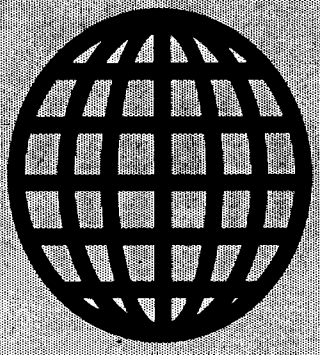
Providing Support Around The World



The Computer Journal

Issue Number 70 November/December 1994 US$4.00

Small System Support

PC/XT Corner: Stepper Motors

Z-System Corner II

The European Beat

Real Computing

Support Groups

Dr. S-100

Multiprocessing for the Impoverished Part 5

Centerfold - Jupiter ACE

The Computer Corner

ISSN # 0748-9331

TCJ - For Having Fun With Any Computer!

The Computer Journal

Founder
Art Carlson

Editor/Publisher
Bill D. Kibler

Technical Consultant
Chris McEwen

Contributing Editors
Herb Johnson
Charles Stafford
Brad Rodriguez
Ronald W. Anderson
Helmut Jungkunz
Ron Mitchell
Dave Baldwin
Frank Sergeant
JW Weaver
Richard Rodman
Jay Sage
Tilmann Reh

The Computer Journal is published six times a year and mailed from *The Computer Journal*, P. O. Box 535, Lincoln, CA 95648, (916) 645-1670.

Subscription rates within the US: $24 one year (6 issues), $44 two years (12 issues). Send subscription, renewals, address changes, or advertising inquires to: *The Computer Journal*, P.O. Box 535, Lincoln,CA 95648.

TCJ The Computer Journal
Issue Number 70 November/December 1994

Editor's Comments 2

Reader to Reader 3
Special Review of Gary Kildall's work.

Z-System Corner II 13
Intro to CP/M and Z-System, part 3.
By Ron Mitchell.

Dr. S-100 17
Pre-Winter Mail Bag.
By Herb R. Johnson.

European Beat 21
Two articles of interest.
By Helmut Jungkunz.

Center Fold 25
Jupiter ACE.

PC/XT Corner 29
Stepper Motors and Forth.
By Guest: V. Henry Vinerts.

Real Computing 32
CD-ROMs and news.
By Rick Rodman.

Small System Support 34
6809 assembly language tutorial.
By Ronald W. Anderson.

Multiprocessing for the Impoverished 38
Part 5: Serial I/O.
By Brad Rodriguez.

Support Groups for the Classics 46
Support groups directory.

The Computer Corner 50
By Bill Kibler.

EDITOR'S COMMENTS

Welcome to issue number 70 and the end of 1994. We finish this year with a collection of letters and projects.

Our Reader to Reader starts out with a look at some of Gary Kildall's early work. All this is from the machine of Emmanuel Roche. He retyped this information so we could understand our loss after Gary's death. Since most is copyrighted, I have extracted what I hope gives you a desire to research his work in greater detail. Having read the entire material, I can see too that he was not only a pivotal person in the industry, but very honest and concerned more about the user than the bank account. We will miss him.

For those wanting to do more with Palmtech's IDE interface, a letter about their single chip controller used on the Z180 single board computer is also in Reader to Reader. And yes it is not all specials, I answer some burning questions from you as well.

Ron Mitchell continues his walk through CP/M and ZCPR. This time we continue the discussion of the basic structure of CP/M and comment on past and present support resources for Z-System.

Dr. S-100, or Herb Johnson has plenty of letters to respond to as he recounts his collecting methods. He ends his correspondence with a short line on PDP-11? Now don't seem so surprised, I have heard from several collectors who have the old machines up and running in their garages. How's that for enjoying the past!

We jump the ocean to hear from Helmut Jungkunz, who can be contacted on GEnie CP/M roundtable at 4PM EST first and third Sundays. I edited together a couple of his pieces of work, which both give you a better idea of the European Beat. Now Helmut has just had a baby and our congratulations or commiserations go out to him, but since I know his time will be shorter to spend on us, might one of you other Europeans like to help him out and contribute to *TCJ*? I know Forth is popular in England and Russia, but what is happening? Got any information you might like to share. Drop me or Helmut some mail (postal or e-mail), we sure want to hear from you.

Taking up the center of the magazine this month is the Jupiter Ace. I wanted to follow the ZX81 with this, but trying to find other information, and my job too, kept getting in the way. So ready or not, check this simple Z80 based system out. I also take this time to start explaining a little fundamental TTL design to help those non-logic users along.

Stepping along to the guest in the house, we hear from V. Henry Vinerts about his stepper motor project. Henry took Frank Sergeant's stepper motor circuit and made it work. So in Franks' place is Henry and his Forth based Etch-A-Sketch.

For those using CD-ROMs and 32 bit machines, Rick Rodman has a few short words. From 32 bits to 8, we jump next to Ronald Anderson and more on his 6809 assembler teachings. Now Ron is a pretty good teacher, so pay attention here if you want to do assembly. Assembly yes, 6809 no, you say? I must remind you that once you fully understand assembly on any chip, moving to others is very easy. So here is that chance to get a good start working with real computer programming.

The last feature this time is Part 5 of the "Scroungemaster II." Brad Rodriguez fills us in on some of the last stages of this project and explains the serial I/O used inside his system. It seems we should be hearing from users soon as Brad has made and sold most of his first run of boards.

Since I am the editor and been writing for *TCJ* forever, I guess my Computer Corner column is rather anticlimactic. That is the problem with running things, I am the last word. Those final words in fact are on two items that came in the mail bag. Sometimes real goodies come in small packages.

Next Time

I have lots of articles sitting in my to do pile. What's in 71? Well more letters, J. G. Owens and his 8048 emulator, Walter J. Rottenkolber and one of his many great articles waiting to see the top of the pile. Space permitting I may slip in a word on power supplies, an XT interface or two, and 8051 development packages. Of course the regulars like Mr. Kaypro who clarifies the different versions of his magic boxes. Ron Anderson continues on with 6809 assembler, but this time taking pseudo program steps and converting them into assembly code.

With that I want to wish everyone a happy and prosperous holiday season. Bill Kibler.

Gary A. Kildall
1942-1994

READER to READER

Letters to the Editor

All Readers

MINI Articles

I have received many letters from Emmanuel Roche, but this last one certainly shows how much he considers Gary Kildall as one of our industries most important persons. Emmanuel's researching of CP/M history and some of it's internal secrets has certainly enlighten many of TCJ's readers. I received these hand typed manuscripts after 69 went to press.

Now I still think that David McGlone's (Z-Letter #32) review of Gary's life and achievements is a must read by everyone, but Emmanuel would like us to remember Gary by his work. In order for you to better understand that work, he has given to me all the articles and discussions by Gary about CP/M. Thus we can get to better understand our loss by looking at Mr. Kildall's achievements.

Unfortunately many of the articles and writings are copyrighted and thus prevent me from running them in their entirety. I am allowed however to "review" them and quote sections in order to give you their flavor as it were. I need to comment that these articles amount to over 80K worth of text (30 pages+), that has been retyped for you and me.

*What follows then, is a review of Gary Kildall's work, in order that we may better understand our loss. (I have put ***** to indicate that I skipped over some text.) Bill Kibler.*

The History of CP/M (The following items were retyped by Emmanuel ROCHE.)

"First word on a floppy-disc operating system" Command language and facilities similar to DECSYSTEM-10 by Jim C. WARREN, Jr., Editor, DDJ, April 1976, p.5

We have the first tidbits of information on the floppy-disc operating system to which we have alluded in past issues:

The system, called "CP/M", runs on an 8080. It is available from Digital Research, P.O. Box 579, PACIFIC GROVE, CA 93950; (408) 373-3403. Its user interface is patterned after that of the DECSYSTEM-10. The file command include RENAME, TYPE, ERASE, DIRECTORY, LOAD, and auto-load/execute facility (type the name of an object file; it will be loaded and begin execution). File-names follow the DEC standard of a 1-8 character name with a 1-3 character suffix. An editor is included that has somewhat the flavor of TECO. There is a PIP facility that allows easy transfer of files to and from any available device, e.g., terminal, paper-tape I/O, cassettes, floppy discs on any drive, etc... Note: PIP is DECeze for Peripheral Interchange Program. Other systems software is likely to be included. *****

This system already exists and has been in use for over a year. It was originally designed and implemented by Dr. Gary KILDALL, a Computer Science Professor at the Naval Postgraduate School in Monterey, and a well-known, independent consultant in the area of microprocessor systems software. Gary is also the designer and implementor of PL/M, the "industry standard" high-level language for microprocessors. PL/M was produced for Intel for their 8000.*****

From our experience, this is the hottest deal going! It's cheap, as far as floppy-disc systems for micros go. The software is well-designed; based on a well-known and easy-to-use operating system that has been around for a DECade. Additionally — major points worth considering — it has been in use for some time, it has been used by a number of people, and it is fairly completely debugged. We know Gary personally, know "where his head's at", and know that he backs his products and is responsive to his customers. Incidentally, he has an excellent and ongoing working relationship with Digital Research.

"The time for floppy's is just about now!" by the Editor, DDJ, Aug.76, p.5.

Things — most notably, prices — are coming down, fast, in the world of rotating mass storage appropriate for home computers. Prior to this, floppy disk subsystems have either been unavailable for hobby machines, or they have been priced for the industrial consumer (e.g., around $3,000 for a dual-drive system). Things have changed: A hobbyist can now reasonably expect to obtain a complete, assembled, single-drive subsystem for a price in the neighborhood of $1K. *****

The best system we know of — and the least expensive — is available from Digital Systems (ask for Dr. John TORODE). This is the same crowd that built Gary KILDALL's original floppy interface over two years ago (see "First Word on a Floppy-Disk Operating System" in the April 1976 issue of the Journal). Gary has yet to have problems with the system. Digital Systems also is marketing a low-cost, floppy-based development system to the industrial market. They know what they are doing. What is much more important is that Dr. KILDALL's fancy, DECsystem-10-like operating system — called CP/M — will run on DS's hardware. CP/M has been in use for OVER TWO YEARS in a production and instructional environment. It is well debugged, well documented, and has some significant software subsystems available with it.

"Disks and CPM". "Disks and the Floppy - A brief overview of how they work" by Richard BATEMAN, in "INMC80", Issue:5, Oct-Dec 1981, p.37

We are all familiar with the cassette recorder attached to our NASCOM, it provides us with low cost mass storage, and permanent storage for programs and data. The method of recording is governed by the type of signals the cassette can record. *****

CP/M and the Floppy. Much has been written about CP/M in this newsletter of late, but no-one has explained what it is. CP/M stands for Control Program for Microcomputers. It was originally written to take the

chore of handling DOS away, and make the disk system as transparent as possible. This end is achieved quite well, although CP/M isn't the friendliest piece of software around. Anyway, CP/M is an operating system much as NAS-SYS is, but revolves around disks, particularly floppy ones. CP/M is divided into 3 parts, CBIOS, CCP and FDOS. The CBIOS does all the I/O and is written for each system, as disk I/O and keyboards differ. Now NAS-SYS and CBIOS are much the same, and indeed could be the same if the entry points were looked at closely.
*****

CP/M reads the Floppy. CP/M reads the disk by using a directory, that relates the logical name (thats what you call the file) and its physical position on the disk. CP/M does not use sector and tracks as in D-DOS but uses block numbers, each block may be, say, 8 sectors, and may therefore be 1024 bytes. The directory entry has 16 bytes set aside to keep the block numbers for that file. If the block size is 1K, then each file can have up to 16K bytes per directory entry, this is known as an extent. Files longer than 16K require two or more consecutive directory entries, the first telling CP/M that it extends to the next extent, etc.
*****

"Upgraded CP/M floppy disk operating system now available" DDJ, Dec 76, p.51

CP/M is a disk operating system designed for diskette-based computer systems which use the Intel 8080 micro-processor. The CP/M software package is now being offered to the small computer user community.
*****

The CP/M operating system is distributed for an Intel MDS micro-computer development system, but can be easily altered to operate with a wide variety of customized hardware environments. Basic requirements are:
a) Intel 8080-based micro-computer mainframe
b) At least 16K of read/write main memory
c) One or two IBM-compatible disk drives and controller

Given these facilities, the CP/M disk system is "patched" by the user to communicate with the specialized hardware. The exact steps to follow in programming and patching the CP/M system are given in the manual "CP/M System Alteration Guide". In fact, several popular mainframe and controller manufacturers currently support their own CP/M patch.
*****

"The Evolution of an Industry: One Person's Viewpoint" (Cf. "Dr. DOBB's Journal", No. 41, JAN 1980, Vol.5, Issue 1, p.6), by Gary A. KILDALL, Digital Research, P.O. Box 579, PACIFIC GROVE CA 93950

1973...

I was sitting quietly at my desk when Masatoshi Shima hurried into my office at Intel and asked me to follow him to his laboratory down the hall. In the middle of his work bench, among the typical snaggle of jumpers, oscilloscopes and multi-meters, sat a binocular microscope with spider-leg probes, all of which were subjecting a minute piece of silicon to helpless investigation. I peered through the microscope at the enlarged regular patterns with particular interest. As a consultant, my job was to design and develop certain software tools for Intel. One was INTERP/80, a program which simulated Intel's newly evolved 8080 micro-processor to be used by Intel customers on time-sharing systems. As I searched for something recognizable, I hoped my simulation resembled the operation of Shima's first 8080 chip which had finally come to life.

My proposal to Intel had been simple: I would provide them with a language, called PL/M, to replace serious systems programming in assembly language. The compiler would first be written in FORTRAN for operation on time-sharing computers and "cross-compiled" to the eight-bit processors. Next, we would write a PL/M compiler in PL/M and "boot-strap" from the time-sharing computer to a resident compiler operating on Intel's new Intellec-8 development system. The first part was complete. PL/M cross compilers and INTERP simulators were implemented for the now-best-forgotten 8008, as well as the 8080. Programs had been written and tested by Intel's software group, consisting of myself and two other people, and we were ready for the real machine. Things were going well: the resident compiler would be the next step.
*****

Meanwhile, John TORODE redesigned and refined our original controller and produced his first complete computer system, marketed under his company name, Digital Systems (which later became Digital Microsystems). The first commercial licensing of CP/M took place in 1975 with contracts between Digital Systems and Omron of America for use in their intelligent terminal, and with Lawrence Livermore Laboratories where CP/M was used to monitor programs in the Octopus network. Little was paid to CP/M for about a year. In my spare time, I worked to improve overall facilities, and added an editor, assembler, and debugger which were predecessors of the current ED, ASM, and DDT programs. By this time, CP/M had been adapted for four different controllers.

In 1976, Glenn EWING approached me with a problem: IMSAI, Incorporated, for whom Glenn consulted, had shipped a large number of disk sub-systems with a promise that an operating system would follow. I was somewhat reluctant to adapt CP/M to yet another controller, and thus the notion of a separated Basic I/O System (BIOS) evolved. In principle, the hardware dependent portions of CP/M were concentrated in the BIOS, thus allowing Glenn, or anyone else, to adapt CP/M to the IMSAI equipment. IMSAI was subsequently licensed to distribute CP/M version 1.3 which eventually evolved into an operating system called IMDOS.

By coincidence, Jim WARREN and I were both consulting at Signetics Corporation during this time. Jim was then the editor of DDJ, and pushed for sale of CP/M to the general public. There was, at the time, a pervading paranoia among software vendors who felt that any and all loose software would be immediately "ripped-off" by this immoral group of computer junkies. Jim's faith in the industry, however, led me to introduce the CP/M 1.3 system for sale on the open market at $70 per copy. In the months that followed, the nature of the computer hobbyist became apparent. In most case he was, like myself, in the computer industry and merely wanted a personal computer for his own endeavors. CP/M gradually gained popularity through a "grassroots" effect and, to the amazement of the skeptics, the rip-off factor was practically nil. A new company called Digital Research was formed to support CP/M, develop new products, and provide administrative functions.
*****

"CP/M: A Family of 8 and 16-Bit Operating Systems" (Cf. BYTE, JUNE 1981, p.216), by Dr Gary KILDALL, Digital Research, P.O. Box 579, 801 LIGHTHOUSE AVENUE, PACIFIC GROVE CA 93950

This article is about microprocessors and CP/M: where they came from, what they are, and what they're going to be. Where they came from is history, what they are today is fact, and what they will become is, like any projection of technology, pure "science fiction" speculation. CP/M is an operating system developed for microcomputers.

But as microprocessors changed, CP/M and its related programming tools evolved into a family of portable operating systems, languages, and applications packages.

The value of computer resources has changed dramatically with the introduction of microprocessors. Three major events have precipitated a revolution in computing: hand-threaded core memory has been replaced by mass-produced semiconductor memory; microprocessors have become plentiful; and IBM decided that the punched card is obsolete. Low-cost memory and processors have reduced the cost of computer systems to a few hundred dollars, but IBM's specification of the floppy disk standard has made the small computer system useful.

In the early days of the 8080 microprocessor, a small company called Shugart Associates was taking shape up the street from Intel. Shugart Associates, along with a number of other companies, viewed the floppy disk as more than a punched card replacement: at that time the primary low-cost storage medium was paper tape (used in applications ranging from program development to word processing). At a cost of $5, a floppy disk held as much data as two hundred feet of paper tape, and a disk drive retailed for only $500 — an unbeatable combination. Memory, processor, and floppy-disk technology improved, and by the mid-1970's, a floppy-based computer could be purchased for about one quarter of a programmer's annual salary. Quite simply, it was no longer to share computer resources.
*****

The emergence of software as a problem-solving tool

Microprocessors are a natural consequence of our technology. I recently visited the British Science Museum, where two particularly interesting historical developments were on display. The first exhibit chronicled the development of the finely matched iron and brass steam engines, complete with magnificent gauges, gears, whistles, and valves, that founded the Industrial Revolution.
*****
I followed the sequence of displays, from BABBAGE's difference and analytic engines to great brass calculators and early punch cards, past relay and vacuum tube processors to unit record equipment, then to transistor and random-logic computers and semiconductors and, finally, to a single Intel 8080 microprocessor.
*****

Application languages

Application languages form the top level of support for application programming. How does this level of language differ from other language levels? First and foremost, an application language contains the operations and data types suitable for expressing programs in a particular problem environment. FORTRAN (FORmula TRANslation), for example, was designed in the late 1950s for scientific applications; FORTRAN programs, therefore, consist primarily of algebraic expressions operating upon binary floating-point numbers expressed in scientific notation.
*****
The evolution of PL/I (Programming Language One) provides a good example of refinement in application languages. PL/I is not a new invention: rather, it was defined by a committee of IBM users in 1960 as a combination of ALGOL (ALGOrithmic Language), FORTRAN, and COBOL, with a liberal sprinkling of new facilities. ALGOL's principal contribution was block structure and nested constructs, while FORTRAN contributed scientific processing and COBOL added commercial facilities. This combination produced a large, unwieldy language with twists and nuances that can trap the unwary programmer. Nevertheless, PL/I was quite comprehensive, and it served as the basis for uncounted numbers of application programs on large systems. One noted use of PL/I was in the implementation of the Multics operating system at MIT under Project MAC.
*****

System languages

A system language is a high-level machine-oriented programming language used to implement so-called "system software", including operating systems, text editors, debuggers, interpreters, and compilers. In the early days of computing, virtually all system software was implemented in assembly language. One revolutionary machine, the Burroughs B5500, used a variant of ALGOL-60 as its only system-programming tool and appeared in the early 1960s. The machine was a commercial success against the other major mainframes, proving that assemblers were no longer necessary. Many successful system languages followed Burroughs' ALGOL, including the C language, produced at Bell Laboratories in the late 1960s, which served as the basis for the UNIX operating system.
*****

PL/M: the base for CP/M

In 1972, MAA (Microcomputer Applications Associates), the predecessor of Digital Research, consulted with the small, aspiring microprocessor division of a semiconductor memory company called Intel Corporation. MAA defined and implemented a new systems-programming language, called PL/M (Programming Language for Microcomputers), to replace assembly-language programming for Intel's 8-bit microprocessor. PL/M is a refinement of the XPL compiler-writing language which is, in turn, a language with elements from Burroughs Corporation's ALGOL and the full set of PL/I.
*****

Operating systems

Operating systems, too, have become more refined. But why do we have operating systems at all? In the 1960s we used expensive mainframes with power-hungry central processors and magnetic-core memory. Downtime for complicated card readers, printers, and backup data-storage devices was high, requiring constant maintenance. A card-oriented "batch" operating system provided two functions.
*****

The CP/M family

CP/M was, however, completed by MAA in 1974. It included a single-user file system designed to eliminate data loss in all but the most unlikely situations, and used recoverable directory information to determine storage allocation rather than a traditional linked-list organization. The simplicity and reliability of the file system was an important key to the success of CP/M: file access to relatively slow floppy disks was immediate, and disks could be changed without losing files or mixing data records. And because CP/M is a Spartan system, today's increased storage-media transfer rates simply improve overall response. The refinements found in CP/M are based on its simplicity, reliability, and a proper match with limited-resource computers.
*****

MP/M

As single-user CP/M became widely accepted, Digital Research began to develop a new operating system for real-time processing. The design called for a real-time nucleus to support cooperating sequential processes, including a CP/M-compatible file manager with terminal-handling capabilities. This operating system, called MP/M (Multi-Programming monitor for Microcomputers), is a further refinement of the process model

found in Intel's RMX and National's Starplex. As a side effect, the combination of MP/M's real-time nucleus with the terminal handler and the CP/M file system produces a traditional timesharing system with multiprogramming and multiterminal features.
*****

CP/NET

CP/NET, introduced in late 1980, leads a series of network-oriented operating systems that distribute operating system functions throughout a network of non-homogeneous processors. CP/NET connects CP/M requesters to MP/M servers through the use of an arbitrary network protocol. Similar to CP/M and MP/M, CP/NET consists of the invariant portion, along with a set of field-reconfigurable subroutines that define the interface to a particular network. For purposes of CP/NET, this interface needs only provide point-to-point data-packet transmission. Since the actual data transmission media are unimportant to CP/NET, any one of the number of standard protocols can be used, from low-speed RS-232-C through high-speed Ethernet. Physical connections are also arbitrary, allowing active hub-star, ring, and common-bus architectures.
*****

PL/I: the application language

In 1978, Digital Research investigated the final level of software support: application languages. One such language was to be supported throughout the operating system product line, and the choice would have to be a multipurpose language. Further the language would have to be an international standard to promote the generation of software by independent vendors. Standard Pascal seemed a logical choice but was rejected for several reasons. First, Pascal is an ALGOL derivative with scientific orientation. Commercial facilities in the standard language are absent: decimal arithmetic, file processing, string operations, and error-exception handling were essential. Further, separate compilation and initialization of tables were not in the language. There was a temptation to extend Pascal in order to include these features, but these extensions would have defeated the benefits of standardization.
*****

New processor architectures

We've spent little time discussing processor refinements. What is happening to our software tools as we augment our 8-bit machines with the more powerful 16-bit processors? Will 16-bit processors replace 8-bit machines, or are they simply a temporary phenomenon in the transition to 32-bit machines?

There are several considerations when answering these questions. First, 8-bit machines are economical to produce, their software systems are mature, and they satisfy the needs of a substantial computer base. Therefore, we can safely assume that 8-bit machines are here to stay. Newer 16-bit machines are marginally faster, but they have substantially more address space. To use this additional address space, the computer must contain more memory, which increases the computer system cost.

As system costs increase, the margin between low-end minicomputers and high-end microcomputers diminishes, placing microcomputer hardware and software manufacturers such as ourselves in direct competition with major minicomputer manufacturers. The 16-bit machines, by their nature, introduce memory segmentation problems that are not present in 32-bit processors.

Finally, we should note that 16-bit minicomputers are already out-moded, and all serious manufacturers are pushing 32-bit machines. This leads to the following conclusion: if we are tracking the minicomputer world, we can assume that the future will be with the 32-bit processors.
*****

Software vendors

We've concerned ourselves with three levels of software tools that support the most important level: the application programs. A major reason for CP/M's popularity is the general availability of good application software. At last count, there were about 500 commercially available CP/M-compatible software products.
*****

"PL/I For Limited Resource Computers" by Gary A. KILDALL (Microsystems, Jan/Feb 1982, pp.28-29)

PL/I, Programming Language One, has in one form or another been with us for nearly twenty years. Although a pragmatic language, it was considered large, unwieldy, and difficult to implement. Recently, however, the language has been revitalized through the efforts of the American National Standards Insitute (ANSI) Technical Committee X3J1 where the General Purpose Subset language was defined. This so-called "Subset-G" language is upward compatible with full PL/I, but is designed expressly for mini-computer implementation. The elements selected for inclusion within Subset-G are the most commonly used facilities used in commercial, scientific, and educational application programming. Redundant language constructs, little-used facilities, and error-prone statement forms were eliminated, resulting in a sub-language which most observers believe is superior to the full language in many ways.
*****

PL/I was originally conceived in the early 1960's by the Advanced Language Development Committee of the SHARE Fortran Project, in the wake of interest created by ALGOL, FORTRAN, and COBOL. Elements of each of these languages were incorporated into the original design: block structure, nested scope of variables, procedure formats, and array referencing were, like PASCAL, derived from ALGOL. Scientific facilities came from FORTRAN, including separate compilation, expression formulation, floating-point arithmetic, some I/O formation, and a wide variety of transcendental functions. Commercial processing in PL/I was derived from COBOL, including structures, decimal arithmetic, file processing, and picture formats. A variety of new statement forms were added to allow character string processing and error-exception handling, which where considered essential for high-level application programming. Real-time multi-tasking facilities were also added to allow PL/I to be used for systems programming as well. The language which resulted from this design effort contains more built-in data types, arithmetic operations, and general-purpose programming facilities than any other programming language available today. But herein lies the primary difficulty with full PL/I. The language is too large to implement effectively on any but the largest mainframes. The complexity of the language also inhibited proper use of all language features, while the unwary programmer was often trapped by strange twists and nuances of the language. Nevertheless, PL/I has proved to be a practical, pragmatic language for application programmers over the past several years, through implementations on a variety of mainframe computers.
*****

The Digital Research PL/I-80 programming system project was started in 1978, and completed two years later. PL/I-80 is based upon Subset-G, with nearly all of the Subset-G features, and operates under the Digital Research CP/M, multiprogramming MP/M, and CP/NET network operating systems for

8080, 8085, and Z-80 microprocessors. The PL/I-80 programming system itself consists of the compiler, macro assembler, linkage editor, program librarian, and run-time subroutine library.
*****

The PL/I-80 programming system is currently being transported to 16-bit processors, with initial support for the Intel 8088 and 8086 processors, so that designers may select either 8-bit or 16-bit processors for their applications programs. The transition to the Intel processors is simplified in two ways. First, the compiler itself is written in PL/M, Intel's high-level system language, with portions of the run-time system written in PL/I. Thus, only the semantic handlers need to be altered, along with conversion of the space and time critical run-time subroutines, such as the floating-point library, which are implemented in assembly language.
*****

I hope you have gotten the flavor and will pursue the references for the entire text. Dr. Dobbs' entire publishing history is available on CDROM from them. The other references should be available in technical libraries and from friends. BDK.

From: RICKR@AIB.COM@INET01#
To: B.KIBLER
Bill:

Roger Hanscom has been connecting 1.2MB, 5" floppy drives to older systems which expect 8" drives with no modification. This is an interesting idea which could greatly enhance the life of these older machines, since all of the 8" drives are wearing out. I suggested he write to you about writing an article about it.

Another thing he's been doing is saving all his EPROMS on floppies. We easily forget that the insulated-gate EPROMs are only guaranteed to keep data for about 7 years - and our older machines could be well beyond that. Imagine trying to find a replacement EPROM for an old S-100 board or a Kaypro or Xerox 820. Scary thought! In fact maybe we should collect a library of the ROM images.

Forwarded message:

From: roger hanscom
<hanscom@athens.dis.anl.gov>
Subject: Re: disk upgrades, etc.

Hi Rick —
There was, on some, a ROM monitor which could be enabled instead of the TurboDOS load routine. There are several models of these boards, too...
- IMS 740 - Z80 + 64K
- IMS 1000 - 80186 + 128K
- IMS xxx - Z80 B or H + 128K

Most of mine are Z80 with 64k memory. The EPROMS on them have a very simple little program that appears to be able to download and store in memory. They have CTC, PIO, SIO/0, and, of course, the 8255 that is the bus interface. I'd love to find out what the port numbers are for all the peripherals. Also I discovered that there is a jumper on the board that allows it to run. This must be under software control, but I've yet to discover how to turn it on, other than installing that jumper.

As it happened I forgot to get the drive. I did get a Kaypro 10 for $20 - it works fine! - and a digitizing pad for $10, and a few other bargains... but I just got too sick from ragweed pollen to see the whole fest.

That's a pretty good price!

In the meantime *both* of the Qume DT8's in my IMS system are flaking out. The system has had more or less continuous use for 15 years, so I can't complain. I have two Shugart 851's, condition unknown, and two Tandon half-heights, but it turns out that one is single-sided. Also have a couple more Qumes in the storage room, one known dead.

Sounds like you really need to get some 1.2Mb 5.25"s

But then I suppose - WAIT - we need a drive which actually changes SPEED, not one of the ones that fakes it by letting you use a different data rate. Does the Mitsumi do that?

There is a jumper to set spindle speed to 300 or 360. There is one configuration that is "300/360 switchable", but I can't see what would "switch" it.

I sent mail to Bill K's compuserve account, so we'll see if he gets it. I've had no luck with getting e-mail to genie.geis.com.

I'm working on getting you a copy of those manuals. Perhaps I can Xerox reduce them.

roger hanscom@athens.dis.anl.gov

OK Roger and Rick, seems this discussion on using drives needs to be settled a bit more. I had understood that 3.5" drives in their high density version (1.4 MB) were identical to 8 inch drives in speed and tracks. You two make me think I am wrong here, and it is AT 5 inch drives. I guess what is really called for is some comparisons and a little bit of research about all the drives, or did I just layout another article to be done.

I believe I did ask Roger by return e-mail to try and do me an article, but I am not sure he has the resources we need to really fill in all the gaps. Now when I worked at Teletek, they sold a simple 8 to 5 adapter that shifted the control lines around. I made a few later for myself and it really is rather simple. So, adapting the cabling from 8 inch interfaces to 5 or 3 inch is rather simple. The problem is not lines, but speeds, steps, and tracking.

Well thanks for the spin on using 5 inch drives instead of 8's. I will be looking to do more or see more later. Thanks. Bill.

From: JDB8042@TAMSUN.TAMU.EDU@INTERNET#
To: B.KIBLER

Dear Mr. Kibler:

It's been a while since I last wrote to _TCJ_, but it seems I don't write unless something provokes me in some manner.

I'm glad to see you haven't lost your enthusiasm for an ISA-bus Z180 system. I'd very much like to see something like this myself. My main question is how you see such a board being implemented. One possible design would be a stand-alone SBC that uses an ordinary AT-class *BM clone as the I/O processor (handling the physical devices such as the keyboard, screen, and disk drives). Another design would have the Z180 card be the bus master on an ISA passive backplane (such as Brad Rodriguez's 6809 multiprocessor cards). What type of implementation did you have in mind?

