 0, 0];
[1,0,0,0,0]->[0,1,0];
[0,0,0,0,0]-> [0,1,0]; "load shift register
                                     " when tclk 0 to 1

[1, 0, 0, 0, 0] -> [1, 1, 0];
[1, x, x, 1, 0] -> [1, 0, 0];
[k, x, X, 1, 0] -> [3, 0, 0];
[k, x, X, 1, 0] -> [2, 0, 0];
[k, x, x, 1, 0] -> [6, 0, 0];
[k, x, X, 1, 0] -> [7, 0, 0];
[k, x, x, 1, 0] -> [5, 0, 0];
[k, x, x, 1, 0] -> [4, 0, 0];
[k, x, X, 1, 0] -> [12, 0, 0];
[k, x, x, 1, 0] -> [13, 0, 0];
[k, x, x, 1, 0] -> [15, 0, 0];
[k, x, x, 1, 0] -> [14, 0, 0];
[k, x, x, 1, 0] -> [10, 0, 0];
[k, x, x, 1, 0] -> [11, 0, 0];
[k, x, X, 1, 0] -> [9, 0, 0];
[k, x, x, 1, 0] -> [8, 0, 1];
[k, x, x, 1, 0] -> [9, 0, 1]; "turn around
[k, x, x, 1, 0] -> [11, 0, 1];
[k, x, x, 1, 0] -> [10, 0, 1];
[k, x, x, 1, 0] -> [0, 0, 1];
[0, x, x, 1, 0] -> [0, 0, 0]; "DACtch cleared

TEST_VECTORS:'FIFO control & status'
  ([tclk,gAddr,grd wrLo,gioLo,strobLo,fifoMtlLo,
    fifoFullLo,reset] -> [fifoRdLo,fifoWrLo,GDI,GDO])
  [0, X, X, 1, 1, x, 1, 1] -> [1, 1, z, z]; "reset
  [0, x, x, 1, 1, 0, 1, 0] -> [1, 1, z, z];
  [1, x, X, 1, 1, 0, 1, 0] -> [1, 1, z, z];
  [k, x, X, 1, 1, 0, 1, 0] -> [1, 1, z, z];
  [k, x, X, 1, 0, 0, 1, 0] -> [1, 0, z, z]; "FIFO write
  [k, X, X, 1, 1, 0, 1, 0] -> [1, 1, z, z];
  (1, X, X, 1, 1, 1, 1, 1, 0) -> (1, 1, z, z);
  [k, X, X, 1, 0, 1, 1, 0] -> [1, 0, z, z];
  [k, x, x, 1, 1, 1, 1, 0] -> [1, 1, 2, z];
  [k, x, x, 1, 0, 1, 1, 0] -> [1, 0, z, z];
  [1, x, x, 1, 0, 1, 0, 0] -> [1, 0, z, z]; "FIFO full
  [k, x, X, 1, 0, 1, 0, 0] -> (1, 0, z, z);
  [k, x, x, 1, 1, 1, 0, 0] -> [1, 1, z, z];

  [1, 1, 1, 0, 1, 1, 0, 0] -> [1, 1, z, z]; "FIFO read
  [0, 1, 1, 0, 1, 1, 0, 0] -> [0, 1, z, z];
  [1, 1, 1, 0, 1, 1, 1, 0] -> [1, 1, z, z];
  [1, 1, 1, 0, 1, 1, 1, 0] -> [1, 1, z, z];
  [0, 1, 1, 0, 1, 1, 1, 0] -> [0, 1, z, z];
  [0, 1, 1, 0, 1, 0, 1, 0] -> [0, 1, z, z]; "empty
                                     "again
  [1, 1, 1, 0, 1, 0, 1, 0] -> [1, 1, z, z];
  [1, 1, 1, 0, 1, 0, 1, 0] -> [1, 1, z, z];
  [0, 1, 1, 0, 1, 0, 1, 0] -> [1, 1, z, 2]; "no read
  [1, 1, 1, 0, 1, 0, 1, 0] -> (1, 1, z, z);

  [1, 1, 1, 1, 1, 0, 1, 0] -> [1, 1, z, z];
  [1, 3, 1, 0, 1, 0, 1, 0] -> [1, 1, 0, 1]; "status
                                     "read
  [0, 3, 1, 0, 1, 0, 1, 0] -> [1, 1, 0, 1]; "FIFO empty
  [1, 3, 1, 1, 1, 1, 1, 1, 0] -> [1, 1, z, z];
  [1, 3, 1, 0, 1, 1, 1, 1, 0] -> (1, 1, 0, 0);
  [0, 3, 1, 0, 1, 1, 1, 1, 0] -> [1, 1, 0, 0];
  [1, 3, 1, 1, 1, 1, 0, 0] -> [1, z, z];
  [1, 3, 1, 0, 1, 1, 0, 0] -> [1, 1, 1, 0]; "FIFO full
  [0, 3, 1, 0, 1, 1, 0, 0] -> [1, 1, 1, 0];
  [1, 3, 1, 1, 1, 1, 0, 0] -> [1, 1, z, 2];

```

END asic\_bus\_control

"The best executive is the one who has sense enough to pick good people to do what he needs done, and self-restraint enough to keep from meddling with them while they do it."

"Most of us are broad-minded enough to admit there are two sides to every question — our own and the side no intelligent, informed, sane and self-respecting person could possibly hold."

## Glossary for Words Added to EBForth 1

### Boot Cod\* Glossary

**#Que!** ( addr — n ) Return the number of values in the queue whose name is addr. reading a value from the data input source. Normally, this variable points to the hardware FIFO read routine. For demonstration purposes, it points to fetching values from the software command queue.

**>Cbuf** ( n addr — ) Insert n into the queue whose name is addr.

**>Que!** ( n addr — ffig ) Store n into the queue whose name is addr and return a FALSE flag if the queue isn't full.  
( n addr — n addr tfig ) If the queue is full, return the value, the address, and a TRUE flag, indicating that there isn't any room in the queue.

**>RH** ( n — ) Write n to the RH register and clear the stale data flag kept in the RX register. This flag gets set whenever data is transferred from RH to the output DAC.

**accuPt ( — n )** Return the sine of the current value of phase and delPhase : scaled to a 50 mV increment and then the current offset is subtracted from it. Used to get the output value in accurate sine mode.

**accuSine ( — )** Accurate sinewave mode command. Sets up the correct mode, the next point execution vector, and the sinewave frequency.

**bufLoopO ( — )** Circular buffer mode. Sets up to output data from data buffer 0 in circular buffer mode.

**bufLoopI ( — )** Circular buffer mode. Sets up to output data from data buffer 1 in circular buffer mode.

**bufMax ( — 2048 )** A constant. Corresponds to the size of the maximum data queue, in words.

**bufOutO ( — )** Output from buffer 0 until it is empty.

**bufOutI ( — )** Output from buffer 1 until it is empty.

**buildCQ ( n — )** Create a queue structure in code space of n words. A defining word. The name of the structure should follow buildCQ. Sets up and initializes the #values variable, the head pointer for extracting values from the queue, the tail pointer for inserting values, and the end pointer for determining when to loop back to the beginning, and it allocates the necessary space to contain n vlues.

**buildDQ ( n — )** Create a queue structure in data space of n words. A defining word. The name of the queue should follow buildDQ; e.g., 50 buildDQ fudd will build a queue named fudd that will hold 50 values. Sets up the lvalues variable, the head, tail, and end pointers, and allocates space for n values.

**calcFreq ( - )** Calculate the actual sinewave frequency from the known values of timebase period and phase increment. Because of the quantized nature of these parameters, the actual output frequency may differ somewhat from the set frequency.

**calcint ( nl n2 - - )** Calculate the required timebase period from the phase increment and sinewave frequency. Also checks the period to make sure that it is at least the mode dependent minimum.

**calcPhaseinc ( freq - )** Calculate the required phase

increment from the set frequency for sinewave output.

**Cbuf>** ( addr — n ) Fetch the next value from the queue whose name is addr. No checking is done to see if the queue is empty. Most useful for outputting from a circular buffer where the data output sequence is repetitive.

**circBuf ( — )** Circular buffer output mode. Fetch the next value from the current data buffer and load it into the RH register to be output to the DAC.

**clearQ ( addr — )** Initialize the #values, the head pointer and the tail pointer for the queue specified by addr.

**cntMsk ( - OFFF )** A constant. A mask used for setting the number of data values input with the IdBuf command.

**cntrDisabl ( — )** Disable the filter clock (hardware) counter. This word writes to ASIC address 1C (hex) as a write only pseudo register. For low values of filter cutoff, the filter clock is controlled with timer 1 interrupts.

**cntrEnabl ( — )** Enable the filter clock (hardware) counter.

**cosAdj ( — n )** Return the cosine adjust value for the current values of phase and delPhase. Used in the accurate sine output mode.

**cutoff ( — addr )** A variable for the lowpass filter setpoint.

**DAC!** ( n — ) Write the 16 bit value to the DAC shift register on the ASIC bus. Writes to ASIC address 18 (hex).

**delPhase ( — addr )** A variable for incremental phase interpolation. Used in accurate sine mode to keep track of the incremental sinewave phase between adjacent values of phase.

**delPhaseInc ( — addr )** A variable for adding to the incremental phase accumulator. This value is added to delPhase for each new output calculation in accurate sine mode.

**div512 ( nl — n2 )** Divide nl by 512 truncated toward 0 (not floored). Used in calculation of the cosine adjust value in accurate sine mode.

**doCmd.** ( n — ) Execute command n, where n is an index into the table of command execution vectors, parse. This word makes sure that the command input from the input stream (the input FIFO) is a valid command and then executes it.

**DOES>** ( - ) An immediate word to define the runtime behavior of a word created in code space. Typically used for addressing or fetching values from a table. Uses \_DOES.

**DQO ( - addr )** Start of data queue 0.

**DQI ( — addr )** Start of data queue 1.

**fastsine ( — )** Fast sinewave mode command. Sets up the mode, the next output routine execution vector, and the sinewave frequency for the fast sinewave output mode. Expects the next word in the input stream to be the desired sinewave frequency.

**fifo@!** ( — n ) Read a value from the input FIFO on the ASIC bus. Reads from ASIC address 18 (hex).

**fifoFul? ( — fig )** Return a TRUE flag if the input FIFO is full. Reads the input FIFO status bits from ASIC address IB (hex) and masks bit 1.

**fifoMT? ( — fig )** Return a TRUE flag if the input FIFO is empty. Reads the input FIFO status bits from ASIC address IB (hex) and masks bit 0.

**filtCnt! ( n — )** Write an 8 bit value to the filter



	be the offset in 100's of microvolts. No scaling or bounds checking is performed on this value. The value input is subtracted from the scaled value of sinewave output.	choose (— n)	Select one of the 6 demos or quit.
setPeriod ( . period — )	Set the timebase sample rate (timer 0 load register value). Makes sure that it is at least the mode dependent minimum value. If in a sinewave output mode then it also readjusts the phase increments to be consistent with the desired output frequency and the desired sample rate.	cmdg! (-)	The start of a circular buffer structure in code space for simulating the hardware command FIFO. This command queue is preloaded with the command codes and associated parameters and data. Then the command execution software is revector to fetch words from the command queue instead of the input FIFO.
sine (- n) . . .	An 800 point table of sine values scaled to 2 VRMS. Used in both fast sinewave : and accurate sinewave output modes. In fast sinewave mode the output values are taken directly from this table. In the accurate sinewave mode interpolation is performed for values in between the table values. This table is in code space.	cmdQMT? (- fig)	Return a TRUE flag if the command queue is empty.
sineFreq (— addr)	A variable corresponding to the sinewave output frequency.	cqMax (— 512)	A constant. Maximum size of the command queue, in words.
start (—)	Start output command. Unmasks timer 0 interrupts.	dMenu. (—)	Display the menu for selecting a demonstration.
stop (—)	Stop output by disabling the timer 0 interrupt.	dVect ( n1 — n2 )	Return the nist demonstration execution vector. The value n1 comes from the word choose.
stop? (— fig)	Return non-zero if the timer 0 interrupt is masked off.	demo (—)	The high level word for initiating the demonstration.
table ( n — )	A defining word to create a data table of n values in code space. Returns the nth value when invoked at run time.	demoAsine1 (—)	Demonstrate the accurate sinewave mode with a 1 KHz sinewave output, 1 VRMS out, 900 mV offset, lowpass filter set to 1 KHz.
tbPeriod ( — addr )	A variable corresponding to the programmed value of timer 0 counter. The actual interrupt period from timer 0 will be one more than this value (in increments corresponding to the 8 MHz TCLK).	demoCircSq ( + )	demonstrate the circular buffer operating mode by outputting a 1 KHz square wave between +1V and -1V. Filter cutoff set to 40 KHz.
toggleFC (—)	Toggle the filter clock output if not using the hardware timer to control it. Writes to ASIC address IF (hex) to cause the filter clock output to toggle.	demoFsinel (—)	Demonstrate the fast sinewave mode with a 1 KHz sinewave output, 2 VRMS out, lowpass filter set to 1 KHz.
twoPi (— 800)	A constant corresponding to the number Of points used to represent one cycle of a sinewave in the sine table.	demoFsine20 (—)	Demonstrate the fast sinewave mode with a 20 KHz sinewave output, 2 VRMS out, lowpass filter set to 20 KHz.
U> ( ul u2 - fig )	Return TRUE if unsigned ul is greater than unsigned u2.	demolnit (—)	Initialize the software for running the demonstration. Performs the same initialization as the normal software with the exception of the read FIFO command vector.
UMAX ( ul u2 — u )	Unsigned version of MAX. Return the larger of ul and u2.	demoMain (—)	The demo version of the high level operating loop. This loop is the same as the normal main loop, except that it terminates when the command queue is empty.
UMIN ( ul u2 — u )	Unsigned version of MIN. Return the smaller of ul and u2.	demoonce ( + )	Demonstrate the single + buffer operating mode. Generates one cycle of a 100 Hz sine that is linearly attenuated to zero.
writePtr ( — addr )	A variable. Contains the address of the current data queue to load data into.	fixCQ (-)	Adjust the tail pointer and end pointer of the command queue after loading the queue to account for its actual size.
_DOES (—)	A low level word in building GOES> and DOES>.	high (— addr)	A variable used for some simple performance testing. Used to keep track of the maximum number of clock cycles to execute tst.
_readFifo (— n)	Read a value from the input FIFO. Waits until there is a value in the FIFO.	low (— addr)	A variable used for some simple performance testing. Used to keep track of the minimum number of clock cycles to execute tst.
<b>Demonstration Software Glossary</b>			
>cq ( n — )	load n into the command queue. Used for simulating the hardware input FIFO.	shoParms (—)	Show some of the parameters for the current operating mode if the operating mode is a sinewave mode. Displays the current values of phase, phase increment, delta phase, delta phase increment, frequency, offset, and scale value.
_rdCQ ' (— n)	Return the next value from the command queue. The normal command fetch is revector to this word for the demonstration.	shoSparms (—)	Displays some of the variable values relevant to the current operating mode. Shows the values pertinent to a sinewave if in one of those mode and displays the lowpass filter cutoff and the output sample period.
atst (—)	A simple performance test to find the minimum and maximum number of clock cycles to execute tst. This loops 10000 times.	tst (-)	A test loop similar to the normal operating loop. Used for some simple timing tests. This word puts a new value into the DAC and then generates the next output value and puts it into the holding register.
btst (-)	A simple performance test to find the minimum and maximum number of clock cycles to execute tst for the ping pong and once out demonstrations. Repeats tst 40 times.		

---

---

# Real Computing

## RBOCs, C Sickness, HST Modems, Metal and More on Minix

By Rick Rodman

---

---

### A Dark Day in the Computer World

July 25, 1991 was a dark day in the history of computers. On that day, Judge Greene reluctantly permitted the RBOCs into the information-providing business. He was not pleased with this himself, but felt that the Justice Department had left him no choice. He foresaw that this will lead to "the elimination of competition from that market and the concentration of the sources of information of the American people in just a few dominant, collaborative conglomerates, with the captive local telephone monopolies as their base."

Why is it so bad that the RBOCs are allowed into this business? After all, won't it result in more and better choices for the customer? Let's look at this question.

The RBOCs are not like any other company. Remember, they have millions of customers who have no choice but to pay what they charge. Wherever you live in the U.S., you have exactly one choice as to your local phone service, and that is whether to pay what the local monopoly charges you. They can't lose money—they're guaranteed a profit by the FCC and the PUCs. A company that can't lose money is not a fair competitor to a company that can.

The other reason they can't compete fairly is that they set the rates charged to their competitors, but don't pay them themselves. How would you like it if your competitors set your costs but didn't have to pay them?

As Judge Greene says, it is inevitable that GENie, Prodigy, CompuServe, Delphi, and whatever other information providers there are, will either be acquired or go out of business. Look at what's happened to the cellular telephone industry: All of the major cellular carriers are now owned by RBOCs. Has this resulted in lower prices, better quality or any other benefit to the consumer? In the words of the immortal Doug Mackenzie, "No way, eh!"

The RBOCs have a history of providing poor service to their captive customers while at the same time tugging and poking at every restriction on them. Deregulating them is anticompetitive, anticonsumer, antidemocratic. If things continue to progress in this fashion, the RBOCs will soon control cable television, too. When they tell you you'll get better service and more choices, think about how much better and cheaper your local telephone service has gotten, and how many more companies you had to choose from. Don't blame it on the MFJ—the MFJ only split the greedy, monopolistic Bell System into seven greedy, monopolistic RBOCs.

### setjmp and longjmp

Ask any C programmer about *setjmp* and *longjmp*, and he's likely to mumble something along the line of "those are the functions you're not supposed to use or know about." But these are handy functions to have around.

See *Real/53*, page 18

---

---

# Real Computing

## X-10 Revisited, Mach, Minix and Desqview/X

By Rick Rodman

*[There are certain disadvantages to publishing a magazine after hours. An obvious one is where you rush into town between business trips to get ready for press, hurry through paste up, run off to the printer and finally to the post office and then find out you dropped the second half of an article! Of course, no one will notice such a gaff, right? Right. My apologies to Rick, and to those who took the time to point out my error. Following is the Real Computing installment from last issue—in its entirety!—Ed.]*

### X-10 Revisited

In issue 48 I wrote glowingly of the X-10 Powerhouse computer interface, known as the CP-290, that "any X-10 commands which come across the line are converted to status messages that the computer can see". Mike Morris writes: "I have a CP-290, and mine

does not produce these messages when I push a button on a manual controller."

After some experimentation and consulting with the X-10 gurus at Home Control Concepts, I have to admit Mike is right. The CP-290 does report to the computer when buttons are pressed on it itself, but it doesn't report codes that come over the wire. It is a send-only, unidirectional device.

Actually, the plot's even thicker than that. Manual controllers, like the wall switch modules, don't send codes at all, they only receive them. There is no way for a central controller to know

See *Real/52*, page 19

---

*Rick Rodman works and plays with computers because he sees that they are the world's greatest machine, appliance, canvas and plaything. He has programmed micros, minis and mainframes and loved them all. In his basement full of aluminum boxes, wire-wrap boards, cables running here and there, and a few recognizable computers, he is somewhere between Leonardo da Vinci and Dr. Frankenstein. Rick can be reached via Usenet at uunetvirtechlrckr or via 1200 bps modem at 703-330-9049.*

Real!53, from page 17

One use of these functions is to implement coroutines, a language feature that C doesn't have but is usually available in Modula-2. As an example, consider the typical Mac/Windows/PM program. It has to be written as a "message loop", bottom-up style, using messy state variables to keep track of what's happening at any given time. But by using coroutines, you can separate the message loop from the main program logic, which can then be written in conventional, more readable, "top-down" style. (The implementation is left as an exercise for the reader.)

The *setjmp* function is supposed to keep enough state information for a C program to be able to be restored to that state later. This means it must save the stack pointer, frame pointer, and all register variables. In most NS32 implementations, registers R2 to R7 are used for register variables. The function is passed a *jmp\_buf* variable, which must be of array type so that it is passed by reference (address). It cannot push things on the stack, because it must return with everything as it was. It returns an integer which is zero when the environment has been saved, or nonzero if it returned via *longjmp*.

The *longjmp* function is passed a *jmp\_buf* variable and an integer value, which should be nonzero. When you call it, it doesn't return. Instead, the state information in the *jmp\_buf* variable (whether it's valid or not) is loaded and execution continues by returning from the *setjmp* function again.

The reason I became interested in these functions is the overwhelming complexity of even the simplest multitasking routines. Without an emulator, it could be simply impossible to debug them. However, with non-preemptive or "cooperative" multitasking, it is not necessary to be so concerned about the machine state, nor deal with the hardware on such an intimate level. So, by using the *setjmp* and *longjmp* calls, plus one additional call which imitates *setjmp*, the basics of Metal's kernel are being constructed from simple C calls. This kernel will be message-based internally, using int values as messages.

Presently, Metal 0.7 is available in an installation kit including all source files. To obtain a copy, simply send me sufficient formatted floppies for 1.4 megabytes in a reusable mailer with return postage, and I'll put it on them and send it back to you.

### C Sickness

After upgrading my C compiler to 6.0, I spent literal hours tracking down long math problems. It seems that Microsoft, in version 6, is even more aggressive in truncating values to ints. Consider the following statement:

```
long_variable = 0x40000L; long_variable = long_variable >> 1;
```

In older versions, and in most C compilers, the value resulting would be 0x20000L. But not Microsoft C 6.0! You see, the shift count 1 is an int, giving them *carte blanche* to truncate the other side of the expression to an int too, causing the result to be zero. So, even though a shift count of more than 16 bits is ridiculous, the shift count has to be changed to a long. Who asked for this?

### Minix on the PC-532

Loading the Minix system onto the PC-532 is a tedious process. You have to download each of several big files into RAM in the debugger, write it to an unused hard disk parti-

tion, boot Minix, "neat" the partition into a file, decompress the file, then "tar x" the file. Just transferring the data takes a long time; even compressed, there's about two megabytes of data to transfer.

So what's it like once the transferring is complete? How's the "feel" of the system? Well, Minix on the PC-532 is probably best characterized as fast, but sparse. Only the barest minimum of necessary files are included with the archives. The remainder of sources can be transferred from another Minix system, or obtained from Internet archives if you have access, or from various mail servers. (A "mail server" is a file archive which works through E-mail. You send it a message like "index" or "send foobar.txt" and it sends you its index or the requested file as a mail message.)

I've just discovered that I can't use the C compiler, because my system doesn't have an FPU. The compiler provided is Gnu C. A small Emacs clone called "xemacs" is provided; it's similar to, but not the same, as "elle", provided with Minix 1.5. Presently I'm running the 1.3 kernel. Probably by the time you read this, I'll have upgraded to the so-called "1.5 hybrid" kernel, which is a mixture of 1.3 and 1.5, and hopefully I'll have a patched C compiler running as well.

In other Minix news, many people have complained that the elvis vi clone included with Minix 1.5 dies with an "alarm clock" message. (Should we capitalize "elvis"? it's named after a "pop singer", I think.) At any rate, the mystery of this message has been solved. It's related to the "keytime" function for slow arrow keys on serial terminals. To disable the "keytime" and prevent the crash, use the command ":set kt=0", or create a file ".exrc" in your home directory and put a line "set kt=0" in it.

The source for elle is not included with Minix 1.5, but the source to elvis and mined are.

### The US Robotics HST Dual-Standard Modem

My BBS is now equipped with a US Robotics HST Dual-Standard modem, acquired through US Robotics' sysop program. This modem is capable of operating in V.32bis or in HST mode, both of which are capable of throughput up to 14.4 kb/s. Further, it is equipped with V.42 data compression as well. Now we're talking fast! Feel free to call up and try it out, and make comments.

### Next time

More on multitasking and the NS32. Also, I've been playing some more with my X-10 devices; and, I hope to have more to report on Metal and on Don Rowe's Kotekan operating system. In the meantime, may your sizeof int's and your sizeof long's ever be equal.#

### Where to write or call

BBS: 1 703 330 9049  
(evenings; 1200, 2400, 9600, or even higher!)

Metal OS, c/o Richard Rodman  
8329 Ivy Glen Court  
Manassas, VA 22110

Home Control Concepts  
9353-C Activity Road  
San Diego, CA 92126  
1-619-693-8887

*Real/52, from page 17*

whether a user has manually turned a light on or not.

There is another computer interface available, which is called the TW-523. This device has an optoisolated TTL-level interface which can be operated through parallel input and output bits. It's "dumber" than the CP-290—it has no battery backup or timers—but it's bi-directional: the computer can send or receive any X-10 code over the power lines. Remember, the manual buttons on the wall switch units don't send any code. However, you can receive codes sent by other controllers or by motion detectors.

If you're planning a system controlled by a dedicated computer, the TW-523 sounds like the way to go. It's available from Home Control Concepts for \$30. It's also available as part of a Powerline Interface Kit for \$69, which includes a C library and sample code as well as a cable for connection to a PC parallel or serial port.

HCC also has available some wireless motion detectors. These detectors send an on-code when motion is detected, and an off-code when motion stops, through the same base unit used by the wireless hand control. Because they're wireless, you can put them anywhere—out in the yard, garage, parapet, moat, wherever. Because they send a code, you can not only turn on a light or sprinkler system, you can also detect the code in your TW-523 and activate a voice synthesizer or camcorder, or switch speakers connected to the stereo, or anything else.

Mike Morris suggests some neat applications for X-10: "I want to be able to turn on the sprinklers when the ground gets dry or if the burglar is leaving, but not the day after a rainstorm. I want to run the dishwasher after everybody goes to bed, then turn the water heater off but back on an hour before I get up. And divert the clothes dryer exhaust into the furnace air intake when appropriate so the furnace grabs the preheated humidified air."

Mike also provided some "from the trenches" information on X-10: "The X-10 signal can have problems 'jumping' across the two halves of a 220 volt household circuit. I ended up acquiring a capacitor of the proper value, installing it in a 220v plug and plugging it in. 'One of these days' I'll install the Leviton capacitor module in the main electrical panel, but for now, it works."

He also points out that some X-10 lamp modules "forget" their brightness setting and will "creep" up or down over several hours, and suggests "refreshing" their settings periodically. (That'll also solve the problem of people using the manual buttons—and staying too long in the bathroom, too!) He points out that the CP-290 doesn't keep accurate time, losing about an hour per month. Mine has forgotten its house code a couple of times. "The battery compartment in the CP-290 does not have a barrier between the battery and the circuit board. Mine leaked and ate away a few traces, which I repaired with a small soldering iron and 30 gauge wire."

Not only X-10, but the whole field of home automation is booming in popularity these days. The CE Bus, a fully-bi-directional superset of X-10, is coming. I'll venture a guess that the reasons for this sudden interest are (a) the increase in crime, especially senseless violence and vandalism, (b) higher energy costs, and (c) it's inexpensive and loads of fun!

#### **Mach on the PC-532**

Carnegie-Mellon has released a "Micro-Kernel" for Mach which contains no AT&T code. If you have ftp access (which

I don't), you can get it and play with it. It's called a "micro-kernel" because it includes no I/O or utilities. A "UX server" task, made apparently of pieces of Unix, is run to provide the missing capabilities. Some folks at the Helsinki University of Technology in Finland have been working on bringing it up on the PC-532. Recently, they reported that they have the kernel itself running. The Mach kernel is "small"—about a quarter of a megabyte. They are cross-compiling using the Gnu C compiler and tools.

At the same time, reportedly, CMU has a number of people working on a "free" (non-AT&T) Unix emulator, and the Free Software Foundation (Gnu folks) are supposedly trying to work Mach into Gnu in some fashion.

Later in the year, the PC-532 Mach will become a part of CMU's standard distribution. The Encore 32532 system is one of three "reference ports" provided with OSF-1, also. Hopefully someone at National will notice that these are not fax machines.

#### **Minix**

In the meantime, patches have been developed for a 386 version of Minix, under which the Gnu tools are used for software development. With all the "smallness" of Minix out of the way, it becomes reasonable to implement subsystems like TCP/IP and X Window under Minix. However, Andy Tanenbaum, the author of Minix, originally intended it as a tutorial operating system for classes, and he doesn't want tremendous increases in the complexity of the package. For this reason, it is unlikely that the 386 version will be available from Prentice-Hall. Instead, if you want to run 386 Minix, you will have to either obtain ftp access to Internet and spend hours transferring files so you can have the fun of trying to cobble the system together from a bunch of patches, after which you can have the joy of trying to debug it. Some call the result "Advanced Minix".

The files you would need are supposedly as follows:

```
12259 Dec 12 1991 pub/Minix/oz/mx386.tute.Z
      John Nalins tutorial.
3623 Jul 11 1990 pub/Minix/oz/mx386_1.1.01.Z
      patch to mx386.
45155 Jun 15 1990 pub/Minix/oz/mx386_1.1.t.Z
      tar file with 386 patches.
12063 Jun 14 1990 pub/Minix/oz/bcc.tar.Z
      tar file with 386 C compiler front-end.
121962 Jun 15 1990 pub/Minix/oz/bccbin16.tar.Z
      C compiler 16-bit binary.
118254 Jun 15 1990 pub/Minix/oz/bccbin32.tar.Z
      C compiler 32-bit binary.
43492 Jun 15 1990 pub/Minix/oz/
bcclib.tar.Z
      386 library sources.
96151 Aug 14 1991 pub/Minix/oz/cpp.tar.Z
      C preprocessor (optional?).
30659 Nov 15 1991 pub/Minix/oz/cppmake.tar.Z
      Earl Chew's eppmake program.
```

The Mars Hotel BBS in Maryland (1-301-277-9408) is supposed to have all these files. It also has a great deal of the comp.os.minix postings. For the moment, I've decided against bringing up 386 Minix, because I already have Minix with GCC on my PC-532.

Minix 1.6 is being beta-tested by some people out there.

*See Real/52, page 8*

---

---

# Zed Fest '91

## Where Beer and BIOS Meet

By Lee Bradley

---

---

*I Talk abounded at the '91 Trenton Computer Festival that 8-bitters should not wait a year to get together. It was decided to hold an affair in October. Lee Bradley, publisher of Eight Bits & Change, took a leadership role. I asked Lee to fill us in on what transpired. In perfect North American style, we claim all "Zed-Fest's," the festivals held in Germany as reported by Jay Sage notwithstanding!—Ed.]*

On October 18th, Ian Cottrell, his wife Nadine, daughter Jenny and son David headed south. John Anderson had sent his smoked YASBEC north a few days before to Paul Chidley; Paul put the smoke back in, for this is what is needed to repair anything smoked, according to Ian. Ian carried John's YASBEC with him.

By the 19th, the Cottrells, John, the working YASBEC, Lee and Linda Bradley, Steve Dresser, Howard Goldstein and Sigurd Kimpel had converged on Walt and Linda Wheeler's home in Nassau, NY. The keys CMAZE were pressed by John on his YASBEC. Lee hit <return>.

Zed Fest '91 had begun.

Others arrived. Jay Sage preceded Stephen Griswold by about an hour; Stephen entered the door muttering something about the map. What really happened with Stephen and the Sandwich Crew (both Lindas, Nadine and kids) remains fuzzy but the Crew, the sandwiches and Stephen united at about 3 PM, much to the relief of all. We later learned the delay was due to T-shirts reading "Our Next Husbands Will Be Normal." An unusual sandwich-making algorithm predated the Distributive Law added to the confusion: For Turkey = 1 to Turkeys\_Ordered; Get\_Roll; Dollup\_of\_Mayo; Turkey\_Slice; LettuceLeaf; Assemble; Wrap; Cleanup; Next Turkey. [Ed: Notice Cleanup is inside the loop?]

Ian got Trenton '92 on the agenda. Jay took notes: April 11 and 12 this year (early!). We'll set up a booth at the Flea Market and have people assigned time slots to man/woman/computer it. Demos and talks will be intermixed in the lecture hall as well, but the emphasis will be outside. Call us if you want to help. We need Banners, Books, Brochures, BDOS', BIOS' and Bodies.

We talked of having our own awards at our own banquet Saturday night. Why should everyone eat gray roast beef and be put to sleep by an acceptance speech when we can pick our own meal and our own winners?

John Anderson's wife Judy joined us for dinner at the Four Brothers restaurant with the memorable quote: "John wants to build a CPM LAN in our living room." The double anchovy pie eating contest was temporarily put on hold when anchovies were not found on the menu. Menus weren't a big thing for those present; we asked the waitress and found anchovies to be an undocumented topping. We ordered. Ian entertained those near him with some great new jokes and repeated the best on for those out of earshot when we returned to the Wheelers'.

Many of us descending into Walt's basement to call GENIE and join the 10 PM Saturday night CPM conference. Chris McEwen, Brian Moore and John Anderson (who had gone home to Albany) were on. Chris asked for a write-up on Zed Fest.

Sunday morning, after a run with Linda Wheeler, Stephen and Jay, we talked briefly of BYE5 and ZREMOTE. Ian mentioned ZMD 2.0 possibilities and then we all watched a video of Walt flying a parachute-equipped ultra-light. Apple pies and Mexican dip materialized after a three hour session during which Jay, Howard, Stephen, Walt and I tried to bring Walt up to Z33 (almost made it but a Trantor/Osborned system got us in the end—where were you, Daryl, when we need you?).

Jay left around 3 PM. My wife and I took Howard and Steve home to New Haven and West Hartford, rolling in at 9 PM.

The photos will have to wait for next issue as black-and-white processing takes longer than color. All in all, it was a lot of fun! Sure hope we will see you at the next gathering!®

---

---

### Computers: A Kid's Perspective

By David Cottrell, Ottawa ONT

I have had many opportunities to attend computer events all the country. I was at Zed Fest (I am Canadian), the fair at Trenton, and Walt Wheeler's get-together in Albany because my father, Ian Cottrell, always promises me fun-filled vacations.

Right.

No offense, but my idea of a good time is not listening to people talk about PBBS, RAM, ROM, Megs and hard drives. I'm into Ultima, Pac-Man and Super Mario. On the bright side, I have made many good friends and have seen some nice cities.

Maybe if I put my mind to it, I might learn what PBBS stands for and if \$300 is a good price for a hard drive. Oh, well. I still enjoy computer shows and if you hear Ian Cottrell will be at a show, then you can almost count on my being there. Now, if only someone would tell me what Eight Bits and Change is supposed to mean....®

---

---

# Z-System Corner

## Implementing a Keyboard Buffer Using the NZCOM Virtual BIOS

By Jay Sage

---

---

### First Some Personal Comments

Until I was 27 years old and met my wife, I had essentially never been out of the United States, and the same is true of my parents to this day. My whole family had the common American myopia that pictures the whole world as thinking and acting pretty much the same way we do.

My wife, on the other hand, came from a very different background. She was born in one country and spent several years living in a second before she came to the United States. Her father's history was similar. When my wife was born, he was living in his fourth country, and today he lives in his sixth!

It was not until our honeymoon that I made the dramatic step of leaving North America and setting foot on another continent. My life and outlook were irrevocably altered. A few years later when my employer offered me the opportunity to go to Japan and spend a year working at the Toshiba Central Research Laboratory, I jumped at the opportunity. It was a splendid year, and I loved Japan. The very different culture, however, helped me see my own culture in a way that was probably not possible without that experience. I came back with an enormously greater appreciation for what my own country offers, an appreciation that continues unabated.

My wife, keenly aware of the importance of a world perspective, wanted to get our children started early. She had the idea that, rather than simply visiting with her parents in Switzerland, we should take the opportunity to expose our children deeply to European culture by enrolling them in the public school there. (The school calendar in Europe is different, and our children can get in about six weeks of school after their school here ends.)

Since we preferred that the children learn a language more widely spoken than a Swiss dialect of German, we decided to rent a vacation house in the mountains of the nearby Black Forest in Germany. This is a wonderful place to spend a vacation, and the children learn the language that my wife's family has used since coming to the United States. Last summer was the sixth year, and I am always thrilled to hear the children talking comfortably with their German friends. Last summer, rather than sleep late and have us drive her to school, my 11-year-old daughter preferred to get up early enough to catch the school bus at 6:30 am (!) so that she could spend the time with her friends on the bus.

### Computer User Groups in Germany

Now we come to the first of the connections between this story and my *TCJ* column for this issue. In the first few years I was in Europe, I tried very hard to promote Z-System there, but it was a very frustrating experience. I knew that MS-DOS computers had not yet made the same inroads that they had here and that even businesses were still using CP/M computers. However, in Germany and Switzerland, the culture apparently did not lend itself to the formation of user groups as in the United States (and England and Holland). I was able to find a few individuals interested in 8-bit computing, but no user groups. Those individuals confirmed my impression; they, too, felt all alone.

One year I traveled to Munich to visit a fellow owner of a BigBoard I computer. I learned of him when he wrote a letter to *MicroCornucopia* magazine. While in Munich, I wandered into computer stores, each time asking if they knew of any CP/M clubs in Germany. The answer was always no. I did find one salesman, Zvonimir Racic, who was interested enough to start one, but it did not last even to the next summer. I have had no further contact with him.

Since I was having no luck searching for 8-bit enthusiasts in person, I decided to try another approach. With great effort (and help from my wife and a German colleague at work here), I composed a letter-to-the-editor in German to one of the major German hobbyist magazines. They never even acknowledged it. At that point I gave up trying.

### ZedFest Germany!

Given those earlier experiences, you can imagine how excited I was this last summer to be at the first European Z-

---

*Jay Sage has been an avid ZCPR proponent since the very first version appeared. He is best known as the author of the latest versions 3.3 and 3.4 of the ZCPR command processor, his ARUNZ alias processor and ZFILER, a "point-and-shoot" shell.*

*When Echelon announced its plan to set up a network of remote access computer systems to support ZCPR3, Jay volunteered immediately. He has been running Z-Node #3 for more than five years and can be reached there electronically at 617-965-7259 (MABOS on PC Pursuit, 8796 on Starlink, pw=DDT). He can also be reached by voice at 617-965-3552 (between 11 p.m. and midnight is a good time to find him at home) or by mail at 1435 Centre Street, Newton Centre, MA 02159. Jay is now the Z-System sysop for the GENIE CP/M Roundtable and can be contacted as JAY.SAGE via GENIE mail, or chatted with live at the Wednesday real-time conferences (10 p.m. Eastern time).*

*In real life, Jay is a physicist at MIT, where he tries to invent devices and circuits that use analog computation to solve problems in signal, image and information processing. His recent interests include artificial neural networks and superconducting electronics. He can be reached at work via Internet as [SAGE@LL.MIT.EDU](mailto:SAGE@LL.MIT.EDU).*



ing one of the discussions I was told about an important issue that constitutes the second connection between computers and my personal comments earlier.

### Programming for Compatibility—Again!

In my previous column I talked about making Z-System programs compatible—or at least tolerant—of vanilla CP/M. Ideally, Z-System programs would work to the extent possible in whatever environment they found themselves; at the least they would terminate in a graceful manner. It is too early to tell how responsive the community will be to my message, but the initial indications are very positive. Howard Goldstein, after proofreading the draft of my column, immediately fixed up LPUT and LBREXT. As soon as Rob Friefeld received and read my column, he started to look at his programs.

There is a second issue in programming for compatibility, and that is programming for compatibility in culture and language. For years, we in the United States have been writing our programs with only our own keyboards, our own screens, and our own language in mind. We did not do this out of disrespect or deliberate disregard; we did it because nothing made us think of a more cosmopolitan picture. In our experience, only Americans were using Z-System. That has now changed, and we need to change.

I won't be able to cover all the implications of this here. For one thing, I do not yet know myself what all the implications are. We will need to communicate with and

learn from others in the world who are using and developing 8-bit computers.

There are two issues that I will mention here. The most important one is the use of special characters. In the United States, we are quite accustomed to using various special characters, often for pseudo-graphics. For example, we may use the verticule (vertical bar) character (ASCII 7C) as a separator. Most European languages (and probably non-European languages, too) have characters with accents, and these are represented by the ASCII codes for special characters. In Germany, for example, the following associations are apparently used:

```
[ upper case A with umlaut
\ upper case O with umlaut
] upper case U with umlaut
{ lower case a with umlaut
| lower case o with umlaut
} lower case u with umlaut
```

When we include those characters in our programs, the screen displays in Europe (and Canada, for that matter) are often not pleasant.

The second issue concerns language. For example, programmers in the United States generally assume that yes/no questions will be answered with 'Y' or 'N'. In Germany, however, it would be nicer if 'J' (for 'ja') and 'N' (for 'nein') could be used. In France it would be 'O' (for 'oui') and 'N'

Listing 2. Rudimentary version of the KEYIN utility for filling the keyboard buffer in KEYBIOS.

