ISSN 1 0748-9331

THE COMPUTER JOURNAL

For Those Who Interface, Build, and Apply Micros

Issue Humber 17

$2.50 US

THE COMPUTER JOURNAL
P.O. Box 1697
Kalispell, Montana 59903
406-257-9119

Edltor/Publlshar
Art Carlson

Art Director
Joan Thompson

Art Assistant
Lois Cawrse

Production Assistant
Judie Overbeek

Circulation
Donna Carlson

Contributing Editor
Ernie Brooner

Contributing Editor
Neil Bungard

Technical Editor
Lance Rose

The Computer Journal[9] is a monthly magazine for those who interface, build, and apply microcomputers.

The subscription rate for twelve issues is $24 in the U.S., $30 in Canada, and$48 airmail in other countries. Please make payment in U.S. funds.

To indicate a change of address, please send your old label and new address.

Postmaster: Send address changes to: The Computer Journal, P.O. Box 1697, Kalispell, Montana, 59903-1697.

Address all editorial, advertising and subscription inquiries to: The Computer Journal, P.O. Box 1697, Kalispell, MT 59901

Editor's Page

How Much Computer Do You Need?

I was recently asked "What computer should I buy?" and of course, my answer was "What do you want to do with it?" This lead to a long discussion and some time spent looking through the many pages of advertising in *Byte* and *The Computer Shopper.* We ended up overwhelmed by the number of choices available in today's market. When I bought my first computer the choice was between Commodore, Radio Shack, Apple, or CP/M, but today there are so many products on the market that it is no longer that simple.

When choosing a computer you should first select the software you need, and then get the system which runs that software. A survey of the current software showed that there were more new programs being released for the IBM-PC and its compatibles than for any other system. In fact, there are probably more releases for the IBM-PC than for all the other systems combined! This means that I'll have to recommend the PC or one of its clones for a non-technical user in a normal business office environment, based on the large number of available programs, user's support, and the general needs of an unsophisticated user.

Helping someone else select a system forced me to think about defining what a computer is, and how much computer is really needed. The usual reaction is to attempt to get one system that is powerful enough to fill all our needs, but the complexity and awkwardness of the system increases rapidly with size, and it can be very dificult to perform simple functions with a large system. There will never be the one "perfect computer" which satisfies all needs for everyone, because we each have different needs. In fact, I'll never be satisfied with just one computer because I have a wide range of applications. Besides, two smaller systems enable me to run two entirely different types of operations at the same time, and will probably cost less than one larger multitasking unit.

A better choice is to define the requirements on as low a level as possible, and then combine these requirements into similar groups in order to determine what type of system or systems are required. Some of us are computer nuts who would like to have one of everything to play with, but the limited funds available for computers force us to take a more realistic view of our needs.

> ***"There will never be the one 'perfect computer' which satisfies all needs for everyone..."***

My uses can be roughly divided into two areas, which are the business of running this magazine, and personal projects, with a lot of overlap since the magazine is about computers. The business applications include word-processing and phototypesetting from disk to produce the copy, a data base for maintaining subscription records and mailing lists, and a spreadsheet for financial forecasting. These needs can be served by either the original Apple II + I started with, or the two S-100 Z-80 systems running CP/M. I prefer the CP/M systems for the business because of the higher capacity 8 * disks, the software, and the operating system.

My personal projects involve general hardware and software hacking, learning additional languages and improving my programming skills, and applying computers for the measurement and control of real world devices. Most of my time has been spent on the magazine, so I haven't been able to do much with my personal projects, but I want to automate my lathe, build a remote weather station, monitor and control a solar heating system, and experiment with robotics. For this I need an open system with good I/O capabilites, an accessible bus, and a flexible operating system. Both the Apple and the CP/M systems work well here, and it is difficult to choose betwen them. The Apple has the advantage of having BASIC and a reasonably decent monitor in ROM, high resolution graphics, good low-cost assemblers, and reasonably priced cards for A/D and interfacing. The S-100 CP/M system has a better selection of languages, high capacity disks, a more powerful and flexible operating system, and better I/O capabilities, but interfacing cards are more expensive. I intend to continue working with both systems for program development, and then will use SBC's (such as Davidge) and microcontrollers (such as Basicon) for dedicated controllers.

What Can You Do With An Old Computer?

When you finally decide to get a newer, more powerful computer you are faced with the problem of deciding what to do with the old one. Because of the rapid advances, it isn't worth much on the used computer market if it is more than two or three years old, and yet it is still working and too good to throw away. One answer is to use it to relieve your main system from some low-level, time consuming operations, such as the print spooler described in Piotrowski's article on "Poor Man's Distributed Processing" in this issue.

I'm satisfied with my two eight bit systems for now because I still have a lot to learn, but I would like to upgrade to a 68000 16 bit system in the future —not because I need it, but just for the challenge *of* new things. One of the things that I really like about the S-100 system is that I can experiment with the 68000 by building the 68008 board described in Kohler's article in the last issue without replacing the whole system.

Right now I have absolutely no desire for an IBM-PC or one of its clones, but I think that their bringing out the PC was of great benefit to hardware hackers. Not that we'll buy their computers, but rather that all the non-technical users are flocking to the IBM-PC standard and dumping non-conforming equipment on the market at fire sale prices! It enables us to pick up great used equipment for very little cost (watch for Kibler's article on his $500 Superbrain in the next issue). You'll have to be able to help yourself when working with this older equipment because the manufacturer will either be out of business or will refuse to support the obsolete equipment. That's one of the purposes of this magazine —to help you learn to use an assembler and a debugger to patch the operating system, and to provide the means for you to contact others who have experience or documentation for the older systems. This is your magazine. . . use it!

A New Look For The Journal

This is our second issue with our new three column format. We made this change for easier readability, to improve the layout with larger illustrations and program listings, and to provide for l/9th page ads. In addition to the smaller ads, we are also adding classified ads in order to help individuals and smaller companies reach their markets and to make new developments in specialized fields available to our readers. The classified ads are 25 cents per word, paid in advance, and can be charged to your Visa or Master Charge, but we prefer not to take these ads over the phone because of the chance for errors.

Information Is For Sharing

The most important function of a journal is to provide a place for you to share your thoughts, ideas, problems, and solutions. We need your articles, letters, and comments. If you disagree with one of our authors, tell us. If you can expand on something we publish, tell us. If you need the answer to a problem, tell us. What you send doesn't have to be formal or fancy, just get us the information so that we can share it with others. ■

Letters From Our Readers

Dear Computer Journal:

I'm writing in response to the article "The State of the Industry," by Bill Kibler in Issue 15 of *The Computer JournaL* While Kibler has some good things to say, there are also some points I disagree with, primarily dealing with his adamant dislike for the IBM PC. While I realize the IBM PC has several shortcomings for us ideal-computer lovers, I also believe IBM has done much more good than harm to the microcomputer industry.

When IBM introduced their PC "...not compatible with anything" as Kibler puts it, I don't think it was quite the joke that Kibler seems to think it was. Although it is impossible to know all of the reasons why IBM chose the architecture that it did (many were economical, to be sure), there were many very good reasons for deviating from what was already available at the time. Among them is the limited memory space permitted by the 8-bit CPUs common to most of the systems of that time. Using a CPU with the ablility to directly address up to one megabyte of main memory allows the PC to run many programs and hold a lot of data that would be impractical or impossible on the typical 8-bit CP/M machines common at the time (and still ubiquitous today).

The interrupt-oriented architecture, DMA capability and standardized hardware expansion slots (that is, its open system architecture) are other positive features of the IBM PC. The expansion slots are one of the most attractive features of the Apple II, in my opinion.

Above all else, IBM did something for the microcomputer industry that needed to be done: they created a standard. The few standards previously established, in particular the Apple II and CP/M, were not sufficient to meet the needs of many businesses and other users. IBM created a standard with an 80 column screen (I never could get used to Apple's 40 columns!), a (reasonably) good keyboard that includes lower case and special function keys, and an open system architecture that allows easy system expansion. IBM also set a standard for the 10 M-byte Winchester drive, helping drop hard-disk prices.

Don't get me wrong. I'm not blind to the shortcomings of the IBM PC. Indeed, I dislike the segmented architecture of the 8086 family (including the PC's 8088). Fortunately, the segmentation problem is transparent to the user in most of the good application software available for the PC. I wish to this day, however, that IBM would have chosen the far-better 68000 family. My opinion concerning the IBM PC family of computers is reflected in an editorial statement by Phil Lemmons, Editor in Chief at *Byte magazine,* in their 1984 Guide to the IBM Personal Computer: "For the present, it makes more sense to enjoy the benefits of the current IBM standard than to curse it because it could be better. But enjoying the benefits of this standard shouldn't prevent us from keeping an eye open for something really new."

R.C.A.
Michigan

Dear Computer Journal:

Please find my check for a one year subscription enclosed. I would like to get a copy of the first two sections of your article "Write Your Own Threaded Language." Part three was in your sample copy and I enjoyed it very much. As an old hobbyist (circa 1972) I have become concerned that the hobby (computers) movement is being steamrollered by highly integrated technology on one side and suffocated by the tide of appliance computers on the other. Thus, I fully support your Journal. My interest currently is in the development of a 32 bit microprocessor based single board computer in the low cost style of the "Big Board" marketed by Digital Research of Texas. The board should have the capability of 4 megabytes of memory, floppy and hard disk peripherals, six to eight serial communication ports, and the same number of parallel ports. I feel that hobbyists need an architecture that is unique to their needs such as concurrency of tasks.

W.F.B.
Massachusetts

Dear Computer Journal:

Recently, I renewed my subscription to The *Computer Journal* Due to a limited budget, both of money and of time, I try to limit my reading to those magazines that cover the technical aspects of computers. By profession I am a programmer; by avocation I enjoy working with the hardware of computers and electronics.

Recently it has been obvious that the magazine industry has gone on a binge of producing computer magazines aimed at the user only, indeed at the novice user, virtually ignoring the avid hobbyist or interested techie. This tendency has even led to the demise of *MicroSystems,* which had been my favorite magazine, and *Microcomputing* which had been reasonably good until it was 'conglomeratized'. Fortunately this trend should be self-correcting, and the disappearance of many of these new user magazines is already taking place. But in the meantime some good magazines are also being lost.

Some people are 'fighting back' by correctly pointing out that the real audience for computer mag's is the sophisticated user, builder, designer, etc.. The nearest analogy is that while almost everyone drives a car there are virtually no magazines that feature articles such as "The Correct Grip on the Steering Wheel," yet there do exist car magazines aimed at the truly interested car enthusiasts, and they survive even while appealing to only a fraction of the car driving public. In fact they survive only by appealing to a limited audience.

I think *The Computer Journal* is a good, even needed magazine, and I want to see it survive. But I think it needs to find its niche. While reviewing the previous year's issues I am struck by the wide range of articles, going all the way from the most basic (Database Design, for instance), to the esoteric ("Wire Wrap a 68008 CPU"). I am also struck by the thinness of the issues; the whole year takes up only as much shelf space as three issues of *By te.* But thinness is relative — better to have a few good pages than a hundred meaningless ones, *(continued)*

One thing that I enjoyed in other magazines such as *Microsystems* was product reviews, especially reviews of products offered as kits. Reviews of kits are helpful to those of us who like to build them and even to the increasingly limited number of suppliers. I like to read kit reviews since I can't build all the kits that are offered (many of them I might not use), and want to know about the ones I would like to build. Reviews and articles about kit building can't but aid the industry, even when they include justified criticism of a particular kit.

Very truly yours,
J.O.
Massachusetts

Dear Computer Journal:

Thank you for a terrific magazine! Just as two other publications, *Microsystems* and *Microcomputing,* disappeared over the horizon, *The Computer Journal* came into view. *Microsystems* was terrific, and so was *Microcomputing's* predecessor, *Kilobaud.* I will miss them. I think *The Computer Journal* will do better than just fill the void.

There is a small group of dedicated microcomputerists in the Los Angeles area called "The Southern California Digital Group Computer Society." We are concerned almost exclusively with the preservation, maintenance, and further development of original Digital Group systems. We address both hardware and software issues. At a recent meeting I spoke about your magazine and almost everyone indicated an interest in subscribing.

Software ranges from operating systems —CO/M, OASIS, [PHIMON, DISKMON, MOPS (native Digital Group op. sys.)], MCOS, and OPUS; to languages-BASIC, FORTRAN, C, FORTH, Assemblers of all sorts; and applications that run the gamut from terminal emulators to accounting systems and data base tools.

Hardware typically is dedicated to the SUDING bus as originally presented by The Digital Group, although there have been many successful adaptations of S-100, Apple and TRS80 components. Recent refinements include 4mhz Z-80 CPU with on-board clock and calendar (with battery for continuous power-off function) and "heart-beat" for interrupt driven multi-user operation with intelligent co-processors provide sophisticated I/O management, terminal emulation, 512K pseudo-disk functions and much more.

Current projects include design and development of multiple concurrent processors (they may be dis similar) and the related software. The local group meets every two months. There is also a national newsletter. Anyone interested in Digital Group systems may contact me at the address below.

Sincerely,
Fred G. Sutton
Pres., SCDGCS
1230 S. Helberta Avenue
Redondo Beach, CA 90277

Dear Computer Journal:

I enjoyed your articles on "Controlling the Hayes Micromodem II From Assembly Language."

As a related question, I wonder what information is available on emulating block mode terminals. I realize that there are several different standards for block mode terminals, but there doesn't appear to be any block mode software available for the Apple II.

As a starting point I'd like to see some general information as to how the data and information codes are packed for transmission. Do you know where I could find some source material?

Sincerely,
F.K.
Los Angeles, CA

Ed: Readers, can you help?

Dear Neil Bungard:

This is to express appreciation for your trouble shooting/interfacing series in *The Computer Journal* I have a file folder at least % ' thick with references to trouble shooting techniques and circuits. It occurred to me that perhaps an annotated bibliography for *The Computerjournal* would be worthwhile. One particular device that I have wanted to build, but could never quite dope out from the printed material, was a test circuit described by Bob Cushman in *EDN* about five years ago, which originated with some Motorola engineers. It's essentially a method of looking at all data lines as latched at a given (thumb-switch selected) address. I have access to a fairly complete file of *Wireless World,* where a number of devices of varying complexity have been described.

I have looked, unsuccessfully, for a suitable circuit for a pulse injection probe with the versatility of the Hewlett Packard device, which senses whether a point is high or low, and pulses it in the appropriate direction. It is (as I recall) somewhat flexible in pulse duration. Any ideas?

Electronics (Australia) in December 1977, published full details on a 40 channel tester in which the condition of up to 40 points was latched and held, under control of a variable time-delayed strobe triggered from a reset (or other 'time zero') system reference. Thus the response of the 40 latched LEDs can be 'walked' through a total time excursion of several milliseconds as the time-delay controlling potentiometer is rotated and the timing sequence thus inferred. It looks as if that will be my next project.

Sincerely,
H.M.
Hinsdale, IL ■

POOR MAN'S DISTRIBUTED PROCESSING:

Cross Development and Using the H-8 as a Print Buffer

by Walt Piotrowski

I wonder how many computers there are in the world that still compute but have been taken out of service because their owners' needs have changed? There are the starter machines which were intended to be outgrown (although their purchasers may not have known that), and there are a growing number of very capable machines that have been replaced because of advances in technology. At the same time, there are a number of things around the house, or the company, that could be done quite well by a computer but are not being done because the newer models are too expensive or powerful to dedicate to these tasks. Energy control and security come to mind fairly quickly. An old computer can also be put to use as a smart peripheral or a data preprocessor for a newer machine. You could, for example, build real world interfaces for the old machine that might void the warranty on your newer machine and then transfer the data between the old one and the new one using a commercially available interface. You could also help advance the state of the art yourself by experimenting with loosely coupled distributed processing.

The old machine that you put to use in this way does not have to be a complete system. It is common to develop software on a fully equipped system and then use that software on another system that does not have a full complement of peripherals. It's regularly done in the commercial and military worlds (automobiles and missiles both have computers in them) and it can also be done by an individual if he's willing to substitute some ingenuity for expensive test setups. This article contains a general discussion of the principles of cross development and then shows their application in the development of a printer buffer using a Heathkit H-8 that had no peripherals of its own.

Cross Development

There are two major processes in software development that actually make use of a computer. The first is code generation. Code generation uses an editor for entry of source statements, an assembler or compiler for translation of source statements to an intermediate object code, and a linker or loader for generation of the final object code. The second process is code testing and, for the kind of program that we are considering here, usually requires additional debug aids of some kind.

The two computers involved in a cross development are called the host and the target. The host is sometimes called the development system and it usually has a disk operation system and a full complement of peripherals. Generally, all parts of the code generation process are done on the host. The target is normally a minimum system and doesn't have enough hardware to support an operation system. Although some tests must be done on the target, it's quite common to do at least part of the testing on the host with only a final test on the target. In addition to these two processes, which are done in any software development, cross development also involves an additional step of transporting the object code from the host to the target. Interestingly, I've heard people who do a great deal of this kind of work talk about normal programming as a cross development in which the host and the target are the same system.

Code generation is less expensive if the host and the target have the same microprocessor as their base machine. Both may be based on 8080s for example. In this case, you can use the host's normal compilers, assembler and loader to produce the object code. If the two systems are not based on the same processor, the extra expense comes from the need to buy (or write) a cross development tool like a cross assembler or a cross compiler. Cross assemblers are advertised regularly in most advanced computing magazines and are also available in the public domain. I haven't seen any cross compilers advertised, but they may be available if you make inquiries in the right places.

The strategy that you adopt for code testing is also influenced greatly by the base processors of the two systems. If they are the same, you test portions of the target's software on the host, using the host's normal debug tools and peripherals. If you are careful when you structure the program, you may be able to test a very large percentage of it on the host and leave only the portion that handles the target's I/O functions for test on the actual target system.