In my first letter, I mentioned that my goals for my Amiga 500 were TeX and CAD work. The CAD work is still too expensive for a poor college student to afford, but TeX is up and running beautifully thanks to Georg Hessmann's PasTeX implementation. Thanks to an Amiga implementation of MetaFont, I can generate fonts and get beautiful hard-copy output on my dot-matrix printer.

My Amiga 500 has grown quite nicely in the year or so since my first letter. It sports some respectable hard disk space and even a CD-ROM drive. Actually, the CD-ROM drive is a Commodore CDTV that is networked to

the A500 via their parallel ports and ParNet device driver software. As soon as the CP/M CD-ROM starts shipping, I'll be ready for it! (By the way, "CDTV" stands for "Commodore Dynamic Total Vision" and is one killer audio CD player in addition to being a complete Amiga 500 with CD-ROM drive itself!) Best part is, it was all cheap to get and simple to put together.

In some brief correspondence that followed my first letter, you addressed the question of "user support" for the Epson QX-10. If you were asking about organized user groups, then I must say that I know of none right now—although I've heard of a few in the past. Even then, I don't have any information about them. QX-10 users seem to be pretty rare in the U.S., but it seems there's a bit stronger following in Europe.

A plea for help came to me from Emmanuel Roche in France, forwarded to me in e-mail by Tilmann Reh. Seems Mr. Roche has a number of Epson QX-10's to which he'd like to adapt Mr. Reh's 8-bit IDE host adapter. At the time I didn't have the skill or confidence to attempt such a project as that, so I forwarded his request to Wayne Sung. Wayne accepted the challenge.

After a few months of working on an appropriate decoder off and on, Wayne had come up with a prototype. It turned out that the IDE drive he had was faulty, so I sent him one I'd scavenged from a dead GRiD System's '286 luggable. With it, Wayne verified that his control sequencer EPROM was working properly. When Wayne returned the drive to me, he included two of the decoder EPROMs and a preliminary schematic.

I set about collecting the parts to build my own version of the IDE host adapter—in this case, for my own QX-10, since it would require the least additional work. (The decoder EPROM is effectively system independent.) I uncovered a couple of errors in the schematic and some errors in the assembly source for the QX-10 BIOS patches. Once those were fixed, I had two 8MB hard disk partitions running on my 20MB Conner CP3022 IDE drive!

I should point out that this was my first construction project ever. I've since used the other EPROM to build a fairly generic IDE host adapter, but I'll need to build some other hardware to interface it to a system. I have a Davidge DSB 4/6 SBC that would be ideal as a test platform, but I've got more to do on it before I can think about writing a hard disk BIOS for it. (Namely, I need to finish disassembling and commenting the existing BIOS—I don't have source code for the actual CBIOS, just a skeleton BIOS.)

One thing I learned is that the support strategy for European QX-10 users differs quite a bit from that for North American (U.S.?) QX-10 users. The biggest example is that European users use a different BIOS implementation (MF-CP/M), get BIOS source code with the package, and even have CP/M Plus for the QX-10. Hard disk systems are up to the user to implement, but at least they have the BIOS source to work with.

North American QX-10 users get a canned banked CP/M 2.2 implementation with (limited) built-in hard disk support and that's it. No BIOS source, no CP/M Plus. (The last I heard, BIOS source is still available, but Epson still wants $500.00 per copy.)

While working with the BIOS patches to enable IDE support, I took steps to improve upon my existing Comrex Comfiler hard disk (the standard hard disk system for North American QX-10 systems). My Comfiler has a 6-head IMI-5018 hard disk in it (as a result of playing "musical parts" with another machine). The stock QX-10 BIOS only understands 4 heads. After a bit of study, I implemented a 16-bit "divide-by-n" routine to replace the original BIOS code that translates the CP/M logical track to cylinder and head for the hard disk and set it up to divide by 6.

I then turned a QX-10 4-head hard disk format program into a 6-head format program. Soon, I had a 14.2MB Comfiler instead of a 9.5MB Comfiler. I'll never fill that up! I also included some assembly- time switches for IDE hard disk support.

I started work on some generic IDE/WD100x utilities, starting with a C version of Tilmann Reh's IDE ID program, written with Aztec C-II v1.06d. I've hit a serious snag with this in that I'm having trouble with unsigned long integers (which I need for holding the total number of sectors on the drive). When trying to print the disk's size in megabytes, I get nonsense. Needless to say, development is on hold until I can either figure out Aztec C's hangups or switch to Hi-Tech C or perhaps Turbo Modula-2 (which are what I have most readily available).

In the week before I had to come back up to school, I learned quite a bit about the SASI/SCSI interface and how to design a host adapter and write a device driver for it. I also managed to find that my Davidge DSB 4/6's expansion connector is tailor-made for attaching a SASI host adapter with DMA I/O capabilities. All interesting stuff and I wish I could spend more time on it.

I probably failed to be specific about anything again. This letter seems to have turned into a "How I Spent My Summer Vacation" report, and probably an incoherent one at that. One of these days I'm going to sit down and write at length and in detail about a single topic, but either the time or the provocation is lacking. In the meantime, there are so many nifty little things to learn about, and I can only learn about them by taking things apart and playing with them.

Take care and enjoy.

John D. Baker ->A TransWarp'802'd Apple //e CardZ180 Z-System nut//
Internet: jdb8042@tamsun.tamu.edu, @blkbox.com, jdbaker@taronga.com
BBSs: JOHN BAKER on PIC of the Mid-Town [(713) 961-5817] 1:106/31,
The Vector Board [(716) 544-1863], Z-Node #45 [(713) 937-8886]

That is great John, your advice and discussion I am sure has helped many of our readers. I have yet to get any "how it went" articles on the IDE drive project. Yours is the first I have heard about. Maybe others now will be willing to fill us all in on their results.

How about more on the QX-10 work, and getting Wayne to start writing again for TCJ. Wayne did some networking articles in times past, and it is nice to see he has time to still tinker. It is most interesting to see the different way companies handle the same item in Europe, compared to the States. That of course is one reason I hunted down Helmut for the European Beat series. I think Atari still sells 90% of their machines to Europe, and leaving the US retailers out to dry (or die as is more correct).

On the ISA Z80 system, I am thinking now the new 40MHZ Z380 might be the way to go. It runs both regular Z80 code as well as advanced commands to do 32 bit operations. It has four sets of registers which means you could do a 4 users CP/M system very simply. As to being a master or slave, I figured a master is best. No need to build a slave, just use MYZ80. But for a real Z80 system using the cheaper ISA bus I/O devices was my idea. I might also point out that many companies are now using 68000 based ISA bus systems for real time control, usually running OS9 68K. From what I have heard, those users are very happy with the

results. So why not do the same, but with a Z80 or Z380 system. Some might think of moving their STD BUS projects to it, especially with the option of 40 MHZ performance.

So let me know what you think of the ISA/Z80 idea. For you, I guess the main question many of the QX-10 users will have now, is how they can get CP/M Plus for their machines, and oh yes, IDE decoders? Thanks John, great letter! Bill Kibler.

Sub: CoCo serial I/O
From: James Jones
<jejones@microware.com>

Concerning your question about CoCo serial I/O in TCJ 69: the stock CoCo indeed bangs the bits on half a PIA (the other half reads switch closings on the keyboard). Tandy made two cartridges for the cartridge slot or Multi-Pak Interface that would do serial I/O with real serial I/O hardware (a 6551 ACIA chip):

1. The "Modem-Pak," which had a 300 bps modem and the serial chip, and, I believe, some minimal comm program in ROM (a fellow named Marty Goodman, whom you should contact if you're interested in the CoCo, came out with instructions on how to disable the ROM and modem, so you could just use the ACIA as a real serial port).

2. The "RS-232 Pak," which had an ACIA chip and a DB-25 connection.

Third-party companies came out with other serial hardware, notably Disto/CRC and Ken-Ton. There are still third-party companies doing CoCo stuff; where serial hardware is concerned, you would want to contact Rick Uland with CoNect, or the Ken-Ton folks, who are still around and also make a good SCSI interface for the CoCo. (One third-party company, which as far as I know is no longer around, made a cartridge with *four* serial ports for the CoCo.)

(BTW, probably the best compact source of CoCo information, this side of ordering the Technical Manual from Tandy, is Frank Swygert's book *Tandy's Little Wonder*. A significant part of it is Alfredo Santos's history of the CoCo, and Mr. Santos mentions OS-9 exactly twice, so IMHO that aspect of the history is not complete, but still it's a book well worth having if you're interested in the CoCo. My sole association with Farna Systems is that I subscribe to *The World of 68 Micros* and think it's a good magazine.)

James Jones (former CoCo user, current MM/1(a) user)

Opinions (and any errors) herein are those of the author, and not necessarily those of any organization.

Well James, nice to hear from you and to get those bits of knowledge. The CoCo is still in my opinion an underrated machine, except by those who still use it. And Yes OS9 is very important to it and many others as well.

I think most users feel "bit-banging" is a bad way to move data, but depending on the instruction set, the overhead is often little more than using regular serial devices. Motorola's 6805 programmers manual has a very good example of bit-banging, if you want to see a simple implementation.

What I guess we need is the actual address or phone numbers of the companies you listed so I can add them to the User Groups support list. MM/1(a)? How about a little info on that one, can't match it up to anything I am aware of, yet. Thanks . Bill.

From: BFINCH@ASP.VET.PURDUE.EDU@INET01#
To: B.KIBLER
Sub: bound volumes

hello and tnxs for putting out a great 'zine' ..... i read it THRU every issue and am seriouly considering gettiing a full set of back issues BUT it is rather $$ (at least for me and my miserly ways) .... in any event i noticed u had indicated issue 32 was TEMPORARILY out of stock but now with issue 69 is back in stock with 15 issues. this brings up a question ..... in earlier issues u had mentioned u would be binding these up and creating volumes .... to wit; are u still considering this? if so when (and if so), and with what issue do u intend to stop doing so? etc ..... since the bound volumes make sense, and are a bit less $$ ... i may well consider this if and when u are complete ... etc ..... in any event PLEASE keep up the gud work and best regards ... baab

Well I guess some answers from me are needed here. I made 15 copies of issue #32 to hold readers till I can make masters for the next volume. Another reason is the other issues in that next group all have at least 15 copies waiting to be sold. So I am sort-of waiting to get the numbers closer to all gone. As to the issues in Volume 5, 32 to 37, and maybe volume 6 will be needed about then as well (with 38 to 43). So there you have the facts from the source. Thanks for the inquiry. Bill.

Dear Bill

Please extend my subscription to The Computer Journal for another 2 years.

CP/M seems to be dying away. I am in the Windsor Bulletin Board, which has had a hardware failure and will not be coming back on stream. Other clubs do exist. I am in BOOG (British Osborne) but if I seek knowledge, all the members say "I have been on a PC for years, you are on your own". They had a so-called repair evening, some machines in a sorry state came in, they were in an equally sorry state when they went home! All back We did have a rather downputting atmosphere in most clubs to what I could call the second generation of members. We were 2nd class. If one had an AMSTRAD, then horror of horrors, how down market. That is why the CP/M UG Died, the magazine was full of editorial comments that drove readers away. Towards the end, it became Yes such a thing can be done, in fact I have done it. So you can do it yourself (go away, implied).

One thing you may wish to mention is that the CP/M disassembly programs advertised (and editorially mentioned) in the TCJ ARE STILL AVAILABLE for those who need them. I do not have the name with me. CP/M 2.2 and CP/M plus.

I have recently purchased Z system, NZ, DSD, ZMAC and others. Be warned, these will NOT LOAD on the OSBORNE Executive. Jay Sage says he has never seen before the error messages that I sent to him. This may be on a par with HISOFT products sold over here (and the USA?). Their CP/M versions will not load. The editor coming with 5 programs vary a little. Each stops at a different part of the autosetup.

Someone has said it is the Osborne interrupts, but Jay wrote it can't be that. Has ANYONE ever got the Z system etc to run on an Exec? IT IS NOT A DUD MACHINE! I have all diagnostics discs, manuals etc and everything agrees with manufacturers spec. Osborne did publish the full details of bios etc, but not knowing what could be blocking the system, this does not help. (This is why I got DSD, I hoped to find out why the HiSoft stuff did not run).

I have been writing to a lot of your old advertisers hoping to find one that still makes the realtime clock that plugs in with a Z80 in piggyback socket. No replies. Do your

readers know of anyone? OR do they have one I can purchase. Even the circuit would help. I could make it, but do not know how to design it. Could you consider a center fold on such small circuits? They should (?) be generic!

I also wrote (my postage bill is high!) to S100 Herb. As he told me the Osborne is Not S100, but I knew that. Could someone write & say why I could not add an extra socket wired to the S100 standard & change any CP/M computer to be an S100 device. I always thought from Sol Libbes articles that it was a hardware thing, and did not need special software? Am I being an idiot? Does the same apply to SCSI (with chip as a line driver if needed)? It is the simple things that baffle!

Again, Osborne etc are SSDD. If I wish to change the hardware and bios (easy) to DSDD WHICH machine's format should I emulate for ease for exchange with others? (For SSDD, I have found getting stuff on Kaypro II is easiest and I then change it to Osborne with Media Master). Lacking a DSDD equivalent of Mediamaster (and NOT wanting to have to use SYDEX PC Programs to alter CP/M discs more than I have to). Is there a semi-generic DSDD (and if so what are its details as far as CP/M needs to know?)

Finally, We are still in the boondocks over here. Email is not readily available in business or at home, so when only that is given as a contact "address" with your authors, it does block me, at least! It must apply to others, even in USA!

Regards! Keep up the good work! John Butler, London, England.

Wow John, lots of stuff going on over your way. To start from the bottom and work up, I do try to get my writers to provide real addresses and contact info, but not everyone wants to cooperate. My feeling is, send it to me and I'll forward or reroute it to them.

As to a disk format, I think your idea of using Kaypro format if you can change the BIOS is correct. As to a standard, the PC clone format is just about the only real standard out there. And I believe Jay has a program for Zsystem that will read DOS disks, so you might think about that option. Actually the way I feel many should consider is adding a hard drive and just doing modem from other sources. Or, get an actual Kaypro and have it talk to the Osborne serially, gives you an extra CP/M machine as well.

On problems with other people wanting CP/M to go away, what can I say? I think they are wrong for considering it dead. It really shows too, as you said when the broken system entered and then left still broken. I find most PC Clone users have very little idea what goes on inside their machine, and often could care less. What TCJ tries to teach by using the older and simpler machines is how they work and why they work that way.

As a teacher I have found many other teachers glossing over or ignoring valid questions that might require study on their part. Since teachers are in the business to explain things and they often don't when it comes to the insides of machines, how can we expect users, often supposed super users to be able to explain things as well.

The final word I guess is that only TCJ is left to aid and provide support for non-pc systems. We have been doing it for over ten years and have to consideration to stop doing so. I do plan on continuing to expand coverage so PC using wanting to know about the insides can get a little help here, but have no fear it will always be just a little help, with the main details found in smaller systems like Z80s or 6809s.

Thanks for writing and don't lose faith. Bill.

Dear Bill,

While going through my CPUZ180 development notes, it entered my mind that the part on a floppy disk problem might make a story for your magazine. I am submitting the result for what it is worth.

On another matter, the PT IDE802 IDE and Centronics interface, derived from the CPUZ180 SBC's CPUZIDE chip, uses the IDE engine from the PT IDE100 S100 controller. This chip can provide most 8-bit CPU based designs with IDE and Centronics capability at very little or no extra logic. The core is an EP1810 PLD, a close cousin of the EP1830 housed in the PT IDE100, but not as power hungry. I enclose a preliminary spec sheet for your perusal. If you think your readers will be interested I can put some words together on it, or feel free to use the sheet itself. The PT IDE802 is currently available from here in sample quantities for AU$55.00 (about US$41.00). Please contact me if you require something more about it.

Since my last letter to you re: the CPUZ180 I made some changes to the memory select circuitry to improve access timing. The result was that even slow 120 ns FLASH memory worked on zero waits. I will still be fitting the faster 70ns parts, but it is nice to know there is plenty of margin. Consequently the beta version boards are operating without wait states. I am getting some photos of a board and will send you one when they come to hand.

Regards, Claude Palm, cnr. Moonah & Wills Sts., Boulia, QLD. 4829, Australia. (077) 463-109.

Thanks Claude for the extra information. I like your article and it will be in #71 space permitting. The Z-Letter printed the picture of your CPUZ180 on their cover and your letter describing it's operation within. As you can see I printed the description and the two drawings of your IDE802. I hope the information is enough for our readers to see if it provides a better way for them to interface than using several TTLs.

Since your project is for more than one unit, and allowed for several variations from one development project, it is easy to see that it is very cost effect in your case (also fits on a single board better than several devices). This then starts my arm twisting, by asking for another article. Many readers have considered using PLDs, but learning and getting the necessary software is a big problem. How about an article that covers just what you had to do to develop the chip. How many failures did you have? What was the cost of the development system? Did you first do it in TTLs? Do you use PLDs and such for one up jobs? If so why?

Well thanks again for the letter and keeping Z80's alive. Bill Kibler.

User Interface Description
IDE

The PT IDE802 performs all 16 to 8 bit conversions as required to make it completely transparent to the user. Simply treat it as if the system was connected to an 8-bit IDE drive.

16-bit disk I/O is performed in MSB - LSB format. Prerecorded data such as ASCII characters in drive identification are thus read out in their correct order. Note however that the IBM PC stores data the opposite way.

Sector data can be read out in any number of bytes, but should be written to the drive in even numbers only. The first byte in a word

is stored in a holding register and is not written to the drive until the second byte is received. Any attempt to write an odd number of bytes, will loose the last byte.

Sector buffer I/O can be interrupted at any time to communicate with the printer without data loss, nor will any IDE task file read access upset a sector buffer data transfer in progress. A write to IDE COMMAND (port 7) will however clear the sequencer, to guarantee a clean start.

Interrupts

CTINT occurs on a high to low transition on the ACK* input. An access to any CSI port 0 to 2 will clear a CTINT. During CTINT, the open collector CTINT* output is pulled low. The inverted state of CTINT* can be read from bit 1, CENT STATUS 3. Trying to read it from CENT STATUS 1 will clear CTINT and return the bit as 0, regardless of what state it was in.

IRQ1, IRQ2 can be used as IDE and some other interrupt or as general purpose inputs. Their states are accessible on bits 0,2 in the CENT STATUS ports 1,3. INT1*, INT2* are open collector outputs reflecting the inverted state of IRQ1, IRQ2.

All 3 outputs are open collector, and can be wire OR'ed into a single line.

STB* output: Auto Strobe:

A write to CENT DATA 0 will assert strobe on the trailing edge of the write pulse. Data is latched and output on the leading edge. If the write pulse is too short to satisfy the printer setup times, the write should be preceded by a write to CENT DATA 2 so that the data is already on the bus. A read from CENT DATA port 0 will disassert STB*. That way it is not necessary to manipulate the control register in order to issue a STB*. Other printer control outputs can be set/reset by writing to CENT CTRL 1, without affecting STB*.

Auto Strobe example:

```
OUT     CENT DATA-2,data     ;place data on bus
OUT     CENT DATA-0,data     ;assert STB*
IN      CENT-DATA-0          ;disassert STB*
```

Manual Strobe:

STB* will follow bit 0 in writes to CENT CTRL 3, and can thus be used as a general purpose output. All CENT DATA accesses should then be performed via CENT DATA 2. Once latched, it can still be manipulate via Auto Strobe at a later stage.

Uni/Bi-directional Mode

The MODE pin is hardwired to select the required operating mode for the Centronics data bus CD0-7.

MODE pin high: Bi-directional mode.

A high to low transition on ACK* or INIT* will place CD0-7 bus in Hi-Z as the pins become inputs. Once the edge is detected, ACK* or INIT* can be disasserted and the bus will remain in the input mode. Input data is not latched, so a read from CENT DATA 0 or 2 will reflect the current state on CD0-7 pins.

Writing to CENT DATA 0 or 2 will place CD0-7 in output mode and latch the data onto the bus. CD0-7 will remain in the output mode and data from any subsequent writes will be latched to the bus. Auto strobe may be used if required. Reading CENT DATA 0 or 2 while in output mode will not change that state, only return the data last written.

The output mode is exited by a low pulse on the ACK* or INIT* pins. As INIT* is controlled from bit 7, CENT CTRL 1 or 3, it can be used to switch to input mode rather than waiting for an ACK* pulse.

MODE pin low: Uni-directional mode.

CD0-7 is always in output mode. Write to CENT DATA 0 or 2 latches the data onto the bus until next write. Reading CENT DATA 0 or 2 will return the data last written. The state of ACK* or INIT* is ignored.

SPEAKER output

This output is intended for a piezzo buzzer or other indicator. It is manipulated via the SPEAKER port and can be SET, CLEARED or TOGGLED. Writing to that port will not affect any other pin or operation, while reading from it returns the current contents of DB0-7.

General

PT IDE802 is based on an EP1810 CMOS PLD

```
Package:    68 pin PLCC
ICC:                typically <100MA
IOH/IOL:    typically >15 mA at TTL levels
VIH/VIL:    TTL compatible

Brief timing specs:
DB0-15 <-> D0-7 <-> CD0-7: 40nS
A0-2 -> AB0-2: 35nS
RD*/WR*/CS0*/CS1* -> HIOR*/HIOW*: 35nS
IRQ1-2 -> INT1-2: 35nS
IO16* valid to data (D0-7 or DB0-15) valid: 35ns
RD* asserted to D0-7 output: 35nS (75nS from CS0*/CS1*)
D0-7 hold from RD* disasserted: 35nS to Hi-Z
D0-7 hold after WR* disasserted: >5nS
DB0-15 hold from WR* disasserted: 35nS to Hi-Z
CS0*/CS1* stable prior to RD*/WR* strobe >5nS
CS0*/CS1* hold after RD*/WR* strobe >0nS
```

Detailed electrical and timing specs to be issued.

Generally, to calculate total IDE access time, add 75nS to the drive's specifications (address stable to read data valid).

A 'SLEEP MODE' version is being investigated. This chip would require (2 octal) buffers to disable the CPU interface during sleep, when ICC should drop to below 100 uA.

Notes on the CPU interface

CS0* and CS1* should be connected to the corresponding pins on the IDE drive and be derived from an address decoder. It is advisable to minimize decoding delays for these inputs.

Neither the drive nor the PT IDE802 will react to spurious pulses on the select lines while RD or WR strobes are inactive. CS0* and CS1* could even be derived directly from two address lines where minimal decoding is acceptable. RD and WR strobes should then be conditioned to occur only during I/O cycles. Conversely, active RD or WR's are ignored when CS0* and CS1* are inactive. Any spurious transitions on the RD/WR pins during active CS0* will misclock the sequencer.

The IDE drive must be given sufficient time to validate its IOCS16* output. RD/WR data may be compromised if that line has not settled at least 35 nS before any data access. The drive IOCS16* pin is a function of its CS0* and A0-2 inputs. For that reason the select lines should be given a higher decoding priority over the RD/WR strobes. Refer to drive specs for actual delay. In any case, PT IDE802 require its select inputs to be stable during the entire RD/WR strobe.

The bidirectional D0-7 bus is driven only during a RD* strobe with either CS0* or CS1* asserted.

The two drawings are on page 20, at the end of Dr. S-100. I might note some interesting considerations in these specs, need for two byte writes, some sequence timing concerns, and chip selects (CS0&1). I don't remember if Tilmann's interface has all of these constraints or not. Some of the handshakes can

be handled in the BIOS, but others are hardware and as such might limit the chip usefulness on some machines. BDK.

Dear Bill,

Thank you for the sample issue. It sold me on your magazine. Please start my subscription and send the following back issues: Volume 2, issue 32, issue 36, and issue 65.

I heard about TCJ from a friend that said Sinclair got the occassional mention. I started by buildin a Sinclair ZX80 and have stayed with them. My main computer is now a Sinclair QL with a Motorola 24 MHz 68020 with 4 MBytes of on-board 32 bit ram. Don't believe anyone that says the QL is dead! My QL is not the highest performance QL available. The QXL is a 20 MHz 68EC040 with 8 MBytes of 32 bit ram that lives on a board that plugs into any PC clone. High performance QLs also are available as plug-in boards for Atari ST and TT models. There also is a QL available as a software only emulator that runs on the Amiga.

Would you mention the following support available for Sinclair and Timex computers:

IQLR (International QL Report). This magazine is in its fourth year and is steadily growing its subscription base. The last issue was 65 pages. IQLR is published 6 times per year and has never been late. Contact Bob Dyl (401)849-3805 or write IQLR, 15 Kilburn Ct., Newport, RI 02840. Subscription rate is $20 per year.

Update Magazine is published 4 times per year. Update covers all Sinclair, Timex and Cambridge computers. Contact Frank Davis (317) 473-8031 or write Update Magazine, P.O. Box 1095, Peru, IN 46970. Subscription rate is $18 per year.

QBox-USA is a BBS supporting all Sinclair and Timex computers. QBox-USA has been on-line 24 hours a day 7 days a week since October 1993. QBox-USA is a point off the Fido-Net in Europe. This means QBox-USA carries all the European Sinclair message traffic. Message areas include: International QL, Minerva, Quanta, QBox and Spectrum. You can communicate with users in the U.K., Germany, the Netherlands and other countries for the cost of a call to Detroit. There also is a healthy file area carrying the latest in public domain software. QBox-USA runs on a Sinclair QL with a USR 14400 modem. Callers from 300 to 14400 baud are welcome. There are no fees. The number is (810) 254-9878.

I'd also volunteer to answer any QL related questions your subscribers might have. I have 16 years of experience as a computer service engineer for one of the largest computer companies. I've also fixed quite a few QLs so have the background to help others.

By the way, CP/M runs on the QL under two different emulators. I have both of them but have not used them in quite awhile. I would be interested in seeing an article about CP/M on the QL.

Regards, Don Walterman, 331 Drace, Rochester, MI 48307, (810) 656-4108

Thanks Don for all that good QL information! Guess I will have to subscribe to IQLR. Since I like 68K systems and really always wanted to get a QL, you make me think I will find one to buy there. I am very interested in all the emulators and co-processor cards you listed. Seems like there could be a real long story about them all. How about it?

I have run CP/M emulation on my Atari St and it works just fine. The 68K systems are very good, in fact much better than most of the PC clone line for sure. Thanks again and keep us posted on QL news. Bill Kibler.

To Whom It May Concern:

I am enclosing a postal money order for the renewal of my subscription to The Computer Journal, beginning with #66.

I am particularly interested in CP/M articles. I am using my original CP/M computer that I started out with, a Timex/Sinclair 2068 with a Portuguese disk system. I also have 3 more of the same disk system as spares. It is like a C128, in that, it has 2 other disk operating system: TOS (Timex Operating System) & the Spectrum disk OS - very similar to TOS, but for a ZX Spectrum. I also have an Eagle, 4 Osborne-1s, a TeleVideo TPC-1 (needs a boot disk), a C64 with the CP/M cartridge, a Sony SMC-70 that I picked up recently for $20 at a flea market (needs all cables, e.g., the video monitor cable) with the original "pinch-to-close" 3.5" floppies, an Osborne Executive in need of 4 disk rails & the lighted on-off switch, and an Industrial Micro Systems (IMS) S-100 that I need to take to the S-100 Doctor.

I use dBASE II ver. 2.43* for my data base needs, WordStar 4.0 for my word processor, SuperCalc 2 for my spreadsheet needs, & IMP for telecommunications. I have many CP/M programs, however, I am currently looking for the following programs for sale:

1) MicroSoft COBOL
2) MicroFocus CIS COBOL
3) CrossTalk (or XTALK) ver. 3.00 or higher
4) VEdit & VSpell
5) The accompanying installation programs for ea I listed above.

I hope you print this letter so that if there is someone out there that can help me, I would appreciate it.

Thank You, Jay S. Siegel, 1274 49TH ST #821, BROOKLYN NY, 11219-3091

P.S. You may , publish my address since it is a mailing address; it's not where I actually live. This is NY, you know, & I wouldn't give,that out!

I understand your concerns about the address Jay, and hope that someone helps you out. I might try Lambda Publishing first for those boot disks or programs. I do have a boot for the TPC-1 (I think). My mother in law had one and I keep copies of the disk in case she blew them up.

I am rather interested in the portugese disk system, why and how? Just seems like the Sinclair machines are more popular than many think. Hope you find your software and thanks for writing. Bill.

WANTED

TCJ needs your embedded story or project!

Our readers are waiting to hear from you about how you developed that embedded project using 8051 or 6805. Used Forth, "C", BASIC, or assembler any language is fine, just tell us what happened, how you did it,and how it ended up. No project too small!

Send those Ideas to:

The Computer Journal
P.O. Box 535
Lincoln, CA 95648-0535

The Z-System Corner II

By Ron Mitchell

Regular Feature

ZCPR Support

CP/M part 3

Keeps Those Cards and Letters Coming!

Many thanks to Al Warsh of the Amstrad PCW SIG in Colton California for his recent letter. Al writes that it took him some time to get used to CP/M and that he's looking forward to learning more about ZCPR. He describes how he came by his copy of ZCPR3 through a deal with Jay Sage involving a surplus Amstrad PCW 8256s and a newsletter trade. All sounds quite familiar. The 21 eight bit computers in my apartment here in Ottawa were acquired in similar fashion, more or less.

Al sent along a copy of his latest Amstrad PCW SIG newsletter, a good read. The issue contains an adeptly chosen balance of hardware and software topics as well as a healthy offering of input from the users.

It's a small world. One of the letters in Al's latest issue was from a gent I know very well out in Qualicum Beach, British Columbia. If you know Canada's west coast at all you'll quickly pinpoint Qualicum Beach about half way up the eastern side of Vancouver Island. Beautiful country. My parents live about 30 miles north which means I get to pay a visit to this particular friend whenever I'm in the neighborhood. We've been computer buddies for some years, having as we do a common interest in another Z-80 machine, the Coleco ADAM.

Thanks also to Fritz Chwolka who sent me an Internet E-mail saying, "Thanks for writing these articles in *TCJ* and I hope reading more from you.."

Well Fritz, we'll give it a good shot and hopefully the beginners will be able to follow and the old hands might have a chuckle or two watching me struggle. Fritz goes on to add that he is also a collector and a reader of Tilmann Reh on the Z280 system.

Keep those cards and letter coming folks. I need your comments and I enjoy reading letters such as Al's.

Past and Future Guidance

While we're on the subject of newsletters, there is a whole series of articles prepared by Jay Sage that precedes my series here. *TCJ* readers of long standing will know that the original Z-System Corner was ably prepared by Jay Sage. As was described in the last issue, Jay is one of the real pioneers of the Z-System. We'll be drawing liberally on his expertise.

If you have a CP/M or Z-Node BBS handy you might just find the earlier installments of Jay's work available as public domain text files. If not, *TCJ* has back issues available (see pages 48 and 49 for details). Jay's work begins in *TCJ*'s issue 25 and continues through to issue 67. While I haven't checked for continuity, I assume that Jay was a fairly regular contributor because that's the way Jay does business. His original submissions appeared under the title, "The ZSIG Corner", and I notice that the third in the series was dated January 1987.