```
; Program:          KEYIN
; Author:           Jay Sage
; Date:            October 1, 1991

; This program manages the keyboard input buffer in the
; special version of the NZCOM virtual BIOS called KEYBIOS.
; It interprets the command tail and adds the indicated
; input to the keyboard buffer (if it fits).

sigoff equ 97H          ; Offset to signature

cr equ Odh
lf equ Oah
bell equ 07h
tab equ 09h

; Library routines

extrn z3init, eprint

keyin:
    jp start
    db 'Z3ENV'
    db 1
env:   dw 0              ; Filled in by ZCPR33+
       dw keyin         ; For type-4 version

; Target signature that identifies KEYBIOS.

sign:  db 'KEYIN'       ; KEYBIOS signature string
nsign equ $ - sign     ; Length of signature

; Signon message

signon:
    call eprint
    db 'KEYIN, version 1.1 (10/01/91)'
    db cr,lf
    db 0
    ret

; Show help message.

help:
    call eprint
    db 'Syntax: KEYIN string',cr,lf
    db 0
    ret

; We get here if the signature is not right. Report problem
; to user and terminate.

badbios:
    call eprint
    db 'KEYBIOS not present!',cr,lf
    db 0
    ret

start:

; Initialize environment

    id hl,(env)
    call Z3init

; Display signon message

    call signon

; See if there is a command tail.

    ld hl,80h          ; Point to tail buffer
    id a,(hl)         ; Get count
    or a              ; Test for zero
    jr z,help        ; If zero, show help screen

; Make sure that the KEYBIOS is running by checking for the
; signature.

    ld hl,(l)         ; Get warmboot address
    id de,sigoff-3   ; Offset to signature from
                    ; .warmboot entry
    add hl,de
```

(for 'non'). We also, of course, display all screen information, including prompts, in English.

Is there anything we can do to be more accommodating in these matters? I'm not sure about the whole solution, but I have some ideas. For yes/no questions, we could allow for all common language possibilities: 'Y', 'J', 'O', and 'S' for the affirmative, for example (do we need anything other than 'N' for the negative?). Another possibility is that we include a language configuration screen with the ZCNFG CFG file provided with programs. Simple language changes, such as the letters used for yes and no and the characters used for special functions, could be changed using this screen. ZCNFG might be able to handle more significant text items for programs with little text output.

When a program provides a lot of text output to the screen, there may be too many changes to handle with ZCNFG, and we might have to go back to the old overlay method of configuration. Our programs could come with source code for language overlays. Authors with skills in other languages might even take a stab at providing some of these overlays themselves. Otherwise, native speakers in

various countries could provide the overlays.

To make it easier to use such overlays, one would want to collect all the messages that a program uses in one area rather than scattering them throughout the program as we generally do today. Rather than using SYSLIB routines like PRINT that display characters provided in-line in the code, we should use routines like PSTR that display characters pointed to by a register pair. Such source code is not as convenient to read, but it would make patching much easier (and it makes debugging easier, too). Another small detail would be to provide some extra spacing between message strings in case a message in another language is longer. Except in very rare instances, the slightly longer code that results from this approach will not be an undue burden.

### New Z-Nodes

Before turning to the main technical subject, I would like to announce four new Z-Nodes that joined in the past two months. Dave Chapman in Victoria, British Columbia (604-380-0007), and Terry Bendell in Collingwood, Ontario (705-444-9234) are our two new nodes in Canada. Ewen McNeill

Id	design		; Point to signature
Id	b,nsign		; Length of signature
sigtest:			
Id	a,(de)		
cp	(hl)		
jr	nz,badbios		; Jump if no match
inc	hl		; Bump pointers
inc	de		
djnz	sigtest		; Test whole signature
; Now we are ready for the real task			
Id	a,(hl)		; Save the size of
Id	(kbufsize)	,a	; .. the keyboard buffer
inc	hl		; Point to pointer
Id	(kptradr),hl		; Save its address
id	e,(hl)		; Get its value into DE
inc	hl		
id	d,(hl)		
inc	hl		; Point to start of buffer
id	(keybuf),hl		; Save the address
Id	hl,buffer		; Point to workingbuffer
id	bc,0		; Initialize count
; Copy any remaining contents of keyboard buffer to working			
; buffer			
copy1:			
Id	a,(de)		; Get next character from
			; ..key buffer
Id	(hl),a		; Write to working buffer
or	a		; Set flag
jr	z,copy2		; If null, break out of loop
inc	de		; Otherwise bump pointers
inc	hl		
inc	c		; Increment count
jr	copy1		; Loop back for next char
; Now we have to add the new characters from the command			
; line. HL still points to working buffer. This code needs			
; to be extended with the full functionality of <a href="http://ECHO.COM">ECHO.COM</a> so			
; that control characters and other special characters (such			
; as semicolons) can be entered.			
copy2:			
Id	de,82h		; Point to second char (if
			; ..any) in command tail
copy2a:			
Id	a,(de)		; Get the character
Id	(hl),a		; Store it in working buffer
or	a		; Set flag
jr	z,fillkey		; If null, we're done
inc	de		; Otherwise, bump pointers
inc	hl		
inc	c		; Increment the count
jr	z,overflow		; If we reach zero,
			; ..definitely too many
jr	copy2a		; Else loop back for more
; Now we have to copy the contents of the work buffer back			
; into the keyboard input buffer.			
fillkey:			
Id	a,(kbufsize)		; Make sure there is room
cp	c		
jr	c,overflow		; Too many characters
Id	de,(keybuf)		; Get start of buffer
id	hl,(kptradr)		; Address of pointer
Id	(hl),e		; Set pointer to beginning
inc	hl		
id	(hl),d		
id	hl,buffer		; Point to working buffer
inc	be		; Adjust counter to include
			; ..trailing null
idir			; Copy it all
ret			; We're done
; We get here if the keyboard buffer is too small to hold			
; all the characters. For the moment we just give a message,			
; but the code should call the error handler if possible. If			
; there is no error handler, then the MCL should probably be			
; cleared.			
overflow:			
call	eprint		
db	'Too many	characters for	
db	'keyboard	input buffer.'	
db	cr,lf		
db	0		
ret			
; Data items			
dseg			
kbufsize:	ds	1	; Size of keyboard buffer
kptradr:	ds	2	; Address of next-char ptr
keybuf:	ds	2	; Address of beg of buffer
buffer:	ds	257	; Working buffer
end			

has established the first Z-Node in New Zealand. His node is running, I believe, on a mainframe computer. Those who call in at 64-4-389-5478 in Wellington will learn about other ways to connect to the system. Finally, we also have a new node in the United States. The Kaypro Club of St. Louis has turned its system, run by sysop Bob Rosenfeld, into a Z-Node. It can be reached at 314-821-0638 on the MOSLO/24 outdial of PC-Pursuit. Give these new nodes a call and welcome them to our ranks.

### An NZCOM Virtual BIOS Keyboard Buffer

Z-System's extended batch processor, ZEX, is a very powerful tool with many fascinating and effective uses. Like SUBMIT or the MS-DOS batch facility, it can carry out a sequence of commands. However, if this is all one needs, then alias scripts should almost always be used instead, since they impose no additional system overhead.

ZEX, because of all the powerful features it provides, takes up quite a lot of memory. I just ran the utility [TPA.COM](#) on my system. Invoked from the command line, it reported a TPA of 51.5K; running it under ZEX gave a figure of 47.5K. Thus ZEX cost 4K of memory: 2K for its own code and 2K because its RSX (resident system extension) locked in the 2K command processor.

The situation where ZEX has been indispensable is when one wants a script not only to invoke commands but also to provide 'interactive' input to a program. If programs can accept input on the command line, then an alias script can handle the situation, but many programs take their input only interactively.

Once ZEX is loaded, it remains in memory until all commands in its script have been totally completed. There have been numerous occasions when I have wanted to feed just a short string of characters to a program before I proceed manually. In such a case, I really hated to suffer the memory penalty of ZEX to accomplish this. In some cases, such as with my database manager, the program would not be able to run with ZEX in place.

The solution I present here provides yet another example of the power of the NZCOM virtual BIOS. I simply added a little code to my normal virtual BIOS to implement a keyboard buffer, and then I wrote a utility to fill that buffer. The BIOS I called KEYBIOS; the utility I called KEYIN. A typical command line (probably in an alias) would look like:

```
KEYIN string for program:PROGRAM
```

KEYIN fills the keyboard buffer. Then, when PROGRAM runs and requests user input, the KEYBIOS sees characters in the buffer and returns them to the program. How this works will be clearer after you see the code for KEYBIOS.

### KEYBIOS

The new code contained in the virtual BIOS to support the keyboard buffer is shown in *Listing 1*. Here is what the code has to accomplish. When a program calls the BIOS to get a character, the virtual BIOS must first look in the keyboard buffer. If it finds a character there, then it returns that character and sets its pointer to the next character. If the keyboard buffer is empty, then the code simply passes the job on to the real BIOS, which will return a character actually typed at the keyboard.

One complication is that the BIOS also supports a function

(called console status) that asks if there is a character ready without actually fetching it. We have to fake out that call, too. We follow a similar strategy. We first look in the keyboard buffer. If there is a character there, then we report back that a character is ready. If there is no character in the buffer, then we pass the job on to the console status routine in the real BIOS. It is amazingly simple!

How do we do all this? Well, I think the code in *Listing 1*, with all its comments, is fairly clear. There are just a couple of things I would like to elaborate on.

The code includes two items that are not actually needed by KEYBIOS for its functioning. First, the code includes a signature string, 'KEYIN', at an established location. This allows a utility, such as the [KEYIN.COM](#) program that we will discuss in more detail shortly, to determine that the appropriate VBIOS is present. Following the signature string is a byte containing the length of the buffer. [KEYIN.COM](#) needs this to know how much space is available in the buffer. Without this information, it might overflow the buffer and clobber code.

The implementation of the buffer itself is much like the multiple command line in ZCPR3. At the beginning of the buffer there is a word that contains the address of the next character available from the buffer. A null character (ASCII value zero) is used to indicate the end of the buffer.

### The KEYIN Utility

*Listing 2* shows a very rudimentary version of the KEYIN utility that is used to add characters to the keyboard buffer in KEYBIOS. Again, the listing with its comments is largely self-explanatory, and I will elaborate on only a few issues.

First, in view of my earlier discussion, I am embarrassed that this code does not have the facilities for language invariance that I recommended. I was tempted to put them in for the listing, but I decided that there would be too much risk of introducing an error. Before releasing the full version of the utility, I certainly will follow my own advice.

The code does try to be quite rigorous. Once the program has displayed its signon message, it checks to see if any data has been passed on the command line. If there is none, a syntax message is displayed and the program terminates. In the final version, the code should check for the standard Z-System help request "/" in the command tail.

Next, the code looks for the signature string at the proper offset in the BIOS. If it does not find it, then an appropriate message is displayed and execution terminates. Otherwise, various information from the buffer header is fetched and stored for later use.

There may be characters already in the buffer, and KEYIN is designed to retain them and to append any new input. In the final version, one might want an option switch to flush any characters that remain in the buffer. To make the work easier, a temporary buffer in KEYIN is used to form the new contents for the keyboard buffer. Therefore, we start out by copying anything in the key buffer to the working buffer.

Next we append characters from the command tail. In the final version of KEYIN, the code should include all the special string interpretation techniques used in [ECHO.COM](#) so that control characters and other special characters that cannot be entered directly in the command tail (such as semicolons) can be included.

Since the structure of KEYBIOS is such that the buffer can never be longer than 255 characters, we monitor the number

of characters in the working buffer and abort if the count exceeds that value. Once the working buffer is completely filled, then we check the actual character account against the actual size of the keyboard buffer. If all the characters will fit, we move them into the buffer and set the pointer to the beginning of the buffer. KEYIN can then return control to the command processor.

If the characters will not all fit in the buffer, then we have to do something else. In the simple version in Listing 2, the program simply gives an error message and aborts. This leaves the key buffer unchanged. In a fully developed version of KEYIN, the error handler should be called so that the user can decide how to deal with the problem. If no error handler is available, then we have a more difficult problem. One course of action would be to clear the key buffer and flush the entire command line buffer (and terminate ZEX if it is running). Another possibility might be to clear the key buffer but allow the subsequent commands in the command line buffer (or ZEX) to run. The user would then have to do manually what KEYIN was trying to care of for the user.

### Very Important Caveats

As it turns out, dealing with characters at the BIOS level, as we do with KEYBIOS, involves some troublesome issues. When I first got KEYBIOS running, I was surprised that I always lost the first character that I put into the buffer. That missing character then appeared the next time the command processor prompt appeared. Many of you have probably seen a mysterious character like this, and you may have recognized it as something you typed earlier that was ignored.

It is beyond the scope of this article to deal with this issue fully. Basically, it derives from a fundamental flaw in the design of CP/M. As we have seen, CP/M has a function to get a character from the BIOS and to ask the BIOS if a character is ready. But there is no way to ask what the character is

without actually fetching it.

Why is that a problem? Well, the disk operating system (BDOS or equivalent) often provides special processing when the characters control-C, control-S, or control-P are pressed. Unfortunately, as we just noted, it has no way of finding out if one of those characters has been pressed without reading the next character. If the character turns out to be one of the three special ones, all is well; it processes the character.

But what if it is some other character. The BDOS would really like to say, "so sorry, not for me" and put the character back for the application program to read it. But it can't do that. So it does the next best thing. It puts it into a special buffer in the BDOS, and the BDOS plays the same kind of trick that we do in KEYBIOS. When a program asks for a character, BDOS first checks its special one-character buffer.

When programs perform all their character I/O using the BDOS or all of their I/O using the BIOS, then things will work reasonably well. However, if calls to the BDOS and BIOS are mixed, then characters can get lost in the BDOS buffer only to appear later when not wanted.

Every time I tried using KEYIN, I found that the command processor swallowed a character when it took control after KEYIN was finished. The pragmatic solution was to include a backspace character as the first one in the keyboard buffer, since the command processor would ignore it.

There are some other issues that KEYBIOS does not address. For example, some programs begin by flushing characters from the BIOS. Here is how they do it. They call CONST, and if there is a character they read it. This is repeated until CONST reports that no characters remain. Such a program will completely defeat KEYIN/KEYBIOS. Joe Wright has addressed these issues very carefully and cleverly in his IOPs (such as NuKey). KEYBIOS, however, is meant to be a very simple, minimal-memory solution, and it seems to do the job in my applications.®

---

*Editor, from page 2*

months. After all these years of reaching around the world with my computer, I guess I assumed everyone was doing it. This turns out to be a false assumption.

CP/M users seem particularly confounded by the Internet. There were no good tools in CP/M to use the net until last year. I have been hoping for an article on using David Goodenough's UUCP tools. Nothing has come up so far. We've drawn another blank with Fido compatible CP/M bulletin board systems, though there are at least three out there. Consider this a Call for Papers on telecommunications.

### On Today's Menu:

We had a short article in the last issue on a Z280 based system from Germany. The information was so new then that I did not even have a return address to the author! Tilmann Reh returns with more for us. This doesn't take away from the YASBEC, but it does prove an adage. When it rains, it pours. These two systems come after a drought of several years in CP/M hardware development.

Wayne Sung presents an interesting way to wire a large LAN with very low cabling costs — piggyback on your in-house cable television system. Of course, this presumes you have found a need to install a cable television system. Don't

be too quick to scoff the idea, though. University campuses are wired and the technology is transferable. Could this idea be expanded to community-wide networking over commercial CATV? Think about it while you read Wayne's article.

Jan Hofland returns with his *Arbitrary Waveform Generator*, discussing the software at greater length. You will recall that we had the schematics for this project last issue. We will finish the series next time with the last of his source code. Jan built this system to provide a test bed for structural testing, and used the Harris RTX2001 and EBForth. This is the kind of article I enjoy. Not your common everyday project!

Terry Hazen completes his topic on IOPs while Al Hawley continues with *Getting Started in Assembly Language*, discussing functions with structured programming. Both have proven to be very popular topics.

I would like to welcome Lee Bradley to our pages. He reports on the Zed-Fest held in Nassau, NY. Lee publishes *Eight Bits & Change*, which we have mentioned several times in the past. You will enjoy his humor. I always like to take a break when Lee's articles come in.

It seems that *Rcader-to-Reader* struck a popular chord. We have received a number of good letters. The direction this publication takes is given by you, and I always look forward to your feedback as do the authors. I will try to include as

*See Editor, page 45*



required by the function must be defined. In addition, there must be provision for error handling if error conditions can occur.

A function is invoked under two circumstances. The first occurs when the user selects a menu item to change; the MCASE routine transfers control to the appropriate IN\_FNx entry. After performing its task IN\_FNx transfers control to LD\_FNx. The second occurs when ZCNFG is first invoked; the initialization routine MAPPER invokes LD\_FNx for every configurable item in the CFG file to make screen image fields reflect the current configuration of the target file.

MCASE and MAPPER both perform a housekeeping task whose main purpose is to make functions easy to write in a standardized manner. They transfer the parameters from the selected Case Table Record to a standard 7 byte data structure at OFFSET: whenever a new configuration item is selected. The data structure is shown in *Listing 1*. The meaning and use of the byte at CFGD depends on the function being performed. For FN7, this byte is a mask which defines the sequence of bits (a field) in a target byte that is to be processed. Usage of CFGD and S\_LIST are fully described for each FN in the documentation distributed with ZCNFG.

There are two kinds of errors possible in most programs: recoverable errors and fatal errors. In ZCNFG recoverable errors are such things as user input which specifies an unimplemented menu choice or an input value which exceeds an allowed range. When a routine encounters a non-fatal error it simply returns without changing anything.

A typical fatal error occurs when the data in a newly loaded CFG file is not consistent with data in the target programs configuration block. This could happen if the wrong CFG file had been specified on the command line or during development of a new CFG file. For fatal errors, ZCNFG contains a list of error message exits, each of which prints an error message and terminates ZCNFG by jumping to a common entry in the main EXIT routine which leaves the target program unaltered. These fatal error exits are grouped together near the end of the ZCNFG source module.

The data structure at OFFSET, the fatal error exits, and the standardized convention for the Carry Flag comprise the Environment within which a function executes. Note that the

definition of environment is in terms of DATA passed to and from the function. Service subroutines like MADC from SYSLIB which sends a byte to the screen in DECIMAL radix, are logically part of the function, not of the environment.

## A Typical Function

*Listing 2* shows the code for the new FN7. This function rotates a bit from right to left in the configured byte. The movement is restricted to a series of bits defined by the mask. Thus, if the mask is 01111000, successive invocations of FN7

*Listing 1.* Data Structure Used by ZCNFG Functions

OFFSET: DS	2	;address of configuration data
CFGD: DS	1	;data byte for current function
S_ADDR: DS	2	/address of current screen field
S_LIST: DS	2	/address of string list or ;min/max data for current function.

might yield xOOOlxxx, xOOIOxxx, xOIOOxxx, xLOOOxxx, xOOOIxxx, etc. The x positions in the byte are not affected; they could contain other configuration data bits. The four bits that are affected might be tested in the target program to, for example, select one of four routines that display a date in 4 different ways. In this example ZCNFG requires (in the CFG file) a list of four strings to display on the screen because the field contains four bit positions. Each string would describe one of the four date formats.

An algorithm is a set of instructions for performing a computational task. It is usually stated in a high level language so that it is independent of the exact coding techniques of the implementation language. For fairly simple tasks, a set of instructions in English is adequate. For more complex computations, a pseudo-HLL language is more appropriate. Such algorithms may look like an inexact version of C or Pascal. I generally start a no trivial coding project with an algorithm like those shown below, making comments out of the steps and filling in source code after each step.

The following algorithm was used for writing IN\_FN7.

1. Use the Mask to isolate the bit field.

*Listing 2.* A typical ZCNFG function

```

/          FUNCTION 7
/          ROTATE A BIT WITHIN A FIELD IN A BYTE
/entry points: IN_FN7 for user update
/          LD_FN7 for initial screen load
/
/The following data structure has been filled in from
/the current record in a Case Table from the CFG file
/          (OFFSET:) = address of   cfg data byte
/          (CFGD:)   = mask defining the field
/          (S_ADDR:) = address of   screen image field
/          (S_LIST:) = address of   list of strings
;
IN_FN7:
/Rotate the bit field in the configuration block.
/exit- byte at (OFFSET:) contains rotated field.
/          Screen image is updated with a string
/          corresponding to the new bit position.
/          all registers used.

LD      HL,(OFFSET)    ;->configurable data
LD      E,(HL)         /data byte in reg E
LD      A,(CFGD)       /bit field mask
LD      B,A           /.. in B
CPL                      /invert it (logical NOT)

```

```

LD      C,A           /..not. mask in C
CPL                      /recover mask
AND     E             /isolate the bit field
RLCA                      /left circular rotate
LD      D,A           /save rotated field
INC     C             /
DEC     C             /:8 bit field?
JR      Z,INFN7X      /..done if so
AND     C             /test for overflow
                      /beyond field
                      ;z=no overflow
JR      Z,INFN7X
;a bit has shifted out of the high position
;of the field. The field is now all Os and
;/the shifted bit belongs in the Isb of the field.
LD      A,C           /synthesize a new field
RLCA                      ; with Isb set
AND     B             /remove all other bits
LD      D,A           ; save the initialized field
INFN7X: LD      A,(HL) /data byte
AND     C             /remove bit field and
OR      D             /replace with rotated field
LD      (HL),A        /new data byte
LD      A,D           /new field in A
JR      LDFN7A        /..for screen update

```

2. Rotate the resulting byte left one bit position.
3. Use the inverted Mask to test for overflow outside the bit field boundaries.
4. If overflow has occurred, create a new bit field with Isb set.
5. Replace the bit field in the configuration byte with the field from 2 or 4.
6. Update the field in the configuration block data
7. Use LD\_FN7 to update the screen image, skipping over the test for an illogical mask.

As the code was written, the algorithmic steps were paraphrased and expanded in the comment field. Note the coding strategy: the byte to be configured is stored in E. The mask is stored in B and its inversion is placed in register C. The rotated bit field is stored in D, where it can be easily replaced in case of overflow (step 5).

Note the sequence 'INC C, DEC C' to test for an 8 bit field. C contains the inverse of the mask and will be all zero bits if the field is 8 bits wide (the entire byte). This sequence is a simple way to set the Z flag from any 8 bit register without changing its value. If the value is zero then the RLCA instruction has properly rotated the byte and there is no overflow to be concerned about. Otherwise, a bit may have been shifted out of the msb of the field, leaving the field itself all zeros.

The first 8 lines of code in LD\_FN7 establish HL as a pointer to the data being configured, load the isolated (masked) bit field into A, and perform an error check on the mask byte. IN\_FN7 has been designed so that, at the end of step 6, HL and A contain the same two values. The entry into LD\_FN7 skips over these first 8 lines because the mask check is not needed; it was done during program initialization by a call to LD\_FN7. Those 8 lines are executed just once. We'll have more to say about them later.

The algorithm for LD\_FN7A...LD\_FN7X is:

1. Shift the bit field right until the Isb of the mask is at bit position 0.
2. Get the pointer to the list of strings (S\_LIST)
3. While <bit field Isb> is zero  
Shift bit field right

Advance pointer to next string

End While

4. Copy the string to the screen image (at S\_ADDR)

Advancing a pointer to the next null terminated string and copying a null terminated string from one place to another are tasks performed in many places in ZCNFG, so these tasks were written as subroutines in a separately linked module, CFGSUBS.Z80. It is typical practice to use subroutine calls for common tasks; ZCNFG depends on subroutines from rel libraries as well. To make use of such subroutines, the programmer must become familiar with what is available in each library (or linked module) and how to use each subroutine. CFGL1B, written for ZCNFG, is accompanied by a HLP file which explains how each subroutine is used and what it does. VLIB, Z3L1B, SYSLIB, and others are similarly documented. Studying such documentation carefully and then using library routines saves a lot of coding and debugging.

Did you try solving the problem stated at the beginning of this article? It sounded a little theoretical, didn't it? The obvious approach is to rotate the byte 8 times, counting 1 bits, followed by a test of the count to see if it is less than 2. That approach will work. It requires a loop (typically implemented with DJNZ), a conditional jump around the code which increments the counter, a compare statement, and code to initialize the bit counter and the loop counter. Such a routine can be written as a subroutine which takes about 17 bytes of code, including the RET statement. The cost in execution time is about 86 T-states. If you got it done in only three in-line bytes, congratulations!

This problem was originally published in Dr. *Dobbs Journal* about 10 years ago, but with an extra qualification that made a real challenge of it. The code was to be *three bytes* long! I didn't solve it. I had to wait for the next issue for the answer. And ZCNFG is the first time I actually had use for it! Here's the explanation and a little rationale for being aware of such things. They are called 'coding tricks', but really are methods of making best use of your CPU architecture and instruction set. Lee Hart gives an excellent discussion of performance oriented programming in *TCJ* issues 39 and 40

```
LD_FN7:
;Copy the string in the overlay corresponding
;to the bit set in the field to the screen image.
;This is the entry for MAPPER, during initialization
;The data structure at OFFSET: contains current data.
;exit- (normal) AF = 0,NZ,NC
;
;      (error) program abort with message
;      on bad CFGD byte.