If the two processors are not the same, there are still several test options open to you. One option is the use of an instruction level simulator (ILS) to simulate execution of the target's instruction set on the host. Instruction level simulators for the simpler processors like the 6502 or 8080 are relatively easy to write, and many people write them in high order languages. Once you have an ILS, you can use it to do the same kind of testing on the host that you would do if the two base processors were the same. (As of this writing, I have not seen any instruction level simulators available commercially or in the public domain.) Another test option, if you are writing part of the target program in a high level language, is the use of two separate compilers. One of these is the cross compiler that you will use to produce the code for the target machine. The other is a compiler to produce code that you will test on the host. If both compilers are of good quality, you can be confident that once the high level language portion of your program works on the host, it will run correctly on the target.

If the target is really a "minimum"

system, testing on the target will probably be at the machine language level. In the professional cross development world, there are exotic test tools (like in-circuit emulators) that allow you to use the power of the host while testing the target, but these require more hardware than you or I will probably ever have. In our environment, test aids on the target system will be sparse. The tools that are available and the complexity of the program that you are testing will influence the amount of work that you will be able to leave for the target machine. Testing with no tools at all might be possible but it would require either that your program be extremely simple or that you possessed an incredible amount of intuitive reasoning capability (or luck). A control panel is the lowest level test tool and the step beyond that, if you are lucky, is a debug program that does not require an operating system for I/O support. A debug program, however, would require that you had a terminal available to run it. The final problem is the transmission of object code from the host to the target. There are several approaches. Writing a diskette or a cassette tape on the host and reading it on the target is certainly the simplest, but is probably the least likely since it requires that the systems have compatible peripherals. For our minimum target system, a more likely solution is a communication link. For most of us who are using old systems, the available link will be RS-232. The protocol for the communication that you do over the link depends a great deal on the intelligence level of the target system when its power is first turned on. In the best case, the target has a ROM that will boot from the link. (In the good old days, we used teletypes and our mass storage was paper tape, so this isn't as far fetched as it sounds.) The next best case, if the target machine has a control panel, is to use the panel to enter a small boot routine by hand. If you choose this as an option, you may want to consider a two stage download. The first stage can be a very unsophisticated program that will only download a more sophisticated loader. This exotic loader can then download the actual software while doing error checking on the transmission. As a bonus, if you do your download via RS-

```
                    ****************************************************
                    i
                    9       H-8 Loader
                    i
                    <       Walt Piotrowski
                    i       State University of NY
                    9       Binshamton, NY 13901
                    i
                    1****************************************=*=a=*=***
                    ;
0005 =              BDOS     EQU     5
0023 =              FSIZE    EQU     35            ;File Size Code
000F =              OPEN     EQU     15            ;File Oren
0010 =              CLOSE    EQU     16            ;File Close
0014 =              RDSEQ    EQU     20            ;Read Seauential
001A =              SETDMA   EQU     26            ;Set disk address
                    7
005C =              FCB      EQU     3CH           ;File Control Block
007D =              FCBSIZ   EQU     7DH           9FCB Size Field
                    ;
0016 =              SYN      EQU     16H           ;ASCII Sune
0002 *              SIX      EQU     2             ;ASCII STX
0020 *              BLANK    EQU     20H           9ASCII Blank
                    9
0000 =              SECSIZ   EQU     128           ;Disk sector size
                    9
0800 =              BUFST    EQU     800H          ;Input buffer
                    9
                    9
0100                         ORG     100H
                    ;
0100 210000         H8LDR    LXI     HrO           ;Clear HL
0103 39                      DAD     SP            ;Make a CORM of SP
0104 225402                  SHLD    STACK         ;Save for exit
0107 315402                  LXI     SPaSTACK      ;Get local stack
                    ;
010A 345000                  LDA     FCB+1         ;Look at file nane
010D FE20                    CPI     BLANK         ;Not supplied?
010F CADB01                  JZ      EREXIT        ;Error - no file nase
                    9
0112 OEOF                    NVI     CaOPEN        ;Open file code
0114 115000                  LXI     D.FCB         IFCB Address
0117 CD0500                  CALL    BDOS          ;Open it
011A 3C                      INR     A             ;Error code is 255
011B CADB01                  JZ      EREXIT        ;Error - no file on disk
                    9
011E 0E23                    NVI     CaFSIZE       ;File size comeand
0120 115C00                  LXI     DaFCB         9FCB Address
0123 CD0500                  CALL    BDOS          ;Get size computed
0126 3A7000                  LDA     FCBSIZ        ;Get size LSBs
0129 320802                  STA     FILSIZ+1      9H8 swaps thee
012C 3A7E00                  LDA     FCBSIZ+1      ;Get NSBs
012F 320A02                  STA     FILSIZ        I Swap these too
                    9
0132 0607           HUL128   NV 1    B,7           9LOOR Ctr
0134 AF             MULLP    XRA     A             ;Clear Carry
0135 3A0B02                  LDA     FILSIZ+1      ;Get LSBs
0138 17                      RAL                   aNult by 2
0139 320B02                  STA     FILSIZ+1      ;Put back
013C 3A0A02                  LDA     FILSIZ        ;Get NSBs
013F 17                      RAL                   ;Mult bw 2
0140 320A02                  STA     FILSIZ        ;Put Back
0143 05                      DCR     B             ;Decrement LOOR Ctr
0144 C23401                  JNZ     MULLP         ;Not done wet
0147 3AOA02                  LDA     FILSIZ        ;Get LSBs
014A 321102                  STA     FILCTR+1      ;Save for counting
014D 3AOBO2                  LDA     FILSIZ+1      ;Get NSBs
0150 321002                  STA     FILCTR        ;Counter - noreal order
                    ;
                    9        LOOR to read file into memory
                    ;
0153 210008                  LXI     HaBUFST       Input buffer start address
0156 225602                  SHLD    BUFAD         ;Save for use
0159 E5                      PUSH    H             ICoPy on stack
015A D1             READLP   POP     D             ;Easu wav to transfer
015B OE1A                    NVI     CaSETDNA      ;Disk address set
015D CD0500                  CALL    BDOS          ;Set to local buf
0160 115000                  LXI     DaFCB         9FCB Address
0163 OE14                    NVI     CaRDSEQ       ;Read Seouential
0165 CD0500                  CALL    BDOS          ;Read next record
0168 C600                    ADI     0             ;Set flags
016A C27B01                  JNZ     CLOSIT        ;Read finished
016D 2A5602                  LHLD    BUFAD         ;Get buffer address
0170 116000                  LXI     DaSECSIZ      ;Get sector size
0173 19                      DAD     D             ;Point to next block
0174 225602                  SHLD    BUFAD         I Put back
0177 £5                      PUSH    H             ;Cory on stack
0178 C35A01                  JNP     READLP        ;Read next
                    9
017B 115000         CLOSIT   LXI     DaFCB         9FCB Address
017E 0E10                    NVI     CaCLOSE       ;File close code
0180 CD0500                  CALL    BDOS          ;Close it
                    9
                    9
                    9        Write to RS-232
                    9
                    9
0183 CD5802                  CALL    OPNDEN        ;Open RS-232
                    ;
```

232, you may be able to run a dubug program in the target by using your host, with a modem program, as a dumb terminal.

Once you have your software up and running properly, you may want to consider streamlining the whole process by eliminating the download from the host. The standard approach is to burn the software into a PROM. A good possibility, if your machine has a boot-up ROM, is to replace the existing ROM with your own. This also opens up the possibility of using your old machine as a stand-alone process controller in remote locations.

An Ideal Target Machine

My first home computer was a Heath H-8, which I bought shortly after they were announced. My initial investment in the computer, a terminal, two cassette recorders and a printer destroyed the family budget for a couple of years. When disk drives became available, I was still making the monthly payments on the equipment that I had already bought. For a lot of reasons, I wasn't active in home computing for a few years after the system was paid for and, when I finally got back to it, it was cheaper to buy a whole new system than to buy disk drives and more memory to upgrade my H-8.

My new system has a lot of bottlenecks, but the most annoying is the 30 character per second printer. (The printer, by the way, is the same DecWriter that I bought with my first system.) One night I was reading an article about a printer buffer and I thought about my unused H-8. It has the two serial ports that are required by my hardware. One had been used for the console terminal and the other was used for the DecWriter. Even though my H-8 only has 16K of memory, some quick arithmetic showed that it would hold about nine minutes worth of printing at 30 cps. Not terrific, but not bad either.

It turns out that the H-8 is almost ideally suited to be a target machine. It is based on the 8080 which means that any system that runs CPM-80 can be used as the development system. It was designed before the days of systems with integral keyboards and monitors, which means that it needed to have an RS-232 port to handle an external terminal. Its most important feature,

```
0186 0614               MVI    Br20        ;Write 20 characters
0188 3E16      SYNLP    MVI    A.SYN       ;Get sune character
018A CD5902             CALL   URMDEM      ;Send to RS-232
010D 05                 DCR    B           ;Loop counter down
018E C28801             JNZ    SYNLP       iDo lore
0191 3E02               MVI    A»STX       ;Get STX
0193 CD5902             CALL   URMDEM      ;Write it
               i
0196 210000             LXI    H.O         ;Clear HL
0199 221202             SHLD   CRCSUM      ;Clear CRC
019C 0608               MVI    B,HDRSIZ    ;Get header size
019E 210802             LXI    HrRECHDR    ;Record header address
01A1 7E        HDRLP    MOV    ArM         ;Get character
01A2 CDE001             CALL   CRC         ;Update check sum
01A5 CD5902             CALL   URMDEM      ; Send to RS-232
01A8 23                 INX    H           ;Point to next char
01A9 05                 DCR    B           ; LOOP ctr down
01AA C2A101             JNZ    HDRLP       ;Send more
               i
01AD 2A1002             LHLD   FILCTR      ;Get number of otes
01B0 EB                 XCHG               ;Put filsiz in DE
01B1 210008             LXI    HrBUFST     ;Buffer Address
0184 7E        DATLP    MOV    ArM         ;Data byte
01B5 CDE001             CALL   CRC         ;Update CRC
01B8 CD5902             CALL   URMDEM      ;Output the char
01BB 23                 INX    H           ;Point to next char
01BC IB                 DCX    D           ;LOOP ctr down
01BD 7A                 MOV    ArD         ;Get LSBs
01BE B7                 ORA    A           ; Set flass
01BF C2B401             JNZ    DATLP       ;Not done
01C2 7B                 MOV    AsE         ;Get MSBs
01C3 B7                 ORA    A           ;Set flass
01C4 C2B401             JNZ    DATLP       iMore chars to send
               $
01C7 2A1202             LHLD   CRCSUM      ; Get CRC
01CA 7C                 MOV    A»H         ;Get MSBs
01CB CD5902             CALL   URMDEM      ;Write it
01CE CDE001             CALL   CRC         ; Put it in too
01D1 7D                 MOV    ArL         ;CRC LSB
01D2 CD5902             CALL   URMDEM      ;Send it
01D5 CDE001             CALL   CRC         ;See if it checks
               ;
01D8 CD5A02             CALL   CLMDEM      ;Close RS-232
               ;
01DB 2A5402    EREXIT   LHLD   STACK       ;Get CPM stack ptr
OIDE F9                 SPHL               ; Back into SP
01DF C9                 RET
               ;
               ;
               ;        CRC - Compute CRC-16
               $
               ;               (X+1 )*(X15 + X + 1 )
               ;
               i               (Duplicate of the H-8 PAM CRC)
               $
               ;
01E0 C5        CRC      PUSH   B           ;Save (BC)
01E1 V0 VO              MV I   BID         ;(B)=Bit count
01E3 E5                 PUSH   H
01E4 2A1202             LHLD   CRCSUM
01E7 07        CRCI     RLC
01E8 4F                 MOV    Ct A        ; (O-Bit
01E9 7D                 MOV    A»L
01EA 87                 ADD    A
01EB 6F                 MOV    LiA
OlEC 7C                 MOV    ArH
OlED 17                 RAL
01EE 67                 MOV    HrA
01EF 17                 RAL
01F0 A9                 XRA    C
01F1 OF                 RRC
01F2 D2FD01             JNC    CRC2        Hf not to XOR
01F5 7C                 MOV    A»H
01F6 EE80               XRI    2000
01F8 67                 MOV    H»A
01F9 7D                 MOV    ArL
01FA EE05               XRI    5Q
OiFC 6F                 MOV    Lr A
OlFD 79        CRC2     MOV    A*C
01FE 05                 DCR    B
01FF C2E701             JNZ    CRCI        ; If more to so
0202 221202             SHLD   CRCSUM
0205 El                 POP    H           ;Restore (HL)
0206 Cl                 POP    B           ;Restore (BC)
0207 C9                 RET                ;Exit
               1
               1        Data
               1
0208 81        RECHDR   DB     81H         ;Tupe (aes insse) 1 EOF
0209 01                 DB     1           ;Record 91
020A 0000      FILSIZ   DB     0,0         ;Number of butes
020C 2040      H8NTRY   DB     20H,40H     ; H8 Entry (040 100)
020E 2040      H8LOAD   DB     20H,40H     ; H8 Load Address
0008 -         HDRSXZ   EQU    t-RECHDR    ;Header length
               1
0210 0000      FILCTR   DU     0           ;Output counter
0212 0000      CRCSUM   DW     0           ; CRC
               ;
```

however, is that it contains a built-in solution to the download and debug problems.

The H-8 ROM contains a program that Heath called the Panel Monitor (PAM). The system boot routine is a part of PAM and the earlier H-8s normally booted from cassette recorders through the Serial I/O and Cassette Board (H8-5). In the early days, Heath was trying to sell systems to people who already had teletypes and paper tape readers and, to accomodate them, they provided a port interchange switch on the H8-5 board. When you flip the switch, the board exchanges the addresses of the console port and the cassette port. The ROM, thinking it is still talking to a cassette, is actually handling the RS-232 line. Booting from an inter-computer link requires pushing just one button (as long as the host machine transmits the file using the protocol that PAM expects). PAM also contains a complete machine language debugger which takes commands from the front panel keys and displays results on the front panel LEDs. (The system would be perfect except for one small frustration: the panel monitor displays everything in octal, and CP/M's assembler prints everything in hex.)

An assembly language program that will download from a CP/M system to an H-8 is provided with this article. The program takes the name of the file to be downloaded from the command line and expects to find a file by that name in COM format on the disk. It assumes that the program will load at the normal H-8 start address of 2040H (040 100 in H-8 split octal). Getting the H-8 program into COM format after assembling it at 2040H requires some manipulations. These are given in a note at the end of the article.

My CP/M system is a Commodore 64. In the C-64, handling the RS-232 port from CP/M requires code for both the Z-80 CP/M co-processor and the native 6510. Since this code is lengthy and is of interest only to C-64 CP/M users, I have not included it in the listing. Instead, you will find a commented section at the end of the listing that shows where you should insert code to handle your host machine's RS-232 port. The comments also identify what the main program expects the subprograms to do. If you are a C-64 owner, contact me

```
0214                    DS 64                      ;Local Stack
0254 0000    STACK      DW      0                  ;CPM Stack save
             ;
0256 0000    BUFAD      DW      0
             ;---------------------------------------------------
             ;
             ;
             ;       RS-232 Interface Routines
             ;
             ;---------------------------------------------------
             ;
             ;       Insert your RS-232 code here. Your routines
             ;       should restore all registers to their original
             ;       values before returning.
             ;
0258 C9      OPMDEM   RET                          ;Insert your RS-232 port setup
                                                   ;here.
             ;
025? c9      WRMDEM   RET                          ;Insert the code to output a
                                                   ;a character to your RS-232
                                                   ;here. Your routine should
                                                   ;include a status check.
             ;
025A C9      CLMDEM   RET                          ;Insert the code required to
                                                   ;shut off your RS-232 port
                                                   ;before returning to CP/M here.
             ;
             ;
025R                  END
```

```
/****************************************************************/
/*                                                              */
/*                                                              */
/*                  Print buffer program for                    */
/*                        Heath H-8                             */
/*                                                              */
/*                  Walt Piotrowski                             */
/*                  State University of NY                      */
/*                  Binghamton, NY                              */
/*                                                              */
/*                                                              */
/****************************************************************/


#include cprtbuf.cl
#include sc80.cc

#define bufsiz 15000
#define true 1
#define false 0

MAIN()
  {
  char circbuf [bufsiz];            /* Circular buffer */
  int inptr,otptr;                  /* Buffer pointers */
  int inpoff;                       /* Input (RTS) off flag */
  char recchar;                     /* Received character */

  /* Initialize                 */
  inset ();                         /* Set up input USART */
  otset ();                         /* Set up output USART */
  inpoff = false;                   /* Input is on */
  inptr = 0;
  otptr = 0;

  /* Main Loop                  */
  while (true)
    {

     /* Character Input */
     if (rdwin ())
       {recchar = chrin();
       if (recchar != 0)
          {circbuf [inptr++] = recchar;
       if ( inptr>=bufsiz) inptr=0;} }
       buffull(inptr,otptr,&inpoff);
       }

     /* Character Output */
     if ( inptr != otptr && rdwot()==true)
       {chrot (circbuf [otptr++]);
       if(otptr>=bufsiz) (otptr=0);}
```

and we can make arrangements for giving you a copy of the entire program.

An H-8 Printer Buffer

My main machine, which will transmit to the printer buffer, uses what's been called an x-line protocol for RS-232 transmission. It responds to Data Terminal Ready (DTR) and Request to Send (RTS) on the RS-232 line. Normally, the DTR signal from the receiving device is controlled by hardware and is asserted whenever the power is on. The RTS line is manipulated dynamically by the receiver to control the data flow. The H-8 program, described later, uses this RTS line to shut off the data flow when the buffer memory is full.

Slight hardware mods were needed to set DTR and RTS signals from the H8-5 board. All of the H-8's serial devices used a 3 line RS-232 interface, which does not provide control functions between the receiver and the sender. For some unknown reason, a great deal, but not all, of what was required to provide DTR and RTS signals was already on the board. The hardware mods provided at the end of the article will make sense if you have an H8-5 logic diagram in front of you. In essence, they do the following:

1) Provide pullups for the collectors of the transistors that provide the DTR and RTS signals.

2) Reverse the sense of the DTR signal so that it goes high when power is on.

