ISSN # 0748-9331

THE COMPUTER JOURNAL®

Programming - Applications - User Support

Issue Number 25 $3.00U.S.

Repairing and Modifying Printed Circuits page 4

Z-Notes

Z-Com Versus the 'Hacker' Version of Z-System page 10

The C Column

Exploring Single Linked Lists page 14

Adding a Serial Port to the AMPRO Little Board page 19

The SCSI Interface

Building a SCSI Adapter page 23

NEW-DOS

Part 4: The CCP Internal Commands page 30

AMPRO 186 Column

Networking With Super DUO page 35

ZSIG page 43

The Computer Corner page 48

THE COMPUTER JOURNAL
190 Sullivan Crossroad
Columbia Falls, Montana
59912
406-257-9119

Editor/Publisher
Art Carlson

Art Director
Donna Carlson

Production Assistant
Judie Overbeek

Circulation
Donna Carlson

Contributing Editors
C.Thomas Hilton
Donald Howes
Jerry Houston
Bill Kibler
Rick Lehrbaum
Peter Ruber

The Computer Journal® is a bimonthly magazine for those who interface, build, and apply microcomputers.

Entire contents copyright © 1986 by The Computer Journal.

Subscription rate—$14 for one year (6 issues), or $24 for two years (12 issues) in the U.S., $22 for one year in Canada and Mexico, and $24 (surface) for one year in other countries. All funds must be in US dollars on a US bank.

Advertising rates—available upon request.

Change of address—please send your old label and new address.

Bulletin Board—Our bulletin board will be on line 24 hours a day at 1200 baud, and the number is (406) 752-1038.

Postmaster: Send address changes to: The Computer Journal, 190 Sullivan Crossroad, Columbis Falls, Montana, 59912.

Address all editorial, advertising and subscription inquiries to: The Computer Journal, 190 Sullivan Crossroad, Columbia Falls, MT 59912.

Editor's Page

TCJ's BBS Again

I should know better than to announce something which I expect to be ready by the time the magazine is out! In the last issue I said that our BBS was running, but as many of you know, it wasn't. This time I made sure that our BBS is really up and running, in fact I can look over my shoulder and watch it while I am writing this.

Thanks to the assistance of Rea Williams (Sysop Z-Node #10), and AMPRO, we now have the Little Board with a 10 Meg hard drive running the Remote Operating System (ROS). ROS was written in Turbo Pascal by Steve Fox (Sysop of the Albuquerque RCP/M, (505) 299-5974), and apppears to be well suited for our needs. One thing which I like is that everything is handled by ROS, and the caller never enters the operating system. We will be publishing a series of articles on running ROS on the AMPRO Little Board, and will offer an AMPRO format ready to run disk.

Now that it is running, we will be adding material as time permits. Check it out and leave a message for the SYSOP with your comment and suggestions. Charge card orders for disks, back issues, book orders, renewals, and subscriptions can be placed as a private message.

SOG '86

We attended Micro Cornucopia's annual SOG (Semi Official Get Together) in Bend, Oregon, and greatly appreciated the opportunity to spend three days talking computers. George Morrow of the former Morrow Computers gave a very interesting talk about where the industry is going, and Norman Kerth topped off the Saturday night banquet with "The Care and Training of Engineers."

In addition to the exhibitor's tables there were about 30 technical seminars, and the discussions lasted far into the night. Dave Thompson and the Micro Cornucopia staff obviously worked hard to present this well organized event, and are to be congratulated on a job well done!

The Ever Changing PC

Hal Hardenbergh (DTACK GROUNDED Flyer) has identified a new trend in computers, where we can have RAM capacity which is greater than the floppy disk capacity — he predicts that by the second half of 1987 the high-end personal computer could have a 32-bit processor and a minimum of 4 megabytes of RAM (expandable to 16 megabytes). He also predicts that three years after that the 4 megabit chips will arrive, so we will have a minimum of 16 megabytes expandable to 64 megabytes! This means that we will have to revise our programming concepts to deal with the entire program and data present in RAM instead of grabbing chunks of the data from disk. In issue #21, I predicted that future computers would eliminate the mechanical drives except for the original loading or backing up of files. With 16 to 64 megabytes of RAM most of us will be able to realize this goal.

One very important new development which George Morrow touched on was the creation of silicon foundries which offer custom chips from your disk file with 24 hour turn around. This is still brand new, and the cost of the software programs and manufacturing service are too high for the average individual's budget, but the prices will soon drop to the point where it is feasible to produce a custom chip for special commercial applications. Eventually, you and I will be able to design our own chips, accessing a library for the common elements, to build a customized hardware system.

Another notable trend is that they are continually putting more functions on a chip. This has lead to a packaging and reliability problem due to the high pin count, but as the trend continues the pin count will drop dramatically as we approach the goal of having the entire system on one chip. At this time the only

(Continued on page 42)

Letters From Our Readers

6502 Fan

I'm not sure who this should be addressed to, so I'll leave it up for grabs.

I own and hack on several Z-80 and several 6502 micros. I find in general that it is easier to write code for the 6502s, that I make fewer mistakes, that I use less memory, and that the code executes faster. I guess that the 6502 is the original RISC.

This leads to the question, why isn't CP/M or one of its descendents available for 6502s? In this connection, I note that SYBEX in 1980 advertised an 8080 simulator for 6502s (see Creative Computing, June 1980, page 179). Wouldn't such a program allow direct porting of CP/M to the 6502? What happened to the program — was it a flop?

D.E.

Morrow DJDMA User

Your editorial in issue #24 raised a sensitive issue to which I have no answer, but hope to share when you receive one. Specifically, the problem of adapting the Morrow DJDMA disk controller on my Decision One to read soft sectored disks. When I bought the unit early in 1983? the ads said that only a ROM change would be required, but when I tried to get the ROM I was told that it would require trading in the controller board to the tune of $250. I passed and have had to live with four different computers, each with a different disk format. Life would be much simpler if I could read and write soft sectored 5¼" disks on my Morrow Decision One.

G.E.

Editor's note:

There are a lot of DJDMA users with the same problem. We'd be very interested in an article (or even better a series) on the DJDMA disk controller.

Feedback On IBM Editorial

I read your editorial "Give IBM The Credit They Deserve" in issue #24. I disagree with some of your points. I don't believe that the pre-PC and post-PC demarcation has much to do with IBM. The term "PC", meaning "Personal Computer" or "Personal Computing", has been around since before IBM got into this market. I remember attending a computer convention in Dallas in 1977 which included a Personal Computer exhibition of over 100 Personal Computer vendors. IBM was not one of them.

Although there was much confusion in the beginning, this was largely due to the fact that it was a new industry with no history to learn from. However, eventually many standards were established such as Tarbel and KCC tape formats, CP/M, S-100, etc. Much of the early work that was done was trial and error, and the late comers such as IBM simply exploited the trials and tribulations of the pioneers.

It is true that "many people in the industry went their various ways instead of cooperating to design a standard easy to use product." We should all be thankful for that. It was these small but courageous companies which risked bankruptcy by experimenting with many different approaches that led to the eventual standardization that we now enjoy. And while this was happening, IBM sat back and waited for all the mistakes to be made before reinventing the wheel and putting their name on it. After all IBM's PC was based entirely on existing components and technologies, including PC-DOS which was originally developed by Seattle Computers, an S-100 manufacturer, and sold to Microsoft.

It would have been impossible, even for IBM, to "design a standard easy to use product" without the years of experimentation and development which was done prior to IBM's entry. It is interesting to note that although IBM sent its representatives to the meetings of the IEEE-696 (S-100) standards commitee, they did not announce their personal computer until after the standard was adopted. In effect, they led everyone to believe that they would support the standard knowing that they were going to sabotage it. Had IBM not come out with their personal computer, the rest of the industry certainly would have continued to progress and develop newer and better standards.

R.R.

More IBM-PC Feedback

Here are a few words in response to your editorial in Issue #24. I don't doubt the validity of your statistics regarding 8-bit CPUs, but remember that a very large majority of the majority is NOT being used for data processing in any conventional sense, but for controlling devices. Instead of finding many 8-bit CPUs in new computers these days, you'll be finding more of them in sewing machines and fuel injectors.

I don't doubt that most 8-bit machines are just as useful as they were when they were purchased (in fact, I've used that argument myself), and of the four computers in this house, three of them are 8-bitters. But just because an old adding machine still works like a brand-new doesn't mean I want to handle MY accounting with one today! 99+% of the serious computing that goes on here gets accomplished on the one MS-DOS computer.

I used to feel that I must defend my 8-bit computer, and I joined in the chorus that derided MS-DOS for being imperfect. Although I wouldn't have admitted it for all the world, I can see now that I was really only trying to convince myself that my significant investment wasn't wasted. When I finally was "forced" to purchase a PC-clone (I bought it AND a hard disk AND 640K of memory for less than it would have cost me to buy the hard disk for my Kaypro 4-'84) I found out the truth.

No, MS-DOS isn't perfect, neither are you or my minister. Yes, it IS without reservation better than CP/M. That's why dedicated hackers go to such lengths to replace CP/M with something a little more useable. Without a standard it's a struggle. I tried for endless weeks to get ZCPR3 to run on my machine, and

couldn't make it work because I had a new release of the operating system. When I finally did get it to work somewhat, I was never convinced that it was worth it. There were so many occesions when I had to cold boot because something wasn't configured just right, that I finally gave up and went back to plain-vanilla CP/M. Yes, I was ripe for a change.

Now it's comforting to be able to add another 21 Mbytes of disk to my conputer, if I want to, for $395, or an additional multifunction card for $89. To be able to configure a ram-disk if I want, just using a utility that's part of the PC-DOS I bought (I do that for my RBBS, so the menus, bulletins, and help files are fast). There's such a community of good programmers producing excellent Public Domain software for the PC now, that I don't miss ANYTHING from my CP/M collection. For that matter, I can even emulate a Z-80 with a CP/M 2.2 system, but I've only done it for curiosity's sake, never because I needed something that wasn't available for free for the PC.

The best programming editor I've ever seen is available for free (EDWIN), plus the best term program I've ever seen (ProComm v2.3). Public Domain programs like PS and DOSamatic partition memory to hold more than one program at a time (made possible by the easy availability of sufficient RAM). There are spreadsheets that damn-near duplicate Lotus 1-2-3, and database systems that keep dBASE III looking over it's shoulder, I'm sure. Because of the large installed base of compatible computers, shareware publishers like Jim Button and others provide a wealth of excellent software that can be tried-out before it's bought.

I don't wast time defending my 8-bit machines any more, or trying to make a patched-together operating system work. I spend it getting important things done with my inexpensive PC-clone.

J.H.

(Continued on page 44)

BD Software, Inc., maker of the original

CP/M-80 C Language Development

System, knows

Time is precious

So the compilation, linkage and execution speeds of BDS C are the fastest available, even (especially!) on floppy-based systems. Just ask any user! With 15,000 + packages sold since 1979, there are *lots* of users . . .

New! Ed Ream's RED text editor has been integrated into the package, making BDS C a truly complete, self-contained C development system.

Powerful original features: CDB symbolic source-level debugger, fully customizable library and run-time package (for convenient ROM-ing of code), XMODEM-compatible telecommunications package, and other sample applications.

National C User's Group provides direct access to the wealth of public-domain software written in BDS C, including text editors and formatters, BBS's, assemblers, C compliers, games and much more.

Complete package price: $150.
All soft-sectored disk formats, plus Apple CP/M, available off-the-shelf. Shipping: free, by UPS, within USA for *prepaid* orders. Canada: $5. Other: $25. VISA, MC, COD, rush orders accepted.

BDSoftware, Inc.

BD Software, Inc.
P O Box 2368
Cambridge MA 02238
617 • 576 • 3828

Repairing and Modifying Printed Circuits

by James O'Connor

Remember those little clouds above the heads of characters in a comic strip that contain the words being spoken, whenever a cartoon character needed to utter some unprintable oath the cartoonist would fill the cloud with *?+%& or †%@**!. Anyone who works with printed circuit boards (PCBs) for very long may one day look up and see one of those clouds floating above them. The cause could be any of a number of events, perhaps a two or three hundred dollar board sacrificed itself to protect a ten-cent fuse (one of Murphy's Laws). A lightning strike, a power surge, a misplaced scope probe shorted out, or perhaps that rarest of all events — you somehow bunged up the board yourself trying to unsolder a defective IC.

In the prior article in this series we took pains to emphasize the importance of evaluating the merits of trying to unsolder a part as opposed to the relatively safe method of cutting it off the board. Still there are times when unsoldering may be the method of choice, maybe you're not really sure the part is defective, or perhaps you need to make a modification. Just as once you begin to solder, you'll soon need to know how to unsolder, so also once you start unsoldering you'll soon need to know how to repair some of the minor damage that can result.

Fortunately there are quite a variety of tools and techniques that go a long way towards restoring the function of a board and even the appearance, in this article some of the most useful methods for the casual builder will be featured. Since many of these techniques apply both to modifications made on purpose and repairs made by necessity, this article combines both facets.

A great many artists and photographers have found printed circuit boards to be aesthetically pleasing, almost miniature surreal works of art. Most technicians realize that how a circuit looks rarely affects how it works (there are a few exceptions to this but mostly concern the original designer rather than the assembler or repairer). Still the manner in which a repair or modification is made can impact the future reliability of a circuit. Improper solder joints, dangling wires, or damaged circuit board material can all reduce the durability of a circuit. In this article we'll be careful to emphasize procedures that preclude this sort of problem. We're not trying for results that can't be seen, but results that when seen are judged sturdy and effective.

Since the features of printed circuits have a direct impact on the ways in which they can be repaired we'll start by wandering thru the geography of printed circuit boards, up and down the byways, following the trace jumpers, even diving into the substrata. Next we take a detour into the land of Mods, where they make deliberate changes to circuit boards. The land of Mods is inhabited by yet another type of jumper which we'll examine in some detail. Many of the methods we meet here are just as applicable in the next place we visit which is why we come this way. Finally we arrive at the destination of our travel the circuit repair depot, where the Flat Wires and the Glue-ons live.

Board Geography

Since the focus of this article is fixing printed circuit boards it may prove helpful to spend some time examining the geography and landscape features of boards. By the way the, traces on a printed circuit are sometimes called lands so use of the word landscape has more than a trace of truth in it.

If you start by assembling a simple kit or a kit that began as a feature construction article in a magazine then chances are good that you'll be working with a single sided board. No you haven't suddenly warped into a world with only two dimensions. In this case a single sided board is one that has circuit traces etched on only one side. These are very easy to manufacture and you can even purchase supplies or complete kits that let you make your own boards.

A circuit board starts out as blank which consists of a substrate material with a coating of copper over one side. The circuit pattern is then either mechanically or more often photographically imprinted onto the copper, where traces are to remain the copper is made resistant. The next step is to immerse the board into an acid solution that etches away the copper except where it is resistant to the acid. This isolates each signal carrying trace from all the others except where they are meant to connect.

On all but the simplest of boards some traces may need to cross over each other to get where they're going. Clearly this is impossible without shorting the signals from one trace to the crossing trace. To get around this problem, one trace is terminated at a donut pad like those used to mount component leads, this will later be drilled out along with all the other pads, on the other side of the crossing trace another pad is positioned and the first trace continues on from there. To join the two pads a short piece of bare wire (insulated wire may be used for long stretches) is soldered between them. The wire allows the signal to jump from one side of the board to the other and then back again, hence these are called 'jumpers'.

As fantastic as it may sound, jumpers have been the center of a mild controversy over the years. Some people claim that they can introduce distortions into the signals they carry, while others scoff at this notion. The truth lies somewhere in between, some circuits, dynamic memories for instance, can be affected by physical routing of the signals, but most circuits are not adversely affected. The real rap against jumpers is that they require labor to make and install and they preclude fully automated insertion of all components. For kits and prototype boards this is irrelevant, the kit assembler namely you and I will gladly supply the needed labor.

As boards grew in complexity and shrank in physical size the problem of jumpers became more acute. To eliminate them the modern double sided circuit board was developed, and the even more recent multi-layer boards. Double sided boards have traces etched on both sides, multi layers have that plus

one or more internal traces (like sandwiches). Wherever traces need to cross each other one trace is routed to the opposite side of the board and then back again, traces can shift as often as necessary. This flexibility permits a more orderly placement of the components. But lacking jumpers how are the traces electrically joined from side to side?

VIAs

A hole is drilled where traces need to join just like the holes used to mount the component leads, in fact all holes are drilled in one step. Then the inside surface of all the holes is plated with metal that bonds to the traces on each side and thus joins them together. Boards that have been so processed are said to have 'plated thru holes'. Where the plated holes join traces is called a via from the Latin word for road or way, the traces are joined by way of the via. The other holes are still called holes, nobody seems to have coined a buzzword for them, at least that I know of.

The manufacture of double or multi layer boards with plated thru holes is much more complex. The plating process is particularly critical since improper plating could result in a board that looks fine but won't work correctly after being assembled. To avoid this most board makers have special testing equipment that checks boards for electrical continuity. When you begin to assemble a kit you should examine such boards for an inspection stamp. You should also look for any visible defects especially in lieu of an inspection stamp. Boards with bad plating can be really tough to fix, not because it's hard to fix a badly plated hole but because it's hard to find them. A board which has been repaired is usually OK as it indicates that the board was tested and any bad spots corrected.

Another problem that can occur with vias themselves is that of the mis-placed lead. Sounds like the famous red-herring of a detective story, if you encounter one you'll need to be a pretty good detective to figure it out. It all started back when the board was being designed, the physical placement of parts was decided upon based on the parts in the maker's inventory. Once the board or kit went into production those original parts were all used up, new parts were then ordered. Most electronic parts have the same size and shape regardless of who made them but not always. Capacitors for example come in many shapes with different lead spacing while still being the same electronically. Most boards will have an outline in white indicating where the body of the part should be placed and most parts will fall right into place. But if a newer part has been substituted then it may need to be bent and prodded into place because it has a different lead spacing. Rarely a via may be lurking next to a component hole and the new part may slip into the via instead of the hole. It's only natural to think a part belongs where it fits best.

With a mis-placed lead the board will fail to function, it's just like having crossed wires or solder bridges, but much harder to spot. Many kit builders have had to pack up the board and return it to the maker, whose technicians are usually able to solve the problem in a flash, they probably get dozens back with the same problem. The poor victim of the mis-placed lead is then told that they put a part in wrong, which only adds to the mystery since they probably checked everything at least thrice. At least if they knew how many others had been ensared they would feel less chagrined.

Would you believe we could spend this much time talking about what are really just tiny holes. There is at least one more point about vias, they can be very handy to have around when you need to modify or repair a board. Because a via can be soldered to just as any other component hole you can route wires from a via to a lead. When you're instructed to modify a board and not given any specific route to follow (frequently the case) you should look for vias and use them when possible. When you have to run a wire some distance on the solder side of a board such a wire is vulnerable to damage. Sometimes glue is used to hold the wire down against the board, but you can also use vias to allow the wire to be routed mostly on the component side where it will be protected. Feed the wire right thru the via, naturally use insulated wire and don't solder it to the vias.

Board Substrates

The material used to make a circuit board is termed the substrate. In the article on unsoldering the two main types of substrate were mentioned because one type is more fragile and thus needs special handling during unsoldering.

Briefly, most circuit boards made in North America are made by companies who also make boards to be used by the Military, and such boards must be very tough indeed. These will have substrates made of glass filled epoxy, an amalgam of fiberglas and epoxy. Everyone gets such durable boards because it's more convenient for the board makers to produce only one type. Boards made in the Far East, typically for consumer goods, may have paper laminate substrates which are less expensive to make and work quite well in their proper environment.

Most computer and digital electronics boards will be of the epoxy type while most audio and video products tend to have paper-laminated boards.

The epoxy type is very rugged and

FREE CATALOG AND SOFTWARE APPLICATIONS GUIDE
CP/M MSDOS
AFFORDABLE ENGINEERING SOFTWARE
TRSD PCD
ANALYSIS
XFER $72.95
• Transfer Function Analysis
• Synthesize Circuits Functions
• Monte Carlo Sensitivities
LOCIPRO $72.95
• Root Locus Analysis
• Up to 25th Order
• Transients with SPP
SPP $72.95
• Signal Processing
• FFT Interface
• Transient Analysis
STAP $72.95
• 2D Thermal Analysis
• 1500 Nodes
CIRCUIT DESIGN
ACTFIL $72.95
• Active Filter Design
• Low, High, Band pass, Bandreject
• Chebychev, Butterworth
• Magnitude, Phase, Delay
ACNAP $72.95
• AC Circuit Analysis
• 10 Nodes, 200 Components
• Monte Carlo Sensitivities
• Transients with SPP
DCNAP $72.95
• DC Circuit Analysis
• 10 Nodes, 200 Components
• Monte Carlo Sensitivities
• ACNAP Compatible Files
GRAPHICS
PDP $72.95
• Pen Plotter Driver
• Linear, Logarithmic
PCPLOT $72.95
• High Resolution Graphics
• Linear, Logarithmic
• Bit Map Printer Dumps
PLOTPRO $72.95
• General Graphics Printing
• Any Computer, Printer
• Long Plots
TEKCALC $72.95
• Scientific Data Plotting
• High Res Graphics
MATHEMATIC
TEKCALC $72.95
MATRIX MAGIC $72.95
COMCALC $72.95
REPORT WRITING
Right Writer $97.95
BV Engineering
Professional Software
2200 Business Way, Suite 207
Riverside, CA 92501 USA
(714) 781-0252

withstands all sorts of accidents, traces can short out and burn right off the board with no physical damage to the board other than some charring. Epoxy boards can be flexed severely without breaking, although this is not true of the circuit traces which will crack. Moisture doesn't affect them at all. All the techniques outlined in this article can be used for epoxy boards if the area to be repaired is smooth and clean.

Paper laminated boards can suffer from accidents, burned out traces can delaminate part of the board. Flexing these boards can result in a fracture in the substrate, oddly enough the traces may still be intact, but not always. Moisture can be real trouble. The techniques used to repair traces in place can only be used on this type if the substrate is still sound. Otherwise the best approach is to use wires to replace or supplement the traces.

Modifications

At the same time that circuit boards were literally exploding in complexity and density the time allocated to development and testing was imploding, this leads to the problem of boards which have errors in logic or design that cannot be solved by producing new boards. That would be prohibitively expensive of both time and money — nor is it absolutely necessary if the errors are sufficiently few. Instead, the boards are modified. Ten years ago the casual builder would hardly ever need to modify a board, today it's a rather common problem. The amount of instructional aid in making the changes can vary widely, some kits simply advise you to make a change, others tell you where, and a few do both plus tell you how to go about it. As the techniques of modification are the same as some of those used to repair boards we'll cover the subject and provide some handy hints to make it easier.

Cutting A Trace

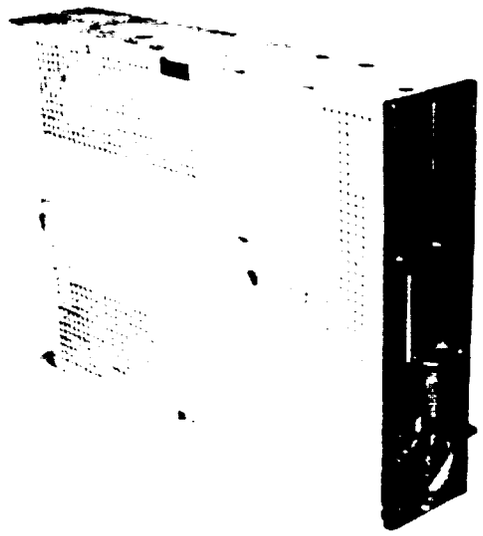
Almost always there will be an existing trace that must be severed from the circuit. Sometimes instructions will be provided telling you precisely where to sever it, but all too often this is not so. The first step then becomes one of finding the route the trace follows and selecting the best spot to sever it. New paths for the signals will be added to replace the severed trace, and one of the easiest ways to do this is to solder wire to a via. When you survey a trace check for via's and take advantage of them whenever possible. That is locate a cut after a via so that the new wire can start from the via. Otherwise simply select a spot that is accessable. Blank boards are slightly easier to work on than completed ones but the following methods apply to both.

The old carpenter's axiom of "Measure thrice, Cut once" should be kept in mind at this point. A good aid in selecting the right trace on very dense boards is to use the low ohms resistance setting of a Ohmmeter or a low voltage continuity checker. Consult the schematic, find the component at each terminus of the trace to be cut, and check for continuity between them. Then move one probe to the site where you've chosen to make the cut and test there for continuity. More than once by using this test I found that I was about to cut the wrong trace.

The recommended method for severing a trace is to cut through it with an X-acto type knife (any sharp easily manuevered utility knife will do). There is a special technique to this, plus here's a handy hint to make the job safer. Once you've identified the spot where you'll cut, take two pieces of masking tape and place them parallel to the trace on either side, trim the tape to fit if needed. Check once more with the continuity tester. The tape does two things, first it marks the spot. The second aid is that as you cut through a trace you'll find that the metal of the trace makes the knife blade drag, as soon as you finish the cut the blade will literally shoot off the trace and slice and scratch everything in its path. The tape will catch the blade and prevent this dangerous slide, a double layer of tape is even better.

In cutting the trace use three or four firm strokes to cut completely through the trace. Once you've cut through move the blade over about a sixteenth of an inch and make another cut. Angle the knife a little so the the two cuts form a 'V' once both cuts are complete the tiny chunk of trace between them should pop off the board. You may have to help it along with a few pokes from a small probe. Be sure to wear eye protection during this step. Check again with the continuity checker, if it reads open circuit you're finished.

There is another way to cut a trace that uses a special tool. This is a combination drill and router bit that is about the width of a standard trace. You simply drill into the right spot until the router part has removed the metal of the trace. These



SUPER DRIVES!!

8" SHUGART ½ Height DSDD

MODEL 860-1 MINT CONDITION WITH 1 FULL YEAR WARRANTY. THESE DRIVES DO NOT REQUIRE AC, TRACK TO TRACK ACCESS TIME 3MS, CAPACITY IS 1.6 MB.

CEC **$279.00**
P.O. Box 1965
Burlingame, CA 94010
(800) 228-3411

tools are great if you have to do a lot of trace cutting, but are expensive to buy and not easy to find. All the same precautions should also be observed when using this tool plus it should not be used on multi layer boards because it could accidently sever an internal trace.

On multi layer boards it may be impossible to sever a trace if it runs internally. In these instances you'll have to lift the lead out of the hole to disconnect the signal path, this is done just like you were unsoldering the part except you work on only one lead. For IC's in sockets just remove the chip, bend the right pin out and reinsertthe chip — sockets to the rescue again. To finish the job you may need to attach a wire to the lifted lead, this can be done by adapting the method outlined in the next part of this article.

Jumpering the Wild Circuit Board

Unlike a damaged trace, a misplaced trace doesn't need repair but complete substitution, for this you must use wire jumpers, here it's the signal rather than the trace that is 'jumpered'. This is a simple procedure but there are a few tricks that can make it go smoother.

To start with you'll need the right type of wire. For power or ground connections use the nearest equivalent solid hook-up wire. The solid wire is easier to form and shape, but stranded wire can be used if that's all you have or if the wire must flex. Always use wire that has at least the same current carrying capacity as the trace it is replacing. How do you determine that? One way is to simply take a small piece of wire and flatten it with gentle taps from a small hammer or very gentle taps from a large hammer. Do this on the anvil part of a vise or use any flat piece of heavy metal as an anvil. This is really mini-blacksmith type work, so you can always head for the nearest spreading chestnut and ask the local blacksmith for some advice. No local blacksmith? Then check your local library for a book on blacksmithing, it may be interesting to learn how much our forefathers knew about manipulating metal. The wire is large enough if you can spread it to at least three-quarters of the width of the trace to replaced. If in doubt use a little larger wire or run two parallel connections, because circuits can do wierd things when they don't receive enough power. I once saw a clever technician cure a flaky circuit by simply paralleling a large piece of wire with the existing power trace, it happened that the trace itself was not large enough. Sometimes such traces reveal themselves by simply burning up so always use a well sized wire in place of a burned trace.

Ninety five percent of the time, jumpers are for logic level signals — five, twelve, or maybe fifteen volt signals that don't need to carry much current. For these the best wire is standard wire-wrap wire available from Radio Shack Cat. Nos. 278-501 thru 504. Of all the different types of wire I have used, wire-wrap is by far the best because it's solid wire so its stays where you put it but it's flexible enough to be put almost anywhere. Frequently the original trace may be larger than would seem adequate for wrap wire, this is a function of drafting PCB's, it's easier to draw a fat line than a skinny one. Study the schematic (making mods without a schematic handy is akin to do-it-yourself brain surgery) and wrap wire should work fine if you're sure that the trace is not a power supply lead.