      LD      A,(CFGD)
;test for illogical mask with          less than 2 bits set
      LD      B,A          ;save the mask byte
      DEC     A          ;invert low order 0s and
                        ;the first 1 encountered
      AND     B          ;restore 0's from original
                        ;byte and set Zflag
                        ;(NZ if 1's remain)
      JP     z,BADOVL    ;error if <2 bits in the mask
      LD      A,B          ;restore original byte in A

      LD      HL,(OFFSET)
      AND     (HL)        ;bit field at (HL) in A

LDFN7A:
;This is the entry from IN_FN7.
;shift bit field right to Isb position for indexing
```

```
/entry- A = field from cfg data, masked
;      B = mask for field data
;      HL ->  cfg data byte
      BIT    0,B          /mask & data
                        /right justified yet?
      JR     NZ,LDFN7B   ;yes, if nz
      RRCA          /move bit field right
      RRC     B          ;and also the mask
      JR     LDFN7A     ;and repeat

LDFN7B:
;index to the      proper string in      .S_LIST '
      LD      HL,(S_LIST)  ;->field data list for screen
      LD      C,A          ;cfg field in C
..INDX: BIT    0,C          ;lsb set?
      JR     NZ,LDPN7X    ;use current list item if so
      CALL   SKIP2Z       ;else skip to next list item
      RRC     C          ;and put next field bit in Isb
      JR     ..INDX      ;..and repeat

LDFN7X:
;copy S-LIST      string to screen      image
      LD      DE,(S_ADDR)
      CALL   MOVE2Z       ;copy null terminated string
      XOR     A          ;no error
      RET
```

(July & Sept, 1989). This is highly recommended reading! But Mr. Hart does not discuss this 'trick'.

By now, if you've been reading the listing of LD\_FN7, you will have found the three bytes. They follow the ';test for.....' comment line. Consider what happens when a byte is decremented (subtracting 1 is equivalent but takes an extra byte). Assume that the byte in A is 01111000, our first example. When this byte is decremented, it becomes 01110111. Note what has changed: all the 0's at the right have become 1's, and the rightmost 1 has become a 0. Bits to the left of the rightmost 1 are unchanged. If this result is ANDed with the original byte, all the changed bits become 0, and the value of the byte is determined only by the unchanged bits. If these bits were originally 0, then the result will be 0. *Figure 2* shows three examples, one for a reasonable mask byte, and two for illogical mask bytes. These latter two are illogical because the mask byte specifies the size of the bit field to process; it makes no sense to rotate a bit in a zero length field or in a field of length one. Since the final AND adjusts the Z flag, a result of NZ implies an acceptable mask and Z an invalid mask. An invalid mask can arise from an error in writing the CFG file or from a damaged file.

### Error Checking

The current example illustrates a type of error that would normally occur only when developing or changing a CFG file: an erroneous value in a case table record. Such an error could arise by forgetting to include an item in the record, making it too short. Because such mistakes are hard to find, the BADOVL error routine takes advantage of the table driven nature of the SETOPT routine. BADOVL uses the information in MENLST and OFFSET to report the Menu and Case table record number in which the error was detected. Except for the remote possibility of a damaged file, the normal user of ZCNFG will never see this class of errors, but the CFG file programmer will be thankful when they don't occur and grateful when they do!

### Exit

Some of the concepts important to structured programming have been illustrated with the implementation of a major function in ZCNFG. We saw that the environment of a program structure comprises Data Structures, register usage, subroutines both local and from other modules, and error handlers. The environment is more commonly referred to as the Interface of the routine with the rest of the program. Data flow analysis was introduced and is a basis for the

*Figure 2. Typical Mask Byte Testing.*

	Good Mask	Bad Maski	Bad MaskO
A	01111000	00001000	00000000
A-1	01110111	00000111	11111111
A & A-1	01110000	00000000	00000000

Algorithm(s) that describe the final goal of the AL code. Much of this is familiar to HLL programmers; such structuring is a major goal of High Level Languages. What is to be learned from the simple coding problem? Probably that there is usually more than one way to implement the solution to a problem, and that other solutions might be more efficient (or even more elegant!). Here we also saw a subtle emphasis shift in our view of one of the CPU instructions: what was viewed as an arithmetic operation is really a specialized logic function which can be used to advantage. INC and DAA may also be viewed as logical operators and might have some clever uses. Can you think of others?

I hope you have become convinced that you, too, could program a function for ZCNFG. Want to try? Take a look at Function 6 (parse/deparsed all or part of a filespec). It cries out for a re-write! Or make a new function whose purpose is to configure a printer control string. At the very least, use the techniques of structured programming to make your programming efforts more elegant and rewarding. There's nothing like clean source code when it comes to the inevitable debugging or enhancement of your program.#

*Reader, from page 2*

*Frankly, I'd feel pretty silly not extending the trial subscription offer to "chaps like you." I was a chap like you—spent twenty-six years in the tropics. Ever hear of Truk? Ponape? Palau?—Ed.*

I became aware of *TCJ* through the Usenet newsgroup *comp.lang.forth*. The articles that show up there periodically mention your magazine. It seems to be of a similar flavour to the late lamented *Micro Cornucopia* and also of early *Byte*.

I like this "older style" of magazine. It tells you how to get stuff done with what you've got instead of forking out big bucks for someone else's solution.

D.C., Ottawa ONT

*You have the basic idea. Not much money in putting out a rag this way, which explains why others aren't doing it. Most of the industry has outgrown its roots.—Ed.*

I appeal to Z-programmers to enforce CP/M compatible programming. An example: implementing a fixed environment array near the beginning of a file and to set the address at 109H, so that in a non-Z environment this fake environment would be used.

Second, I thought to bring the use of semigraphics to owners of poor terminals as well. The way to go would be to use a dummy graphics on/off sequence (or omit it totally) and to use "+" and "-" for Graphic Character replacements. This would certainly open truly portable graphical programs to everybody. It would mean reworking the Z3TCAP.TCP to this standard for all terminals, and then leaving it up to individual users to soup them up to the real stuff, except for those extended TCAPs already known.

Also I'd like to see a remake of TCMAKE, (which used to work nicely in the old days of simple terminals), that would create the new standard: TCAPs compatible to VLIB 4D menu-driven, so a typical Auto-Install-Z-system-user can do this himself!

H. J., Ismaning Germany

*You will appreciate Jay Sage's topic this issue.—Ed.*

I have been a journal reader now for about two years or more. I enjoyed it very much before, but I must say that I enjoy it even more, now. Your new ideas, perspectives and format are very good and I look forward to each edition. The

*See Reader, page 44*

---

# The NZCOM IOP

## A General-purpose IOP Loader Module

By Terry Hazen

---

When an IOP REL module is loaded into the NZCOM IOP buffer by NZCOM or JetLDR, you have no way to configure the IOP module. The only way its configuration can be changed is to modify the source code and reassemble and relink it.

This time, we'll look at IOPLDR, a small general-purpose IOP loader REL module that does most of the standard work that it takes to load, control and remove an IOP module. It is combined with an IOP REL module and some IOP module-specific routines to create a stand-alone COM file that will load, control and remove the IOP module and allow the IOP module to be configured by ZCNFG before loading.

We'll demonstrate IOPLDR by using it to create the stand-alone IOP clock display utility [IOPCLK.COM](#). IOPCLK provides the IOP clock-specific routines and messages needed by IOPLDR as well as the configuration area for use by ZCNFG. It also takes care of automatically finding the ZSDOS clock driver address at run time. Keep in mind that while IOPCLK is the specific example in this article, IOPLDR is a general-purpose module. You may use it to create a utility to load an IOP module you already have or a special one you've always wanted to write.

IOPLDR is based on RSXLDR, a similar generalized RSX loader I wrote for HP14 and is similar to, but less specialized than, the IOPLDR.REL module included in HP14. IOPLDR also incorporates parts of several other utilities, including Hal Bower's SPEEDLDR.Z80. Portions of the code were derived from Bridger Mitchell's *Advanced CP/M* column on the Plu\*Perfect CP/M 2.2 RSX standards in *TCJ* 34 and the word-wide relocater was derived from his column on relocation in *TCJ* 33.

IOPLDR is a generalized IOP loader module. It performs preload checks, environment validation and relocates the IOP

module to the IOP buffer. It is called by a module-specific IOP loader utility that contains all module-specific installation routines and messages. The actual IOP module is assembled and linked to a PRL file and appended to the end of the module-specific IOP loader COM file.

The IOPLDR module (*Listing 1*) and the loader utility, IOPCLK (*Listing 2*), are closely related, like interlinked fingers. Their relationship is also somewhat backward from the way utilities normally work with the REL modules they call. Here, IOPLDR contains all of the main routines and does all of the standard work. It's IOPCLK's job to provide all of the special routines and information that allow IOPLDR to deal with the specific requirements of the IOP clock module. In fact, even though the name of the utility is IOPCLK, IOPLDR is calling routines in IOPCLK rather than the other way around.

### IOPLDR

If you look at the IOPCLK code, you'll notice that the first thing it does when it is run, is jump to IOPLDR, which then takes control. IOPLDR starts out by doing all the mundane and necessary things a utility usually does. It saves the system stack pointer, sets a local stack, displays the sign-on banner and does some system checking to make sure it's operating under ZCPR3 with an extended environment.

If the system passes muster, IOPLDR locates the start of the PRL header for the IOP clock module that has been appended to [IOPCLK.COM](#). \$MEMORY is a word that the linker fills in with the address of the next available byte of memory after the last module used by IOPCLK has been loaded and resolved. Since IOPCLK will be assembled and linked without the IOP clock module PRL file, which will be appended later, the \$MEMORY pointer will help us locate the beginning of the appended CLKIOP.PRL file. We'll file the information away for future use and then locate and save the IOP module information we'll need for relocation later on.

Now that we know where to find the IOP module, we can get and save a pointer to its name so that we can use it in the help message. Only then do we check to see if help has been requested. When a help message is requested, you'll notice that the help routine, MHELP, displays several module-specific strings that are located in the calling utility, IOPCLK. IOPCLK contains MDESC, the description of the IOP module. The description for

---

*Terry Hazen has a background in analog electronic and mechanical engineering. He is currently a product design consultant, specializing in medical electronic systems. He encountered his first computer in the 1960's and got very frustrated trying to write small punched-card batch-processed ALGOL programs. He got his first Z80 computer in 1982 and has been pursuing Z80 hardware and software projects ever since. His company, n/SYSTEMS, produces the MDISK 1 megabyte add-on RAM disk for Ampro LB computers. MDISK also provides the Ampro with bank-switching capabilities for operating system expansion. Terry enjoys designing and building varied types of hardware and software projects, not all of them computer-related. His recent software projects include the HP and HPC RPN calculators and the REMIND appointment reminder utility as well as upgrades to his SCAN text file viewing utility and ZP file/disk/memory record patcher. He may be reached by voice at (408)354-7188 or by message on Ladera Z-node #2. His address is 21460 Bear Creek Road, Los Gatos, CA 95030.*



IOPCLK is 'Video Clock IOP'. MHELP can also display any special loader utility commands and loaded IOP commands located at MCML and MLDMSG respectively.

At MCML (in IOPCLK), we find a description of the 'D' option, which controls the IOP clock's display format. In this case, IOPCLK doesn't need a message at MLDMSG, so it simply returns. In more complex IOPs, the message at MLDMSG can display commands that can be executed by the loaded IOP. The Alpha Systems RECORder IOP, for example, has commands that allow you to turn the recording functions on or off and allow you to specify the names of the files to which the console and list output are being written.

If no help has been requested, we can now get the address and size of the NZCOM IOP buffer from the environment and make sure it exists and that it is big enough for our IOP module.

### Removing the IOP Module

Now we can parse the command line and see if we are being asked to load or remove our IOP clock module. If we're removing it, we only have to make sure that the current IOP module has the same name as ours. If it does, we deselect it by calling the SELECT routine in the IOP with B=OFFh. We can then tell the user that the module has been removed and quit. Even though it's probably not absolutely necessary, we do the name test to make sure that each IOP module is removed by its own utility just in case there's something special about it or about its removal process.

### Loading the IOP Module

If we're not being asked to remove the IOP module, we call MPARS in IOPCLK to parse the command line for any special commands and do any special system checks we might need. In our case, we want to check to make sure we have a working ZSDOS/ZDDOS system clock. We first use the extended DOS ID call 48 to make sure we are operating under ZSDOS or ZDDOS. If we are, we make sure we have a clock by using the DOS 98 call to do a trial clock read.

If the user has requested that we install the IOP, we copy the IOP module to the 10P buffer and using the PRL bitmap, relocate it in place by adjusting the module addresses to reflect the move (see Bridger Mitchell's *Advanced CP/M* column in *TCJ* 33 for more information on relocation).

Now that the module is in place, we call MCFG in IOPCLK to do any module-specific configuration that may be required. In our case, we need to find and save the location of the ZSDOS clock driver and we need to move the configuration buffer from the start of [IOPCLK.COM](#) to the loaded 10P clock module.

Like all good IOP loaders must, we then call the initialization routine in the 10P module. Before we can quit, we call MINST in IOPCLK to do any module-specific initialization. In our case, none is required. We then display the address of the IOP buffer and finally, call MLDMSG in IOPCLK to display any special load message our module might require. IOPCLK doesn't require anything special, so we're done. Our IOP is loaded, configured and running until we remove or replace it.

### Configuring the IOP Module

If you chose to load CLK10P.REL with NZCOM or JETLDR, you had to add all the configuration information to the CLKIOP source code before you assembled it to a REL

file. This means, of course, that you'll have to change and reassemble and relink the source code each time you want to change the configuration. To make configuration easier and more flexible, there is a configuration buffer at the beginning of [IOPCLK.COM](#). This buffer can be configured with ZCNFG and is copied to the IOP module configuration buffer after the module is loaded into the IOP buffer.

You can configure the 10P clock module to sound an alarm every hour, to display the time as hh:mm and update the display every minute or to display the time as hh:mm:ss and update the display every second, and to display the time in 12 or 24 hour format. This is also where you must specify the special terminal control sequence that is required to place the clock display in the terminal's host computer message field.

### Producing [IOPCLK.COM](#)

We can now assemble and link IOPCLK.Z80 with IOPLDR.REL and SYSLIB to make [IOPCLK.COM](#). Note that IOPLDR.REL requests the required SYSLIB routines, which are public:

```
ZMAC IOPCLK:ZML IOPCLK
```

We need to have the CLKIOP module available as a PRL file, which contains the bitmap that we need in order to relocate it (see Al Hawley's *Getting Started in Assembly Language* column, *TCJ* 50). First we assemble CLK10P.Z80 to a REL file and then link it to a PRL file:

```
ZMAC CLKIOP:ZML CLKIOP /P
```

We can then use CONCAT with the binary-append options to append CLKIOP.PRL to the end of [IOPCLK.COM](#) to produce our final stand-alone IOP clock utility:

```
CONCAT IOPCLK.COM=CLKIOP.PRL /OA
```

You can now run IOPCLK to load the 10P clock module. You can also use it to remove the IOP module and to change the display from 12 to 24 hour time.

You can find the full source and documentation for IOPLDR in IOPLDR11.LBR on your favorite Z-Node. I hope IOPCLK and IOPLDR will give someone an idea that will soon lead to a new and innovative 10P module showing up in these pages!®

## TCJ On-Line

Readers and authors are invited to join in discussions with their peers in any of three on-line forums.

- GENie Forth Interest Group (page 710)
- GENie CP/M Interest Group (page 685)
- Socrates Z-Node 32

For access to GENie, set your modem to half duplex, and call 1-800-638-8369. Upon connection, enter HHH. At the U#= prompt, enter XTX99486,GENIE and press RETURN. Have a credit card or your checking account number handy.

Or call Socrates Z-Node, at (908) 754-9067. PC Pursuit users, use the NJNBR outdial.