3) Provide a cable and back panel connector to get the additional RS-232 signals from the H8-5 board onto an RS-232 cable.

Listings for the H-8 print buffer program are provided along with this article. The program is in two parts. The control portion, written in Small C, contains an infinite loop which polls the input line for data and also polls the output line to see if it is ready to transmit another character. Since the input rate is higher than the output rate, the excess characters go into a circular buffer. When the circular buffer is dangerously close to full, the program shuts off the input by dropping the RTS (Request to Send) signal on the input line. It turns the input back on again when there is more room in the buffer. The actual I/O to handle the H-8's USARTs is done with assembly

```
        buffull(inptrotptr,zinpoff),
        }
    }
  }
/* Buffer full check */
buffull(inptr,otptr,pinpofr)
   int i npt r»otpt r , *pinpof f;

   <
   int slotslfti                          /* Number of slots left */

   slotsift = otptr-inptr;
   if (slotsift <= 0)
       (slotslft = slotsift + bufsiz;
       >
   if ((slotslft < 20) & (pinpofr == false))
       {*pinpoff = true ;
       trnof();
       }
   if ((slotslft > 20) 1 (pinpofr == true))
       {*pinpoff = false ;
       trnon()*
       )
   >
♦ asm
;*****************************************************
;
i             H-8 Print Buffer Subroutines
;                Walt Piotrowski
;
;
;*****************************************************
;
;
;       Input  USART Eauates
;
INMOD    EQU     116Q            ;Mode Inst
INCMON   EQU     0640            ;Cnd RTS on
INCMOFF  EQU     024Q            iCnd RTS off
INCTL    EQU     371Q            ;Control Port
INDATA   EQU     370Q            iData Port
INSTAT   EQU     371Q            ;Status Port
;
;       Output USART Eauates
;
OUTMOD   EQU     116Q            5 Mode Inst
OUTCMD   EQU     1               $ Command Inst
OUTCTL   EQU     3770            ;Control Port
OUTDATA  EQU     376Q            ;Data Port
OUTSTAT  EQU     3770            ;Status Port
;
TXREDY   EQU     1               ;TX Reads Status Bit
RXREDY   EQU     2               ;RX Ready Status Bit
;
TRUE     EQU     1
FALSE    EQU     0
;
;------
;
?       Shut  input off
;
QZTRNOF  MUI     A,INCMOFF       ;Get off command
         OUT     INCTL           ;Send to USART
         RET
;
;------
;
5       Turn  input on
;
QZTRNON  MVI     A*INCHON        ;Get on command
         OUT     INCTL           ;Send to USART
         RET
```

language subprograms. These are given in the third listing.

By using Small C, it was possible to check out the control portion on the host by INCLUDEing a test library in place of the actual 1/0 routine library. Since I had never written a program in C before, this was an important consideration for me. When the program ran satisfactorily on the host, I transmitted it to the target with only the I/O left to be checked.

H8-5 Hardware Mods

Board Changes:

1) Cut the solder trace from IC122 pin 1 to IC 124 pin 23.

2) Cut the solder trace from IC122 pin 3 toR154.

3) Cut the solder trace from IC122 pin 6 to R155.

4) Connect IC122 pin 6 to IC122 pin 1.

5) Connect IC122 pin 3 to R155 (same end as step 3).

6) Connect IC124 pin 23 to IC117 pins 12 and 13.

7) Connect IC117 pin 11 to R154 (same end as step 2).

8) Connect a 2200 ohm 1/2 watt resistor between P102 pin 9 and P102 pin 1.

9) Connect a 2200 ohm 1/2 watt resistor between P102 pin 9 and P102 pin 2.

Cables:

The following signals at P102 on the H8-5 board need to be brought out to the back panel and from there to the RS-232 cable. The RS-232 connections shown assume that your host computer is wired as Data Terminal Equipment (DTE).

Signal	P102	Ra-232
RTS	Pin 1	Pin 5
DTR	Pin 2	Pin 20
GND	Pin 4	Pinl
Data In	Pin 5	Pin 2
Data Out	Pin 8	Pin 3

Connectors:

The following connectors are those used by Heath:

S102-

Molex 22-01-2105

G C Electronics 41-130

Back Panel Connectors:

Molex 03-06-2151 (plug)

Molex 03-06-1151 (socket)

Sold as a package by Waldon 1625-15 PRT

Miscellaneous Software Procedures

To make a COM file from a HEX file that has been ORGed at 2040H use DDT with the following commands:

iFN.HEX

rEOCO

If you are using Small C 1.1, which generates a file for ASM, you can make a HEX file for your target machine by modifying the first few lines of the ASM file produced by the compiler. In the ASM file, change the ORG to the address appropriate for your machine (2040H in this case) and change the stack pointer setup to point to the top of your target's memory.

```
;
i        See if input reads
;
QZRDYIN  IN      INSTAT          ;Get status
         ANI     RXREDY          ;Mask
         JZ      RDYINI          10 = not reads
         LXI     HrTRUE          ;Char is ready
         RET
 BtCPRTBUF.CL

RDYINI   LXI      H,FALSE        INo char
         RET
;
y------
i
i        See if output reads
$
OZRDYOT  IN       OUTSTAT        (Get status
         ANI      TXREDY         I Mask
         JZ       RDYOTI         10 = not ready
         LXI      Hr TRUE        ;Ready for output
         RET
RDYOT1   LXI      H»FALSE        ;Not ready
         RET
;
A
;------
$
1        Set UP input USART
1
QZINSET  MVI      A»INCHON       ;Command
         OUT      INCTL          ;Out to control port
         RET
i
;------
;
i        Set UP output USART
$
QZOTSET  MVI      ArOUTMOD       ;Hode
         OUT      OUTCTL         ;Out to control port
         MVI      ArO            ISetup for wait loop
OUTDLY   INR      A              iIncrement
         JNZ      OUTDLY         ;Wait for USART
         MVI      ArOUTCMD       I Command
         OUT      OUTCTL         »Out to control port
         RET
;
;------
;
i        Input a character
$
QZCHRIN  IN       INDATA         IRead it
         MOV      L»A            ILow order of param
         MVI      H»O            ;Hi order
         RET
;
$------------
```

Listing continued on page 14

BASE

A Series on How To Design and Write Your Own Database

By E.G. Brooner

We come now to the theoretical means that can be used to 'find' some particular bit of information or some related set of data items. Placing information into some particular order and finding it again involves techniques generally summarized *assorting and searching.* We are assuming that the information has been originally stored in some random, un-ordered manner.

Sorting, of course consists of placing the data in some kind of ascending or descending order, alphabetically or numerically. There is actually little difference between the two in computer applications because sorting is based on the ASCII value of the characters —the ASCII value of the numeral '9' is larger than the value of the numeral '8,' and the value of 'B' is greater than 'A.'

As we will probably enter data more or less haphazardly, as it comes to us, this sorting has to be done by the program after the data has been entered. It may have to be done again from time to time as more data is added. There are several interesting programming techniques used to accomplish this end.

Searching is also a diversified concept, and the means used depends on how the information is ordered and stored in the files. One technique we'll discuss is indexing, which is used almost exactly as the index is used in a book or catalog. Another is the apparently magical means of using the data itself as a clue to its location on the disk; this is known as 'hashing.' Binary searching is another method ideally suited to computer use, since it is based on the kind of logic computers use.

The sequential search. Assume a list of names or numbers, which may or may not be in any particular order; assume then that you wish to locate one particular item. Your only choice is to start at the beginning and check each item until you find the correct one. This is O.K. for a single printed page or for a data file of a few dozen items, but it can be time-consuming if the list is long. Many file programs use the sequential access method; it is simple and for some purposes is perfectly adequate. In some cases it is mandatory — in a database it is often necessary to find several entries that meet the same criteria, which means that the entire file has to be read to make sure none are missed.

Sorted order. Next, consider that the list of names has been sorted into alphabetical order (as in a phone directory or dictionary). We open the list and look at an entry; if we are looking for 'Jones' and the list falls open to 'Conrad' we know to look beyond that point. If it falls open to 'Smith' we flip back toward the beginning. Repeating this process narrows the search until we find the entry for which we are looking. This is the basis for the binary search we will discuss later.

Direct addressing. Now consider a similar list that is numbered in sequence. If we know that the item we want is entry number 876 we can go directly to it. In effect this is what we frequently do with a data file, if and when we know which relatively numbered record it is that we want. If we are using a so-called

```
REM      ********************83
REM      ISINGLE-SEARCH ROUTINES
REM      ssssssssssssssssssssss
5000     GOSUB 9999
REM      DEFINE THE FILE STRUCTURE
          FILES="B"+NAMES+".EXT"
          OPEN FILES AS 16
          READ #16;EXT%                              REM HOW HANY RECORDS
          CLOSE 16                                   REH IN THIS FILE?
          FILES="B"+NAME$+".DEF"
          NBR.OF.FLDS%=0
          OPEN FILES AS 16
          FOR X%=1 TO 12
                    IF END #16 THEN 5050
                    READ 416;FLD.NAHES,FLX(XX)
                    NBR. OF. FLDSX-NBR. OF. FLDS%+ 1
                    PRINT XX; TAB <5) ; FLD. NAMES      REH LIST THE NAMES
                    FIELD.NAMES(XX)-FLD.NAMES            REM OF THE FIELDS
          NEXT XX
5050 CLOSE 16

5100 PRINT "TO TERMINATE SEARCH, ENTER • ";NBR.OF.FLDS%+1 >PRINT
          INPUT "SEARCH ON FIELD NUMBER ";FLDX
REM ANY FIELD CAN BE USED AS THE 'KEY' FOR SEARCHING
          IF FLD%<1 THEN 5100
          IF FLDX> NBR.OF.FLDSX THEN 1300
          INPUT "SEARCH-KEY (ANY 4 LEFTMOST CHAR OF FIELD ";KEYS
          K.LX«LEN(KEYS>
REM      YOU CAN USE PART OF THE FIELD AS A KEY
REM      AND FIND AN EXACT MATCH, OR ALL GREATER OR LESS THAN KEY
          PRINT "RELATION OF RECORD TO KEY: 1-EQUAL TO"
          PRINT TAB(2B);"2—GREATER THAN"
          PRINT TAB(2B);"3-LESS THAN"
          INPUT RELX
          IF RELX<1 OR RELX>3 THEN 5100
REM CREATE FILE NAME, OPEN IT, AND START LOOKING
          FILES="B" +NAME$+STRS (FLDX) ♦".DAT"
          OPEN FILES RECL FLX(FLDX)+5 AS 16
          IF END •16 THEN 5200
          FOR NX—1 TO EXTX
                    READ a 16;DATUMS
                    DATUMS-LEFTS(DATUMS,K.LX)
                    ON RELX GOTO 5110,5120,5130
REM COMPARE RELEVANT PART OF FIELD W/CHOSEN RELATIONSHIP

5110                IF DATUMS-KEYS THEN 5140
                    GOTO 5150
5120                IF DATUMS>KEYS THEN 5140
```

'random access' (or 'relative') file, the operating software keeps track of where each record is located and we simply ask for the record by number.

Keyed access. Sorted order and direct addressing can be combined in a very useful way. If the records are numbered we can first sort the 'key' (names, in this example) and rearrange the record numbers in accordance with the alphabetical order of the names. Doing this results in a 'mixed-up' list of record numbers. Now if we read the records in the 'mixed-up' order we will find that the resulting list of names will come out in the sorted order. This will be illustrated when we get to the portion of BASE that does the actual sorting.

This kind of arrangement has a particular advantage for computer use. After sorting the record numbers as described, we store them as a separate list. This list is then known as a 'key file' or index file. The advantage is that the original list of names has not been altered in any way from its random order. But by referring to the key file we can go directly to the information as if it were in alphabetical order.

Binary search. The technique just described does not, by itself, solve all problems. We still might need a quick way of leafing through the key file to find out which record number corresponds to the name 'Jones.' The binary search is one way to do so. A key file that is to be used in this way has to contain the key fields in their sorted order, along with their record numbers in whatever order they happen to be. The binary search process then looks at the key fields and uses the associated record number to find the complete record.

The binary search only needs to know the length of the file, or list, and whether it is in ascending or descending order. It reads the key in the center, and learns whether the desired record is higher or lower in the order of things. It then examines the center of either the upper or lower half, as the case may be, and gets that much closer. About half a dozen 'looks' will find almost any entry in a list of a thousand or so items. Doubling the list's length only adds one more 'look,' and so on. The binary search is blindingly fast when using an in-memory array; it is quite impressive even when reading from a disk file. On the average, a binary search will find the desired record in 4 tries for a list of 25,6 tries for 100, and 9 or 10 tries for a 1000 record file. 2000 records needs 11 tries, 5000 about 12 or 13, and 10000 only one more. Even searches of this magnitude, reading the records from a disk, take only a few seconds.

```
                    GOTO 5150
5130                IF DATUMSKEYS THEN 5140
                    GOTO 5150
REM                WHEN KEY FOUND, GO READ ENTIRE RECORD

5140               FOUND%=N%          REM KEY MATCHES THIS RECORD
                    PRINT "RECORD NUMBER ";FOUND%:PRINT
                    GOSUB 9000 REM READ THE WHOLE RECCRD
                    IF CONTINUES="M" THEN 5200
5150      NEXT N7.
5200      CLOSE 16
           GOTO 5000

REM       SEARCH HAS ENDED
6000      CHAIN "FILESORT.COM"                 REM IF    OPTION CHOSEN

7000      CLOSE 17,10:GOTO 1000 REM & START          OVER
8000      CHAIN "PRTFORM.COM"                  REM IF    OPTION CHOSEN
REM THE '.COM' EXTENSION IS USED ONLY IN THE CB-80 VERSION

REM       ************************
REM       «FIND AND READ FILE 'N'»
REM       ************************
REM       SUBROUTINE CALLED BY SEARCH SECTION WHEN KEY FOUND

9000      FOR X7.= I TO NBR. OF. FLDS7.
                    FILES="B"+NAMES+STR»(X %) + ".DAT"
                    OPEN FILES RECL FL% (X%) +5 AS x%
                    READ #X%,FOUND%;DATAS(X%)
                    PRINT FIELD.NAME*<X7.) .DATAS(X%)
           NEXT X7.
           FOR X%=1 TO NBR.0F.FLDS7.
                    CLOSE XK
           NEXT X%
           PRINT "TYPE <CR> TO CONTINUE SEARCH OR <M> FOR MENU"
REM WHEN THROUGH VIEWING THE DATA, PRESS RETURN TO CONTINUE
REM SEARCH AND DISPLAY, OR 'M' TO END SEARCH
           INPUT LINE CONTINUE*
           RETURN


9999 PRINT CHR»(26)                    REM CLEAR SCREEN
           RETURN                      REM CHANGE FOR YOUR TERMINAL
REM THE DATA STATEMENTS NECESSARY ONLY IN CB-80 VERSION
REM WHICH HAS TO RESERVE DATA AREA FOR CHAINED PROGRAMS.
           DATA "A","B","C","D","E","F","G","H","J",“K”,“L","M"
           END
```

In our database examples we will probably have to provide for more than one kind of search. We might, for example, sort the records for some kinas of access, and 'find' by relative adress for others; at other times we might read the entire tile sequentially and check every entry. It's obvious, then, that we will want to provide for more than one way of reading any particular file or set of files. This will be explored more fully when we come to the sections of the program that actually handle these chores.

At this writing we have not added any of the more exotic methods of sorting and searching to the main BASE program, but they are worth describing and considering in the general context of database programming. As a matter of actual fact, the main body of BASE uses a simple sequential search, the options being only to match a key, or find those either larger or smaller. For the latter two conditions a sequential search is a necessity anyway.

The simple sort program that will be shown in another column builds key files consisting only of the record addresses; this permits a file to be printed in ascending order based on any field. Two other programs are in existence that operate on BASE's files. One of these (called MATCH) allows us to match two fields, such as first name and telephone area code —i.e: Joe who lives in Seattle, area code 206.) The other (called BINARY) sorts selected files, as does the main program, but stores the keys in such a way as to provide a binary search which is part of the same program. If time and space permit, these auxiliary programs will be published at a later date. Keep in mind, though, that the portions of BASE shown to date, even without any sorting, can be very useful for modest-size databases.

In BASE we have kept each field (of any set of records) as a separate file —this makes sequential searching (for a single field) quick and easy and simplifies using any field as 'key.' It also makes each of those mini-files easy to sort into a key file. Whether sorted or unsorted, the individual field in any field-file is inexorably related to the rest of its record by direct-addressing. Thus, we will be able to search such files at least three different ways and retrieve the remainder of the record after the key is located. Since all fields can be 'key' in this system, your searches can be as flexible as you wish to make them.

The actual sorting of files for BASE has been kept a completely separate operation, and is in fact an auxiliary program that is 'chained' when we select that option from the main menu. The printing of reports is also a separate, chained program in this package. This was done to keep the program(s) small enough for a small memory, and to make the specialized sections easy to modify and/or experiment with.

If you choose to combine the listings up to this point you will find that the source program runs around 400 lines and takes about 10K of filespace. When compiled with CB-80 it results in a machine language file of approximately 18K. The sorting and printing programs are both considerably smaller. If you have to use the CRUN version, though, you will have less memory available because of the presence of the runtime program, so if memory is limited you might consider stopping here for awhile.

How the Searching Goes

Selective searches enable the user to extract different 'sub-sets' of information from a larger collection of data. All zip codes for Montana, for example, begin with 59; if I had several thousand subscribers in a mailing list I could extract those in Montana by asking for any zip beginning with 59. This search could be narrowed to one particular distribution point (sub-area) by asking for 598, or 599, or the exact complete code could be used to pinpoint addresses at a single post office. By the same reasoning one might want to list all customers having a given phone area code. One feature I included in my personal mailing list was a 'code' field. If the code is XC, that address is one to whom I send Christmas cards. Once a year, then, I can extract my card list from the hundreds of addresses I keep for other purposes. In my humble opinion being able to 'key-in' on any field, and to use partial keys, is essential to database operation.