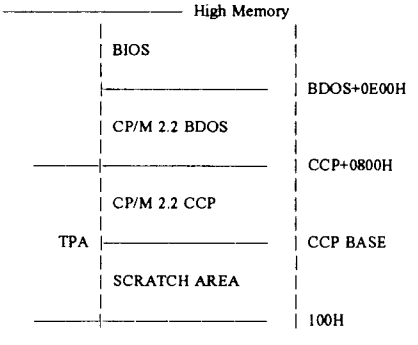
You might find Jay's work considerably more advanced than the level we're proposing to follow here, but it is valuable stuff to read, and you'll do well to give it a try if you have it available.

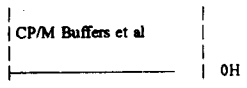
While you're looking for Jay's material, keep an eye open for something called the RCPM list. There you'll probably find the Z-NODE list close by. The RCPM list is a compilation of all known CP/M BBS's throughout North America and around the world. You'll find it helpful when looking for CP/M or Z-System software. The Z-NODE list provides the names, locations, and telephone numbers of Z-System contacts around the continent and as far away as Europe and New Zealand. If you're looking for help, you might just find someone close by on one of these two lists. They have both been updated to September 1994.

Down to business

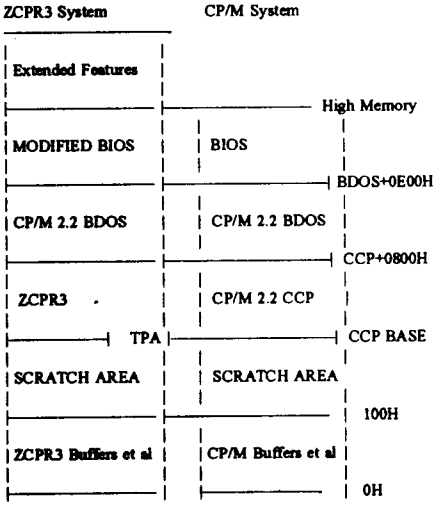
To quote author Richard Conn; (ZCPR 3 - The Manual) "CP/M is an operating system: that is, a computer program whose function is to manage the resources of the computer and to provide services in response to standardized requests from applications programs (which manipulate data in ways that serve the user's specific needs)."

That's about as clear as it gets. It doesn't matter whether you are talking about CP/M, MS-DOS UNIX , OS2, OS9, or Rubberjunk's Bottleneck, that is what operating systems do. In the last issue (#69) we drew a small model of the CP/M operating system; it looked as follows:





And that is the layout of CP/M 2.2 or as it's sometimes disparagingly known as "vanilla CP/M". Remember the terms: BDOS is Basic Disk Operating System; BIOS is Basic Input Output System; and CCP is Console Command Processor. The area between memory location 0H and 0100H is known as page 0. A page is 256 bytes or 100H bytes.



The TPA is the Transient Program Area. This is the space which your application has available. It ranges typically between 48K and 61K depending upon how much RAM your system has. It can even be smaller than that. Note that an application that is going to obliterate or replace the CCP (either CP/M 2.2's or ZCPR's) had better make sure that it comes equipped with some other self-contained means of getting input from the keyboard, or from wherever it's designed to get input. It had also better replace its divots when it is finished; ie. replace the CCP or re-boot the operating system.

The difference as is obvious from the diagram is an enhanced Console Command Processor and the extended features in high memory. This is what we mean to explore.

Begin with the BIOS

To complete our discussion of the rock bottom basics, let's return for a moment to the 'computer resources' that Richard Conn was talking about. What are they? We listed them in the last installment: memory, processors and processes (running programs), devices, and information. Remember that it is device management where CP/M excels and that device management is performed by the BIOS. Control of and communication with disk drives, keyboards, printers, modems, interface cards, and other wonderful widgets are taken care of, boss. You don't have to worry about a thing. Amongst the functions performed by the BIOS there are the following, (and I'm quoting Conn again because it's the first place I've seen it described in understandable terms):

1. Initialization functions; cold boot, warm boot, (setting preliminary values in strategic memory locations for later use, defining pointers to various important places in memory);

2. Character Input/Output functions; (includes status checks - got anything for me)- console, printer, and auxiliary input/output;

3. Disk Input/Output Functions; 'homing a drive', selecting a drive, selecting a track and sector, reading and writing a block of data, selecting a memory address from which to get it or at which to put it, and some mystical function called 'logical to physical' sector translation. (About this latter we shall have to learn.)

Busy fellow this BIOS.

I don't want to get hung up on this stuff. Everything I've been talking about so far is located in the first 5 pages of Richard Conn's book and it seems like we're not making much progress. We certainly haven't said much yet about ZCPR3 or Z-System except to draw a picture of it. In your own basic books on CP/M you're likely to find similar descriptions of how various elements of this fascinating piece of programming fit together. My purpose here is only to provide a broad brush background of where Z-System sits.

It is in fact, as we have said, an enhanced Console Command Processor. In our last installment we said that 'vanilla' CP/M knows only six words: DIR ERA REN USER SAVE TYPE. Everything else has to come from a transient command or program.

What's Wrong with CP/M?

And while we're running down 'vanilla' CP/M, let's really do a number.

1) Each time you change disks, you have to 'warm boot' the system before you write to the new disk, that is you must press <CONTROL> C or an infuriating error message will result, usually where you least need it.

2) There is no standard method of clearing the screen from the operating system level. In some cases it's <CONTROL> L, in some cases it's <CONTROL> Z. Who knows?

3) You can't assign names to user areas.

4) Each time you obtain a new piece of software for your system you have to configure it specifically for your particular setup. In the public domain world, you'll be lucky to get the Install file along with the main program, and that will lead to trouble.

5) You cannot set a search path. The program you call up had better be in the drive and user area you're logged into or the system won't find it.

6) Programmers have no access to flow control at the operating system level. If you're assembling the source code of a program and that source contains an error, there is little point in carrying on with the linking and loading of that file. The resulting .COM file probably won't run.

7) Shells, scripts, alias's? What are these?

8) With 'vanilla' what you see is what you must carry with you. There is no possibility of configuring the system to the specific requirements of the job at hand.

9) No multiple command lines.

10) No access to a vast and comprehensive set of tools available under Z-System.

There's a payoff all right. Now I am prepared to grant you that many of the points in my foregoing list of ten are things that only a CP/M programmer would appreciate. If you're going to use CP/M to run applications without looking under the hood, then you might miss the importance of some of this. So let me describe some personal experience that will perhaps pave the way for a more positive description of where we're going.

Home of Micro's 22

I have a collection of old computers. Several of them are CP/M compatible. There's an Osborne 1, a couple of Commodore 128's, an Apple II, a PMC 101 Micromate, five Coleco ADAM's, and a partridge in a pear tree. There's more than that lining my wall space but they're not CP/M machines so they don't count.

On the ADAM I had the benefit of a little help from my friends, who made me what I am today. (That still has them worried.) The advent of a CP/M workalike called TDOS (Tony Morehen and Guy Cousineau, AJM Software) spoiled me rotten when it came to CP/M. All of a sudden I could clear the screen of my computer with a simple CLS command. I could copy files without invoking COPY.COM or the dreaded PIP.COM. There were many many utilities that came with this thing, utilities which I've since come to recognize as part of the Z-System package.

So I got used to all of this and the housekeeping of my CP/M collection became a pleasure rather than a chore. Then out of the blue, somebody put a Xerox 820-II on my desk at work (this was a few years ago), and I suddenly began to appreciate the value of these little enhancements that had been put at my disposal. The Xerox was 'vanilla' CP/M 2.2 and stood out in stark contrast because of what it couldn't do. The Osborne was the same type of experience. Operating a pair of 180K (give or take) disk drives without a search path was a real challenge. There's little enough disk space (when you've been used to a 20 or 40 MEG hard drive elsewhere) and you need a 'if-the-program-isn't-here-then-go-look-there' capability that the search path provides.

There was also the benefit of being able to operate in a uniform fashion across several different machines. The Osborne in my collection has had more of a workout since NZ-COM arrived in this apartment than ever before. It sits by my easy-chair in the living room where I can fall asleep and compute at the same time. Ain't life grand!

Tall Tales and TCAPS

One of the more sizable advantages offered by Z-System is its ability to include right in the operating system an environment descriptor or terminal capability. Shorten that to TCAP. The various functions of a video terminal, as you may know, are controlled by a series of codes usually consisting of the 'escape' character plus one or more characters sent either in ASCII or HEX. On the Cybernex XL-87 attached to my PMC101 Micromate for example, the code to clear the screen is 1B 45 hex or <ESCAPE> E. On the Coleco ADAM attached to my Hazeltine 1510 the same operation is accomplished by sending a 7E 1C hex or ~ <CONTROL> \. On the Commodore 128 (Terminal emulates a Lear Siegler ADM 3A) it's a 1A hex or <CONTROL> Z. Three different terminals, three different ways to clear the screen. If I wanted to run a CP/M application on these three computers I would have to either patch it with the appropriate codes or use MLOAD.COM with a suitable overlay. Each new application to be run on any of these machines would have to be similarly installed. The same applies to printers which also perform various functions such as bolding and underlining, etc.

The Z-System TCAP takes care of all that. You install your terminal only once and when you run Z-System compatible software from then on it, there is no installation required. And that's only the beginning of the payoff.

As these issues unfold we'll be exploring the many other benefits that Z-System has to offer. I presently have Z3PLUS installed on this PMC 101 Micromate and NZ-COM installed on Ossie the Osborne. The installation procedure went right by the book with Ossie. With the Z3PLUS version there was one glitch that the book didn't tell me about, and I want to check that out with Jay Sage before proceeding further. I'll get into details on that in the next article.

I'd like to finish this article with a little more philosophy. I'd like to let you know exactly where I'm at in terms of my knowledge of CP/M and Z-System so that there'll be absolutely no misunderstandings about what you're getting here.

I'm writing this from the perspective of one who is discovering it all for the first time. That's the way it is. The Z-System has been resident in this household for only a few months, and sooner or later the experts following these articles are going to call my bluff. I'll counter that threat before it even get's here by saying that I certainly wouldn't want to be mistaken for an expert. We've covered this ground before.

There is lot's of help available. I have an able group of CP/M 'gurus' right in my back yard. Ian Cottrell has been a good friend for many years. Many of you will recognize him as the co-author and present guardian of the PBBS 5.x BBS program. Ian's INFOCENTRE board is the local Z-NODE which offers an impressive list of software to chew on. Then there's my long time friend and fellow ADAM owner, Guy Cousineau, who's implementation of CP/M 2.2 on the Coleco ADAM, known as TDOS, had me enjoying the benefits of Z-System without really knowing it. As previously said, it's only when I return to a CP/M machine that runs 'vanilla' that I realize how far we've come. There are others up here upon whose considerable expertise I can draw.

Further afield, but only as far away as the nearest FIDONET or INTERNET drop there are many many experienced Z-System and CP/M people who are more

than willing to lend a hand. As you increase your contacts with these people on the CP/M Tech Echo and COMP.OS.CPM you'll find them friendly and helpful. Don't be afraid to ask questions.

Speaking of which, it seems appropriate to finish this installment with something of a quick and dirty shopping list. What do we want to know?

You're likely to find me wandering all over the place. I was leafing through the Z section of Ian's BBS last night and found a literal ton of programs waiting to be written about. Each could generate it's own article about some aspect of Z-System and that would take more than a lifetime. In amongst the programs were the earlier installments of the Z-System Corner by Jay Sage as already mentioned. I took a look at the first three and realized how much of this I don't know yet. Jay introduces some of the new Z utilities that were being written in and around 1987 and describes techniques for speeding up the performance of Z-System through strategic placement on the storage media of the various elements. Then, at Ian Cottrell's place last week, he showed me the power of an 'alias', a string of several Z-System commands contained in a single .COM file and usable simply by typing one command. There is much to learn.

The value of it all will depend upon what you're doing with your computer and what capabilities you need from your operating system. We'll have to explore what benefits there might be for those who simply want to use one or two applications such as word processing perhaps or a spreadsheet. Those making use of various disk maintenance and programming utilities might stand to benefit to a greater degree, Those interested in assembly language programming and the development of applications will be looking for a completely different set of capabilities from Z-system.

How do you rate these things? Improved efficiency? Doing a particular computing job in less time with fewer steps? A more comfortable feel to the user interface because you've designed it yourself? Greater transient program area made available by not having to use space for program routines that you don't need? The ability to change your operating system on the fly? Designing your own tools to get the job done? This is only a small portion of the list of benefits claimed for Z-System.

Upcoming articles will describe my own experiences with the product. So far I've removed it from the box and, as mentioned, have installed it on two computers. That's the sum total of my achievement so far. Beginning with the next installment, we're actually going to start using Z-System. We'll install the TCAP for both NZ-COM and Z3PLUS, and take a look at organizing a system disk for our Z-System work.

In the meantime, here's the quick and dirty shopping list that describes in very rough fashion what we can hope to explore over the next few sessions.

1) What is an alias?

2) What is a shell, and what function do shells perform?

3) There are all sorts of acronyms to be mastered; two of the most common are the FCP and the RCP. What do these stand for, and how do these elements of Z-System contribute to it's power as an operating system.

4) We're going to make a startup disk for both Z3PLUS and for NZ-COM. What do we want on this disk. Another way of putting this question would be that you're marooned on a desert island with your favorite computer and only one floppy disk. What would that disk contain?

5) How does one go about installing Z-System, and what are the pitfalls to watch out for?

6) What do the programs ARUNZ and SALIAS do? What other utilities are available under Z-System?

7) What is the effect of Z-System on the availability of transient program area? What is the cost in terms of overhead?

8) How can Z-System be configured for different jobs?

9) What are the advantages of flow control, and how is it achieved?

10) What about multiple command lines? What good are they, and where are they most frequently used?

There are many many more questions that we'll be dealing with in future articles. These 10 are not mentioned in any sort of order, and they'll be answered as we get to them.

Dr. S-100

By Herb R. Johnson

Regular Feature

Intermediate

Letters In The Mail Bag

"Dr. S-100's pre-Winter column" by Herb Johnson (c) Oct 1994
Internet: hjohnson@pluto.njcc.com

Introduction

A lot of Internet messages appear in my column, partly because my volume of US mail has declined a bit. Mail should pick up in the winter. This column I follow up on interests in my S-100 IDE interface from last issue, and a bit on the Western Digital hard disk interfaces from a knowledgeable reader. A few of my colleagues get some press for their help (and probably to you in the future), and I tell you who has given me what lately, and how you can get yours.

(By the way, let me know if you like reading my correspondence, or want more hardware-packed articles, or what. You know where I am! Also, I apologize for any spelling errors in this issue's column, but my spelling checker stops at every 5th word, like IDE, net, CPMTECH, etc....)

Networking

I'm getting more and more traffic on the Internet, after I announced my netmail address last column. I've also noticed the annual decline of mail in the FidoNet echo area CPMTECH. At least, I hope it is only the summer slowdown and not a loss of interest! People are busy playing in the summer, and only get back to their basements - I mean their computer workshops - in the winter after the chores of fall are completed.

Correspondence

As always, please include your network address or BBS location in your written or phone correspondence. This will allow us to respond to your requests rapidly and cheaply. And, if you use an address of any sort from my column, please note the source! And, if you want our correspondence private, please tell me so: trust me to use some discretion in quoting you and in my comments.

Denis J Carlos of Newman CA has a number of California Computer Systems cards among others to sell that "would be nice to see put to use again." They include 16k static RAM, Buss terminator, 4-port serial, floppy and Z80 by CCS; a George Morrow HD (hard disk) controller and a Godbout 18 slot buss board. Contact him for more information.

Trades and orphans

Many people have S-100 systems that are about to be dumped. Perhaps you are one of them: the spouse insists that the garage be used for cars instead of computers; or the desk be cleared for the "new" computer; or the spare bedroom be used for a precious collection of natural disaster newspaper clippings instead of your computer museum. Maybe you simply want to "pass on" your system to someone who will learn from it.

The Dr. gets a number of calls of this sort, and tries to accommodate when he can. Jim Briggs of Mt Laurel NJ called me last month, to try to "place" his Compupro 8/16 system in a good home. And it is quite a system. Many readers may recognize this system as the "predecessor" of the Heath/Zenith Z-100 which was an unlicensed copy of the 8/16. It has an 8088 and an 8085 processor and runs both MS-DOS and CP/M. The Z-100 is one of the more popular "bridge" systems of the post-IBM PC period, bought in large part by the military for base use and "dumped" in surplus over the last few years.

Jim's S-100 system includes some nice color video cards and a pair of 8" Qume drives, and included both software and manuals. It is rare to be offered such a well-documented system. Jim was so pleased to be able to provide it to a knowledgeable person that he offered to deliver it the day of his call! Well, the Dr. knows when to ACCEPT a house call! Jim arrived a few hours later, not only with his system but with an Epson LX-80 printer and a Canon bubble jet color printer! When the cold of winter sets in, I'll have more time to describe this system in detail. As it is one of the most popular and powerful S-100 systems of its time and deserves a full review.

For the reader, I would suggest one way YOU can find a good system is to simply place an add in the computer section of your local paper. Note that you will provide "a good home" for a CP/M system or old computer. You should be specific that you don't want an IBM or Mac or Commodore, etc. as your tastes dictate. This will save you from many calls. Don't think of this as begging, but as a kind of "adoption." You might mention "don't take it to the curb" too.

Colleagues

My colleague David McGlone of Lambda Software and I often assist each other in hardware and software for our clients. David also had a chance to "rescue" several systems and offered me a share in the loot - I mean the to-be orphaned systems. As a result I will receive a Compupro 68000 card and some other

Compupro docs and disks. David is strictly a Z-80 kind of guy, and I am an S-100 specialist (all doctors are specialists these days) so our interests overlap occasionally. David is a good resource, as is his magazine The Z-Letter which I recommend to everyone.

Another colleague of high standing in the classic computer community is Lee Hart, "the Heathman," who is also a regular letter writer to TCJ. While assisting me with a Z-100 customer, he sent me two Zenith binders of documentation on CP/M 80 and CP/M 86 for the Z-100. And, he has a lending library of books and documentation on these and many other systems, as well as Z80 and other processors. His fees are extremely modest: just a few dollars plus postage. If you need some more information on classic systems, particularly the Heath/Zenith line, send him a request and he'll send you a list of books and stuff for loan. (You should throw in a dollar or two to cover his costs. Mention my name so he knows who is buttering his bread.)

S-100 IDE fan mail

As I write, the previous issue of The Computer Journal has only been out a few weeks. Even so, I've received a few letters and netmail on the S-100 IDE design by Claude Palm of Palmtech in Australia. I've had quite a net correspondence with John Wilson <wilsonj@rpi.edu> about IDE and S-100 stuff, and I've mailed him a copy of the IEEE-696 specification for the S-100. John had developed an IDE for the IBM (?) and has REALLY classic computers going back to the PDP-11! (more on that later). One of the things I enjoy about my column is that it introduces me to a variety of active hobbyists and to early developers of classic systems.

One message of note is from Tilmann Reh, TCJ's correspondent from Germany:

" ...reading your column in TCJ #67, I think you should consider the following. When you are thinking of an IDE interface board for the S-100 bus, you should carefully check if the single-chip solution using the PLA of the Australian [Claude Palm] also is the cheapest solution.

In my experience with circuit design, it often has proved that a circuit with more but cheap TTL chips was much cheaper than the highly integrated (and more elegant) solution. When having a look at my IDE interface board for the ECB bus, I could also have done it with a single PLA (except for the bus drivers), but the TTL solution is much cheaper. When using GAL [gate array logic, another programmable logic chip class] I normally use the cheap standard parts like 16V8 and 20V8...even a 22V10 is much more expensive and might not compensate [for] several more TTL's!

Some more hints on planned options: If you want to implement an FDC, use the SMC FDC37C65C. I can recommend it very much, and it works up to 1 Mbps data rates (as used in the ED drives and disks [example: the 2.88 Mbyte IBM floppies]. However, don't let the FDC select the four drives, use an external TTL register chip instead! If you need details, contact me freely.

For the serial interface, you might consider using an 82C452 chip which is commonly used in PC's. This is very cheap, and will do for most purposes. Compared to other cheap devices, it has two on-chip programmable baud rate generators. If you want to [instead] do the best, use one of the Zilog SSC family chips.

In any case, design the decoding circuitry so that each additional interface might be built up optionally. This way, you keep the cost low for those who just want the IDE interface."

[Tilmann is of course correct: given the large "real estate" of the S-100 card, and that most printed circuit board shops charge by the square inch and not by the hole, a collection of cheap chips is cheaper than one expensive chip. Of course, Claude was designing for space rather than cost and so the single-chip solution was best for him. Nonetheless, his choice for prototyping was the S-100 bus!]

(BTW, Tilmann also reminds me that my previous correspondence was from "Emmanuel Roche" and not "Roche Emmanuel", who has the habit of swapping his first name and surname in his letterhead.)

Another bit of Internet traffic on the S100-IDE comes from Patrick Logan <PLOGAN@leonis.dsccc.com>:

"Herbert, I am good for 2 of the little darlings. How do you want payment ? Instead of a S100 board, how about a generic board that mounts to the IDE-drive and cables over to the CPU-board ? or can be mounted to a S100 board ? or ....."

Yeah, there's only a couple of people right now suggesting that they will have a "small" or "generic" IDE board available very soon now.....Seems there is always someone "out there" who has something that can be adapted, but somehow it never happens. The fact is, it is hard to make SPECIFIC hardware and especially software "general." Ask anybody who has done Z-system development.

There's one more dance in the old girl yet...and yet another fan of the 'IDE is Bob Finch <bfinch@asp.vet.purdue.edu> who writes in an interesting style which I will quote unaltered:

"wow!!!...i am impressed with the stuff in issue 69 of tcj.....to wit;....is this chip available in more generic flavors for z-80 NON-S100 systems...if so what are the particuulars ...and
what about software etc.....i have a single board with scsi interface....i am currently ruunning my de-block into a adaptec 4000 (a) board and formatting rll drives as sif they were mfm....
BUT ide may be option to seriouly consider.....running zcpr3 something, of course.....

nice columm and keep up the gud work...tjc is an interesting mag.....but sometimes i want mor e COMPLETE stuff.....i know this can be difficult for any number of reasons, and i do sincerely hope tcj doesn'tt sell out on either 6800 or z:-80 coverage to the pc et al.....

please taae care es best regards.....baab"

[After a quick and sharp reply by The Doctor (that's me) that suggested that he was looking for a simpler kit than I had considered providing, and what does he really want anyway. Bob writes:]

"thank you for the quick reply!... so, anyways, i really didn't mean to leave u with a wrong impression to wit; yes sometimes a 'kit' would be easier, sometimes just software and hardware as component items is sufficient, sometimes just software, uh (at least for me) seldom just hardware.....so i am looking at software, and was writting u to find out about chip availability (and styles), as well as software availability.....AS PER YOUR BOLD LETTERING ON PAGE 25 OF THE CURENT ISSUE (69)OF TCJ

as for my comment about sometimes things are skimpy in tcj it was NOT directed at you. your collumns are exceptionally interesting, and I have no complaints. in fact i have no complaints about anyone's col., but sometimes the authors DO assume that hardware or software is trivial. it may be to them, but not to some readers. puesdo code is often times not enough, sometimes is, and because magazines such as tcj have to serve a larger interest, col. size is limited. i guess my concerns are related to a lack of space, which, by it's very nature, is not a critique applicable to the authors.....so please don't take my musing too seriously

finally yes ide could well be more attractive to me.....i don't know, but seriously doubt that my adaptec 4000 board is fully compliant with modern scsi drives (or vice versa) and that it wouldn't just be easier to meld something else in.....i haven't really checked, but alot of water has gone under the bridge since i put this together in the very early 80's

again i have NO complaints of any nature with you or your column, it's content or otherwise, and YES i would be interested in this new development of ide drives for z-80's and z-sys (or cp/m).....please do continue with software and statemachine info etc.....and thanxs for being a resourse...

sincerely;

bob finch (baab)"

[Does anyone recognize the literary style of Bob's writing? The subtitle gives a hint.]

Bob makes the same point that Bill Kibler makes to me: everything MUST BE EXPLAINED! New readers, or old readers who lack experience, or just plain confused readers, need to know the details! And, readers with one system may greatly desire the features in an article on another system but will have great difficulty adapting it over if details are lacking. I try in PRIVATE correspondence to make up for what lacks in my PUBLIC writing, if only to save space; but we can all do a bit more.

So far these are the nibbles of interest in IDE, but I have had no flood of mail or messages! If you want to see anymore of this project, call or write or netmail me NOW! Even with a few requests, I hope to provide chips and docs to those interested even if I don't publish the rest of this project.

More info on Western Digital controllers: pre-IDE?

John Baker <jdb8042@blkbox.com> replies to my previous column's notes about Western Digital (WD) hard disk controller cards among other's:

"I just received issue #69 of _TCJ_ and I was a tad concerned that you assumed that the WD-1002-05 was in any way related to SASI/SCSI. [I had presumed it was a SASI controller, like some of the Adaptec and other controllers of the era.] The WD100x-yyy boards bear absolutely no resemblance to SASI/SCSI. I've read a number of comments from people in various fora [forums?] in which they assert that these boards are SASI.

The WD100x boards are found in a number of classic computer systems, such as the Kaypro 10, Televideo 80xH, and Epson QX-10 (via Comrex Comfiler). They are also the original hard disk controllers developed for the IBM PC's.

A few notable variants:

WD1002-05 — stand-alone hard disk and floppy disk controller card uses WD1010 HDC and WD2797 FDC

WD1002-HDO — "Hard Disk Only" controller card, like above but without FDC circuitry (not installed). These are the boards most commonly found in Kaypro and Epson.

WD1005-yyy etc. IBM PC-bus version of the above. Much of the trouble that users experience with modern IDE drives comes from some limitations of the WD1010 HDC used in these boards.

Yes, the WD100x series of hard disk controllers were the fore-runners of today's IDE hard disks. In fact, the WD100x Task File Interface is nearly identical to the IDE Task File Interface. The only changes are re-assignment of some bits in the Size/Drive/Head register and that all 8 bits of the Cylinder High register are available for IDE.

As you may recall from my traffic in CPMTECH this summer, I explained this to several people. I have an IDE drive running on my Epson QX-10 and the only significant BIOS modifications required were to increment the sector number by 1 and explicitly set a 1-sector transfer. All other code is the original routines for the WD1002-HDO that was supplied by Epson.

Remember, WD-100x-yyy is NOT SASI/SCSI, but IS very close to IDE....

Take care." [John's message "signature" follows as a matter of interest:]

John D. Baker ->A TransWarp'802'd Apple //e CardZ180 Z-System nut // Internet: jdb8042@tamsun.tamu.edu, @blkbox.com, jdbaker@taronga.com BBS: JOHN BAKER on PIC of the Mid-Town [(713) 961-5817] 1:106/31, Z-Node #45 [(713) 937-8886], The Vector Board [(716) 544-1863]

(hurrumph!) Well, I guess I stand corrected. By the way (BTW), IDE at the

REGISTER and COMMAND level is a "stretch" of the Western Digital floppy disk register and command set.

More Classics: the PDP-11

My day job in the 1980's at Eastman Kodak included some work with Digital Equipment Corporation PDP-11 minicomputers and VAXes. The 11's had been around for several years and I used them in college too. They have a straightforward instruction set, 16-bit registers, and good amounts of memory. Peripherals were memory-mapped, and in fact a Motorola processor programmer would find a lot of PDP 11 code and practices familiar.

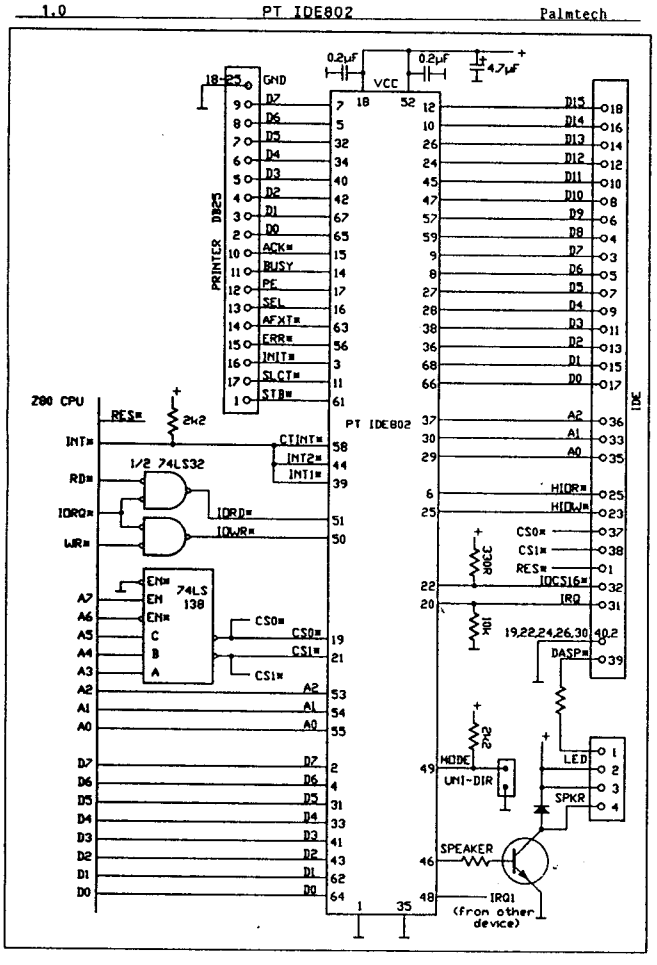
As John Wilson wrote me about S-100 and IDE stuff, it emerged that he has been working on a PDP-11 simulator on the IBM-PC for some time. It supports a number of serial ports for terminals, and has been running some of the operating systems by emulation. It supports 8-inch floppy drives with the Compaticard floppy controller.

Interested? Contact John or download it from the Internet: John says, "Well it's free, they can snag it from FTP.UPDATE.UU.SE, pub/ibmpc/emulators, files e11.com and e11.doc, whenever they want (starting with the next version it'll be e11.exe, I ran out of space). If you are interested in commercial use "for running your factory off an old PDP 11/45" as John suggests, he is interested in providing customer support for a fee. I asked John about writing a TCJ article: send him encouragements (me too!).