```

inc hl ; Point to first name byte
id de,name ; Move name to our buffer
id bc,8
ldir
;
getprl: ld hl,($memry) ; Get end of code
id a,l ; Find start of PRL header
or a
jr z,start2
cp 80h
jr z,starts
add 80h
jr nc,start1
inc h
;
startl: and 80h
ld l,a
;
start2: push hl ; HL=PRL header
inc hl ; Point to PRL size
id e,(hl)
inc hl
id d,(hl)
id (size),de ; Save PRL size
;
pop hl ; HL=PRL header
inc h ; HL=IOP
push hl
pop iy ; HL=IY=local IOP module
id (iop),hl
id de,oiopid
add hl,de ; HL=address of IOP name
ld (iopid),hl ; Save it
;
ld hl,fcbl+2 ; Check for help request
ld a,(hl)
cp '/' ; Help request?
jp z,mhelp ; Display help screen
;
; Get the IOP buffer address
;
getiop: ld l,(ix+15) ; IOP buffer address in the
; environment
id h,(ix+16)
ld (nziop),hl ; Save IOPbuffer address
;
ld a,(ix+17) ; Get IOPSize in records
or a ; Check for zero
jp z,noiop ; Quit, no IOP
;
rra ; *80h
ld h,a ; HL=size
ld l,0
ld (iops),hl ; Save it
or a ; Clear CARRY
ld de,(size) ; Size of IOP
sbc hl,de
jp c,smiop ; IOP too small for this
module
;
id hl,fcbl+1 ; Parse command line
ld be,8 ; Scan name field
ld a,'R' ; Check for 'R' remove
cpir
jr z,remove ; Yes, remove IOP
;
ld hl,fcbl+1 ; Parse command line
call mpars ; Do any special parsing and
; routines
ret c ; Error, so quit
jp nz,install ; Ok to install IOP
;
id hl,fcbl+1 ; Point to command tail
id a,(hl)
cp '' ; Any command tail?
jp z,install ; No, OK to install IOP
jp mhhelp ; Yes, must be bad command
;
; Remove IOP module
;
;
; Get IOP buffer address
; Offset to IOP name
; HL=IOP name
; DE=our name
; Same?
; No, don't remove
; Deselect IOP
;
; IOP module removed',cr,if,0
;
; eprint
; 'IOP select function not supported',cr,if,0
;
; eprint
; 'Remove current IOP module ',0
;
; de,rmsg
; push de
; call pn$0 ; Display name
;
; db ' with its own utility.'
; db cr,lf,0
; ret
;
; Call IOP module SELECT routine
;
select: ld hl,(nziop) ; Get IOP buffer address
id de,3 ; Offset to IOP SELECT
add hl,de
jp (hl) ; Call IOP SELECT
;
; Copy the IOP module to the IOP buffer
;
install: ld hl,(iop) ; HL=local IOP module
ld de,(nziop) ; DE=loaded IOP buffer
push de ; Save 4 copies
push de
push de
push de
id be,(size) ; Length of IOP-bitmap
ldir ; Move the module
call reloc ; Relocate the code
;
pop hl ; Restore loaded IOP pointer
pop ix ; HL=IX
call mefg ; Do module-specific
; configuration
ret z ; Configuration error
;
pop hl ; HL=loaded IOP buffer
id de,9 ; Offset to IOPINIT
add hl,de
call jphl ; Call the IOP install
routine
;
pop hl ; Restore loaded IOP pointer
call minst ; Do module-specific install
; routine
ret z ; Installation error
;
; IOP module installed at ',0
; Get IOP load address
; Display load address
; Display module-specific
; load message
;
; Main exit point
;
exit: ld sp,(stack)

```

```

ret
;-----
;
; Error returns:
;
; No extended Z-system present
;
notez: call    eprint
             db    'Requires ZCPR3 system with '
             db    'extended environment!'
             db    cr,lf,bell,0
             ret    ; And quit
;
; No IOP present
;
noiop: call   eprint
             db    'No IOP buffer!',cr,lf,bell,0
             ret    ; And quit
;
; IOP buffer too small
;
smiop: call   eprint
             db    'IOP buffer is too small!',cr,lf,bell,0
             ret    ; And quit
;-----
;
; Bridger Mitchell's word-wide PRL relocater (TCJ#33 p14).
; Patches IOP addresses after IOP has been loaded to high
; memory. Bitmap remains in local memory.
;
reloc: Id    a,i    ; Save interrupt vector in
AF''
             ex    af,af
             di    ; Disable interrupts while we
             ; use stack
;
             id    ix,0    ; IX=zero
             add   ix,sp    ; Save stack pointer in IX
;
             id    hl,(nziop) ; Relocation address
             push  hl
             exx
             pop   de    ; Set DE'=base address of
code
             dec   d    ; Compensate for IOOh org in
             ; PRL file
             exx
;
             id    sp,hl    ; SP=start of code - 1 byte,
             dec   sp    ; since we need to mark high
             ; byte
;
             id    hl,(iop)    ; Start of local IOP module
             Id    be,(size)    ; Length of relocated code
             add   hl,be    ; HL=bitmap
             id    e,l    ; Init the rotation byte,
             ; which will set CARRY every
             ; 8 bytes
;
rloop: Id    a,b    ; Check byte count
             or    c
             jr    2,rdone    ; Done
;
             dec   be    ; Reduce byte count
             rre   e    ; Set CARRY every 8 bits
             jr    nc,rsame    ; Not set
;
             Id    d,(hl)    ; D=next byte from bitmap
             inc   hl    ; Advance bitmap pointer
;
rsame: rlc   d    ; Shift bitmap byte into CARRY
             jr    nc,noof    ; No relocation needed
;
             exx
             pop   hl    ; Get word to relocate from
             ; stack
;-----
;
address add    hi,de    ; Relocate by DE'=load
             push  hl    ; Put it back
             exx
;
noof: inc    sp    ; Point to next byte of code
             jr    rloop    ; And deal with it
;
rdone: Id    sp,ix    ; Restore the stack
             ex    af,af    ; Restore interrupt vector
             ret    po    ; Interrupt was DI, so skip
             ei    ; enable
             ; Otherwise enable interrupts
             ret
;-----
;
; Display the IOP module name and the trailing inline message.
;
mname: push  hl    ; Save message pointer
             Id    hl,(iopid)    ; Point to name
             jr    pn$0    ; Display it
;
; Display 3 leading spaces, the filename and the trailing
; inline message.
;
uname: push  hl    ; Save message pointer
             Id    b,3    ; 3 leading spaces
             id    a,''
;
splp: call   cout    ; Display the spaces
             djnz splp
;
             Id    hl,name    ; Point to our name buffer
pn$0: Id    b,8
;
pn$loop:Id    a,(hl)    ; Get byte
             or    a
             jr    z,pn$dun    ; Quit at 0
;
             inc   hl    ; Point to next character
             and   07fh    ; Filter high bit
             cp    ' '
             jr    z,pn$dun    ; Quit at first space
             call  cout
             djnz pn$loop
;
pn$dun: pop   hl    ; Restore message pointer
             jp    eprint    ; Display trailing message
;-----
;
; Display hex load address, start new line.
;
at: call    ph14hc
             call  eprint
             db    'H',cr,lf,0
             ret
;-----
;
; Call (HL)
;
jphl: jp    (hl)
;=====
;
; Initialized data area
;
iop: ds    2    ; Start of local IOP module
size: ds    2    ; Size of PRL
nziop: ds    2    ; Start of NZIOP buffer
iops: ds    2    ; Size of IOP buffer
iopid: ds    2    ; Address of local IOP name
;
; Local stack
stack: ds    2    ; System stack pointer
             end
;
; End of IOPLDR.Z80

```

"There are a lot of lies going around.... and half of them are true."

—Winston Churchill

Listing 2

```

; Module: IOPCLK
; Author: Terry Hazen
; Date: 06/07/91
; Version: 1.0
;-----
;
; Equates
;
off equ 0
on equ Offh
bdos equ 5
bell equ 7
lf equ 10
cr equ 13
esc equ 27
;
; Program version for use in banner and CFG filename
;
i
ravers equ 10
;
; Entry points for module-specific addresses and routines
; required by the IOPLDR module:
;
; public name,mdesc,moml,mldmsg,mpars,mcfg,minst
;
; From IOPLDR we need:
;
; .request iopldr
; ext iopldr, iop.nziop,uname
; ext cout, eprint ,ph14hc,$memry
;=====
;
enter: jp iopldr ; IOPLDR does all the work
; db 'Z3ENV' ; ZCPR3 Utility
; db 1 ; Type 1
z3eadr: dw 0 ; Z3ENV address provided
; ; by ZCPR33+
; dw enter ; ZCPR34 pad bytes
;
; db 'IOPCLK' ; Default CFG filename
; db mvers/10+' 0 '*',mvers mod 10+'0'
; ; 8 characters total
; db 0 ; Termination
;-----
;
; IOP Configuration area - copied to IOP module at load time
;
; config:
beep: db on ; ON to beep every hour
seconds:db on ; ON to include seconds in
; clock display
time: db on ; ON for 12hr time,
; OFF for 24hr time
zsclk: dw 0 ; ZSDOS clock driver address
;
; Prefix to put clock display in terminal message line. The
; prefix and termination character are completely terminal-
; dependant. This example is for the WYSE75:
;
; term: db esc,'[>+\' ; Message prefix
; ds 3 ; 8 bytes total
; db 0 ; Termination
;
; termend:db '\ ' ; Message termination character
;
; spaces: db 22 ; Spaces to start of clock
; ; display
;
; cfglen equ $-config
;=====
;
; IOPLDR-specific routines called by IOPLDR:
;=====
;
; NAME is the zero-terminated loader name and version number
; for use in the banner.
;
; name: db 'IOPCLK vers '
; db mvers/10+' 0 '*',' ',mvers mod 10+'0'
; db 0 ; Termination

```

```

;
; MDESC is a zero-terminated description of the module
; function that is displayed after the module name in the
; command line help screen.
;
; mdesc: db ' Video Clock IOP',0
;-----
;
; MCML is displayed after the command line help screen to
; allow the display of extended command line commands.
;
; mcml: call uname
; db 'D Toggle time display between '
; db '(hh:mm) ) and (hh:mm:ss)'
; db cr,lf,0
; ret
;-----
;
; MLDMSG is displayed after IOP load/exists messages to allow
; the display of IOP commands.
;
; mldmsg: ret ; No message required here
;-----
;
; MPARS does any required special command line parsing, system
; checks and execution of commands other than install and
; remove.
;
; Entry: HL=FCB+1
; Exit: CARRY SET for error requiring exit without installing
; IOP Z set if system checks ok but no command found. Look
; forbadcmd. NZ set if system checks ok and command found.
; Install IOP.
;
; mpars: push hl ; Save pointer
; id c,48 ; Extended DOS check
; call bdos
; id a,h
; cp 'S' ; Must be ZSDOS:
; jr z,clkchk
; cp 'D' ; OrZDDOS
; jr z,clkchk
;
; call eprint
; db 'ZSDOS/ZDDOS required!',cr,if,bell,0
; jr erret
;
; Check for good clock
;
; clkchk: Id de,clock ; Check for clock
; id c,98 ; BDOS get time
; call bdos
; inc a
; jr nz,pars ; Clock present
;
; call eprint
; db 'Can't read clock!'
; db cr,lf,bell,0
;
; erret: pop hl ; Discardpointer
; scf ; Set fatal error
; ret
;
; Parse command line
;
; pars: pop hl ; Restore pointer
; Id a,'D' ; Include seconds in display?
; Id be,8 ; Scan name field
; cpir
; jr z,tsec ; Found
;
; xor a ; Set Z-check for bad command
; ret
;
; tsec: id a,(seconds) ; Otherwise toggle
; ; display flag
; cpl
; id (seconds),a
; jr good ; Set good return
;-----

```

```

;
; MCFG - does all required special configuration of the IOP
; module after it has been relocated into high
; memory and before internal IOP initialization is
run.
; Entry: HL,IX=address of IOP buffer, IY=address of local IOP
; Exit: Z set if installation error
; NZ set if ok
; Uses: IX, IY must be preserved.
;
mcfg:   exx                ; Save IOP addr
        id hl,(109h)      ; Get address of environment
        ld de,42h        ; Offset to ZSDOS clock
driver
        ; address
        add hl, de
        call ihhl        ; Get ZSDOS address
        ld de,16h       ; Offset to clock address
        add hl, de
        call ihhl        ; HL=address of ZSDOS clock
        ; routine
        id (zsclock),hl ; Save it locally
;
        exx                ; Restore IOP address
        ld de,52h        ; Offset to IOP config area
        add hl, de
        ex de,hl         ; DE=loaded IOP config
address
        id hl,config     ; HL=local config area
        ld bc,cfklen     ; BC=length of config area
        idir
;
good:   or on            ; Set good (NZ) return
        ret
;
; MINST - does all required special installation of the IOP
; module after it has been relocated into high memory
; and after internal IOP initialization is run.
; Entry: HL, IX=address of IOP buffer, IY=address of local IOP
; Exit: Z set if installation error
; NZ set if ok
;
minst:  ret              ; Nothing special required
;
; Load HL with the two bytes it is pointing to
;
ihhl:   ld a,(hl)        ; Low order to A
        inc hl
        ld h,(hl)        ; High order to H
        ld l,a           ; Low order to L
        ret
;=====
;
; Data area
;
clock:  ds 6             ; Initial clock read buffer
        end
;
; End of IOPCLK.Z80

```

*Computer Corner, from page 48*

picture, text, drawing, and fancy graphics. It sure makes me miss using a Macintosh. Icons in and of themselves are not the answer to easy use. In Mac machines, especially with the release of version 7, the look and feel is so uniform, that once a operation or technique is learned, you can use it in all programs.

Contrast this to PC based windows programs and you can easily see why the Macintosh is still best. When using a PC, I have stopped using the mouse as it is too much work. Most of the programs I have all use the mouse differently—no standardization. Windows 3 added a more constant look and feel, but at present it is still each programming shop to their own way of doing things. I feel another 5 years from now and maybe the PC world will reach the same level of user friendliness that has been available on Macs for more than 5 years.

For Mac users or want-to-be-users you do not need to spend lots of money. No I am not talking about price drops but about other systems. A friend of mine just bought an Amiga with a MAC adapter card. He has been able to run almost everything available with better graphics and also have regular Amiga programs as well. I plan on doing the same here soon with my Atari ST. I think I can turn my 1 Meg machine into a Mac clone for about \$300. Now I am not positive about prices right now, but I think you could buy a new Atari ST for say \$1000 (color monitor). The Mac adapter at \$300 means an Atari/Macintosh for \$1300 total. Now this is not getting you a classic model, but something closer to their 11s. For people who want to step up to 68K machines and real user friendly software it sure beats the "real thing".

**Up Last**

A last topic is industrial controllers. I was planning on reporting on a new controller running Forth, but I haven't

received the information yet. The person is promising a demo model so I feel strongly I should wait and give a more complete accounting of the product later.

What I can report on is the flood of new products and interest in this area. I have commented before how it seems to be the last area for home grown production. Well it seems that a lot of people are taking me at my words. I am sure the number of small units available has never been greater. It is possible to get everything from 8031 to the newest chip on the market in something that fits in your hand.

If you are going this way a few words of advice. First off do your homework. Find someone who needs the product and work with them to get it going. This will help you shake out all those little "oops" that can slip by so easily. Think about your user and just what they need. Don't forget what level of knowledge your user may have. Make it a point of building and testing everything you say the system can do. Be able to prove beyond a shadow of doubt that it will work as advertised. Lastly don't do it if you really don't have the experience and knowledge needed.

I receive several industrial trade magazines and have seen lately talks about the dangers of poorly designed and tested industrial controllers. The project I worked on years ago was held up and changed several times due to concerns about product liability. When designing the system knowing what will happen if this or that fails may be the difference between someone saving their life or losing it. These are not items to be treated lightly or haphazardly.

**Caution To the Winds**

At this point I feel like throwing caution to the winds and saying how next time I will have information for you that you will wonder how you ever lived without it. But then I also got abridge to sell as well....have fun!!!!#

---

---

# Z-Best Software

## Spotlight on Gene Pizzetta

By Bill Tishey

---

---

What a great time this is for anyone involved with Z-System! The "team" of Z developers continues to fill the Z-Nodes with new tools and refinements to existing utilities. As you can see, the list of new and revised programs is even

MS-DOS self-extracting (.EXE) ZIPfiles. I feel that this column will serve its readers best if there are such questions to guide it. Let me hear from you!

### Spotlight on Gene Pizzetta

Gene provides us four new tools and updates to three others. As you read the following paragraphs, one thing should be obvious—Gene's continuing work with program support for ZSDOS and datestamping in general.

#### CPD vs 1.0 (ZSUS Vol 2 #11)

Gene's ComParE-Directory utility, CPD, was an idea derived from CDIR vs 2.0 (by Rob Wilcox and Richard Brewster) and the result of some prompting from Howard Schwartz. See *Figure 1* for CPD's syntax.

CPD compares two directories and indicates which files exist in both directories and which exist only in the first directory. If no option is given, all matching non-system files in the first directory are displayed with files marked which also exist in the second directory. The duplicated files are marked in standout video (if supported by one's TCAP) and by an asterisk following the filename. The asterisk

Listing 1  
New Releases:

Name	Vers	S	ZSUS	Siz	Rec	CRC	Library/Size	Issued	Author	
<a href="#">CFGZ.COM</a>	1.00			00		6 47	F829 CFGZ10	5 05/03/91	Terry Hazen	
SYS User-modified version of AMPRO CONFIG 2.6 utility to allow correct HLP=N operation under both CP/M and NZCOM. CFG=Y										
<a href="#">CMAZE.COM</a>	1.80			0 0		28 220	D2AD CMAZE18	39 09/16/91	Lee Bradley	
SYS MBASIC/BASCOM game which uses Z-System TCAP data. Creates a maze of HLP=N walls and asks you to position a ball in the maze. Data entry is CFG=N best done with ZEX scripts. Uses Z3BAS library of routines.										
<a href="#">CPD.COM</a>	1.00			0 V211		4 29	83D1 CPD10	21 08/10/91	Gene Pizzetta	
DIR Compares two directories and indicates which files exist in both and HLP=Y which exist only in the first. Allows setting of archive bit on CFG=Y those files which exist in both directories.										
<a href="#">FILT.COM</a>	0.80			00		4 30	9588 FILT8	23 09/13/91	Gene Pizzetta	
HP ZCPR3 rework of Irv Hoff's FILT7 tool which sets or expands tabs and HLP=N removes several types of unwanted characters in ASCII text, WordStar CFG=Y documents or assembler source code files.										
<a href="#">JUST.COM</a>	1.20			00		4 31	548B JUST12	40 09/13/91	Gene Pizzetta	
WP ZCPR3 rework of Irv Hoff's tool to justify ASCII and WordStar text HLP=N files. Full command-line operation, DU support, error-flag setting, CFG=Y quiet mode, and transfers create datestamps under ZSDOS.										

longer than that of last issue! This, however, makes my job as this columnist even tougher. Due to the limited space here, it's impossible to cover everything in detail. What I would like to do each issue is provide some encapsulated reports on what is new and improved.

Sometimes, as in this issue, I may focus on the contributions of a particular developer. In addition, however, I would like to highlight some programs of your choosing. Drop me a line (either directly or through the editor) with any programs which you'd like to see discussed. Maybe you want to know more about LSH, Rob Friefeld's command-line editing shell. Maybe you'd like to see some applications for Terry Hazen's ZFIND utility. Maybe you're looking for a good file-compare utility, or something that will unZIP

has the added function of allowing visible markings in printer output.

CPD's options allow 1) display of only those files which exist in both directories, 2) display of files in the first direc-

---

*Bill Tishey has been a ZCPR user since 1985, when he found the right combination of ZCPR2 and Microsoft's Softcard CP/M for his three-year-old Apple 11+. After graduating to ZCPR30 and PCPI's Applicard CP/M, he did a "manual install" of ZCPR3.3 (with help from a lot of friends'), and in late 1988 switched to NZCOM and ZSDOS, all on the same vintage Apple 11+. Bill is the author of the Z3HELP system, a monthly-updated system of help files for Z-System programs, as well as comprehensive listings of available Z-System software. Bill is the editor of the Z-System Software Update Service and has compiled such offerings as the Z3COM package and the Z-System Programmer's Toolkit. Bill is a language analyst for the federal government and frequents the Foreign Language Forum (FLEFO) on CompuServe. He can be reached there (76320,22), on Genie (WATISHE), on Jay Sage's Z-Node #3 (617-965-7259) and by regular mail at 8335 Dubbs Drive, Severn, MD 21144.*

```

LOCNDOPT.HEX 1.00 00 1 3 8FB5 LOCNDOPT 3'08/01/91 Bruce Morgen
PROG4 Patch to LOCNDO.IOM vs 1.2 which adds 1) Z3 extended ENV drive vector
HLP=N support, 2) "/" help query and 3) proper trap if no valid ENV.
CFG=N

MKZ3BASE.COM 1.00 00 2 9 241E MKZ3BASE 5 22/22/91 H. zur Nedden
PROG Builds a Z3BASE.LIB by reading the current values for the equates
HLP=N from the environment descriptor.
CFG=N

SSORTPAT.Z80 1.00 00 2 9 AD42 SSORTPAT 2'08/23/91 Bruce Morgen
PROG4 Patch for MicroPro SuperSort v1.60 (SORT.COM) to allow multiple
HLP=N SuperSort commands under ZCPR33+ (including BGii 1.3x, NZCOM, Z3PLUS)
CFG=N from the system prompt or in Z-System aliases and other scripts.

TYPELZH.COM 2.00 CO 7 54 191C TLZH20 60 07/27/91 Roger Warren
LBR Types LZH-encoded, Crunched, and Squeezed files. Based on TYPELZ.
HLP=N Vs 1.0 (8/15/89) by Roger Warren.
CFG=N

TYPELZHR.COM 2.00 c0 7 54 B980 TLZH20 60 07/27/91 Roger Warren
LBR See TYPELZH.COM. For R/CPMs with wheel at 03EH.
HLP=N
CFG=N

WINDWLIB.REL 1.00 00 6 47 A8D2 WINDOW10 18 04/29/91 W. Schmitt
PROG3 Routines which allow use of up to 15 windows on (any) CP/M computer.
HLP=N
CFG=N

XRUN-ENG.COM 2.50 00 4 25 45BC XRUN25 38 02/28/91 Olaf Krumnow
SYS An ECP for ZCPR33. Looks for files along the search path and
HLP=N constructs a command line depending on the extension of the file
CFG=N found. Supports shell variables.