The following section will work with the files that the earlier portions of the program created, regardless of whether the data has been sorted or not. It is a simple sequential search; however, it goes quite rapidly because only one field (of each record) need be read until the sought-for record is found. If we examine the basic searching routines of BASE (which follow), we will have looked at the entire main program. Keep in mind that they do not depend on the files being sorted, so you can actually run the program and get some use from it by combining the listings that were included in the earlier columns plus this one. Sorting and printing will be considered in future columns. If you wish, they can be completely omitted from the package, or you can design your own if ours don't fit your needs. Those that will be included in the package are kept simple for training purposes but have been adequate for most uses.

The main function of the sort program that will be presented is to create and update key files. Once they are created they can be used in a variety of ways.

In keeping with this program's design, a search first determines that the file(s) exists and 'learns' the structure of its records. (Remember, there may be several mini-databases on the same disk, running from this one program.) You then choose which field to use as key, and which portion of it (for example, ZAN for ZANZIBAR), and specify that you wish to match the key, or see records greater or less than the key. You might, for example, specify 'less than ZZZ' to view the entire file. A recently added feature permits us to specify a range of values such as greater than A, and less than C, for example, to list all the existing entries that begin with B. Although it is not shown in this listing it can be added quite simply, since it is just a combination of the 'greater than' and 'less than' comparisons.

When the key is found the entire record is read and displayed. You have the option of either continuing to look through the file or returning to the menu to proceed with another function. ■

H-8 Print Buffer Subroutines, continued from page 10

```
$
        Output a character
$
QZCHROT POP     D                       ;Get return
        POP     H                       ;Get character
        PUSH    H                       ;Fix stack
        PUSH    D                       ;Fix stack
        MOV     Art                     Unto A
        OUT     OUTDATA                 (Send it
        RET
•endass
```

FAX-64: Facsimile Pictures on a Micro

by Michael J. Keryan

Several years ago, most home computers were owned by "hackers." These people knew how every integrated circuit in their computer worked and they could explain every byte of the code in their computer's operating system. Although there are probably just as many hackers around today, there are a lot more owners of home computers classified as "users." Due to the rapid advancement in software, you no longer have to be a computer genius to use a computer. The most popular uses for home computers now include games, word processing, spreadsheets, education, and graphics. Graphic software packages for home computers like the Commodore 64 are becoming more and more powerful.

To create your own masterpiece of art with a graphic input device such as a joystick, light pen, track ball, or graphic tablet can take several hours, and the results are highly dependent on your talent as an artist. Suppose you would like to include a more realistic picture of you, a relative, or a pet in a Basic program you are writing. If you lack the artistic talent to draw these pictures, all is not lost. You can input pictures to your computer through an electronic scanning device.

In this article, I'll describe the interface for a machine that can read any picture from a piece of paper and translate the picture to signals that your computer can handle. The machine is a facsimile machine, commonly referred to as FAX.

What is a FAX?

The facsimile machine is quite common in the business world. It is used to transfer a page of information (usually 8' by 11 inches) over the phone line. At the transmitting end, a sheet of paper with text, graphics, or whatever, is fed into the machine. After a few minutes of whirring sounds, the paper comes back out of the machine. At the receiving end, the page image is reconstructed on a thermal printing device. The two FAX machines are connected to each other by modems in very much the same way that two computers communicate over phone lines.

FAX machines have been used to transmit both text and graphics. With the advent of low cost computers, however, it is now becoming quicker and cheaper to send text from one point to another by purely digital means. FAX machines are still popular and will always be used for special applications (AT&T still says a FAX is the only way to transmit hand-drawings, signed legal documents, etc.), but more and more companies are replacing their FAX networks with computer networks for text transmission.

The result of this change-over in technology is that businesses are now dumping used FAX machines into the surplus market. These machines, which cost up to $4000 new, can now be obtained for less than $200. Used FAXs in good working condition are quite a bargain and can be obtained from dealers of used office equipment and large electronic/computer surplus dealers.

Some Facts about FAX

A FAX machine is actually two machines in one. The transmitter feeds in a sheet of paper, scans it, translates the image to an electrical signal, modulates the signal, and sends it over a telephone line. This is the part we're interested in. The receiver takes the telephone signal, demodulates it, and translates the electrical signal to an image on a fresh piece of paper. We are not interested in the receiver end, although it can be used for a very slow, low quality printer. (Note — in most FAX machines, looping the output signal of the transmitter to the input of the receiver will turn the machine into a copying machine.)

There are probably just as many FAX "standards" as there are companies that make the machines. The signal can be amplitude modulated by a carrier of constant frequency, or it can be frequency modulated, giving a constant amplitude saw-tooth signal in which the frequency varies with brightness level of the scan. The signal can be digital in nature (only two levels: black and white; commonly used for text), or the signal can be analog, in which an infinite variation in gray levels are possible. An analog type signal is required for pictures. Various scan rates are used, from a fast 5 scan lines per second to 2 scans per second. In addition, the vertical resolution can vary from about 80 to over 200 scan lines per inch.

For our purptoses, what standard should be used? Since the primary purpose is to digitize pictures, we will need an analog signal. Since most home computers can easily keep up with the scan rates involved by using machine language routines, we should use the fastest scan rate available. Another consideration is deciding between AM versus FM signal modulation. FM will reduce the amount of noise in the picture, but a few pixels of noise are not really noticeable in a digitized picture.

Luckily, the machine I obtained had a multitude of switches that could be used for just about any standard. The machine is a Burroughs DEX 4100, which I currently have set up in the following mode:

Machine	DEX mode
Speed	High
Res	Norm
ResX2	Off
TX Level	Norm
Doc	Photo
Simplex	On

The resulting output is an amplitude modulated signal with a carrier frequency of 1920Hz. The peak-to-peak signal varies from approximately 1 volt (black) to nearly 0 volts (white). The scan rate is exactly 5 Hz, giving 88 scan lines per inch. An entire 11 inch long sheet of paper is scanned by approximately 955 lines in a little over three minutes.

The hardware and software presented in this article will work with a DEX 4100 FAX machine connected to a Commodore 64 computer. Other FAX machines, other transmission standards, and other 6502 computers can be used. However, other equipment will require revisions in the machine language software and possibly in the interface circuitry as well. But the techniques shown can be used as a starting point for any other configuration.

Some Design Considerations

Before jumping into the hardware and software design, let's think about how we will use the machine with our C-64. In the mode I chose to use, the FAX can digitize graphics at a resolution of over 50 dots/inch horizontally (along a scan line) and over 80 dots/inch vertically (from line to line). The most important criteria is that the aspect ratio of a picture is unchanged. A circle on the original should still look like a circle on the digitized image; it should not look like an oval. Another nice-to-have feature is that the picture will not have to be rotated 90 degrees to look at it. Most 8 1/2 by 11 or 8 x 10 pictures are oriented vertically (like the page you are now reading); this means that we would only use about half the page. The C-64's graphic resolution is 320 horizontal, 200 vertical. 320 dots with about 50 dots/inch gives a little over 6 inches out of the total 8 or 81/2 inch picture width. This is acceptable because the important picture content is almost always near the center. Due to the C-64 aspect ratio, the height of the digitized image on the original is a little over 4 inches. The FAX's vertical resolution is twice what we need, so we'll plan on using only every other scan line.

If you've been keeping up on the C-64 graphic articles, you know that there are two distinct bit mapped modes: HIRES with 320 x 200 pixels and two colors, and MULTI with 160 x 200 pixels and four colors. Actually, the two or four color restrictions pertain to an 8x8 grid of dots and other 8x8 grids can have other color combinations. But since the scanning and digitization will be completely automatic, it is much simpler to restrict our pictures to two colors in HIRES mode and four colors

This example of an image produced by FAX-64 has been reduced from the printout size of 7%2 x 9.

in MULTI mode. However, we won't restrict our colors to black, white, and shades of gray. It is very desirable to be able to choose any color we want for any level of intensity.

Another feature that we would like to have is the ability to control where the top of the picture should be, by use of the keyboard. After the picture has been transferred to the computer and is displayed on the screen, we would like to save it to a disk file in a format that is compatible with other graphic aid and graphic print programs. This way, we can further enhance the pictures and get hard copies of them.

FAX to C-64 Interface

The signal coming out of the FAX is a relatively low voltage modulated analog signal. The interface must amplify the signal, demodulate it, and convert it to a digital signal (D/A converter). The extremely simple circuit I came up with, shown in Figure I, will do all the required signal conversions.The five volt power supply in the Commodore 64 is used to power the interface. The signal coming from the FAX is divided by a 50Kohm potentiometer. This functions as a brightness control. The reduced signal is amplified by Al, one quarter of a low cost quad op-amp (ICI: LM3900). The output of Al is inverted by A2. The outputs of both Al and A2 are summed through diodes by A3, which acts as a full wave rectifier and demodulator. A4 inverts the signal and buffers it. The output of A4 varies from about two to four volts, in direct proportion to the brightness of the FAX scan at that instant.

IC2 (LM 339) is a quad comparator. A5, A6, and A7 are set to switch at about 3.75,3.12, and 2.5 volts, respectively. The output of A4 is fed to all three comparators, which digitize the signal into four distinct levels from dark black to bright white. The comparator outputs are connected directly to three I/O bits of one of the C-64's CIA chips, through the USER port. (For computers other than the C-64, any PIA type I/O port could be used: 6820,6821, 6520,6522, etc.) The software driver will convert the three bits to a two bit binary code to signify gray level. For other computers that can display more gray levels, a more sophisticated analog to digital conversion would be required. But for the C-64 (and most inexpensive home computers), four distinct gray levels are most appropriate.

The capacitor attached to A3 demodulates the signal by filtering out the higher frequency carrier. The value shown results in a good compromise of low noise and acceptable resolution. For other standards, you may desire to change the capacitor values. If the horizontal resolution is found to be less than desired, reduce this value. *If* the output has too many light to dark transitions, smooth it out by increasing the value of the capacitor.

The required cost of the interface is almost ridiculously low. The connector that attaches to the C-64 is the most expensive part (about $4.00). Any type of layout is probably O.K., since fairly low frequencies are involved. Use shielded cable to the FAX machine (shield grounded). The most desirable configuration is to connect the PC board to the C-64 connector, so that the whole unit can plug into the computer's USER port.

Synchronization of the Signal

The translation of the signal from analog to digital was pretty straightforward. But at this point, a good question is "How in the world do you synchronize this signal to the computer?" This proved to be the most difficult aspect of the project. Initially, I used a very stable crystal controlled clock and divider chain. It was impossible to adjust the timing so that the image was stationary on the screen. A vertical line from the FAX would drift as much as 10 to 12 pixels to the left or right on the screen image. Next, I tried a phase lock loop oscillator, synchronized to the power line (60Hz). This was even worse; the vertical line ended up somewhat sinusoidal.

I tried to use the modulation frequency of the FAX itself (1920Hz), but this oddball frequency was not acceptable. It required a conversion to another frequency and a phase locked loop, since on white scenes, the modulated signal dropped to zero. This technique proved to be overkill and needlessly complex.

Another way to do it is to lock onto the picture signal itself with software. A representation of the demodulated signal (the output of A4) is shown in Figure 2. If an 81/2 inch piece of paper is centered in the machine, the scan will give black guard bands off the edges. Between these black bands is a white

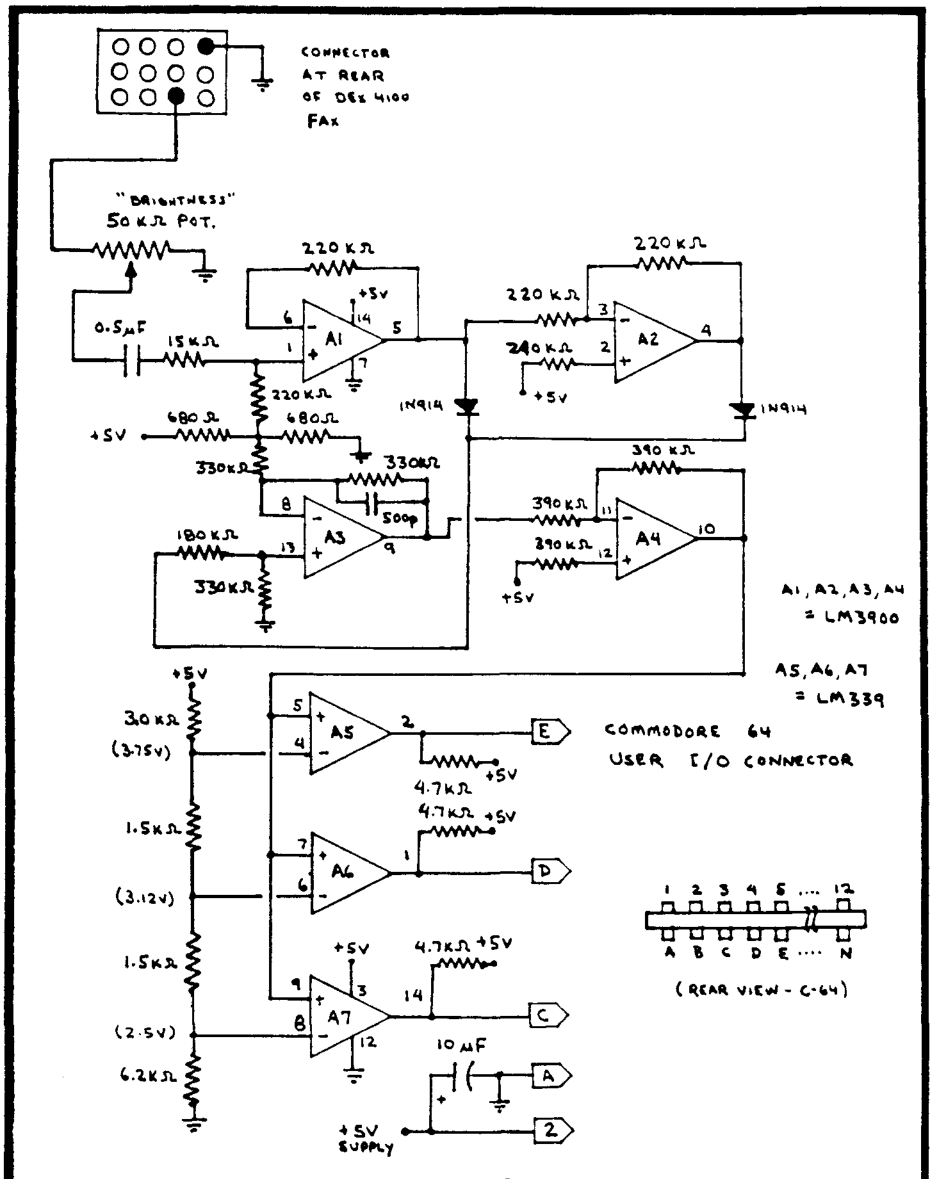


Figure 1: Interface Schematic

Listing 1

```
0000                    ;*********************************
0000                    ;* FAX DRIVER FOR C-64               *
0000                    ;* INPUTS THROUGH USER PORT          X
0000                    ;*                                   X
0000                    ;X M. J. KERYAN 9-04-84              X
0000                    ;*********************************
0000                    ;
0000                    PL      - $FD        ;PAGE ZERO
0000                    PH      = $FE        ; POINTERS AND
0000                    TEMP = »FB           ; TEMPORARY
0000                    TEMPEU = »FC         ; REGISTERS
0000                    ;
0000                    DATA IN - SDD01      ;INPUT PORT
0000                    ICR     = SDD0D      (INTER. CONTROL
0000                    ;
0000                    LTA8 = *4300         ;THIS TABLE IS
0000                    LTABA - $4300        ; USED TO
0000                    HTAB = $4400         ; CONSTRUCT
0000                    HTABA - $44D0        ; ADDRESSES
0000                    ;
0000                    X       - *4500
4500                    ■
4500    78              NEWMI SEI            (TURN OFF INTER
4501    2C 0D DD              BIT ICR        ;FAX INTERRUPT?
4504    30 01                 BM1 SAUREG
```

sync pulse. Since the scan rate is constant (I scan line every 0.2 seconds), the width of the sync pulse is constant. I decided to use the white sync pulse, followed by the black guard band, to sync the picture. This proved to be very stable, the only noticable by-product being a plus or minus one pixel uncertainty. Since I use the FAX mostly for pictures, this one pixel uncertainty is usually undetectable.

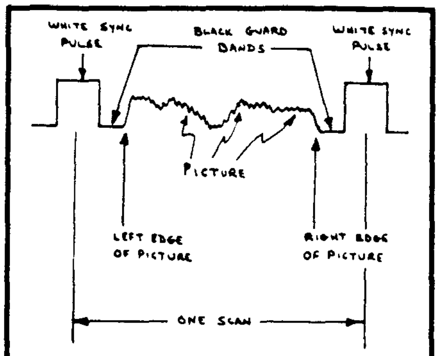


Figure 1: Modulated signal (output of A4).

The dot clock was generated by software. By doing so much in software, we have significantly reduced the compexity and cost of the interface. That's the good news. The bad news is, of course, that the software required to support the hardware is quite complex.

So far, we learned how a facsimile machine (FAX) works and we looked at a simple interface circuit that can be used to connect a FAX machine to a home computer, such as a Commodore 64. Before jumping into the software, an overview of the operation is helpful.