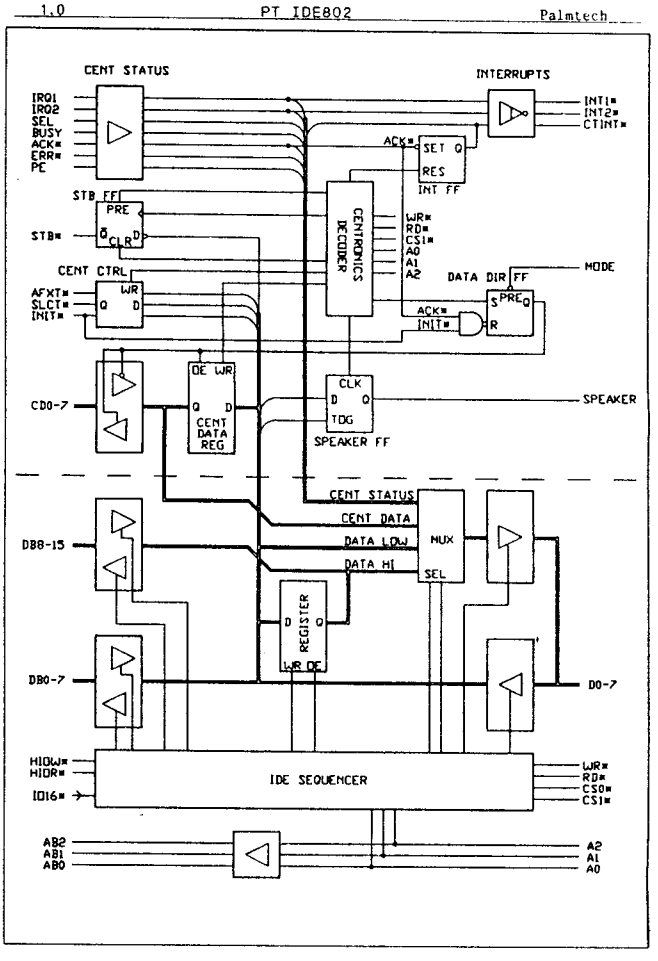
References

John Wilson <wilsonj@rpi.edu>, 11 Bank St Troy NY 12180.
Tilmann Reh <tilmann@hrz.uni-siegen.d400.de>.
David McGlone, Lambda Software - see his ad this issue.
Lee Hart <lhart@P02.mn10.honeywell.com> 4209 France Ave N. Robbinsdale MN 55422.
Dennis J Carlos, 568 Real Ct, Newman CA 95360.
Patrick Logan <PLOGAN@leonis.dsccc.com>.
John Baker <jdb8042@blkbox.COM>
Bob Finch <bfinch@asp.vet.purdue.edu>
Herb Johnson <hjohnson@pluto.njcc.com>

Reader To Reader Continued:



Design example to interface the PT IDE802 with a Z80 CPU.



PT IDE802 Block Diagram.

The European Beat

by Helmut Jungkunz

Regular Feature

All Users

An AMSTRAD Expert?

Helmut sent me this first half by E-Mail just after I took #69 to the printer. Since I think it helps explain about his background, here it is. BDK.

How to become an expert 8-bit idiot

I still remember the time, when the image of a computer still related to ENIAC or monsters of that size. I also remember leaving my home town to settle out for a new life in the big city. After four weeks of jobbing in a carpet company and endless jam sessions with fellow musicians (I wanted to become a pro then), I got this job with a certain American company, Digital Equipment Corporation. I worked as a stock clerk there, when soon I detected my ability to memorize all the part numbers involved, in fact, even today, 21 years later, I remember a number like 70-6267-01 for the "Head, Flying Upper" and I was one of the best and fastest databases around. Before I could turn my head to say no, I found myself in a position as a leading stock organizer for all Germany. I even flew to a central European meeting in Amsterdam. Mind you, I still didn't know what they needed all these parts for, I just had some vague ideas, how Teletype devices could be used for data entry and output.

This tells the basics of the rest of my "computer career". I always worked high above my actual knowledge level, creating funny situations sometimes, when "colleagues" started to throw their vocabulary at me, expecting me to fully comprehend everything they said, maybe even give them some advice!

When we had a sudden vacancy in our flat, a guy moved in with a Sinclair ZX81. Boy, was I impressed, how they managed to get such a fantastic flight-simulator on a tiny little machine like that! The last computer I had seen before that, had been a PDP-8e, quite a monster, like a refrigerator with drawers and dozens of toggle switches.

After a couple of months, the guy moved out, but my contacts with him survived. He advised me to buy my first home computer, the AMSTRAD CPC 464, like you might have read before. Again, here I developed a video interface to connect the CPC to a colour tv through stealing some ideas from a technical magazine's test generator. I wrote articles on subjects that I had to dig up information to know the basics.

This is maybe not too serious, but there was a certain side effect to that: in order to stay honest, I tried my best to keep the information authentical. This again forced me to read a lot and get deeper into many subjects, than I would ever had without the articles. After a short while, I was writing more articles on "monitors", "the video side" and so on for a computer mag. I volunteered to test listings of readers (my BASIC knowledge was still very limited then) and even managed to improve some of them after a while. I wrote articles for our club magazine and soon advanced to the head of the club.

In the meantime, I got to test hardware for the CPCs. I used my mere likes or dislikes to judge the handiness of some gear, being fully aware, that exactly that view served the readers most. Of course, again, I learnt increasing amount of facts, and could be of great value in the later development of new hardware for the CPC.

All of a sudden, I was presented with a dying magazine thatdidn't have enough people to properly support the readers. So they asked me to assist with this job. As always, things spiraled out of hand and I ended up answering all the letters to the editor. Sometimes I was completely lost as to what the people wanted to know about. I spent lots of time on the phone, but there were few people who knew as much on the subject as I did. Now this was a new situation for me. I had become a specialist, not being aware of it. While not being able to understand the more sophisticated BASIC programs I received, I spent much of my time reading the documents from some of the SIG/M public domain discs I had purchased. After all, I found it much more interesting to read about this ZCPR thing they got wild about. I even got more information from the MORROW OWNER's REVIEW (MOR), when I read articles like "Tools for Tyros" or "Forever Z". Still, I wasn't even able to properly use MOVCPM. The tool I mostly used, was DDTZ, a German Z80 version of DDT, with a string query option and a write function built in.

Thus, I garbled up several versions of PONG, ALIENS and the like. I don't know why, but our club members started to talk about me as "public domain expert". I don't know, why this always happens to me, after collecting all the Z-System files and distributing some Z-System programs, everybody expected me to be a Z-guru. In order to read some news from comp.os.cpm I got an account in a publicly accessible UNIX machine. To be able to at least do the routine stuff, I had to read lots o' man pages. In the middle of getting less dumb, I got the o fer to

work as a UNIX teacher! I promise, I hadn't intended to do anything in that direction, but it seems, these things just "happen" to me!

So I guess, being an idiot is not such a big problem, as long as everybody else is a) either equipped with less knowledge than you, b) believes you know more than they do, c) is ignorant of the right time to say yes.

In any case, I can now call myself an expert idiot.

Thanks for the background info and now on to this issue's regular fair from Europe. BDK.

Sweet End of Sugar for Computers (More about AMSTRADs)

Well, last time I gave you the insights into AMSTRAD's CP/M machines, the software and the sales development. You got your first impression of "European feel" in the computer world.

AMSTRAD DPBs and Typical System Commands Beyond the Usual

The AMSTRAD CP/M world had always been split into two parts: CP/M 2.2 and CP/M Plus. Not only was there total incompatibility of terminal commands, but different disk formats and tools.

In the traditional CP/M 2.2 world there was SETUP tools, and a lot of things could be performed with a batch file via SUBMIT, which influenced CP/M Plus. In CP/M 2.2 there is only the so-called I/O-Byte that connected the devices to the operating system calls. In CP/M Plus a generic AUX device is accessible, independent of any settings.

The AMSTRAD CP/M 2.2 SETUP allows you to modify the keyboard settings, such as the arrow keys in Normal, Shift, and Control level. This required quite some concentration and lots of skill, since small errors could affect screen colors or windowing.The CPC supports true windowing by use of one of the most popular characters: CTRL-Z! So, whenever you accidentally ran a program set up for Televideo or ADM3A characteristics, you ended up in a screen size of 0, since those terminals use CTRL-Z as the CLEARSCREEN sequence, followed by a Null-Byte.

Today, a convenient adaptation of the CP/M Plus SETKEYS has been written for CP/M 2.2 by one of the most skilled authors in our club, Robert Steindl.

The greater handicap is the AMSTRAD CP/M 2.2 BIOS (AMSDOS) itself, which supports only single-sided disk formats with 40 tracks, although it includes a (nowadays unused) standard format, the SS40 IBM format with 8 sectors (old CP/M86 format). Mind you, there is no such thing as a 3" disk on an IBM CP/M86 machine! Of course, some companies offered programs to manipulate the Disk Parameter Block (DPB) in order to handle alien disk formats, but then they were not widely known and were available only in certain areas.

Again, CP/M Plus put an end to most of that. The whole Disk Parameter Block resides in one line inside the CP/M Plus memory, making it easy to patch and modify even from the command line, provided one has the proper tools (PK.COM). If a small patch was made to the system file, the .EMS file on the boot disk, even two-sided formats could be used. By using the proper system function, GETDPB, any altered CP/M Plus system could deliver the address of it's DPB, and so any program could search for it and modify it. Unfortunately, most PCW programmers are unaware of these standards and keep on using fixed addresses, hardcoded inside their programs. Those will have to be rewritten in parts to run on the CPC's CP/M Plus.

An example is the program DU+49 (the programmers weren't even aware of the existing DU-series programs), which can cause tremendous heartache, even to owners of a PCW, if they have any modifications or add-ons in their machine (or attached to it). As a matter of fact, we have a certain circle of programmers busy repairing the sloppy output of others.

There is a certain goodie buried in the Disk Parameter Block of the AMSTRAD CP/M Plus - the so-called Freeze-Flag. The name signals it's function: if set (FF Hex), the DPB is "frozen" and cannot automatically adjust itself to the inserted new disk's format (especially on the PCW, this Auto-Login information can be a serious trap). That is the only way that an alien disk format can be used properly. Otherwise, every warm boot or disk reset would cause a reset of the DPB information to the values stored in the BIOS, and sometimes this happened by accident when CP/M Plus read some code similar to it's log-information at the beginning of a disk.

The CPC's CP/M Plus differs from the PCW series quite a bit, although the typical functions of the BIOS are the same and properly implemented by a BIOS call (BDOS 50, I think) with an offset parameter to specify which BIOS function was called. This is very important when a disk formatting program is used on both computers. The average PCW programmer tends to use absolute addresses and thus causes a lot of trouble. Just imagine: a two-sided disk is specified by 01 (Hex) on the CPCs, where on the PCWs the value is 81 (Hex). Although this just means using the high bit, many disk tools fail to serve both computer types, although they are made by the same company.

With these thoughts, I'd like to conclude our first excursion into the AMSTRAD world (for most of you). If you find this subject interesting, I could try to shed more light on other aspects as well, so feel free to write letters to TCJ or to me directly. You may reach me through either CompuServe 100024,1545 or via GEnie: helmut@genie.geis.com would be the INTERNET version of my address there. GEnie users will probably have detected me by now as a contributing sysop for CP/M Roundtable there.

Many people from the US can hardly imagine how differently CP/M developed in the US and in Germany. This is why I want to tell a little about the backgrounds as I experienced them.

In 1982 CP/M had made its way into German offices. High-priced computers and high-priced software lived a happy, reliable (and virusfree!) existence in the holy places of industry. Before 1984, it was hard to get low-priced, quality equipment for private use. So, no wonder an active scene started to develop, following the spirit of the HEATH radio amateurs in the sense of building, soldering, and even developing hardware and software.

Of course, there were the industry standards: Apple II, Radio Shack's (Tandy) TRS-80, EPSON QX10, Wavemate Bullet, Bondwell 12/14, and Kaypro II, IV, and 10. Many others one only knew by name, since they either never made it out of the offices, or they were not in use at all in Europe. This fact shows up clearly when you get a format converter/reader like 22DISK (DOS, sorry) that contains lots and lots of formats that people in Europe never heard of. On the other hand, most of the common European formats are missing and have to be defined by hand.

A good example is the GENIE IIs, a German post-development of existing hardware, compatible to the TRS-80 and disk-compatible to the COLOR GENIE. The operating system "NEWDOS 80" was so-so, but "G-DOS", the German homebrew operating system, had exciting extra features. That gave me a hard time when I tried to convince people of the superiority of Z-System. Unfortunately, I don't remember exactly what those features were. One good thing was that the GENIE IIs uses the ECB-bus concept, giving it access to a variety of add-ons, like EPROM-cards and the like.

This brings us to the main big thing in Europe: single-board computers (sometimes called EMUFs) and ECB-bus type computers. Of course, there were always some that made up their own "standard", like the NDR-Mikrocomputer. Notice the "k" in micro. Zeds tsherman! But what is this ECB-bus like? The concept is based on a back plane with several sockets of 96 contacts each, that connects all I/O, clock, and supply wiring along its "bus". The 96 pins are divided into three rows of 32, named a, b, c. Most of the time, b is not used, thus leaving a so-called a-c wiring. The male connector slides quite well over the female on the back plane by means of a square edge and also establishes a very good mechanical connection.

This system is easy, professional, and less hay-wire than the IBM slot system still used in today's 486 clones. Maybe that is the reason why in 8-bit environments the European industry still makes extensive use of the ECB-bus system. Other bus systems, like the S100 bus, never really made it in Europe.

One of the developers for today's industrial ECB applications is Tilmann Reh, of whom we shall hear again later on.

The idea behind it all was to be able to use all sorts of different hardware on a given system. An ECB-bus EPROM-programming card could be used on every ECB-bus machine, given the correct I/O-address and software adaptations. Soon there were SASI cards, which opened doors to hard-disk and backup media, as well as IDE interfaces, disk controllers, and even complete terminal cards, suited for the Hercules monitor and IBM-type keyboard. But then, of course, there was a long way to go before that goal was reached. One of the most widespread home systems was KONITEC's PROF-80, widely featured in the c't magazine at that time. They, too, offered variants of that design and add-on cards, like the PROMMER-80, an effective hardware tool to produce all sorts of BIOS variants. Some people added an OMTI-controller and got together to write a new BIOS to integrate a hard disk. Uwe Herzceg, one of my early Z-contacts, finally succeeded. It was one of those great moments when he had a full-blown system and had managed to implement my favorite disk format (CPC VORTEX 704K) into his BIOS!

Another strong line was pushed even by television, the NDR Klein Computer. There were unfortunately at least two different bus types. One of them was the ECB bus, but the other one was good for that machine alone. There were a lot of efforts to keep the system design "open", so it takes quite some work to keep track of all the "native" formats. I have about five NDR customers, each using a different format. Great. I think, this shows best what the European CP/M situation was like.

Few companies took the pain to really do things properly, as KONTRON and a few others did. They implemented a progressive mainboard concept in a 19" rack with an optional hard disk controller board. The floppies were already in 5.25" format, and the terminal could be integrated with the machine. Such combinations with high-resolution graphics were widely used in medical environments for ultrasonic and cardiographic applications. Every now and then, one of those machines appears from nowhere. Most often it is a PSION 80, the most advanced model at that time, but using a single-density disk format, which is not supported by 22DISK on the IBM clones. The BIOS was well written, and boot-up could be interrupted to jump into a system monitor capable of assembly language instructions. The system could be configured to boot off floppy or the hard disk.

Needless to say, they used their own terminal codes ..

So, to view the whole, you can see that the European CP/M scene started out with complete machines, most of them with an integrated terminal and specialized for certain tasks, not too many of them having standard CP/M operation in mind. This created the situation I (among others) face today: first to make the real CP/M environment accessible to the later owners of those industrial dropout beasts, then bring Z-system or the like to them for their own benefit.

The weirdest situation, though, was in Eastern Germany (Wall scenario). There was only one company (Robotron) really capable of building usable machines (the state said one is

enough) and so they built hardware (they copied and stole some of the AMSTRAD techniques and a few others) and they cloned software. They made rewrites of WordStar (TP.COM), dBASE II (REDABAS.COM) and others. When the Wall fell, the company broke up, and the programs became orphaned. If you are looking for a database program, give REDABAS a try. You'd have to give it a little hack to accept .CMD files as program input, and sometimes the MODIFY COMMAND does not work properly, but the rest is fairly acceptable. I recommend using ZDE anyway, since dBASE, too, has a bug in it's editor and always trails garbage at the end of an edited program file (probably no CTRL-Z).

Naturally, as soon as I got my first contacts, I tried to implement NZ-COM on the CPA (CP/M clone) and the SCP (the other CP/M clone). The termcap turned out to be a slight disaster at first. The cursor sequence looked as if it used only ESC. After my first fright, I used a debugger to look into TP.COM (WS-clone). There I found what I had been hunting for: ESC was followed by HEX 80, which naturally was trapped by the output mask of CP/M (HEX 7F). After that, it didn't take long, and NZ-COM could be sold for the ROBOTRON machines. The disk format is quite easy and uses a logical layout. There are four different formats that I know, the most used being a 780K format (DSDD or "Quad-Density", a term somebody had made up but really doesn't exist).

This whole mixture of CP/M machines could only be dealt with by a machine with a standard controller and a very good program to handle all the formats. Of course I hear you say "22DISK on my IBM PC!". But that is not what it is all about. I mean a CP/M machine, able to do real mean tricks in sector translation, side access, skews, mixed or even high-density or the like. I kept looking. I glanced at the PROF 80 of Uwe Herzceg and felt pretty poor with my AMSTRAD CPC, even though I could read some formats.

Then Uwe gave my name to a guy who was looking for an assembler for the work on a new processor, the Z280. So this guy, Tilmann Reh, called me up. After some telephone calls, we were in good contact, and the more I learned about his plans, the more interested I got. When he announced his new single-board ECB-bus computer CPU280 and told me about the format manager, I packed my stuff and traveled to the next meeting he held in his house. Imagine, a living CP/M machine constructor and supporter! I immediately ordered my "machine". So, after some weeks, I got a parcel containing wonderful shiny little capacitors, resistors, carefully packed chips, most of them in conventional form, some in new square housings, one of them being the legendary Z280. Tilmann had never before in his life tended to swear and curse, but talking about his experience with the sloppy development of the Z280 and all it's mask bugs, he had to add a few words to his vocabulary. After two years, he still has problems to say the name ZILOG in a cool, unaffected manner.

Anyway, after four hours of concentrated soldering, and three hours of setting up my AMPEX terminal, I got the thing going! It worked like a charm, but soon, after seeing an increase in the need for high-density drives, I had to exchange them. Nowadays, I am still using four drives (I could use 8"-drives, but I don't need to), two 5.25" HD drives and a 3.5" HD drive, plus an optional 3" Hitachi drive, that allows me to work with AMSTRAD disks as well. The 3" drive is not needed constantly, so it's connected externally. That was the time when many ECB bus machines were torn apart, leaving only the drives and the power supply. Some (the PROF80 users) had an intelligent terminal card, the GRIP card, that they also left in there, and some even had EPROMers and static, battery-backed RAM-disks.

As the CPU280 community grew, Tilmann was encouraged to offer add-ons, aside from his software improvements and BIOS updates. So, not long after our start-up, the IDE controller card was presented to us. This was the final thing I had been waiting for: 80 MB right at hand from CP/M! I must say, whenever one of my two IBM PC clones let me down, I looked at the CPU280 with a grin and said: "good to have you, I don't really need the bastards, ey?" Well, it is partly true. One of them became the home of the reissue of ZNODE #51, the only CP/M and ZCPR bulletin board and file source in Germany accessible for free, no download charges, cross-system discussions, local programmers forum, and of course, all that Z.

Let's hope I come up with a good idea for my next article. My little baby, YANNICK, is taking lots of daddy's time and attention.

Regards and cu

Helmut Jungkunz

(P.S: I'd be glad to receive your comments on my articles, best through e-mail, since this creates a computer file that can be passed to others.)

TCJ Center Fold

Special Feature

All Users

Jupiter ACE

Some time back I had mentioned wanting to use a Jupiter Ace. This is the ZX81 variation that ran Forth from ROM. I saw several versions when they were produced, but considered their price a bit steep for me. The unit comes with a very good book by Steven Vickers Ph.D. that gives you all the Forth information needed to write programs on the ACE. The book explains that ACE stands for Automatic Computing Engine. The back cover states that: "After Studying mathematics for eight years, first at King's College, Cambridge and then at Leeds University where he gained a PhD in algebra, Steven Vickers turned to computer programming and played a large part in developing the Sinclair ZX81 and ZX Spectrum."

"In 1980 he read The Hitchhiker's Guide to the Galaxy and, realising that its high content of factual errors made it unreliable for practical use, decided to rewrite it. The result was what is by now probably the best-selling computer manual ever, for the ZX81. He also wrote the manual for the ZX Spectrum."

"In 1982 he and Richard Altwasser set up Jupiter Cantab Ltd and designed the Jupiter Ace." That sounds like great credentials to me. There is also a chapter in the back of the book that gives a hardware example of using the ACE to control stop lights. The pinout of the expansion interface is given, plus indicating that it is compatible with the ZX81.

I received a bare board and ROM from Donald A. Gaubatz. We have been e-mailing and talking about getting the source code for the ROM's. Donald became acquainted with the Jupiter Ace while he was doing a Ph.D. at Cambridge University Computer Laboratory.

In a letter to me, Donald states "Jupiter Cantab Ltd. stopped business operations in late 1983 and the product was taken over by a local computer retailer. I bought a Jupiter ACE, games, etc., for my children to play with, and, subsequently acquired some hardware and firmware items when the retailer also stopped selling the product. I purchased these things with the goal of modifying the design, specifically with regards to the cassette tape interface."

He continues, "The firmware disks I have are 8" hard-sectored, as used by the ZILOG MCZ Development System. I also acquired a ZILOG MCZ-1 (again, repatriated from England), but it does not seem to work. Presumably, these disks contain the source for the ACE ROMs." I am aware that several of *TCJ*'s readers have these old development systems, possibly one still working, and we need to see if we can get one to Donald in Massachusetts (use dag@world.std.com).

As you review the design of the ACE you can see why it makes a good design for training. The design is simple, without any special chips. The keyboard could be built using 40 simple push buttons, although I feel that a serial port might be more appropriate for getting data into and out of the machine. I dare say most user would have all the parts sitting in their junk boxes waiting for this design.

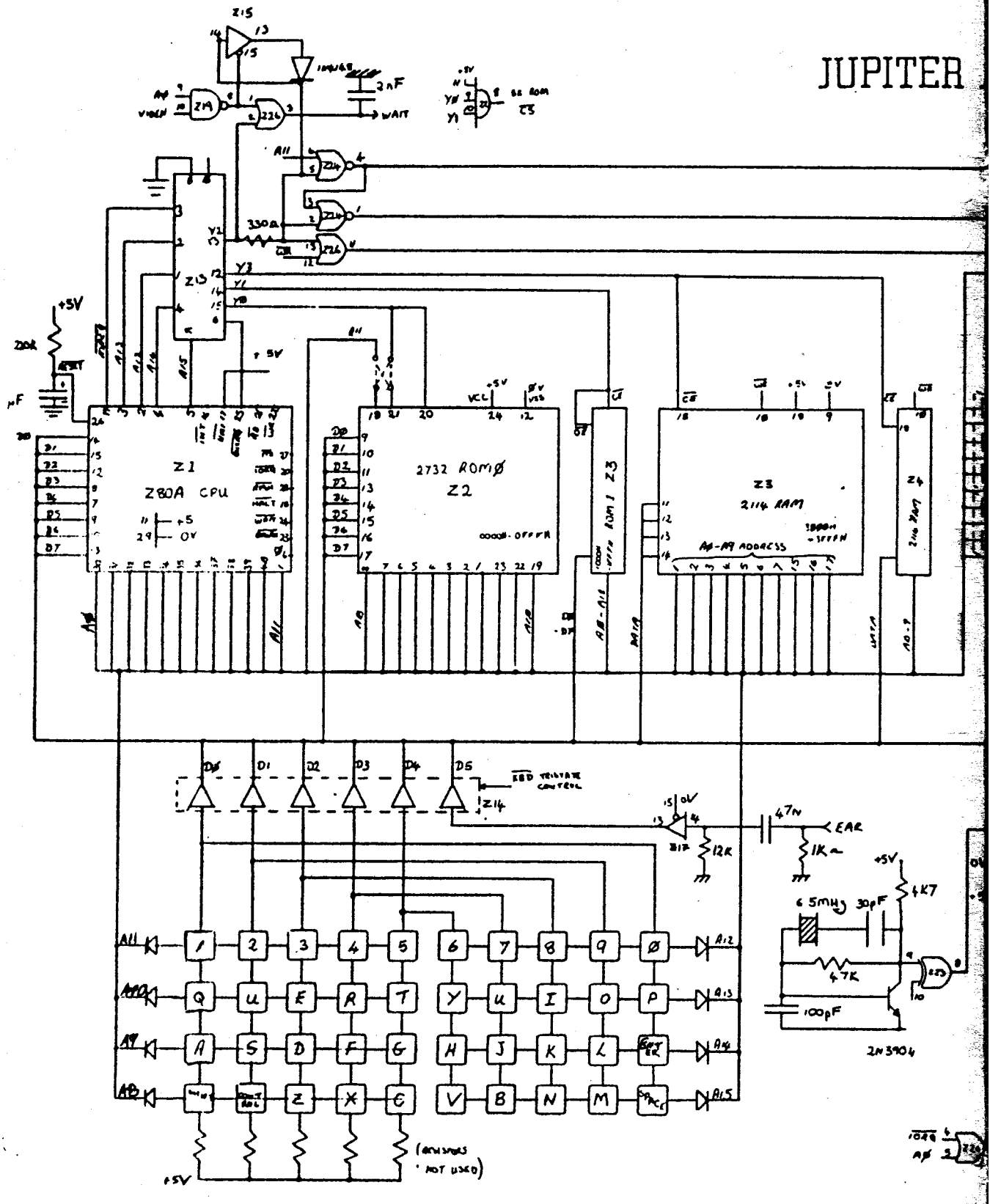
To help understand the design you need to check out chapter 25 and 26 of the book. The RAM and ROM addresses are barely decoded and as such appear in many places in the memory map. In the early days of micros, that was a very common design, after all few could afford lots of memory. The problems came later as people tried to expand the memory. Basically the address decode section needs to be redone. As the design now stands, address lines 10 and 11 are not decoded! That means all devices get decoded four times in the map.

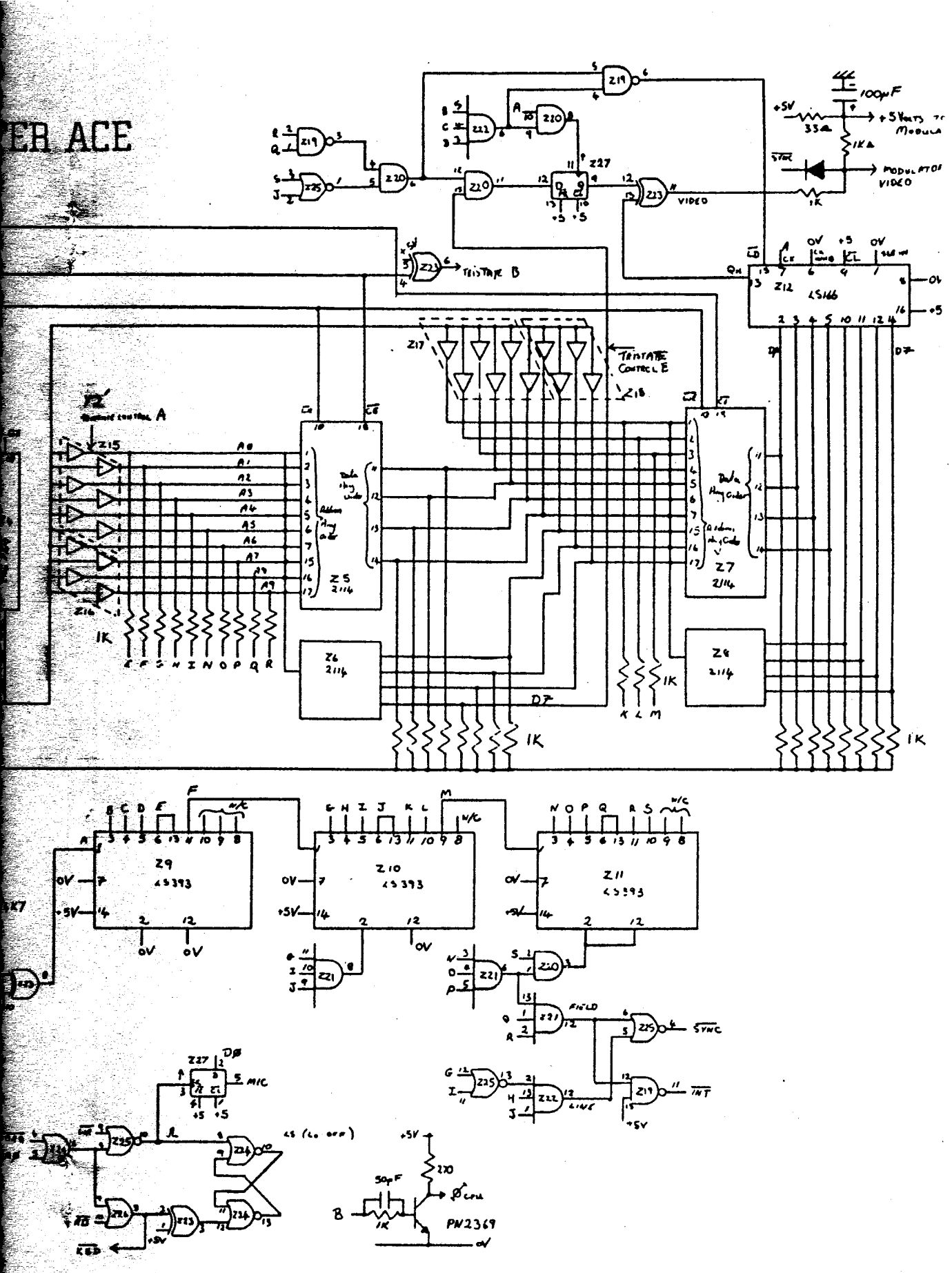
There are a few interesting chips that need to be pointed out to the novice designer. The LS166 shift register provides the means of shifting the video bits into the video stream. Characters are changed from ASCII values to bit mapped representations and stored in the video RAM. The bit mapped values are then read from RAM in parallel into the shift register that converts the data serially into the video stream.