If you buy some wire wrap wire it's also useful to buy the wire wrap tool, Radio Shack No. 276-1570. You can use this to help with the jumper connections plus it can be used with headers. Those are the rows of straight pins that are so common on circuit boards, they act as switches by means of little black interconnecting devices called shunts. Shunts are sneaky little rascals since they disappear all the time leaving you with a shortage. You can use wire wrap connections to replace them, especially the ones you know will hardly ever be altered.

To make a good neat jumper is easy, use the same method of identifying the proper leads to be jumpered as when cutting a trace, use bits of tape to mark the right pins. Use via's as jumper points when possible but you'll almost always have to connect to at least one lead.

Unsolder the leads (see the article on unsoldering), no need to be extra fussy, you just want to remove the solder fillet. Measure out the length of wire needed plus about an inch more for each end. Strip the ends as instructed with the wire wrap tool. Make a couple of wraps around the exposed lead, it will be too short for more than two or three wraps, let the excess wire just stick up. Do this regardless of whether the lead is a round one or flat like an IC lead. The tool will wobble and will not work as well as it would on a wire wrap pin, but the object is not to make a wire-wrap connection but to merely hold the wire in place. Route the wire as best you can to protect it and make it lay flat, use via's if possible to place most of it on the component side where it will be fully protected. Wrap the other end.

You want to be sure to keep these jumpers insulated as they will cross over other traces. This is why wrap wire is the best type, it has a good tough, thin insulation. But some jumpers may be very short, an inch or less, and striping the wire so that such a short piece of insulation remains is difficult. Here's one way to do that, first strip off about an inch and a half from one end (a half inch more than usual). Now cut the wire about one to two inches longer than needed. On the stripped end wrap the extra half inch around the jaws of a needle nose pliers, insert the other end into the stripping tool. Push down to cut through the insulation, now lift the wire out and rotate it around and push it into the stripping jaws again, this extra step trims the insulation around the entire circumference. Now holding with the pliers pull off the insulation. When it works right you'll have a very short piece of insulation on the wire, use your cutters to trim away the crumpled end where the pliers held it and proceed to make the jumper. All this is necessary because when trying to strip a small piece of wire the whole length of insulation may pull off, and I defy anyone to thread it back on again.

The wire wrap joints are very fragile and are not finished yet but you can test the alteration if you wish by handling the board very carefully. To complete the jumper, re-solder both leads right over the wrap joints and trim off the e wire to finish.

The other way to do this is the Q method, you can guess what the ini stand for. In this technique you just s about an eighth of inch of insulation, lay the bare wire over the solder t and apply a hot soldering iron. Once th solder liquifies the wire should fall into it remove the iron and let it harden Another name for this is 'tacking' th wire.

It sounds real easy and maybe it is, bu I've always found tacking to be, we tacky. First it's really hard to control th wire and keep it from moving about Holding it with your fingers is easy if yo have asbestos skin, otherwise you hav to use something like tape to keep steady and that works poorly at best. Th big gripe is that the wire always seems t stick out of the connection at an angl instead of laying flat and smooth. Man times these connections are weak eithe

because the wire doesn't fully seat into the fillet or the tacking step results in an improperly heated solder joint, a cold joint. At the very minimum, removing the solder fillet first and then placing the wire into position makes for better results. The Q & D approach is required for power and ground jumpers made with larger wire.

Wire wrapping the jumper into place may seem like overkill but I find that really speeds up the process and produces good looking jumpers. And the wire wrap joints are so minimal that they can be easily unsoldered should that ever prove necessary. In any case, if you've never done this process then practice first on a scrap circuit board and experiment with each method till you find the one you're comfortable with.

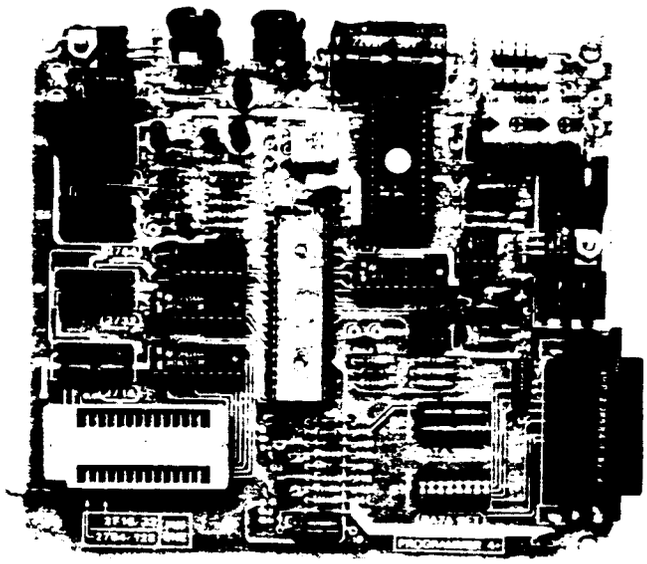
Lifted Traces

A lifted trace is one that has delaminated from the board but is otherwise undamaged, it looks fine except that it isn't stuck down to the board. Unsoldering is the chief culprit forthis condition. What happens is that a lead is not completely free from the trace and when the part is pulled out the trace is torn away from the board. Alternately unsoldering heat may cause the trace to seperate. It has also been my experience that the size and shape of the trace contributes to its tendency to lift off, so don't denigrate your technique unnecessarily when this happens to you.

There are only three possible fixes for this. The first is to simply insert the new part carefully manuevering the lifted hole over the lead and then pressing it down against the board. Then solder the lead to the trace, the solder joint becomes the bond that holds lead and trace in position. For very minor problems this works well, but it can be difficult to perform the procedure, when the iron is applied you'll find that the trace wants to stick to it and may lift away as you remove the iron.

The next best way to fix this condition is to restick the trace to the board. To do this you'll need some clear 5-minute epoxy. Mix a small batch of epoxy on a hard surface that can be easly cleaned, a piece of ceramic tile is excellent. Then use the flattened end of a toothpick or similiar tool to paint a thin coating of the epoxy under the trace. Next using a piece of wax paper press the trace down and hold it for a few minutes until the epoxy sets. All this must be done in one smooth

PROGRAMMER/4+
NEW LOWER SALE PRICE



A LOW COST SOLUTION TO EPROM PROGRAMMING

Reads and programs 2716, 2732, 2764, and 27128 EPROMS.
Reads 2-16K ROMS.
Direct connect to any RS232C terminal or computer.
Plug selectable as either a data set or data terminal.
All voltages made on board, (no power supplies needed).
(User supplies power Xformer, 25.2 to 30 VAC C.T.1 Amp.).
Power electronically switched, (can't damage EPROMS).
Zero insertion force socket for EPROM.
Programs, verifies, and dumps in both ASCII and hex.
Edit buffer (like DDT).
Saves hex and/or image files to and from disk.
Saves or loads all or partial buffer.
Completely menu driven for ease of operation.
Commands of Test, Read, Display, Save, Load, Program and more.
Check sum calculation.
All software on disk including well commented source code.
Both CP/M & MS-DOS systems supported.
Detailed owners manual including schematic.
All chips socketed.
Not a kit! Completely built and tested.
48 hour dynamic burn-in and test before shipment.
90 day limited warranty on parts and workmanship.
24 hour return policy on repairs.
Delivery from stock.

PROGRAMMER 4+ WITH OWNERS MANUAL AND DISK. $169.95
FOR EXTRA DISK FORMAT ADD $15.00

Order from

1659 Scott Blvd., Suite 1
Santa Clara, CA 95050
(408) 354-5084

VISA and MASTERCARD telephone orders welcome.

Please specify Disk format
CP/M 8" IBM format, KAYPRO II, XEROX 820, OSBORNE I, MS-DOS, etc.

Specify method of shipment, UPS or Postal Service.
California residents add 6% Sales Tax.

operation from the time you begin to mix the epoxy, because it will start to set up. That's why it's called 5-minute epoxy, so rehearse the procedure beforehand. Be sure to mix the epoxy correctly, do a test batch and check that it hardens within about 5 minutes time. Practice spreading a thin layer, too little and it won't stick, too much and the trace won't lay flat against the surface. Once the epoxy fully sets (at least ten minutes) carefully solder the new lead in place. Done carefully, it's almost impossible to know a problem occured.

The third way to fix a trace is to replace it entirely. Sometimes this is the best approach when it's obvious that the first two methods won't work or it's unnecessary to spend that much time and effort. There are two methods to replace a trace one is the traditional way and the other uses a clever new product.

Repairing With Wires

The traditional repair for a damaged trace is to simply run a wire from one lead to another in place of the the trace. This is very simple. First step is to cut off any part of the trace that has lifted away from the board. Then proceed exactly as though you were adding a modification jumper.

Sometimes the damage is just in one spot and you may want to bridge over the two good ends of a trace. One way is to simply solder over the ends if they are close together — a sixteenth of an inch or less apart. This is not recommended because solder is not considered a good conductor, this method may work for signal level voltages but should never be used for power or ground traces. The best approach which works even for wider gaps is to take a piece of bare solid wire long enough to bridge the gap plus a half inch additional. Using a small hammer flatten it on each side until it is about as wide as the traces to be joined, and trim it to overlap each trace end by about a quarter of an inch. Then solder it to each end. The trick is to somehow hold the piece of wire in position as you solder. A clamp such as a heat sink may work if the repair is near the edge of a board. Elsewhere that ever handy clamp 'gravity' may suffice especially for very tiny pieces. Just be sure that the board is level and that the soldering iron is placed on and then removed perpendicular to the board. Tape may do the trick for larger stretches — or even glue just be sure not to glue where the wire laps over the trace ends. Feel free to use any and all tools that your imagination can devise.

Glue On Traces

A very clever new product has come on the market in the last few years that makes repairs much easier and faster. The product called EZ Circuit is made by Bishop Graphics and while intended for building of prototype circuit boards, it works great for repairs also. It's actually a whole family of circuit board shapes and patterns such as donut shaped pads, elbows, IC DIP pads, etc. plus tapes in various widths to use as traces. All of these are made of copper just like real traces and have a special adhesive on the back. These items can be used to replace or even add a missing pad and trace.

To use them just be sure the surface of the board is clean and then cut the shape from the sheet and stick it down, for traces just unroll the tape starting with a little overlap at a pad shape and also at the end of the original trace. Then solder the lead into the new pad. When you solder to these pads be careful not to let the iron slide around, press down gently but squarely over the pad. The heat of the soldering will make the adhesive pliable and the iron could push the pad around at this time. Once the iron is removed the adhesive will re-adhere. Where a pad and a tape join, or a tape and elbow, or tape and original trace, you should solder each of these junctions following the same precaution about not sliding the iron on the joint.

For the casual builder only a couple of shapes and tapes are needed to accomplish most repairs. For instance a package of donut pads with an O.D. of 0.100" Cat. No. EZ7201, and a tape with a width of 0.050" Cat. No. EZ300501. My favorite shape is the dual terminal, even for replacing single pads. The most common problem encountered is a donut pad that has been delaminated along with a tiny bit of trace. Using the dual pad shape I can slip one pad over the lead and the other one under the partially lifted trace end. Next solder the lead and finish by soldering the trace to the other pad. The lead keeps things anchored and the other pad provides the bridge to the trace end. Sometimes too much trace has been lost so I just use either glue-on tape or a piece of flatened wire from trace end to extra pad. Dual terminals come in many sizes, the best general purpose size is an O.D. of 0.100", Cat. No. EZ1346.

A kit consisting of various shapes especially handy for board repair is available. Bishop Graphics also sells kits designed to allow the construction of prototype boards including many of the popular bus boards (S-100, PC Bus, STD etc.) and the parts left over from these kits can then be kept for future repairs. A catalog is available from the manufacturer and the products are sold by many electronics stores and mail order firms. Because they can be bought in small quantity, the EZ Circuit product would most suit the casual builder.

Another firm, Datak also sells a kit for repairing circuit traces, called Circuit-Fix. The kit includes a special cutting guide that lets you slice off adhesive-backed traces exactly as wide as needed. Donut pads in various sizes are also included. The pads and traces are then used in the same way as the Bishop Graphics products. The kit costs about $22 and you must start with the complete kit in order to obtain the cutting guide, extra supplies of donut pads and trace material are available at a cost of $4 dollars each. Due to its high intial cost the Datak product is most suited for someone doing extensive modifications.

Wrapup

One final cautionary word about repaired traces, all the methods outlined in this article should restore a circuit to the same level of reliability and performance as it had when new. Which means that if it was flaky before in all likelihood it will continue to be flaky, if it wasn't then there's no reason for the repair to make it start being flaky. Still strange things can and occasionally do happe trust nothing but rigorous and car testing. The one thing you can't expec a repair is that it will withstand being soldered. The original parts coul withstand that so why expect the re to do any better. If you must, ther ahead and unsolder, but be prepared repair it again.

Sources

You can write to Bishop Graphics for a catalog and list of distributors: EZ Circuit by Bishop Graphics, 5388 Sterling Center Drive, Westlake Village, CA 91359, Phone (213) 991-2600.

Datak's catalog including the Circuit Fix products can be obtained by writing to: The Datak Corporation, 65 71st Street Guttenberg, NJ 07093, Phone (201) 869 2200.

(Continued on page 4

Z-Notes

Z-Com Versus the 'Hacker' Version of Z-System

by Morris Simon, Ph.D.

The ultimate advantage of Echelon's Z-System® over other operating systems is choice. If you're accustomed to the restrictions of CP/M® 2.2, CP/M Plus or MS DOS® , the flexibility of Z-System will amaze you. If you're new to microcomputers, you'll be spoiled by the luxury of having user options for nearly everything in a fast, efficient 8-bit operating system. The first important choice you must make is the one between the commercial auto-install version ("Z-Com") and the bulkier manual-install public domain version floating around the world on Z-Nodes — the one I call the 'hacker' version with great respect for Rich Conn, its primary author and engineer.

This first choice is critical because it may determine which features of Z-System you can customize later. Generally, the Z-Com version is more difficult to change than the hacker version, but is far easier to install. That's just one of many tradeoff options available to you with Z-System. In this article, I compare the two varieties, describe their relative strengths and weaknesses, and offer some suggestions which might help you decide between them. I assume that you know what ZCPR3, ZRDOS and Z-System are, but let's be sure that we're thinking of the same things by each of those terms.

Some Quick Definitions:

Z-System — A specific term used by Echelon for the operating system itself (i.e., the combination of ZCPR3, ZRDOS and the CBIOS extensions of ZCPR3). The same phrase is also used more generally to cover all of Echelon's products designed for the ZCPR3-ZRDOS environment, including such things as libraries of Z80 subroutines, specialized utilities and tools, manuals and support items or services, such as Z-News (a fortnightly newsletter) and Z-Nodes (an international network of Z-System remote bulletin boards).

Z-Com — Joseph Wright's auto-install version of Z-System, consisting of ZCPR3, ZRDOS, and a large set of utilities.

ZCPR3 — Rich Conn's replacement for CP/M's console command processor (CCP). It consists of a 2K nucleus located (like the CCP) below the BDOS or ZRDOS, and a variable number of extensions into the CBIOS which strengthen ZCPR3's internal features and add powerful new memory-resident utilities.

ZRDOS — Dennis (not Z-Com's Joseph) Wright's optimized Z80 replacement for the CP/M 2.2 BDOS, consisting of all the original system calls (with some important changes) plus four new ones specific to Z-System. ZRDOS is optional with the manual-install version, but comes as part of the Z-Com package.

SYSLIB3 — Richard Conn's most general library of Z80 assembly language subroutines; used for any Z80 system development work under either CP/M or Z-System.

Z3LIB — Special Z80 subroutines derived from SYSLIB3 which were used to write ZCPR3 and ZRDOS as well as the various Z-System tools and utilities.

VLIB — Additional Z80 subroutines used in Z-System programs for video displays and effects.

The Hacker Version

When you install ZCPR3 manually, you must first edit a number of assembly language files in order to tailor the various system segments, buffers and even ZCPR3 itself to suit your needs and to fit within existing RAM locations used by both the operating system and your CBIOS. The finished product will be truly unique, even among users of identical hardware, because every decision you make regarding a particular system feature will influence both the space and locations allotted for other features.

You may choose to copy one of several memory patterns, such as the one illustrated in Figure 1, or you can design your own layout including some or all of the components shown. All addresses and sizes of system segments are left to you in the manual-install version, and you can redesign practically any of these Z-System features. You determine the contents of each segment or buffer by editing a cluster of Z80 assembly language files. You should have the following source code files in order to install a complete 'hacker' version of ZCPR3 (from Rich Conn's ZCPR3: The Manual, p. 242):

SYSENV.ASM	SYSFCP1.LIB	Z3BASE.LIB
SYSFCP.ASM	SYSFCP2.LIB	Z3BASE1.LIB
SYSIOP.ASM	SYSNDR.LIB	Z3BASE2.LIB
SYSNDR.ASM	SYSRCP1.LIB	Z3HDR.LIB
SYSRCP.ASM	SYSRCP2.LIB	Z3HDR1.LIB
ZCPR3.ASM	SYSRCP3.LIB	Z3HDR2.LIB
SYSENV.LIB	SYSRCP4.LIB	ZEX.ASM

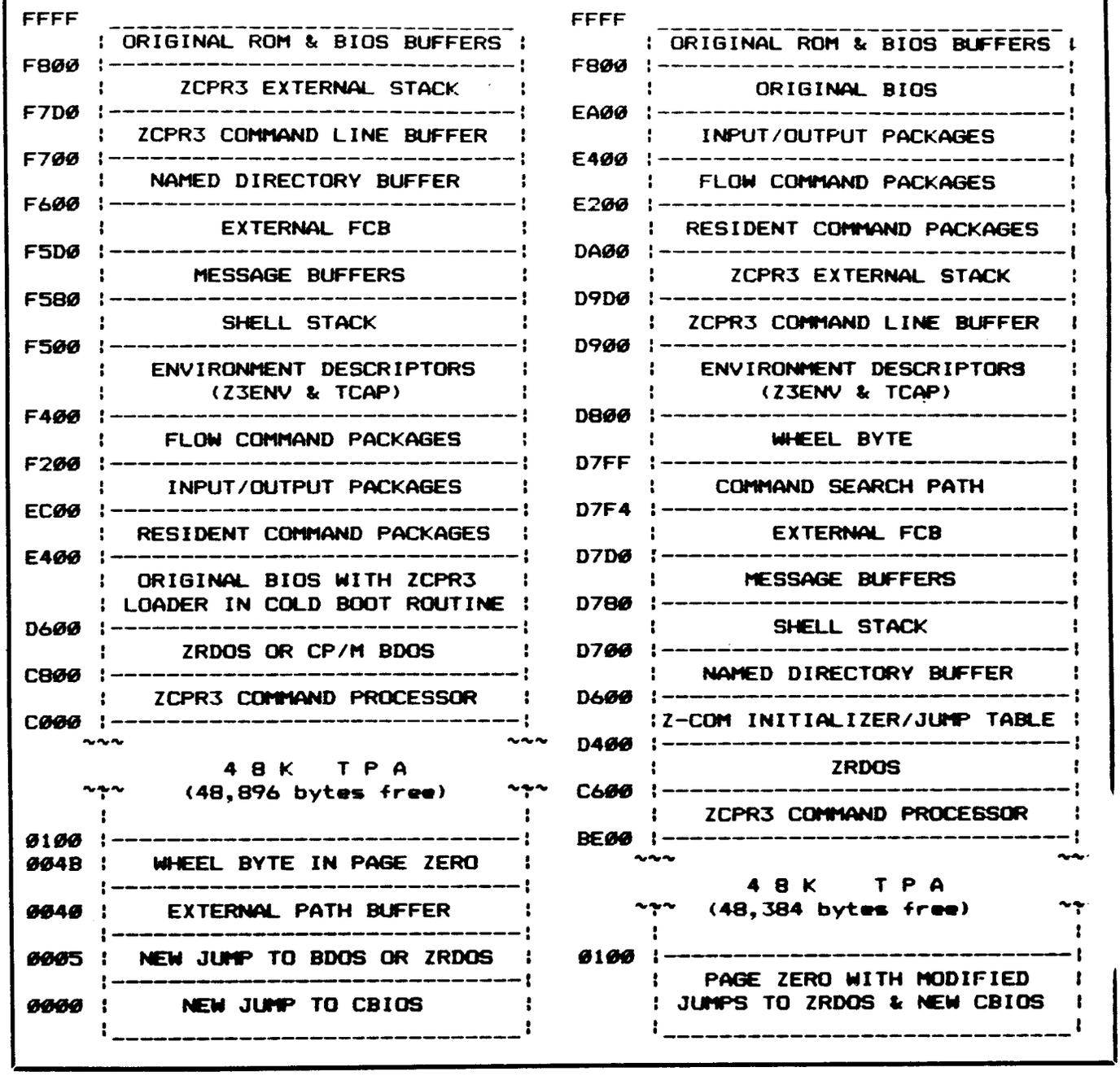
The only component not represented by its source code is Dennis Wright's optional ZRDOS, which is a proprietary item. Unless you're a very advanced Z80 programmer, you shouldn't tamper with it anyway; the object code version is powerful enough as it is. Everything else is there, ready for you to cut, customize and rearrange to fit your needs, whims or guesses. While you're redesigning the various system segments and buffers, you might even wish to modify ZCPR3 itself. That way, you can avoid duplicating internal ZCPR3 commands and features when you install more powerful RAM-resident external routines.

After modifying the segment source files, you must then assemble them into clusters of relocatable object code with your favorite Z80 assembler and accessories. Digital Research's Intel-based development package (RMAC/LINK/LIB) is adequate for public domain copies of ZCPR3 which are written in extended Intel mnemonics. For more recent versions in Zilog mnemonics (look for source code files with ".Z80" rather than ".ASM" extensions), either Microsoft's utilities (M80/L80/LIB) or Echelon's "Z-Tools" (ZAS/ZLINK/ZLIB) are excellent Z80 development packages. I favor the Echelon products because they're faster and compatible with mnemonic upgrades for the new Hitachi HD64180 and Zilog Z800 "superchips," which should keep you current in 8-bit programming for a while.

Finally, you'll need standard CP/M 2.2 system alteration utilities, such as the MOVCPM and SYSGEN utilities and a good

Figure 1

COMPARISON OF 'HACKER' & Z-COM MEMORY MAPS



debugger (or a powerful disk editor like DU, if you know how to use it). Conn also recommends some additional files to make things easier: Z3LOC.COM, Z3INS.COM, and ZEX.ZEX. If you're using either of Digital Research's SID or ZSID debuggers, a file called RELS.UTL will be useful.

To insert your new object code segments into their planned RAM locations, you must do some major surgery on the existing CP/M 2.2 system image. There are several ways to do this, and you'll have your favorite one. Mine is DU2, the disk-doctoring utility of ZCPR2, but that can be pretty tricky for beginners. You might wish to use the standard CP/M alteration and debugging utilities, MOVCPM, SYSGEN and DDT. The goals are the same, no matter how you do it:

(1) to lower the CBIOS by around 5000 bytes;
(2) to replace the CCP with ZCPR3;
(3) to replace the BDOS with ZRDOS [optional];
(4) to poke the ZCPR3 initialization code into the cold boot loader area of your original CBIOS in order to load all system segments and buffers; and (5) to replace the CCP and BDOS images on the system tracks with ZCPR3 and ZRDOS, if present, in order to have a bootable Z-System disk.

Z sets you free!

WHO WE ARE
Echelon is a unique company, oriented exclusively toward your CP/M-compatible computer. Echelon offers top quality software at extremely low prices; our customers are overwhelmed at the amount of software they receive when buying our products. For example, the Z-Com product comes with approximately 80 utility programs; and our TERM III communications package runs to a full megabyte of files. This is real value for your software dollar.

ZCPR3
Echelon is famous for our operating systems products. ZCPR3, our CP/M enhancement, was written by a software professional who wanted to add features normally found in minicomputer and mainframe operating systems to his home computer. He succeeded wonderfully, and ZCPR3 has become the environment of choice for "power" CP/M users.

Multiple Commands per Line
You can easily use multiple commands per line under ZCPR3. Simply separate the individual commands with semicolons. For example, "PIP B:=A:*.TXT;STAT B:*.*" will copy files and then show you the STAT results.

User-Programmed menu systems
ZCPR3 comes with three different menu systems that you can use to create custom menu-driven "front ends" for your computer. This is especially useful for setting up menus for your spouse or co-workers to use the computer, as they never have to see the A> prompt. All they have to do is press a single key to run any single or multiple CP/M programs, and when the task is done, control is automatically returned to the menu (ordinary CP/M menu programs cannot do this).

Extended Command Processing
When you type a command under CP/M, it will only look for the program in the current drive and user area. ZCPR3 gives you more flexibility by additionally searching other disks and user areas when resolving commands. You have full control of this function, called the PATH. This is probably the one element of ZCPR3 that is missed most if you return to "ordinary" CP/M.

Also, ZCPR3 supports the capability of grouping all your commonly used utility programs into a library file (*.LBR). This is great for systems with a small number of directory entries per disk, as the library file only uses one entry. It also has the advantage of reducing disk space requirements for a given set of programs, allowing you to put more programs on a disk. And the programs in the library file are invokable from the command line just like any other program not in the library.

Other Features
There's much more to ZCPR3, like named directories, online help system, etc., but it can't be described on one page. If you would like more information, consider the books shown below.

Z-SYSTEM
Perhaps the only shortcoming of ZCPR3 is that it is not a complete replacement for CP/M. This is what the Z-System does. The Z-System contains ZCPR3 and an additional module, ZRDOS, and is a complete replacement for CP/M. ZRDOS adds even more utility programs, and has the nice feature of no need to warm boot (^C) after changing a disk. Hard disk users can take advantage of ZRDOS "archive" status file handling to make incremental backup fast and easy. Because ZRDOS is written to take full advantage of the Z80, it executes faster than ordinary CP/M and can improve your system's performance by up to 10%.

INSTALLING ZCPR3/Z-SYSTEM
Echelon offers ZCPR3/Z-System in many different forms. For $44 you get the complete source code to ZCPR3 and the installation files. However, this takes some experience with assembly language programming to get running, as you must perform the installation yourself.

For users who are not qualified in assembly language programming, Echelon offers our "auto-install" products. Z-Com is our 100% complete Z-System which even a monkey can install, because it installs itself. Z-Com includes many interesting utility programs, like UNERASE, MENU, VFILER, and much more.

Echelon also offers "bootable" disks for some CP/M computers, which require absolutely no installation, and are capable of reconfiguration to change ZCPR3's memory requirements. At present, only Kaypro computers have this option available.

BOOKS
We sometimes joke around the office that we are really in the business of publishing, not selling software. We have books. Lots of books. We have to have lots of books, considering how powerful our software is and the large quantity of different packages we offer. Here are our best sellers:

ZCPR3: The Manual
This is the "bible" for the ZCPR3 user. An exhaustive technical reference, bound softcover, 350 pages. Contains descriptions of each ZCPR3 utility program, a detailed discussion about the innards of ZCPR3, and a full installation manual for those doing their own installation. You could order it from B. Dalton, but why? Get it from us.

The Z-System User's Guide
For those who are not technically inclined. This is an excellent tutorial-style manual filled with examples of how to use the power of ZCPR3/Z-System most effectively, written by two highly experienced Z users. (One user is a lawyer, the other a writer; this proves that anyone can use Z and benefit from it.)

ZCPR3: The Libraries
The extensive documentation for the libraries of ZCPR3, SYSLIB, Z3LIB, and VLIB. A must for any serious user of these programming tools. Loose-leaf notebook style; easy to work with as it will lay flat on your desk.

THERE'S MORE
We couldn't fit all Echelon has to offer on a single page (you see how small this type is). We haven't begun to talk about the many additional software packages and publications we offer. Send in the order form below and just check the "Requesting Literature" box for more information.