The CIA integrated circuit (6526) in the C-64 is used as a parallel input port to accept three bits of digitized information from the interface circuit. In a completely different application, the timer in the CIA is used as a clock signal. The clock is used to generate Non-Maskable Interrupts (NMI) at a frequency of approximately 3000Hz. During a scanning operation, everytime a NMI occours, data is sampled from the input port, converted to a pixel (picture element), and stored in graphic memory. Since this operation happens in the background, we can have a Basic program and even another machine language program running at the same time. This foreground/background mode of operation greatly simplifies the programming.

```
4506  40             RTI              ;NO, IGNORE IT
4507  48       SAREG PHA
4508  8A             TXA
4509  48             PHA
450A  98             TYA
450B  48             PHA
450C  EE F8 46       INC COUNTL       (ADD 1 TO
450F  D0 03          BNE N0           ; COUNTERS
4511  EE F9 46       INC COUNTH
4514  EE FA 46 N0    INC COUNTS       (COUNTS COUNTS
4517  AD FA 46       LDA COUNTS       ; PULSES BY 8
451A  C9 08          CMP #$08         (WHEN = 8
451C  00 08          BNE SYNC         ; COLUMN COUNT
451E  A9 00          LOA #See         ; IS INCREMENTED
4520  80 FA 46       STA COUNTS
4523  EE FC 46       INC COLUMN
4526           i
4526           ; SYNCFLG IS A FLAG TO DENOTE
4526           ; STATE OF SYNCHRONIZATION:
4526           ;    = 128 OR HIGHER — IN SYNC
4526           ;    = 1 TO 127 — PHASING MODE
4526           (    = 0 -- OUT OF PHASE
4526           ;
4526  AD FF 46 SYNC LDA SYNCFL        (IS FLAG >127?
4529  30 58          BMI N1           (YES, PHASED
4528  00 42          BNE LOCKED       (NO, LOCKING?
452D  AD 01 00       LDA DATAIN       (NO, LETS GET
4530  29 07          AND #s07         (IN PHASE THEN
4532  C9 07          CMP #$07         (DATA = BLACK'
4534  F0 14          BED LASTBL       (YES.............
4534  A5 FC          LDA TEMPEV       ;NO, WAS LAST
4538  C9 07          CMP #$07         (ONE BLACK'
453A  F0 05          BEG ZWHT         (YES, .............
453C  E6 FB          INC TEMP         (NO, MORE WHITE
453E  4C 80 45       JMP LRT
4541  A9 01    ZWHT LDA #se I
4543  85 FB          STA TEMP         ;1 WHITE
4545  85 FC          STA TEMPEV       (LAST = WHITE
4547  4C 80 45       JMP LRT
454A  A5 FC    LASTBL LDA TEMPEV      (LAST DATA
454C  C9 07          CMP #se7         ;WAs IT BLACK'
454E  00 0B          BNE CHKWH        (NO, ................
4550  A9 00    WHZERO LDA #s00        ;YES ZERO WHITE
4552  85 FB          STA TEMP         (COUNTER
4554  A9 07          LDA #$07         (MAKE LAST ONE
4556  85 FC          STA TEMPEV       ; BLACK
4558  4C 80 45       JMP LRT
4558  A5 FB    CHKWH LDA TEMP         (WHITE COUNTER
4550  C9 4C          CMP #76          ( <76?
455F  90 EF          BCC WHZERO       (YES, WAIT
4561  C9 50          CMP 080          ( >79?
4563  80 EB          BCS WHZERO       ;YES, PAST IT
4565  A9 65          LDA 0101         ;A HIT! NOW
4567  80 FF 46       STA SYNCFL       (SET TO LOCKED
456A  85 FB          STA TEMP         (COUNT FROM 48
456C  4C 80 45       JMP LRT
456F  C6 FB    LOCKED DEC TEMP        (IS COUNTER DOWN
4571  D0 00          BNE LRT          (TO ZERO?
4573  A9 FF          LDA #SFF         (YES, SET SYNCFL
4575  80 FF 46       STA SYNCFL       (TO SCAN
4578  A9 FF          LDA 0»FF         (RESET COUNTERS
457A  8D FB 46       STA LINE         ( THROUGH CODE
4570  4C C4 45       JMP N2           ( AT N2
4580  4C F2 46 LRT   JMP RETURN
4583  AD F9 46 NI    LDA COUNTH
4586  C9 04          CMP #$04         (HI BYTE (42
4588  90 6B          BCC N4           (YES, BRANCH
458A  AD F8 46       LDA COUNTL
4580  C9 B0          CMP 0*80         (COUNT?1199?
458F  80 33          BCS N2           (YES, MAX COUNT
4591  C9 45    KSYNC CMP 069          (IS COUNTER
4593  90 2C          BCC KRT          (WITHIN LIMITS'
4595  C9 51          CMP 081
4597  80 28          BCS KRT
4599  AD 00 47       LDA CHKSFL       (SHOULD WE
459C  De 23          BNE KRT          (CHECK FOR SYNC?
459E  AD 01 DD       LDA DATAIN       (YES, GET DATA
45A1  29 07          AND 0«07
45A3  C9 02          CMP 0*02         (IS IT DARK?
45A5  80 07          BCS K1           (YES.............
45A7  A9 01          LDA 0*01         (NO, MAKE LAST
45A9  85 FC          STA TEMPEV       (LIGHT
45A8  4C CI 45       JMP KRT
```

Machine Langauage FAX Driver

The assembler code for the machine language program is shown in Listing 1. The first thing to describe is the new NMI routine. The pointers for this routine are poked into memory and activated by a Basic program described later. NEWNMI first checks to see if the NMI actually came from the CIA chip. If so, all registers are saved. The program counts each NMI (each dot) by eight because eight dots make up a byte of screen data.

A flag is used to denote the state of synchronization. If the flag is 0 (initial state), the signal is out of sync. If the flag is between 1 and 127, the program goes into a special phasing mode. If >127, the flag denotes that sync is established. We'll look at each sync mode separately.

If out of sync, the program will look for a string of white bytes that are between 76 to 80 dots wide. This string must be bounded on both sides by black dots. If any dot is out of sequence, this routine will reset and continue to look for this sequence. This white area is the sync pulse described previously. The length (76-80 dots) is dependent on the machine scan speed and software timer period. Any change in these will require a new window size. Once the sync pulse is found, the program changes the sync flag so that operation will go to the special phasing Mode (LOCKED). A count of 101 is stored into a counter which is decremented by each NMI (each dot). At this point, the correct phase is established, and we are at the left margin of a picture. The flag is then changed to denote that sync is established and counters are zeroed for the vertical line number, the horizontal count, and the horizontal byte number (column).

If the NMI routine is entered while in sync, it first checks to see if 1200 dots have occurred. If so, then one complete scan line (actually two physical scans since we ignore alternate scan data) is completed and we are then at the end of a line. We use this opportunity to check the keyboard for the'T' key. T is pressed when you want the picture started at the top again, so the line number is zeroed. At the end of the 200th line, the picture is complete, so the routine kills itself by disabling FAX interrupts (see N3).

```
45AE   A5 FC       K 1     LDA TEMPEV    ; ! S LAST DARK"
4580   09 02               CMP **02
4582   90 05               BCC K2        ;NO............
4584   85 FC               STA TEMPEV    ;YES, MAKE SURE
4586   40 01 45            JMP KRT       ;DARK NOW
4589   A9 4B       K2      LDA #75       ;SYNC IT TO
45BB   8D F8 46            STA COUNTL    jA COMMON POINT
45BE   8D 00 47            STA CHKSFL    (SET FLAG
45C 1  40 F2 46    KRT     JMP RETURN
45C4   A9 00       N2      LDA #s00      ;ZERO OUT
45C6   80 F8 46            STA COUNTL    ; COUNTERS
45C9   80 F9 46            STA COUNTH
45CC   80 FA 46            STA COUNTS
45CF   8D FC 46            STA COLUMN
4502   80 00 47            STA CHKSFL
45D5   88 FB 46            INC LINE      (GO DOWN 1 LINE
4508   AD FB 46            LDA LINE
45DB   09 C8               CMP *200      ;LINES> 1992
4500   80 08               BCS N3        ;YES, BRANCH
45DF   A5 C5               LDA *C5       |CURRENT KEY
4581   09 16               CMP *22       ; IS IT T'
4583   00 05               BNE NRT       ; FOR 'TOP'?
4585   A9 00               LDA **00      ;YES, START AT
4587   8D FB 46            STA LINE      ; TOP
458A   40 27 46    NRT     JMP ACTIVE    ;NO, START LINE
4580   A9 7F       N3      LDA #$7F      ;DI SABLE FAX
458F   8D 00 00            STA ICR       ; INTERRUPTS
45F2   40 F2 46            JMP RETURN    ; FOR NOW
45F5   09 01       N4      CMP *♦0 1     ;HI BYTE <1?
45F7   90 2E               BCC ACTIVE    ;YES, SCREEN
45F9   09 02               CMP **0 2     ;          )1?
45FB   80 07               BCS N5        ;IF SO, IGNORE
45FD   AD F8 46            LDA COUNTL
4600   09 40               CMP **40      ;COUNT <*01409
4602   90 23               BCC ACTIVE    1YES, SCREEN
4604   A5 C5       N5      LDA ♦ C5      ;CURRENT KEY
4606   09 00               CMP *13       ;IS IT 'S' FOR
4608   00 1A               BNE N6        ; 'SYNC?
460A   A9 00               LDA **00      ;RESTART ALL
460C   8D FF 46            STA SYNCFL
460F   8D 00 47    NSTART  STA CHKSFL
4612   8D F8 46            STA COUNTL
4615   8D F9 46            STA COUNTH
4618   80 FA 46            STA COUNTS
461B   80 FB 46            STA LINE
4618   80 FC 46            STA COLUMN
4621   40 50 45            JMP WHZERO
4624   40 F2 46    N6      JMP RETURN    ; IGNORE PULSE
4627                1
4627               ; IN THE ACTIVE SCAN    AREA, FIRST
4627                ( CHANGE  THE 3 BIT   DATA TO A TWO
4627               ; BIT      BINARY CODE.   THEN SAVE ODD
4627               ; DATA AND COMBINE     THEM WITH EVEN
4627               ; DATA SAMPLES. WHEN IN MULTI -
4627               ; COLOR MODE, AVERAGE THE TWO.
4627               0
4627   AD 0 1 DD   ACTIVE  LDA DATAIN    ;GET DATA
462A   29 07               AND *♦0 7     jKWSK FOR 3 BITS
4620   09 02               CMP M02       ;DATA <27
4628   90 0A               BCC TWOB1T    ;YES, 0 OR 1 OK
4630   09 04               CMP *♦04      ;        >37
4632   80 04               BCS WHITE     ;YES, BRIGHTEST
4634   A9 02               LDA #*02      {LEVEL = 2
4636   00 02               BNE TWOBIT
4638   A9 03       WHITE   LDA **03      {LEVEL - 3
463A   85 FB       TWOBIT  STA TEMP      {SAVE DATA
4630   AD FB 46            LDA COUNTL    {IS COUNTER
463F   29 0 1              AND *♦0 1     {EVEN OR ODD?
4641   00 07               BNE ODD -     {BRANCH ON ODD
4643   A5 FB               LDA TEMP      {IF EVEN THEN
4645   85 FC               STA TEMPEV    (SAVE TTLL NEXT
4647   40 F2 46            JMP RETURN    {TIME (ODD)
464A   AD 0 1 47   000     LDA MODE      {MULTICOLOR OR
4640   F0 09               BED HI RES    (HIRES MODE?
464F   A5 FB       MULTI   LDA TEMP      (MULTI MODE
4651   18                  CLC           {HERE, WE'LL
4652   65 FC               ADC TEMPEV    (AVERAGE TWO
4654   4A                  LSR A         (CONSECUTIVE
4655   40 63 46            JMP CALCBT    (BYTES
4658   A5 FC       HIRES   LOA TEMPEV    (HIRES MODE
465A   29 02               AND **02      {PLACE THE TWO
4650   85 FC               STA TEMPEV    {BITS IN THE
465E   A5 FB               LDA TEMP      {CORRECT ORDER
4660   4A                  LSR A         I (PRI OR = HIGH
```

During the 'dead' part of the scan, which is the physical scan line that we ignore, the keyboard is looked at again, this time for the 'S' key. On very rare occasions, a glitch may destroy the synchronization. The S key is used to start everything over again to re-establish sync.

If the sync is established, and we are in the active scan area (dots 0 to 319), the program jumps to ACTIVE. Here the data is sampled from the three input lines and converted to a two bit binary code: 00,01,10, or 11. Next, the mode of operation is looked at. If we are in HIRES mode, then we only want two levels of intensity for each dot. If the MULTIcolor mode is desired, then we will look at two consecutive dots, average them, and give the corresponding two-bit code for the average intensity level of the two dots. In either case, two bits are used to update the screen in the active scan area.

The addresses for the bit-map area are determined from a set of lookup tables (see Listing 2). These addresses are constructed based on the vertical line number (0-199) and the horizontal column number (0-39). The new two bits are placed at the correct bit locations into the byte of data, and this byte is stored back in the 8K bit map memory area. Since there are four possible places to put these two bits, there are four small routines for this: BO, B2, B4, and B6.

You can see that the NMI routine actually does quite a bit. It handles all the synchronization of the FAX, inputs the scan data, checks the keyboard for S or T keys, keeps track of the screen locations, and handles all the screen writing in either HIRES or MULTI modes.

Other Machine Language Utilities

Listing 1 also shows a few more utility programs. These are not part of the FAX driver but are instead called by SYS statements from our Basic program. First is a routine to clear the 8K bit-map area of memory. Also included are routines to clear the 1K areas for the screen and color memory. Actually, with a POKE from Basic, these routines can change the colors to any desired. A routine (SAVE) is included to move the different memory areas to other areas that are com-

```
4661   05 FC              ORA TEMPER      •.CURRENT = LOH)
4663   80 FD 46   CALCBT  STA TEMPI       ;SAVE IT FOR NOW
4666   AD FC 46           LDA COLUMN      ;IS COLUMN #
4669   C9 28              CMP N40         ; <40?
4668   90 03              BCC SCR0        ;YES, GO ON
466D   4C F2 46           JMP RETURN      (OTHERWISE EXIT
4670   A8         SCR0    TAY             (SAME COL Y REG.
467 1  AE FB 46           LOX LINE        ;LINE# IN X REG.
4674   BD 00 43           LDA LTAB,X      (CONSTRUCT
4677   85 FD              STA PL          (VIDEO ADDRESS
4679   BD 80 44           LOA HTAB,X      (FROM X=LINE
467C   85 FE              STA PH          ( <0-199)
467E   B9 00 43           LDA LTABA,Y
4681   18                 CLC             (AND Y-HORIZ.
4682   65 FD              ADC PL          (BYTE <0-39>
4684   85 FD              STA PL
4686   90 0 2             BCC SCRI
4688   E6 FE              INC PH
468A   89 00 44   SCRI    LDA HTABA,Y
468D   18                 CLC
468E   65 FE              ADC PH
4690   85 FE              STA PH
4692   A0 00              LDY #see        (FINALLY, GET
4694   BI FD              LDA (PL) ,Y     (BYTE FROM SCREEN
4696   8D FE 46           STA TEMP2       (HOLD IT
4699   AD F8 46           LDA COUNTL      (FIND MOD<8) OF
469C   29 07              AND #s07        ; PULSE
469E   C9 82              CMP #$02        i>I?
46A0   B0 14              BCS B2          (YES, BRANCH
46A2   AD FE 46   B0      LDA TEMP2
46A5   29 3F              AND #$3F        (MASK- 00111111
46A7   80 FE 46           STA TEMP2
46AA   AD FD 46           LDA TEMPI
46AD   0A                 ASL A           (MOVE THE TWO
46AE   0A                 ASL A           (BITS TO THE
46AF   0A                 ASL A           (TWO HIGH BITS
4680   0A                 ASL A           (7 AND 6
46B1   0A                 ASL A
46B2   0A                 ASL A
46B3   4C EB 46           JMP B8
46B6   C9 04      82      CMP #$84        ;>3?
46B8   B0 12              BCS B4          (YES, BRANCH
46BA   AD FE 46           LDA TEMP2
46BD   29 CF              AND #SCF        (MASK— 11001111
46BF   80 FE 46           STA TEMP2
46C2   AD FD 46           LDA TEMPI
46C5   0A                 ASL A           (MOVE THE TWO
46C6   0A                 ASL A           (BITS TO THE
46C7   0A                 ASL A           (BITS NUMBERED
46C8   0A                 ASL A           ;5 AND 4
46C9   4C EB 46           JMP B8
46CC   C9 06      B4      CMP #$06        (>5?
46CE   88 18              BCS B6          (YES, BRANCH
46D0   AD FE 46           LDA TEMP2
46D3   29 F3              AND #$F3        (MASK- 11110011
4605   BD FE 46           STA TEMP2
46D8   AD FD 46           LDA TEMPI
460B   8A                 ASL A           (SHIFT THEM TO
46DC   0A                 ASL A           (BITS 3 AND 2
4600   4C EB 46           JMP B8
46E0   AO FE 46   B6      LDA TEMP2
46E3   29 FC              AND MFC         (MASK- 11111100
46E5   8D FE 46           STA TEMP2
46E8   AD FD 46           LDA TEMPI       (ONLY THESE BITS
46EB   0D FE 46   88      ORA TEMP2       (AFFECT DATA
46EE   A8 08              LDY M00
46F0   91 FD              STA (PL) ,Y     (CHANGE SCREEN
46F2   68         RETURN  PLA             (RESTORE
46F3   A8                 TAY             ( REGISTERS
46F4   68                 PLA
46F5   AA                 TAX
46F6   68                 PLA
46F7   40                 RTI
46F8                      I
46F8   00         COUNTL  .BYTE 0
46F9   00         COUNTH  .BYTE 0
46FA   00         COUNTS  .BYTE 0
46FB   00         LINE    .BYTE 0
46FC   00         COLUMN  .BYTE 0
46FD   00         TEMP 1  .BYTE 0
46FE   00         TEMP2   .BYTE 0
46FF   00         SYNCFL  .BYTE 0
4700   00         CHKSFL  .BYTE 0
470 1  00         MODE    .BYTE 0
```

patible with popular graphics programs. Here we move the 8K bit-map area starting at $2000 to $6000 + . The 1K screen area starting at $0400 is moved to two locations, $5C00 and $7F40. The 1K color memory starting at $D800 is moved to $8328, and the background color from $D021 to $8710. The reason for these memory moves are to prepare for a disk save routine. The above locations are compatible with two packages, "DOODLE", a graphics program by Omni Unlimited, and "Koalapainter", by Audio Light. DOODLE is used for HIRES pictures, and KOALA for MULTIcolor pictures.

The DISKSA routine will create a disk file in either mode, depending on the state of the mode flag. To use this routine, the file name and length of the name have to be previously stored in memory.