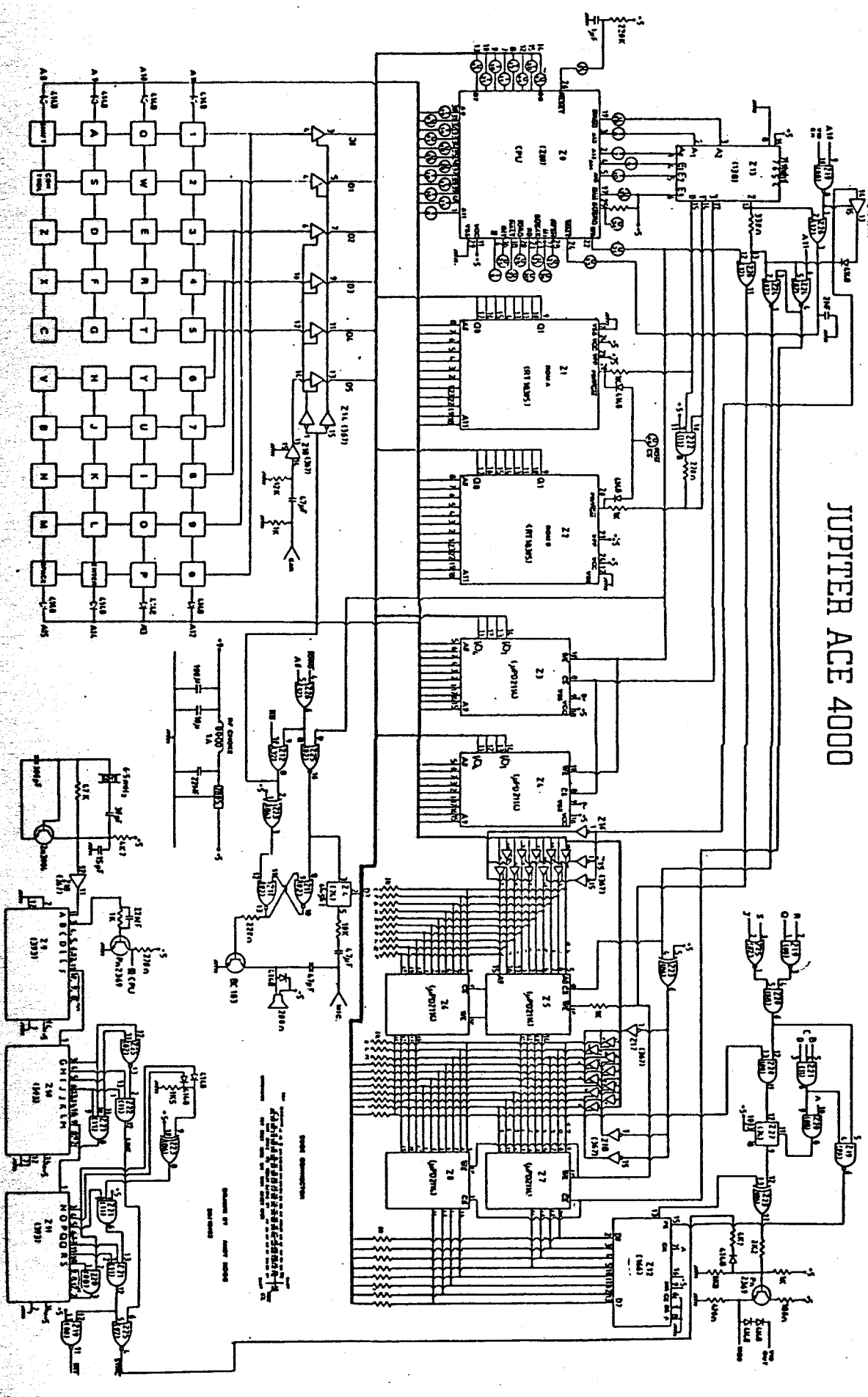
The LS393 are ripple counters that are used as clock dividers. That means that these devices control the timing of when things happen in the machine. We start out with a 6.5 Mhz clock and divide by a half. That gives the 3.25 Mhz clock to a transistor that inverts the signal to drive the Z80 CPU. It is a very simple way of getting a number of signals to drive the system and maintain proper relationships. If something is to take place every four clock cycles, dividing the clock by 4 will give a pulse at just the right time.

Since the frequency of TV sets in Europe is different than in the USA, other clock frequencies where used. The European version of the Jupiter Ace is shown on page 28.

I hope you understand the simplistic nature of this design and see how a modern day version could be constructed.


JUPITER
Z1
Z80A CPU
2732 ROMØ
Z2
Z3
2114 RAM
Z4
Z13
Z14
+5V
WAIT
EAR
4K7
1K
12K
47N
100pF
2N3904
6.5MHz
30pF
4.7K
(UNUSED)
* NOT USED


Z19
Z20
Z22
Z21
Z27
Z23
VIDEO
100μF
+5V
33Ω
1KΩ
MODULATOR VIDEO
Z24
TRISTATE B
Z12
LS166
Z17
TRISTATE CONTROL B
Z18
Z15
A0
A1
A2
A3
A4
A5
A6
A7
A9
Z16
1K
Z5
2114
Z6
2114
Z7
2114
Z8
2114
DZ
K L M
Z9
LS393
Z10
LS393
Z11
LS393
Z21
Z20
Z21
Z19
FIELD
SYNC
Z25
Z22
LINE
INT
Z27
MIC
Z25
Z24
Z26
Z23
Z24
50pF
220
PN2369


JUPITER ACE 4000

PC/XT Corner

Guest: V. Henry Vinerts

Regular Feature

Intermediate Users

A Stepper Motor Project

While at The Embedded Systems Conference, I found a copy of Frank Sergeants system running and stepping. That is to say a real life size version of the article. The booth was Silicon Valley Forth Interest Groups and Dr. Ting and V. Henry Vinerts were maning it. I talked to Dr. Ting for a minute, then settled in for a who, what, and how of Frank's design. For those who have seen me in person, I have rightfully earned a reputation for arm twisting in hopes of getting an article. Well Henry's arm must still be intact since he sent me these drawings and explanations. BDK.

October 10, 1994
Dear Bill,

Just received your note on GEnie, and this writing comes at your suggestion, with a copy to Frank Sergeant. As I said before, I was glad to see you stop by the Forth booth at the Embedded Systems Conference in Santa Clara, a few weeks ago; and thanks for the request for a blurb about my stepper-motor demos.

There were two: the Etch-a-Sketch and the Peanuts Gang. The first featured two stepper motors connected with rubber bands to a medium-sized Etch-a-Sketch screen from the local toy store; the second- Lucy, Charlie and Snoopy doing dances on turntables mounted on stepper motors. The controls were by a PC laptop through the parallel port with software provided by Frank Sergeant's Pygmy Forth. So far we still haven't drawn the perfect circle on the Etch-a-Sketch, nor multi-tasked the dance routines with music, but I am working on it. Some pictures are still in the camera, but, I think, my sketches from the project may be of more interest to your readers. (I am sending the basic controller sketch by U.S. mail.)

The idea for the Etch-a-Sketch came to me some time ago, but I probably wouldn't have gotten very far without Frank's recent article in the PC/XT Corner. The main message I want to convey here is that I feel good about actually having built something that combines elementary hardware and software, that I _almost_ understand and can control with my own will (and that is unusual in this hi-tech world of ours). It means much more to me to do something simple well, than to dabble in and fumble with complex systems whose inner makings are guarded by the self-elected few who may have created the complexity in the first place.

Yet, a task well done looks simple, a problem once solved, is easy. To get to this point with the steppers, I've had to learn quite a bit. It would take too long to go over all of that here, but let me list a few thoughts and a few questions—they may germinate some material that belongs to the "raison d'etre" of TCJ.

There must be more variations of stepper motors than there are dialects of Forth. The number of wires, steps, voltages—all can vary, but the first thing is to recognize whether a stepper is bipolar or unipolar (bifilar), that is, whether it needs a switching power supply or a regular unipolar power supply. (There are enough mystery words right here in one sentence. How about "EFSS" region of a stepper, that Charlie has to watch in order to stay in synch with Lucie? You see what I mean by getting to the inner secrets of the state of the art? If we need to search further to get the job done, there are books and more books.) My first stepper was bipolar and would not work with the scheme that Frank described. The present steppers (at $3.50 each, from a surplus store) are Airpax specials, and I was able to get a nice catalog-"Stepper Motor Handbook"-from Airpax in Cheshire, CT, that helped a lot. So, get to know your stepper!

Next, whatever you wish to turn, will require a little engineering. Engineering means numbers. "Let me make some numbers" means that an engineer wants to turn over the napkin or the envelope, doodle, scribble and calculate a bit and come up with a prediction of whether something will or will not work as Mother Nature intends it to be. Of course, the decimal point must be in the right place, though. Now, in my case, I started out without making any numbers and wound up "cooking" some chips. I still don't know what defines a power transistor, but the TIP120 Darlingtons from Radio Shack can handle the 1 amp at 5 or more volts going through my steppers.

The trouble is that all that juice still wouldn't work the Etch-a-Sketch. This is where I remembered about the "numbers". Turned out that it took 32 U.S. quarters in a bag tied to a string that hung from a 1 11/16" dia. homemade pulley (from closet-pole sockets) to turn one shaft of the Etch-a-Sketch. The kitchen scale told me that 40 quarters weigh a half-a-pound (that means you can buy a pound of quarters for $20), so the torque needed to work the sketch worked out to be about 5.44 ounce-inches. The stepper motor did not like that at higher speeds, so I had to equip it with a small pulley—about 11/16"

dia. screen door roller—in order to get a workwise transmission. For right now, various combinations of rubber bands have been tried, but they all create errors in motion reversal from slack to stretch. It is pretty hard to program a perfect circle, but I must say that my 12-sided figure program produces the same inaccurate tracings with amazingly reliable repeatability. (I am shopping for a set of plastic sprockets and chains, but my interest dwindles. Control is possible, corrections can be made with software. Q.E.D.)

Last, but not least, as I work on adding music to the dancers, I find that my Toshiba T1000SE laptop behaves differently than my XT or AT as far as the 8253 chip features are concerned. I cannot get the sound to continue. Is there a different chip on the Toshiba? Also, if there are any unhappy Toshiba T1000SE owners out there, three of us here in the Silicon Valley Forth Interest Group have developed an inexpensive battery pack renovation procedure. Would it be worth writing about?

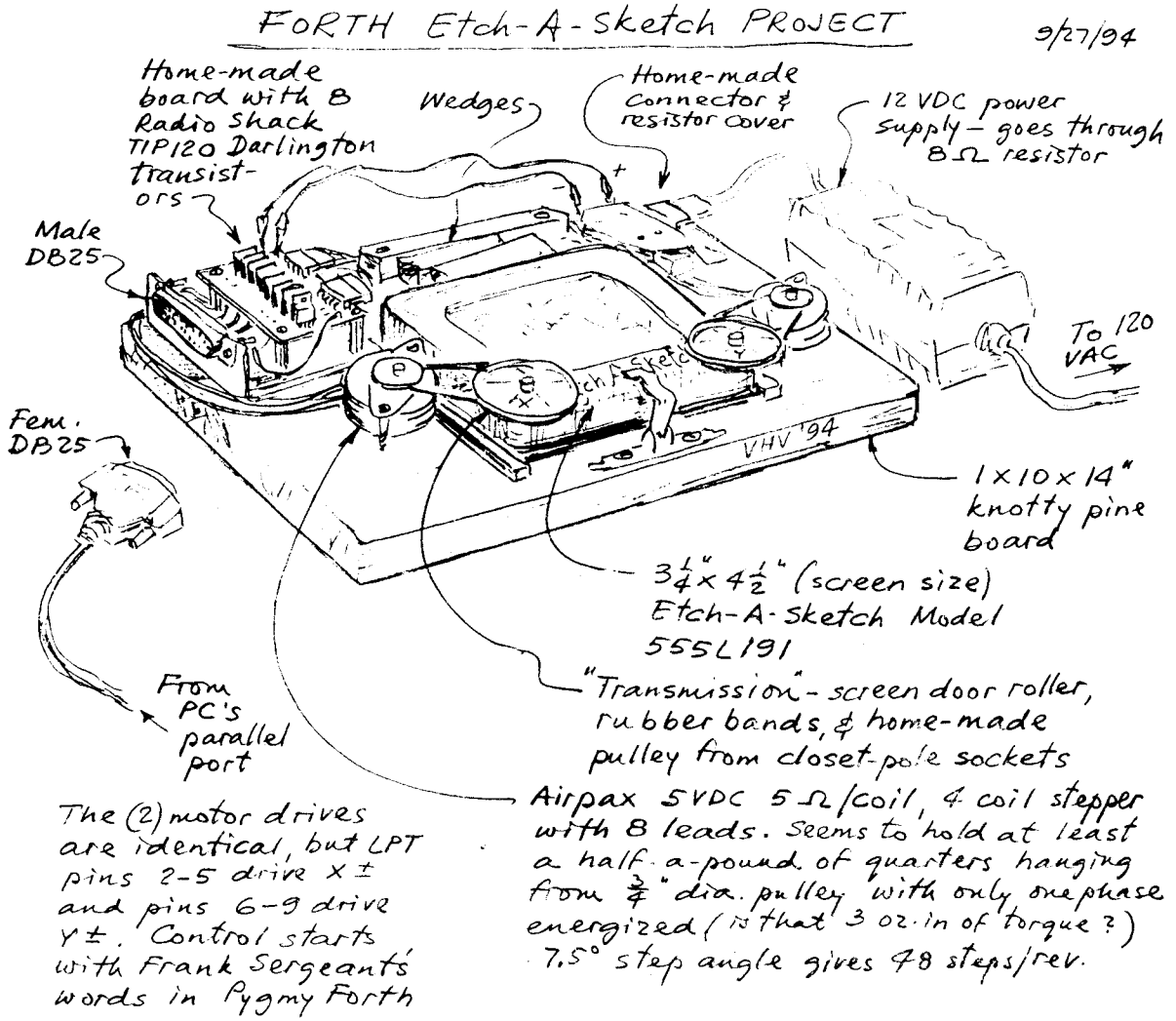
So much for now. Sincerely, V.Henry Vinerts, 36139 Chelsea Drive, Newark, CA 94560.

Thanks Henry! I especially like the way you found out how much energy was needed to turn the wheels, very novel idea!

Yes, it is important to sometimes crunch those numbers. I must admit to sometimes being lazy and not doing the math, but then experience can sometimes give us rules of thumb that work just as well. I usually like to get my steppers from working machines, so I can compare and see what circuits and pulley sizes were used. This gives me an idea about where to start, and if I need more force than they did - change wheel sizes accordingly.

You really didn't explain about your major problem, which I hope you've solved. When doing simple projects like this (well simple results - but complex research work) it is hard to justify spending money on real notched belts. As you explained to me, what is needed is real drive belts that can be tightened and have enough "traction" that slippage goes away. Frank's desire of an XY table has got him into considering real worm drives in which the slippage is minimal. The slippage we speak of, is the loss of movement that often occurs when you change direction.

Slippage is always a problem, but very expensive drive mechanisms that have small amounts of slip, explain the cost that prohibits Frank from buying them. The best are worm gears and mating collars that have been machined such that the gap between parts amounts to a good film of oil. Less accurate but with more acceptable slip, are cogged or toothed belts. Slip-



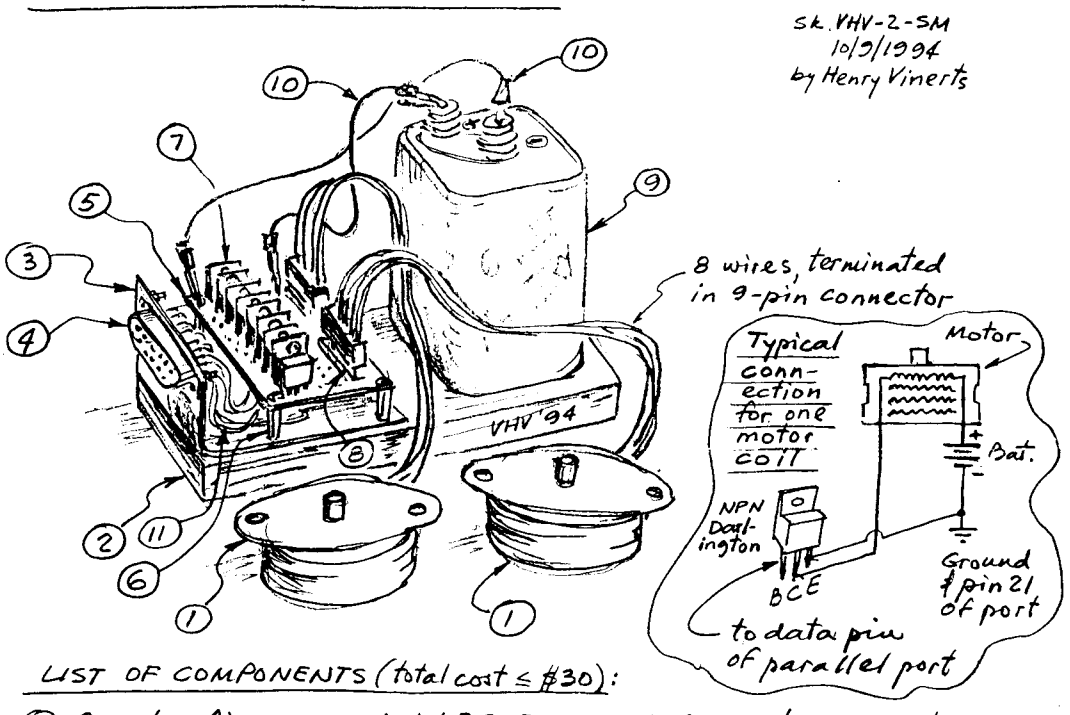
page occurs here from stretching in the belts, loose belts that allow the cogs to slide a bit out of the tooth of the gear. Backlash describes the slippage (and caused Henry his big problem) when directional changes may waste a full step or more. If you have a little slip in one direction, changing direction will normally produce twice that amount of slip.

There are some trick one can use to mitigate slippage. Adding drag or friction so that the drive mechanism is under a constant pressure often helps prevent overshooting. Too much drag can however cause stretching of the belt as the force needed to get moving is greater. Spring loaded idlers (often seen on car fan belts) can cut slippage in both directions by adjusting the tension automatically as the belt moves. This adjusting keeps the "slop" to a minimum without adding to the drag, even though the results is about the same or better - a constant tension on the belt in both directions.

I do not know of a beginners level engineering book that would not require 4 years of college to understand that might help our readers master the mechanical side of steppers. If you Henry, or any reader know of one, please let me know. Just remember that plenty of inventions have been created by non-professionals using tools as simple as a bag of coins.

Thanks Henry for your great project. Bill Kibler

NON-ARTISTS SKETCH OF PC-PARALLEL-PORT STEPPER MOTOR CONTROLLER:



LIST OF COMPONENTS (total cost ≤ $30):

Total cost:			
7.00	①	2 each	Airpax model LB82731-M1 bifilar stepper motor, 5 VDC, 5Ω/coil, 7.5° step angle (from el. surplus store)
	②	1 "	½" plywood x 3¼" x 6", or equiv.
0.49	③	1 "	3"x4" bent sheet metal from TECO truss tie plate, or eq., with 3 wood screws
1.49	④	1 "	DB25 male submin. solder-type connector, Radio Shack 276-1547B, with 2 suitable mounting screws
1.19	⑤	1 "	1.85"x2.8" gen. purpose component PC board, Radio Shack 276-149A
1.29	⑥	4 "	13/16" (20mm) high PCB standoff, RS 276-195A
10.32	⑦	8 "	Archer (Radio Shack) TIP120 Darlington NPN transistor
0.20	⑧	2 "	9 (1 unused) pin male connector, modified for soldering to ⑤. (From surplus electronics store)
3.99	⑨	1 "	6-volt lantern battery or equiv. (wears down to 5 volts or less soon, if any coils are left energized.)
0.75	⑩	2 "	12" ± jumper, 22 AWG
0.10	⑪	17 "	1½" to 2" lengths of stranded 22 AWG hook-up wire
—	⑫	(not shown)	Bare ground wire, solder, lots of patience
26.82			

32-Bit Systems

All Readers

CD-ROMs and News

Real Computing

By Rick Rodman

Recordable CD-ROM, aka CD-R

Let's talk cents per media megabyte. Hard drives are going for around 80 cents per megabyte, down to about 50 cents for high-capacity drives. Syquest and Bernoulli are around the same price. Floppy diskettes are about 30 cents per megabyte. Optical (WORM or MO) cartridges can be as low as 10 cents per megabyte. Tape is around 8 cents per megabyte.

Recordable CD-ROM media sells for *2 to 3 cents* per megabyte.

Not only is CD-R cheap, read-only drives are plentiful and cheap. By contrast, optical and tape drives are expensive and involve device drivers and/or driver programs. A more important factor is the ISO-9660 standard: You can write a CD on a PC, and read it on a Mac, a Sun, a VAX, or anything else.

There are drawbacks - you have to write a whole disk at a time, the capacity is small for some applications, and CD-R disks are fragile, with media life uncertain - but the cost and standardization advantages are so overwhelming that we can expect CD-R technology to become the most widespread data-publishing method used for the next ten years.

Okay, you've read all that before, and this is TCJ, so let's stop eating hush puppies and get on to the flounder.

Writing your own CDs is easy to do. Finding enough data to fill a CD may be a little harder. To write a CD, you need a fast computer with a fast, large, hard drive. A Pentium PC with a 1GB SCSI drive is a good choice. Second, you should *probably* try to have a dedicated SCSI adapter for the writer. No scrimping here - I suggest an Adaptec 1540 or 1740 (for EISA machines).

Then you need a writer. These drives all have very differentcharacteristics. Some are faster, some are smaller, some are cheaper. The new Yamaha 4x speed drive is the fastest and the smallest, but it costs about $5,000 and, being faster, will require a very fast machine to write to it. Mine is a Ricoh RS-9200, which is a single-speed drive, small, and inexpensive - about $2,500 and falling.

Of course, you need software. And there are lots of choices. The software that comes bundled with the drive may or may not be the best for you. There are lots of decisions to make - Do you want Orange Book? Do you want multisession? Do you want CD-DA? Do you want Rock Ridge? For my purposes, I only wanted the basic ISO-9660, to write plain single-session CD-ROMs, for backup and for data.

The software that was bundled with my drive is Incat Systems' Easy-CD-Pro. This is a Windows package, with a slick drag-and-drop graphical interface, buttons to click, and a nice manual. Very easy to use - just drag a directory and drop it on a "virtual CD". Then write when you're finished. Drag, drop - and wait. Drag, drop - and wait.

For some people, this may be fine. For me, it was tedious. Each time you drop a directory, you have to wait *several minutes* before dropping another one. Graphical interfaces become annoying when you're looking at hourglasses for long periods of time. Building the "virtual CD" can take hours - and it has to be done every time you write a CD.

The most annoying part of Windows-type graphical programs is their invisibility. You can never tell what's going on. Your choices are always severely limited, so there's never any way to ask anything. For example, several times I got a message "Servo error" with a single "Ok" button. What does this mean? What should I do about it? Not only is there nowhere to look, but there's no option at all - just grunt and click "Ok".

There's an old saying that free advice is worth every penny you paid for it, and I feel the same way about some free software. I'd rather not get any software with my drive, and use the money saved by a lower price to go buy software I want. In my case, the software I bought was Corel SCSI version 2.

Corel SCSI has a simple program called CDCOPY that makes writing a CD as simple as using XCOPY. They also have an XCOMP program that lets you verify an entire CD, producing a log file of errors. Better yet, CDCOPY lets you use a "CFG" file to write a list of directories to a CD. Best of all, they let you put *comments* in the file so you can keep it as a documented "source listing" of the CD backup operation!

I about fell out of my chair when I read that you could add comments. This is a mark of a good programmer somewhere at Corel. If you want to tell a real programmer from a rookie, look at their comments - real programmers write comments nearly every line; rookie programmers go without comments for pages, maybe even whole files.

As an aside, I had to chuckle when I read in Mr. Anderson's column that someone thought of 100 bytes as a large

program. Program size is usually measured in source lines, and the common criterion is that less than 10,000 lines or less is a small program, 10,000-99,999 a medium size program, and 100,000 and up a large program. Assembly-language programs seldom go into the large range. As for myself, I am on only the second large program in my career, and both have been in C.

Some folks will tell you that drive speed is the main criterion for selecting a CD writer. I'm here to tell you, it isn't. Philips says that their CDD-521 can write a CD in half an hour, whereas the Ricoh takes a whole hour. By this logic, the Yamaha could write a CD in fifteen minutes. *Don't believe it!* There is so much overhead in writing a CD that writer speed is almost irrelevant. You can count on writing a CD taking at least two hours, and of course you have to verify the CD, which actually takes even longer. With Corel SCSI, I can do the verify in a double-speed read-only drive. You can figure on about four hours total to do a CD. The Yamaha might get it down to three, but I wouldn't count on it.

Think about it... in 1981, companies paid $5,000 for XTs which are being tossed out today. For $5,000 you could set up a PC for them with a 2-gig SCSI hard drive and a CD writer. You supply some batch files that xcopy their data from their fileserver, which they would run on Friday evening. As a backup system, it's even cheaper than tape, since they wouldn't need an off-site tape drive to read backups in the event of fire - just a PC with a CD-ROM drive. And who said they could only use it for backups? Look at all of the other possibilities it'd open up.

A mammoth change is taking place in the computer world before your eyes. If you can't profit from it, at least be ready for it.

Cabinets

I don't like the Philips because, although it's only a little more expensive than the Ricoh, it's a lot more *expansive*. It's as large as an old AT, with only a few little lights on its big, blank, tan face.

This is the way computers are today - big, blank tan boxes made of cheap-looking plastic. When will these guys ever learn? Some PC makers nowadays are starting to put little "louvres" or subtle indentations on their boxes, and some are coming out with all-black cabinets instead of ubiquitous tan, but these are only very minor changes in a packaging trend which can only be characterized as dull, duller, dullest.

Nominees for worst cabinets of all time are: Dell, for its plastic PC chassis which has to be warped and twisted to line up the screws, which strip out readily; Compaq, for its Systempro chassis, which, once opened, is almost impossible to close without a hammer; and Zenith, for its PC chassis which, though metal, still require bending and twisting to get the screws on its side (echh!) to line up. Nominations are still open, of course. Let us share your horror stories.

Best computer cabinet of all time? No contest here. Undisputed winner - the Imsai 8080. A pleasure to look upon, with all of its LEDs and switches - a cabinet that says, Here is a real computer.

Dull tan boxes just show the infancy of computer technology. In the more mature world of stereos, where interfaces are standardized and controls mostly agreed upon, cabinets have been an area of real design innovation for at least thirty years. Set an expensive PC next to a cheap stereo. See what I mean?

Here's an idea for an entrepreneur out there: Make a display box which fits in a drive bay of a PC. My plan was to put a circuit board on a 3.5-to-5 drive adaptor. This circuit board would be connected by a ribbon cable to a board plugged into the backplane. The front would be smoke plexiglas, behind which was a dazzling array of LEDs in various colors and of various types. I'd like an array of small LEDs showing the address bus, latched by certain bus conditions. Some LEDs would be under program control. This thing would not only be neat to look at, but would actually be *useful* for program debugging. And it should be cheap and easy to build, too.

Tiny-TCP

"Enclosed is the documented vector usage for the Xerox 820-II," writes Duane Ruck in response to my queries in #68 regarding the Xerox 820. "It looks like the first 8 should be just the thing for your networking code." Enclosed were excerpts of a monitor listing showing interrupt vectors. The listings also show the keyboard FIFO and the time-of-day locations. Thanks for the excellent information!

I'd also like to thank Mr. Kaypro, Chuck Stafford, for his quality documentation on the Kaypro 10. This is one of the main reasons that older machines have kept their value: Excellent documentation is available. By contrast, PC makers guard technical information as though it were family jewels.

I have little to report myself. In trying to use Eco-C on the Z-80, I've had lots of problems. Jay Sage has provided me a copy of Hi-Tech C, which looks pretty good. I'll end up either using this or using BDS C 1.6. I've been doing a lot of compiling on a Pentium PC using Z80MU, rather than on the Z-80 itself. According to Z80MU, my P90 is emulating a Z-80 running at 52 MHz!

Next time

We get back to the Linux scene and review happenings there. Then it's on to TCP/IP land for a discussion of routing, network addresses, and subnets. Time and space permitting, from there it's networking, NFS, CD-ROM sharing, and other neat stuff. Until then: Compute happy. Compute hard. Compute proud.

Where to call or write

Real Computing BBS or Fax: +1 703 330 9049, E-mail: rickr@aib.com
Mail: 8329 Ivy Glen Court, Manassas VA 22110

Corel SCSI version 2, $99 at a store near you. Corel Corporation, 1600 Carling Ave., Ottawa, Ontario K1Z 8R7 Canada

Regular Feature

68xx/68xxx Support

6809 Assembly & Flex

Small System Support

By Ronald W. Anderson

A Correction (or a better way)

Two columns ago, I presented a short utility to set a printer to a different print mode, specifically my Citizen to the 12 cpi mode. I showed a really dumb C program that output a character at a time, and then realized that I could output the commands as a string, but drew a blank on outputting a null (ASCII 0) in the middle of a string since C uses the null as a string terminator. I wrote my own output routine (true, it was only one line and was a way to get the job done). It has since occurred to me that I could have used the "escape character" \0 in my string and C's fputs() function would handle that just fine. I just tested that out, and it worked. I'll import the C source file here:

```
// set citizen to draft 12 pitch hi density
#include <stdio.h>
#define ESC 0x1b

FILE *printer;
char codes[] = {0x12,ESC,0x4d,ESC,0x78,'\0',ESC,0x7e,0x42,0x01};

void main()
        {
                printer = fopen("lpt1","w");
                fputs(codes,printer);
                fclose(printer);
                exit(0);
        }
```

This version is 9053 bytes long, actually a few bytes longer than my previous effort, but it is at least aesthetically more pleasing to use a feature of C that lets me use one of the standard library functions to advantage.

I think someone ought to write a book "C for Dummies" to go along with the standard "DOS for Dummies". I've been using a comfortable subset of C for quite a while, but recently I decided to at least begin to explore some of the more "advanced" features. The language PL/9 that I have been using for so long to program 6809 systems is excellent as far as it goes, but there is no provision for data types more complicated than singly dimensioned arrays. I got comfortable with that level of language and when I got into Pascal and C, I didn't stretch my capabilities immediately by taking advantage of multi-dimensioned arrays and structures or records as they are called in Pascal.

While I was looking at Modula 2 briefly, I found a use for the Union in C or what is called an "alias" in PL/9. That is a way to have two different data types occupy the same memory space under different names (as alias implies). This is valuable when you want to manipulate, say, a float type variable byte by byte, for example to change the exponent for a square root routine etc.

My project for the near future is to become as conversant with the more advanced features of C as is possible, and to take advantage of them to write clearer and shorter programs. I'll try to demonstrate progress in future discussions. Meanwhile, you experienced 'C' programmers please bear with me for a while.

6809 Assembler Part 2

Last time we did a first program in Assembler to clear the screen of a terminal. What might we do for an encore? There is a fairly easy program that we can do that shows a few more of the capabilities of calls to FLEX routines. What I have in mind is a program to calculate and output the sum of two numbers included on the command line. For now, of course, we will use integer numbers. "Great", you say. "A program to add 2 and 2". That's right, but in the process we'll look at a few more assembler instructions and FLEX calls. Once more we'll take it a line at a time and then put it all together.

We need a little background first. There is an assembler directive RMB. RMB means Reserve Memory Byte(s). It takes one operand or argument, the number of bytes to reserve, and it usually is preceded by a label. In our program we are going to RMB 2 bytes with the label NUMBER. We have set aside two bytes of storage in memory to hold a 16 bit number. This is essentially what a higher level language does when you define a variable. We have therefore set aside space for the variable NUMBER. The label NUMBER is equal to the address of the variable NUMBER, and the contents of the two bytes at that address is the contents of the variable. This might make more sense when we look at the assembler output.

The X register of the 6809 is a 16 bit register usually used as a "pointer", generally called an "Index Register". That is, it frequently contains the address of a variable. We use it once as such in this program, and once simply as a place to store a 16 bit value. Here is the program:

```
* TEST ADD 2 NUMBERS ON COMMAND LINE
```

Default origin is $0000, which is OK for this program since we don't classify it as a FLEX utility.

```
OUTDEC EQU    $CD39
```

This is another FLEX routine. It outputs a 16 bit value (as a decimal number) that is "pointed at" by the X index register. More about this below.

```
INDEC  EQU    $CD48
```

FLEX routine to get a number on the command line (ASCII) and translate it to a 16 bit binary number and return it in the X register. We note here that X is designed as an index register but it can just as well hold a value that is being used in the program.

```
PCRLF  EQU    $CD24
```

This FLEX routine outputs a carriage return and linefeed to the terminal. (Print Carriage Return - LineFeed).

```
WARMS  EQU    $CD03
```

We saw this one last time. It is the JMP address to return to FLEX when the program is completed. (WARM Start).

```
NUMBER RMB    2
```

The RMB directive stands for Reserve Memory Byte. It sets aside two bytes in memory at the current address in the program counter.