Item	Name	Price
1	ZCPR3 Core Installation Package	$44.00 (3 disks)*
2	ZCPR3 Utilities Package	$89.00 (9 disks)
3	Z3-Dot-Com (Auto-Install ZCPR3)	$99.00 (6 disks)*
4	Z3-Dot-Com "Bare Minimum"	$49.95 (1 disk)
5	Z-Com (Auto-Install Z-System)	$119.00 (7 disks)*
6	Z-Com "Bare Minimum"	$69.95 (2 disks)
12	PUBLIC ZRDOS Plus (by itself)	$59.50 (1 disk)
13	Kaypro Z-System Bootable Disk	$69.95 (3 disks)
20	ZAS/ZLINK Macro Assembler and Linker	$69.00 (1 disk)
21	ZDM Debugger for 8080/Z80/HD64180 CPU's	$50.00 (1 disk)
22	Translators for Assembler Source Code	$51.00 (1 disk)
23	REVAS3/4 Disassembler	$90.00 (1 disk)
24	Special – Items 20 through 23	$150.00 (4 disks)
25	DSD-80 Full Screen Debugger	$129.95 (1 disk)
27	The Libraries. SYSLIB, Z3LIB and VLIB	$69.00 (8 disks)
28	Graphics and Windows Libraries	$49.00 (1 disk)
29	Special – Items 27, 28, and 82	$129.00 (9 disks)
40	Input/Output Recorder IOP (IOR)	$39.95 (1 disk)
41	Background Printer IOP (BPrinter)	$39.95 (1 disk)
42	Programmable Key IOP (PKey)	$39.95 (1 disk)
43	Special – Items 40 through 42	$89.95 (3 disks)
60	DISCAT Disk cataloging system	$39.99 (1 disk)
61	TERM3 Communications System	$99.00 (6 disks)
64	Z-Msg Message Handling System	$99.00 (1 disk)
81	ZCPR3: The Manual bound, 350 pages	$19.95
82	ZCPR3: The Libraries 310 pages	$29.95
83	Z-NEWS Newsletter, 1 yr subscription	$24.00
84	ZCPR3 and IOPs 50 pages	$9.95
85	ZRDOS Programmers's Manual 35 pages	$8.95
88	Z-System User's Guide 80 page tutorial	$14.95

*Includes ZCPR3: The Manual

Echelon, Inc.
885 N. San Antonio Road, Los Altos, CA 94022 USA
415/948-3820 (order line and tech support)

NAME _____
ADDRESS _____
_____
TELEPHONE _____ DISK FORMAT _____
☐ REQUESTING LITERATURE

ORDER FORM

Payment to be made by:
☐ Cash
☐ Check
☐ Money Order
☐ UPS COD
☐ Mastercard/Visa:
_____
Exp. Date _____

California residents add 7% sales tax.
Add $4.00 shipping/handling.

ITEM	PRICE
_____	_____
_____	_____
_____	_____
Subtotal	_____
Sales Tax	_____
Shipping/Handling	_____
Total	_____

For added power, you might want to customize your favorite Z-System utilities as well. Those changes will require access to the major Z80 libraries (SYSLIB3, Z3LIB, and VLIB) as well as to the source codes for individual utilities. Depending upon how fancy you want the utilities to be, you might let them address various system segments, such as the TCAP descriptor for video effects, or the FCP for conditional operations.

That's it! Now you have a completely customized hacker version of Z-System. You'll be amazed at its efficiency and utility because it's an operating system carefully tailored to suit your exact requirements. Of course, the most important advantage is that you can change the operating system again anytime you like by loading special system segments designed for particular applications. Z-System will evolve with you! Unix is the only other operating system which might approach this degree of flexibility, but it's much too bulky to take advantage of Z80 elegance and efficiency.

Before you accept the challenge and fun of the hacker version, ask yourself a few questions and answer them as honestly as possible. Can you really handle a manual installation? Do you have the right software tools, time and skills to do the job? Is it worth the effort to customize your version of Z-System? As you'll read below, the auto-install version called "Z-Com" is less expensive, relatively foolproof and permits you to start using most of Z-System's power instantly.

The Z-COM Version

At $119, Joseph Wright's copyrighted auto-install version of Z-system actually costs around $50 less than Echelon's a la carte hacker version ($182 with ZRDOS, $133 without). The difference in price is due to the manual-install version's voluminous source code for all ZCPR3 components and the seventy-odd utilities. If you don't already have a good development package, you must add that cost as well to the price of the hacker version. If you answered "no" to any of the questions in the last paragraph, you probably should buy Z-Com and save yourself some expense, time, and trouble.

The entirely automated installation of a Z-Com version only takes around ten minutes, depending upon how many utilities you wish to install. The only files you'll need to edit are the Z3ENV and TCAP descriptors, which define special terminal and system attributes, and the list of utilities you wish to install with those attributes. At the beginning, you won't be able to customize your Z-System beyond these simple steps, but you won't be missing much power. By the time you explore the intricacies of Z-System well enough to know what additional features you want, you'll also know how to make other changes.

Figure 1 compares the memory structure of Conn's manual-install model in ZCPR3: The Manual (pp. 247-248) with that of a Z-Com version containing identical components except the initializing code. Note the differences in the location of the system segments and buffers. With Z-Com, the original CBIOS remains in place, eliminating the need for MOVCPM alterations, and all external ZCPR3 components are placed between the old CBIOS and the end of ZRDOS. The original BIOS jump table remains intact at EA00H, but the starting address of the modified CBIOS is shifted downward to D400H, where a new jump table relays all BIOS calls either to the original CBIOS routines or to new ones in the system segments. All additional system segments and buffers are sandwiched between these two BIOS jump tables, while ZRDOS and ZCPR3 sink lower in memory by 5600 bytes to make room for them.

The end result is virtually identical for either the hacker or Z-Com versions, give or take around 512 bytes. Digital Research's CCP and BDOS vanish, replaced by Echelon's more powerful operating system. The original CBIOS is still there, interwoven with the new ZCPR3 segments and buffers. Some of these additional features can be modified in place by standard Z-System utilities, while others will require either reassembly of source code or patching of object code. With the Z-Com version, only the patch approach can be used to customize the various object code files, either in place or as unloaded components. The Z-Node bulletin board network is beginning to distribute patches for most features which you may wish to customize.

The major operational difference between Z-Com and the hacker version of Z-System is the system boot. With Z-Com, you must maintain your Z-System files on a bootable disk with CP/M's loader, CCP, BDOS and CBIOS on the system tracks. In effect, Z-Com is a command file which CP/M's CCP loads at 100H like any other .COM file. It then copies itself into place (among other things) and erases the CP/M 2.2 system. You can even toggle between Z-System and CP/M by using a special ZCX.COM file, which calls the CP/M back from the disk's system tracks. With the hacker model, CP/M no longer exists, even on the system tracks, and you cold boot with Z-System itself.

Which Z-System is Right for You?

The initial trade off decision you must make is between ease of installation and maximum power. The differences in price may be considerable when you take the cost of assembly tools into consideration. Your computing habits will determine which of the two will suit you better. If you want to start using Z-System immediately, Z-Com is the better choice, but if you need to redesign any of the system segments or ZCPR3 itself, you should buy or download the hacker version of ZCPR3 and supplement it with ZRDOS to convert your environment completely to Z-System.

I enjoy playing with Zilog mnemonics, and pushing my system to its limits. For those reasons, I like having access to SYSLIB3, Z3LIB, VLIB and other subroutine libraries, as w as to the source codes of all Z-System components. If you're as fond as I am of assembly language programming and wan avoid the meticulous details of a manual installation, purch the Z-Com version. There are a few application programs wh I still run under CP/M 2.2, and Z-Com allows me to exit from re-enter Z-System for this purpose. You'll enjoy the elegant co venience and lower cost of the Z-Com model, which provides instant access to the powerful features of the most advanced eight-bit operating system available anywhere today.

If you want more details about Z-System, read this column regularly. In future issues, I'll show you the front streets and back alleys of this fine operating system, covering the topics I think you'll enjoy. Let me know your Z-System interests so that I can plan the column with them in mind. You may write to me directly or in care of The Computer Journal (include a self-addressed, stamped envelope if you want a reply):

Morris Simon
118 Brookhaven
Tuscaloosa, AL 35405

■■■

The C Column

Exploring Singly Linked Lists

By Donald Howes

Exploring Singly Linked Lists

This time around, I'll be looking at the use of linked lists. This type of data structure has many applications, such as stacks, queues and dynamic tables. The inherently dynamic structure of a linked list lends itself well to use in data base applications, or any code where the amount of data is not known in advance, or can vary within large limits. By dynamically allocating and releasing memory in a linked list, the programmer can avoid the memory overhead of allocating an array large enough to handle the maximum amount of data thought likely to be manipulated by the program.

Structure Allocation

As an example of a singly linked list, I'll be developing the code for a stack. The dynamic capabilities of this application are achieved through the use of pointers to structures as the elements of each list node. The stack nodes consist of a pair of pointers to an associated data structure and to the following node. All the "useful" information, that is, the actual data being placed on the stack, is contained in the separately defined data structure. This allows the stack code to be used in all circumstances, without regard for the type of data being pushed on the stack.

To create any linked list, three things are needed (Figure 1). First, the list needs a header, which will minimally tell the length of the list and the location of the first node (for other than stack applications, it is a good idea to include a pointer to the last item on the list as well). In addition, each node of the list must be declared, with a pointer to the data structure for that node and a pointer to the location of the next node on the list. Last, the tail of the list must be marked. This can be accomplished most easily by the use of a standard list node, where the pointer to the following node is declared as a NULL pointer.

Listing 1 gives the structures which are used to declare the stack header and the stack nodes. Both of them are very simple. The only thing to note is the declaration of the data pointer in the stack node. As stated above, this data structure can be anything (simple or complex) which is required to get the job done. The structure itself is declared in the program using the stack code.

A Look At Stacks

The usual way to give a visualization of how a stack operates is to draw the analogy with the type of plate dispenser usually seen in cafeterias. These little devices are spring loaded, so that, as the plates are loaded, the level of the stack of plates drops. This makes only the top plate available for use.

The analogy is a good one, since this is exactly the way a software stack operates. Only the uppermost value which has been placed on the stack is accessible, and data can only be added, or removed, from the top of the stack. A stack is, therefore, what is technically known as a LIFO (Last In, First Out) data structure. Of course, there is also a FIFO (First In, First Out) data structure, which is commonly known as a queue (in analogy to lines at bank windows or cashiers, although these queues generally move a lot faster).

Listing 1. Header File for Stack Code

```
/*****************************************************
*              stack.h                  *
*        Header file for stack code.        *
*****************************************************

#define   MALLOC(x)  ((x *)malloc(sizeof(x)))
#define   NULL     0

/****** typedefs for stack data structures *****/

typedef struct node{      /* structure for stack node */
  struct info *data;
  struct node *next_node;
}stk_node;

typedef struct head{      /* structure for stack header */
  int length;
  struct node *top;
}stk_head;
```

To manage a stack, the programmer has to be able to accomplish four things. First, allocate space for the stack header to start the stack. Second, allocate space for a stack node and push the node and associated data onto the stack. Third, pop data from the stack when needed and deallocate the space for the popped node. Fourth, deallocate the space for the stack header (and any left over stack nodes) after all access to the stack has been completed. This will free up all space used by the stack for other purposes.

The Code

The code for the stack functions are found in Listing 2. In the listing, the code actually relating to stack management is restricted to the last five functions. These are; crt_stk(), crt_node(), push(), pop() and del_stk(). The rest of the code in the listing is a short demonstration program to illustrate the use of the stack functions.

Starting at the top, there is the structure definition for the data structure used in the application. In this case, the structure contains only a single integer value, but could be made as complicated as needed, with a variety of variables and variable types. Following the structure is the list of error messages which will be used by the p_error() function. Following those are declarations for four of the stack functions. These functions

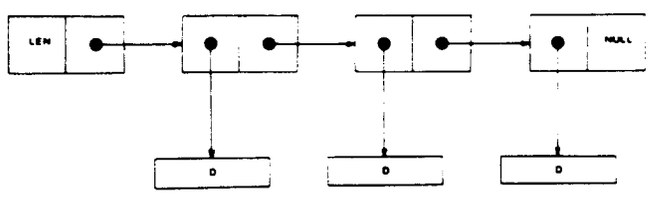


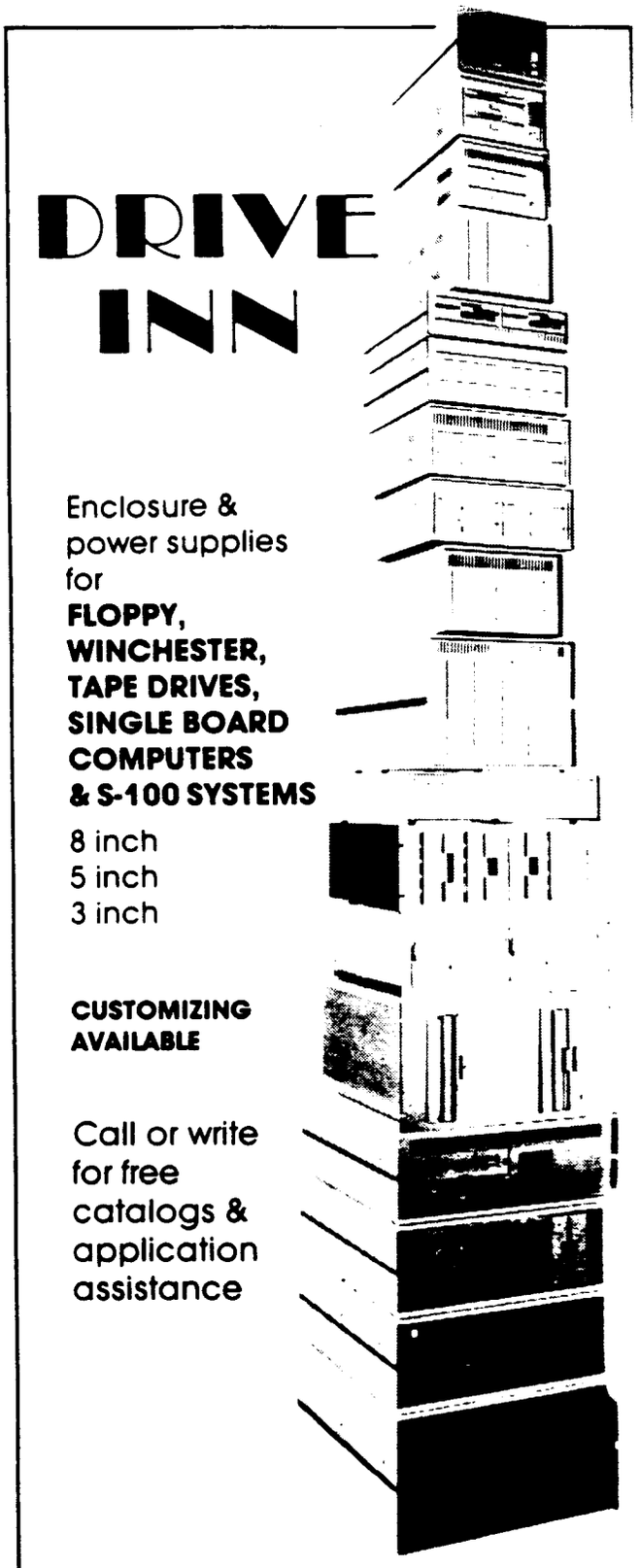
Figure 1

must be declared externally to the main() function for two reasons. First, if functions are not explicitly declared as to their return value (if that value is other than an int), then the compiler will assume that the return value for the function is an interger and make the appropriate changes to the actual return value to conform with those conditions. In other words, if you are using a function (say sqrt(), which returns a double) and do not declare that function to return a double, then an int will be returned. This can lead to some very interesting and hard to track down bugs, and is one of the common programming mistakes made by most, if not all, C programmers at one time or another. The second reason that these functions are declared outside of main() is related to scoping. The declaration outside of main() allows the functions to be scoped globally to all functions within the same file. If the functions had been declared inside of main(), then their use in functions other than main() [such as the call to crt_node() from inside of push()] would result in the same problem just described. The actual return value of the function wouldn't be known and a default value of int would be returned.

Moving on to main() itself, space for the stack header is created using crt_stk(). This function uses the macro MALLOC() which was defined in the header file (Listing 1). This provides a nice shorthand way to invoke the function malloc(), which is used to allocate memory (see your compiler documentation for a longer explanation). If the space for the header is not available, the function will return a NULL pointer and the program will terminate with an error message. If everything goes according to plan, the function will return a pointer to the location of the stack header, and we're in business. Although I didn't do it here, there is nothing to prevent the programmer from creating multiple stacks with this code, each stack being handled with a separate header variable. That is the nice thing about this type of stack building using pointer referencing, it allows the programmer to make maximum use of the memory available on the computer, without having to make decisions about memory allocation prior to run time.

Once the header has been allocated, the guts of the program run inside a single switch() function located inside of a while statement (note that braces are not necessary here, since only a single logical statement follows the while, even though that statement is relatively complex). The while calls the function menu() which displays a menu of valid operations. Since the while statement continues to loop until an 'S' is entered, the code is self checking in the case of invalid input. In the switch(), each case statement only looks for uppercase letters, since menu() translates input to uppercase before passing the value back to main().

Working through the switch() will take us through the rest of the program. Entering a 'P' from the menu will call the function get_data(), which allocates space for a data structure. If this space is available, the function input() is called, with an appropriate input request string and the MIN and MAX values for the input range. Input() does appropriate range checking, not returning a value until one within the declared range is entered. If no space was available, a NULL pointer is returned immediately to the function call in the switch() and the program is terminated with an error message. If everything works, then push() is called, being passed the new data structure and the stack header. One important thing to note here is that I am passing the actual structures to the function. This is in accordance with one of the changes being proposed by the ANSI C committee, but older versions of a compiler will not support this



feature. If you try to compile this code and it doesn't work properly, you will probably have to modify the function calls to pass pointer to the structures, rather than the actual structure. An example of this would be changing the call to push() from push(new,stk) to push(&new,&stk), everything should work properly then.

Once push() is called, it immediately calls crt_node(), which allocates space for the stack node and assigns the pointers for the location of the data structure and the next node in the stack. If a NULL pointer is not returned, the pointer for the top of the stack is set to that of the newly created node and the counter for the length of the stack is incremented. If a NULL pointer is returned, the value FALSE is returned to get_data(), which returns a zero (0) to the switch and causes the program to terminate with an error message (yeah, I know, it seemed a lot easier when I wrote the code). In (hopefully) simpler terms, what we have just done can be visualized using those little pop-together plastic beads. In inserting the new node at the head of the stack, we detached the node following the header (which is pointed to by the value of "top" in the header) and assigned that pointer to "next_node" in the new node structure. Once that was done, we assigned the pointer returned for the new node to "top", and everything was done.

Popping data from the stack is even easier, since there is less fiddling and error checking required (everything is already present, all we're trying to do is get at it). Entering "O" from the menu calls disp_data(). This function determines if anything is on the stack, then calls pop() and displays an appropriate message to explain the output and waits for the user to

CACHE22+CP/M 2.2
= CP/M Max!

CACHE22 is a front-end system program that buries all of CP/M 2.2 in banked memory. It helps 8080/Z80 computers to survive by providing up to 63.25K of TPA, plus the ability to speed disk operations, eliminate system tracks, and run Sidekick-style software without loss of transient program space. Complete source and installation manual, $50.00.

CP/M is a trademark of Digital Research Inc
Sidekick is a trademark of Borland International

MIKEN OPTICAL COMPANY
53 Abbett Avenue, Morristown, NJ 07960
(201) 267-1210

Listing 2. Source Code for Stack Application.

```
/*
  stack.c

  Copyright 1986 Donald Howes
    All rights reserved.

  This program may be copied for personal, non-commercial use
  only, provided that the copyright notice is included in all
  copies.

  Possible compiler dependent functions are:
  1. cursor()
  2. clrscr()
*/

#include  <stack.h>
#include  <ctype.h>
#include  <stdio.h>
#include  <malloc.h>

#define  MIN  -100
#define  MAX  100

/***** structure for data to be placed on stack *****/

typedef struct info{
  int data;
}stk_info;

static int *err_mess[3] = {
    'Can't create the stack.',
    'Could not push data onto the stack.',
    'Stack is empty, can't display anything.'
};

stk_head *crt_stk(), del_stk();
stk_node *crt_node();
stk_info *pop();

main()
{
  int c, test;
  stk_head *stack;

  stack = crt_stk();
  if (stack = = NULL)
  {
    p_error(0);
    exit();
  }

  while ((c = menu()) ! = 'S')
     switch(c){
      case 'P':
        if (!(test = get_data(stack)))
        {
          p_error(1);
          exit();
        }
        break;
      case 'O':
        disp_data(stack);
        break;
      case 'Q':
        query_stk(stack);
        break;
      default:
        putchar(' \ 007');
     }
  clrscr();
  del_stk();
} /* end of main */
```

```
int p__error(erno)
int erno;
{
  printf(err__mess[erno]);
} /* end of p__error */

int menu()
{
  int c;

  clrscr();
  cursor(10,20);
  printf('[P]ush data on the stack \ n');
  cursor(11,20);
  printf('p[O]p data from the stack \ n');
  cursor(12,20);
  printf('[Q]uery stack items \ n');
  cursor(13,20);
  printf('[S]top \ n');
  cursor(15,20);
  printf('select function letter : ');
  c = toupper(getch());

  return(c);
} /* end of menu */

void query__stk(stk)
stk__head *stk;
{
  clrscr();
  cursor(10,10);
  printf('%d items are on the stack. \ n',stk->length);
  cursor(12,10);
  printf('press any key to continue.');
  getch();
} /* end of query__stk */

int get__data(stk)
stk__head *stk;
{
  stk__info *new;

  clrscr();

  if (new = MALLOC(stk__info))
  {
    cursor(10,10);
    new->data = input('enter data value : ',MIN,MAX);
  }
  else
    return(new);

  return(push(new,stk));
} /* end of get__data */

int input(prompt,low,high)
char *prompt;
int low;
int high;
{
  int ival;
  printf('%s',prompt);

  while(1)
  {
    scanf('%d',&ival);
    getch();

    if((ival > = low) && (ival < = high))
      break;
    cursor(10,29);
  }

  return(ival);
} /* end of input */

void disp__data(stk)
stk__head *stk;
{
  stk__info *new;

  clrscr();

  if (stk->length)
  {
    cursor(10,10);
    new = pop(stk);
    printf('the value is %d',new->data);
    free((char *)new);
    cursor(12,10);
    printf('press any key to continue.');
    getch();
  }
  else
  {
    cursor(10,10);
    p__error(2);
    cursor(12,10);
    printf('press any key to continue.');
    getch();
  }
} /* end of disp__data */

stk__head *crt__stk() /* create a stack */
{
  stk__head  *new;

  if (new = MALLOC(stk__head))
  {
    new->length = 0;
    new->top = NULL;
  }
  return(new);
} /* end of crt__stk */

stk__node *crt__node(val,ptr) /* create a node on the stack */
  stk__info *val;
  stk__node *ptr;
{
  stk__node *new;
```


DISK DRIVE SERVICE

5¼"..... $35
8"..... $45

SERVICE SPECIALS

Apple II Drives..... $30
Shugart SA 400/400L..... $25
Shugart SA 800/801..... $25
Shugart SA 850/851..... $35

DRIVES FOR SALE

Shugart SA 800-2 (wide frame)..... $59
Shugart SA 850 (wide frame)..... $99
MPI 52S 5¼" DS/DD full ht..... $55
Tandon 100-2 DS/DD full ht..... $70
Tandon 100-1 SS/DD full ht. (new)..... $60
Apple II Drives..... $85
Genuine "IBM" (PC) floppy contr..... $60

60 day warranty on all drives and service. Turnaround time usually 24-48 hours. Trade-in available for drives too costly to repair. Prices do not include parts or shipping. If parts are more than $20 we get permission before repairing. Units returned UPS COD unless otherwise requested. All drives for sale are reconditioned unless otherwise noted and documentation is included.

LDL ELECTRONICS

13392 158 St. N., Jupiter, FL 33478 (305) 747-7384

```
  if(new = MALLOC(stk_node))
  {
    new->data = val;
    new->next_node = ptr;
  }
  return(new);
} /* end of crt_node */

int push(data,stk) /* push a node onto the stack */
  stk_info *data;
  stk_head *stk;
{
  stk_node *new;

  if(new = crt_node(data,stk->top))
  {
    stk->top = new;
    stk->length++;
    return(TRUE);
  }
  else
    return(FALSE);
}  /* end of push */

stk_info *pop(stk)  /* pop data off the stack */
  stk_head *stk;
{
  stk_node *temp;
  stk_info *data;

  temp = stk->top;
  data = temp->data;
  stk->top = temp->next_node;
  stk->length--;
  free((char *)temp);
  return(data);
} /* end of pop */

stk_head del_stk(stk) /* delete the stack header */
  stk_head *stk;
{
  stk_info *data;

  while(stk->length > 0)
  {
    data = pop(stk);
    free((char *)data);
  }
  free((char *)stk);
} /* end of del_stk */
```

press a key before returning to the menu. Pop() requires that a temporary node be created before the top of the stack is popped. What is done is the reverse of pushing a node onto the stack. The values of the first node of the stack are assigned to the temporary node and the value of "next_node" of the temporary structure is assigned to "top" in the header. A pointer to the data structure associated with temp is returned to the calling function and the space allocated to the node is freed.

The third menu option is to query the size of the stack. A call to query_stk() displays a message giving the length of the stack, which is determined from the value of "length" in the stack header. Finally, entering 'S' from the menu will stop the program, calling del_stk(), which deallocates the storage for any nodes remaining on the stack [by calling pop()] and that of the stack header. Under normal circumstances, there shouldn't be anything remaining on the stack when del_stk() is called, and you may wish to add some error checking code which will flag this condition if it occurs.

Erratum

I'd like to correct a mistake I made in my last column. In talking about how the program "strip" worked, I said that the bit pattern for 0x7F was "10000000". Anyone who added up the value of that pattern found it was equivalent to 0x80. The real bit pattern for the 0x7F mask is "01111111" and the masking process is a bitwise AND, not an add. Don't worry, the code works alright, I just didn't explain it very well. I guess it's a mistake to write these columns late at night. ■

* These listings are available on disk, or on the TCJ BBS (406) 752-1038.

Surplus Parts Resource

Here's a catalog any serious computer tinkerer needs. It's a treasure-trove of stepper motors, gear motors, bearings, gears, power supplies, lab items, parts and pieces of mechanical and electrical assemblies, science doo-dads, goofy things, plus project boxes, lamps, lights, switches, computer furniture, and stuff you might have never realized you needed.

All at deep discounts cause they are surplus!

Published every couple of months, and consecutive issues are completely different. Send $1.00 for next three issues.

JERRYCO, INC. 601 Linden Place, Evanston, Illinois 60202

Adding a Serial Port to the AMPRO Little Board

by Terry Hazen

My first computer had only a serial printer port, so to go with it I bought an Epson MX-80® printer and a serial adapter board with a high speed 64k serial print buffer. The buffer was especially nice for printing long documents without tying up the computer and ran at 9600 baud for speedy file transfers.

Later, when I upgraded my system to an AMPRO Little Board, I found I had only two serial ports and a terminal, printer, and modem to connect to them. I tried to cope by plugging and unplugging peripherals, but whenever I wanted to use the printer and the modem at the same time I had to bypass the serial buffer board and use the AMPRO parallel printer port. This slowed my printing considerably and tied up the computer during printing.

Finally, I just couldn't take it any more. I decided to design and build an adapter board to convert the Little Board parallel printer port into a serial printer port. The resulting adapter board plugs onto the Little Board parallel printer connector, J2. It takes its power from the Little Board through 4 unconnected pins on J2 and is wired to an RS-232C connector mounted on the back panel of the computer enclosure. A crystal controlled baud rate rate between 150 and 9600 baud is selected by a jumper on a double row pin header. The rest of the serial configuration, an 8 bit data word, 2 stop bits, and no parity, were selected during design and are set by connecting the appropriate UART pins to +5 volts. Hardware handshaking is used.

Circuit Operation

IC1, a General Instrument AY3-1015D Universal Asynchronous Receiver/Transmitter (UART), converts the 8 parallel data lines from the Little Board to a serial data stream. Two sections of a 74LS04 hex inverter, IC5, are configured as a 4.9152 MHZ crystal oscillator. A jumper selects the proper output from a 74HC4040 binary counter, IC4, to reduce the clock frequency to 16 times the required baud rate. An MC1488 line driver, IC6, converts the TTL serial data signal to RS-232C levels, and an MC1489 receiver, IC7, converts the RS-232C handshake signal from the printer to TTL levels.

When the power is turned on, IC1 and IC3 are reset by inverter IC5e, R1, and C1, setting the Little Board's BUSY line low, and telling it that the UART is ready for data. When the Little Board has data ready, it sends a low DS* pulse to IC1, telling it to load the parallel data bits into an internal holding register. The pulse also sets flip-flop IC3, holding the BUSY line high and telling the Little Board to wait before supplying more data. When IC1 finishes translating the parallel data bits into a serial data stream, it sets its End-of-Character (EOC) line high, resetting IC3 and setting the BUSY line low. Another DS* pulse from the Little Board will start the process over again for the next character.