Basic Program for FAX Driver

Listing 3 is a Basic program that is used to control the machine language programs. First it reads into memory the ML part ("FAX64.ML") and the table of Listing 2 ("TABLE"). Then the top of memory is set to avoid conflicts with the graphics.

The main menu allows four options: FAX scan, Display last scan, Save scan to disk, or Quit. Obviously, the F option is chosen the first time. You are then given the choice of a HIRES scan (only two colors) or a MULTIcolor scan (four colors). Then some other commands are shown and you are instructed to start the FAX machine.

POKEs are made to start the interrupt routine (NMI) and the software timer, as well as to configure the screen for graphics. At this point, the NMI routine is active in the background. The Basic program is in a do-nothing state, checking the keyboard for Q to quit or for color change keys.

If MULTI mode is chosen, the colors can be changed either during a scan or after. The four function keys can be used to change any of the four colors. F1 is used for the brightest level (usually white), F3 for the next, etc. With these four keys, any color combination is possible. The number keys are used to select any of 10 preset color combinations. The '2' key selects shades of white (gray, black), the '3' key is used for shades of red, etc. Also, the 'C' key can be used to rotate from 1 to 2

```
4702                    ;
4702   00               LENGTH  .BYTE *00             ;FOR DISKSA
4703   00               NAME    .BYTE 0,0,0,0,0,0,0,0
4704   00
4705   00
4706   00
4707   00
4708   00
4709   00
470A   00
4708   00               CONTO   .BYTE 0,0,0,0,0,0,0,0
470C   00
4700   00
470E   00
470F   00
47 10  00
471 1  00
4712   00
47 13                   ;
4713                    SAVEKN =    65496             ;KERNAL
4713                    SETLFS =    65466             ; ROUTINES
4713                    SETNAM =    65469
4713                    1
4713                    ;       SUBROUTINES FOR CLEAR
47 13                   ;       SCREEN, SET COLORS
47 13                   ;
4713   A0  20           CLRBIT  LOY #$20
47 15  8C  43 47                STY LOOPCL+2
47 18  A9  AA                   LDA #SAA
47 1A  4C  3F 47                JMP CLEAR
471D                    ;
471D   AD  0 1 47       CLRCOL  LDA MODE
4720   00  0A                   BNE CLRMUL
4722   A9  0 1          CLRHIR  LDA #se l
4724   A0  04           CLR 1   LDY 4*04
4726   8C  43 47                STY LOOPCL+2
4729   4C  3F 47                JMP CLEAR
472C                    ;
472C   A9  0 1          CLRMUL  LDA 4*01
472E   80  21 D0                STA *D0 21
4731   A9  FC                   LDA 4«FC
4733   20  24 47                JSR CLR1
4736   A9  00                   LDA 4*00
4738   A0  08                   LDY 4»D8
473A   8C  43 47                STY LOOPCL+2
4730   A0  04                   LDY #$04
473F   A2  00           CLEAR   LDX 4*00
474 1  90  00 04        LOOPCL  STA *0400,X
4744   E8                       I NX
4745   00  FA                   BNE LOOPCL
4747   EE  43 47                INC LOOPCL+2
474A   88                       DEY
474B   D0  F4                   BNE LOOPCL
4740   60                       RTS
474E                    ;
474E   A0  20           SAME    LDY 4*20              ;8K BIT-MAP
4750   8C  B3 47                STY MOVE 1+2          ; FROM *2000
4753   A9  60                   LDA 4«60              ; TO *6000
4755   80  B6 47                STA MVTO+2
4758   A9  00                   LDA 4«00
475A   8D  B2 47                STA MOUE 1+!
4750   8D  B5 47                STA MVTO+ 1
4760   20  AF 47                JSR MOUE
4763   A0  04                   LDY 4«04              ;1K SCREEN
4765   8C  B3 47                STY MOUE 1 + 2        1 FROM *0400
4768   A9  5C                   LDA 4*5C              ; TO »5C00
476A   80  B6 47                STA MVTO+2
476D   A9  00                   LDA 4«00
476F   80  B2 47                STA MOUE!+1
4772   >«D B5 47                STA MUT0+1
4775   20  AF 47                JSR MOUE
4778   A0  04                   LDY 4*04              J1K SCREEN
477A   8C  B3 47                STY MOUE 1 + 2        | ALSO TO
4770   A9  00                   LDA 4*00              ;         *7F40
477F   80  B2 47                STA MOUE 1+1
4782   A9  7F                   LDA #$7F
4784   80  B6 47                STA MVTO+2
4787   A9  40                   LDA 4*40
4789   8D  85 47                STA MVTO+ 1
478C   20  AF 47                JSR MOUE
478F   A9  D8                   LDA 4*08              ;1K COLOR
4791   8D  B3 47                STA MOUE 1 + 2        1 FROM *D800
4794   A9  00                   LDA 4*00              | TO *8328
4796   80  B2 47                STA MOUE 1+1
```

to 3, and so on. It should be noted that these color changes can be made while the machine is scanning since the program is running simultaneously with the NMI routine.

The Save option first moves the memory areas, turns on the alpha screen, and asks you for a file name. This name is then configured to be compatible with either KOALA or DOODLE, and the name and its length are POKED into memory. You are then instructed to place a diskette into the disk drive. You can change disks at this point, but be sure you use previously formatted disks. Then the ML Save routine is called and you see the menu again.

What Good is a FAX?

What can you do with the FAX? Well, actually, it can prove to be the best graphic input device that you can use with your home computer. You can capture an image of any picture into your machine. By creating a disk file of the picture, you can then add text, fill in color areas, or enhance the images with other graphic packages. You can dump these pictures to your printer with an appropriate printer dump program. And you can do it even if you are not very good at drawing.

Some Helpful Hints

Make sure the picture you feed into the FAX is of sufficient contrast and is not bathed in dark shadows. Do not attempt to use it to read in fine text —the interface is configured to reduce the resolution to that of C-64 graphics capability. Make sure the 'brightness' control is set up right so that the machine is putting out four different levels to your computer. To set it, make yourself a 'test pattern' with different gray levels, and digitize it in MULTI mode.

If you have more questions, suggestions, or have found a unique application for the FAX machine, write to me at 713 Locust Drive, Tallmadge, Ohio 44278.1 can supply the C-64 FAX driver programs in Commodore 1541 disk format for $10. A FAX machine as used in this article can be obtained from Computer Products & Peripherals Unlimited, Box 204, Newton, New Hampshire 03858, for approx. $169. ■

```
4799  A9 83                 LOA #$83
4798  80 86 47              STA MVTO+2
479E  A9 28                 LDA #$28
4 7A0 80 B5 47              STA MVTO*1
47A3  A0 04                 LDY #se4
47A5  20 AF 47              JSR MOVE
47A8  AD 21 00              LDA $D021       ; 1 BYTE SD021
47AB  8D 10 87              STA $8710       ;     TO $8718
47AE  60                    RTS
47AF               ;
47AF  A2 00        MOVE     LDX wsee        ;MOVE
47B1  AD 00 04     MOVE 1   LDA $0400       ; SUBROUTINE
4784  80 40 7F     MVTO     STA »7F40
47B7  EE B2 47              INC MOVE 1*1
47BA  AD B2 47              LDA MOVE 1+ 1
47BD  D0 03                 BNE MV 1
47BF  EE B3 47              INC MOVE 1*2
47C2  EE B5 47     MV!      INC MVTO*1
47C5  AD B5 47              LDA MVTO*1
47C8  D0 03                 BNE MV2
47CA  EE B6 47              INC MVT0+2
47CD  E8           MV2      I NX
47CE  D0 El                 BNE MOVE 1
47D0  88                    DEY
4701  D0 DE                 BNE MOVE!
4703  60                    RTS
47D4               ;
47D4  A2 08        DI SKSA  LDX #se8        ; ROUTINE TO
4706  A9 07                 LDA #$07        ; SAVE THE
4708  A0 00                 LDY #$00        ; GRAPHICS
47DA  20 BA FF              JSR SETLFS      ; TO DISK IN
4700  AD 02 47              LDA LENGTH      ; FORMATS:
47E0  A2 03                 LDX WCNAME      ;
47E2  A0 47                 LDY W>NAME      ; HIRES-
47E4  20 BD FF              JSR SETNAM      ;    DOODLE
47E7  AD 01 47              LDA MODE        ;
47EA  F0 18                 BEO HIRESC      ; MULTI -
47EC  A9 60                 LDA #$60        ;    KOALA
47EE  85 FE                 STA PL*1
47F0  A2 1 1                LDX #s 1 1
47F2  A0 87                 LDY #$87
47F4  A9 00        TOSAVE   LDA 4»00
47F6  85 FD                 STA PL
47F8  A9 FD                 LDA *<PL
47FA  20 D8 FF              JSR SAVEKN
47FD  80 02                 BCS ERR
47FF  A9 00                 LDA #s00
480 1 85 FD        ERR      STA PL
4803  60                    RTS
4804  A9 5C        HIRESC   LDA #$5C
4806  85 FE                 STA PL*1
4808  A2 00                 LDX #$80
480A  A0 80                 LDY #s80
480C  D0 E6                 BNE TOSAVE
480E               ;
480E               .END
```

ERRORS ⁼00008

SYMBOL TABLE

SYMBOL VALUE

ACTIVE	4627	B0	46A2	B2	4686	B4	46CC
B6	46E0	B8	46EB	CALCBT	4663	CHKSFL	4700
CHKWH	455B	CLEAR	473F	CLR1	4724	CLRBIT	4713
CLRCOL	4710	CLRHIR	4722	CLRMUL	472C	COLUMN	46FC
CONTD	470B	COUNTS	46FA	C0UNTH	46F9	COUNTL	46F8
DATAIN	000 1	DISKSA	4704	ERR	480 1	HIRES	4658
HIRESC	4804	HTAB	4400	HTABA	4400	ICR	DD0D
K1	45AE	K2	45B9	KRT	45C1	KSYNC	4591
LASTBL	454A	LENGTH	4702	LINE	46FB	LOCKED	456F
LOOPCL	4741	LRT	4580	LTAB	4308	LTABA	43D0
MODE	470 1	MOVE	47AF	MOVE 1	47B1	MULTI	464F
MVI	47C2	M2	47CD	MVTO	47B4	N0	4514
NI	4583	N2	45C4	N3	45ED	N4	45F5
N5	4604	N6	4624	NAME	4703	NEWNRI	4500
NRT	45EA	NSTART	460F	ODD	464A	PH	80FE
PL	00FD	RETURN	46F2	SAVE	474E	SAVEKN	FFD8
SAVREG	4507	SCR0	4670	SCRI	468A	SETLFS	FFBA
SETNAM	FFBD	SYNC	4526	SYNCFL	46FF	TEMP	00F8
TEMPI	46FD	TEMP2	46FE	TEMPEV	00FC	TOSAVE	47F4
TWOBIT	463A	WHITE	4638	WHZERO	4550	ZWHT	454 1

Listing 2. Table of Offsets -for Screen Memory

Addr	Data															
4300	00	0 1	02	03	04	05	06	07	40	4 1	42	43	44	45	46	47
4310	80	8 1	82	83	84	85	86	87	C0	c 1	c2	c3	C4	c 5	c 6	c7
4320	00	0 1	02	03	04	05	06	07	40	4 1	42	43	44	45	46	47
4330	80	8 1	82	83	84	85	86	87	C0	c 1	c2	c3	c4	c 5	c 6	c7
4340	00	0 1	02	03	04	05	0 6	07	40	4 1	42	4 3	44	45	46	47
4350	80	81	82	83	84	85	86	87	C0	c 1	c2	c 3	c4	c 5	c 6	c7
4360	00	0 1	0 2	03	04	05	0 6	07	40	4 1	42	43	44	45	46	47
4370	80	81	82	83	84	85	86	87	c 0	c 1	c2	c 3	c4	c 5	c6	c7
4380	00	0 1	02	03	0 4	05	0 6	07	40	4 1	42	43	44	45	46	47
4390	80	81	82	83	84	85	86	87	C0	c 1	c 2	c3	c4	c5	c 6	c7
43a0	00	0 1	02	03	04	0 5	0 6	07	40	4 1	42	43	44	45	46	47
43b0	80	81	82	83	84	85	86	87	C0	c 1	c2	c 3	c4	c 5	c 6	c7
43c0	00	0 1	02	03	04	05	06	07	00	00	00	00	00	00	00	0 0
43d0	00	08	10	18	20	28	30	38	40	48	50	58	60	68	70	78
43e0	80	88	90	98	a0	a ti	bei	b8	C 0	c 8	du	d8	e0	e8	f 0	f8
43f 0	00	08	10	18	20	28	30	38	00	00	00	00	00	00	00	00
4400	20	20	20	20	20	20	20	20	21	2 1	21	21	21	21	21	21
4410	22	22	22	22	2 2	22	22	22	23	23	23	23	23	23	23	23
4420	25	25	25	25	Z 5	25	25	25	26	26	26	26	26	26	26	26
4430	27	27	27	27	27'	27	27	27	28	28	28	28	28	28	28	28
4440	2a	2a	2a	2 &	2 a	2 a	2 a	2a	2b	2b	2b	2b	2b	2 b	2b	2 b
4450	2c	2c	2c	2c	2c	2c	2c	2c	2d	2d	2d	2d	2d	2d	2d	2d
4460	2-f	24	2-f	2-f	2-f	2-f	2-f	24	30	30	30	30	30	30	30	30
4470	31	3 1	31	31	3 1	3 1	31	31	32	32	32	32	32	32	32	32
4480	34	34	34	34	34	34	34	34	35	35	35	35	35	35	35	35
4490	36	36	36	36	36	36	36	36	37	37	37	37	37	37	37	37
44a0	39	39	39	39	:9	39	37	39	3a	3a	3 3	3a	3&	3a	3a	s 2
44b0	3b	3b	3b	3b	3b	3b	3b	3b	3c	3c	3c	3c	3c	3c	3c	3c
44c0	3e	3e	3e	3e	3e	Le	3e	3e	00	00	0 0	00	0 0	00	00	0 0
44d0	00	00	00	00	00	00	00	00	00	0 0	00	00	00	00	00	00
44e0	00	00	00	00	00	0 0	00	0 0	00	00	0 0	00	00	00	00	00
4440	0 1	0 1	0 1	0 1	0 1	0 1	0 1	0 1	00	00	00	00	00	00	00	00

LISTING 3. BASIC FAX DRIVER PROGRAM

```
18 REM****** XX MMX ***** ****** ******** XX
20 REMX                                          X
30 REMX        FAX-DRIVER    M.J.KERYAN          X
40 REMX         FOR C-64     9-04-84             X
50 REMX                                          X
60 REM*******************************
70 IF A=0 THEN A=1: LOAD " TABLE",8,1
80 IF A=1 THEN A=2: LOAD " FAX64.ML",8,1
90 POKE52,32:P0KE56,32sPOKE644,32: CLR
100 DIM C<9,3)i CR=2
110 FOR I-0TO9I FOR J-1T03: READ C(I,J): NEXT: NEXT
120 C1=18221: C2=18226: C3=18231
130 DATA 1,173,6, 15,148,0
140 DATA 1,252,0, 1,162,0
150 DATA 3,230,0, 2,164,6
160 DATA 1,213,0, 1,230,0
170 DATA 1,120,9, 1,169,0
180 REM MENU
190 GOSUB 1410
200 PRINT"                   <RON)FAX MENUCDWNCDWN)"
210  PRINT"        <F>       FAX  SCAN
220  PRINT"<DWN)           <D)      DISPLAY LAST SCAN
230  PRINT*<DWN)           <S>      SAVE SCAN TO DISK
240  PRINT"<DWN>           <Q)      QUIT" : PRINT
250  PRINT"          79
260  GET K»; IF     KS©* ' " THEN   260
270  GET Ks: IF     KS="•   THEN  270
280 IF KS="F* THEN SC-1: GOTO 330
290 IF Ks="Q" THEN GOTO 1340
300 IF K0—"D" THEN 930
310 IF K0—"S" THEN GOSUB 860; GOTO 1090
```

Watch For These Upcoming Articles

The following list is a sample of some of the interesting articles which are in process Your suggestions for future articles are welcome

- Interfacing a parallel printer through the Apple game port.
- Dumping the S-100 graphics card mage to an Epson printer.
- Using bank switching for an extended CBIOS.
- Communicating with your computer over the phone line without a modem by using TONE CONTROL.
- Review of the Light Speed-100 256K S-100 Disk Simulator Kit
- A $500 Superbrain Computer, the pros and cons of buying out of date used equipment
- FORTH—Using the CREATE.. . DOES construct.
- Source code drivers for the NEC7220 graphics chip.
- Accessing the Apple H's graphics from within a CP/M program using a Z-80 card
- Kit Building—soldering, desoldering, and repairing printed circuit boards

Reviewers Needed

We are looking for qualified people to review technical programs and hardware for The Computer Journal. We do not need reviews of Lotus 1 -2-3 or similar spreadsheets, wordprocessors. or general business type programs: we'll leave that to the general interest magazines What we do need are reviews of compilers, assemblers, disassemblers, debuggers, programming utility libraries, scientific and engineering programs, data acquisition and analysis programs, operating system enhancements. and similar items which are used by programmers.

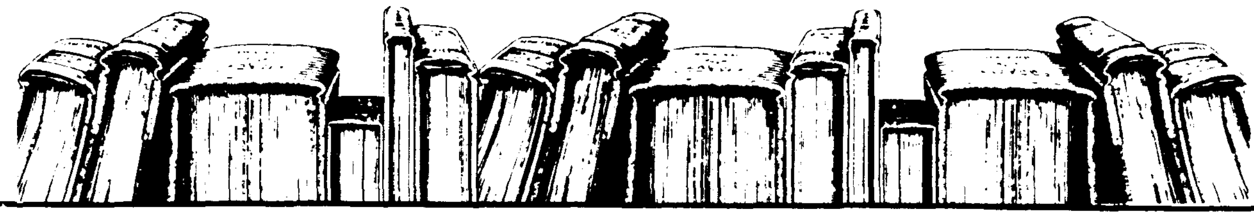
We are also interested in reviews of specialized hardware such as A/D and D/A interfaces. EPROM programmers, stepper motor controllers, and kits—but not most new computers or peripherals, unless there is some technical aspect of special interest to our readers.