Since we didn't specify an ORG, that will be $0000. Since this is not a FLEX utility that is as good an address as any for our program at least for the time being. Note that RMB sets aside memory but it does nothing to initialize the contents of that memory. NUMBER is a label that takes the value of the address of this variable, in this case $0000.

```
START  JSR    INDEC
```

This gets a decimal number on the command line and moves the command line pointer along to the next separator (space, comma, or CR). The number is translated from it's ASCII representation on the command line into a binary 16 bit number, and when subroutine INDEC returns, the value is in the X register. This program has two labels. START has the value $0002, the address of this first instruction of the program. See the assembler output listing later.

```
       STX    NUMBER
```

STX stores the value contained in the X register in the memory location NUMBER (or we might say in the variable called NUMBER). Note the lack of an # sign here. We have an instance of an addressing mode we hadn't discussed. It might be less confusing if I didn't bring up addressing modes just now, but indulge me a little. We'll have a summary at the end of this. At any rate, this instruction stores the contents of X at the memory location NUMBER that we reserved with the RMB instruction above.

```
       JSR    INDEC
```

If we are going to add two numbers we need two of them to work on, so we get the second from the command line. Incidentally since these are 16 bit binary numbers, the values given on the command line can't exceed 32,767. Their sum can't exceed this value either. (Actually, INDEC can only handle positive numbers as well).

```
       TFR    X,D
```

We have one of the values in the memory locations labeled NUMBER and the second in the X register. We transfer it to the D register. Recall from a few times ago that accumulator D is the concatenation of the A and B accumulators, A being the high order byte. After this instruction the second argument or value is in the D register.

```
       ADDD   NUMBER
```

Ah! Finally we can add the two values. This again is an extended address mode instruction. It says to add the contents of the variable (the memory with the label) NUMBER to the contents of the D register. (You are not seeing triple, the extra D is not an error. There is an ADDA and an ADDB instruction for the A and B accumulators also. (The index registers can't be used for this operation. There is no ADDX instruction, so we had to do the TFR X,D first). The 6809 is very regular in it's instructions. The results of any operation like this where a value in memory is operated upon by or with a value in a register, always ends up in the register. That is, the sum of the D register contents and the contents of the memory location(s) NUMBER is now in the D register.

```
       STD    NUMBER
```

Now we store the sum back in the variable NUMBER, since we don't need the previous contents.

```
       JSR    PCRLF
```

This outputs a CRLF to the terminal and gets us off of the command line so we can print the result on a line by itself.

```
       LDX    #NUMBER
```

Now we point X at the variable NUMBER. We have yet another addressing mode. The X register is an index register. We placed the ADDRESS of the variable NUMBER in it with the above instruction. Remember we said that NUMBER is a label that is equated to the address that was RMB'd above. The # sign here does mean "immediate".

```
        JSR     OUTDEC
```

Now with X "pointing at" NUMBER, we call the FLEX routine that translates the 16 bit value pointed at by X to an ASCII string and prints it to the terminal. (OUT DECimal).

```
        JSR     PCRLF
```

Let's put the result on a line by itself by doing another CRLF.

```
        JMP     WARMS
```

Return to FLEX

```
        END     START
```

Housekeeping telling the assembler that the program starts at the label START.

How do we run the program after it is assembled? Supposing that the working drive is 1, the assembler default output file extension is .BIN so we run the program with the command:

```
1.ADD.BIN 123 456 <cr>
```

The result is printed on the next line, 579. You can copy ADD.BIN to your System drive and rename it ADD.CMD. Then the program can be run with the simple command ADD 123 456 <cr>. Rather than just printing the listing without the explanation between every line, I'll show the assembler output:

```
* TEST ADD 2 NUMBERS ON COMMAND LINE

CD39  OUTDEC   EQU   $CD39
CD48  INDEC    EQU   $CD48
CD24  PCRLF    EQU   $CD24
CD03  WARMS    EQU   $CD03
0000  NUMBER    RMB   2

* DEFAULT ORG 0 IS OK IF WE DON'T USE DIRECT PAGE ADDRESSING

0002   BD CD48  START  JSR I   NDEC
0005   9F 00           STX     NUMBER
0007   BD CD48         JSR     INDEC
000A   1F 10           TFR     X,D
000C   D3 00           ADDD    NUMBER
000E   DD 00           STD     NUMBER
0010   BD CD24         JSR     PCRLF
0013   8E 0000         LDX     #NUMBER
0016   BD CD39         JSR     OUTDEC
0019   BD CD24         JSR     PCRLF
001C   7E CD03         JMP     WARMS
                       END     START
0 ERROR(S) DETECTED
```

I am presently having a bit of trouble transferring assembler output files imported into a 6809 text file and then moved to a PC formatted disk. It is necessary to add a linefeed to every line since FLEX terminates lines with CR only and the PC expects a CR followed by an LF at the end of every line. Most of the trouble was that my filter program is not quite right yet and it collapsed the assembler listing, removing all spaces from every line. I will fix it. Meanwhile I don't guarantee that the above listing is formatted just as the assembler output is.

There are several things to talk about here. First, NUMBER was given two memory bytes at address $0000. The label START is at the address $0002. The assembler did something automatically for me that I hadn't mentioned. It used a special short addressing mode in the instructions ADDD NUMBER and STD NUMBER. Since number is at address $0000 it used the direct page addressing mode. This is a special mode that meant a great deal back in the days of 4K memory. If you put your variables all on one page (in this case addresses $0000 through $00ff) they could all be reached using a single byte address rather than a two byte address. (A page is defined as all the memory that can be accessed with an 8 bit address. That number is 256 bytes). Thus every reference to the variables saved a byte compared to extended addressing mode. The 6809 has a Direct Page Register that can be set to any page in memory. When the processor is reset by power on, it resets to $00. Sometimes you can make use of this to squeeze a program size a bit. Actually the direct addressing mode saves a few clock cycles too, so the program will run a bit faster. Let's look at the addressing modes we have "tripped over" so far:

Direct STA $10 Stores contents of A at $(DP)10, i.e. the DP register holds the upper byte of the address for the STA instruction. **Extended STA $1000** Stores contents at $1000. **Immediate LDA #$56** Loads the value $56 into the accumulator

Excuse me if I seem to be repeating myself here. I am trying to say everything a number of different ways and to repeat as much as possible. When I was learning how to program I was confused over the immediate addressing mode for a long while, so I'll repeat. Using labels seems to add to the confusion, so let's do it with numbers first and then with labels:

LDD #$1234 Loads ACCD with the hexadecimal value 1234. **LDD $1234** Loads ACCD with the CONTENTS of memory address $1234. **LDX #NUMBER** loads X with the value assigned to the label NUMBER. In the case of the above program $0000. **LDD NUMBER** would load D with the contents of the variable NUMBER.

Note that the immediate addressing mode doesn't make sense for store operations, only for load or add or such. Immediate values can be thought of as program "constants".

We've kind of half looked at the "indexed" addressing mode too. We didn't use it explicitly in our program but OUTDEC required us to set up X as a pointer to the variable NUMBER. Note that the label NUMBER has the value $0000 and the label START has the value $0002 (see the assembler output listing leftmost column). The instruction LDX #NUMBER therefore loads X with the value $0000. If we had left out the # sign (a common error) it would have loaded X with the contents of the variable, our sum, 579 decimal. OUTDEC wants X to contain the address of the number to be output, not the value of the number, so we had to use the immediate mode. Perhaps I've said that enough times in enough slightly different ways.

Down the line we will complicate things a bit more when we talk about position independent code and putting variables at

addresses that are tied to the program address. When this is done, the program can be loaded anywhere in memory and it will run there. For now, let's try to keep it a little simple.

As we get into more complex programs you will notice that about half of the instructions involve the LD or ST instructions, i. e. LDA, LDB LDD LDX LDY LDU LDS and STA STB STD STX STY. These are memory reference instructions. They cause information to be moved from memory to a register or from a register to memory. There is the TFR instruction to transfer data from one register to another and the EXG instruction that EXchanGes values between registers. When we use an LD instruction to get a value from memory, that value in memory is unchanged. Only the value in the destination register changes. Similarly when we store a value to memory, the value in the source register doesn't change, only the value in the destination memory locations. Load and store are not the only memory reference instructions. We used ADDD for example. That instruction added the contents of a memory location to the D register, again leaving the memory value alone and only modifying the contents of the D register. To save the result, we had to store it in memory again.

If you are following this series let me say that an hour of practice is worth a year of reading. For example, you could enter the above program and assemble it. Try it out and when you get tired of seeing the sum of two numbers, see if you can modify it to add three numbers. Some time we'll extend it to add any string of numbers until you enter zero.

Don't be fooled. We've used a couple of rather powerful FLEX routines to do some things that would take a lot of code to do. Adding the two numbers is no chore, but converting the input ASCII string to it's binary representation and converting the sum back to an ASCII string are non-trivial operations. Perhaps we can write our own down the way a bit and show how it might be done within FLEX. Generally if you want to keep a program small and simple, it is advantageous to use operating system calls as much as possible. They only cost a few bytes in the way of a JSR to an extended address!

I have a few more of what my daughter calls "factoids" for you concerning the 6809, and for that matter all of the Motorola processors.

1) Values larger than one byte are stored highest order byte in lowest memory, (what I generally think of as "left to right" order). That is, as you progress through memory counting from low addresses to high, you encounter 16 bit and floating point values with the high order byte first. This is backwards from what the Intel processors use. (I'll resist the temptation to say that Intel does it backwards)! It has caused me no end of frustration when I first started to work with manipulating numbers in memory in a PC environment. Strangely both the 6809 and the Intel processors store strings with the first letter in the lowest memory. That is, a string and a number read "low to high" in a Motorola processor environment, but in the Intel, a string reads low to high memory but a number reads high to low. (If I remember correctly, the 6502 stores 16 bit values in the same order as the Intel processors).

2) 16 bit values may be stored at either even or odd addresses. The 6809 doesn't care. You can store the contents of X at address $0101 or at $0102. It doesn't matter. This is NOT true of the 68000 where 16 and 32 bit values must be stored at even addresses.

More on C

A while ago I did a little note on C, and using arrays vs pointers. My words drew a letter from Richard Brewster (Issue 67) saying that source code using arrays truely generates more object code than source code using pointers. I guess I can say that the book that I quoted lied! (It basically said that the compiler would convert array notation into the same object code as pointer notation). I'm convinced. That was written several months ago. Since then I agreed to teach an informal class in C so I dug up two tutorials and did a lot of reading. I am beginning to feel comfortable writing a program in C and expecting it to work rather quickly if not on the first try. I understand pointer notation in C, and did when I wrote the above, but understanding it, feeling comfortable using it, and having it's use be second nature (like riding a bicycle or driving a car) are three different things. I am about at the second of those three stages now.

Unrelated

Detroit has a morning radio show host named J. P. McCarthy. One morning this week he was talking to someone about a "roast" at which he is to be the "roastee". He mentioned a high price per plate at the event and then added that it would, of course, be "a benefit for the hungress and homely". Next try it came out "hungress and homeless", and not until the third try did it come out right as "hungry and homeless". I chuckled all the way to work! I got to thinking about what a benefit for the homely might entail... a grant for plastic surgery? Masks? Maybe I could qualify for one or the other.

A friend of mine who sang regularly on the radio (WMBI in Chicago) once was singing "The Lord's Prayer" and because of previous fooling around with the words, they came out "Lead us not into Penn Station"! He said he would never more play with the words. He got to that point and then just couldn't remember which was correct!

Lastly, I once worked at the University of Illinois School of Chemical Sciences. That school had a very nice hard working cooperative business manager named Larry Hess. We got to calling him Harry Less (or was it Hairy Les) privately, and one day one of the staff called the business office and, of course, asked for Harry Less. "You mean Larry Hess", chuckled one of the secretaries. Hopefully she thought it was just a slip of the tongue, but we stopped our private joke!

Special Feature

Intermediate Users

Part 5: Serial I/O

MULTIPROCESSING FOR THE IMPOVERISHED

by Brad Rodriguez

At long last, the final installment of the "Scroungemaster II" multiprocessor. I may in the future write some software articles around this board — a 6809 Forth is certain — but this wraps up the hardware description. Go, and prototype!

SERIAL INTERFACES

Figures 1 and 2 contain the two UARTs which provide four serial ports for the Scroungemaster II. Since these circuits are nearly identical, I will discuss them together. Afterwards I will mention the differences.

U10 and U11 are Zilog Z8530 Serial Communications Controllers (SCCs). These are *very* versatile chips, each of which includes two serial ports and two programmable baud rate generators, plus some miscellaneous I/O. All of the common asynchronous formats are supported, plus many synchronous ones (see sidebar). The 4 MHz part will transfer data at up to 1 Mbit/second. (I'm told Apple uses this chip for the Appletalk network in the Macintosh.)

U10 is enabled by chip select IO0\, and U11 by IO1\. Each chip looks at only the low two address lines, so each occupies four consecutive locations in memory. As wired here, the SCC registers appear as follows:

U10 (U11) address	register
FFC00 (FFC80)	Port B (D) command
FFC01 (FFC81)	Port B (D) data
FFC02 (FFC82)	Port A (C) command
FFC03 (FFC83)	Port A (C) data

(Remember that these are addresses in the mapped 1 MB memory space.) The ports of each chip are called A and B in the Zilog documentation. On the Scroungmaster II board I refer to U10 as ports A and B, and U11 as ports C and D. If only two serial ports are needed, U11 can be removed.

The serial ports of most personal computers use signal levels adhering to the Electronics Industries Association (EIA) RS-232 standard. This standard specifies a voltage of -5 to -15 volts for logic "1", and +5 to +15 volts for logic "0" [PIP85]. U26 and U25 generate and receive RS-232 levels for the Scroungemaster II. U26 is an LM1488 quad RS-232 driver, and U25 is an LM1489 quad RS-232 receiver. Only the Receive Data (RXD) and Transmit Data (TXD) from each serial port are converted to RS-232 levels. (The RS-232 standard specifies a number of control signals, such as DCD, DTR, DSR, RTS, and CTS, all of which must use these same logic levels. This board does not support these signals.)

Note that if *and only if* RS-232 drivers are used, +12v and -12v power supply voltages must be provided. (Everything else runs on +5v only.)

RS-232 is limited to short cable runs, relatively low data rates, and point-to-point connections. To allow higher data rates or longer cables, EIA defined the RS-485 standard. RS-485, an improved version of the earlier RS-422, uses balanced transmission, which means that each signal is carried on a *pair* of wires: one wire carries the "true" signal, and the other wire carries its logical inverse. Signal levels are approximately 5 volts and 0 volts, but the detailed specifications of RS-485 require the use of special driver chips and not just TTL outputs. The Scroungemaster II uses a 75174 quad RS-485 driver (U13), and a 75175 quad RS-485 receiver (U12). Again, only the serial RXD and TXD are converted to these levels.

Important: On this board, the choice of RS-232 vs. RS-485 is made by which pair of driver chips you install. This means that the four ports may be all RS-485 or all RS-232, but you can't have a mix of RS-485 and RS-232. Do *not* install both sets of drivers! Install only U12/U13, or U25/U26.

Another advantage of RS-485 is that it allows *multidrop* operation, in which several serial ports are connected to one pair of wires. When you connect a network of CPUs in this way, any CPU can talk to any other (and you can also save a lot of wire!). For this to work, though, only one CPU can drive the wires at any one time. The other CPUs must have their RS-485 drivers *disabled*, i.e., in a high-impedance state, so they neither drive nor load the serial "bus." (The problem of deciding *when* each CPU can enable its driver is worth several more articles, and I won't talk about it here.)

The 75174 chip (U13) is somewhat unusual in that its RS-485 drivers are enabled in pairs. (There are only two enable inputs for the four drivers.) This means that you can't use all four Scroungemaster II serial ports for multidrop, since each multidrop port needs an independently controlled transmitter. I've arranged the drivers so that ports A and C are enabled together, and likewise B and D. This means that if you have

only one SCC installed (U10), you have two fully-functional multidrop RS-485 serial ports. If you install both SCCs, you can run all four ports RS-485 point-to-point (since for point-to-point operation you leave the driver always enabled). Or you can run two ports point-to-point and one port multidrop (e.g. B and D point-to-point, A multidrop, with C unusable because it's driver is enabled according to A's needs).

The A and C drivers are enabled when the RTSA\ output of U10 is high. The B and D drivers are enabled by RTSB\ of U10. (Thus U11 is optional in a two-port system.) The RTS\ outputs must be controlled by software. The RS-485 receivers are always enabled.

The serial signals are brought out on 10-pin headers such that two ports may be connected together by putting a half twist in the ribbon cable:

pin	signal	signal	pin	
1	CTS\	<->	RXRDY\	10
2	RXC	<->	TXC	9
3	GND	<->	GND	8
4	RX-	<->	TX-	7
5	RX+	<->	TX+	6
6	TX+	<->	RX+	5
7	TX-	<->	RX-	4
8	GND	<->	GND	3
9	TXC	<->	RXC	2
10	RXRDY\	<->	CTS\	1

In other words, a half twist serves as a null modem. (If RS-232 signals are used, the signals appear on the TX- and RX- pins.)

TXC and RXC are the serial clock output and input, respectively, of the serial port. (Strictly speaking, TXC can be programmed to be either an output *or* an input, but only the former is useful here.) If the TXC of one port is connected to the RXC of another, the two ports can use the x1 clock mode, and all synchronous modes. This is intended for 1 Mbit/sec operation between adjacent CPUs. TXC is not buffered, so the cable length can be only a few inches.

Inverters U9C through U9F generate "Rx ready" signals that I hope will allow automatic flow control between two connected serial ports, as follows: The SCC's W/REQ\ pin can be programmed to go low when the receive buffer contains a character ("DMA Request on Receive" mode). Thus, when the receive buffer is empty, W/REQ\ is high and RXRDY\ is low. This is connected to the CTS\ input of the "sending" SCC, which should be programmed in the "Auto Enables" mode. In this mode, a low on CTS\ enables the transmitter, and a high disables the transmitter (after it finishes sending any character in progress). This scheme is still to be tested, but if it works, it will prevent one serial port from "overrunning" the other...very handy at 1 Mbit/second! Note that the RXRDY signals are not buffered, and, like the clock signals, are only usable over short distances.

Only the RX and TX signals should be run over long distances (up to 50 feet for RS-232, or 4000 feet for RS-485). In the absence of the clock signals, use a x16 or x64 asynchronous mode, or use the DPLL for clock recovery in synchronous modes. (For further details refer to the Zilog technical manual.) Flow control must be done in software.

You can run the clock and flow control signals long distances if you provide suitable external drivers (RS-232 or RS-485). You may also want to add external logic functions or level translation (say, to produce a MIDI port). To facilitate this, pin 10 of the serial connector can be jumpered to +5 volts (JP10-JP13). Of course, you lose the RX Ready signal if you do this.

The DTR\ pins of the SCC can be used as general-purpose outputs. These two outputs from U10 provide as the the TASK and SWREQ\ control signals, used elsewhere in the Scroungemaster II. (Another reason why you should keep U10, and not U11, if you need only two serial ports.)

The DCD\ pins of the SCC can be used as general-purpose inputs. These can also be programmed to generate an interrupt whenever the level changes in either direction. So, these two inputs on U10 are used for SWGRANT\ and IRQ4\. But in the Auto Enables mode, the DCD\ inputs must be grounded to enable the SCC receivers. This can be done with JP14 and JP15 (disabling the SWGRANT\ and IRQ4\ functions.)

The DTR\ and RTS\ pins of U11 have no assigned function, and so are brought out to a 4-pin header. The DCD\ pins are permanently grounded on U11.

JP7 and JP8 control interrupt assignments of the SCCs. Along with JP9 and JP4, which control the PIO and PC bus interrupts (respectively), they allow the following possible assignments:

SCC#1	SCC#2	PIO	bus interrupt
	IRQ	IRQ	IRQ
FIRQ	FIRQ		
NMI		NMI	
			IRQ4 (via U10)

All three I/O chips may have distinct interrupts, which makes for the simplest interrupt routines. Either or both SCCs may use the 6809's fast interrupt (FIRQ) for high data rates; but of course, when two chips share an interrupt, your software must poll them to identify which is generating the interrupt. Also, if a chip is connected to the Non-Maskable Interrupt (NMI), its interrupt cannot be disabled by the CPU — it only be disabled by writing to that chip's interrupt control register. (Both the SCC and the PIO have such a register.)

If you are using PC bus interrupts and all three I/O chips, then two of the four interrupt sources must share one of the three 6809 interrupts (and polling must be used). Or, you can route the PC bus interrupts to "IRQ4" (U10's DCDB\ input), in which case PC bus interrupts will be processed through U10's interrupt logic.

LED DISPLAY

Whenever I design a CPU board, I always include at least one LED — the most useful tool for bringing up a new (or dead) board. By popular demand, the Scroungemaster II has *eight* programmable LEDs, in a 10-segment bargraph, D1. (The middle two segments are used as a power indicator.) Octal latch U23 (74HCT377) serves as an 8-bit parallel output port, selected by IO3\ (mapped address FFD80). When a "0" is written to any bit of this output port, the corresponding LED will illuminate. For simplicity, the LEDs share a single current limiting resistor, so the more LEDs you turn on, the dimmer they get. A 74HCT377 is recommended instead of a 74LS377 because of the HCT part's greater current capacity.

PROGRAMMABLE INTERRUPT GENERATOR

One of the experiments I want to perform with my multiprocessor system is having one processor interrupt another, as a method of signalling. U29 and U30, if installed on one CPU board, make a programmable interrupt generator. The comparator U29 (74LS688) identifies when a write occurs to any I/O address 000-007 on the IBM PC bus. (In IBM PCs, these addresses are reserved for the motherboard, so we know no plug-in peripheral card will ever use them.) When such a write occurs, the programmable latch U30 (74LS259) writes the data bit XD0 into one of eight latches. Which of the eight is selected by address bits XA0-XA2.

The eight latches output to interrupt lines IRQ0-IRQ7. Each of the eight possible CPUs can be jumpered to one of these lines. So, if CPU #5 is jumpered to receive IRQ5, you can interrupt this CPU by writing a "1" to PC bus I/O address 005. (Recall that this appears as mapped address DFC05 to the 6809. Also, PC bus interrupts are active *high*, so writing a "1" asserts the interrupt, and a "0" removes the interrupt.) Since U29 and U30 are electrically connected directly to the IBM PC bus, and not to the board's "internal" bus, *any* CPU can write to them. So, by installing U29 and U30 on one board, and jumpering the interrupts appropriately, any CPU can interrupt any other.

Normally, U29 and U30 will be installed on the bus master. *These should not be installed if IBM PC peripheral cards will be generating interrupts.* This is because U30's outputs are always active — like many PC perhiperals — and if one board is pulling an interrupt line high while another pulls it low, someone's drivers will be damaged.

TIMING DIAGRAM

When designing microprocessor circuits, it's not enough to get the voltage levels and the logic right. You have to get the *timing* right, too. Timing specifications are published by the manufacturer (usually in the data books, e.g. [MOT83, TEX85, ZIL88]) for all microprocessor and logic chips. These specs describe:

a) when each input is *required*,
b) when each output is *valid*, and
c) how long the chip will take to perform a given function.

While automated tools exist which will analyze the timing of any digital circuit, many engineers still do it manually by constructing a *timing diagram*.

Figure 3 is a partial timing diagram for the Scroungemaster II. The timing is shown for the fastest CPU (68B09 with an 8 MHz oscillator) and the slowest recommended "glue" logic (74LS TTL). This is usually the "worst case" for timing analysis, since the CPU imposes the "deadlines" while the glue logic introduces the "delays." If any part of the logic adds too much delay, I can replace chips with a faster equivalent (74ALS, 74AS, or 74F TTL). *Important:* although supposedly equivalent, many 74HCT parts are actually slower than 74LS. If you want to use 74HCT, verify the timing first. Better still, use the faster 74HCTLS, 74ACT, or 74AHCT families. Also, you may be able to substitute "plain" 74 or 74S family parts in some places, but beware of their much higher power consumption.

Each line of Figure 3 represents one signal (or a group of signals with identical timing). The horizontal axis is time; each division is 50 nsec. Only the most critical signals are shown. For many devices (e.g. RAM and EPROM) you can compare the timing requirements of the chip directly with this diagram — you don't need to draw every signal.

The first line is the 6809 E signal. The "falling edge" of E (from +5V to 0V) is the start of a 6809 memory cycle, and is the "reference point" for most of the timing specs, so I have drawn the diagram with time T=0 at the falling edge. Note that, according to Motorola specs, E could be low for as little as 210 nsec. Thus I've drawn the rising edge of E to span a range from T=210 to 250 nsec. During this "window", E could rise at any time, so we *don't know* for sure whether E will be high or low.

The second line is the 6809's "quadrature" clock signal, Q. If the falling edge of E is T=0, then Q will rise between times T=80 and T=125 nsec. Although the Scroungemaster II doesn't use the Q signal, some of the 6809 signals are specified relative to Q, so it needs to be on the diagram.

The 68B09's address and R/W\ lines are guaranteed to be valid and stable 15 nsec before the rising edge of Q. They remain stable throughout the cycle, and 20 nsec into the next cycle, i.e., 20 nsec after the *next* falling edge of E. (Obviously, for the first 20 nsec of *this* cycle, address and R/W\ carry "old" data.) On the diagram, I have "X'ed out" the period when these lines are changing (and thus unknown). Also, when they are stable they could be high *or* low — the address could be anything! — so I've drawn *both* signal levels (0 and +5) during this period. These are common conventions used in timing diagrams.

The 68B09 requires that data read from memory or I/O be valid 40 nsec before the start of the falling edge of E. Because of E's rise and fall times, this edge could be as soon as T=470 nsec, so data must be stable at T=430 nsec. Also, this data must remain stable until 10 nsec after the end of the falling edge of

E, i.e., until T=510 nsec. This is a *need* — a requirement to be satisfied.

The 68B09 guarantees that data written to memory or I/O will be valid 110 nsec after the rising edge of Q, and will remain valid until 30 nsec after the next falling edge of E. On this diagram, this is from T=235 to T=530 nsec. This is a *given* — a known timing characteristic of a signal.

Now we can analyze the Scroungemaster II circuit. First, look at the memory mapping and decoding schematic. All of the address and control inputs to the 74S189's are stable at T=110 nsec. The 74S189 takes at most 35 nsec to access its data, so its outputs (and thus the "mapped address") are valid at T=145 nsec.

The WRMAP\ and OEPROM\ signals are derived from A15, R/W\, and E. You can see that A15, R/W\, and the inverted R/W\ (U7D) will be stable long before E goes high. So E is the deciding factor, and the WRMAP\ or OEPROM\ signal will go low 10 to 15 nsec after E goes high (assuming that the CPU is writing the map or reading the EPROM, respectively). WRMAP\ or OEPROM\ will remain low until 10-15 nsec into the next cycle. This is the propagation delay of a 74LS10 NAND gate.

The second stage of decoding, U4 and U8A, produces the signals IOZONE\ and ONBOARD\. These signals are logically derived from the CPU address and the mapped address, and we know the CPU address must be stable before MA12-MA19 can be stable. So IOZONE\ and ONBOARD\ will be stable at T=160 nsec, 15 nsec (the maximum delay of the NAND gates) after the mapped address. In this case we're not worried about the minimum delay of the NAND gates (which would tell us how *soon* IOZONE\ and ONBOARD\ could *possibly* be asserted). Both of these signals will remain stable until the address changes, at least 20 nsec into the next cycle — which, we will see, is sufficient.

IOZONE\ is the last input of U24 to become stable, so U24's outputs — the IOn\ select lines — will follow IOZONE\ by 32 nsec, the maximum delay of the 74LS138. Similarly, the OFFBD\ signal (U6D) follows ONBOARD\ by 15 nsec.

The read and write strobes, however, are not output by U5 until E goes high. You can see from the diagram that all the other inputs of U5 will be stable before the rising edge of E. So, the various read and write strobes will trail E by 14 to 26 nsec (the propagation delay from the 74LS138's G1 input to its outputs). In this case we want to know upper *and* lower bounds on the strobes.

The next two lines show transitions propagating through the memory request logic. If the bus is not available, MRDY will be pulled low no later than time T=250 nsec. For MRDY to stretch a memory cycle, the 68B09 needs to see it pulled low no later than T=360 nsec (110 nsec before the falling edge of E). You can see that, even in the worst case, we have 110 nsec to spare!

If we're doing a three-cycle stretch, we can't have glitches on the REQ signal (NAND gate U6A). To ensure this, STRETCH\ must go low before OFFBD\ goes high. STRETCH\ will go low no later than T=520 nsec (20 nsec after the falling edge of E). We know that OFFBD\ follows ONBOARD\, which follows the mapped address, which follows the 68B09 address, which remains valid 20 nsec into the next cycle...so there's no problem.

We can control (via JP6) how much extra delay is added to a bus cycle when the 6809 has to wait for ownership. What happens if this CPU already owns the bus? The next six lines of the timing diagram show that DRIVENBL\ is asserted (low) at T=205 nsec. The address will then be output to the bus no later than T=235 nsec, and the bus RD\ and WR\ strobes will be asserted no later than T=310 nsec, and will be at least 185 nsec wide. Read data must be valid *on the bus* no later than T=420 nsec, to allow for a 10 nsec delay through the bus buffer. Write data will be valid *on the bus* no later than T=245 nsec. Finally, if a bus peripheral (such as a CGA card) wants to stretch the memory cycle, it must assert IORDY no later than T=330 nsec. (Does this meet the specs of PC peripherals? Who knows? Only experimentation will tell.)

We now have enough information to determine if any given memory or I/O chip will work with the Scroungemaster II CPU. The first example shown is the Z8530 SCC chip. (This analysis is also valid for the Z8536 parallel I/O chip, since it has nearly identical timing requirements.) For convenience, I have duplicated the lines showing the read and write strobes and the IOn\ selects.

The Z8530 requires that its address lines be valid from T=125 to T=520 nsec. Since A0-A1 come from the CPU, we can see that this requirment is met.

If this chip is to be accessed, the Z8530's CE\ input (IO0\ or IO1\) must go low before RD\ or WR\ go low, and must stay low as long as RD\ or WR\ is low. (As Zilog specs it, CE\ must go low at least 0 nsec before RD\ or WR\ goes low, and must stay low at least 0 nsec after RD\ or WR\ goes high.) This requirement is met with time to spare. If this chip is *not* to be accessed, CE\ must be high at least 100 nsec before RD\ or WR\ goes low...which means that the IO0\ or IO1\ select must not stretch more than 125 nsec into the next cycle! Again, no problem.

The 8 MHz Z8530-8 will output read data no later than T=430 nsec (140 nsec after its RD\ goes low), which exactly matches the 68B09's need. This data will remain valid at least 14 nsec — the delay of U5 — after E falls; the 68B09 needs it to be held only 10 nsec after the fall of E. This is a case where using a faster logic family (for U5) could theoretically *introduce* a timing problem!