The hardware handshaking signal from the printer is received by the RS-232C line receiver IC7 and translated into TTL levels. It is combined with the output of flip-flop IC3 to drive the Little Board's BUSY input line, telling the Little Board to wait whenever the printer is busy.

The adapter board requires +5 volts at approximately 150ma and ±9 to 15 volts at 20ma for the RS-232C line driver. I chose to take the power directly from the Little Board. I added jumpers from the Little Board power connector to J2 pins 23 and 24 for +5 volts, to pin 25 for +12 volts and to pin 26 for −12 volts. An alternative method is to use a +12 volt line to power an IC inverter, shown in Fig 2, that provides ±9 volts for the RS-232C line driver, IC6.

Construction

I chose to construct the adapter on a 2.5 inch × 3.5 inch single-sided PC board, applying resist directly with a marking pen and etching the board in a plastic zip-lock bag filled with warm etchant. On a subsequent project, however, I found that Datak JotDraft dry transfer pad patterns and tape produce a much neater and more trouble-free direct resist pattern. Datak Tinnit tin plating solution is a nice finishing touch. Wire wrap or point to point soldering are alternate construction methods.

Mount a double row 26 pin female header connector on the back of the adapter board to plug onto J2, which then supports the adapter board. All IC's are mounted on the front of the board and socketed for easier trouble-shooting. A double row header connector is provided for the baud rate selection jumper. Three wires run from the adapter board to the 25 pin female RS-232C connector mounted on the rear panel of the computer enclosure.

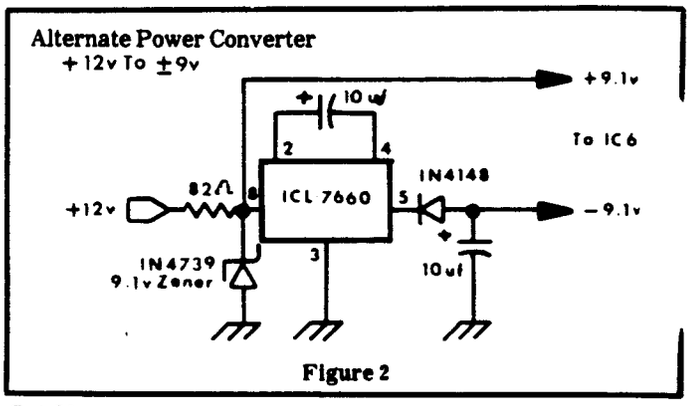


Figure 2

Set-Up

After constructing the adapter board and checking the wiring, and before plugging in any ICs, carefully test the power supply (or if you brought power up from the Little Board, carefully check ALL the power wiring to avoid blowing up any Little Board IC's). Plug the adapter board onto J2 on the Little Board and turn on the power. If each IC socket has the proper supply voltages, turn the power off and plug in the IC's. Use a jumper to select the desired baud rate. No other set-up is required. The printer now sees the Little Board as a serial port while the Little Board sees a parallel printer.

Note: In the text, a following '*' indicates a logical NOT (DS* indicates DS NOT).

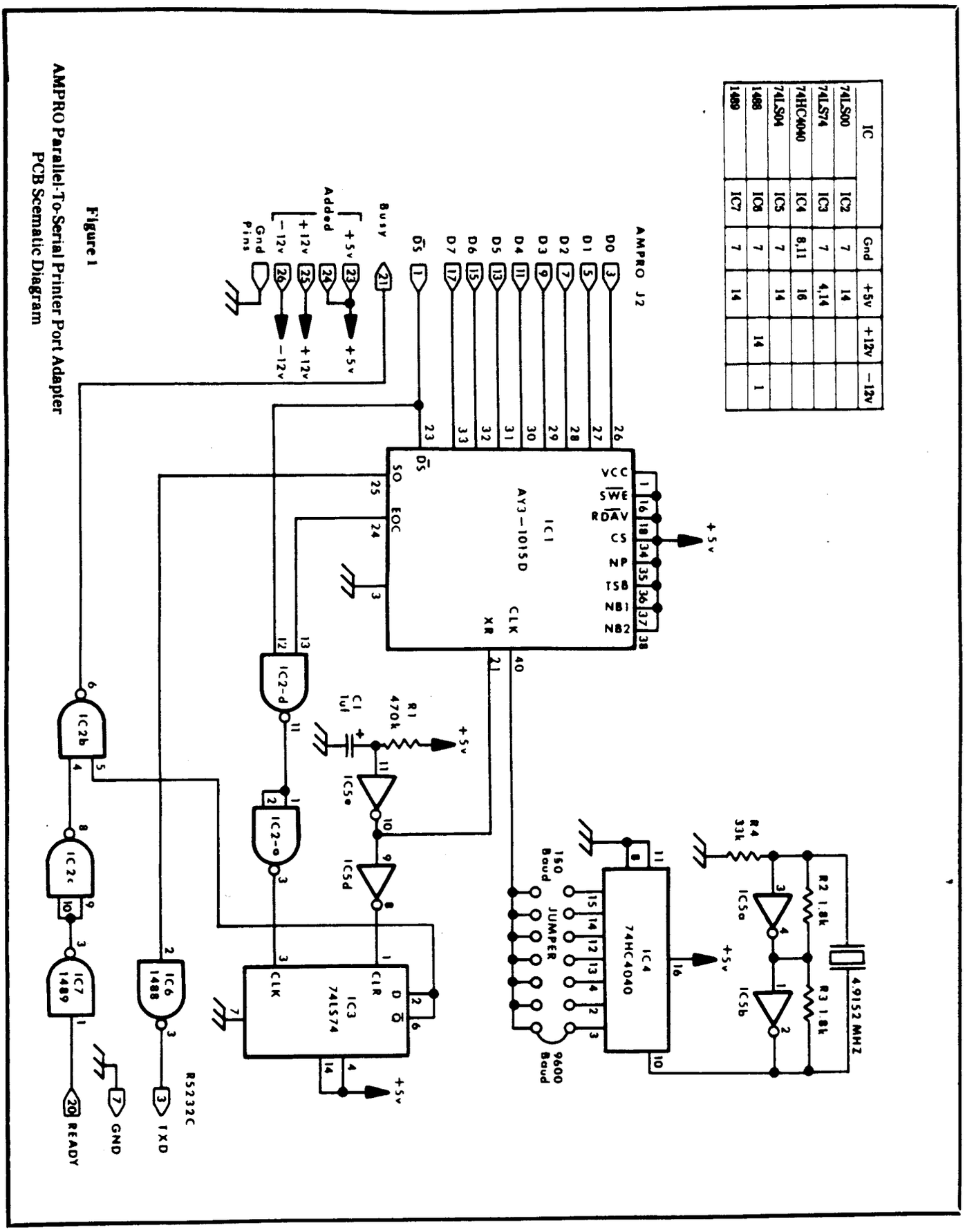


Figure 1

AMPRO Parallel-To-Serial Printer Port Adapter PCB Scematic Diagram

Serial Configuration Modifications

The characteristics of the serial bit stream can be changed by connecting some of the UART pins to either +5 volts (logic "1") or ground (logic "0".) UART pins 37-38 control the number of bits/character as shown in the following Table.

Pin 37	Pin 38	Bits/Character
0	0	5
0	1	6
1	0	7
1	1	8

A logic "0" on pin 36 will insert one stop bit while a logic "1" will insert two stop bits. A logic "1" on pin 35 will eliminate the parity bit while a logic "0" will supply a parity bit. If a parity bit is selected, a logic "0" on pin 39 will select odd parity while a logic "1" will select even parity.

One Port, Two Printers

If you have two printers (perhaps a daisy wheel at 1200 baud for final copy and a dot matrix at 9600 baud for draft work), you can add a 3-pole 2-position switch and a second baud rate jumper header to your apapter. One pole of the switch selects the baud rate header and the other two poles switch the data-out and handshake lines to the proper printer connector. The baud rate for each printer can be independently selected. The switch can be mounted on the rear panel near the printer connectors. ■

CTM

"...received my moneys worth with just one issue..."

—J. Trenbick

"...always stop to read CTM, even though most other magazines I receive (and write for) only get cursory examination..."

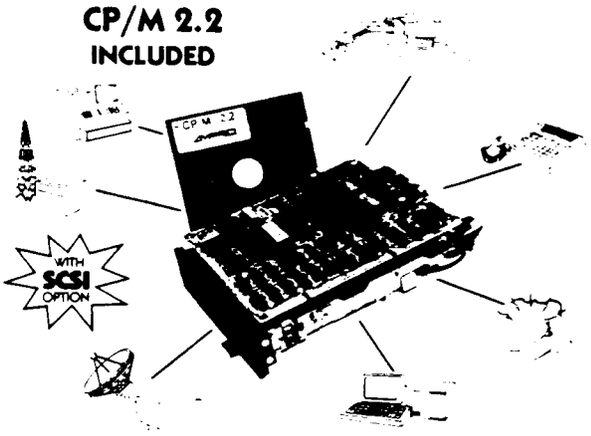
—Fred Blechman, K6UGT

U.S.A.	$15.00 for 1 year
Mexico, Canada	$25.00
Foreign	$35.00(land) - $55.00(air)
(U.S. funds only)	
Permanent (U.S. Subscription)	$100.00
Sample Copy	$3.50

CHET LAMBERT, W4WDR
1704 Sam Drive • Birmingham, AL 35235
(205) 854-0271

Little Board™....$249
The World's Least Expensive CP/M Engine

CP/M 2.2 INCLUDED



- 4 MHz Z80A CPU, 64K RAM, Z80A CTC, 4-32K EPROM
- Mini/Micro Floppy Controller (1-4 Drives, Single/Double Density, 1-2 sided 40/80 track)
- 2 RS232C Serial Ports (75-9600 baud & 75-38, 400 baud), 1 Centronics Printer Port
- Power Requirement: +5VDC at .75A; +12VDC at .05A / On board -12V converter
- Only 5.75 x 7.75 inches, mounts directly to a 5-1/4" disk drive
- Comprehensive Software Included:
  - Enhanced CP/M 2.2 operating system with ZCPR3
  - Read/write/format dozens of floppy formats (IBM PC-DOS, KAYPRO, OSBORNE, MORROW...)
  - Menu-based system customization
  - Operator-friendly MENU shell
- OPTIONS:
  - Source Code
  - TurboDOS
  - ZRDOS
  - Hard disk expansion to 60 megabytes
  - SCSI/PLUS™ multi-master I/O expansion bus
  - Local Area Network
  - STD Bus Adapter

BOOKSHELF™ *Series 100*
Fast, Compact, High Quality, Easy-to-use CP/M System

Priced from $895.00
10MB System Only $1645.

- Ready-to-use professional CP/M computer system
- Works with any RS232C ASCII terminal (not included)
- Network available
- Compact 7.3 x 6.5 x 10.5 inches, 12.5 pounds, all-metal construction
- Powerful and Versatile:
  - Based on Little Board single-board computer
  - One or two 400 or 800 KB floppy drives
  - 10-MB internal hard disk drive option
- Comprehensive Software Include
  - Enhanced CP/M operating sys with ZCPR3
  - Word processing, spreadsheet relational database, spelling checker, and data encrypt/ decrypt (T/MAKER III™)
  - Operator-friendly shells, Menu, Friendly
  - Read/write and format dozens of floppy formats (IBM PC-DOS, KAYPRO, OSBORNE, MORROW...)
  - Menu-based system customization

DISTRIBUTORS

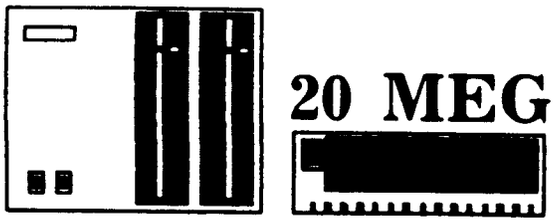
ARGENTINA: FACTORIAL, S.A., (1) 41-0018, TLX 22408 BELGIUM: CENTRE ELECTRONIQUE LEMPEREUR, (041) 23-4541, TLX 49621 CANADA: DYNACOMP COMPUTER SYSTEMS LTD., (604) 879-7737 ENGLAND: QUANT SYSTEMS, (01) 253-8423, TLX 946240 REF 19003131 FRANCE: EGAL+, (1) 509-1800, TLX 620893 SPAIN: XENIOS INFORMATICA, 593-0822, TLX 50364 AUSTRALIA: ASP MICROCOMPUTERS, 613) 500-0928 BRAZIL: CNC DATA LEADER LTDA., (41) 262-2262, TLX 041 [illegible] DENMARK: DANBIT, (03) 66-80-80, TLX 43558 FINLAND: SYMMETRIC OY, (0) 585-322, TLX 121394 ISRAEL: ALPHA TERMINALS, LTD., (3) 49-16-95, TLX 341667 SWEDEN: AB AKTA, (08) 54-80-90, TLX 13709 USA: CONTACT AMPRO COMPUTERS INC., TEL. (415) 962-0230 TELEX 4940309

AMPRO COMPUTERS INCORPORATED

IBM® IBM Corp. CP/M® Digital Research Inc. TurboDOS® Echelon Inc. SCSI/PLUS® Software 2000, Inc. T/MAKER® [illegible] Co.

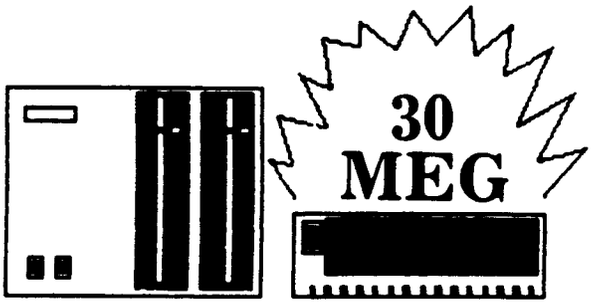
67 East Evelyn Ave. • Mountain View, CA 94041 • (415) 962-0230 • TELEX 4940309

PERIPHERAL LAND
HARD DISK FOR
THE
AMPRO COMPUTER



20 MEGABYTE SUBSYSTEM.....$799.00

FEATURES:
-SCSI CONTROLLER
-20 MEGABYTE HALF HEIGHT HARD DISK
-ENCLOSURE WITH FAN &POWER SUPPLY
-100% COMPLETE & TESTED SUBSYTEM



30 MEGABYTE SUBSYSTEM.....$995.00

FEATURES:
-SCSI CONTROLLER
-30 MEGABYTE HALF HEIGH HARD DISK
-ENCLOSURE WITH FAN &POWER SUPPLY
-100% COMPLETE & TESTED SUBSYSTEM

Peripheral Land
3677 Enochs St.
Santa Clara, Ca. 95051
Tel. (408)-733-7600

The SCSI Interface

Building a SCSI Adapter

by Rick Lehrbaum, Vice President Engineering, AMPRO

SCSI25

Introduction

As we discussed in previous issues of The Computer Journal (issues #22-24), the key advantage of the Small Computer System Interface (SCSI) is that it provides a host-independant interface to a wide variety of peripheral devices. This host independance exists on both a hardware and a software level.

To take advantage of the benefits of SCSI, your computer needs two things:

1. A hardware Adapter to the SCSI bus.
2. A software driver allowing you to control the SCSI devices.

In order to best illustrate the principles of both of these, a very simple construction project will be undertaken. The hardware portion will be covered in this issue, and the software next time.

Regarding the hardware construction, we've assumed that you can wire up the adapter from the schematic and other drawings given — no actual step-by-step construction procedures are included. If you need help, find a friend who knows how to construct a project of this nature.

As reference materials in this project, you may require the following:

- ANSC X3T9.2 SCSI Specification
- NCR 5380 Design Manual

These are both available from AMPRO Computers. In addition, the SCSI Adapter is available assembled, and as a partial kit. (See below.)

A SCSI Adapter

The SCSI Adapter is a small daughter board which plugs directly into the Z80 CPU socket of a Z80-based host, adding a SCSI bus interface to the host computer. As discussed in past articles, the added "SCSI" bus can be used in a diverse range of applications including SASI/SCSI hard-disk controller interface, I/O port expansion, communications, industrial control, multi-processing, networking, etc. (AMPRO even offers an STD bus Adapter that can be run from the SCSI bus of the SCSI Adapter.)

The SCSI Adapter's bus interface meets the specifications for the popular Small Computer System Interface (SCSI) — formerly called "SASI." SCSI's single- and multi-master modes and disconnect/reconnect functions are fully supported through the use of the NCR 5380 SCSI interface controller device.

Alternately, the SCSI Adapter's seventeen bidirectional I/O lines and 8-bit ID input port may be used as general purpose digital input/output lines (without SCSI compatibility) in a manner determined by the system designer.

Although the SCSI Adapter was designed specifically for use with the original AMPRO Little Board computer (which did not provide a SCSI bus), it can be used in most Z80 and Z80-based applications. Among the known users are owners of TRS-80 and Morrow CP/M systems.

To allow compatibility with the greatest number of Z80 systems, the bus load and power requirements of the SCSI Adapter have been kept minimal. However, whether the SCSI Adapter can be successfully used in a particular application depends on the specific Z80 or Z80-based host circuit design. Be sure to perform both theoretical analysis (of bus loading and system timing) and actual in-system testing before using the SCSI Adapter in a particular system.

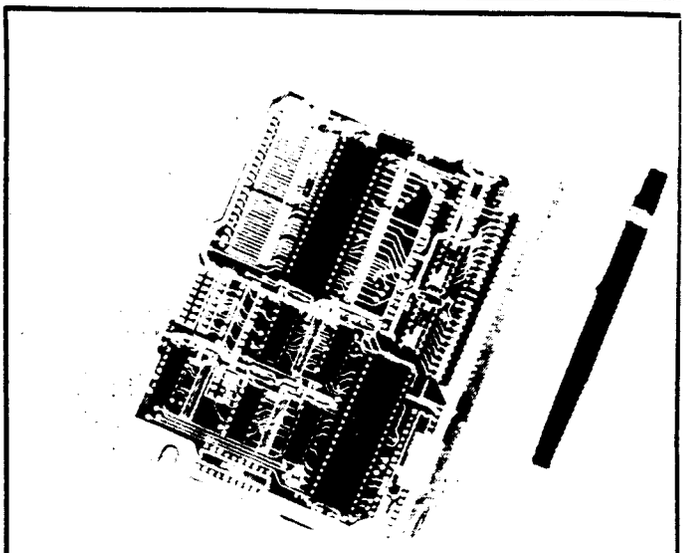


Figure 1: Picture of Board.

How it Works

The SCSI Adapter is extremely simple electronically. Figures 1-5). It consists mainly of a Z80(A) (transferred to SCSI Adapter from the host CPU board), an optional Z80 DMA controller, a SCSI bus controller IC (NCR 5380), an jumperable bus ID input port, and some random logic.

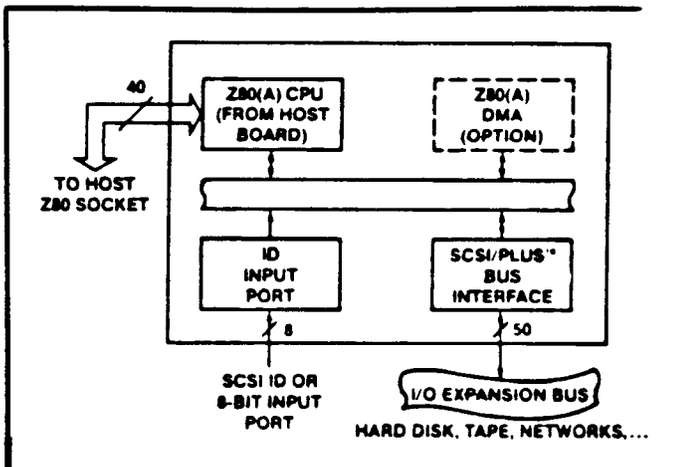
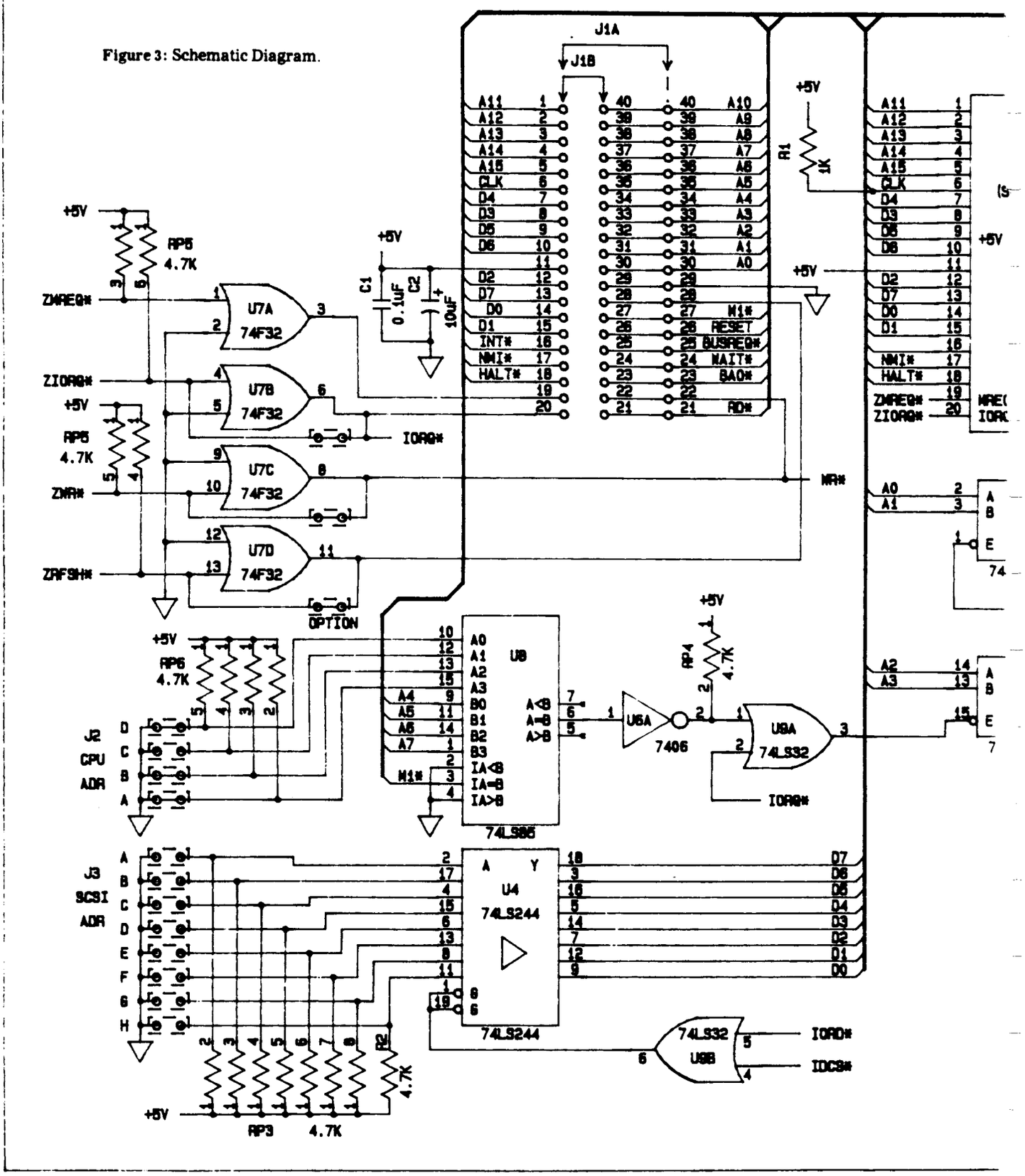
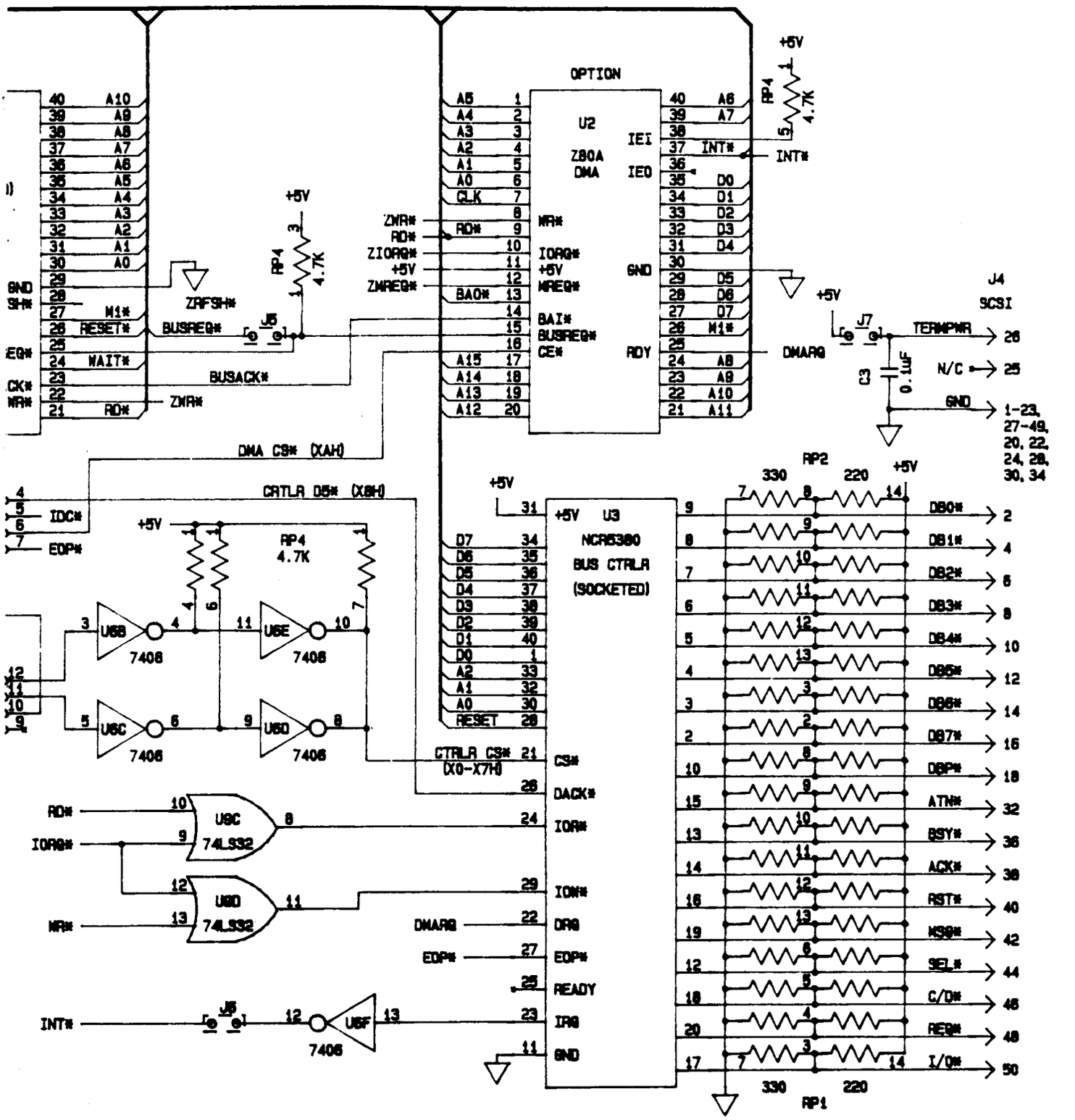


Figure 2: Block Diagram.

Figure 3: Schematic Diagram.