ZBIB.COM 0.50 00 8 64 529C ZBIB05 23 09/15/91 Joe Mortensen
7DBASE Bibliographic database manager based on ZDB.
HLP=Y
CFG=Y

ZTIME+.COM,1.30 00 4 27 F6B4 ZTIME13 25 09/11/91 Gene Pizzetta
DATE See ZTIME.COM. Z3PLUS version.
HLP=N
CFG=Y

ZTIME.COM 1.30 00 4 27 7E7B ZTIME13 25 09/11/91 Gene Pizzetta
DATE A hardware independent clock utility for setting or displaying the
HLP=N date and time under ZSDOS or Z3PLUS. Options include the ability to
CFG=Y measure elapsed time. ZSDOS version.

```

```

Listing 2
Revised Programs:

Name          Vers S ZSUS Siz Rec CRC Library/Size Issued Author
-----
CFGLIB.REL 1.00 4 V212 2 10 E955 ZCNFG19 109 08/21/91 Al Hawley
PROG3 Library of routines used to link .CFG configuration files used by
HLP=Y ZCNFG.COM.
CFG=N

DATSTP-P.30M 1.70 3 0 5 39 33C5 DATSTP17 64 08/30/91 Gene Pizzetta
DATE See DATSTP-U.COM. For Z3PLUS only. Type 3 at 8000h.
HLP=Y
CFG=Y

DATSTP-P.40M 1.70 4 0 6 46 3A31 DATSTP17 64 08/30/91 Gene Pizzetta
DATE See DATSTP-U.COM. For Z3PLUS only. Type 4.
HLP=Y
CFG=Y

DATSTP-P.COM 1.70 00 5 39 690C DATSTP17 64 08/30/91 Gene Pizzetta
DATE See DATSTP-U.COM. For Z3PLUS only.
HLP=Y
CFG=Y

DATSTP-U.30M 1.70 30 6 44 EE45 DATSTP17 64 08/30/91 Gene Pizzetta
DATE See DATSTP-U.COM. Universal version (ZSDOS, ZDDOS and Z3PLUS).
HLP=Y Type 3 at SOOOh.
CFG=Y

```

tory which do not exist in the second, and 3) setting the archive attribute on those files which exist in both directories, facilitating copying those that do not.

**FILT vs 8.0, JUST vs 1.2**

Gene has reworked several of Irv Hoff's text-handling utilities for ZCPR. FILT8 is an update to FILT7, Irv's widely-used CP/M tool which sets or expands tabs and removes several types of unwanted characters in ASCII text, WordStar documents, or assembler source code files. JUST12 is a reworking of Irv's JUSTIFY tool which justifies ASCII and WordStar text files (see Figure 2 for their syntax). Both tools are now command-line driven, provide DU and DIR support, error flag setting, error handler invocation, quiet mode and ZCNFG configuration. Under ZSDOS and ZDDOS, both will preserve file create date stamps.

As Gene puts it, the hardest part in converting FILT7 and JUSTIFY was "deciding how to make the variously interactively chosen modes into a group of logical command-line options." In FILT8, mode options are provided for use with source code (S), ASCII text (A), WordStar documents "without" dot commands (W), and WordStar documents "with" dot commands (D). In JUST12, options allow for justifying lines starting with a space (S), for justifying lines regardless of their original length (L), and for retaining embedded form feeds. Both tools also have a quiet mode option (Q). FILT8 and JUST12 are a welcome addition to the word-processing/text-handling toolset and, if Gene had not intended it so, are also a fitting reminder to us all of the legacy of Irv Hoff.

**ZTIME vs 1.3**

ZTIME is a new, hardware-independent clock utility for setting or displaying the date and time under ZSDOS or Z3PLUS. See Figure 3 for its syntax.

Date and time entry with ZTIME is simplified by a syntax which allows missing elements to be filled in from the current setting of the clock. Thus, if you want to change the hour, say to daylight saving time, you need only type "ZTIME hh:". To adjust the minutes, type "ZTIME :mm", or to set the seconds, type "ZTIME ::ss". A unique feature of ZTIME is its ability to meas-

**Figure 1**  
 CPD Version 1.0  
 Usage:  
 CPD (dir:){afn} {dir:}{/}options  
 Displays files in first directory, marking files that are duplicated in second directory.  
 Options:  
 B display only files in both directories  
 M display only files missing from second directory  
 A set archive attribute if in both directories  
 S include system files  
 P don't page display  
 L echo to printer  
 F send final form feed  
 Options B and M are mutually exclusive.

**Figure 3**  
 ZTIME Version 1.3  
 Displays or sets ZSDOS date and time.  
 Usage:  
 ZTIME {{mm}/{dd}/{yy}} {{hh}:{mm} :{ss}}  
 sets clock, or  
 ZTIME {/}option  
 Options:  
 C Show date and time continuously  
 S Set date and time interactively  
 M Store date and time in memory registers 18-23  
 E Show elapsed time since using option M  
 If no date and time string or option is given, the current date and time is displayed.

**Figure 2**  
 FILTER Version 8.0 Copyright (c) 1986 by Irv Hoff  
 Usage:  
 FILTER {dir:}infile {dir:}outfile {/}options  
 The outfile defaults to the name of the infile.  
 Mode Options:  
 S Source Code  
 A ASCII Text [default]  
 W WordStar-remove dot commands  
 D WordStar-retain dot commands  
 Other Options:  
 T don't use tabs  
 Q quiet mode on  
 JUSTIFY Version 1.2 Copyright (c) 1988 by Irv Hoff  
 Usage:  
 JUST {dir:}infile {dir:}{outfile} {/}options  
 If no outfile is given, default is filetype "JUS".  
 In-Text Options, in first column of source file:  
 ) center this line  
 ] justify this indented paragraph  
 ; do not justify this line  
 Command Line Options:  
 n line width for justifying (default 65)  
 S justify lines starting with space  
 L justify lines regardless of length  
 F retain form feeds  
 Q quiet mode on

DATSTP-U.40M 1.70 4 0 7 52 F6BA DATSTP17 64 08/30/91 Gene Pizzetta  
 DATE See DATSTP-[U.COM](#). Universal version (ZSDOS, ZDDOS, or Z3PLUS).  
 HLP=Y Type 4.  
 CFG=Y

DATSTP-[U.COM](#) 1.70 00 6 44 A480 DATSTP17 64 08/30/91 Gene Pizzetta  
 DATE Displays or changes the create and modify date stamps on any file  
 HLP=Y from the command line. Universal version (ZSDOS/ZDDOS/ZRDOS with  
 CFG=Y DateStamper, under Z3PLUS will display but not change date stamps)

DATSTP-Z.30M 1.70 30 4 31 3B3E DATSTP17 64 08/30/91 Gene Pizzetta  
 DATE See DATSTP-[U.COM](#). For ZSDOS and ZDDOS only. Type 3 at 8000h.  
 HLP=Y  
 CFG=Y

DATSTP-Z.40M 1.70 40 5 36 970E DATSTP17 64 08/30/91 Gene Pizzetta  
 DATE See DATSTP-[U.COM](#). For ZSDOS and ZDDOS only. Type 4.  
 HLP=Y  
 CFG=Y

DATSTP-[Z.COM](#) 1.70 30 4 31 486D DATSTP17 64 08/30/91 Gene Pizzetta  
 DATE See DATSTP-[U.COM](#). For ZSDOS and ZDDOS only.  
 HLP=Y  
 CFG=Y

[DSTATS.COM](#) 1.20 3 V211 2 16 1A0E DSTATS12 19 08/09/91 Terry Hazen  
 DISK ZCPR3 disk/user statistics utility. Displays disk block size, disk  
 HLP=Y capacity, allocated and free space, list of active user areas, etc.  
 CFG=Y Combines DSKMAP and many functions of UMAP.

EXTEND.40M 1.40 4 V213 2 11 D4ED EXTEND14 10 08/14/91 Bruce Morgen  
 WP See [EXTEND.COM](#). Type 4.  
 HLP=Y  
 CFG=N

[EXTEND.COM](#) 1.40 4 V213 1 8 DEOD EXTEND14 10 08/14/91 Bruce Morgen  
 WP Text file extender for all Z80 machines. Appends input line to new  
 HLP=Y or existing ASCII file. Vs 1.0 (09/81) by Ron Fowler.  
 CFG=N

[HELPC.COM](#) 1.50 0 V213 5 37 5794 HELPC15 24 09/16/91 Howard Goldstein  
 HELP Replacement for the standard Z-System HELP utility. Handles crunched  
 HLP=N as well as normal, uncompressed help files. Print options are  
 CFG=Y disabled when wheel byte is turned off. Configurable with ZCNFG.

ure elapsed time. An "M" option stores the current date and time in memory registers, and an "E" option compares the current time with that stored in the registers to calculate the elapsed time in hours, minutes and seconds (up to 24 hours maximum). ZTIME also has an option to show the time continuously. Two versions are available for ZSDOS and Z3PLUS, with wheel byte support (setting the clock will not work unless the wheel byte is set).

### ZSLIB vs 3.2

As mentioned last time, Gene has released an update (now vs 3.2) of ZSLIB, a set of routines which provides extensive date and time and file stamp support for ZSDOS, Z3PLUS and CP/M-Plus (in addition, it supplements SYSLIB with a number of general-purpose routines). Improvements include: greater compatibility with DSLIB routines, switchable date output in either American (mm/dd/yy) or European (dd.mm.yy) date order, switchable time output in either military (24 hour) or civilian format, and more flexible command-line parsing of date and time specs. Version 3.2 also adds a new set of video highlighting routines which are much smaller (although somewhat slower) than similar modules in VLIB4.

## DATSTP vs 1.7

Gene has also provided some improvements to DATSTP, his utility for displaying or changing (from the command line) ZSDOS and Z3PLUS create and modify date stamps. See *Figure 4* for its syntax.

DATSTP does not just "replace" date and time specs, but "edits" them. An editing buffer is loaded with a file's current create date stamp and the command-line spec is applied to it. Any missing fields in the input spec remain unchanged in the buffer, allowing for very flexible entry. Create and modify date stamps can be edited individually or both given the same stamp. The major changes to DATSTP since last October (vs 1.4) are the use of ZSLIB32's more flexible parsing of date/time specs and the revival of support for DateStamper (DATSTP will now function as long as there is a !!!TIME&.DAT file on disk; DateStamper need not even be running). Version 1.7 adds a C option which allows use of the current system time as the stamp editing source. It moves the current date into the edit buffer as the default date, but does not affect which stamp is modified. You must, for instance, use the C and M options together to change the modify stamp to the current date.

## RCOPY vs 1.2

RCOPY is Gene's tool to automate transfer of a set of files to a RAM disk at boot-up. As a side benefit, it can reduce considerably the size of STARTUP aliases. See *Figure 5* for syntax.

RCOPY can also be used to copy a

<a href="#">HELPLSH.COM</a> 1.10	4 V212	3 21 5611 LSH11 66 08/24/91 Rob Friefeld
HELP Help program for LSH vs 1.1.		
HLP=Y		
CFG=N		
<a href="#">LBREXT.COM</a> 3.30	0 V213	8 64 159F LBREXT33 35 08/02/91 Howard Goldstein
LBR Extracts crunched, squeezed and LZH-encoded files from LBRs.		
HLP=Y Vs 2.0 by Bob Peddicord.		
CFG=Y		
<a href="#">LDIR-B.COM</a> 2.20	0 V211	2 16 A343 LDIRB22 25 09/10/91 Bruce Morgen
LBR Displays LBR directories showing file dates and sizes. Includes a		
HLP=Y summary line of active/free/deleted/total member entries. For CP/M		
CFG=N 2.2 or CP/M+ with runtime ZCPR3 support. Vs 1.0 (87) by S. Greenberg.		
<a href="#">LHC.COM</a> 2.00	0 V213	5 40 29F0 LBRHLP20 45 09/16/91 Howard Goldstein
HELP Remake of HELP 5.3 which reads crunched .HLP files in LBRs.		
HLP=N		
CFG=Y		
<a href="#">LHO.COM</a> 2.00	0 V213	5 37 166D LBRHLP20 45 09/16/91 Howard Goldstein
HELP Remake of HELP 5.3 which reads squeezed .HLP files in LBRs.		
HLP=N		
CFG=Y		
<a href="#">LPUT.COM</a> 2.20	4 V213	6 48 3D8A LPUT22 35 08/26/91 Howard Goldstein
LBR Automated CPM/ZCPR3 library maker. Does for LBR creation what LGET		
HLP=Y does for extraction.		
CFG=N		
<a href="#">LSH.30M</a> 1.10	4 V212	8 59 FFE8 LSH11 66 08/24/91 Rob Friefeld
SHELL Log SHell is a screen-oriented command line editing shell.		
HLP=Y <a href="#">LSHINST.COM</a> installs program defaults and control key bindings.		
CFG=N		
<a href="#">LSH.40M</a> 1.10	4 V212	9 68 9256 LSH11 66 08/24/91 Rob Friefeld
SHELL See LSH.30M. Type 4.		
HLP=Y		
CFG=N		
<a href="#">LSHF.30M</a> 1.10	3 V212	8 59 17A6 LSH11 66 08/24/91 Rob Friefeld
SYS See LSH.30M. "Fixed-file", Type-3 version.		
HLP=Y		
CFG=N		
<a href="#">LSHF.40M</a> 1.10	4 V212	9 68 86B3 LSH11 66 08/24/91 Rob Friefeld
SYS See LSH.30M. "Fixed-file", Type-4 version.		
HLP=Y		
CFG=N		
<a href="#">LSHINST.COM</a> 1.10	4 V212 11 85 6909	LSH11 66 08/24/91 Rob Friefeld
SHELL Install program for LSH vs 1.0r.		
HLP=Y		
CFG=N		

Figure 4

DATSTP Version U-1.7 (loaded at OIOOh)  
Edits and displays file date stamps under ZSDOS, Z3PLUS,  
& DateStamper.

### USAGE:

DATSTP {dir:}filename {date} {time} {/}options

If no date and time or option is given, date stamps  
are displayed only. Edit and write default to the  
create stamp.

### DATE/TIME FORMAT:

{mm}/{dd}/{yy} {hh}:{mm} clock time  
{mm}/{dd}/{Yy} +{nnnn} relative time

### OPTIONS:

C edit the current date and time  
M edit and write the modify stamp  
B write both the create and modify stamps  
Q toggle quiet mode on

Option C overrides option M for edit.

Figure 5

RCOPY Version 1.2  
Copies list of files from AO: to AO:  
Usage:  
RCOPY12 {/}option  
Option:  
Q toggle quiet mode on

# MOVING?

## Don't leave us behind!

Send Change of Address six weeks prior to move.

list of other types of files from one directory to another (e.g., WordStar overlay files before a word-processing session). The source and destination directories and list of files is internally configured (with ZCNFG). The file list can contain up to 20 files (a limitation of ZCNFG; an unlimited number can be coded internally). Improvements since the last version (9/90) include the preservation of date stamps under ZSDOS/ZDDOS and the ability to set archive, system and/or no-stamp attributes on the destination files.

### Program of the Month

#### ZCNFG vs 1.9 (ZSUS Vol 2 #12)

Remember when you had to edit the data options and reassemble the source code to configure a program such as FileFind to your particular system and style of usage? If you were lucky, someone provided an overlay patch so you didn't have to touch the source code itself. Well, much of this is behind us now. Al Hawley's universal configuration utility, ZCNFG, greatly simplifies the configuration of *any* program. Yes, given that conventions are followed in programming of its overlay (.CFG) file, any Z-System or CP/M program can now benefit from its simple, menu-approach to configuration.

ZCNFG's operation is simple. See *Figure 6* for ZCNFG's syntax and a sample configuration screen.

The data options and current selection for each option are displayed to the user in one or more menus. These may be in the form of "toggles" for yes/no options, multiple-choice options, or various edit options (the hex or decimal value of a byte, a DU: or Z3-style file spec, a printer init string, etc.) which the user changes to suit his needs. Help screens are usually available to explain each configuration option. When all options are "set", exiting with X or ESC overlays the appropriate data to the configuration block in the first page of the target program. That simple!

Al has made some major improvements to ZCNFG since vs 1.6 (7/90). its search logic, for instance, has been changed to speed up response for common types of usage. Now, if either a partial or complete DU is given, the alternate directory is *not* searched. If the CFG name is taken from the target file's configuration block, the alternate directory *is* searched first. A colon or

<a href="#">LT.COM</a>	3.00	CO 7 53 55D6	LT30 56 07/17/91	C. B. Falconer					
LBR Library Type can type normal, LZH-encoded, crunched or squeezed files									
HLP=N whether standalone or in a .LBR. Can extract/uncrunch any/all files									
CFG=N at the same time. Adapted from Steven Holtzclaw's LUXTYP (06/83).									
<a href="#">NUKEYCLK.COM</a>	0.30		0 0	2 16 7FCC	NKYCLK03	6 07/13/91	Joe Mortensen		
IOP Enables IOP Nukey to have current date/time available as macros									
HLP=N within other applications such as WordStar. For ZCPR3+ systems									
CFG=N with built-in clock.									
<a href="#">RCOPY.COM</a>	1.20		0 0	3 20 72FD	RCOPY12	20 08/17/91	Gene Pizzetta		
FILE Copies a list of files between two directories.									
HLP=Y									
CFG=Y									
<a href="#">STATPAT.Z80</a>	3.00		0 0	4 30 0B9A	STATPAT3	3 08/26/88	Bruce Morgen		
PROG4 Patches <a href="#">STAT.COM</a> to work under ZCPR3.									
HLP=Y									
CFG=N									
<a href="#">TPA.COM</a>	3.3a	CO 15	EDBB	TPA33A	12 08/19/91	Jay Sage			
SYS Reports amount of memory available in TPA (Transient Program Area)									
HLP=N and allows one to temporarily lower the size of TPA. Vs. 3.3 adds									
CFG=N ZCPR "/" help screen, display in kilobytes.									
<a href="#">ZCNFG.COM</a>	1.90		4 V212	6 48 E021	ZCNFG19	109 08/21/91	Al Hawley		
SYS Universal configuration utility which configures option data in									
HLP=Y executable files. Uses .CFG overlay file.									
CFG=Y									
<a href="#">ZCRCK.30M</a>	1.40		3 V211	3 24 806E	ZCRCK14	16 08/08/91	Bruce Morgen		
FILE See <a href="#">ZCRCK.COM</a> . Type 3 at 8000h.									
HLP=N									
CFG=N									
<a href="#">ZCRCK.40M</a>	1.40		4 V211	4 29 B3D6	ZCRCK14	16 08/08/91	Bruce Morgen		
FILE See <a href="#">ZCRCK.COM</a> . Type 4.									
HLP=N									
CFG=N									
<a href="#">ZCRCK.COM</a>	1.40		3 V211	3 24 2EEA	ZCRCK14	16 08/08/91	Bruce Morgen		
FILE ZCPR3-compatible version of Sigi Kluger's NZCRCK1, which was									
HLP=Y designed to combine the features of CRCK and CHEK on RCP/M systems.									
CFG=N Original (05/86) by Bruce Morgen.									
<a href="#">ZDB.COM</a>	1.50	00		8 64 F4D7	ZDB15	23 09/09/91	Joe Mortensen		
DBASE Small, fast name and address manager with built-in label and									
HLP=Y envelope addressing features. NZTCAP and VLIB4D support.									
CFG=N									
<a href="#">ZDT.COM</a>	1.00		0 0	8 64 3034	ZDT10	43 08/05/91	Joe Mortensen		
DBASE Z-System Day.Timer, a daily planning calendar derived from ZDB.									
HLP=N Automatically reads the real-time clock and displays the current									
CFG=N day's schedule. Requires ZCPR3.0+ and extended TCAP.									
<a href="#">ZERR.30M</a>	1.60		3 V212	3 23 DFDA	ZERR16	49 08/24/91	Rob Friefeld		
ERROR Bare-bones error handler derived from EASE vs 1.62. Configured									
HLP=Y with <a href="#">ZERRINST.COM</a> .									
CFG=N									
<a href="#">ZERR.40M</a>	1.60		4 V212	4 31 A98A	ZERR16	49 08/24/91	Rob Friefeld		
ERROR See ZERR.30M. Type 4.									
HLP=N									
CFG=N									
<a href="#">ZERRINST.COM</a>	1.60		3 V212	4 31 65D1	ZERR16	49 08/24/91	Rob Friefeld		
ERROR Install program for <a href="#">ZERR.COM</a> vs 1.6.									
HLP=N									
CFG=N									
<a href="#">ZERRLSH.COM</a>	1.10		3 V212	4 29 B69B	LSH11	66 08/24/91	Rob Friefeld		
ERROR ZCPR 3.3+ error handler which performs all the functions of ZERR.									
HLP=Y Automatically updates history file, if LSH history shell is active.									
CFG=N Configured with <a href="#">ZERRINST.COM</a> .									
<a href="#">ZERRLSHF.COM</a>	1.10		3 V212	4 29 6719	LSH11	66 08/24/91	Rob Friefeld		
ERROR See <a href="#">ZERRLSH.COM</a> . "Fixed-file" version.									
HLP=Y									
CFG=N									

<a href="#">ZFIND.COM</a>	1.40	00	4	30	B54E ZFIND14 35 08/11/91	Terry Hazen
FILE ZCPR3 string search utility which very quickly finds ASCII strings in HLP=Y text files. Found string can be displayed in either line or delimited CFG=Y block. Output can be written or appended to a file.						
<a href="#">ZLT.COM</a>	1.50	00	6	47	E427 ZLT15	38 09/12/91 Howard Goldstein
LBR Z-System Library Typer, , basically a Z'ified LT29 with i file HLP=Y extraction and parsing code removed. Va 1.1 (09/88) by Bruce CFG=Y Morgen.						
ZP.30M	1.50	30	8	64	81DA ZP15	86 08/23/91 Terry Hazen
FILE See <a href="#">ZP.COM</a> . Type 3 at 8000h. HLP=Y CFG=Y						
ZP.40M	1.50	40	10	74	FAA2 ZP15	86 08/23/91 Terry Hazen
FILE See <a href="#">ZP.COM</a> . Type 4. HLP=Y CFG=Y						
<a href="#">ZP.COM</a>	1.50	00	8	64	A257 ZP15	86 08/23/91 Terry Hazen
FILE ZCPR33+/Z3PLUS/BGii screen-oriented file/disk/memory record patcher HLP=Y using the ZPATCH command set. Requires a VLIB4D+ Z3TCAP. One-record CFG=Y cache can be exchanged with file/disk/memory records.						
ZSLIBM.REL	3.20	40	27	212	E201 ZSLIB32 94 09/08/91	Gene Pizzetta
PROG1 Assembly language routines to assist programmers in handling date- HLP=Y stamp maintenance under ZSDOS, Z3PLUS, and CP/M Plus. Microsoft CFG=N REL format.						
ZSLIBS.REL	3.20	40	25	197	412F ZSLIB32 94 09/08/91	Gene Pizzetta
PROG1 See ZSLIBM.REL. SLR format. KLP=Y CFG=Y						
<a href="#">ZSWEEP.COM</a>	1.30	00	16	122	C80C ZS13 38 09/11/91	Pete Pardoe
LBR Z'ified version of NSWP207 with ability to act upon current file with HLP=N any Z-System command up to 127 chars. NDR support, datestamp display. CFG=N Requires ZCPR33+ with extended TCAP.						
<a href="#">ZSWEEP.COM</a>	1.20	00	19	152	073A ZS12 49 08/03/91	Pete Pardoe
LBR See <a href="#">ZSWEEP.COM</a> . Generic version (regular TCAP). HLP=N CFG=N						

Figure 6  
ZCNFG, Z-SYSTEM CONFIGURATION UTILITY  
Version 1.9, 08/20/91

Configures option data in Executable files.

**Syntax:**  
ZCNFG [du/dir: ]<nautil>[.<ex1>] [du/dir: ]<nam2>[.<ex2>]

du/dir: defaults to the current drive and user  
<nam1> is the Executable file to configure.  
<nam2> is the configuration overlay file.  
<ex1> defaults to COM, <ex2> defaults to CFG

Example: ZCNFG ZCNFG ;configures itself.

A related configuration data file must be present to provide  
Screen layout, Menus, and configuration data.