The Z8530 parts are peculiar, and perverse, in requiring write data to be valid before the *falling* edge of WR\. 10 nsec before, to be precise. If the decoding logic is running near its fastest

spec, WR\ could be asserted as soon as T=220 nsec. But the CPU doesn't guarantee write data until T=235 nsec! This 15 nsec discrepancy can be resolved several ways:

a) hope that the 68B09 and Z8530 timing specs are sufficiently conservative to cover this 15 nsec discrepancy (e.g., the 68B09 may output data sooner);
b) hope that the decoding logic operates near its nominal timing and not its minimum timing specs (another case where faster logic spells trouble!);
c) add some gates to delay the RD\ and WR\ signals, being careful not to violate the data hold time spec; or
d) use the CMOS Z85C30, which intelligently latches its write data on the *rising* edge of WR\.

So far I've had good luck relying on (a) and (b). If I run into problems, (d) is an easy out. If I were being paid for this I might expend the extra effort to do (c). But note that the write data must be held at least as long as WR\ is low, and that WR\ can go back high as late as T=525 nsec. The 68B09 holds write data until T=530 nsec. Any delaying circuit must therefore delay the falling edge of WR\, but *not* the rising edge!

The second example is the 2 MHz 6522A Versatile Interface Adapter. This was briefly considered for the SM II's parallel I/O. You can see the problem with this chip immediately: it requires that its CS\ be valid at T=130 nsec (80 nsec before the rising edge of E). But the IOn\ select lines are not valid until T=192 nsec! This 62 nsec discrepancy is too big to ignore, and there is no way to fix it short of putting the 6522A into the "unmapped" memory space...a solution I did not want to use.

Observe also that the 6522A provides read data 10 nsec too late for the 68B09. There's no easy way to fix this, since this is a 190 nsec delay from the rising edge of E. If it had been a delay from some other signal, we could try to provide that other signal sooner (e.g., with faster logic). As it is, we can only use a faster part...and a faster 6522 doesn't exist.

I conclude that 74LS logic can be safely used throughout the

SERIAL DATA FORMATS

In parallel data transmission, such as on the PC bus, several data bits are put on several wires at the same time, and there are usually additional wires to control the timing of the transfer. When you want to send data over a *single* wire, you have to send the data bits one at a time, or *serially*. Serial transfer opens a whole new can of timing and control problems.

Usually, the bits will be placed on the wire one at a time, for a fixed period, starting with the least-significant bit (D0). The *baud rate* is the reciprocal of the bit period, so that at 9600 baud, each bit is placed on the line for 1/9600th of a second. (For all our purposes, anyway; you can find a more detailed discussion in [ARR93].) Both ends of the serial link use a crystal-controlled clock to measure the bit period accurately.

But when does this bit period start...and in a continuous stream of data, how do we identify the first bit of a new byte?

In *asynchronous* transmission, each byte can be sent independently ("asynchronously"). Between bytes, the wire is held at a steady logic "1". To signal the start of a byte, a logic "0" is placed on the wire for exactly one bit period. This is called the "start bit." The 1-to-0 transition marks the "edge" of the bit period, and all succeeding bit periods can be timed from that reference point. Even if the two ports' clocks are a few percent different, the accumulated error over eight bits is small enough that the bits can be recovered. But over several bytes the error will build up, so after each byte a logic "1" is output for one or two bit periods — one or two "stop bits." Having a stop bit means that the next byte will always start with a 1-to-0 transition, resynchronizing the bit timing. (Two stop bits were required in the days of mechanical teletypes. Electronic serial ports get by fine with just one.)

From five to eight data bits can be sent between the start and stop bits (least-significant bit first). After the data bits, and before the stop bit, can be an additional bit called the *parity* bit. The parity bit is designed to make the total number of "1" bits either Even or Odd. If one bit is received incorrectly, the parity will be wrong, causing a "parity error." (The parity scheme thus detects all "one-bit errors." If two bits get flipped, the parity is unchanged.) Sometimes the parity bit is forced to "0" ("Space") or "1" ("Mark"), or is omitted entirely ("None"). So, when you see an IBM PC set up for "9600,N,8,1", you can now read that as "9600 baud, No parity, 8 data bits, 1 stop bit."

In *synchronous* transmission, the transmitter clock is connected to the receiver, and the bytes are sent in a continuous uninterrupted stream, without start and stop bits. Once the starting point is found, we simply read every eight bit periods as a new byte — there had better not be delays between the bytes! To identify the starting point we send a *synchronization (sync) character* — a unique bit pattern that we can detect as bits are shifted through the receiver. (It's also possible to use a separate sync signal, but this is uncommon.) One advantage of synchronous transmission is that it's 20% faster — a byte is sent in eight bit periods, and not ten (eight data + one start + one stop).

Sometimes you don't need to physically connect the transmitter clock to the receiver. A digital phase-locked-loop (DPLL) such as contained in the Zilog SCC can synchronize the receiver almost perfectly to the transmitter, provided that there are frequent 1-to-0 or 0-to-1 transitions in the data. There are several ways to ensure this. One software scheme is group-coded recording (GCR), used in Apple floppy disks. One hardware scheme supported by the SCC is the synchronous data-link control (SDLC) protocol.

The hardware can also add extra transitions to the bit stream, to make "clock recovery" easier. Among the methods are FM (frequency modulation, also called "bi-phase" encoding), MFM (Modified FM), and Manchester encoding. (MFM is used for double-density floppy disks and many hard disks.) The details are too long to relate here; just be aware that the Zilog SCC can send and receive FM, and can receive Manchester encoded data. [ZIL86]

The bit timer (in asynchronous modes) and the DPLL (in synchronous modes) run from a crystal-controlled timebase (a programmable counter called the "baud rate generator"). If the timebase is 16 times faster than the bit period, then the bit period can be counted off in 16 steps — we can say that the timing resolution is 1/16th of a bit. This is called a "16x" (16-times) clock. In its various modes, the Zilog SCC can use a 64x, 32x, 16x, or 1x clock. If a 1x clock is used, the receiver has to be synchronized perfectly to the transmitter (by a physical connection), since the timer can't be "tweaked" in fractions of a bit. This is why the Scroungemaster II serial ports allow the receiver to be connected to the transmitter clock.

SM II, even with the faster CPU. I make the (perhaps unjustified) assumption that if 74LS logic is fast enough for the 68B09, the 1.5 MHz 68A09 and the 1 MHz 6809 will pose no problem. In view of the timing discrepancies of the Z8530, I may be too hasty assuming this, and I should do a fresh analysis at 1 MHz. I suspect that with the slower CPUs, the slower 74LS189 can be used instead of the 74S189 in the memory mapping logic. (The last time I checked, though, the 74S189 was still less expensive.) I'll leave these as exercises for the student, and I'll be interested to hear from you.

TEST PROGRAM

Listing 1 is a simple test program for the Scroungmaster II, written for the PseudoSam Level 1 freeware assembler (A6809). It begins by initializaing the memory mapping registers (which are in a random state on power-up). It then counts in binary from 0 to 255 on the LEDs, initializes one SCC (U10), and enters a loop of receiving and sending characters on serial port A. The serial port will run at 4800 baud with the basic 4 MHz oscillator, or 9600 baud with an 8 MHz oscillator (requiring 68B09 CPU and Z8530-08 SCC). Either an 8Kx8, 32Kx8, or 128Kx8 static RAM must be installed for the serial initialization subroutine to work.

REFERENCES

[ARR93] American Radio Relay League, The ARRL Handbook for Radio Amateurs, ARRL, Newington, CT (1993). A new edition is published every year. One of the best references for electronics and radio, but rather weak on computers and digital stuff — except for communications, naturally. For some reason the ARRL thinks that "baud" has a plural, "bauds."

[MOT83] Motorola Inc., Motorola 8-Bit Microprocessor and Peripheral Data (1983).

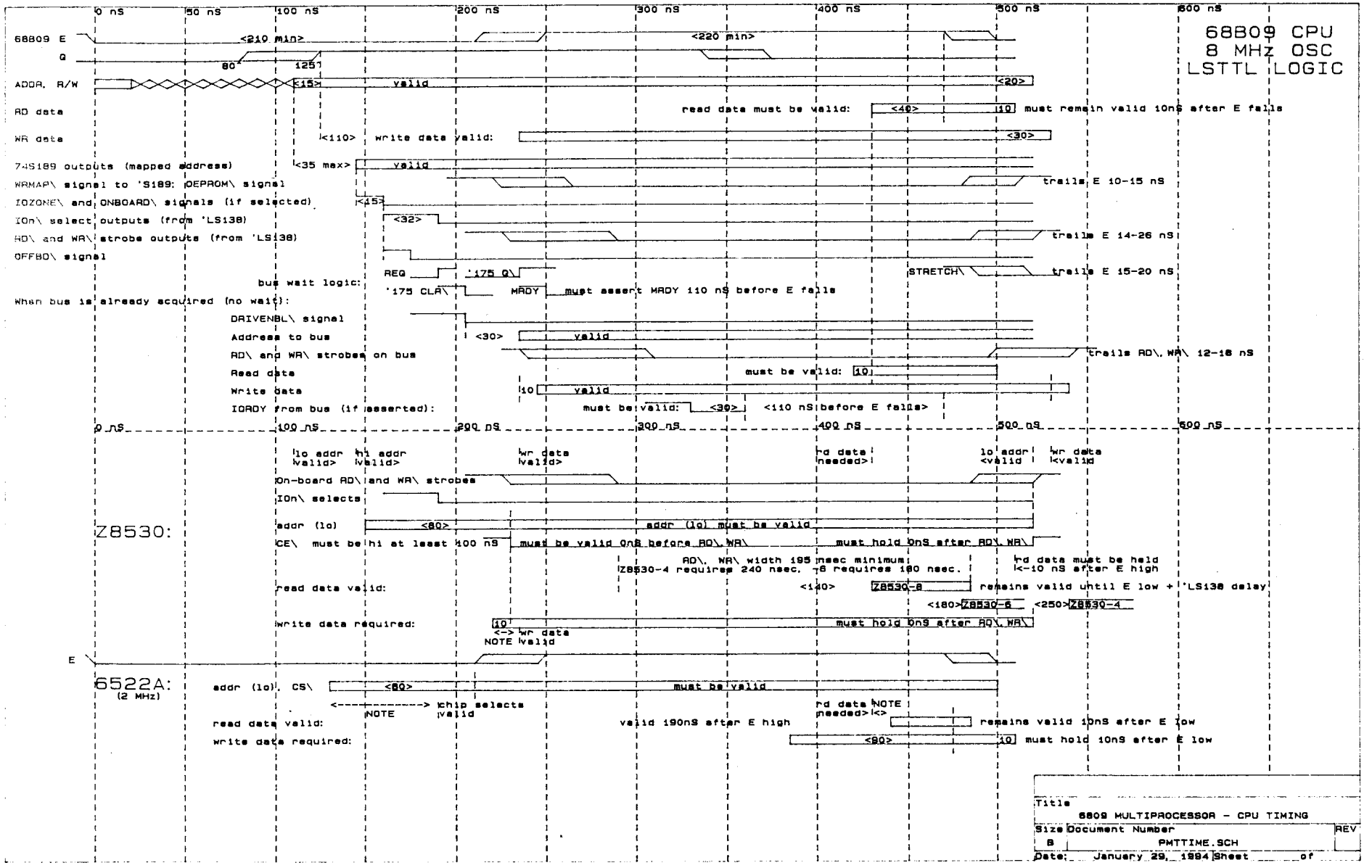
[PIP85] Pippenger, Tobaben, et al., Linear and Interface Circuits Applications, Volume 2: Line Circuits, Display Drivers, Texas Instruments (1985), ISBN 0-89512-185-9. An excellent sourcebook which, alas, may no longer be available from Texas Instruments. Well worth finding.

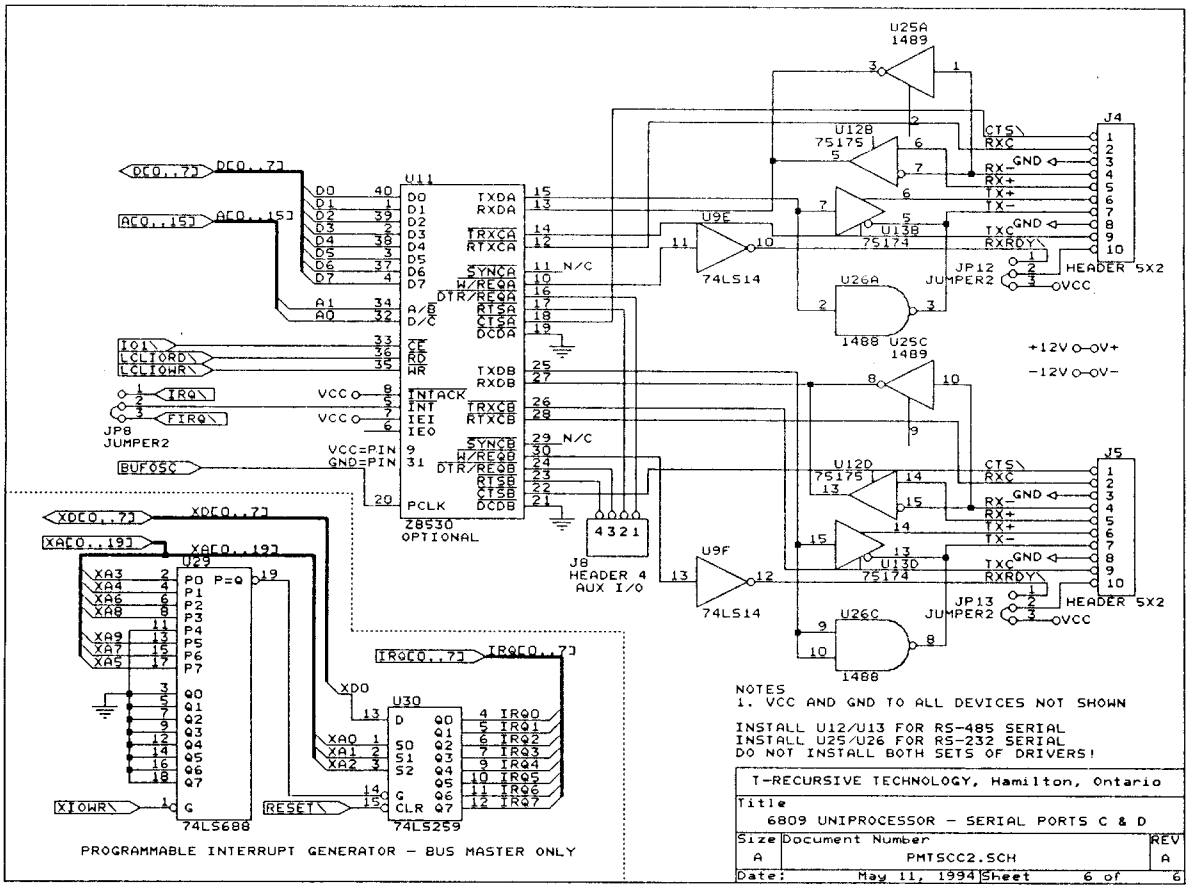
[TEX85] Texas Instruments Inc., The TTL Data Book, Volume 2 (1985).

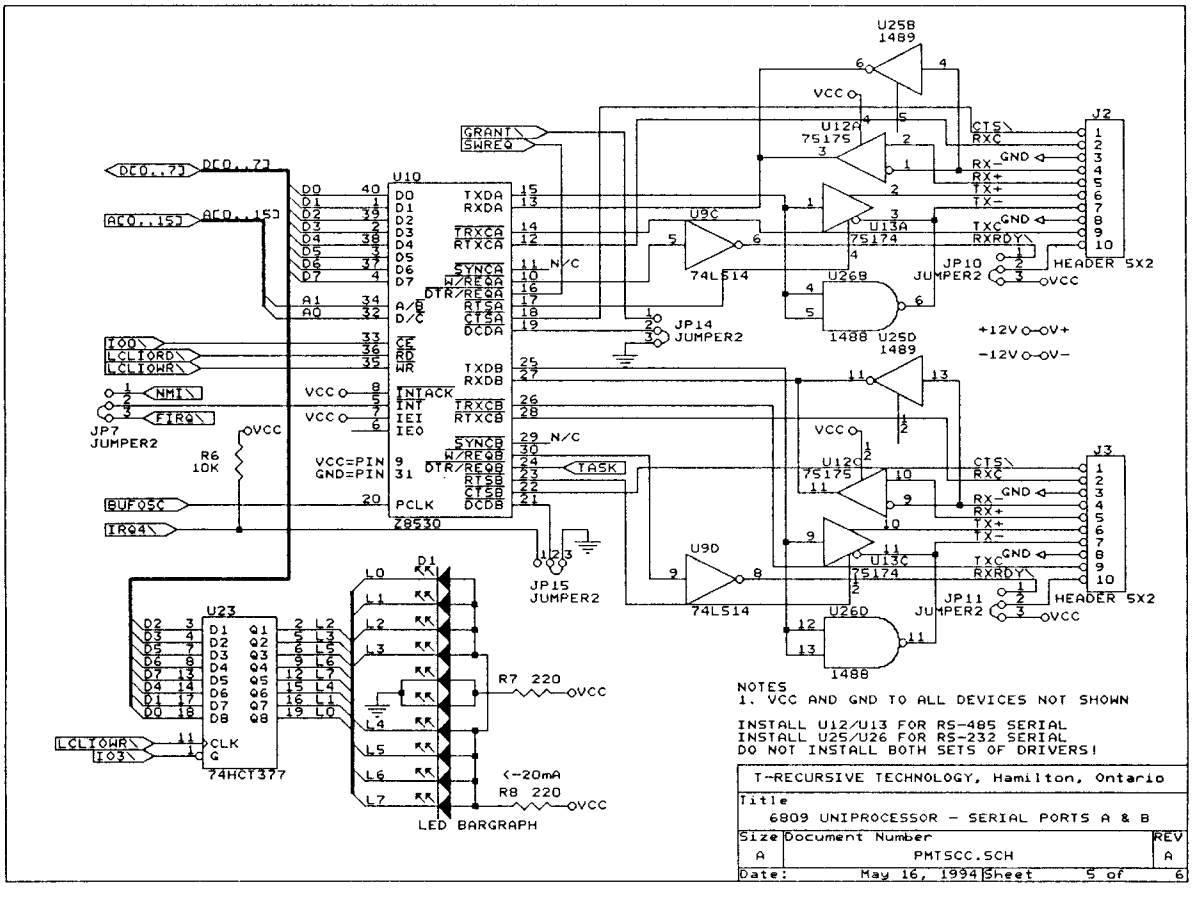
[ZIL86] Zilog Inc., Z8030 Z-BUS SCC / Z8530 SCC Serial Communications Controller Technical Manual (September 1986). Essential if you want to do anything fancy with the SCC.

[ZIL88] Zilog Inc., Z8000 Family Data Book (November 1988).


```
        .command -ai ; output in Intel hex format
;
; Listing 1. Test program for Scroungemaster II.
; Can be run from a 2764, 27128, or 27256 EPROM.

; 6809 reset vector
        .org h'fffe
        .dw entry

; program addresses of on-board I/O,
;       when 7xxx is mapped to FFxxx.
        .equ endram,h'7c00
        .equ sccbcmd,h'7c00
        .equ sccbdta,h'7c01
        .equ sccacmd,h'7c02
        .equ sccadta,h'7c03
        .equ sccdcmd,h'7c80
        .equ sccddta,h'7c81
        .equ scccmd,h'7c82
        .equ sccdta,h'7c83
        .equ pio,h'7d00
        .equ led,h'7d80
        .equ io4,h'7e00
        .equ io5,h'7e80
        .equ io6,h'7f00
        .equ io7,h'7f80

; Scroungemaster II minimal initialization.
; When the SM II is powered up, the mapping RAM
; is in an unknown state, so only the EPROM and
; mapping RAM can be accessed. The first thing
; we must do is put the mapping RAM in a known
; state. This mapping will make the on-board I/O
; accessible. No matter what RAM chip is
; installed, 7K of RAM will be available from
; program addresses E000 to 7BFF. Remember that
; the mapping values are logically inverted by
; the mapping RAM.
        .org h'fe00
entry:  clra
        sta h'f000      ; map 7xxx -> FFxxx
        inca            ;(on-board RAM & I/O)
        sta h'e000      ; map 6xxx -> FExxx
        inca            ;(on-board RAM)
        sta h'd000      ; map 5xxx -> FDxxx
        inca
        sta h'c000      ; map 4xxx -> FCxxx
        inca
        sta h'b000      ; map 3xxx -> FBxxx
        inca
        sta h'a000      ; map 2xxx -> FAxxx
        inca
        sta h'9000      ; map 1xxx -> F9xxx
        inca
        sta h'8000      ; map 0xxx -> F8xxx

; A simple loop to output 00->FF to LEDs,
; without using any RAM. Remember that
; the LEDs will appear logically inverted
; ("1" = off, "0" = on).
        clrb
tloop:  leax 1,x        ; increment counter lo 16
        bne tloop
        incb            ; increment counter hi 8
        stb led         ; output to LEDs
        bne tloop

; Initialize one Zilog SCC. Subroutines
; require the use of the stack, so this
; needs RAM. Also note that some common
; SCC registers are set up in the port A
; table, and some in the port B table, so
; you need to initialize BOTH ports.
        lds #endram     ; point stack to RAM top
        ldx #sccatbl    ; port A setup table
        jsr sccinit     ; common init routine
        ldx #sccbtbl    ; port B setup table
        jsr sccinit     ; common init routine

; A simple loop to receive a character,
; output it to the LEDs, increment it,
; and output it to the serial port. We
; know that by the time one character is
; received, the transmitter will be ready
; to send one character.
qloop:  ldb sccacmd     ; wait for rx char avail.
        andb #1
        beq qloop
        ldb sccadta     ; input char from scc
        stb led         ; output char to led
        incb
        stb sccadta     ; output char+1 to scc
        bra qloop

; Zilog SCC initialization routine,
; entered with table address in X.
; You can use this routine in your own
; applications.
sccinit: ldy ,x++       ; get scc port address
        ldb ,x+         ; get # of bytes to output
sccloop: lda ,x+        ; get byte from table
        sta ,y          ; store to SCC port
        decb
        bne sccloop
        rts

sccatbl: .dw sccacmd    ; port address
        .db 37          ; 37 bytes follow
        .db h'0         ; just in case, reset reg ptr
        .db h'9,h'0c0   ; hardware reset, irpts off
        .db h'4,h'44    ; 16x clock,async,1 stop,no par.
        .db h'1,h'0     ; no dma, all irpts disabled
        .db h'2,h'0     ; irpt vector (for future use)
        .db h'3,h'0c0   ; rx 8 bits, rx disabled
        .db h'5,h'60    ; tx 8 bits, tx disabld, RTSA hi
        .db h'9,h'1     ; status low, irpts off
        .db h'0a,h'0    ; nrz encoding
        .db h'0b,h'50   ; no xtal, BRG->rxc txc, TRxC in
        .db h'0c,h'18   ; BRG lo byte - 4800 baud at 16x
        .db h'0d,h'0    ;hi byte - w/ 4 MHz BRG clk
        .db h'0e,h'2    ; DTR pgm'd, BRG from PCLK
        .db h'0e,h'3    ; as above, plus BRG enabled
        .db h'3,h'0c1   ; as above, plus rx enabled
        .db h'5,h'68    ; as above, plus tx enabled
        .db h'0f,h'0    ; no ext/sts interrupts
        .db h'10,h'10   ; reset ext/sts interrupts twice
        .db h'1,h'0     ; no dma, all irpts disabled

sccbtbl: .dw sccbcmd    ; port address
        .db 31          ; 31 bytes follow
        .db h'0         ; just in case, reset reg ptr
        .db h'4,h'44    ; 16x clock,async,1 stop,no par.
        .db h'1,h'0     ; no dma, all irpts disabled
        .db h'3,h'0c0   ; rx 8 bits, rx disabled
        .db h'5,h'60    ; tx 8 bits, tx disabld, RTSB hi
        .db h'0a,h'0    ; nrz encoding
        .db h'0b,h'50   ; no xtal, BRG->rxc txc, TRxC in
        .db h'0c,h'18   ; BRG lo byte - 4800 baud at 16x
        .db h'0d,h'0    ;hi byte - w/ 4 MHz BRG clk
        .db h'0e,h'2    ; DTR pgm'd, BRG from PCLK
        .db h'0e,h'3    ; as above, plus BRG enabled
        .db h'3,h'0c1   ; as above, plus rx enabled
        .db h'5,h'68    ; as above, plus tx enabled
        .db h'0f,h'0    ; no ext/sts interrupts
        .db h'10,h'10   ; reset ext/sts interrupts twice
        .db h'1,h'0     ; no dma, all irpts disabled
;       .db h'9,h'9     ; as above,plus irpt master enable

        .end
```


0 nS
50 nS
100 nS
200 nS
300 nS
400 nS
500 nS
600 nS
68809 CPU
8 MHz OSC
LSTTL LOGIC
68809 E
Q
ADDR, R/W
RD data
WR data
74S189 outputs (mapped address)
WRMAP\ signal to 'S189: OEPROM\ signal
IOZONE\ and ONBOARD\ signals (if selected)
IOn\ select outputs (from 'LS138)
RD\ and WR\ strobe outputs (from 'LS138)
OFFBD\ signal
<210 min>
<220 min>
80
125
<15>
valid
<20>
read data must be valid:
<40>
10
must remain valid 10nS after E falls
<110>
write data valid:
<30>
<35 max>
trails E 10-15 nS
<25>
<32>
trails E 14-26 nS
bus wait logic:
REQ
'175 Q\
STRETCH\
trails E 15-20 nS
'175 CLR\
MRDY
must assert MRDY 110 nS before E falls
When bus is already acquired (no wait):
DRIVENBL\ signal
Address to bus
RD\ and WR\ strobes on bus
Read data
Write data
IORDY from bus (if asserted):
trails RD\, WR\ 12-18 nS
must be valid:
must be valid:
<39>
<110 nS before E falls>
lo addr valid>
hi addr valid>
wr data valid>
rd data needed>
lo addr <valid
wr data <valid
On-board RD\ and WR\ strobes
IOn\ selects
Z8530:
addr (lo)
<80>
addr (lo) must be valid
CE\ must be hi at least 100 nS
must be valid 0nS before RD\, WR\
must hold 0nS after RD\, WR\
RD\, WR\ width 195 nsec minimum;
Z8530-4 requires 240 nsec, -8 requires 180 nsec.
rd data must be held
<-10 nS after E high
read data valid:
<140>
Z8530-8
remains valid until E low + 'LS138 delay
<180>Z8530-6
<250>Z8530-4
write data required:
must hold 0nS after RD\, WR\
<-> wr data
NOTE valid
E
6522A:
(2 MHz)
addr (lo), CS\
<80>
must be valid
<----------> chip selects
NOTE valid
rd data NOTE
needed><>
read data valid:
valid 190nS after E high
remains valid 10nS after E low
write data required:
<80>
10
must hold 10nS after E low
Title
6809 MULTIPROCESSOR - CPU TIMING
Size
B
Document Number
PMTTIME.SCH
REV
Date: January 29, 1994
Sheet
of


U25A
1489
U12B
75175
U13B
75174
U26A
1488
U25C
1489
U12D
75175
U13D
75174
U26C
1488
U9E
74LS14
U9F
74LS14
J4
J5
JP12
JUMPER2
JP13
JUMPER2
HEADER 5X2
+12V
-12V
U11
Z8530
OPTIONAL
JP8
JUMPER2
J8
HEADER 4
AUX I/O
U29
74LS688
U30
74LS259
PROGRAMMABLE INTERRUPT GENERATOR - BUS MASTER ONLY
NOTES
1. VCC AND GND TO ALL DEVICES NOT SHOWN
INSTALL U12/U13 FOR RS-485 SERIAL
INSTALL U25/U26 FOR RS-232 SERIAL
DO NOT INSTALL BOTH SETS OF DRIVERS!
T-RECURSIVE TECHNOLOGY, Hamilton, Ontario
Title
6809 UNIPROCESSOR - SERIAL PORTS C & D
Size A
Document Number PMTSCC2.SCH
REV A
Date: May 11, 1994
Sheet 6 of 6


U25B
1489
U12A
75175
U13A
75174
U26B
1488
U25D
1489
U12C
75175
U13C
75174
U26D
1488
U9C
74LS14
U9D
74LS14
J2
J3
JP10
JUMPER2
JP11
JUMPER2
HEADER 5X2
+12V
-12V
U10
Z8530
JP14
JUMPER2
JP15
JUMPER2
JP7
JUMPER2
R6
10K
U23
74HCT377
D1
LED BARGRAPH
R7 220
R8 220
<-20mA
NOTES
1. VCC AND GND TO ALL DEVICES NOT SHOWN
INSTALL U12/U13 FOR RS-485 SERIAL
INSTALL U25/U26 FOR RS-232 SERIAL
DO NOT INSTALL BOTH SETS OF DRIVERS!
T-RECURSIVE TECHNOLOGY, Hamilton, Ontario
Title
6809 UNIPROCESSOR - SERIAL PORTS A & B
Size A
Document Number PMTSCC.SCH
REV A
Date: May 16, 1994
Sheet 5 of 6

Regular Feature

Group Reviews

SUPPORT GROUPS FOR THE CLASSICS

TCJ Staff Contacts

TCJ Editor: Bill D. Kibler, PO Box 535, Lincoln, CA 95648, (916)645-1670, GEnie: B.Kibler, CompuServe: 71563,2243, E-mail: B.Kibler@Genie.geis.com.

Z-System Support: Jay Sage,1435 Centre St. Newton Centre, MA 02159-2469, (617)965-3552, BBS: (617)965-7259; E-mail: Sage@ll.mit.edu. Also sells Z-System software.

32Bit Support: Rick Rodman, BBS:(703)330-9049 (eves), E-mail: rickr@virtech.vti.com.

Kaypro Support: Charles Stafford, 4000 Norris Ave., Sacramento, CA 95821, (916)483-0312 (eves). Also sells Kaypro upgrades, see ad inside back cover. CompuServe 73664,2470 (73664.2470@cis).

S-100 Support: Herb Johnson, CN 5256 #105, Princeton, NJ 08543, (609)771-1503. Also sells used S-100 boards and systems, see inside back cover.

6800/6809 Support: Ronald Anderson, 3540 Sturbridge Ct., Ann Arbor, MI 48105.

Regular Contributors:
Dave Baldwin, Voice/FAX (916)722-3877, or DIBs BBS (916) 722-5799 (use "computer", "journal", pswd "subscriber" as log on), Internet dibald@netcom.com, CompuServe 70403,2444.

Brad Rodriguez,Box 77, McMaster Univ., 1280 Main St. West, Hamilton, ONT, L8S 1C0, Canada, Genie: B.Rodriguez2, E-mail: b.rodriguez2@genie.geis.com.

Frank Sergeant, 809 W. San Antonio St., San Marcos, TX 78666, E-mail: fs07675@academia.swt.edu.

Tilmann Reh, Germany, E-mail: tilmann.reh@hrz.uni-siegen.d400.de. Has many programs for CP/M+ and is active with Z180/280 ECB bus/Modular/Embedded computers. USA contact Jay Sage.

Helmut Jungkunz, Munich, Germany, ZNODE #51, 8N1, 300-14.4, +49.89.9614574, or CompuServe 100024,1545.