OPTION
U2
Z80A
DMA
IEI
IEO
INT#
MR#
IORQ#
GND
MREQ#
BAI#
BUSREQ#
CE#
RDY
+5V
RP4
4.7K
ZWR#
RD#
ZIORQ#
ZMREQ#
BAO#
A15
A14
A13
A12
A5
A4
A3
A2
A1
A0
CLK
D7
D6
D5
D4
D3
D2
D1
D0
M1#
RESET#
WAIT#
ZRFSH#
BUSACK#
DMA CS# (XAH)
CRTLR DS# (X8H)
IDC#
EOP#
DMARQ
J4
SCSI
J5
J7
TERMPWR
N/C
C3
0.1uF
GND
1-23, 27-49, 20, 22, 24, 28, 30, 34
RP2
330
220
RP1
U3
NCR5380
BUS CTRLR
(SOCKETED)
RESET
CS#
DACK#
IOR#
IOW#
DRQ
EOP#
READY
IRQ
DB0#
DB1#
DB2#
DB3#
DB4#
DB5#
DB6#
DB7#
DBP#
ATN#
BSY#
ACK#
RST#
MSG#
SEL#
C/D#
REQ#
I/O#
U6B
U6C
U6D
U6E
U6F
7406
74LS32
CTRLR CS# (X0-X7H)

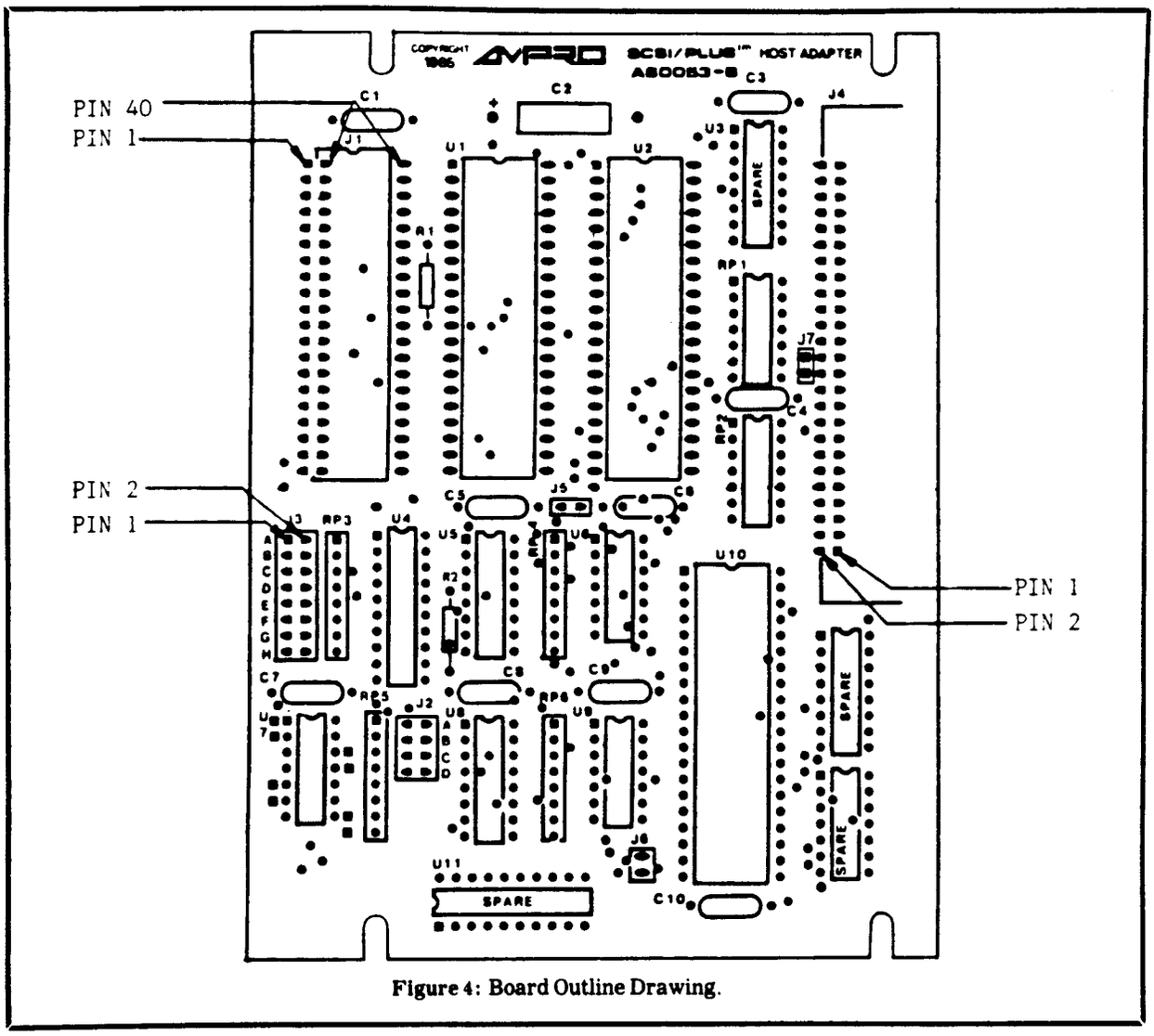


Figure 4: Board Outline Drawing.

Item	Qty	Description
C1, 3-10	9	CAP CER AXIAL 0.1UF +80% −20% 50V
C2	1	CAP ELC AXIAL 10UF −10%+50% 25V
J1	2	CONN HDR 20×1
J2	1	CONN HDR 4×2 .1×.1
J5,6,7	2	CONN HDR 2×1 .1×.1
J3	1	CONN HDR 8×2 .1×.1
J4	1	CONN HDR 25×2 .1×.1 RT-ANG
R1	1	RES CF 1K 5% ¼W
R2	1	RES CF 4.7K 5% ¼W
RP1,2	2	RES N/W DIP 12-220/330
RP3,4,5,6	4	RES N/W SIP 7-4.7K
U1,2,10	3	Socket 40-PIN
U4	1	IC 74HCT244
U5	1	IC 74HCT139
U6	1	IC 7406
U7	1	IC 74F32
U8	1	IC 74HCT85
U9	1	IC 74HCT32
U10	1	IC NCR 5380
RP1,2	2	Socket, IC 14 Pin

Figure 5: Parts list.

At position J1, two rows of PCB interconnect pins (on 0.3" centers) allow direct plug-in into the host CPU's Z80(A) socket. Alternatively, the pinout of J1 allows substitution of either a 40-pin 0.1" × 0.1" header connector, or a 40-pin socket-type (0.3" × 0.1") DIP plug. Pins of the host CPU Z80(A) socket (U4) are connected directly to their counterparts at the Z80(A) socket (U1) on the SCSI Adapter.

CPU Interface

A 74LS85 quad comparator senses when the Z80(A)'s address bits A4-A7 match the configuration of four user-definable jumpers. The SCSI Adapter occupies a block of sixteen I/O addresses, according to the setup of these jumpers. A 74LS139 decoder generates five device select strobes: three for the 5380, one for the ID input port, and one for the optional DMA controller.

Because no access to the host CPU's interrupt daisy chain is available via the CPU socket, the Z80(A)'s interrupt mode 1 must be used if interrupts from the SCSI Adapter are to be used. If not, the host CPU can be used in any desired interrupt mode. A jumper option (J6) is provided which allows enabling or disabling interrupts from the 5380.

The SCSI Interface

The main item of interest on the SCSI Adapter is the SCSI bus interface. All of the signals of the I/O expansion bus interface connect directly to pins of U10, the 5380 bus controller IC. Nothing could be simpler!

A 74LS244 is used as an input port, allowing the state of eight user-configurable jumpers to be read under software control. Alternatively, this port may be used for general purpose input sensing.

The 5380 integrates approximately 20 components in a single 40-pin package. It allows full programmable control of 17 bi-directional bus signals, and provides both buffered (low leakage) bus inputs and high current (48 mA) bus output drive capacity. The 5380 is described thoroughly in its data sheet, which is available from NCR (and AMPRO).

The 5380 provides automatic REQ/ACK handshaking when it is used in its DMA mode. As connected in the SCSI Adapter, the 5380's DMA acknowledge (DACK) input is used as a second chip select. This allows both DMA and programmed I/O usage of the 5380's automatic REQ/ACK handshaking. When used during programmed I/O data transfers, the automatic handshaking reduces transfer overhead considerably. This is called "pseudo-DMA."

A variety of interrupt features are included in the 5380, such as interrupt on phase change, interrupt on selection, interrupt on bus reset, interrupt on loss of the SCSI bus BUSY signal, and interrupt on SCSI bus parity errors. The 5380's interrupt output can be enabled and disabled with a jumper wire (J6). In addition, the 5380's command registers allow each interrupt condition to be masked or conditioned, under software control.

DMA Option

If pseudo-DMA isn't enough to suit your fancy, circuitry is present on the SCSI Adapter which permits addition of a Z80(A) DMA controller device. In most cases, adding a DMA controller requires you to cut four traces and add a SIP resistor network. (This is described below). The use of this option is often not possible in host CPU's which use the Z80(A) to control RAM refresh or read/write timing (such as the AMPRO Little Board), due to timing restrictions. Some experimentation may be required.

Note that the DMA controller is only useable for transfers through the SCSI Adapter; it can not assist with the I/O transfers of devices on the host CPU, due to the lack of DMA request signals from those devices. Also, if the DMA controller is used, it may require use in the "byte-at-a-time" DMA mode, in applications which use the Z80(A) CPU to either control or assist with RAM refresh.

If the Z80(A) DMA controller is present, its ready (RDY) input is driven by the DMA request (DRQ) output of the NCR 5380. On transfers, it uses the same read and write control signals as the Z80(A) CPU.

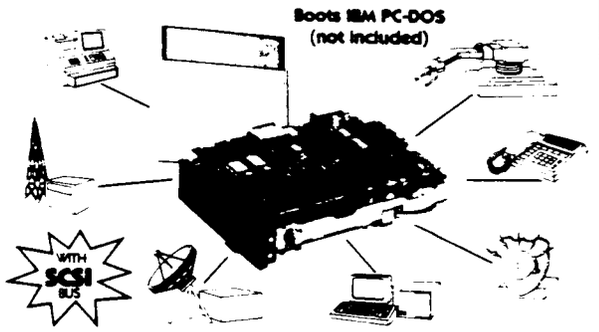
Build It! — Then Read On.

Once you have completed the SCSI Adapter, you need to set it up for use and plug it into your host computer. In the following discussion, references will be made to the board's connectors and jumpers as implemented on the adapter board (see Figure 4).

SCSI ENGINES

Little Board/186™ $495

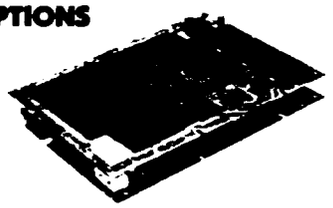
High Performance, Low Cost PC-DOS Engine

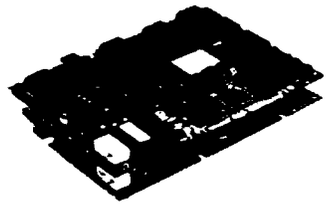


- Three times the COMPUTING POWER of a PC
- Data and File Compatible with IBM PC, runs "MS-DOS generic" programs
- 8 MHz 80186 CPU, DMA, Counter/Timers, 128/512K RAM zero wait states, 16-128K EPROM
- Mini/Micro Floppy Controller (1-4 Drives, Single/Double Density, 1-2 sided, 40/80 track)
- 2 RS232C Serial Ports (50-38,400 baud), 1 Centronics Printer Port
- Only 5.75 x 7.75 inches, mounts directly to a 5-1/4" disk drive
- Power Requirement: +5VDC at 1.25A; +12VDC at .05A; On board -12V converter
- SCSI/PLUS™ multi-master I/O expansion bus
- Software included:
  - PC-DOS compatible ROM-BIOS boots DOS 2.x and 3.x
  - Hard Disk support

OPTIONS

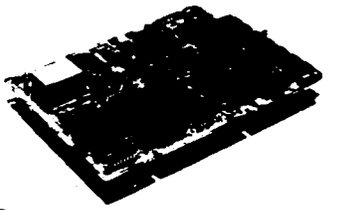
PROJECT BOARD/186™ - adds 25 square inches of wire wrap prototype area with buffered and pre-decoded 80186 bus interface for Little Board/186





EXPANSION/186™ - adds five key options to Little Board/186
- 512K RAM
- 8087 co-processor
- Battery-backed Real Time Clock
- 2 RS232/422 sync/async serial ports
- I/O expansion bus

VIDEO RAM EMULATOR™ - allows use of software that writes to display controller "VIDEO RAM"



SCSI/108™ - permits connection of off-the-shelf STD bus industrial I/O interfaces (analog, digital, serial, display, power control, etc.)

DISTRIBUTORS

ARGENTINA: FACTORIAL, S.A., 41-0018 TLX 22408 AUSTRALIA: ASP MICROCOMPUTERS, (613) 500-0698, TLX 36587 BELGIUM: CENTRE ELECTRONIQUE LEMPEREUR, (041) 23-45-41, TLX 49681 BRAZIL: COMPULEADER COMPUTADORES LTDA., (41) 868-1939, TLX 416132 CANADA: TRI-M, (604) 438-9012 DENMARK: DANBIT, (03) 66 80 80, TLX 43558 UK: AMBAR SYSTEMS LTD., 0296 35511, TLX 837487 FINLAND: SYMMETRIC OY, 358-0-585-322, TLX 121394 FRANCE: EGAL PLUS, (1) 4509-1800, TLX 680893 ISRAEL: ALPHA TERMINALS, LTD., (03) 49-16-95, TLX 341667 SWEDEN: AB AKTA, (08) 54-90-90, TLX 13709 USA: CONTACT AMPRO COMPUTERS INC.

IBM®, IBM Corp., 80186®, Intel, Corp.

AMPRO COMPUTERS, INCORPORATED

67 East Evelyn Ave, • Mountain View, CA 94041 • (415) 962-0230
TELEX 4940308 • FAX (415) 962-1042

Board Connectors

J1 — CPU Board Interconnect: A pair of 20-pin single-in-line terminal strips, with 0.1" pin spacing and 0.3" space between rows, provides the means by which the SCSI Adapter plugs directly into a Z80 CPU socket. The pins extend below the SCSI Adapter. When plugged into a Z80 CPU socket, the resulting space between boards is 0.5".

The AMPRO board's printed circuit layout allows two additional options for connection to the host CPU board.

(1) DIP socket option — a 40-pin DIP socket can be substituted for the inter-board terminal strips, allowing the use of a ribbon cable interconnect.

(2) Male header option — a 40-pin dual row (0.1" × 0.1") male header can be substituted for the inter-board terminal strips, again allowing the use of a ribbon cable interconnect.

If either option (1) or (2) is used, care must be taken to minimize cable length.

Pinout of J1 is identical to that of the Z80 (or Z80A) CPU.

J3 — ID Input Port Connector: A 16-pin dual-row male header with 0.1 × 0.1 inch pin spacing allows either ribbon cable connection to external circuitry, or use of standard shorting plugs (or wire-wrap wire) to program a specific bit pattern.

Each of the jumper pairs, labeled A-H on the board, may be read under software control. For each pair not shorted, the corresponding data bit will be read as a 1; for each pair shorted, the bit will read 0. The pairs are mapped to the CPU's data bus as shown in Figure 6.

Pin	Label	Function
1-15 (odd)		Signal ground
2	A	External input "MSB"
4	B	:
6	C	:
8	D	:
10	E	:
12	F	:
14	G	:
16	H	External input "LSB"

Figure 6. J3 Connector Pinout

Pin	Signal Name	Function
1-49 (odd)*	Ground	Signal ground
2	-DB0	Data bit 0 ("LSB")
4	-DB1	Data bit 1
6	-DB2	Data bit 2
8	-DB3	Data bit 3
10	-DB4	Data bit 4
12	-DB5	Data bit 5
14	-DB6	Data bit 6
16	-DB7	Data bit 7 ("MSB")
18	-DBP	Data parity
20,22,24	Ground	Signal ground
25	No connection	
26	TERMPWR	Termination +5V
28,30	Ground	Signal ground
32	-ATN	Attention
34	Ground	Signal ground
36	-BSY	Busy
38	-ACK	Transfer acknowledge
40	-RST	Reset
42	-MSG	Message
44	-SEL	Select
46	-C/D	Control/Data
48	-REQ	Transfer request
50	-I/O	Input/Output

* Except pin 25.

Figure 7. J4 Connector Pinout

J4 — SCSI Bus Connector: This 50-pin right-angle dual-row header is compatible with the ANSI standard X3T9.2, for single-ended drivers and receivers. J4's pinout is shown in Figure (7).

Jumper Options

The SCSI Adapter does not require that any jumpers be set for normal use with an AMPRO Little Board single board computer. However, additional flexibility of application has been made possible by means of several jumper options. In most cases, traces on the bottom (circuit side) of the PC board set the default jumpering; these will need to be cut if custom jumpering is desired.

J2 — Board Base Address (from the Z80): The 4-bit base address of the SCSI Adapter with respect to the Z80 CPU's I/O address space can be programmed by means of the four pairs of pads at position J2. A shorted pair requires the associated address bit to be a 0. The four pairs of pads are labeled A, B, C, and D on the PC board, and are assigned as follows:

Pair A - Address bit A7	Default: shorted
Pair B - Address bit A6	Default: shorted
Pair C - Address bit A5	Default: open
Pair D - Address bit A4	Default: shorted

Default: Pairs A, B, and D are normally shorted by traces on the bottom (circuit side) of the PC board, resulting in a default base address of 20h. These must be cut if an open pair is required.

Example: To change the board's base address to F0h requires cutting the three traces between pairs A, B, and D. The board then occupies addresses F0-FFh instead of 20-2Fh.

J5 — SCSI ID As mentioned above, J5 can serve as either a spare 8-bit input port, or to provide the board's SCSI Initiator ID to software. Whether a board SCSI ID is required or not depends on the software you will use.

Default: the host's SCSI ID is usually set to 7, which has the highest bus priority. Leaving all ID jumpers off results in an ID of 7.

J5 — DMA Daisy Chain: If a DMA controller is added (see below) short this pair of pads only if the Z80-based host system has additional DMA devices on it.

Default: Unshorted.

J6 — Bus Controller Interrupt: Short this option if interrupts from the bus interface controller (NCR 5380) are desired.

Default: Unshorted (no interrupts).

Bus Termination

A pair of resistive termination networks are included on the SCSI Adapter, in sockets labeled RP1 and RP2. Each of these two devices contains twelve 220/330 ohm resistor pairs, providing the recommended SCSI bus termination.

On a SCSI bus, termination should occur at the two ends of the bus. The networks (RP1 and RP2) are socketed so that they can be removed if the bus is terminated (in two places) elsewhere.

Some applications may allow substitution of networks with alternate resistor values in order to conserve power. For example, 14-pin, 13 resistor, 4.7K pullup networks used in place of the 220/330 networks will save approximately 1W of power. Note that this requires care in the timing and use of the bus signals, due to the increase of bus impedance from 132 ohms to 4.7K ohms.

J4 provides the option to provide a source of +5 volts DC to pin 26 of the I/O bus interface (J4), for use by an external bus ter-

mination network. To enable this feature, a jumper or soldered wire must be installed at J4.

NOTE

A current blocking diode is not included between the board's +5V and the SCSI bus DC line, so be sure no other device drives this bus line if this jumper is shorted.

Default: unshorted.

Adding a DMA Controller

A Z80A DMA controller can be added at position U2. Several modifications to the board may be required when this is done, depending on the specific design of the host Z80(A) CPU logic. Whether DMA can be used at all with a particular host CPU board requires careful analysis of logic design and busloading constraints.

NOTE

This option should not be used with an AMPRO Little Board single board computer, due to the use of the Z80A to control dynamic RAM refresh and read/write timing.

When DMA is used, the Z80 control signals MREQ, IORQ, WR, and RFSH must be terminated due to the periodic tri-stating of those signals which occurs during transfer of control between the CPU and DMA controller, to prevent uncontrolled memory or I/O accesses. If termination of these signals is desired, a 4.7K SIP (8 pin, 7 resistor) resistor should be added to the board, at position RP5.

In addition, if the above optional step is performed to add signal termination, the Z80(A) CPU may require additional buffering of the four terminated signals. This can be accomplished by adding a 74S32 or 74F32 IC at position U7. Before adding this device, four traces near U7 must be cut. Each of the traces you must cut is located on the bottom side of the PC board, and runs between a pair of square pads; they are located adjacent to U7 pins 1 & 2, 5 & 6, 8 & 9, and 12 & 13.

Testing the Board

Installation of the SCSI Adapter is normally a simple matter, consisting of the following steps:

1. Carefully unplug the Z80(A) from the host CPU board. Be careful not to bend any of the IC's pins.

The Z80(A) is static sensitive, and should be handled with care. Handle the device by its plastic body; do not touch the device's pins.

2. Orient the SCSI Adapter so that the board-to-board interconnect pins are directly above the host CPU board's Z80(A) socket. Carefully insert the SCSI Adapter's interconnect pins (J1) into the Z80(A) socket on the host CPU board, being certain that each pin mates properly with the corresponding socket contact.

BE SURE PIN 1 OF THE ADAPTER'S Z80 PLUG MATES WITH PIN 1 OF THE HOST'S Z80 SOCKET.

3. Use brackets or spacers to mount the Adapter securely.

When installing on an AMPRO Little Board, use four 0.5" spacers between the SCSI Adapter and the Little Board at the four board mounting slots, and mount the pair of boards securely to their chassis.

4. After installing the Adapter, you should be able to powerup your computer and boot your operating system as before. Assuming that there is no I/O port address conflict, your computer should behave normally.

Coming Soon...

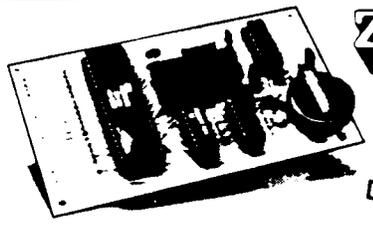
In the next issue, we'll cover the software side of things, including how to address the board's I/O ports, how to perform simple bidirectional I/O with the 5380 SCSI controller, and how a SCSI driver works. ■

ITEMS AVAILABLE FROM AMPRO COMPUTERS, INC.

The following items relating to the SCSI Adapter project are available from:

AMPRO Computers, Inc.
PO Box 390427
Mountain View, CA 94039
Telephone: (415) 962-0230

1. SCSI Adapter partial kit. Includes bare printed circuit board, 5380 SCSI controller, bus termination networks (2), 40-pin board-to-board Z80 socket interface connector, and SCSI Adapter technical manual. $69
2. Fully assembled and tested SCSI Adapter. Includes technical manual. $99
3. SCSI Adapter technical manual. $10
4. NCR 5380 Design Manual (58 pages). $10
5. ANSC SCSI Specification. The SCSI "bible". $20
6. AMPRO Z80 Hard Disk Software.



Ztime-I
CALENDAR/CLOCK
$69 KIT
NOW WITH FILE DATE STAMPING!

- Works with any Z-80 based computer.
- Currently being used in Ampro, Kaypro 2, 4 & 10, Morrow, Northstar, Osborne, Xerox, Zorba and many other computers
- Piggybacks in Z80 socket.
- Uses National MM58167 clock chip, as featured in May '82 Byte.
- Battery backup keeps time with CPU power off!
- Optional software is available for file date stamping, screen time displays, etc.
- Specify computer type when ordering.
- Packages available:

Fully assembled and tested	$99.
Complete kit	$69.
Bare board and software	$29.
UPS ground shipping	$ 3.

MASTERCARD, VISA, PERSONAL CHECKS, MONEY ORDERS & C.O.D.'s ACCEPTED.
N.Y. STATE RESIDENTS ADD 8% SALES TAX

KCT
KENMORE COMPUTER TECHNOLOGIES
P.O. Box 635, Kenmore, New York 14217 (716) 877-0617

NEW-DOS

Part 4: The CCP Internal Commands

by C. Thomas Hilton

The next personality procedure we will deal with is the erase procedure. You may have noted that you must spell out the entire word, rather than using the CP/M convention of "ERA." The full word convention was selected to make the commands easier to use in a voice computing environment.

The erase procedure does not erase the document. It only signifies the index entry is inactive. The document may be recovered, using a number of utilities, provided you have not written any new data to the disk, nor have performed any procedure which would require the allocation of new disk space. Any procedure which requires space will use the space allocations previously controlled by the inactive document. We begin our discussion with Listing 1.

We begin this procedure by checking for the expected parameter, or object of the command verb instructions. The first thing we wish to know is how many wild cards the parsing routine placed into the DCB. Remember that question marks are the only wild card characters understood by the FDOS segment, though the CCP will accept astrix characters as field universal wild cards. If both the eight character document name field, and the document type field contain wild cards the number of wild card characters counted by the parsing routine will be eleven. Any other number will indicate that we have more processing to do. A count of exactly eleven will indicate that all documents in the index are to be declared inactive.

Declaring all documents to be inactive, thus eliminating all documents, is a drastic measure. We therefore force the operator to verify this intention. If we do not get an indication that this is indeed the action requested we immediately abort the entire command procedure. We do not take chances with destructive command procedures. Note that the character we receive from the key press response has been capitalized and that we do not except any other character but an affirmative "Y."

We begin the actual erasing procedure by informing the operator that we are erasing a document to be described. This is an imperfect terminal operation as we may have an error code, which ruins the appearance of the added extra. This however is a rare occurrence, so added code was not included to hold the erasing message until a valid action had taken place. Additionally I did not want redundant "erasing" messages. Having informed the operator of our intentions we then select any specified drive. Recall that the select specified drive procedure will abort if no selection is required. Next we clear the private file index mask, and use a portion of the index procedure to write the document about to be erased upon the terminal. Note that if the index procedure is removed you will lose the verification of document about to be erased feature. Having done all we wanted to do we load the DE register pair with the address of the DCB containing the document specification and call the FDOS vector to do the actual erasing for us, and return to caller.

Recall that when we enter the Personality Module the return address of the RESTART CCP vector is on the stack. When we return there the default drive and file partition number will be reassigned, in the event that this procedure has selected another drive.

The DELETE support subroutine has little code and is designed only to assure that the call to the FDOS segment returns system control to a position within the procedure code segment.

The print procedure is a close relative of the read procedure, so it is "stacked" upon the READ procedure, though it is given the status of an individual personality procedure. The same functioning may be had by engaging the screen echo terminal control, ^W, and instructing the CCP to "READ" a document. There are no paging routines available in the basic model for the PRINT procedure. Refer now to Listing 2.

The only effect of the PRINT procedure is to set a flag for use by slave output routines to determine where the output character is to be sent. This concept will be discussed below.

For the READ procedure we enter here, and clear the printer flag. If the printer flag is other than a zero value then the document to be read will be sent to the Line Printer instead of the terminal. It should be noted that other devices could be indicated by setting and testing this variable at the output routines, routing the output character to the described device.

If we entered at the PRINTER procedure we would have set the accumulator with a nonzero value, and entered here, where the printer flag is actually set.

Next, we have the parsing routine transfer the document name into the DCB, and if any wild cards were detected we will jump out to an error code. We may read only one document at a time, only one document may be open at a time, hence any ambiguous reference would violate basic concepts.

With the document name in the DCB we then check to see if there is a single character token left in the command line which may be an instruction to disable the paging routines. If there is a token then it will have been returned in the accumulator. If not then we will have returned a null character, indicating the end of the line. In either event we place the character into the paging variable. If it was an end of line indicator then we jump out.

If the end of the line was not reached then we increment the pointer and place it into the command pointer variable after swapping registers.

With the basic housekeeping taken care of we are ready to begin reading the specified document. For this discussion refer to Listing 3. We first call the select the specified disk routine, to activate the operator specified drive, where the document is supposed to be. If the operator has not specified a drive then we will use the currently active drive to look for the specified document. The next step is to "open" the document, reading in the index and allocation data. In the event that the document does not exist, or cannot be found, we will return with the "Z" flag set. This will indicate an error condition and we will jump out of the current code segment.