We prefer reviews from people who are actually using the product rather than from someone who reviews many different products without using any one of them long enough to become completely familiar with all of its features. The reviews should be truthful and should tell it like it is, but the best reviews are the ones you write about products that you like and want to encourage others to use

It you are interested in writing reviews, send us a short letter with your background and qualifications, and a phone number where you can be reached in the evening. Include products which you now have available for review. and also items which you would be interested in reviewing if we could obtain a review copy ■.

```
328 GOTO 268
338 PRINT'CDMNENTER: (0) FOR HIRES (BLACK/WHITE)*
348 PRINT"CDWN) OR <1) FOR MULTI (4 COLOR LEVELS)"
358 GETK»: 1FK«<>"" THEN 358
368 PRINT"<DWN)              ";I INPUT MODE:IF MODE<8 OR MODEM
    THEN 336
378 PRINT" <CLRMDWNMRON>WHI LE SCANNING, PRESS:"
388 PRINT"CDWN)              <T)   TO START AT TOP"
398 PRINT" <DWN)             <S>   TO SYNCHRONIZE & RESTART"
488 IF   MODE>8THEN  PRINT"<DWN>      <C>     TO ROTATE COLORS"
410 IF   MODE>8THEN   PRINT*DWN)      <0-9)   TO CHANGE COLORS"
428 IF   MODE>8THEN    PRINT*<DN)     <F1-F7) TO CHANGE A
    COLOR"
438 PRINT'(DMN)              <Q) TO QUIT"
448 PRINT" <DWNXDWNXDWNXRON>NOW START THE FAX MACHINE."
458 FOR I-1T018688: NEXT I
466 SYS 18195: REM CLEAR 8K BIT-MAP
476 POKE 18177, MODE
488 POKE C1,C(CR,1): POKE C2,C(CR,2): POKE C3,C(CR,3)
498 IF KO2 THEN POKE Cl,CA: POKE C2,CB: POKE C3,CC
588 SYS 18285: REM SET COLORS
510 FOR 1" 18168TO18176: POKEI.0: NEXTI
520 POKE 53272,(PEEK<53272) OR 8)
536 POKE 53265,(PEEK<53265) OR 32)
548 IF MODE-0 THEN POKE 53270,8
550 IF MODE-1 THEN POKE 53270,24
560 POKE 792,8: POKE 793,69
578 POKE 56591,0: POKE 56579,0: POKE 56589,127
580 POKE 56582,84: POKE 56583,1: POKE 56591,23: POKE 56589,
    130
590 REM NOW THE NMI ROUTINE HAS STARTED
600 GET K$: IF KSO"" THEN 600
610 GET Ks: IF Ks=** THEN 610
620 IF K*-"Q" OR K»="E" THEN SYS 18254: GOSUB 860: GOTO
    188
630 IF MODE<1 THEN 600
640 IF ASC<KO>47 AND ASC(K$)(58 THEN KC-1 : GOTO 680
650 IF KS="C* THEN KC-2: GOTO 710
660 IF Ks="< Fl>" OR Ks="< F3>" OR Ks="( F5>" OR KS="<
    F7>" THEN KC-3: GOTO 740
670 GOTO 600
680 CR=VAL(K$)
690 POKE Cl,C(CR,l): POKE C2,C(CR,2): POKE C3,C<CR,3)
700 GOTO 850
710 CR—CR+1: IF CR>9 THEN CR-0
720 POKE Cl,C(CR,l): POKE C2,C(CR,2): POKE C3,C(CR,3>
730 GOTO 858
740 IF KSQ*C F1>" THEN 770
750 CA—PEEK(C1)+1: IF CA>15 THEN CA-0
760 POKE C1,CA: GOTO 850
770 IF KX>"< F3>" THEN 800
780 CB—<PEEK(C2)AND240)/16+1: IF CB>15 THEN CB-0
798 POKE C2,(CB*16)+(PEEK(C2)AND15) : CB=PEEK<C2): GOTO
    850
800 IF K»<>"< F5>" THEN 830
810 CB—<PEEK(C2)AND15>+1: IF CB)15 THEN CB-0
820 POKE C2,<PEEK<C2)AND248)+CB: CB-PEEK<C2): GOTO 850
830 CC—PEEK(C3) + 1: IF CO 15 THEN CC-0
840 POKE C3,CC: GOTO 850
858 SYS 18205: GOTO 608
860 REM RESET SCREEN TO ALPHA
870 POKE 53265,<PEEK(53265) AND 223)
880 POKE 53270,8: POKE 53272,21
898 SYS 64931: SYS 64789
988 SYS 65371
918 GOSUB 1418
928 RETURN
938 PRINT"<DWNXRON>WHILE VIEWING, PRESS:"
948 IF   MODE>8THEN  PRINT"<DWN)      <C>     TO ROTATE COLORS"
956 IF   MODE>8THEN  PRINT" <DVN)     <8-9>   TO CHANGE COLORS"
968 IF   MODE>8THEN  PRINT"<DHN>      <F1-F7> TO CHANGE A
    COLOR"
978 PRINT*<DWN               <Q> TO QUIT"
988 FOR I-1TO4888: NEXTI
998 POKE 18177, MODE
1888 POKE Cl,C(CR,l): POKE C2,C(CR,2): POKE C3,C<CR,3)
1818 IF KO2 THEN POKE Cl.CA: POKE C2,CB: POKE C3,CC
```

continued on page 31



The Bookshelf

Soul of CP/M: Using and Modifying CP/M's Internal Features

Teaches you how to modify BIOS. use CP/M system calls in your own programs, and more! Excellent for those who have read *CP/M Primer or* who otherwise understand CP/M's outerlayer utilities. By Mitchell Waite. Approximately 160pages, 8x9', comb. ©1983 ....$18.96

The Programmer's CP/M Handbook

An exhaustive coverage of CPM-80* . its internal structure and major components is presented. Written for the programmer. this volume includes subroutine examples for each of the CP/M system calls and information on how to customize CP/M - complete with detailed source codes for all examples. A dozen utility programs are shown with heavily annotated C-language source codes. An invaluable and comprehensive tool for the serious programmer. By Andy Johnson-Laird. 750 pages. 7'/2x9', softbound....................$21.95

Interfacing to S-100 (IEEE 696) Microcomputers

This book is a must if you want to design a custom interlace between an S-100 microcomputer and almost any type of peripheral device. Mechanical and electrical design is covered, along with logical and electrical relationships, bus interconnections and more. By Sol Libes and Mark Garetz. 322 pages. 61x9', softbound....................$16.95

Microprocessors for Measurement and Control

You'll learn to design mechanical and process equipment using microprocessor based "real time" computer systems. This book presents plans for prototype systems which allow even those unfamiliar with machine or assembly language to initiate projects. By D.M. Ausiander and P. Sagues. 310 pages, 7 3/8x9 1/4. softbound....................$18.96

Understanding Digital Logic Circuits

A working handbook for service technicians and others who need to know more about digital electronics in radio, television, audio, or related areas of electronic troubleshooting and repair. You're given an overview of the anatomy of digital-logic diagrams and introduced to the many commercial IC packages on the market. By Robert G. Middleton. 392 pages. 51/1x8% , softbound....................$18.95.

Real Time Programming: Neglected Topics

This book presents an original approach to the terms, skills. and standard hardware devices needed to connect a computer to numerous peripheral devices. It distills technical knowledge used by hobbyists and computer scientists alike to useable. comprehensible methods. It explains such computer and electronics concepts as simple and hierarchical interrupts, ports, PIAs. timers, converters, the sampling theorem, digital filters. closed loop control systems, multiplexing, buses, communication, and distributed computer systems. By Caxton C. Foster, 190 pages, 6*x9'4. softbound....................$9.95

Interfacing Microcomputers to the Real World

Here is a complete guide for using a microcomputer to computerize the home, office, or laboratory. It shows how to design and build the interfaces necessary to connect a microcomputer to real world devices. With this book, microcomputers can be programmed to provide fast, accurate monitoring and control of virtually all electronic functions - from controlling houselights. thermostats, sensors, and switches, to operating motors, keyboards, and displays. This book is based on both the hardware and software principles of the Z80 microprocessor (found in several minicomputers. Tandy Corporation's famous TRS-80. and others). By Murray Sargent Ill and Richard Shoemaker. 288 pages. 6'4x9', softbound....................$15.55

Mastering CP/M

Now you can use CP/M to do more than just copy files. For CP/M users or systems programmers — this book takes up where our CP/M handbook leaves off. It will give you an in-depth understanding of the CP/M modules such as. CCP (Console Command Processor), BIOS (Basic Input/Output System), and BDOS (Basic Disk Operating System). Find out how to: incorporate additional peripherals with your system, use console 1,0. use the file control block and much more. This book includes a specal feature —a library of useful macros. A comprehensive set of appendices is included as a practical reference tool. Take advantage of the versatility of your operating system! By Alan R. Miller. 398 pages. 6*x9'. softbound....................$18.95

FORTH Tools, Volume One

FORTH Tools is a comprehensive introduction to the new international FORTH -83 Standard and all its extensions. It gives careful treatment to the CREATE - DOES construct, which is used to extend the language through new classes of intelligent data structures. FORTH Tools gives the reader an in-depth view of input and output, from reading the input stream to writing a simple mailing list program. Each topic is presented with practical examples and numerous illustrations. Problems (and solutions) are provided at the end of each chapter. FORTH Tools is the required textbook for the UCLA and IC Berkeley extension courses on FORTH. By Anita Anderson and Martin Tracy, 218 pages. 5'18'. softbound....................$20.00

TTL Cookbook

Popular Sama author Dan Lancaster gives you a complete look at TTL logic cireuits, the most inexpensive, most widely applicable form of electronic logic. In no-nonsense language, he spells out just what TTL is, bow it works, and bow you can use it. Many practical TTL applications are examined, including digital counters, eleetronic stopwatches, digital voltmeters. and digital tachometers. By Don Lancaster. 338 pages. 6'z8',soft. ©1974....................$12.96

The Computer Journal

PO Box 1697 Kalispell, MT 59903

Order Date:__________

Print Name__________

Address__________

City State Zip__________

□ Check OMastercard | Visa □

Card No. Expires__________

Signature for Charge__________

Qty	Title	Price	Total

Shipping charges are: $1.00 for the first book, and $.50 for all subsequent books. Please allow 4 weeks for delivery.

Book Total

Shipping

TOTAL

THE COMPUTER CORNER

A Column by Bill Kibler

Well, so starts a new year and a new column. In the past year of writing for *The Computer Journal,* I have passed over many little topics and interesting tidbits. My recent articles on "tricks of the trade" were attempts to cover some of these topics, yet far too many words of wisdom never make it into print. Considering too, the many questions I receive from fellow computerists, there seems to be a need for a regular column.

As a contributing editor of *The Computer Journal* it will be my duty to answer your letters of inquiry when possible, as well as those I receive in some of my other activites. This will not stop the major articles on other topics but will allow you to see how I am doing with a project, and in fact, give you a chance to comment on it before completion. A major problem I face in writing articles about hardware is the long development time needed to prepare an article. I am currently working on a series of articles based around the Superbrain computer. Just gathering and sifting the data has taken several months. It is now possible to actually sit down and get started on writing the articles and making the changes I have in mind.

My primary work is with industrial computers, mainly those controlling process systems. Although these units are different than the normal personal computer, there are many facts and concepts that I come across which will be of interest to readers. One such area is the use of small systems running Forth for control applications. I am currently toying with building the Rockwell Forth system or making a Z80 Forth unit. In either case I am interested in others' experiences and in your comments. Having the Rockwell unit in my hands for one night was fun, but at the time I did not have all the documents I needed to do a write up on it.

Being somewhat of a purist, it has taken me some time to give up my 8' drives. Still, the price of new 5'A ' drives has dropped so much that I must admit to shifting over to them. Now don't get me wrong —I still have an 8" disk system — but now most of my work is on minis. Is this trend important? I think so. Why? There are many things happening with computers these days, and most of them are not technical. The hardware is becoming the least important aspect of the system, and software is definitively the new challenge on the horizon. This change means a lack of available supplies for older systems. The industry is going where the most money is to be made, and the money is in minis, not maxis. It is getting harder to find 8' diskettes for under two dollars, but I can get 5's for a dollar even.

My Z100 computer has found a new home, and Gerry, the new owner, is finding out about all the little points I never had time to investigate. One complaint of mine was the absence of a configuration program. The problem concerns the ever-increasing size of BIOSs. These started out under 2K in length and now run several "K" long. The Z 100's BIOS is in two parts under CP/M 85, and the MSDOS is several inches thick. When adding 8' drives or changing to non-standard units it will be necessary to patch the BIOSs. In the old days this was no problem, but now the Z100 is a nightmare and a half. The problem is not one of bringing out the control values into accessible tables. The sign of a good BIOS is that all of the parameters are located in one place, making patching and configuration programs possible. Users with ZDOS 1.0 will be pleased to see ZDOS 2.0 with a configuration program.

When dealing with different systems, I guess the most common problem for me is that of transferring data. My solution is Modem? and its file transferring options. Running two systems and doing my work on 5's and then shipping it out on 8's has me using Modem? all the time. When I was working at Micropro we had a program for our development systems that allowed one unit to be a slave to the other. Similar to BYE, this program caused the slave drives to become "C" and "D" drives. If somebody has written such a program, please let me know, as I haven't found a copy of the old one. Reinventing this program for generic CP/M systems is my next project, so let me know if you have any ideas on just such a program.

I spent the other day reading about CPNET and got some ideas on the transfer program. Seems Digital Research uses the BDOS calls to control their headers in packetting the data to transfer. This has got me thinking of doing some pushing of registers to create the data packet, and then just popping them and calling the BDOS entry point. This sounds simple until you sit down and start writing the code, but now I have a point to start from. There is also a HAM radio packet program on SIG/M disks that may shed more light on the subject. As I study the problem more, it appears that getting the data packet or format is the part that can cause the most problems.

The new year is here and with it the return of the swap meets. I went to my first one of the year last weekend, and was rather surprised at the change of products. Prices are down as many companies are going under and unloading their warehouses. Another change I've seen is the absence of S-100 boards, or at least a change in their quantity. In the past, S-100 was the most common product at swap meets, but single boards and hard disks are now taking up most of the spaces. After the weekend meet, I need to change my statement that it is possible to build a system for under $800 -1 think it is less than $500 now.

Well that's about it for this month. Next month should contain reports on tying systems together, some $80 minis, and what it is like being the editor of a local computer club newsletter. ■

Interfacing Tips and Troubles

A Column by Neil Bungard I

This month I would like to diverge from my series of articles on interfacing tools to stress a point concerning *The Computer Journal,* and to show you an easier way to interface your Sinclair ZX81 computer.

There is so much information being generated in the area of computers that it is impossible for one person to keep up with it all. As an example, consider my recent articles on interfacing the Sinclair computers *(The Computer Journal,* Issues 13 and 14). In part one of this series I made the following statement: "The Sinclair machines do not support MMIO (Memory Mapped 10)." I made this statement because while investigating the capabilities of the Sinclair machines, I was unable to make the machines respond using MMIO. However, as a result of a letter from LED of Michigan (Issue 14), I must "happily" retract my statement concerning MMIO on the Sinclair ZX81.1 say "happily" because using MMIO simplifies the task of interfacing the ZX81 considerably. The interfacing task is simplified because there are no machine language routines required, and the hardware problems associated with AIO do not occur when using MMIO. The only disadvantage of MMIO is that it is slower than AIO on the ZX81, but it is my experience that the speed limitations are not a problem in most applications.

The hardware trick for using MMIO, as explained in LED's letter, is to direct the MMIO operations into the ZX81's memory space between addresses 8192(D) and 16383(D) (the (D) denotes decimal values). In addition, when information is transferred to/from this memory space, a signal (logic 1) must be generated and placed on the ZX81's ROMCS edge connector (pin 23B). With these details taken care of, information can be transferred to/from an interface circuit using PEEK and POKE instructions directly from BASIC. Using MMIO eliminates the need to write BASIC routines to load machine language programs, allocate space for machine language routines in REM statements, determine how data will be passed from the machine language programs to BASIC, and work around crashes and masked bits, all of which were required when using AIO on the ZX81.

MMIO Hardware

The circuit required to accomplish MMIO with the ZX81 is shown in Figure 1. Address lines A13, A14, A15, and 3 gates (two "OR" gates and 1 "INVERTER") from ICs 1 and 5 are required to decode the memory space which is used for MMIO on the ZX81. The 3 gates from ICs 1 and 5 configure a decoder which outputs a logic 0 on pin 6 of IC 1 any time memory locations 8192 through 16383 are addressed. Pin 6 of IC1 is connected to the output enable (pin 1) of tristate buffer IC6.