```
BO:WORK>zcnfg zcnfg

ZCNFG, Z-SYSTEM CONFIGURATION UTILITY
Version 1.9, 08/20/91

ZCNFG CONFIGURATION

T) Target Program Default Filetype          COM
O) Overlay file Default Filetype            CFG
Z) Z3ENV auto-install for ZCPR3              YES
A) Alternate D/U for Overlay files           D15
L) console Lines per Screen                  24
C) Configuration LBR name                    CONFIG.LBR
D) Use TARGET DU as the default for the CFG filespec :
```

ZCNFG INSTALLATION CONTROL

X or Esc =Save changes & eXit Q, °C =Quit with no changes saved  
/ or ? =Explain Options > or . =Next Menu < or , =Previous Menu  
Which choice?

form-feed character has also been added as the first character of a line in help screens to invoke paging. But that isn't all. Back in May, I asked AI if ZCNFG couldn't be made to recognize its configuration (CFG) files from inside an LBR in a specified directory. There were so many of these files lying around that they were getting hard to keep track of and, for me, were taking up a lot of precious hard-disk space. Well, with version 1.9, AI has added this very feature, giving you the option now of allowing them to reside unLBR'ed in an alternate directory or setting up an LBR of CFGs, with a name and location of your choice. CONFIG.LBR (the default name of the LBR) is expected to be found in ZCNFG's alternate directory and is now searched first for the CFG file if there is no DIR form (D:, (J:, DU: or DIR:) specified in the invoking command tail.

As Z-Librarian, I try to keep an up-to-date package of the latest Z-System COM files, along with their associated HLP and CFG files. Being able to maintain the CFG files in one, single library saves considerable time in finding needed files and disk space in storing them. In September I uploaded to the Z-Nodes CFG01.LBR, a full set of CFG files for the latest existing executables for use with ZCNFG19. It contained 66 files (there are now well over 70) and took up only 149k, instead of 312k if the files were to reside separately on a hard disk with 4k-per-file allocation.

What can I say? ZCNFG has improved to the point where it is now one of the most "essential" of Z-System tools. I think we can safely say that "Z-CoNFiGuration" is *the* "standard" for setting up configuration options for users in Z-System programs. It should become a standard for CP/M. Many thanks, AI!®

The ultimate test  
of Man's conscience  
may be in his  
willingness to sacrifice  
something for  
future generations  
whose words of thanks  
will not be heard.  
—Gaylord Nelson

Reader, from page 30

journal has become my favorite computer magazine.

Please don't forget that some readers tire still in the learning mode and are not on the same knowledge plane as the editor or the authors. I hope that I can feel free to ask some dumb questions from time to time and that the journal can be a learning tool while still being a method of exchanging ideas among peers.

B.R., Cabot PA

May I suggest the entire computer elite suffers from a common malady. It is *not* elegant to employ cryptic abbreviations and "well-known" mnemonics. The proper way to expose is to assume that *no* reader is an insider but that *every* reader truly desires to understand.

You should print no article without a complete glossary explaining any possibly obscure term. Take a paper you consider worth publishing. Place it in the hands of someone who is decidedly not knowledgeable. Extract from the subject an evaluation unencumbered by any form of intimidation or put-down by the resident experts. If it ain't right, fix it.

I am quite isolated. Have had this Atrix for several years and can operate it as an appliance, make simple repairs and do a bit of non-elegant BASIC programming. I would very much like to upgrade. The machine has an IEEE-48 port, a 50-pin D-connector receptacle on the motherboard and the ability to address external double sided 5 1/4 and 8-inch drives. I would like to add a hard disk or 8-inch floppy. So far, I have not found any source of information to help. Ya wanna help?

E.B. Swisshome OR

*True, some readers are not at the level of the authors. In fact, no one is at the level of all the authors. TCJ was never meant to be light reading. This is a technical journal and authors should not oversimplify advanced topics as we frequently cover. Still, I hope that our authors note your thoughts and take the extra time to explain things. We are all intelligent people here, but we come from different computer disciplines and need terms defined.*

*Please do feel free to ask questions. Most authors give their address in their biographical paragraph. Otherwise, ask me and I will forward the question on.*

*We are promised an article on building a generic SCSI port daughterboard. This would allow you to add a hard drive to your Atrix. —Ed.*

Thank you for providing a publication for the serious hobbyist!

R.S. Millers MD

I really appreciate your practical hardware and software approach and your Forth coverage.

I would like see a project of building a very small Z80-based portable, even if it requires a separate monitor and keyboard, using the super-integrated chips with CTC, DMA, SIO and PIO. My idea would be to mount aboard on the side of a slim 3.5" floppy drive. Ideally, it would have native Forth in ROM that could boot CP/M.

I am an electronics technician who switched to software six and a half years ago. I have been making my living programming in Forth ever since. I would like to work on such a project with someone.

E.J. Portland OR

*This man reads my mind! These were my exact thoughts the*

*first time I saw a YASBEC. Anyone for a good hack?—Ed.*

I love it! Could you suggest good back issues for someone just starting out in embedded systems?

G.S., Richmond Height OH

*Take a look at issues 40 onward. I found Tim McDonough's articles in 45, 46 and 47 particularly good and continue to hope he will submit again. Matt Mercaldo's pieces starting in issue 44 are excellent.—Ed.*

I am so enthusiastic to have found a publication that fits my interests that I plan to send a letter detailing just what kind of person your newest subscriber is. Is there a method of transmitting this letter electronically? This is a "must read" publication!

J.B., Brookshire TX

*Letters and articles can be submitted over the Internet to [cmcewen@gnat.rent.com](mailto:cmcewen@gnat.rent.com), on GENie at TCJ\$ or on my bulletin board at (90S) 754-9067. The Internet address is courtesy of Andy Meyer. The GENie address goes to all the TCJ editors, so don't give away state secrets! Don't be shy about submitting an idea for an article, either.—Ed.*

I believe a simple mistake has been made in relation to my subscription. The renewal notice states that my subscription expires with issue 54. It should expire six issues later as I subscribed for two years.

P.C., Balcatta WA Australia

*Oops! My mistake. This happens now and then. Did you notice that the address label shows your expiration? Let me know if it isn't right. All I need is a note to look into it.—Ed.*

One small note of editorial concern: In issue 52, Richard Rodman's column gets to the bottom of page 49, then launches out into hyperspace, never to appear again in the magazine.

B. M., Gainesville FL

*Your electronic mail was my first clue of this mistake. It was the first of a flood, and I learned my lesson: Never mess with Richard's column!—Ed.*

I wanted you to know that I am impressed with *TCJ's* content! It is by far the most bang-for-the-buck I've seen in ten years in the magazine industry.

P.T., West Lafayette IN

*Thanks! Glad you're enjoying it.*

I own a CP/M computer named "Alphatronic PC" here in Germany and "Royal Alphatronic" in the UK. It was never intended to add a hard disk to this machine, which I would like to do. I read that there exists a kind of SCSI card for systems like mine that one needs only to remove the Z80 chip, then plug in that board and then put in the Z80 chip in that board. Do you know where in the States I can order such a board, preferably with an appropriate hard disk and the software? I have the BIOS for my system, so software is not the problem. Were there some articles in former *TCJ* issues describing how to add a hard disk to systems like mine?

U. N., Bonn, Germany

*Check Wayne Sung's article in issue 40 on adding a Bernoulli drive. He tells how to adapt the parallel port to a generic SCSI. Also, George Warner has been saying he will write an article about this.—Ed.*

## CP/M SOFTWARE

100 page Public Domain Catalog, \$8.50 plus \$1.50 shipping and handling. New Digital Research CP/M 2.2 manual, \$19.95 plus \$3.00 shipping and handling. Also, MS/PC-DOS Software. Disk Copying, including AMSTRAD. Send self addressed, stamped envelope for free Flyer, Catalog \$1.00

Elliam Associates  
Box 2664  
Atascadero, CA 93423  
805-466-8440

## Advent Kaypro Upgrades

**TurboROM.** Allows flexible configuration of your entire system, read/write additional formats and more. \$35

**Hard drive conversion kit** Includes interface, controller, TurboROM, software and manual—Everything needed to install a hard drive except the cable and drive! \$175 without clock, \$200 with clock.

**Personality Decoder Board.** Run more than two drives, use quad density drives when used with TurboROM. \$25

*Limited Stock — Subject to prior sale*

Call 916-483-0312 eves/weekends or write Chuck Stafford, 4000 Norris Avenue, Sacramento CA 95821

## TGJ The Computer Journal Market Place Advertising for Small Business

Looking for a way to get your message across?  
Advertise in the Market Place!

First Insertion:	\$50
Reinsertions:	\$35

Rates include typesetting. Payment must accompany order. Foreign orders paid in US funds drawn on a US bank or international money order. Resetting of ad constitutes a new advertisement at first insertion rate. Camera ready copy from laser printers, photo typesetters, etc., are acceptable. Dot matrix, daisy wheel, typewriter output not accepted. Inquire for rates for larger ads if required. Deadline is eight weeks prior to publication date. Mail to:

The Computer Journal  
Market Place  
PO Box 12

S. Plainfield NJ 07080-0012 USA

## Z-System Software Update Service

Provides Z-System public domain software by mail.

Regular Subscription Service  
Z3COM Package of over 1.5 MB of COM files  
Z3HELP Package with over 1.3 MB of online documentation  
Z-SUS Programmers Pack, 8 disks full  
Z-SUS Word Processing Toolkit  
And More!

For catalog on disk, send \$2.00 (\$4.00 outside North America)

and your computer format to:  
**Sage Microsystems East**  
1435 Centre Street

*Editor, from page 26*

many letters as we have room for. The authors also welcome your comments.

What is missing may be more important than what is here! Time just ran out and I had to bump some very promising articles to the next issue, Bridger Mitchell returns to discuss I/O *Redirection*. David Goodenough will be here to discuss *Interrupts and the Z80*, while Brian Moore will tell us how to install *ZCPR on a 16-bit Intel Platform*. George Warner tells us how to double the clock speed on an Ampro. We have a good article on cyclical redundancy checking in Forth, and I expect more on the system software for the Y ASBEC. Stay tuned!#

I am one of the REH CPU280 users and love it. It's fast and the operating system Tilman built with the automatic changing of formats is better than Uniform. It's great for a member of a club who often has to change formats for send software disks to members.

We have a OMTI 55xx card working with the CPU280 and I use a 68MB RLL Toshiba HD. Tilman built a AT-BUS adapter card and Uwe Herzceg made the BIOS for the AT-BUS hard disk. It will work together with my CPU280 and I hope I have the time doing the same with my GENIE III.

I am collecting old CP/M computers and have a Genie I, III, a Kaypro II, an Epson Px-8, a REH CPU280, a Morrow MD3, a Sharp 3541, a Tatung TPC-2000. The C128 is for the only PROM burner I have.

If someone has a Genie III or His or a Speedmaster I can send him the opcode of the BIOS (OS) CP/M 2.2 and 3.

Maybe that these nice computers found the way to the USA.

I want to get all the old TCJs. What would it cost and can I get them?

F.C., Saarstr. Germany

*Your computer room has more inventory than most stores over here! To the best of my knowledge, some of your machines never made it to these shores. You must have your work cut out for you in supporting your users with disks. I know what I go through for ZSUS!*

*Your group is likely the most active in the world, and you surely have some high powered help. I look forward to hearing more about what everyone is up to.*

*Take a look on pages 46 and 47 for a listing of the available back issues. Some are sold out, others nearly so. Still, there are enough issues still available to make it through a cold long winter.—Ed.9*

# The Computer Journal

## Back Issues

Sales limited to supplies in stock.

**Special Close Out Sale on these back issues only.**

3 or more \$1.50 each postpaid in the US. ~~OR \$3.00 postpaid airmail outside US.~~

### Issue Number 22:

• RS 232 Interface Part 1  
• Telecomputing with the Apple  
• 1 Keypers' Comm Gedding Maned.  
• Build an A/D Converter for Apple II

### Issue Number 23:

• FileTrasales Programs for CP/M  
• RSM2encePan2  
• BuildHarwePrktSpoderPen1  
• Review of Ropy Disk Formats  
• Sending Moras Gode with an Apple II  
• Beginner's Column: Basic Concepts and Formulas

### Issue Number 24:

• Add an 8087 Math Chip to Your Dux Processor Board  
• Build an A/D Converter for Apple II  
• Modems i Maus  
• The CRM Operating System  
• Build-Hardware Print Spooler Part 2

### Issue Number 25:

• Optronics, Part 1: Detecting, Generating and Using Light in Electronics  
• Multi-User At Introduction  
• Making the CPW User Function More Useful  
• Build-Hardware Print Spooler, Part 1  
• Beginner's Column: Power Supply Design

### Issue Number 26:

• SUM VIC-20 EPROM Programmer  
• MuUserCPANer  
• Build High Resolution S-100 Graphics Board, Part 3  
• Systems Integration, EadsCPMM30  
• Mrwr Optimization with Mictos

### Issue Number 18:

• Parallel Interface for Apple II Game Port  
• The Hacker's MAC: A Letter from Lee Felsenstein  
• S-100 Graphics Screen Dump  
• The LS-100 Disk Simulator Kit  
• BASE: Part Six  
• Interfacing Tips & Troubles: Communicating with Telephone Tone Control, Part 1

### Issue Number 19:

• Using the Versatility of Fort  
• The C Column: A Graphics Primitive Package  
• The Hitachi HD64180: New Life for 8-bit Systems  
• ZSIG Corner: Command Line Generators and Aliases  
• A Tutor Program in Forth: Writing a Forth Tutor in Forth  
• Disk Parameters: Modifying the CP/M Disk Parameter Block for Foreign Disk Formats

### Issue Number 20:

• Designing an 8035 SBC  
• Using Apple Graphics from CP/M: Turbo Pascal Controls Apple Graphics  
• Soldering & Other Strange Tales  
• Build an S-100 Floppy Disk Controller: WD2797 Controller for CP/M 68K

### Issue Number 21:

• Extending Turbo Pascal: Customize with Procedures & Functions  
• Unsoldering: The Arcane Art  
• Analog Data Acquisition & Control: Connecting Your Computer to the Real World  
• Programming the 8035 SBC

### Issue Number 22:

• NEW-DOS: Write Your Own Operating System  
• Variability in the BDS C Standard Library  
• The SCSI Interface: Introductory Column  
• Using Turbo Pascal ISAM Files  
• The Ampro Little Board Column

### Issue Number 23:

• C Column: Flow Control & Program Structure  
• The Z Column: Getting Started with Directories & User Areas  
• The SCSI Interface: Introduction to SCSI  
• NEW-DOS: The Console Command Processor  
• Editing the CP/M Operating System  
• INDEXER: Turbo Pascal Program to Create an Index  
• The Ampro Little Board Column

### Issue Number 24:

\* Selecting & Building a System  
• The SCSI Interface: SCSI Command Protocol  
• Introduction to Assemble Code for CP/M  
• The C Column: Software Text Filters  
• Ampro 186 Column: Installing MS-DOS Software  
• The Z-Column  
• NEW-DOS: The CCP Internal Commands  
• ZTime-1: A Real Time Clock for the Ampro Z-80 Little Board

### Issue Number 25:

• Repairing & Modifying Printed Circuits  
• Z-Com vs. Hacker Version of Z-System  
• Exploring Single Linked Lists in C  
• Adding Serial Port to Ampro LB  
• Building a SCSI Adapter  
• NEW-DOS: CCP Internal Commands  
• Ampro 186 Networking with SuperDUO  
• ZSIG Column

### Issue Number 26:

• Bus Systems: Selecting a System Bus  
• Using the SB 180 Real Time Clock  
• The SCSI Interface: Software for the SCSI Adapter  
• Inside Ampro Computers  
• NEW-DOS: The CCP Commands (continued)  
• ZSIG Corner  
• Affordable C Compilers  
• Concurrent Multitasking: A Review of DoubleDOS

### Issue Number 27:

• 68000 TinyGiant: Hawthorne's Low Cost 16-bit SBC and Operating System  
• The Art of Source Code Generation: Disassembling Z-80 Software  
• Feedback Control System Analysis: Using Root Locus Analysis & Feedback Loop Compensation  
• The C Column: A Graphics Primitive Package  
• The Hitachi HD64180: New Life for 8-bit Systems  
• ZSIG Corner: Command Line Generators and Aliases  
• A Tutor Program in Forth: Writing a Forth Tutor in Forth  
• Disk Parameters: Modifying the CP/M Disk Parameter Block for Foreign Disk Formats

### Issue Number 28:

• Starting Your Own BBS  
• Build an A/D Converter for the Ampro Little Board  
• HD64180: Setting the Wait States & RAM Refresh using PRT & DMA  
• Using SCSI for Real Time Control  
• Open Letter to STD Bus Manufacturers  
• Patching Turbo Pascal  
• Choosing a Language for Machine Control

### Issue Number 29:

• Better Software Filter Design  
• MDISK: Adding a 1 Meg RAM Disk to Ampro Little Board, Part 1  
• Using the Hitachi hd64180: Embedded Processor Design  
• 68000: Why use a new OS and the 68000?  
• Detecting the 8087 Math Chip  
• Floppy Disk Track Structure  
• The ZCPR3 Corner

### Issue Number 30:

• Double Density Floppy Controller  
• ZCPR3 IOP for the Ampro Little Board  
• 3200 Hackers' Language  
• MDISK: Adding a 1 Meg RAM Disk to Ampro Little Board, Part 2  
• Non-Preemptive Multitasking  
• Software Timers for the 68000  
• Lilliput Z-Node  
• The ZCPR3 Corner  
• The CP/M Corner

### Issue Number 31:

• Using SCSI for Generalized I/O  
• Communicating with Floppy Disks: Disk Parameters & their variations  
• XBIOS: A Replacement BIOS for the SB180  
• K-OS ONE and the SAGE: Demystifying Operating Systems  
• Remote: Designing a Remote System Program  
• The ZCPR3 Corner: ARUNZ Documentation

### Issue Number 32:

• Language Development: Automatic Generation of Parsers for Interactive Systems  
• Designing Operating Systems: A ROM based OS for the Z81  
• Advanced CP/M: Boosting Performance  
• Systematic Elimination of MS-DOS Files: Part 1, Deleting Root Directories & an In-Depth Look at the FCB  
• WordStar 4.0 on Generic MS-DOS Systems: Patching for ASCII Terminal Based Systems  
• K-OS ONE and the SAGE: System Layout and Hardware Configuration  
• The ZCPR3 Corner: NZCOM and ZCPR34

### Issue Number 33:

• Data File Conversion: Writing a Filter to Convert Foreign File Formats  
• Advanced CP/M: ZCPR3PLUS & How to Write Self Relocating Code  
• DataBase: The First in a Series on Data Bases and Information Processing  
• SCSI for the S-100 Bus: Another Example of SCSI's Versatility  
• A Mouse on any Hardware: Implementing the Mouse on a Z80 System  
• Systematic Elimination of MS-DOS Files: Part 2, Subdirectories & Extended DOS Services  
• ZCPR3 Corner: ARUNZ Shells & Patching WordStar 4.0

### Issue Number 34:

• Developing a File Encryption System.  
• Database: A continuation of the data base primer series.  
• A Simple Multitasking Executive: Designing an embedded controller multitasking executive.  
• ZCPR3: Relocatable code, PRL files, ZCPR34, and Type 4 programs.  
• New Microcontrollers Have Smarts: Chips with BASIC or Forth in ROM are easy to program.  
• Advanced CP/M: Operating system extensions to BDOS and BIOS, RSXs for CP/M 2.2.  
• Macintosh Data File Conversion in Turbo Pascal.  
• The Computer Corner

### Issue Number 35:

• All This & Modula-2: A Pascal-like alternative with scope and parameter passing.  