Assuming we did not have an error we clear a new line upon the terminal. The number of vertical lines we may display upon

;LISTING ONE

```
;   PROCEDURE IPP.02
;   ----------------
;   Purpose: Free disk space for other documents by indicating that the
;        index entry is vacant. Documents may be recovered by using
;        RECOVER.COM, if no index changes have been made since the
;        'erasure.'
;   Verb Syntax:
;        erase < ambiguous document name >
;        erase document.typ
;        erase *.*    (clears all index entries)
;
ERASE:  CALL    PARSE       ;parse document specification, if any
        CP      11          ;upon return accumulator contains the number
                            ;of wild card tokens, ('?'). If the value in
;the accumulator is eleven then both document name and type are wild
;
        JR      NZ,ERASE1   ;if the operator has not specified that all
                            ;documents are to be erased then jump out
        CALL    SYSPTC      ;else verify operator's intentions
        DEFB    'Erase All Documents?',0
        CALL    INPTRM      ;by a single key press response
        CP      'Y'         ;did the operator verify intent?
        JP      NZ,RESTRT   ;no, so abort erase procedure
ERASE1: CALL    SYSPTC      ;tell operator what we are doing, and to whom
        DEFB    'Erasing ',0
        CALL    SSELDSK     ;select the specified drive, if any
        XOR     A           ;clear accumulator for private document mask
        LD      B,A         ;there are no private document exclusions here
                            ;so save the mask
        CALL    WRTNDX      ;and write a list of erased documents
        LD      DE,DCBDN    ;then load DCB address in preparation of making
        CALL    DELETE      ;and tell FDOS to delete the documents from the
                            ;Index
        RET                 ;and return to CCP as a subroutine to reselect
                            ;the operator's default drive and file
;
;The Delete document microcall is first called by ERASE so included here
;
DELETE: LD      C,13H       ;load the delete document specified in DCB code
        JP      FDOS        ;and make the FDOS call
```

;LISTING TWO

```
;   PROCEDURE IPP.03
;   ----------------
;   Purpose: Print the specified document on the Line Printer Device.
;   Verb Syntax:
;        print < unambiguous document name >
;   Example:  print document.txt
;
PRINT:  LD      A,0FFH      ;set the printer flag and use the READ
        JR      READ0       ;proced
```

```
;   PROCEDURE IPP.04
;   ----------------
;   Purpose: Read the specified document on the logical terminal device.
;   Verb Syntax:
;       read < unambiguous document name >
;       read < unambiguous document name > N   (disables pagination)
;   Examples:  read document.txt
;          read document.txt N
;   Local Equates:
;
NLINES  EQU     24          ;number of displayable terminal lines of text
                            ;Hermit DOS assumes an 80 x 24 terminal display
PGDFLG  EQU     'N'         ;this flag causes document to be read without
                            ;pagination
;
READ:   XOR     A           ;turn off printer flag
READ0:  LD      (PRFLG),A   ;set device flag. if zero read document on the
                            ;terminal, if nonzero send it to the printer
        CALL    PARSE       ;parse out the document name and type
        JP      NZ,ERROR    ;if any wild cards found we have an error as
                            ;wild cards not allowed here.
        CALL    ADVAN       ;check to see if operator wants to disable
                            ;pagination
        LD      (PGFLG),A   ;save it as a flag
        JR      Z,NOSLAS    ;if we have reached the end of the command line
                            ;with the ADVANce call then we jump out
        INC     DE          ;else advance the buffer pointer
        EX      DE,HL       ;swap them
        LD      (CMDPTR),HL ;and update the command line pointer
NOSLAS: CALL    SSELDSK     ;then select a disk if any requested
        CALL    OPNDOC      ;and open the specified document
        JP      Z,READ4     ;if the open document routine returns with the
                            ;'Z' flag set then we have an error
```

;LISTING THREE

```
        CALL    CRLF        ;other wise clear a new line
        LD      A,NLINES-1  ;set the line count
        LD      (PAGCNT),A  ;and put it into the count variable
        LD      HL,CHRCNT   ;then we get the address of the character
                            ;position counter variable
        LD      (HL),0FFH   ;and set it to an empty line status then
        LD      B,0         ;set the tab character counter
READ1:  LD      HL,CHRCNT   ;and load pointer to char position/count
        LD      A,(HL)      ;and check to see
        CP      80H         ;if we are at the end of the buffer
        JR      C,READ2     ;if not then jump out
        PUSH    HL          ;else save HL
        CALL    SYRDF       ;and read a record from the document
        POP     HL          ;and get back the pointer
        JR      NZ,READ3    ;did we have a read error?
        XOR     A           ;if not then reset count to zero
        LD      (HL),A      ;and stuff the counter
```

```
READ2:  INC     (HL)            ;Increment it
        LD      HL,PZBUFF       ;and point to the buffer
        CALL    ADDAH           ;then compute address of next character
        LD      A,(HL)          ;get the next char
        AND     7FH             ;mask off the msb
        CP      1AH             ;and check for an end of document marker
        RET     Z               ;exiting the procedure if character is eod
        CP      CR              ;else see if it is the end of a line. if so
        JR      Z,TABRST        ;we have to reset the tab count
        CP      LF              ;as we would if we found a line feed character
        JR      Z,TABRST        ;used by some as an end of line marker
        CP      TAB             ;none of them, so is it a tab character?
        JR      Z,LTAB          ;yup so jump out
        CALL    LCOUT           ;else send out the character to terminal or
                                ;printer, and when you get back
        INC     B               ;increment the character count
        JR      READ2L          ;and jump over some utility routines
TABRST: CALL    LCOUT           ;if here we detected an end of line so send
                                ;a carriage return or line
        LD      B,0             ;and reset the tab counter
        JR      READ2L          ;then join the rest of the gang
```

;LISTING FOUR

```
LTAB:   LD      A,' '           ;if here we detected a tab character so have
        CALL    LCOUT           ;to expand the tabs by printing spaces
        INC     B               ;incrementing the position counter as we go
        LD      A,B             ;but checking to see if the position
        AND     7               ;is a multiple of tab stops
        JR      NZ,LTAB         ;looping until it is else dropping through
READ2L: CALL    TRMSTAT         ;to normal processing, where all roads lead,
                                ;and check to see if operator wants to abort
        JR      Z,READ1         ;if no key press then keep reading
        CP      ESC             ;else check to see if operator wants to escape
        RET     Z               ;aborting procedure is so
        JR      READ1           ;ignoring other characters so as to continue
READ3:  DEC     A               ;decrement the FDOS code byte, error?
        RET     Z               ;if zero adjusted code just exit procedure
READ4:  JP      ERRLOG          ;else select default disk before going to the
SYRDF:  LD      DE,DCBDN        ;we come here to read a record from the
                                ;document and load the DCB address
SYRD:   LD      C,20            ;and the function code and
        JP      FDOSB           ;make the call saving the BC register pair.
```

;LISTING FIVE

```
LCOUT:  PUSH    AF              ;we come here to determine where to send the
                                ;character
PRFLG   EQU     $+1             ;pointer for printer flag variable
        LD      A,0             ;LD's immediate arg is the print flag
        OR      A               ;if the flag is zero we write to the terminal
        JR      Z,LC1           ;and go there else we write on the printer
        POP     AF              ;get the character back
LSTOUT: PUSH    BC              ;save the BC pair
        LD      C,5             ;load the function code for sending character
        JP      OUTPUT          ;and jump to the character out routine
LC1:    POP     AF              ;if we come here want to write on the terminal
                                ;so get the character
        PUSH    AF              ;put it back on the stack, we only want a copy
        CALL    WRTERM          ;and write it on the terminal
        POP     AF              ;then get it back and see
        CP      LF              ;if it is an end of line marker
        JP      Z,PAGER         ;and if so then do we need to separate pages?
        RET                     ;if not end of line then get another character
;    READ PROCEDURE paging routines. the pager counts down lines and
;pauses for operator input if the page count expires. Pagset sets lines/page
;count.
;
PAGER:  PUSH    HL              ;save the pointer
        LD      HL,PAGCNT       ;and load the page count variable address
        DEC     (HL)            ;decrement it
        JR      NZ,PGBAK        ;jump out if not end of page
        LD      (HL),NLINES-2;else refill the page counter
PGFLG   EQU     $+1             ;pointer page flag variable
        LD      A,0             ;get the page flag
        CP      PGDFLG          ;and see if pagination is wanted
        JR      Z,PGBAK         ;if zero then do not page, jump out
        CALL    INPTRM          ;else see if operator is done reading yet
        CP      ESC             ;did operator want to abort?
        JP      Z,RSTCCP        ;yup so abort procedure
PGBAK:  POP     HL              ;no, wants more so restore HL
        RET                     ;and go for more
;
PAGCNT: DEFB    NLINES-2        ;lines left on page
CHRCNT: DEFB    0               ;char count for read
```

the terminal is loaded into the accumulator. We then place the value into a working register in order to keep track of where we are in the read/write process.

The next step is to set another internal variable to a "fresh line" status, and prepare the B register for use in expanding any tab characters we may encounter in the document. We then fall into the main procedure loop.

In this portion of the loop we check to see if we have reached the end of the sector buffer. All disk transfers, you will recall are done a sector at a time. If the character counter, which also may be thought of as the sector position counter, does not indicate that we have reached the end of the sector, then we will jump out.

If we have detected that we need another sector from the document on disk then we will save our pointers, and call a slave routine to bring in another sector. If there is an error we will jump out, else we will clear the character counter and fall into the character handling code segment.

We load the character count, which will serve as an offset value. The character count is a relative position from the start of the buffer, and as such is of little use to us in actually reading the sector. With the offset in the character variable we may use other slave routines to calculate the literal address of the next character, using the DMA buffer's first character as a base address, and adding our character counter offset to the base value. Upon return we have the literal address in the HL register pair, and load the character from the indicated memory location. We then strip off any parity, or WordStar bit, and check to see if it is an end of document marker. If we have reached the end of the document we are finished, and return to the CCP command level through the RSTCCP address which is waiting for us on the stack.

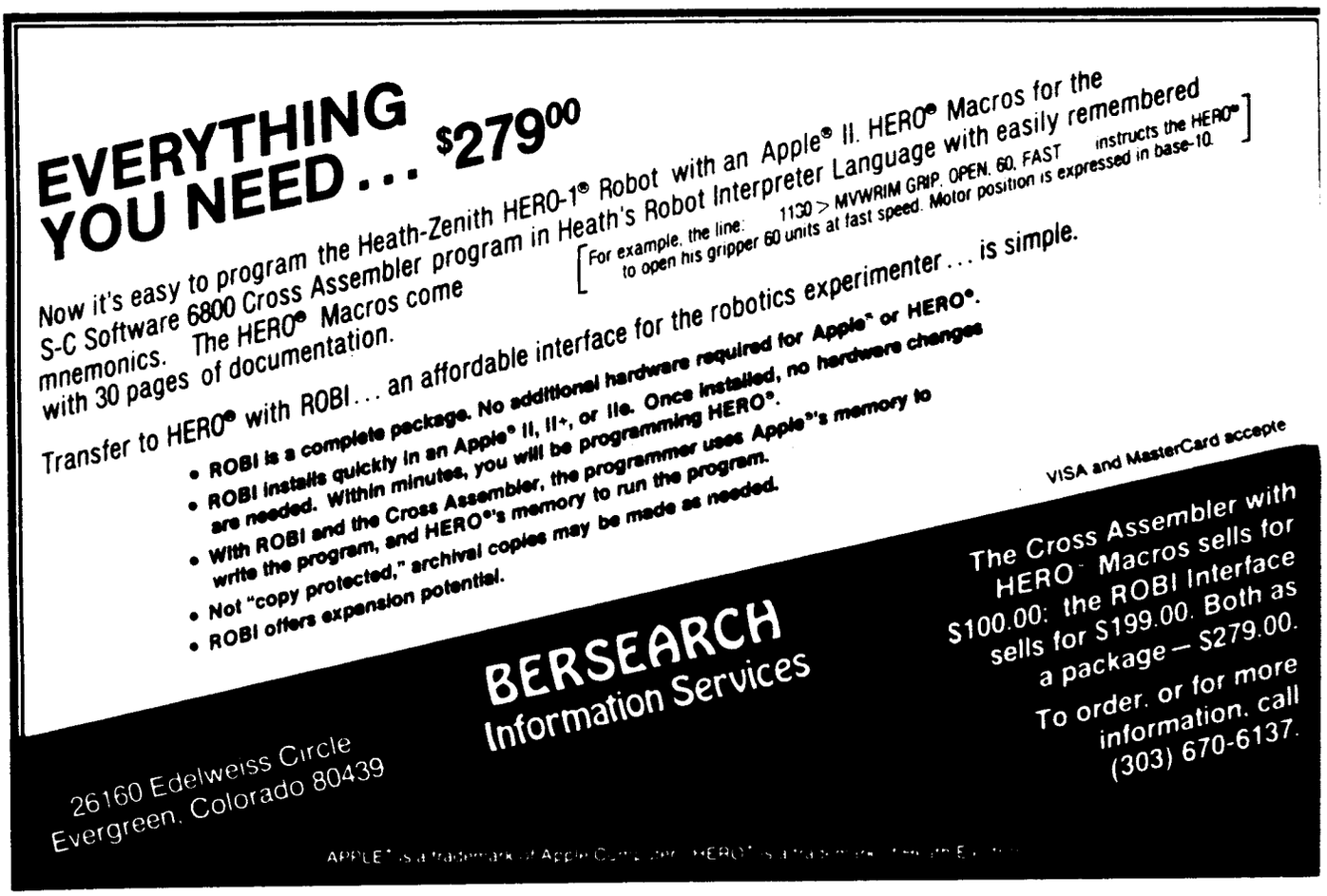
If the character pulled from the sector buffer is not an end of document marker then perhaps it is an end of line marker. If it is either a RETURN character, or a Line Feed character, then we will jump out. If it was neither an end of document, nor end of line indicator then we will check to see if the character is a horizontal tabulation character. Some systems may not be able to handle ASCII control characters, so a tab should be expanded. If it is a tab character then we will jump out, else we will send the character to the terminal, or to the printer, depending upon the state of the printer flag variable. Upon return we will increment the character counter, and jump over some of the special character handling routines.

If we had detected an end of line marker we would have come to the code shown in Listing 4, where we print the character and reset the character counter to indicate a new line and jump out.

If the character was a tab character, we will send space characters to the concerned device until the character counter states that we have reached a multiple of eight characters, which define tab stops in Hermit DOS.

After all write sequences we check for an operator abort command. If no key has been pressed, or the key was not the ESC key, we continue.

If we had any type of error condition detected in the procedure we will come to READ4. If the error condition was an expected error, the end of the document was detected, then decrementing the status code in the accumulator will clear the "Z" flag. If the condition was more serious then we will jump to a fatal error handling routine, notifying the operator of the error.



The slave routine at the end of the listing performs the mundane tasks of setting the registers for a read sequential record FDOS service call, and making the call through a generic FDOS processing vector. As with most disk services the FDOS requires the location of the user's DCB, and assumes the user has defined the desired DMA buffer address prior to making the service call.

The support code in Listing 5 is the beginning of the code which actually sends the document character to the concerned device, either the terminal or the printer. Upon entry we save the character on the stack as we must determine where the character is to be sent. Recall that upon entry to the procedure we defined the state of the immediate argument to the LD instruction. If the byte is a zero value the character will be sent to the terminal, otherwise the character will be sent to the printer. The reader will note that the testing routine is highly simplistic, and may be expanded in scope to include other output devices. If the value of the in line variable was a zero, indicating that the character is to be sent to the terminal we will jump over the code segment at LSTOUT.

Having made the required determination we get the character back off the stack, save the counter in the B register upon the stack, load the service code for sending a character to the printer, and jump to a generic character output routine.

We have determined that we are to write the character upon the terminal. This sequence also requires that we keep track of the display process, and hold upon reaching each full screen of displayed characters, giving the operator time to read the display. Upon entry we get the character back off the stack, but only a copy of it, and write it upon the terminal. We then get the character back, after making the FDOS call, and check for an end of line marker. If it is an end of line marker, then we will have to update the line counter, else we return for more characters.

Note that if you use a text editor which does not use a RETurn/Line Feed sequence to terminate a text line, or does not include a line feed at all, the above code may be modified for proper operation of these routines with your system. All the above code wishes to determine is whether the end of the text line has been reached.

When we come to PAGER we have detected the end of the text line. The first thing we do is save our index pointer. Then we prepare to access the page counter variable. This variable contains the number of lines we are to write upon the terminal. When we first entered the procedure we loaded this variable with the maximum number of lines, hence it is a "count down" process. Having prepared the variable for access we decrement the counter. If the result of this decrement does not zero the counter then we have more lines to write on the terminal screen and simply jump out to a code segment which will restore our character pointer, and return us to the main procedure loop.

Otherwise we will reload the counter for a new screen, and check to see if we are to wait for operator input. The operator may have indicated that paging was not desired. Recall that we placed the token in the immediate argument above. If we reached the end of the line without detecting a nonpaging token the argument would be a zero value, in which case we would jump out of the current code sequence, and return to the main loop.

If we have reached the end of the screen "page" then we will halt the display process and wait for the operator to signify a readiness for a new screen of text from the document. When the character is returned to us we will check for an ESCape character. If we have an abort request then we will return to the CCP command level, resetting the system to the default drive and file partition, else we will return to the the main procedure display loop.

The procedure specific variables, which are not contained in the code itself, are included at the end of the procedure, just to keep them in the concerned area.

It is perfectly reasonable that the user may include paging for the document print routine, with page numbering and other formatting, if there is room in the personality module for the additional code. This is a project that has been on my "one of these days" list, but was not included here as I wanted to present a simple, easy to comprehend CCP for use as a base for your study and enhancement. ■

* These listings are available on disk, or on the TCJ BBS (406) 752-1038.

APROTEK 1000™ EPROM PROGRAMMER

only $250.00

TECHNICAL BREAKTHROUGH NOW ALLOWS A PRICE BREAKTHROUGH

A SIMPLE, INEXPENSIVE SOLUTION TO PROGRAMMING EPROMS

The ***APROTEK 1000*** can program 5 volt, 25XX series through 2564, 27XX series through 27256 and 68XX devices plus any CMOS versions of the above types included with each programmer is a personality module of your choice (others are only $10.00 ea. when purchased with ***APROTEK 1000***). Later, you may require future modules at only $15.00 ea., postage paid. Available personality modules: PM2716, PM2732, PM2732A, PM2764, PM2764A, PM27128, PM27256, PM2532, PM2564, PM68764 (includes 68766). (Please specify modules by these numbers)

APROTEK 1000 comes complete with a menu driven BASIC driver programmer listing which allows READ, WRITE, COPY, and VERIFY with Checksum. Easily adapted for use with IBM, Apple, Kaypro, and other microcomputers with a RS-232 port. Also included is a menu driven CPM assembly language driver listing with Z-80 (DART) and 8080 (8251) I/O port examples. Interface is a simple 3 wire RS-232C with a female DB-25 connector. A handshake character is sent by the programmer after programming each byte. The interface is switch selectable at the following 6 baud rates: 300, 1.2k, 2.4k, 4.8k, 9.6k and 19.2k baud. Data format for programming is "absolute code" (i.e., it will program exactly what it is sent starting at EPROM address 0). Other standard downloading formats are easily converted to absolute (object) code.

The ***APROTEK 1000*** is truly universal. It comes standard at 117 VAC 50-60 HZ and may be internally jumpered for 220-240 VAC 50-60 AZ. FCC verification (CLASS B) has been obtained for the ***APROTEK 1000***.

APROTEK 1000 is covered by a 1 year parts and labor warranty.

FINALLY – A Simple, Inexpensive Solution To Erasing EPROMS

APROTEK-200™ EPROM ERASER
Simply insert one or two EPROMS and switch ON. In about 10 minutes, you switch OFF and are ready to reprogram.
APROTEK-200™ only ***$45.00***.

APROTEK-300™ only ***$60.00***
This eraser is identical to ***APROTEK 200™*** but has a built in timer so that the ultraviolet lamp automatically turns off in 10 minutes, eliminating any risk of overexposure damage to your EPROMS.
APROTEK-300™ only ***$60.00***

APROPOS TECHNOLOGY
1071-A Avenida Acaso, Camarillo, CA 93010
CALL OUR TOLL FREE ORDER LINES TODAY
1-(800) 962-5800 USA or 1-(800) 962-3800 CALIFORNIA
TECHNICAL INFORMATION 1-(805) 482-3604
Add Shipping Per Item $3.00 Cont. U.S. $6.00 CAN. Mexico HI AK UPS Blue

AMPRO 186 Column

Networking With Super Duo

by Peter Ruber

When someone brings up the subject of computers talking to each other, what is the first thought that comes to mind? Plugging in a modem and dialing a bulletin board? Probably. But this is a very static relationship, because all you are really able to do is to either read selected data from the remote computer, download it, or upload a message or file.

The same kind of relationship exists when you connect two like computers through a null modem cable. Once a terminal program has been loaded into each computer, you are limited to transferring files or tapping messages on one another's screen. Walkie talkies are cheaper.

In business environments, where there may be dozens of computers connected through a LAN (Local Area Network) device, the options are somewhat more sophisticated. You can exchange messages (often called Electronic Mail) and also use the computer for your usual activities. Networking hardware is generally costly, requiring a host adaptor card in each computer and master server units that connect all devices like a chain.

Most home computerists at one time or another have had aspirations of creating their own mini-networks; but once they investigate the costs the idea is no longer appealing — that is, until the Ampro 186 PC-DOS Little Board and a program called SUPER DUO came along. You can now have your cake and be able to eat it, too, without choking on the price.

In terms of computing power and being able to adapt to an endless variety of applications, the Ampro 186 PC-DOS Little Board should be considered a "must" purchase for any owner of an IBM-PC or compatible computer who often needs more than one computer for program development, database management, but can't justify the extra expense of a second system.

With SUPER DUO your IBM-PC or compatible becomes a terminal for the Ampro 186 PC-DOS board. The unique aspect of this relationship is that each system retains control of its own functions but can be assigned computing chores simultaneously. If this sounds a little confusing, just bear with me.

Double Tasking

SUPER DUO was designed to take advantage of the Little Board's superior power and speed as a DOS engine (or co-processor) for PC compatible computers. Interfacing is simple. You connect either the PC's COM1 port to the Ampro's Serial A port using the same cable you now have connected to your terminal. Or, you leave your terminal attached to the Ampro's Serial A port and connect Serial B port using a DTE configuration whereby the TxD and RxD lines are reversed. The only other lines that have to be connected are the Signal and Chassis grounds. Handshaking does not have to be set.

SUPER DUO is a series of programs that accomplish several interesting things. When the respective programs are booted on the PC and the 186 Little Board, each computer expects to find the other computer to be connected to it. This establishes a communications path between the two computers, who retain the ability to operate in their native mode while sharing the PC's keyboard and monitor. A simple CTRL-A and CTRL-P toggles the control between each computer. A second set of programs allow you transfer programs and files.

What we have here is a classic example of a very powerful multi-tasking system that offers not only flexibility but some interesting applications. Before we discuss how to use SUPER DUO, I want to detail the the software installation.

Setting Up The System

First create a bootable system disk for each computer using the FORMAT B:/S option.

Copy the file named SDAMP.SYS to the Ampro from the SUPER DUO distribution disk. Do not include any of the terminal device drivers from the Ampro support software that you would normally use when setting up a standard system. SDAMP.SYS is the device driver that enables the PC to act as the terminal for the Ampro.

Create or add to your CONFIG.SYS file the following line:

```
DEVICE = SDAMP.SYS
```

This should be the first line on your CONFIG.SYS file. Then create an AUTOEXEC.BAT file that reads:

```
PROMPT [AMPRO] $P$G
```

When you boot up or reset the Ampro, the AUTOEXEC.BAT file will automatically load SUPER DUO without further user intervention.

Set up your PC by copying to the system disk a file named SDPC.SYS. Create a CONFIG.SYS file that contains this line:

```
DEVICE = SDPC.SYS
```

Then create an AUTOEXEC.BAT file that reac

```
PROMPT [PC] $P$G
```

That's all there is to the software installation. The default Baud Rate is 9600. You do not have to include this in the CONFIG.SYS file unless you wish to use a different baud rate. Both SDPC and SDAMP have several options that can be invoked, such as changing the default communications port, baud rate, a screen blanking options that will turn off the CRT screen after a predetermined length of time if no key is pressed, an option to assign different graphic cards within the PC for use by each computer, and the ability to reassign certain special function keys.

If you are going to use specific application programs with your computers in the SUPER DUO mode, you might wish to consider creating several different disks each containing those programs and including their names in the AUTOEXEC.BAT file. This will speed up the loading of the programs if you have to reset either system.

Using Super Duo

The first order of business is to establish just what you wish to accomplish and how you are going to use this system. Any operations that require assembling programs, linking files, sorting mailing lists, extracting files from large data bases or mathematical computations are best performed on the Ampro 186 because it will handle these tasks 300-400% faster than the standard PC or compatible.

If you are developing a program, load the Source Code into the PC and the Assembler into the Ampro. File transfers between the two computers is accomplished with SUPER DUO utilities SDCOPY (for the PC) and SDACOPY (for the Ampro). Type CTRL-P to make certain that you are in the PC command mode. The syntax to transfer a file from the PC to the Ampro is

SDCOPY <filename> TOAMPRO <destination>

The syntax for file transfers from the Ampro to the PC is:

SDACOPY <filename> TOPC <destination>

The destination is, of course, the drive letter to which you want the file transferred to. Assuming your source file was sent to the Ampro, type CTRL-A to toggle control to the Ampro. Enter the commands to begin the program's assembly. Type CTRL-P to toggle back to the PC to perform other tasks.

A neat feature of SUPER DUO is that you reset each computer independently of the other. Although the Ampro has full access to the PC's keyboard, it cannot use the CTRL-ALT-DEL reset sequence. Resetting the Ampro is still accomplished by pressing the reset switch. Therefore, when the Ampro is assembling the program, you can reset the PC, load in other work; and when the Ampro has finished its assembly, you can reload SUPER DUO into the PC again and toggle control to the Ampro to transfer the assembled program to the PC for a test run. This is especially productive if you are preparing a program that is modular in structure. As you develop each module in the PC the Ampro works in the background assembling your work. When completed, it can link them together.

But you don't have to be a dyed-in-the-weskit programmer to appreciate SUPER DUO. I use my system primarily for word processing. My Ampro is mated to a Shugart 1610-4 hard disk controller and a 20-MB hard disk as the default drive. I also have 2 48-tpi drives on line, but I seldom turn them on except to back up my hard disk files or to copy programs to the hard disk.

I don't have a hard disk on my PC because I don't use it that much for daily work to justify the expense. So, when I have the need to store notes and articles on a hard disk, I transfer them to the hard disk on the Ampro.

Since I usually have several articles in progress, I like to work on them at the same time. And SUPER DUO allows me to switch back and forth to the different systems while hammering away at the same keyboard. But while the Ampro cannot run any of the heavily-protected PC-DOS software, or software that writes pixel graphics or addresses the PC's video RAM, there are enough major programs that are compatible to both systems that I haven't felt constricted in my work.

However, when I'm in the SUPER DUO mode, I can have certain specialty programs resident in my PC for background productivity. To be specific, I have Borland's SIDEKICK resident on most of my PC word processing disks. I find that some of the SIDEKICK utilities are inappropriate for my needs, so I only use the Notebook, Dialer and Calendar modules. The Notebook option is a handy way of jotting down thoughts and references as they occur instead of littering my desk with dozens of slips of paper.

I have also found it useful to have the Ampro serve as a printer spooler. I miss this feature on my PC after having used SPOOLDISK 89 (a 128k plug-in print spooler and silent disk on my Heath/Zenith '89 computer) for many years. I load the Ampro 186 with the files I want printed out and go back to work on my PC.

As I was working on this article, I called Wordcraft and asked if they had additional programs available (or in the planning stage) for the Ampro 186. Charles Pine informed me that they have a powerful telecommunications program available called MICROLINK II, which has been especially configured for both the Ampro CP/M Little Board-PLUS and the Ampro 186 PC-DOS board to take advantage of their unique features. I hope to obtain a copy of MICROLINK II in the near future and prepare a detailed report.

In the meantime, Wordcraft is planning a series of enhancements for SUPER DUO that will increase its versatility.

Color Commander: Push A Button And Your RGB Monitor Becomes A Chameleon

When using SUPER DUO with an RGB monitor attached to your PC, your Ampro screen will only be in black & white, because the 186 PC-DOS board doesn't support IBM graphics at this time.

However, there is a new product on the market from Perma Power Electronics, Inc. (the people who make those stand-by power supplies) called the COLOR COMMANDER that will greatly enhance the color versatility of a standard IBM color graphic card. Used in conjunction with the 186 LB and SUPER DUO, the COLOR COMMANDER will perform some interesting color manipulations at the push of a button.

COLOR COMMANDER is a peripheral device that sits under your RGB monitor. Your computer's video cable plugs into the COLOR COMMANDER where the data bits, one each for red, green and blue, and one for intensity, are intercepted. A separate cable then connects the COLOR COMMANDER to the monitor. Inside this unit each four bit color data byte is decoded and programmed to light an LED, which then indicates the particular colors that are being called for by the software program.

The COLOR COMMANDER cannot add more colors to your screen, because the IBM-PC and compatibles in medium resolution mode of 320 × 200 requires 4000 bytes for each color. Thus the maximum screen memory of 16,000 bytes is quickly used up. Even more limiting is that you can only have one background color and three foreground colors to work with; and the actual colors from the pre-assigned "palettes" are sometimes restrictive.

What COLOR COMMANDER can do for you is to alter any of these colors to one that is more pleasing or appropriate for your graphic design. I liked the fact that I could alter the black & white screen output while I am working in the Ampro mode, as the double-dot image of word processing text becomes irritatingly difficult to read after a while, and the intensity of the white lettering on a black screen a strain on the eyes.