Ron Mitchell, Apt 1107, 210 Gloucester St., Ottawa Ontario, Canada, K2P 2K4. GEnie as R.Mitchell31, or CompuServe 70323,2267.

USER GROUPS

Connecticut CP/M Users Group, contact Stephen Griswold, PO Box 74, Canton CT 06019-0074, BBS: (203)665-1100. Sponsors East Coast Z-fests.

Sacramento Microcomputer Users Group, PO Box 161513, Sacramento, CA 95816-1513, BBS: (916)372-3646. Publishes newsletter, $15.00 membership, meetings at SMUD 6201 S st., Sacramento CA.

CAPDUG: The Capital Area Public Domain Users Group, Newsletter $20, Al Siegel Associates, Inc., PO Box 34667, Betherda MD 20827. BBS (301) 292-7955.

NOVAOUG: The Northern Virginia Osborne Users Group, Newsletter $12, Robert L. Crities, 7512 Fairwood Lane, Falls Church, VA 22046. Info (703) 534-1186, BBS use CAPDUG's.

The Windsor Bulletin Board Users' Group: England, Contact Rodney Hannis, 34 Falmouth Road, Reading, RG2 8QR, or Mark Minting, 94 Undley Common, Lakenheath, Brandon, Suffolk, IP27 9BZ, Phone 0842-860469 (also sells NZCOM/Z3PLUS).

L.I.S.T.: Long Island Sinclair and Timex support group, contact Harvey Rait, 5 Peri Lane, Valley Stream, NY 11581.

ADAM-Link User's Group, Salt Lake City, Utah, BBS: (801)484-5114. Supporting Coleco ADAM machines, with Newsletter and BBS.

Adam International Media, Adam's House, Route 2, Box 2756, 1829-1 County Rd. 130, Pearland TX 77581-9503, (713)482-5040. Contact Terry R. Fowler for information.

AUGER, Emerald Coast ADAM Users Group, PO Box 4934, Fort Walton Beach FL 32549-4934, (904)244-1516. Contact Norman J. Deere, treasurer and editor for pricing and newsletter information.

MOAUG, Metro Orlando Adam Users Group, Contact James Poulin, 1146 Manatee Dr. Rockledge FL 32955, (407)631-0958.

Metro Toronto Adam Group, Box 165, 260 Adelaide St. E., Toronto, ONT M5A 1N0, Canada, (416)424-1352.

Omaha ADAM Users Club, Contact Norman R. Castro, 809 W. 33rd Ave. Bellevue NE 68005, (402)291-4405. Suppose to be oldest ADAM group.

Vancouver Island Senior ADAMphiles, ADVISA newsletter by David Cobley, 17885 Berwick Rd. Qualicum Beach, B.C., Canada V9K 1N7, (604)752-1984.

Northern Illiana ADAMS User's Group, 9389 Bay Colony Dr. #3E, Des Plaines IL 60016, (708)296-0675.

San Diego OS-9 Users Group, Contact Warren Hrach (619)221-8246, BBS: (619)224-4878.

ACCESS, PO Box 1354, Sacramento, CA 95812, Contact Bob Drews (916)423-1573. Meets first Thurdays at SMUD 59Th St. (ed. bldg.).

Forth Interest Group, PO Box 2154, Oakland CA 94621 510-89-FORTH. International support of the Forth language. Contact for list of local chapters.

The Pacific Northwest Heath Users Group, contact Jim Moore, PO Box 9223, Seattle, WA 98109-0223.

The SNO-KING Kaypro User Group, contact Donald Anderson, 13227 2nd Ave South, Burien, WA 98168-2637.

SeaFOG (Seattle FOG User's Group, Formerly Osborne Users Group) PO Box 12214, Seattle, WA 98102-0214.

OTHER PUBLICATIONS

The Z-Letter, supporting Z-System and CP/M users. David A.J. McGlone, Lambda Software Publishing, 149 West Hillard Lane, Eugene, OR 97404-3057, (503)688-3563. Bi-Monthly user oriented newsletter (20 pages+). Also sells CP/M Boot disks, software.

The Analytical Engine, by the Computer History Association of California, 1001 Elm Ct. El Cerrito, CA 94530-2602. A ASCII text file distributed by Internet, issue #1 was July 1993. E-mail: kcrosby@crayola.win.net.

Z-100 LifeLine, Steven W. Vagts, 2409 Riddick Rd. Elizabeth City, NC 27909, (919)338-8302. Publication for Z-100 (a S-100 machine).

The Staunch 8/89'er, Kirk L. Thompson editor, PO Box 548, West Branch IA 52358, (319)643-7136. $15/yr(US) publication for H-8/89s.

The SEBHC Journal, Leonard Geisler, 895 Starwick Dr., Ann Arbor MI 48105, (313)662-0750. Magazine of the Society of Eight-Bit Heath computerists, H-8 and H-89 support.

Sanyo PC Hackers Newsletter, Victor R. Frank editor, 12450 Skyline Blvd. Woodside, CA 94062-4541, (415)851-7031. Support for orphaned Sanyo computers and software.

the world of 68' micros, by FARNA Systems, PO Box 321, Warner Robins, GA 31099-0321. E-mail: dsrtfox@delphi.com. New magazine for support of old CoCo's and other 68xx(x) systems.

Amstrad PCW SIG, newsletter by Al Warsh, 2751 Reche Cyn Rd. #93, Colton, CA 92324. $9 for 6 bi-monthly newsletters on Amstrad CP/M machines.

Historically Brewed, A publication of the Historical Computer Society. Bimonthly at $18 a year. HCS, 2962 Park Street #1, Jacksonville, FL 32205. Editor David Greelish. Computer History and more.

IQLR (International QL Report), contact Bob Dyl, 15 Kilburn Ct. Newport, RI 02840. Subscription is $20 per year.

Update Magazine, PO Box 1095, Peru, IN 46970, Subs $18 per year, supports Sinclair, Timex, and Cambridge computers.

Other Support Businesses

Hal Bower writes, sells, and supports B/PBios for Ampro, SB180, and YASBEC. $69.95. Hal Bower, 7914 Redglobe Ct., Severn MD 21144-1048, (410)551-5922.

Sydex, PO Box 5700, Eugene OR 97405, (503)683-6033. Sells several CP/M programs for use with PC Clones ('22Disk' format/copies CP/M disks using PC files system).

Elliam Associates, PO Box 2664, Atascadero CA 93423, (805)466-8440. Sells CP/M user group disks and Amstrad PCW products. See ad inside back cover.

Discus Distribution Services, Inc. sells CP/M for $150, CBASIC $600, Fortran-77 $350, Pascal/MT+ $600. 8020 San Miguel Canyon Rd., Salinas CA 93907, (408)663-6966.

Microcomputer Mail-Order Library of books, manuals, and periodicals in general and H/Zenith in particular. Borrow items for small fees. Contact Lee Hart, 4209 France Ave. North, Robbinsdale MN 55422, (612)533-3226.

Star-K Software Systems Corp. PO Box 209, Mt. Kisco, NY 10549, (914)241-0287, BBS: (914)241-3307. 6809/68000 operating system and software. Some educational products, call for catalog.

Peripheral Technology, 1250 E. Piedmont Rd., Marietta, GA 30067, (404)973-2156. 6809/68000 single board system. 68K ISA bus compatible system. See inside front cover.

Hazelwood Computers, RR#1, Box 36, Hwy 94@Bluffton, Rhineland, MO 65069, (314)236-4372. Some SS-50 6809 boards and new 68000 systems.

AAA Chicago Computers, Jerry Koppel, (708)681-3782. SS-50 6809 boards and systems. Very limited quanity, call for information.

MicroSolutions Computer Products, 132 W. Lincoln Hwy, DeKalb, IL 60115, (815)756-3411. Make disk copying program for CP/M systems, that runs on CP/M sytems, UNIFROM Format-translation. Also PC/Z80 CompatiCard and UniDos products.

GIMIX/OS-9, GMX, 3223 Arnold Lane, Northbrook, IL 60062, (800)559-0909, (708)559-0909, FAX (708)559-0942. Repair and support of new and old 6800/6809/68K/SS-50 systems.

n/SYSTEMS, Terry Hazen, 21460 Bear Creek Rd, Los Gatos CA 95030-9429, (408)354-7188, sells and supports the MDISK add-on RAM disk for the Ampro LB. PCB $29, assembled PCB $129, includes driver software, manual.

Corvatek, 561 N.W. Van Buren St. Corvallis OR 97330, (503)752-4833. PC style to serial keyboard adapter for Xerox, Kaypros, Franklin, Apples, $129. Other models supported.

Morgan, Thielmann & Associates services NON-PC compatible computers including CP/M as well as clones. Call Jerry Davis for more information (408) 972-1965.

Jim S. Thale Jr., 1150 Somerset Ave., Deerfield IL 60015-2944, (708)948-5731. Sells I/O board for YASBEC. Adds HD drives, 2 serial, 2 parallel ports. Partial kit $150, complete kit $210.

Trio Comapny of Cheektowaga, Ltd., PO Box 594, Cheektowaga NY 14225, (716)892-9630. Sells CP/M (& PC) packages: InfoStar 1.5 ($160); SuperSort 1.6 ($130), and WordStar 4.0 ($130).

Parts is Parts, Mike Zinkow, 137 Barkley Ave., Clifton NJ 07011-3244, (201)340-7333. Supports Zenith Z-100 with parts and service.

The Computer Journal

Back Issues

Sales limited to supplies in stock.

Volume Number 1:

- **Issues 1 to 9**
- **Serial interfacing and Modem transfers**
- **Floppy disk formats, Print spooler.**
- **Adding 8087 Math Chip, Fiber optics**
- **S-100 HI-RES graphics.**
- **Controlling DC motors, Multi-user column.**
- **VIC-20 EPROM Programmer, CP/M 3.0.**
- **CP/M user functions and integration.**

Volume Number 2:

- **Issues 10 to 19**
- **Forth tutorial and Write Your Own.**
- **68008 CPU for S-100.**
- **RPM vs CP/M, BIOS Enhancements.**
- **Poor Man's Distributed Processing.**
- **Controlling Apple Stepper Motors.**
- **Facsimile Pictures on a Micro.**
- **Memory Mapped I/O on a ZX81.**

Volume Number 3:

- **Issues 20 to 25**
- Designing an 8035 SBC
- Using Apple Graphics from CP/M
- Soldering & Other Strange Tales
- Build an S-100 Floppy Disk Controller: WD2797 Controller for CP/M 68K
- Extending Turbo Pascal: series
- Unsoldering: The Arcane Art
- Analog Data Acquisition & Control: Connecting Your Computer to the Real World
- Programming the 8035 SBC
- NEW-DOS: series
- Variability in the BDS C Standard Library
- The SCSI Interface: series
- Using Turbo Pascal ISAM Files
- The Ampro Little Board Column: series
- C Column: series
- The Z Column: series
- The SCSI Interface: Introduction to SCSI
- Editing the CP/M Operating System
- INDEXER: Turbo Pascal Program to Create an Index
- Selecting & Building a System
- Introduction to Assemble Code for CP/M
- Ampro 186 Column
- ZTime-1: A Real Time Clock for the Ampro Z-80 Little Board

Volume Number 4:

- **Issues 26 to 31**
- Bus Systems: Selecting a System Bus
- Using the SB180 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
- 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
- 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
- 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
- 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
- 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:

- 15 copies now available -

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.

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.

Issue Number 36:

- Information Engineering: Introduction.
- Modula-2: A list of reference books.
- Temperature Measurement & Control: Agricultural computer application. -
- ZCPR3 Corner: Z-Nodes, Z-Plan, 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.

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.

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-Systems.
- 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.

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 REL file.
- WordTech's dBXL: 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

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.

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.

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.

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.

Issue Number 45:

- Embedded Systems for the Tenderfoot: Getting started with the 8031.
- The Z-System Corner: Using scripts - MEX.
- The Z-System and Turbo Pascal: P 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.

Issue Number 46:

- Build a Long Distance Printer Driver.
- Using the 8031's built-in UART for serial communications.
- Foundational Modules in Modula 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

The Computer Journal Back Issues

- Long Distance Printer Driver: correction
- ROBO-SOG 90

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

ssue 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

ssue 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 ZCPR3 IOP
- PMATE/ZMATE Macros, Pt. 3
- Z-System Corner, PCED
- Z-Best Software
- Real Computing, 32FX16, Caches

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

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
- The NZCOM 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

Issue Number 53:
- The CPU280
- Local Area Networks
- Am Arbitrary Waveform Generator
- Real Computing
- Zed Fest '91
- Z-System Corner
- Getting Started in Assembly Language
- The NZCOM IOP
- Z-BEST Software

Issue Number 54:
- Z-System Corner
- B.Y.O. Assembler
- Local Area Networks
- Advanced CP/M
- ZCPR on a 16-Bit Intel Platform
- Real Computing
- Interrupts and the Z80
- 8 MHZ on a Ampro
- Hardware Heavenn
- What Zilog never told you about the Super8
- An Arbitary Waveform Generator
- The Development of TDOS

Issue Number 55:
- Fuzzilogy 101
- The Cyclic Redundancy Check in Forth
- The Internetwork Protocol (IP)
- Z-System Corner
- Hardware Heaven
- Real Computing
- Remapping Disk Drives through the Virtual BIOS
- The Bumbling Mathmatician
- YASMEM
- Z-BEST Software

Issue Number 56:
- TCJ - The Next Ten Years
- Input Expansion for 8031
- Connecting IDE Drives to 8-Bit Systems
- Real Computing
- 8 Queens in Forth
- Z-System Corner
- Kaypro-84 Direct File Transfers
- Analog Signal Generation

Issue Number 57:
- Home Automation with X10
- File Transfer Protocols
- MDISK at 8 MHZ.
- Real Computing
- Shell Sort in Forth
- Z-System Corner
- Introduction to Forth
- DR. S-100
- Z AT Last!

Issue Number 58:
- Multitasking Forth
- Computing Timer Values
- Affordable Development Tools
- Real Computing
- Z-System Corner
- Mr. Kaypro
- DR. S-100

Issue Number 59:
- Moving Forth
- Center Fold IMSAI MPU-A
- Developing Forth Applications
- Real Computing
- Z-System Corner
- Mr. Kaypro Review
- DR. S-100

Issue Number 60:
- Moving Forth Part II
- Center Fold IMSAI CPA
- Four for Forth
- Real Computing
- Debugging Forth
- Support Groups for Classics
- Z-System Corner
- Mr. Kaypro Review
- DR. S-100

Issue Number 61:
- Multiprocessing 6809 part I
- Center Fold XEROX 820
- Quality Control
- Real Computing
- Support Groups for Classics
- Z-System Corner
- Operating Systems - CP/M
- Mr. Kaypro 5MHZ

Issue Number 62:
- SCSI EPROM Programmer
- Center Fold XEROX 820
- DR S-100
- Real Computing
- Moving Forth part III
- Z-System Corner
- Programming the 6526 CIA
- Reminiscing and Musings
- Modem Scripts

Issue Number 63:
- SCSI EPROM Programmer part II
- Center Fold XEROX 820
- DR S-100
- Real Computing
- Multiprocessing Part II
- Z-System Corner
- 6809 Operating Systems
- Reminiscing and Musings
- IDE Drives Part II

Issue Number 64:
- Small-C?
- Center Fold last XEROX 820
- DR S-100
- Real Computing
- Moving Forth Part IV
- Z-System Corner
- Small Systems
- Mr. Kaypro
- IDE Drives Part III

Issue Number 65:
- Small System Support
- Center Fold ZX80/81
- DR S-100
- Real Computing
- European Beat
- PC/XT Corner
- Little Circuits
- Levels of Forth
- Sinclair ZX81

Issue Number 66:
- Small System Support
- Center Fold: Advent Decoder
- DR S-100
- Real Computing
- Connecting IDE Drives
- PC/XT Corner
- Little Circuits
- Multiprocessing Part III
- Z-System Corner

Issue Number 67:
- Small System Support
- Center Fold: SS-50/SS-30
- DR S-100
- Real Computing
- Serial Kaypro Interrupts
- Little Circuits
- Moving Forth Part 5
- European Beat

Issue Number 68:
- Small System Support
- Center Fold: Pertec/Mits 4PIO
- Z-System Corner II
- Real Computing
- PC/XT Corner
- Little Circuits
- Multiprocessing Forth Part 4
- Mr. Kaypro

Issue Number 69:
- Small System Support
- Center Fold: S-100 IDE
- Z-System Corner II
- Real Computing
- PC/XT Corner
- DR. S-100
- Moving Forth Part 6
- Mr. Kaypro

SPECIAL DISCOUNT

15% on cost of Back Issues when buying from 1 to Current Issue.
10% on 10 or more issues.

Subscriptions (CA not taxable)	U.S.	Canada/Mexico (Surface)	Canada/Mexico (Air)	Europe/Other (Surface)	Europe/Other (Air)
1year (6 issues)	$24.00	$32.00	$34.00	$34.00	$44.00
2 years (12 issues)	$44.00	$60.00	$64.00	$64.00	$84.00
Back Issues (CA tax)	add these shipping costs for each issue ordered				
Bound Volumes $20.00 ea	+$3.00	+$3.50	+$6.50	+$4.00	+$17.00
#20 thru #43 are $3.00 ea.	+$1.00	+$1.00	+$1.25	+$1.50	+$2.50
#44 and up are $4.00ea.	+$1.25	+$1.25	+$1.75	+$2.00	+$3.50

Items: _____

_____

_____ Back Issues Total _____

_____ Shipping Total _____

California state Residents add 7.25% Sales TAX _____

Subscription Total _____

Total Enclosed _____

Name: _____

Address: _____

_____

_____

_____

Credit Card # _____-_____-_____-_____ exp ____/____

Payment is accepted by check, money order, or Credit Card (M/C, VISA, CarteBlanche, Diners Club). Checks must be in US funds, drawn on a US bank. Credit Card orders can call 1(800) 424-8825.

TCJ *The Computer Journal*

P.O. Box 535, Lincoln, CA 95648-0535
Phone (916) 645-1670

Regular Feature

Editorial Comment

MY-Z-DEMO Plus

The Computer Corner

By Bill Kibler

I may be behind in getting to projects for *TCJ*, but lots of fun things have been happening. At work a few more unexpected trips to the job site happened, but they should all be done for now. Since that project is done, others are popping up. One of which is determining how to support 8051 code.

I would like to move to Forth based systems, as I feel the built in diagnostics would help resolve problems faster. The projects are already done in Intel 8051 ASM. The Intel assembler has many features and options that more current versions have since dropped. We have tried a few newer assemblers, and even one IDE (integrated development Environment - Turbo Pascal like edit/assemble/run) on Windows.

The results of testing indicate major editing would be needed of our older code for any of the newer assemblers. Why change then if the Intel version is working fine? The answer is not clear, but going to the IDE type of environment is suppose to improve overall efficiency. I rather question that idea, but then people who have not done 15 years of assembly, might find the IDE faster and easier to use.

There is no question that old style assemblers require a bit more inside knowledge of DOS and BATCH file usage. I have set up several assembly development work groups using just BATCH files. What is needed is a major effort to arrange your sub directories such that you can divide files, functions, and history files for easy manipulation by batch files. Of course the most important tool is not the assembler but a version control system.

Version control allows tracking and back tracking of changes to the source file. What I find most people not doing enough of is archiving off their versions of work. Whatever you do, you should store off your work both in archived areas of your disk, and on separate disks. You can buy version control programs that work very well for tracking when you reached this or that release stage. But simple archiving to other disks with clear labeling can work just fine. IDE's not having version control built in, are probably not worth considering.

When I am working on something very important, I may have four or five floppy disks, along with several zipped files. This file system gives me both archive and safety of mind. A close friend just wasted 3 weeks recovering data from a non backed up crash. He had taped it some months before, but not that day the machine crashed. He now follows my lead and users both tape (religiously) and floppy as well. With my system, the worse that could happen might be a few hours of lost work, easy to replace.

SOME MAIL

Got two interesting items in the mail this week. From Lee Bradley I received the MY-Z-DEMO disk. Now Lee only charges $10 for this disk. That is really a great bargain, and only the CP/M CD ROM will have more information in one place (Beta CD ROM completed - mastering in process as of early Nov.) on ZCPR. If you run a PC DOS machine this is the way to go. It is fast, already installed with Zsystem when un-zipped, comes with 22DSK, and 216 programs of mostly the "Z" type.

I took it to work the other day and over lunch used it to copy Emmanuel Roche's QX-10 formatted disk, and then unpack them with the programs supplied as well. It was simple, painless, and fast to use. What always surprises me is that many of these programs are better than any of their equivalent in MSDOS.

Now I know that Lee is not going to get rich on this, and in fact it really is just a service to help people get over the learning curve with Zsystem. Because of that and the fact that he really does want you to send the authors of 22DSK and MYZ80 money if you start using the program, I think you should drop the tenner in the mail ASAP. Here is Lee's address: Lee Bradley, 24 East Cedar Street, Newington, CT 06111-2534, USA.

The other item of interest was a book on using 8052 BASIC. Being an editor and getting an occasional free book com with the duties. The few I get are so much PC based and usually so bad I let the bugs have at them. The one from Lakeview Research was different for a change. It is a book our readers can probably understand and actually make use of.

I am not high on doing BASIC with 8051 or 8052's, but then I passed the beginner stage too long ago to remember. Jan Axelson, a writer for our competitors *The MicroComputer Journal* (names seems a bit familiar doesn't it..) has a good writing style and a strong grasp of getting beginners going.

The book, "*The Microcontroller Idea Book*" has about 270 pages of good size text and schematics of projects that should help you understand embedded

system without too much fuss. The topics are relevant and the BASIC code should have you paying more attention to the hardware or project than getting lost in assembly coding. Price is $31.95, plus $3.00 shipping from Lakeview Research, 2209 Winnebago St. Madison, WI 53704 (608) 241-5824.

One fact I picked out of Jan's book was BASIC being available in source form. In chapter 14 she talks about using external memory to run BASIC and explains how you can extend the BASIC in your own ROM. She states that Intel and Phillips BBS's have the assembly source and some vendors like Iota Systems have products with their extensions added to that source.

Since one of *TCJ*'s reluctance to use BASIC was lack of uniform source code, maybe here is an answer. Get the source from the BBS's and adapt it to Z80/6809/6800/??? systems. That would give us common code across 8051 to Z80 or more. Might entail some work replacing some smaller systems current BASIC, but then the user would have the source, which most don't have now. Sounds like an area the look further into to me.

And look I did the other day. On AMR's BBS I found the Intel version and two other. One looks a little smaller than the 8052, because it is for 8051's. The other is a version of Tiny Basic. Now Tiny Basic is a bit like Small C in that many versions have already been done for most CPU types. I believe I have a 68000 version around somewhere, and reasonably sure I have seen a Z80 version as well. This ROMable version has many good features and with source might be adaptable to all small systems. Think about it.

Buy the way, Jan's book has many sections you might find familiar if you subscribe to the *TMCJ*. It also made me consider one area not yet in CD ROM, embedded support. I have yet to see a CD ROM that is devoted entirely to supporting smaller systems. The Source CDs have all the cross assemblers and a few utilities, but no actual 8051 programs. I know that Motorola has the old 6800 users group programs on their BBS. But that material is many years old now, and yet 6805 or 6811 code samples are floating all around. Maybe I can twist the Walnut Creek people to do one, only a bit faster since embedded markets are moving faster than CP/M.

40MHZ Z80

I got my copies of the latest information from Zilog. A bit confusing and yet very amazing as well. One of the sheets was on the new Z380, with 4 sets of 32 bit Z80 compatible registers. 32 Bit addressing, 16 bit data paths, DRAM refresh controller built in. 100 pin package, with 25 MHZ speeds at 3 volts and 40 MHZ at 5 volts, wow.

Another booklet lists all the major Z80 related variations ZILOG produces. A rather good cross sample of integrated devices that should fill any application you might meet. I zeroed in on the embedded controllers to find the Z80L180 with a 33MHZ speed listed (at 3.3 volts). The same page shows the Z80380 with a 18 MHZ speed rating. The results is some leg pulling here, but in which direction I am not sure. I checked dates to see if one was a pre release, but none to be found.

One other device to catch my eye, was the Z85C80, which contains the 85C30 SCC (Serial Communications Controller - two serial ports) and the 53C80 SCSI controller in one package. Also some math chips that might make a great Forth engine at 40 MHZ (Z86193). I need to comment again, the speeds seemed to vary in this flyer a bit more than made sense. But high or lower, the selection is great and I hope the new Z380 might get more interest started in Z80's again, this time from the embedded people. Only time will tell.

Till Next Time..

Until the next issue and next year, please keep the letters coming and let me know if there is some direction not supported you need help on. Till then, keep hacking!

MYZ80 DEMO files list: -DEMO.001, -DEMO.002, ADIR.COM, ALIAS.CMD, ALIAS.COM, ARKZS.COM, AT.COM, AUTO.COM, BCOMP.COM, BEGIN.TXT, CD.COM, CHKDIR.COM, CHOP.COM, CHRS.COM, CL.COM, CLEANA.ZEX, CMD.COM, CMDRUN.COM, CMP.COM, COLDBOOT.COM, COLOUR.COM, COMP.COM, CONCAT.COM, COPY.COM, CPD.COM, CRCZ.COM, CRLZH.COM, CRLZW.COM, D.COM, DATSTP.COM, DEMO.ADV, DEMO.LTR, DEMO.NOT, DEMO.TXT, DDTZ.COM, DIR.COM, DIRBAR.COM, DMAP.COM, DOSDIR.COM, DRUNZ.COM, DSP.COM, DSTATS.COM, ECHO.COM, EDITND.COM, EDZCM.COM, ENVCFG.COM, ENVCFG12.CFG, ENVSRC.COM, EP.COM, EP.INI, ERA.COM, EXPORT.COM, FATCAT.COM, FATCAT2.CHN, FATCAT3.000, FATCAT3.001, FATCAT3.002, FATCAT3.003, FATCAT3.004, FATCAT3.005, FATCAT3.006, FATCAT3.CHN, FF.COM, FILEATTR.COM, FILEDATE.COM, FILESZ.COM, H.COM, HELPCH.COM, HELPCK.COM, HELPLSH.COM, HELPPR.COM, IF.COM, IMPORT.COM, INDEX.COM, JETLDR.COM, KEY.COM, KEYIN.COM, LBREXT.COM, LD.COM, LHC.COM, LHH.COM, LIFE.COM, LPUT.COM, LREPAIR.COM, LSH.COM, LSH.VAR, LSHINST.COM, LT.COM, LX.COM, MC.COM, MC.HLP, MCDEMO.MCS, MENU.COM, MENUCK.COM, MLOAD.COM, MOUSEP.COM, MOVE.COM, MYLOAD.COM, MYZ80.CLR, MYZ80.COM, MYZ80.KEY, MYZ80.NDR, MYZ80.Z3T, MYZ80GO.COM, NAME.COM, NTS.COM, NZBLITZ.COM, NZCOM.CCP, NZCPM.COM, OUTCAT.000, OUTCAT.001, OUTCAT.COM, PACK.COM, PACKLIST, PATH.COM, PAUSE.COM, PEEK.COM, PIP.COM, POKE.COM, PRINT.COM, PUTDS.COM, PWD.COM, QL.COM, QUATRIS.COM, QUIET.COM, QUIT.COM, RCOPY.COM, RCOPY.LST, RDUMP.COM, REDIR.COM, REG.COM, REMIND.COM, REN.COM, RESOLVE.COM, RLEPRT.COM, SAK.COM, SALIAS.COM, SAP.COM, SAVE.COM, SAVNDR.COM, SCAN.COM, SETFILE.COM, SHCTRL.COM, SHDEFINE.COM, SHFILE.COM, SHOW.COM, SHRINK.COM, SHSET.COM, SILENT.COM, SLOWDISP.COM, SORT.COM, SP.COM, SREN.COM, SSTAT.COM, STANDARD.CFG, STAT.COM, SUB.COM, TCAP.COM, TCSELECT.COM, TCSRC.COM, TCVIEW.COM, TERMINAL.COM, TIMEROMA.FN2, TPA.COM, TYPE.COM, UMAP.COM, UNARCZ.COM, UNCRLZH.COM, UNCRLZW.COM, UNERA.COM, UNSPOOL.COM, UNZIP.COM, UUDECODE.COM, UUENCODE.COM, VCED.COM, VERROR.COM, VLU.COM, VMENU.COM, VMENUCK.COM, W.COM, WHEEL.COM, XOX.COM, XOXINST.COM, Z3INTP.COM, Z3LOC.COM, ZCAL.COM, ZCNFG.COM, ZCRCK.COM, ZD.COM, ZDB.COM, ZDB23.CFG, ZDE.COM, ZDENST.COM, ZDT.COM, ZDT.DTA, ZERR.COM, ZEX.COM, ZF.COM, ZFILER.CMD, ZFIND.COM, ZGOLF.COM, ZLT.COM, ZMCONFIG.OVR, ZMINIT.OVR, ZMP.CFG, ZMP.COM, ZMP.FON, ZMP.HLP, ZMTERM.OVR, ZMXFER.OVR, ZP.COM, ZPLOT.COM, ZTIME.COM, ZTYPE.COM, ZXD.COM,

TCJ CLASSIFIED

CLASSIFIED RATES!
$5.00 PER LISTING!

TCJ Classified ads are on a prepaid basis only. The cost is $5.00 per ad entry. Support wanted is a free service to subscribers who need to find old or missing documentation or software. Please limit your requests to one type of system.

Commercial Advertising Rates:

Size	Once	4+
Full	$120	$90
1/2 Page	$75	$60
1/3 Page	$60	$45
1/4 Page	$50	$40
Market Place	$25	$100/yr

Send your items to:
The Computer Journal
P.O. Box 535
Lincoln, CA 95648-0535

Historically Brewed. The magazine of the Historical Computer Society. Read about the people and machines which changed our world. Buy, sell and trade "antique" computers. Subscriptions $18, or try an issue for $3. HCS, 2962 Park Street #1, Jacksonville, FL 32205

Wanted: Good complete floating-point package (IEEE single and/or double precision) for the 8051 Micro. Should be public domain, but commercial better than nothing. Send info to tilmann.reh@hrz.uni-siegen.d400.de.

For Sale: KAYPRO COMPUTER hardware and software. Kaypro II, Kaypro 4. Kaypro 4-84. All in good working condition with Turbo-ROMs and speed mods. Lots of software with documentation. Call Ted at 414-377-1846 and leave message, will return call.

Notice: Historically Brewed has moved. The new address is : 2962 Park Street #1, Jacksonville, FL 32205.

TCJ ADS WORK!

Classified ads in *TCJ*
get results, FAST!
Need to sell that special older system - TRY *TCJ*.
World Wide Coverage with Readers interested in what **YOU** have to sell.
Provide a support service, our readers are looking for assistance with their older systems - all the time.
The best deal in magzines, *TCJ Classified* it works!

SUPPORT OUR ADVERTISERS TELL THEM "I SAW IT IN TCJ"

TCJ The Computer Journal
Post Office Box 535
Lincoln, CA 95648-0535
United States

BULK RATE
US POSTAGE
PAID
Lincoln, CA
PERMIT NO. 91

ADDRESS CORRECTION REQUESTED
FORWARDING AND RETURN POSTAGE
GUARANTEED

Telephone: (916) 645-1670