• A Short Course in Source Code Generation: Disassembling 8088 software to produce modifiable assem. source code.  
• Real Computing: The NS32032.  
• S-100: EPROM Burner project for S-100 hardware hackers.  
• Advanced CP/M: An up-to-date DOS, plus details on file structure and formats.  
• REL-Style Assembly Language for CP/M and Z-System.. Part 1: Selecting your assembler, linker and debugger.  
• The Computer Corner

### Issue Number 36:

• Information Engineering: Introduction.  
• Modula-2: A list of reference books.  
• Temperature Measurement & Control: Agricultural computer application.  
• ZCPR3 Corner: Z-Nodes, Z-Pian, Amstrand computer, and ZFILE.  
• Real Computing: NS32032 hardware for experimenter, CPUs in series, software options.  
• SPRINT: A review.  
• REL-Style Assembly Language for CP/M & ZSystems, part 2.  
• Advanced CP/M: Environmental programming.  
• The Computer Corner.

### Issue Number 37:

• C Pointers, Arrays & Structures Made Easier: Part 1, Pointers.  
• ZCPR3 Corner: Z-Nodes, patching for NZCOM, ZFILER.  
• Information Engineering: Basic Concepts: fields, field definition, client worksheets.  
• Shells: Using ZCPR3 named shell variables to store date variables.  
• Resident Programs: A detailed look at TSRs & how they can lead to chaos.  
• Advanced CP/M: Raw and cooked console I/O.  
• Real Computing: The NS 32000.  
• ZSDOS: Anatomy of an Operating System: Part 1.  
• The Computer Corner.

### Issue Number 38:

• C Math: Handling Dollars and Cents With C.  
• Advanced CP/M: Batch Processing and a New ZEX.  
• C Pointers, Arrays & Structures Made Easier: Part 2, Arrays.  
• The Z-System Corner: Shells and ZEX, new Z-Node Central, system security under Z-System s.  
• Information Engineering: The portable Information Age.  
• Computer Aided Publishing: Introduction to publishing and Desk Top Publishing.  
• Shells: ZEX and hard disk backups.  
• Real Computing: The National Semiconductor NS320XX.  
• ZSDOS: Anatomy of an Operating System, Part 2.

### Issue Number 39:

• Programming for Performance: Assembly Language techniques.  
• Computer Aided Publishing: The Hewlett Packard LaserJet.  
• The Z-System Corner: System enhancements with NZCOM.  
• Generating LaserJet Fonts: A review of Digi-Fonts.  
• Advanced CP/M: Making old programs Z-System aware.  
• C Pointers, Arrays & Structures Made Easier: Part 3: Structures.  
• Shells: Using ARUNZ alias with ZCAL.  
• Real Computing: The National Semiconductor NS320XX.  
• The Computer Corner.

# The Computer Journal

## Back Issues

Sales limited to supplies in stock.

### Issue Number 40:

- Programming the LaserJet: Using the escape codes.
- \* Beginning Forth Column: Introduction.
- Advanced Forth Column: Variant Records and Modules.
- LINKPRL: Generating the bit maps for PRL files from a RELfile.
- WordTech's dBOXL: Writing your own custom designed business program.
- Advanced CP/M: ZEX 5.0: The machine and the language.
- Programming for Performance: Assembly language techniques.
- Programming Input/Output With C: Keyboard and screen functions.
- The Z-System Corner: Remote access systems and BDS C.
- Real Computing: The NS320XX
- The Computer Corner.

### Issue Number 41:

- Forth Column: ADTs, Object Oriented Concepts.
- Improving the Ampro LB: Overcoming the 88Mb hard drive limit.
- How to add Data Structures in Forth
- Advanced CP/M: CP/M is hacker's haven, and Z-System/Command Scheduler.
- The Z-System Corner: Extended Multiple Command Line, and aliases.
- Programming disk and printer functions with C.
- LINKPRL: Making RSXes easy.
- SCOPY: Copying a series of unrelated files.
- The Computer Corner.

### Issue Number 42:

- Dynamic Memory Allocation: Allocating memory at runtime with examples in Forth.
- Using BYE with NZCOM.
- C and the MS-DOS Screen Character Attributes.
- Forth Column: Lists and object oriented Forth.
- The Z-System Corner: Genie, BDS Z and Z-System Fundamentals.
- 68705 Embedded Controller Application: An example of a single-chip microcontroller application.
- Advanced CP/M: PluPerfect Writer and using BDS C with REL files.
- Real Computing: The NS 32000.
- The Computer Corner

### Issue Number 43:

- Standardize Your Floppy Disk Drives.
- A New History Shell for ZSystem.
- Heath's HDOS, Then and Now.
- The ZSystem Corner: Software update service, and customizing NZCOM.
- Graphics Programming With C: Graphics routines for the IBM PC, and the Turbo C graphics library.
- Lazy Evaluation: End the evaluation as soon as the result is known.
- S-100: There's still life in the old bus.
- Advanced CP/M: Passing parameters, and complex error recovery.
- \* Real Computing: The NS32000.
- The Computer Corner.

### Issue Number 44:

- Animation with Turbo C Part 1: The Basic Tools.
- Multitasking in Forth: New Micros F68FC11 and Max Forth.
- Mysteries of PC Floppy Disks Revealed: FM, MFM, and the twisted cable.
- DosDisk: MS-DOS disk format emulator for CP/M.
- Advanced CP/M: ZMATE and using lookup and dispatch for passing parameters.
- Real Computing: The NS32000.
- Forth Column: Handling Strings.
- Z-System Corner: MEX and telecommunications.
- The Computer Corner

### Issue Number 45:

- Embedded Systems for the Tenderfoot: Getting started with the 8031.
- The Z-System Corner: Using scripts with MEX.
- The Z-System and Turbo Pascal: Patching [TURBO.COM](#) to access the Z-System.
- \* Embedded Applications: Designing a Z80 RS-232 communications gateway, part 1.
- Advanced CP/M: String searches and tuning Jetfind.
- Animation with Turbo C: Part 2, screen interactions.
- Real Computing: The NS32000.
- The Computer Corner.

### Issue Number 46:

- Build a Long Distance Printer Driver.
- Using the 8031's built-in UART for serial communications.
- Foundational Modules in Module 2.
- The Z-System Corner: Patching The Word Plus spell checker, and the ZMATE macro text editor.
- Animation with Turbo C: Text in the graphics mode.
- \* Z80 Communications Gateway: Prototyping, Counter/Timers, and using the Z80 CTC.

### Issue Number 47:

- Controlling Stepper Motors with the 68HC11F
- Z-System Corner: ZMATE Macro Language - Using 8031 Interrupts
- T-1: What it is & Why You Need to Know
- ZCPR3 & Modula, Too
- Tips on Using LCDs: Interfacing to the 68HC705
- Real Computing: Debugging, NS32 Multitasking & Distributed Systems
- Long Distance Printer Driver: correction
- ROBO-SOG 90
- The Computer Corner

### Issue Number 48:

- Fast Math/Using Logarithms
- Forth and Forth Assembler
- Modula-2 and the TCAP
- Adding a Bernoulli Drive to a CP/M Computer (Building a SCSI Interface)
- Review of BDS Z
- PMATE/ZMATE Macros, Pt. 1
- Real Computing
- Z-System Corner: Patching MEX-Plus and TheWord, Using ZEX
- Z-Best Software!
- The Computer Corner

### Issue Number 49:

- Computer Network Power Protection
- Floppy Disk Alignment w/RTXEB, Pt. 1
- Motor Control with the F68HC11
- Controlling Home Heating & Lighting, Pt. 1 Getting Started in Assembly Language
- LAN Basics
- PMATE/ZMATE Macros, Pt 2
- \* Real Computing
- Z-System Corner
- Z-Best Software
- The Computer Corner

### Issue Number 50:

- Offload a System CPU with the Z181
- Floppy Disk Alignment w/RTXEB, Pt. 2
- Motor Control with the F68HC11
- Modula-2 and the Command Line
- Controlling Home Heating & Lighting, Pt. 2
- Getting Started in Assembly Language Pt 2
- Local Area Networks
- Using the ZCPR3IOP
- PMATE/ZMATE Macros, Pt 3
- Z-System Corner, PCED
- Z-Best Software
- Real Computing, 32FX16, Caches
- The Computer Corner

### Issue Number 51:

- Introducing the YASBEC
- Floppy Disk Alignment w/RTXEB, Pt 3
- High Speed Modems on Eight Bit Systems
- A Z8 Talker and Host
- Local Area Networks—Ethernet
- UNIX Connectivity on the Cheap
- \* PC Hard Disk Partition Table
- A Short Introduction to Forth
- Stepped Inference as a Technique for Intelligent Real-Time Embedded Control
- Real Computing, the 32CG160, Swordfish, DOS Command Processor
- PMATE/ZMATE Macros
- Z-System Corner, The Trenton Festival
- Z-Best Software, the Z3HELP System
- The Computer Corner

### Issue Number 52:

- YASBEC, The Hardware
- An Arbitrary Waveform Generator, Pt. 1
- B.Y.O. Assembler... in Forth
- Getting Started in Assembly Language, Pt. 3
- TheNZCOM IOP
- Servos and the F68HC11
- Z-System Corner, Programming for Compatibility
- Z-Best Software
- Real Computing, X10 Revisited
- PMATE/ZMATE Macros
- Controlling Home Heating & Lighting, Pt. 3
- The CPU280, A High Performance Single Board Computer
- The Computer Corner

	U.S.	Foreign Foreign Total	
		(Surface) (Airmail)	
<b>Subscriptions</b>			
1 year (6 issues)	\$18.00	\$24.00 \$38.00	_____
2 years (12 issues)	\$32.00	\$44.00 \$72.00	_____
<b>Back issues:</b>			
18 thru #43	\$3.50 ea.	\$5.00 ea.	_____ My Interests: _____
6 or mon	\$3.00 @a.	\$4.50 ea.	_____
#44 and up	\$4.50 ea.	\$6.00 ea.	_____
S or mon	\$4.00 ea.	\$5.50 ea.	_____
<b>Back Issues Ordered:</b>			
	Subscription Total	_____	
	Back Issues Total	_____	
	Total Enclosed	_____	

Name: \_\_\_\_\_

Address: \_\_\_\_\_

\_\_\_\_\_

\_\_\_\_\_

Payment is accepted by check or money order. Checks must be in US funds, drawn on a US bank. Personal checks within the US are welcome.

**The Computer Journal**  
P.O. Box 12, S. Plainfield, NJ 07080-0012  
Phone (908) 755-6186

---

# The Computer Corner

By Bill Kibler

---

It was a busy summer and not just with outside activities. The computer industry seems to have been on double shift these days. The number of new items seems to belie any signs of there being a recession. So lets talk about two new items to play with.

## DOS 5.00

By the time you read this, I doubt any of you will not have tried or installed the new DOS 5.00. I have mine and have been doing some playing with it. First off let me say it appears to work correctly and as advertised. For Microsoft that is a new step up. Not only are they usually very late with their product releases but they have a tendency to fall somewhat short of expectations.

The expectations of DOS 5.00 however have never been too great, so if it does what you want, then it must be OK. In reviewing some of the programs I can see that for the most part the new features are those public domain utilities which everyone used to make up for DOS's shortcomings. An example is DOSKEY which stores keystrokes for later recalling and editing. I know lots of people who have been using similar programs like that for years.

So what does this new version give you to make it worth all the hoopla, relocatable drivers. Beside working without bugs (at least so far) the ability to put DOS and their drivers into extended memory is probably the most important change. We use LAN based programs and the LAN drivers had our program memory space down to 340K. Several programs have trouble working in that space, but with DOS 5.00 I have been able to get 478K of program space. It is still not what I really need and any 68K based computer would not have these problems, but for DOS it is a great step forward.

It took some time of playing around with the extended memory drivers before I got the extra memory. It may be possible to get more but it took half a day to get that. The book is fair for support. They have tried to explain things in term for all levels of users. Unfortunately the drivers section is a complex issue and so much variability exists in user systems that I am not sure it can be explained properly anywhere. Microsoft's attempt is good but falls short. I think it needs a more cook book approach with some complete and typical examples. As it is, the instructions are broken into many small sections and the overall relationships between the parts is lost.

After a few hours of trying different options I believe I understand the relationships between the parts. It appears that the drivers can go only into what I call the shadow memory behind the ROMS and video ram. That means maximum of 340K of RAM less system and other special block or in my case 140K of upper memory. And yes there are all

kinds of new terms bantered around. UMB got used everywhere and is defined as Upper Memory Block, but how it is really used is left out.

I guess my only complaint is the absence of a real explanation for how it works internally, and thus how you can beat or overcome their limits if your case doesn't fit one of their normal options. The overall report however is good and I reserve the right to change my mind about it as time goes on.

## Time Moves On

Well I have spent some time working with MINIX and reached some stumbling blocks. It is not so much actual problems as it is philosophical ones. I have gotten lazy in my latter years and a two year old boy eats up what little time remains. When starting on MINIX I discovered just how many learning steps were involved. The question is do I want to know about all these new items.

Those items range from editors, to assemblers, make files to linking scripts. The number of steps to put together a working system is mountainous. What it has boiled down to is do I want to become a UNIX-like guru or do I want a simple operating system that can be ported easily between platforms.

Currently I am still mulling over the dilemma and would tend to lean toward the small C operating system. I used a commercial industrial controller based on it and found it adequate. The whole code fits on a 360K floppy without zipping. Compare that to MINIX and the multiple sections of code, many disks of compressed files, complex assembling and linking, and dozens of new utilities. The small C operating system uses C based utilities to give it a DOS like feel.

DOS has made me lazy over the years and I have found their utilities rather user friendly, simple, and straight forward. MINIX utilities at present for me are complex, cryptic, and hard to use. I know if I spend many hours playing with MINIX I will probably find the tools they have more powerful and in time easier to use. For now however I am not sure I have the time or desire to learn them. I am finding this fact even with DOS 5.00 that many of their new utilities are better and more helpful and yet I am not sure I will learn their full use if any use at all. I often remark on how CP/M did everything I needed for many years, so why do I need more complexity in my life now?

## Make It Simple

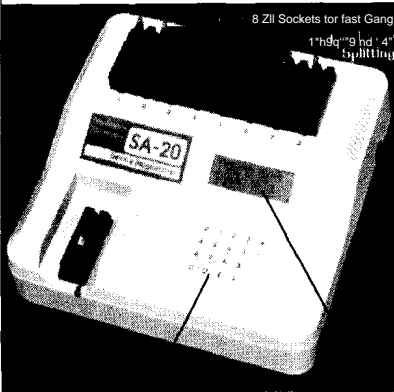
I took a refresher course the other day on Aldus Page-Maker 4.0 and had a wonderful time. We put together a simple one page newsletter in three hours that included a

*See Computer Corner, page 37*

# EPROM PROGRAMMERS

Stand-Alone Gang Programmer

**\$750.00**



8 ZIF Sockets for fast Gang

- Completely stand-alone or PC driven
- Programs E(EP)ROMs
- 1M0b-m010N/M
- UooupiroMtoSIMollitit
- .3./MFI0000t.NS-232
- Pointol in anti Out
- 32K internal Flash EEPROM for easy firmware upgrades
- Quick Polio Alporllim (27111 loim, 1 Magabit in 17 ate.)
- 2 year warranty
- Made in U.S.A.
- Technical support by phone
- Complete manual and schematic
- Sloplo SocM Proprommit oho available. \$330.00
- Split and Shuffle 10 & 32 bit
- 100 User Definable Macros, 10 User Definable Configurations
- Intelligent Identifier
- Binary, Intel Hex. and Motorola 8

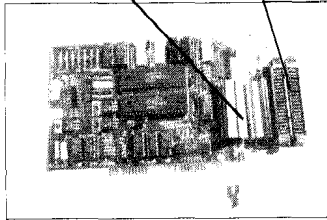
20 Key Tactile Keypad (not membrane) 20 x 4 Line LCD Display

Internal Programmer for I'

**\$139.95**

NewintelligentAveraging Algorithm. Programs 64Ain10 sec.,258in 1 min., 1 Meg (27010,011)in2 min.45 sec., 2 Msg (27C2001) in 5 min. Internal card with external 40 pin ZIF 21 cable

- Reads, verifies, and programs 2716, 32,32A, 64, 64A,128,128A,286, 812,513,010,011,301, 27C2001, MOM 68764, 2532
- Autmitullit ult progtmmiini witiqi
- Load and save buffer to disk
- Binary, Intel Hex, and Motorola S formats
- Upamti-bit to 32 Mtg EPHOM\*
- No poruillitly modullit ripulnd
- 1 year warranty • 10 day money back guarantee
- Adapters available for 8748,49, 61, 751,62, 55 TMS 7742,27210,8701024, and memory cards
- Made in U.S.A.



**NEEDHAM'S ELECTRONICS**

4530 Orings Grove Ave. • Sacramento, CA 95841  
Mon, Fri. Sam • Spin PST

Call for more Information  
(916) 924-8037  
FAX (916) 972-9980

## Cross-Assemblers as low as \$50.00 Simulators as low as \$100.00 Cross-Disassemblers as low as \$100.00 Developer Packages as low as \$200.00 (a \$50.00 Savings)

### A New Project

Our line of macro Cross-assemblers are easy to use and full featured, including conditional assembly and unlimited include files

### Get It To Market-FAST

Don't wait until the hardware is finished to debug your software, Our Simulators can test your program logic before the hardware is built.

### No Source!

A minor glitch has shown up in the firmware, and you can't find the original source program, Our line of disassemblers can help you re-create the original assembly language source.

### Set To Go

Buy our developer package and the next time your boss says "Get to work," you'll be ready for anything.

### Quality Solutions

PseudoCorp has been providing quality solutions for microprocessor problems since 1985.

### BROAD RANGE OF SUPPORT

Currently we support the following microprocessor families (with more in development):

Intel 8048	RCA 1802,05	Intel 8051	Intel 8096
Motorola 6800	Motorola 6801	Motorola 88HC11	Motorola 6805
Hitachi 6301	Motorola 6809	MOS Tech 6502	WDC85C02
Rockwell 65C02	Intel 8080,85	Zilog 280	NSC 800
Hitachi HD64180	Motorola 68000,8	Motorola 68010	Intel 80C198

All products require an IBM PC or compatible

**So What Are You Waiting For? Call us:**

**PseudoCorp**

Professional Development Products Group  
716 Thimble Shoals Blvd, Suite H  
Newport News, VA 23606

**(804) 873-1947 FAX: (804)873-2154**



**William P Woodall • Software Specialist**

Custom Software Solutions for Industry:

Industrial Controls  
Operating Systems  
Image Processing

Hardware Interfacing  
Proprietary Languages  
Component Lists

Custom Software Solutions for Business:

Order Entry  
Warehouse Automation  
Inventory Control  
Wide Area Networks

Point-of-Sale  
Accounting Systems  
Local Area Networks  
Telecommunications

Publishing Services:

Desktop Systems  
Books  
CET

Format Conversions  
Directories  
Interactive Video

33 North Doughty Ave, Somerville, NJ 08876 • (908) 526-5980

# SAGE MICROSYSTEMS EAST

## Selling & Supporting the Best in 8-Bit Software

- Automatic, Dynamic, Universal Z-Systems: Z3PLUS for CP/M-Plus computers, NZCOM for CP/M-2.2 computers (\$70 each)
- XBIOS: the banked-BIOS Z-System for SB180 computers at a new, lower price (\$50)
- PCED — the closest thing to Z-System ARUNZ, and LSH under MS-DOS (\$50)
- DSD: Dynamic Screen Debugger, the fabulous full-screen debugger and simulator, at an incredible new price, down from \$130 (\$50)
- ZSUS: Z-System Software Update Service, public-domain software distribution service (write for a flyer with full information)
- Plu\*Perfect Systems
  - Backgrounder ii: CP/M-2.2 multitasker (\$75)
  - ZSDOS/ZDDOS: date-stamping DOS (\$75, \$60 for ZRDOS owners, \$10 for Programmer's Manual)
  - DosDisk: MS-DOS disk-format emulator, supports subdirectories and date stamps (\$30 standard, \$35 XBIOS BSX, \$45 kit)
  - JetFind: super fast, extremely flexible regular-expression text file scanner (\$50)
- ZMATE: macro text editor and customizable wordprocessor (\$50)
- BDS C — including special Z-System version (\$90)
- Turbo Pascal — with new loose-leaf manual (\$60)
- ZMAC — Al Hawley's Z-System macro assembler with linker and librarian (\$50 with documentation on disk, \$70 with printed manual)
- SLR Systems (The Ultimate Assembly Language Tools)
  - Z80 assemblers using Zilog (Z80ASM), Hitachi (SLR180), or Intel (SLRMAC) mnemonics, and general-purpose linker SLRNK
  - TPA-based (\$50 *each* tool) or virtual-memory (\$160 *each* tool)
- NightOwl (advanced telecommunications, CP/M and MS-DOS versions)
  - MEX-Plus: automated modem operation with scripts (\$60)
  - MEX-Pack: remote operation, terminal emulation (\$100)

Next-day shipping of most products with modem download and support available. Order by phone, mail, or modem. Shipping and handling \$3 per order (USA). Check, VISA, or MasterCard. Specify exact disk format.

### Sage Microsystems East

1435 Centre St., Newton Centre, MA 02159-2469

Voice: 617-965-3552 (9:00am- 11:30pm)

Modem: 617-965-7259 (pw-DDT) (MABOS on PC-Pursuit)