COLOR COMMANDER let's me change the black screen to white, and the white lettering to black; or the lettering to yellow and the foreground to brown. The effect is interesting to watch and gives one a feeling of omnipotent power over the limitations of the IBM color graphics adapter.

Using the COLOR COMMANDER for IBM specific applications, especially in high resolution graphic mode (640 ×

320), for architectural drawings or for CAD engineering and scientific graphs or drawings, you can greatly enhance the images by adding color. And, if you are dumping your drawings to a color plotter, you can actually preview the colors on the screen to see how they would actually translate on paper.

This is definitely a device worth adding to your arsenal of computer peripherals, made all the more attractive by its $359 price tag.

SUPER DUO costs $49.00 and is available from:
AMPRO COMPUTERS, INC.
67 East Evelyn Avenue
Mt. View, CA 94041

It is published by

WORDCRAFT
3827 Penniman Ave.
Oakland, CA 94619

.The COLOR COMMANDER Model CC100 is manufactured by:
PERMA POWER ELECTRONICS, INC.
5601 West Howard Avenue
Chicago, IL 60648

Miscellaneous Notes

My CP/M SCSI Little Board recently went belly-up during a hectic afternoon when I was switching equipment around in order to get all of my Little Boards hooked up to hard disks. When the dust settled and the last cable was installed, my CP/M LB refused to respond.

I have been told that the failure rate of these boards during normal use is less than 1%, so I naturally assumed that I had goofed and possibly destroyed one of the major components. I switched some chips with ones I had but it was futile. On those I didn't have, Ampro supplied me with replacements. Those didn't work either.

As a final resort, I returned the board to Ampro for a checkout. I called Fred Willink, head of Ampro's Technical Support, a few days later and he told me the board was working nicely. Seems I had removed a jumper during my fiddling and had forgotten about it.

Rule Number One: Don't fool around with jumpers unless you keep the technical reference manual handy.

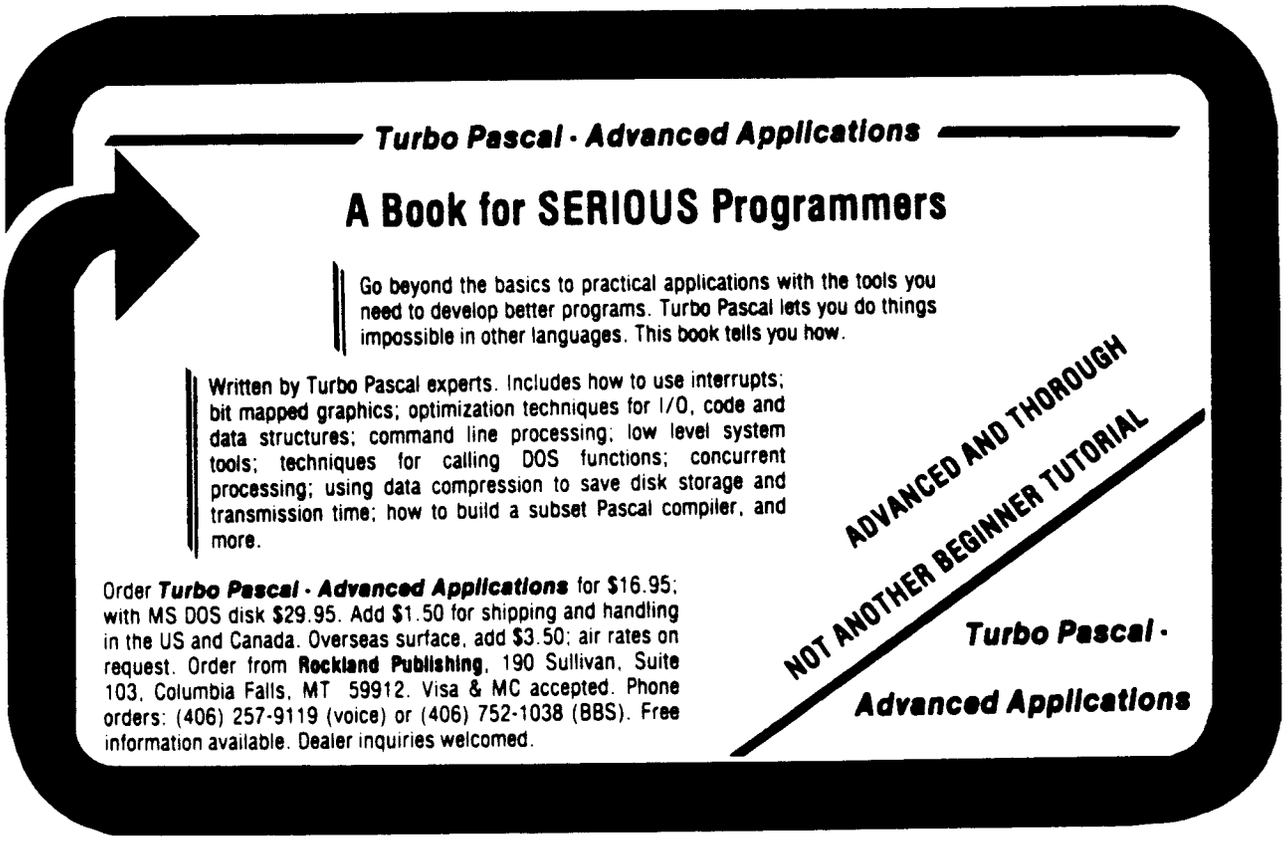
Hard Disks

"Device Not Ready" errors are always frustrating, especially when you are setting up a hard disk. But I was greeted with this persistent message when I tried to install my first hard disk on the Ampro 186 LB.

Ampro's Support Software Manual instructs you to type FORMAT C:/S/V in order to format the hard disk. A quick check into the IBM and MS-DOS manuals confirmed the syntax, but the same error message repeated itself.

Standard PC-DOS (and MS-DOS) generally assumes that you will only have two floppy drives on your system, thus C: becomes the designated assignment for the first hard disk. Ampro's ROM BIOS, on the other hand, supports up to 4 logical or physical floppy drives, and they are letters A: through D:. Your hard disk therefore becomes drive E:.

Something else to bear in mind when you format your hard disk after the HFORMAT partitioning program has run its course, type only FORMAT E:/S. Do not include the /V option. The system will not respond to this command. I don't know why. Are you reading this, Fred? ■



Back Issues Available:

Issue Number 1:
- RS-232 Interface Part One
- Telecomputing with the Apple II
- Beginner's Column: Getting Started
- Build an "Epram"

Issue Number 2:
- File Transfer Programs for CP/M
- RS-232 Interface Part Two
- Build Hardware Print Spooler: Part 1
- Review of Floppy Disk Formats
- Sending Morse Code with an Apple II
- Beginner's Column: Basic Concepts and Formulas

Issue Number 3:
- Add an 8087 Math Chip to Your Dual Processor Board
- Build an A/D Converter for the Apple II
- Modems for Micros
- The CP/M Operating System
- Build Hardware Print Spooler: Part 2

Issue Number 4:
- Optronics, Part 1: Detecting, Generating, and Using Light in Electronics
- Multi-User: An Introduction
- Making the CP/M User Function More Useful
- Build Hardware Print Spooler: Part 3
- Beginner's Column: Power Supply Design

Issue Number 5:
- Optronics, Part 2: Practical Applications
- Multi-Processor Systems
- True RMS Measurements
- Gemini-10X: Modifications to Allow both Serial and Parallel Operation

Issue Number 6:
- Build High Resolution S-100 Graphics Board: Part 1
- System Integration, Part 1: Selecting System Components
- Optronics, Part 3: Fiber Optics
- Controlling DC Motors
- Multi-User: Local Area Networks
- DC Motor Applications

Issue Number 8:
- Build VIC-20 EPROM Programmer
- Multi-User: CP/Net
- Build High Resolution S-100 Graphics Board: Part 3
- System Integration, Part 3: CP/M 3.0
- Linear Optimization with Micros

Issue Number 9:
- Threaded Interpretive Language, Part 1: Introduction and Elementary Routines
- Interfacing Tips & Troubles: DC to DC Converters
- Multi-User: C-NET
- Reading PCDOS Diskettes with the Morrow Micro Decision
- DOS Wars
- Build a Code Photoreader

Issue Number 10:
- The Forth Language: A Learner's Perspective
- An Affordable Graphics Tablet for the Apple II
- Interfacing Tips & Troubles: Noise Problems, Part 1
- Multi-User: Some Generic Components & Techniques
- Write Your Own Threaded Language, Part 2: Input-Output Routines and Dictionary Management
- Make a Simple TTL Logic Tester

Issue Number 14:
- Hardware Tricks
- Controlling the Hayes Micromodem II from Assembly Language, Part 1
- S-100 8 to 16 Bit RAM Conversion
- Time-Frequency Domain Analysis
- BASE: Part Two
- Interfacing Tips and Troubles: Interfacing the Sinclair Computers, Part 2

Issue Number 15:
- Interfacing the 6522 to the Apple II
- Interfacing Tips & Troubles: Building a Poor-Man's Logic Analyzer
- Controlling the Hayes Micromodem II From Assembly Language, Part 2
- The State of the Industry
- Lowering Power Consumption in 8" Floppy Disk Drives
- BASE: Part Three

Issue Number 16:
- Debugging 8087 Code
- Using the Apple Game Port
- BASE: Part Four
- Using the S-100 Bus and the 68008 CPU
- Interfacing Tips & Troubles: Build a "Jellybean" Logic-to-RS232 Converter

Issue Number 17:
- Poor Man's Distributed Processing
- BASE: Part Five
- FAX-64: Facsimile Pictures on a Micro
- The Computer Corner

Interfacing Tips & Troubles: Memory Mapped I/O on the ZX81

Issue Number 18:
- Parallel Interface for Apple II Game Port
- The Hacker's MAC: A Letter from Lee Felsenstein
- S-100 Graphics Screen Dump
- The LS-100 Disk Simulator Kit
- BASE: Part Six
- Interfacing Tips & Troubles: Communicating with Telephone Tone Control, Part 1
- The Computer Corner

Issue Number 19:
- Using The Extensibility of Forth
- Extended CBIOS
- A $500 Superbrain Computer
- BASE: Part Seven
- Interfacing Tips & Troubles: Communicating with Telephone Tone Control, Part 2
- Multitasking and Windows with CP/M: A Review of MTBASIC
- The Computer Corner

Issue Number 20:
- Designing an 8035 SBC
- Using Apple Graphics from CP/M: Turbo Pascal Controls Apple Graphics
- Soldering and Other Strange Tales
- Build a S-100 Floppy Disk Controller: WD2797 Controller for CP/M 68K
- The Computer Corner

Issue Number 21:
- Extending Turbo Pascal: Customize with Procedures and Functions
- Unsoldering: The Arcane Art
- Analog Data Acquisition and Control: Connecting Your Computer to the Real World
- Programming the 8035 SBC
- The Computer Corner

Issue Number 22:
- NEW-DOS: Write Your Own Operating System
- Variability in the BDS C Standard Library
- The SCSI Interface: Introductory Column
- Using Turbo Pascal ISAM Files
- The AMPRO Little Board Column
- The Computer Corner

Issue Number 23:
- C Column: Flow Control & Program Structure
- The Z Column: Getting Started with Directories & User Areas
- The SCSI Interface: Introduction to SCSI
- NEW-DOS: The Console Command Processor
- Editing The CP/M Operating System
- INDEXER: Turbo Pascal Program to Create Index
- The AMPRO Little Board Column
- The Computer Corner

Issue Number 24:
- Selecting and Building a System
- The SCSI Interface: SCSI Command Protocol
- Introduction to Assembly Code for CP/M
- The C Column: Software Text Filters
- AMPRO 186 Column: Installing MS-DOS Software
- The Z Column
- NEW-DOS: The CCP Internal Commands
- ZTIME-1: A Realtime Clock for the AMPRO Z-80 Little Board
- The Computer Corner

TCJ ORDER FORM

Subscription:	U.S.	Can & Mex	Surface Foreign	Total
6 issues per year ☐New ☐Renewal	1 yr. $14.00 2 yr. $24.00	$22.00	$24.00	
Back Issues #'s 1 thru 21	$3.25	$3.25	$4.75	
#'s 22 thru 24	$3.75	$3.75	$5.25	
#'s				
User Disk	$10	$10	$10	
Description:				
Size:				
Format:				
System:				
			Total Enclosed	

Check or Money Order on U.S. Bank in U.S. Funds.
Make payable to THE COMPUTER JOURNAL.

☐ Check enclosed ☐ VISA ☐ MasterCard Card # _____

Expiration date _____ Signature _____

Name _____

Address _____

City _____ State _____ ZIP _____

The Computer Journal

190 Sullivan Crossroad, Columbia Falls, MT 59912 Phone (406) 257-9119

*** **Orders can also be entered by modem on the TCJ BBS (406) 752-1038 by leaving a private message for SYSOP with the required information.**

Books of Interest

The Free Software Handbook
PC-DOS/MS-DOS Edition

A Book Review
by
Walter E. Pfiester
1 Skadden Terrace
Tully, N.Y. 13159

Introduction

This paperback contains instructions and documentation for using over 60 public domain (PD) pieces of software in 285 pages. Yes, the software is either PD or Shareware and is available from bulletin boards across the country. The software is free. This text costs $17.95 and should be on your shelf along with your manuals on DOS.

More than a collection of instructions, there are many sections devoted to BASIC, Graphics, Autoexec and communications. On a score of 0 to 10, I rate this publication a 9. The only reason it does not deserve a 10 is that some of the programs referenced have been superseded, however, the basic instructions for these packages (covered in this edition) still apply.

Style

The folio talks to the reader in a chatty form that is delightful, with a good deal of humor thrown in; while at the same time not insulting, even to the most advanced programmer. Many explanations are used to introduce various software concepts so that the reader understands the value of the methodologies used to solve a particular problem.

In many cases, CP/M software has been rewritten for MS-DOS. Where this is done, it is noted in the text. I applaud the authors for this straight forward approach and honesty.

Both the Table of Contents and the Program/File Reference (in the back of the text) are well done with short annotations. Twenty additional pages in the rear of the book contain advertisements for additional products available from Peopletalk Associates, the publisher.

Games

This handbook (about the same size as a Readers Digest) covers 24 popular games, including WILLIAM TELL (MUSIC) to Backgammon and some special goodies for you teachers out there: a Word Processor for kids and a States and Capitals game called MAP. Why this is included in the game section is a mystery to me, but then again I may not understand the name of the game.

Utilities

There are five additional sections to this book including Directory Assistance (File Management). With five programs under this title, these pieces of software are a MUST have: such as UNDEL.COM (undeletes erased files), and two versions of Sorted Directories.

In the Miscellaneous Utilities section there are eight programs including a Screen Blanker, Screen Typer, Squeezed file typer and a Word Counter. For uploading to distant computers, the WordStar to ASCII converter utility is imperative.

Libraries

Pages 143-170 cover three library systems. I criticize the book for not covering the ARC system (and the differences between .LBR and the .ARC files).

Two good cataloguing systems are also covered in this section (makes keeping track of files on floppies simple). If you have a hard disk, this system is still indispensable (for those backup and sacred floppies).

Communications and Hacking

QMODEM (destined to be a classic) and a simpler version: MINTEL are covered in this section. Nice programs, both! The last section is devoted to those of us that are in to modifying programs including a file editor (JAZ) to a Basic Cross Referencer (MBXREF.BAS).

Conclusion

For those of you with PEOPLETALK'S earlier FREE SOFTWARE HANDBOOK (for CP/M), you'll find this book written in the same style while still living up to their original level of excellence.

For the beginner, this book is indispensable reading while the more advanced user will-find new tricks and an array of programs in the public domain never thought possible. I have mentioned only a fraction of the programs covered in the text. At $17.95 the cost per program works out to be a paltry 30 cents.

I recommended (as a reviewer) their CP/M book years ago very highly. This MS-DOS version is just as eloquent and every bit as readable. It's a treasure house.

The Free Software Handbook
PC-DOS/MS-DOS Edition
By
Patricia Law Hatcher
and
Walter J. Palmer III
Published by
Peopletalk Associates, Inc.
P.O. Box 863652 Plano, Texas 75086

Programming with Turbo Pascal —
by David W. Carroll
Published by Micro Text Productions, Inc, 1985, 307 pages, 6 × 9".

This is an introductory level book, and the preface states that it focuses on learning the basic concepts of programming in Standard Pascal, as opposed to the advanced features and extensions provided by Turbo Pascal. The text includes a special limited Turbo Pascal version 1.0 compiler especially designed for this book, which supports most of the features of ISO Standard Pascal. The contents include the following:

Pascal Program Structure; Data Types; Expressions and Simple Statements; Procedures and Functions; Terminal Input and Output; Simple Programs; Decision Statements; Arrays and Strings; Records and Sets; Pointers and Dynamic Data Structures; Device and File I/O; Advanced Programs; Machine Level Interface; Advanced Compiler Topics.

Complete Turbo Pascal — by Jeff Duntemann
Published by Scott, Foresman & Co., 1986, 458 pages, 7½ × 9½".

Duntemann focuses on programming by successive refinement; beginning with a one sentance statement of what the program should do and breaking it down into subsidiary statements which lead to code development. There are many illustrative short code segments used to clarify each concept, and the programs are available from the author on disk. The contents include the following:

Turbo Pascal As a Programming Environment; Basic Elements of a Pascal Program; Programming by Understanding the Problem First; Successive Refinement; Simple Constants; Simple Data Types; Subranges and Enumerated Types; Data Structures; Pointers; Operators and Expressions; Statements; Controlling Program Flow; Procedures and Functions; How to Use Strings; Standard Functions; Controlling Your CRT; Console, Printer, and Disk I/O; Dynamic Variables and Pointers; Low Level Machine Hooks; Using the Turbo Pascal Editor; Using the Turbo Pascal Compiler; Turbo Pascal Overlays.

Using Turbo Pascal — by Steve Wood
Published by Osborne McGraw-Hill, 1986, 297 pages, 7 × 9".

This book covers the "how to" of basic Turbo Pascal in the first few chapter and moves right on to more advanced topics. It is not a rank beginner's book, but better perhaps for those somewhat familiar with Pascal. The book is broken into two parts—the first covering the Turbo Pascal environment, and the second building a practical applications program. An unusual, and very helpful, feature is the numerous tables that show which reserved words, predefined types, functions, and procedures are present in standard Pascal and Turbo Pascal for the 8 and 16 bit systems. The source code listings are available on disk. The contents are as follows:

Pascal as a Language; A Structured Approach to Programming; Parts of a Pascal Program; Definitions and Declarations; Statements; Coding Conventions; User Defined Types; Simple Types; Subrange Types; Structured Types; Arrays; Strings; Records; External Devices; Using Text Files and Devices; Set Types; Pointer Types; Parameters; Functions; Recursive Subprograms; Scope; A Sort Program; Turbo's Development System; Compiler Options; Video and Keyboard; External Devices and Disk File; String Handling; Dynamic Memory Allocation; Memory Manipulation; Type Conversion and Scalar Functions; Arithmetic Functions; Compiler Directives; Using Include Files; Using Chain and Execute; MS-DOS and CP/M Memory Allocation Options; Overlay Files; I/O Error Trapping; Absolute Addressing; Constants, Types, and Variables; Miscellaneous Global Subprograms; Controlled Keyboard Input; Valid Input Functions; Video Display Subprograms; developing a loan amortization program

---

Turbo Pascal Programming With Applications — by Dr. Leon A. Wortman
Published by Tab Books, 1985, 234 pages, 7 × 9".

This book is slanted towards business applications with many ready to use programs. The programs and their explanations also serve as tutorials. The book is not for beginners, as it assumes some familiarity with standard Pascal and with using an editor and a compiler. Each chapter presents complete source code for a program example as well as tutorial material designed to explain the function of the program and its interesting features. The chapters include the following programs which are also available on disk:

Counting Words of Text; Histogram Maker; Determining the Contribution to Profit; Measuring Cash in Float with DAYSCRED; Checkbook Program; Depreciating Capital Assets; Evaluating Investments; Mortgage and Loan Table Generator; Calculating Your Net Worth; Trip Report and Expenses; Control Codes to Printer; Using Time Trend Analysis to Smooth Rough Spots; Plotting Equidistant Data Points; Case Statement in English/Metric Conversions; Alphanumeric Sorting; Personal Appraisal; Break Even Point; Commissions for Sales Representatives; Analyzing Sales Performance; Four Function Calculator; Copying an ASCII file to Disk, Display, or Printer; Three Part Database Program.

FOR TRS-80 MODELS 1, 3 & 4
IBM PC, XT, AND COMPAQ

WHICH FORTH has all the POWERFUL APPLICATIONS?

- DATAHANDLER database
- FORTHWRITE word processor
- FORTHCOM communications
- GENERAL LEDGER accounting
- GAMES for fun and technique
- EXPERT-2 expert system
- TRADESHOW commodities terminal
- GRAPHICS, 8087 support, many other utilities

You've Been Thinking About It. Isn't It Time to Put It to Work?

MMSFORTH

The total software environment for IBM PC, TRS-80 Model 1, 3, 4 and close friends.

- Personal License (required):
  - MMSFORTH System Disk (IBM PC) . . . . $ 99.95
  - MMSFORTH System Disk (TRS-80 1, 3 or 4) 129.95
- Personal License (optional modules):
  - FORTHCOM communications module . . . . $ 39.95
  - UTILITIES 39.95
  - GAMES 39.95
  - EXPERT-2 expert system 69.95
  - DATAHANDLER 59.95
  - DATAHANDLER-PLUS (PC only, 128K req.) 99.95
  - FORTHWRITE word processor 175.00
- Corporate Site License Extensions . . . . . from $1,000
- Some recommended Forth books:
  - UNDERSTANDING FORTH (overview) . . . $ 2.95
  - STARTING FORTH (programming) 16.95
  - THINKING FORTH (technique) 15.95
  - BEGINNING FORTH (w MMSFORTH) . . . 16.95

Shipping/handling & tax extra. No returns on software.

Ask your dealer to show you the world of MMSFORTH, or request our free brochure.

MILLER MICROCOMPUTER SERVICES
61 Lake Shore Road, Natick, MA 01760
(617) 653-6136

Editor

(continued from page 1)

pins we'll need will be for input, output, and power — all the interconnections will be within the chip! I'm predicting that we'll see mid-level micro computers with everything (including 1MB or more of RAM) on one chip. Besides the pressure of economics, this is the only way that we'll be able to provide the short signal paths mandatory for operating 32 bit systems at 25 to 50 Megahertz.

Hacking the 68000

The primary technical advances in the micro industry have come from pioneering individuals and small entrepreneurs. The large manufacturers wait until the designs are proven, but by that time they are technically outdated.

I've been looking for 16 or 32 bit system suitable for hardware and software hacking, without being isolated with a one-of-a-kind oddity. I want to be able to go off in my own direction, doing my own thing, but I still want to be able to use supported utility programs and to correspond with others working with the same general system. The IBM-PC doesn't interest me at all (except possibly as an appliance to run certain software). I'm more interested in the 68000 series, but the Mac, the Amiga, and the Atari systems are too closed and aimed at a graphics and sound market which isn't my area or interest. What we need is a good 68000 hacker's system!

While at SOG, I discussed this with Joe Bartel (Hawthorne Technology, 8836 S.E. Stark, Portland, OR 97216, (503) 254-2005) who has developed a 68000 multi-user system based on his own board and operating system, complete with editor, assembler, and programming language. He has offered to head a 68000 user group based on a wire wrap board (maybe printed circuit later) with his modified single user operating system. He estimated the price of the operating system with editor, assembler and language at $50 to $100. Dave Thompson will be supporting this thru Micro Cornucopia, and we will cooperate by supporting it without duplicating Dave's coverage. My impression is that this will be an open system which we can hack on instead of the current "take it or leave it" closed boxes. Let me know if you're interested (use the BBS if you have a modem) and perhaps we'll start by adding a generic 68000 section to the board.

Management Styles

Why is Japan beating us in so many areas? It's not just due to lower wages because they produce better designed, higher quality products with less worker-hours per unit. One of the very significant reasons is that they base their decisions on where they want to be five years or more in the future, while our managers concentrate on the bottom line of the next quarterly report! They plan for the future because their managers will probably still be there, while we plan on making a fast buck next quarter to impress the venture capitalists because our managers will have moved somewhere else before the roof caves in.

Another aspect of this difference was pointed out in Carole Patton's column in the April 3, issue of Electronic Design. Carol talks about the lack of engineers in the U.S. management ranks and reports that Yoshi Tsurumi, an expert in international business, stated that since 1950, two-thirds of the chief executives in U.S. Fortune 500 coprorations have come from the financial, legal, and advertising ranks. In contrast, two-thirds of Japan's top corporations are led by individuals with scientific, engineering, or sales expertise. "The financial, legal, and bureaucratic manipulation of the corporation and its employees has come to be equated with corporate management," Tsurumi said. But is technological leadership possible without technical leaders?

An example of how the U.S. management's attitude sends business to Japan was related by a letter from M. Tony Young (Electronic Design, May 15, 1986). Tony states that they needed a slight package size modification and said, "It looks like we're ready to hand Japan another victory. Many, and especially the largest of the U.S. manufacturers, are refusing our requests for a new package with obvious benefits, while the Japanese are responding positively." He also said, "The ZIP is a definite winner in high-performance applications. U.S. manufacturers of ICs had better wake up to that fact before they lose another proving ground, and the revenue that goes with it." Our engineers are tops, but our out-of-touch bureaucratic managers insist on deciding what the customer shall be allowed to use, while the Japanese provide what the customer wants!

Another study pointed out that a large percentage of the Japanese engineers spend their time out in the production area while our engineers spend their time in a distant office. Japan is beating us because we are no longer competetive in manufacturing techniques, producing a well designed quality product, or listening to our customers.

Required reading for anyone really interested in the industry should include Electronic Engineering Times (600 Community Drive, Manhasset, NY 11030), Electronic Design (P.O. Box 1039, Southeastern, PA 19398-1039), and EDN (P.O. Box 5262, Denver, CO 80217-5262).

Generating Source Code

Do you ever get the urge to dig into a program to find out how it works, or to make some minor patches? How can you do this if all you have is the .COM file? The answer is to disassemble it and generate commented source code. This can be very difficult, but Clark Calkins (C.C. Software, 1907 Alvarado Ave., Walnut Creek, CA 94596) has developed source code generators for CP/M 2.2, CP/M+ 3.1, and the Z-80 version of Turbo Pascal 3.0. These are really amazing, as they generate fully commented assembly source code files which can be studied, revised, and reassembled.

These techniques can be very helpful for fixing bugs, and Clark has promised to write a series of articles describing how he goes about generating commented source code from a .COM file. Until then, get his very reasonably priced programs to learn the secrets of CP/M and Turbo Pascal.

CAD to the Rescue!

We hear a lot about how the large companies are using CAD (Computer Aided Drafting), but don't always relate this to our own small scale operation. The schematic drawing for the SCSI article in this issue was prepared by Judy Skinner (Computer Wizards, P.O. Box 5326, Aloha, OR 97007 (503) 642-9617), and is a vast improvement over our hand drawn efforts. Judy is very knoweldgeable and helpful, so be sure to check with her for your CAD and CAE needs, including PCB layout, CAD training, selecting a CAD system, mechanical design, etc. ■

ZSIG

by Jay Sage, ZSIG Software Librarian

This article inaugurates a new feature of The Computer Journal, a column devoted to the activities of the recently formed organization called ZSIG. At the beginning of the year Richard Jacobson, Bruce Morgen, and I founded ZSIG, which stands for ZCPR Systems Interest Group. When Digital Research's CP/M operating system first became popular and attracted a significant group of programmers who freely offered their programs, complete with source code, to the CP/M community, organizations like CP/MUG (CP/M User Group) and SIG/M (Special Interest Group/Microcomputers) were formed to promote and distribute that software to a wide audience. Today, Echelon, like the Digital Research of years ago, has reached the point where it offers a solid line of commercial Z-System products to a growing and enthusiastic user community. Richard, Bruce, and I thought that it was now time for the supporters of advanced 8-bit operating systems to form an organization to promote the development and dissemination of public domain Z-System software.

Who's Who and What's What in ZSIG

When I first proposed a Z-System support group, Bruce Morgen's NAOG (North American One-Eighty Group), formed to support Steve Ciarcia's SB180 computer and other computers based on the Hitachi HD64180 chip, was already well established. Bruce suggested that ZSIG be made a part of NAOG, and thus the full organization became NAOG/ZSIG. Bruce heads it and writes and produces the newsletter "One-Eighty File."