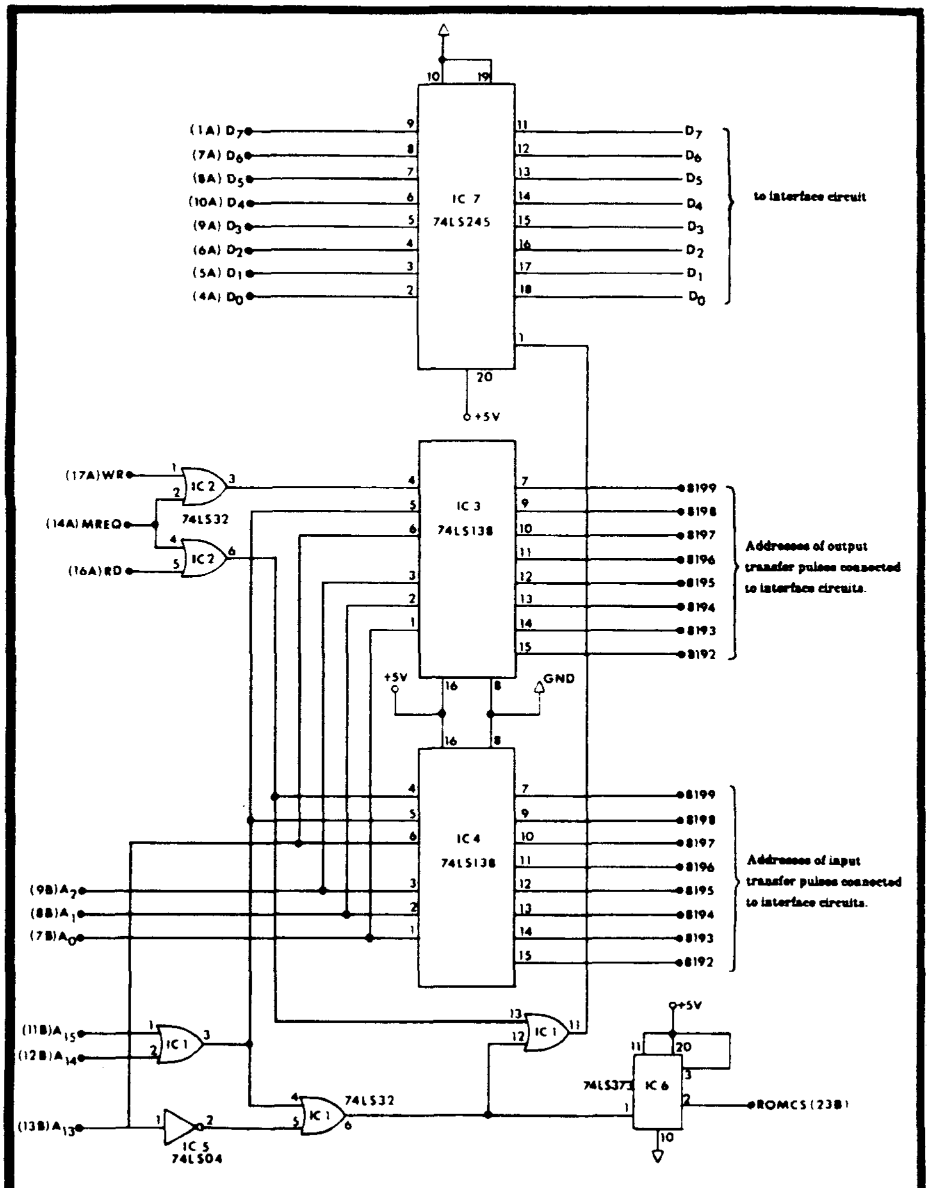


Figure 1: Schematic. Numbers in parenthesis denote pin number on ZX81 edge connector.

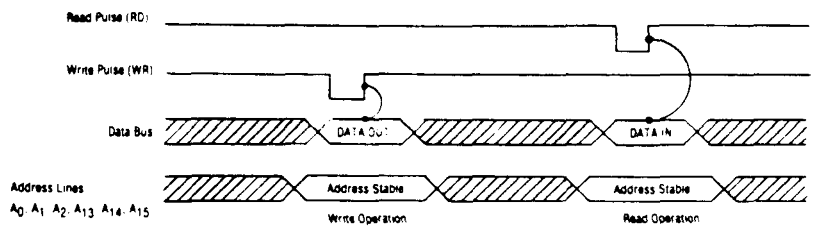


Figure 2: Timing Diagram of MMIO operations.

When pin 1 of IC6 is at a logic 0, the tristate buffer input (a logic 1 at pin 3) is connected through the buffer's output (pin 2) to the ZX81's ROMCS input. A logic 1 on the ROMCS input disables the ZX81's internal ROM, thus allowing MMIO operations to be accomplished into the 8192(D) to 16383(D) memory space.

Two 74LS138s (IC3 and IC4) generate 8 input and 8 output transfer pulses by decoding the ZX81's 3 lowest order address lines AO, Al, and A2. Address lines AO, Al, and A2 are connected to pins 1,2, and 3 respectively of the 74LS138s and pins 5 and 6 of the 74LS138s additionally decode the address bus by detecting when the 8192(D) to 16383(D) memory space is being accessed. Timing of the transfer pulses is accomplished via a memory write (WR) signal and a memory read (RD) signal connected to pins 4 of ICs 3 and 4 respectively. Figure 2 shows the timing diagrams of the MMIO operations.

Information flows into (and out of) the ZX81 via an octal bus transceiver, IC7 in Figure 1. If address space 8192(D) through 16383(D) is accessed, and a memory read operation is being performed, pin l of IC7 will be at a logic 0, allowing data to be transferred into the ZX81. If address space 8192(D) through 16383(D) is accessed, and a memory write operation is being performed, pin 1 of IC7 will be at a logic 1, allowing data to be transferred out of the ZX81. The purpose of this octal bus transceiver is twofold. First of all, the transceiver isolates the ZX81 from the interface circuit, which would save the ZX81's internal circuitry if something went wrong on the interface circuit. Secondly, the transceiver will boost the ZX81's fanout. This means that more devices can be placed on the ZX81 data bus without loading the bus and causing current deficit problems.

MMIO Software

As mentioned earlier in this article, all information transfer between the ZX81 and an interface circuit can be accomplished from the BASIC language set using PEEK and POKE instructions. To accomplish MMIO using BASIC, you must first know where, within the 8192(D) to 16383(D) memory space, the interface circuit is actually mapped. When using the hardware configuration explained above, the 8 memory locations between 8192(D) and 16383(D) are used for input/output. Figure 3 shows which output pin on the 74LS138s will supply the correct transfer pulse when each of the 8 memory addresses are accessed. The software instructions which accomplish the MMIO are straightforward. To input data from an interface mapped into memory location 8192(D), you would use the following instruction:

LET APEEK (8192)

This instruction assigns the value obtained from the interface circuit (which will be a value between 0 and 255) to the variable name A. Likewise, to output data to an interface mapped into memory location 8192(D), you would use the following instruction:

POKE 8192,A

This instruction transfers the value previously assigned to the variable name A (a value between 0 and 255) to the interface circuit.

Input AddressPin W on IC3
PEEK 819215
PEEK 819314
PEEK 819413
PEEK 819512
PEEK 819611
PEEK 819710
PEEK 81989
PEEK 81997

Output AddressPin W on IC4
POKE 819215
POKE 819314
POKE 819413
POKE 819512
POKE 819611
POKE 819710
POKE 81989
POKE 81997

Figure 3

Conclusion

In conclusion, with memory mapped I/O and accumulator I/O now explained, the Sinclair ZX81 can be a very versatile computer for interfacing. AIO has its place where speed is a critical factor, as in applications where a number of values must be obtained in a second or less. But if your application requires acquisition times on the order of seconds or even greater, then MMIO offers you the simplicity to get your system working quickly and easily. As always, we appreciate your response to articles in *The Computer Journal* and look forward to hearing from you if you have questions or comments. ■

Classified

The Computer Journal will carry Classified Ads. The rate is $.25 per word. All Classified Ads must be paid in advance, and will be published in the next available issue. No checking copies or proofs are supplied, so please type your ad or print legibly.

ANNOUNCEMENTS

Artificial Intelligence Conference

The premiere Artificial Intelligence and Advanced Computer Technology Conference/Exhibition is scheduled for April 30, May 1 and 2,1985, at the Long Beach Convention Center, Long Beach, California.

The exhibition showcases commercial and industrial applications of advanced computers and software. Technical experts will present a conference focusing on AI in automated manufacturing, office automation, medicine, robotics, business, training, microcomputers, aerospace, and graphic simulation. Other topics will be: fifth generation computers, natural language interfaces, expert systems-development systems, speech recognition, image processing, cognitive modeling, knowledge information processing, and AI languages including LISP and PROLOG.

Compete details are available from Tower Conference Management Co., 331 W. Wesley St., Wheaton, IL 60187, phone (312) 668-8100. ■

Computer Interfacing Workshop

Virginia Tech has announced a workshop on Personal Computer and STD Computer Interfacing for Scientific Instrument Automation. These courses, directed by David E. Larsen and Dr. Paul E. Field, will be held August 22,23, and 24 in the Washington DC area, and September 19,20, and 21 in Greensboro, NC. The cost is $450 for the three day session, and details can be obtained from Dr. Linda Leffel, C.E.C., Virginia Polytechnic Institute, Blacksburg, VA 24061. phone (703) 961-4848. ■

Universal RS-232 Data Acquisition

Elexor has announced their PL-100 intelligent peripheral which interfaces with any computer or terminal via a standard RS-232 serial port. It has 16 channels of 12 bit A/D, 2 channels of 12 bit D/A, 32 bits of digital I/O, 8K of ROM and 8K of RAM, plus provision for internal rechargeable batteries and two additional I/O boards. An on-board microprocessor supports simple ASCII commands or internal BASIC interpreter. In addition, internal intelligence makes remote unattended applications possible using only a modem (no computer necessary).

Prices start at $549, and more information is available from Elexor Associates, PO Box 246, Morris Plains, NJ 07950, phone (201) 299-1615. ■

DSD80 Debugger

Soft Advances announces DSD80, a full screen symbolic debugging program for 8080,8085 and Z80 microcomputers running CP/M-80 and compatible operating systems. The dynamic display has instruction, register, stack and two memory windows. The Z80 instruction set is fully supported using either extended Intel or Zilog mnemonics. DSD80 has on-line help and comes with a fifty page user's manual. The price is $125 plus shipping from Soft Advances, PO Box 49473, Austin, TX 78765, phone (512) 478-4763. ■

IBM-PC Data Acquisition Software

Data Translation has announced a series of application software packages to support its IBM-PC compatible data acquisition and control boards. These packages, intended for such applications as chromatography, physiological and speech research, materials testing, and industrial control, do not require the user to write original programs.

DT/Notebook is an integrated, menu driven software package for real time data acquisition, process control, data analysis, and graphic display. It performs data acquisition at up to 20,000 samples per second and real time graphic display of data at up to 600 samples per second.

DT/ILS-PC 1 is an interactive, command driven digital signal processing package which supports continuous data acquisition to disk at up to 27,500 samples per second.

ASYST is a command driven package for real time data acquisition and control, data analysis, and graphic displays able to acquire data at up to 27,500 samples per second. More information on these products and their analog I/O boards can be obtained from Shari L. Supernault at Data Translation, 100 Locke Drive, Marlboro, MA 01752, phone (617) 481-3700. ■

FAX-64 Listing 3, continued from page 25

```
1020 SYS 18205: REM SET COLORS
1030 POKE 53272, (PEEK(53272) OR 8)
1040 POKE 53265,(PEEK<53265) OR 32)
1050 IF MODE=0 THEN POKE 53270,8
1060 IF MODE-1 THEN POKE 53270,24
1070 POKE 56591,0: POKE 56579,0: POKE 56589,127
1080 GOTO 600
1090 PRINT*<CLR>CDWNCDWN)CDWN<DWN) CRONDENTER A NEW
    NAME FOR FILECROF)*
1100 FL=14-MODEX6: PRINT" " FL" CHARACTERS MAXIMUM. "
1110 FS="NONAME* : INPUTCDWN) "jF*
1120 IF MODE>0 THEN: IF LENCFSX8 THEN Fs=F$+" ": GOTO
    1 120
1130 IF LEN(FS)FL THEN F$=LEFT$(F$,FL)
1140 IF MODE-0 THEN FS="DD*+FS
1150 IF MODE-1 THEN F*=CHR»(129)+"PIC X "+F*
1160 CLOSE15: OPEN 15,8,15
1170 PRINT"<DrtM> <RON)PUT DISK IN DRIVE 0, PRESS A
    KEY"
1180 GET Ks: IF Ks="* THEN 1180
1190 IF K»—"Q" THEN 1290
1200 PRINTS 15,'10": GOSUB 1300: IF EC 1 THEN 1250
1210 PRINT*<DWN) CRON) PUT IN DISK NOW AND PRESS A KEY*
1220 GET Ks: IF K*-"" THEN 1220
1230 IF K»—"Q" THEN 1290
1240 GOTO 1200
1250 LL-LEN(F$): POKE18178,LL
1260 FOR IL—1TOLL: POKE 18178+IL, ASCCMIDS( F»,IL,1)):
    NEXT IL
1270 SYS 18388: REM SAVE TO DISK FILE
1280 SC-0
1290 CLOSE7: CLOSE 15: GOTO 180
1300 REM ERROR CHECK
1310 INPUT#15,E,E*,E2,E3
1320 IFETHENPRINT"                ****E$*****
1330 RETURN
1340 IF SC>0 THEN 1360
1350 POKE52.160: POKE56.160: POKE644.160: CLR: END
1360 PRINT"<DWND<RON>YOU HAVE NOT SAVED THE LAST PICTURE."
1370 PRINT" <DWMXRON> DO YOU WANT TO QUIT?": KS=N"
1380 PRINT" <DWNENTER <Q> TO QUIT " ; : INPUT K»
1390 IF K»-"Q" THEN 1350
1400 GOTO 180
1410 PRINT*<CLR>CBLU><DWN>(DN>(DWN)(DN)*: POKE 53280,
    7: POKE 53281,1: RETURN
READY.
```

Back Issues Available:

Ordering Information: Back issues of The Computer Journal are $3.25 in the U.S. Canada, and $5.50 in other countries (air mail postage included). Send your complete name and address with payment to The Computer Journal, PO Box 1697, Kalispell, MT 59903. Please allow three to four weeks for delivery.

Volume 1, Number 1 (issue *1):
- *The RS-232-C Serial Interface, Part One*
- *Telecomputing with the Apple]]: Transferring Binary Files*
- *Beginner's Column, Part One: Getting Started*
- *Build an "Epram"*

Volume 1, Number 4 (Issue #4):
- *Optoelectronics, Part One: Detecting, Generating, and Using Light in Electronics*
- *Multi-user: An Introduction*
- *Making the CP/M User Function More Useful*
- *Build a Hardware Print Spooler, Part Three: Enhancements*
- *Beginner's Colimn, Part Three: Power Supply Design*

Volume 2, Number 1 (Issue *5):
- *Optoelectronics, Part Two: Practical Applications*
- *Multi-user: Multi-Processor Systems*
- *True RMS Measurements*
- *Gemlnl-10X: Modifications to Allow both Serial and Parallel Operation*

Volume 2, Number 2 (Issue #6):
- *Build a High Resolution S-100 Graphics Board, Part One: Video Displays*
- *System Integration, Part One: Selecting System Components*
- *Optoelectronics, Part Three: Fiber Optics*
- *Controlling DC Motors*
- *Multi-User: Local Area Networks*
- *DC Motor Applications*

Volume 2, Number 3 (issue #7):
- *Heuristic Search In HI-Q*
- *Build a High-Resolution S-100 Graphics Board, Part Two: Theory of Operation*
- *Multi-user: Etherseries*
- Sys fem *Integration, Part Two: Disk Controllers and CPIM 2.2 System Generation*

Volume 2, Number 4 (Issue *8):
- *Build a VIC-20 EPROM Programmer*
- *Multi-user: CP/Net*
- *Build a High-Resolution S-100 Graphics Board, Part Three: Construction*
- *System Integration, Part Three: CPIM 3.0*
- *Linear Optimization with Micros*
- *LSTTL Reference Chart*

Volume 2, Number 5 (Issue #9):
- *Threaded Interpretive Language, Part One: Introduction and Elementary Routines*
- *Interfacing Tips and Troubles: DC to DC Converters*
- *Multi-user: C-NET*
- *Reading PCDOS Diskettes with the Morrow Micro Decision*
- *LSTTL Reference Chart*
- *DOS Wars*
- *Build a Code Photoreader*

Volume 2, Number 6 (Issue #10):
- *The FORTH Language: A Learner's Perspective*
- *An Affordable Graphics Tablet for the Apple][*
- *Interfacing Tips and Troubles: Noise Problems, Part One*
- *LSTTL Reference Chart*
- *Multi-user: Some Generic Components and Techniques*
- *Write Your Own Threaded Language, Part Two: Input-Output Routines and Dictionary Management*
- *Make a Simple TTL Logic Tester*

Volume 2, Number 7 (Issue #11):
- *Putting the CPIM IOBYTE To Work*
- *Write Your Own Threaded Language, Part Three: Secondary Words*
- *Interfacing Tips and Troubles: Noise Problems, Part Two*
- *Build a 68008 CPU Board For the S-100 Bus*
- *Writing and Evaluating Documentation*
- *Electronic Dial Indicator: A Reader Design Project*

Volume 2, Number 8 (Issue *12):
- *Tricks of the Trade: Installing New I/O Drivers in a BIOS*
- *Write Your Own Threaded Language, Part Four: Conclusion*
- *Interfacing Tips and Troubles: Noise Problems, Part Three*
- *Multi-user: Cables and Topology*
- *LSTTL Reference Chart*

Volume 2, Number 9 (Issue *13):
- *Controlling the Apple Disk I Stepper Motor*
- *Interfacing Tips and Troubles: Interfacing the Sinclair Computers, Part One*
- *RPM vs ZCPR: A Comparison of Two CPIM Enhancements*
- *AC Circuit Anaysls on a Micro*
- *BASE: Part One in a Series on How to Design and Write Your Own Database*
- *Understanding System Design: CPU, Memory, and I/O*

Issue Number 14:
- *Hardware Tricks*
- *Controlling the Hayes Micromodem II From Assembly Language -*
- *S-1008 to 16 Bit RAM Conversion*
- *Time-Frequency Domain Analysis*
- *BASE: Part Two*
- *Interfacing Tips and Troubles: Interfacing the Sinclair Computers, Part Two*

Issue Number 15:
- *Interfacing the 6522 to the Apple I and Yé*
- *Interfacing Tips and Troubles: Building* a Poor-Man's *Logic Analyzer*
- *Controlling the Hayes Micromodem II From Assembly Language, Part Two*
- *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 and Troubles: Build a "Jellybean" Logic-to-RS232 Converter*

Issue Number 17:
- *Poor Man's Distributed Processing*
- *Base: Part Five*
- *FAX-64:Facsimlle Pictures* on a Micro
- *The Computer Corner*
- *Interfacing Tips and Troubles: Memory Mapped 110 on the ZX81*