Since information about public-domain software tends today to be exchanged more by remote access computer systems (RASs) than by mailing diskettes, we wanted to have a large, centrally located RAS to serve as a focus for ZSIG telecommunications activities. Richard Jacobson volunteered one system of his dual Z-Node in Chicago. The Lillipute Z-Nodes are usually available on a yearly subscription basis, but Richard arranged to provide access to a special NAOG/ZSIG directory in System 1 for all ZSIG members. A reduced price subscription to the entire system is also available for members.

I took on the position of software librarian, with the duties of organizing collections of appropriate software into volumes for official release, encouraging the submission of new programs, and regulating the way in which new revisions are generated and released. I am in the process of assembling a committee to help review software submissions so that we can try to maintain a consistently high level of quality and reliability in the programs we release.

Joining NAOG/ZSIG

Application forms for NAOG/ZSIG are included in the file ZSIG-FOR.ALL available from many Z-Node RASs. Alternatively, request an application from from NAOG/ZSIG at P. O. Box 2781, Warminster, PA 18974. If you are impatient and just can't wait to join, simply send the following to the above address: your name, address, and telephone number and a U.S. dollar check or money order for $15 ($25 if beyond the range of a 22-cent stamp). If you want to subscribe to the full services of the Lillipute Z-Nodes, include an additional $35 (regular fee is $40 per year). Please also indicate with your application if you consent to making your name, address, and phone number available to other ZSIG members.

Submitting Programs to ZSIG

ZSIG has two aims. First, we want to encourage cross-fertilization among Z-System programmers on every level of expertise and between users and program contributors. On the other hand, we want to avoid a chaotic pattern of upgrades as has occurred in some public domain software channels. As a result, we intend to exert a greater level of control over the programs that are submitted to ZSIG. A formal document describing the procedures to be followed will have been released by the time this column appears in print.

Here is a brief summary of some of the main points. First of all, we very much want your participation, and you do not have to be a member of NAOG/ZSIG to submit programs. The programs, however, must be ZCPR3 or Z-System oriented. That means that, as a minimum, they must know about the ZCPR3 environment (Z3ENV). File specifications should allow the ZCPR3 standard DU and DIR (named directory) forms, and any screen oriented output should obtain the necessary screen codes from the TCAP so that no terminal specific installation is required. The use of the SYSLIB, Z3LIB, and VLIB libraries is strongly encouraged, both for ease of programming and for maintainability.

Programs can be submitted in a number of convenient ways—either by modem or by diskette. Modem uploads should be made, if possible, to my Newton Centre Z-Node #3 in Massachusetts at 617-965-7259. Make the upload to the private area (use the 'RP' – receive private – option of XMODEM or KMD). Alternative upload submission nodes are the Lillipute Z-Node #15 in Chicago at 312-649-1730 or Al Hawley's Ladera Z-Node #2 in Los Angeles at 213-670-9465. Non NAOG/ZSIG members who wish to upload to the limited access Lillipute Z-Node should make a special arrangement by leaving a private message to the sysop.

Software submitted on diskette should be sent to me at 1435 Centre St., Newton Centre, MA 02159. IBM 8" standard SSSD or any 5" format that can be read by the Ampro MULTIDSK program or by the MediaMaster program on an IBM-PC/AT is acceptable. Among the more than one hundred acceptable 5" formats are the following: Ampro (48 and 96 tpi, single and double side), Kaypro (all), Morrow (all), Osborne (all), Xerox (all), and VT-180.

Submissions should include source code, an object file, a DOC file, and a ZCPR3 format help file. Special arrangements can be made in cases where the author desires to treat the source code as proprietary, and we also have volunteers who will prepare the

HLP file from the DOC file if you are unable to do it. Be sure to include your name, address, and voice phone number in a separate file for ZSIG use if it is not already in the release material.

The source and object code should bear a copyright notice of the form: Copyright (current year) by (name of author). The source can include a release for free personal use of the program. While ZSIG encourages and promotes the distribution of free software, we feel it is desirable that the author and ZSIG retain some control over modifications to a ZSIG library program. We will employ a checkout system whereby any person suggesting a program improvement, whether a ZSIG member or not, will be given a set period of time in which to work on the program and resubmit it.

There is an important role in ZSIG for non-programmers also. The real ingenuity in software is not so much in the actual coding as in the conceiving of new program ideas. So even if you can't write a program yourself, if you have a good idea, send it in! In the next column in The Computer Journal I plan to discuss some ideas for new programs and program enhancements. I have a few idea, but I sure could use some more from you. ■

Letters

(Continued from page 3)

S-100 & SB180

Thanks for the second copy of TCJ. I was beginning to wonder if it was going to come as I was really looking forward to it after the first issue. Finally a really useful magazine and thank god not too much MS-DOS IBM crap.

Here we are running both an S-100 system under ZRDOS+ as well as an SB180 with two 5¼" 48TPI drives (drives A & B), two 8" DSDD 1.1MB floppys (drives C & D), plus a NES FD1165 hard disk with Adaptek ACB-4000 controller (40MB potential).

Unfortunately the BIOS and hard disk formatting program only allow us to get 32.6MB at 8.152MB/drive (drives E, F, G, & H). Hopefully we will get an answer to that soon. Maybe one of your readers has done it already. Any help would be appreciated.

Keep up the good work and keep TCJ coming. If you maintain the present standards you will be a real winner in a worl buried in the mediocrity of IBM.

J.T.

An Information-a-holic

The reason that I ordered TCJ was because I wanted to find out about the SCSI interface and what made it tick. It seems that I got a lot more than what I bargained for.

Right now it is 16:15 and the mail came at 12:00. Until now, I had a very busy schedule today. I have guests coming over for dinner, I absolutely had to get out two reports, my wife is pregnant and I'm trying to figure our where the baby will reside in our small home.

I have not done one thing today except read TCJ. And now, with the cooperation of my Otrona Attache and WordStar, I'm banging out a letter to you. It all started when I read, "Restating Our Objectives" in issue 22. Your objectives and mine are darn near identical. I have tried in vain to find a publication like your's since I got this little machine in 1983. Like you "I want the computer to do what I want it to do." From standard user applications like WordStar and dBase to controlling real time situations in and around my home. I tinker. Sometimes I get somewhere, sometimes I don't. I have been involved in the electronics industry (generally at the managerial level) for 23 years.

I am not a designer or even a technician, but I can generally fix a busted radio or stereo or TV camera. I even re-aligned my "A" drive when it failed to boot the OS. Professionaly, I'm a small business consultant. I am in fact "a generalist" with excellent problem solving abilities. Once I understand the lexicon of any discipline and the concepts behind the words, I can do nearly anything. If I have a problem, I know who to turn to for assistance.

Now that you have some basic information about myself, let me explain some of the tools that I need to allow me to make my machine do what I want it to.

Most articles are informative, to a point. It gets down to the lexicon. "It was rather simple to port NEW.PGM over to the Kaypro 2." I realize that "port over" means to change the program in such a way that it works on a different hardware/software system. What I don't understand is "what" was changed to accomplish the task. As far as memory (RAM) is concerned; what is virtual and paged? What is "interleaving" on a Hard Disk?

Although I haven't had time to look close at NEW-DOS, I'll bet I don't find a thing about the BIOS. I realize that the BIOS is configured for each specific computer, what I don't know is how or why it works within CP/M. I caught a glimmer from NEW-DOS that seems to say you can do with CP/M what you can not do with MS-DOS because of the ROM BIOS. Does this mean that part of MS-DOS is on the disk and the other part is locked up in ROM? My little Otrona has "bit mapped graphics." How, in a program can I access the graphics RAM and write that "Bit Map" pattern (read Graphic) to a disk file?

Although the questions posed above are real and I would like them answered, they are in this instance rhetorical. Just because some people would like to hack at a pc-board or an operating system is not sufficient. The concepts; the big picture is needed first. People like myself would like to see a continuing column on the very basics of computers. A column that would explain the nature and concepts of these words that we hear batted around by the people that make new things happen. They had to learn it somewhere.

Because of my occupation, I recommend many computer systems. For smaller organizations many times I suggest CP/M systems after telling them how the current market is; I give them the facts and they make the decision. But here, we are talking about straight business users. Most customers I steer to the Leading Edge Model "D" machines. I've never had a problem with them. But I also, at times call in specialista for larger applications using UNIX and other systems. I do keep up on new products in hardware and software as well as the business in general.

In truth I am an Information-a-holic.

In issue 23, you ask about Bus Information. I am not bored. Although I have never seen a VME or STD Bus system, I feel the possibilities are endless. Just yesterday I got some information from WinSytems, Inc. (PO Box 121361, Arlington, TX (817) 274-7553) on their HD-64180 boards. I am real excited about the 64180 and am contemplating the purchase of the SB-180. I believe that a good pro/con debate on the virtues of each of the bus systems could be explored in TCJ. I don't think any one would object, and it might stimulate the imagination.

One final note. I admired your comments on source code and its availability, and copy protection schemes. When I purchased my Otrona, I paid $2,500. That was a lot of bucks even two years ago. I did it because it came complete with many (good) application programs and

utilities, was already modified with ZCPR, and most important, I got a full copy of the BIOS source code and a complete service manual. Even though I still can't use the BIOS now, when I need it, I'll have it. It was a major factor in my purchase.

A very close friend in my area has a Compaq, loaded to the nines. He laughed at my little 14 pound Otrona at first. Then one day we were working together and he noticed that my 8 bits were working nearly a third faster than his 16. He also commented on how nice it was to be able to set the baud rates on both serial ports by simply hitting the Control and Escape keys, then selecting the port I want and step to the rate needed. He likes the feel and the layout of the Atache keyboard as well. He no longer laughs at 8 bit hachines.

The one thing that I would like to ask your readers is: "Does anyone know what the format of the Attache DSDD 48 TPI is and if it is compatible with any other system? For the matter, if anyone know how to access the "Bit mapped Graphics" I'd love to know.

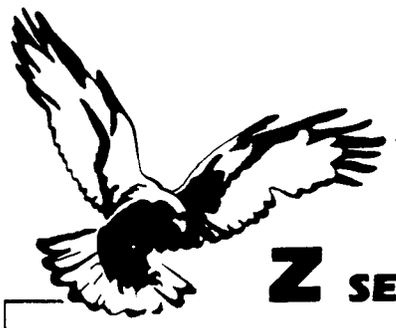
Thanks for a good read, and a lot of good information. Hope I have helped to answer some of your concerns, and I thank you for any help you can give to mine.

P.H.

Editor's Note: You raise a number of very interesting questions, which I'm sure that our readers will respond to. As far as the BIOS is concerned, Tom Hilton is working on the BIOS section of his series.

CP/M & WordStar Tips

Do not use the underscore "_" character as any part of a file name if you are using the CP/M operating system. Any number of errors will occur as a result of doing this (even though the character seems to be valid). Many other operating systems permit this character (even encourage its use, ie DEC's VMS). This character can be changed by using any of the popular machine code editors or by "Renaming" the file from inside WordStar. Yes, a file name with the underscore character can be created with WordStar, but try reading this file later, using "TYPE" or erasing it. Forget it! The operating system will not recognize the file. (All is not perfect with DEC either, the VMS operating system does not permit the dash "-" as part of the file



Z SETS YOU FREE!

Free to create computer environments right for you . . . free to automate repetitive tasks . . . free to increase your productivity. **Z-System**, the high-performance 8-bit operating system that flies! Optimized assembly language code — full software development system with linkable libraries of often needed subroutines —relocating (ROM and RAM) macro assembler, linker, librarian, cross-reference table generator, debuggers, translators, disassembler — ready to free you!

TERM III New generation communications package provides levels of flexibility, functionality, performance not available until now. Replaces BYE and XMODEM. master server local area network capability . . . public or private bulletin board and electronic message handling are integral features . . . auto-dial answer . . . menu install . . . XMODEM (CRC Checksum), MODEM7 Batch, Kermit, CIS, and XON/XOFF protocols . . . 100-page manual **$99.00**

Z-MSG Rolls Royce of message handling systems . . . mates with TERM III or BYE for most advanced overall electronic mail file transfer capabilities . . . menu installed . . . extreme configurability . . . many levels of access and security . . . word, phrase editor, field search . . . complete message manipulation and database maintenance **$99.95**

DISCAT Elegant, menu and command-line driven file and disk catalog manager. Generates and controls multiple master catalogs, working catalog used for update quickness. Nine flexible modules easily altered by user for custom requirements. Works with Z shells (VMENU, VFILER, MENU), aliases, and multiple commands per line **$39.99**

ZCPR3: The Manual Bound, 350 pages, typeset book describes features of ZCPR3 command processor, how it works, how to install, and detailed command usage. Bible to understand Z-System **$19.95**

ZCPR3 and I/OPS Loose-leaf book, 50 pages, 8-1/2" by 11", describes ins-and-outs of input/output processing using Z-System. Shows how to modify your BIOS to include I/O redirection . . . complements **The Manual** **$9.95**

More missing links found — Z Application Progams! Fly with eagles! Our programs promote high performance through flexibility! Productivity results from dynamically changeable work environments, matching operator to tasks and machines.

Above programs require 48K-byte memory, ZCPR3, Z-Com, or Z-System, and Z80, NSC800, HD64180-based computer. Shipping from stock. State desired disk format, plus two acceptable alternatives. As payment, we accept Visa, Mastercard, personal checks, money orders, and purchase orders from established companies. We also ship UPS COD.

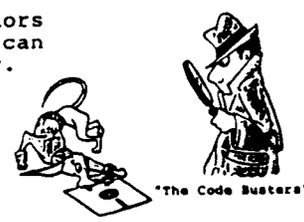
Call or write to place order or to obtain literature.

Echelon, Inc. 101 First Street • Suite 427 • Los Altos, CA 94022 • 415 948-3820

Ever Wondered What Makes *TURBOPASCAL* Tick?

Source Code Generators by C. C. Software can give you the answer.

"The darndest thing I ever did see..." "... if you're at all interested in what's going on in your system, it's worth it." Jerry Pournelle, BYTE, Sept '83



MasterCard VISA

The SCG-TP program produces fully commented and labeled source code for your TURBO-Pascal system. To modify, just edit and assemble. Version 3.00A (Z80) is $45. SCG's available for CP/M 2.2 ($45) and CP/M+ ($75). Please include $1.50 postage (in Calif add 6.5%).

C. C. Software, 1907 Alvarado Ave. Dept M
Walnut Creek, CA 94596 (415)939-8153

CP/M is a registered trademark of Digital Research, Inc.
TURBO Pascal is a trademark of Borland International

Advertiser's Index

AMPRO Computers 21, 27
Apropos 34
BD Software 3
Bersearch 33
BV Engineering 5
C.C. Software 45
CEC 6
Computer Journal 38, 39
Computer Trader 21
Echelon, Inc. 12, 45
Integrand 15
Jerryco 18
Kenmore Computer Tech. 29
LDL Electronics 17
Miken Optical 16
Miller Microcomputer Services ... 41
Periphco 8
Peripheral Land 22
Public Domain Software 47
Rockland Publishing 37

name). Guaranteed, you will also drive magazine editors nuts if your manuscripts are delivered on disk with files having the underscore as part of the file name. This is not a good way to start a relationship with editors! If editors use VAX'es, reread the above paragraph and pay attention to the dash.

The bottom line: If you have any files that have the underscore character as part of the file name, rename them by using WordStar's "Rename" facility at the opening menu.

Typing ^P<ESC> in a WordStar document will result in a ^[being shown on the screen. This is the escape sequence. It is now imbedded in the text, and may be outputed to ANY device. I use this trick to control my Gemini 10X printer, which can use over 55 <ESC> codes. Follow the ^P<ESC> sequence by whatever command you want to go to the printer (or any other device hung out there on your RS-232 or Centronics ports). See the back pages of your printer's manual for the codes needed.

Not finished, yet! These commands will NOT output from inside of WordStar. After leaving WordStar (you did use the non-document of WordStar, right?) use the LIST command or ^P (if you don't have the LIST command). Yep, this'll work with MS-DOS too!

Ben Guyer told me of this one; first credits to the VISUAL1050 newsletter. What a short tip, What a powerful command! Thanks, Ben.

W.P.

SB-180 and Z-System

I'm writing to let you know that "The Z Column" is a most useful series, and I hope you continue it for many issues! I've just set up a MicroMint SB-180, and it's a great system, but I found it slow getting started. Your series of articles will be very helpful. Specific comments:

(1) I'd like to see more on ALIASES and startup files.

(2) Is there an overlay or installation package for MODEM7?

(3) Don't forget the SB-180! Although the AMPRO systems had a head start, I bought the SB-180 because of the RAM disk and the ability to read both 5¼ and 8 disks. What's involved in adding a hard disk, like the $399 25MB ST225 Seagate from Priority One?

(4) The Z-System is public domain, but surely the BIOS from MicroMint or AMPRO is not public domain; specifically, which files is one allowed ot copy for friends?

(5) Finally, please include author's addresses (and maybe three lines on what they do for a living and for a hobby); it adds a nice personal touch, and facilitates getting in touch with them.

Delighted to see a journal for 8-bit machines! Keep it up.

D.D.

■ ■ ■

Repairing Printed Circuits

(Continued from page 9)

In the article I mentioned visiting your local blacksmith to learn some of his or her methods, I wasn't really being facetious as I actually did so one pleasant afternoon at Mystic Seaport in Mystic, Connecticut which is an historical re-creation of an mid 19th Century New England Seaport complete with various supporting trades such as the blacksmith who made ship's hardware. You can watch him work and even ask questions. There are similiar museums throughout the country, don't miss the blacksmith shop if you are able to visit one. Or by all means visit your library as there are many excellent books on working metal.

In a couple of places I mentioned using the schematics, now some of you may find a schematic to be about as readable as hieroglyphics. I possess and have seen many excellent books on reading schematics so again a visit to your local library may open up the mystery of electronic schematics. If you wish to buy your own, Hayden Book Company, 50 Essex Street, Rochelle Park, NJ 07662, publishes "How to Read and Interpret Schematic Diagrams" by J. Richard Johnson, Cat. 0868, price $9.95. Hayden has a toll free order line 1-800-631-0856. Howard W. Sams & Co. Inc., 4300 West 62nd Street, Indianapolis, IN 42602, has "How to Read Schematic Diagrams" by Donald E. Herrington Cat. 21127, price $7.50. Both books may be available at your local bookstore or can be ordered directly. The Johnson book includes a section on computer schematics so may be better for those principally interested in computers. ■

NOTE: For further information, see O'Connor's articles:

- Soldering and Other Strange Tales, in issue #20.
- Unsoldering the Arcance Art, in issue#21.

Computer Corner

(Continued from page 48)

the fastest way to find the correct chip was to remove one from a socket bank and see which memory chip it was. Usually you will find the rows to be the same in each bank but each row doesn't have to be in any order. I have worked on systems in which their order can be 1,4,7,3... and without PC board markings it is almost impossible to find a given bits row. My checking proved that in this case their order is parity,1,2... and so my second soldering job had the unit up and running.

On soldering new chips into units, the most common problem I see with other technicians is the desire to try and save the old chip. Chips are cheap, but PC boards are not. I use a pair of close cutting snips to remove the chip first (just cut all the pins as high on the chip as possible). This leaves the lead sticking out where you can grab it with a pair of pliers. Add solder to the pins on the back side (this helps heat transfer from the iron) and then heat and remove the pins. Next use a large style solder sucker (I find the small ones bounce too much, getting poor sucks) and clean the holes. Should you have trouble sucking the hole clean just add more solder and try again (if there is too little solder around the hole, you will just heat the foil off the board). Put your new socket in and resolder, checking for solder bridges. I have cut the pins flush with the board and then removed the solder and remainder of the pin with the sucker. This however can clog up the sucker causing more problems (or worse take too much heat and lift the foil).

My use of debuggers spans more than finding bad memory locations. When I talked about the EPROM emulator I talked about how I use it instead of fancy programs. This comes up quite often, where I find someone going through all the trials of making a utility program which reinvents the wheel. Most debuggers have the disk and I/O routines to handle most problems. I have used several PROM burning programs and still like mine best. Under CP/M, DDT/Sid/Zsid, all use RST 7 as their re-entry point. What you do is write your routines, so they end by calling RST 7. This will jump to location 38h where the debugger has put its re-entrant jump code. Doing this will give you the DDT prompt after it has completed your routine. For PROM burning I use DDT to load the code into memory, then jump to the PROM burning code which does a RST 7 after the PROM is burned. If that was all I did other programs would be just as easy, but there are times when I'll read two 2716s into consectative locations (I can specify starting address) and then burn a 2732 PROM. Most fancy programs would have you getting out of them after saving two seperate files. You would then use DDT to combine them into one new file. Then the new file would be used to burn the new PROM. The result was in and out of several programs to do what could have been done in one.

For use with the emulator it is possible to assemble code using the assemble option, load the new code to the emulator's RAM,reset the test systems, and check code for proper operation. If the operation proves to be missing something, just add it with the subsitute option, then rewrite to RAM. In checking the I/O of my 68K S100 system, I was able to check out problems by changing single words of code and see the results immediately. This took a half hour to do several days worth of PROM burning and code perfecting. For code perfecting the debuggers are also great. There have been many times when I'am not sure what the results will be for a certain routine. I usually have a good idea but feel it would be nice to be sure, especially if a PROM must be burned. To answer the question I simply load a program into memory (usually assemble option) then do a trace routine on it. In DDT to do a trace, set the program counter to the beginning of the routine with a Gxxx,xxx, that is short for go there and stop. Next simply do a T10, that is trace ten hex times and you will get a display of all your registers and their values as they change through the program. Many a time I thought I had the proper mask figured out for I/O option only to test it and find I was getting the zero flag set when the results was to be a non-zero operation.

What I am getting at is how fancy programs are not always time savers. If you can learn enough about what it is you are doing and how your system works, it is possible to do things faster and easier without large and expensive programs. Memory testers are fine but Debugging gives the ability to check many different functions. You can have sticking bits, bits that don't change, address decode problems, and refresh problems that will show up under buggers if you know what to look for. Experience is still the best teacher and using other programs will only keep you from knowing more about your system.

Parting Shots

A while ago I compared old style computers (like CP/M) to the automobile industry. Recently, I have seen others doing the same. The last comparison called CP/M the "VW Beetle" of the computer industry. I find that both amusing and enlightening, especially every time I see one on the road. When comparing the two, it is possible to see how there will proably be positions available for all levels of computer people, just like there is for all type of people in the automobile industry. What has taken new meaning now is my old statement of using used cars salesmen to sell computers. What I'm waiting to hear is how a little old lady only used the system once a week to balance her checkbook. ■

GET PUBLIC DOMAIN SOFTWARE!
HUNDREDS OF FREE PROGRAMS AVAILABLE TO COPY!

PUBLIC DOMAIN Software is not copyrighted so no fees to pay! Accounting, data-base business, games, languages and utilities free for the taking! Some of these programs sold for hundreds of dollars before being placed in public domain. Join hundreds of users enjoying a wealth of inexpensive software. Copy yourself and save!

USER GROUP LIBRARIES

	Rent	Buy
IBMPC-SIG 1-390 Disksides	$410.00	$850.00
IBMPC-BLUE 1-154 Disksides	$175.00	$435.00
SIG/M UG 1-240 Disksides	$155.00	$650.00
CP/M UG 1-92 Disksides	$ 45.00	$250.00
PICO NET 1-34 Disksides	$ 25.00	$100.00
KAYPRO UG 1-54 Disksides	$ 65.00	$200.00
EPSON UG 1-52 Disksides	$ 65.00	$200.00
COMMODORE CBM 1-28 Disksides	$ 25.00	$ 65.00

Get a PD User Group Catalog Disk - $5.00 PP — Specify Format!

Library rentals are for seven (7) days after receipt, three (3) more days grace to return. If you use your credit card — no disk deposit. Shipping, Handling and Insurance $9.50 per library. Call (619) 727-1015 for 3 min. recording. Call (619) 941-0925 orders and tech info.

DINERS CLUB AM EX VISA

NATIONAL PUBLIC DOMAIN SOFTWARE
1533 Avohill Drive, Vista, CA 92084
1-800-621-5640 wait for tone, dial 782542

THE COMPUTER CORNER

A Column by Bill Kibler

Well time keeps screaming by and it seems like I have more projects than time but then what's new. I got dragged into some performing again (folk dancing) and another editors job, so I am still behind on my major projects. Let's review some minor additions to my computer mania and talk about debuggers.

I must admit that this is being written on a clone. My work has reached a point where several projects may need the use of clones, so we bought the parts and put one together. For those not aware of the term clone, it means an exact copy of the IBM PC computer but at half the price (typically $800). My first impression was more horror than amusment. The actual products were fairly straight forward with their design and construction. When the units go together however it amazes me that so many of them are being sold. My reasons for saying this is my past experience with designing computer packaging. I helped design a case for a single board computer and the main designer really knew his stuff. The unit went together quickly and was sevicable. The clones however neither go together easily nor are seviceable.

To be more specific, the disk drives (if you use full size) cover part of the main board (heat problems), making it necessary to install the main board before the drives. I had to remove the drive mounting bracket before I could easily get the main board in. What this means is that for servicing the unit it will be necessary to dismantle the complete unit just to check some of the circuits. The power supply has an internal fuse and fits up tight aginst the drive cables. The cables are the joke of the century with their cut and twisted layout, all to save having to change jumpers on the back of the drives. I'll proably have more to say about monitors as I tried every one in the house and found that none of them would work at the clone's sweep frequency. This trying included tweeking the horizontal frequency slug to it's maximum and only an ADDS monitor is able to come close (the right side rolls slightly).

My favorite complaint about documentation is also relevant for clones. IBM does have a considerable amount of information available for their PCs (at an extra cost) which will provide all the needed information. The clones documentation assumes you have the PC documents and also have a third grade english ability. About all they give you for information is how to stick it in and what jumpers are normal, heaven help you if your use is not standard or you want to know what parameters must be met to use it. The monitor problem was typical, the documentation did not mention anywhere anything at all about what type, frequency range, or interface needed for the monitor.

Anyway it is up and running and I have found one feature which is quite nice, mainly systems disks. We took several disks from other systems by different manufactures and were able to boot with each of them. Now for the CP/M world it was always possible to use programs that made standard calls on any system. Clones can go one better by using any boot disk from any clone. However the compatibility stops there, as some programs make hardware calls by going around the operating system, the end results being hardware compatible without installation programs. Other than some improvements in compatiblity my vedict is still not in on the overall system. I can say however that it apears at first glance that what was learned over the ten years of running CP/M has not all been transferred to MSDOS programmers. We had plannned on doing some comparisons on how fast programs could be brougth from scratch between a 68K and the clones, but a minor problem occurred.

We wanted to transfer some data between our Atari 520ST and the clone, but forgot to check one thing, AC grounding. The plant I work at has lots of people wiring things up and some of the electricians don't always get the ground connnected. Well the clone was not grounded and when we hooked it to the 520 sparks jumped across the conector melting the ground pin. The clone fared the best (which gets points) by only blowing out the 1488/89 drivers. The 520 however blew not only the drivers but also the 68901 mutli I/O chip and YM2149F (AY-3-8910) sound parallel driver. I think that is all as it still is not up and running yet. Having had this happen to a friend of mine with the same results, I would advise everyone to at least check the units before connecting them the first time. Gounding of units, or problem power supplies can make RS232 voltages much greater than the 12V they are designed for.

DEBUG I have had several units brought in for repair this week. One an ATR8000 with a bad CTC chip, another a Otrona with a bad memory device. The ATR would not run the disk drives properly, and changing the disk controller didn't help. While probing around I remembered how this unit is software timer driven, and so took a chance and tried the CTC. The unit would try and boot but without the timer there really wasn't any booting going on, althought I first thought so.

The Otrona is a very small clone portable, with a nice packaging layout. The first time you take one apart you may not think too much but after you see how everything fits, it is pretty nice. This unit was flashing 64K memory OK not the 256K it should have been. I tried their test programs and they all said memory was fine, which doesn't say much for their test utilities. I use a great memory tester for CP/M but haven't seen one for MSDOS yet, so I had to use the DEBUGger to find it. I dumped the second bank of memory (2000:0) and the 35th byte was 01, indicating the first bit stuck on. The initialization program writes AA to memory then clears to 00 and will stop at the first place that doesn't change properly. Changing this to physical memory required unsoldering the chip and putting in a socket and trying the new chip. My first attempt failed as I found the parity chip and not chip one. The second 128K bank was socketed already (an optional expansion) so it suddenly dawned on me that

(Continued on page 47)