ISSN # 0748-9331

THE COMPUTER JOURNAL®

For Those Who Interface, Build, and Apply Micros

Issue Number 15

$2.50 US

Editor's Page

Assembly Language Renaissance?

Choosing a language for the first micros was easy—you entered everything in hexadecimal because there wasn't anything else available! Hackers soon wrote assemblers, which were a great step forward because you could enter mnemonics which the assembler used to generate the hexadecimal code. They also wrote disassemblers which converted the hexadecimal code back to the mnemonics so that you could analyse and change existing programs and debuggers. This was followed by BASIC and FORTRAN, and these three languages were used to write the majority of the programs which made the microcomputer so useful.

As the micros became more popular, many additional languages intended to improve on the early entries were made available. These easier-to-use languages were welcomed by the new market of non-technical micro users who were scared to death of assembly, and it was predicted that assembly language was no longer needed because the new languages could do everything so much better.

Some programmers continued to use assembly language instead of, or in addition to, the higher level languages because the code was smaller, ran faster, and they could do anything that the system was capable of without the limitations imposed by the developer of the language. The proponents of the new languages said "So what if it takes an extra fifteen or twenty Kilobytes of code for the libraries and runtime packages? Memory is getting cheaper." They said "So what if the programs take five to ten times as long to run? CPUs are running faster." They said "So what if you can't read or write directly to the ports, the disk, or to memory? Nobody needs to do that anymore."

Most new programmers welcomed the higher level languages because they appeared to make things much easier for the programmer, but the resulting programs were not necessarily better for the user. There is a place for high level programs in applications where a small number of copies will be used and the speed is not important, but there are also instances where a program written in assembly language will perform much more satisfactorily. Even today the most successful commercial programs are written in assembly language, and several programs which were originally written in a high level language had to be rewritten in assembly language in order to match the performance of the competition. It is argued that the biggest advantage of high level language is the ease with which the programs can be ported over to other machines, but I cannot understand why poor operation on many machines is preferred over superior operation on one machine.

When choosing a language you have to consider the application. There is no one language that is best for every application, and I feel that anyone serious about working in the area of real world interfacing and control must be reasonably proficient in several different languages—and one of them has to be assembly. How can you customize your operating system or read out and modify a PROM without knowing at least a little about assembly language? There are also applications for other languages. For example, I am using the Apple® game port to monitor and control temperature with a short BASIC program. I used BASIC because it is easy to

continued on page 5

Editor/Publisher	*Art Carlson*
Art Director	*Joan Thompson*
Art Assistant	*Lois Cawrse*
Production Assistant	*Judie Overbeek*
Technical Editor	*Lance Rose*
Contributing Editor	*Ernie Brooner*

The Computer Journal® is published 12 times a year. Annual subscription is $24 in the U.S., $30 in Canada, and $48 airmail in other countries.

Entire contents copyright © 1984 by The Computer Journal.

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

Address all editorial, advertising and subscription inquires to: The Computer Journal, P.O. Box 1697, Kalispell, MT 59903-1697.

Interfacing the 6522 to the Apple][and][e

by John Bell Jr.

The Apple][® computer with its eight peripheral slots is one of the easiest computers to use for control applications. By adding one or more 6522 VIAs (Versatile Interface Adapters) you have a very powerful controller for such things as lab experiments or industrial control. The 6522 VIA has two 8-bit parallel I/O ports, and each port also has two handshake lines. Beside the two ports, this chip also has two 16-bit timers.

Interfacing VIAs, PIAs (Parallel Interface Adapters), and ACIAs directly to the Apple bus is not recommended by Apple because of timing errors. I will describe the error and show the solution to this problem.

Interfacing the 6522 VIA to the Apple][bus is very straight-forward except for the timing problem. First we are going to take care of the straight-forward interface parts, and then I will discuss the problem. Figure 2 is a block diagram of the 6522. First we have the eight data lines, DO THROUGH D7. These lines connect directly to the Apple peripheral connector D0 through D7, and are used to transfer data to and from the 6522. The 6522 has 16 internal registers. The desired register is selected using lines RS0 through RS3. These are the register select lines and they connect directly to the Apple bus address lines, A0 through A3. The R/W (read-write) line of the 6522 determines whether the data is coming to or from the 6502 processor in the Apple, and the R/W line of the 6522 connects directly to the reset line on the Apple bus. The reset line sets both parallel ports as input and also disables all interrupts to make sure that the computer starts up correctly. There are two chip select lines on the 6522; CS1 and CS2. The bar above CS2 indicates that it is a negative active signal. The Apple][peripherals connectors have two chip select or device select pins available. Pin 1, which is called the I/O select has 256 addresses set aside for each peripheral connector, and pin 41, which is called device select, has 16 addresses for each connector. Both pin 1 and pin 41 are negative active signals, so, depending on the

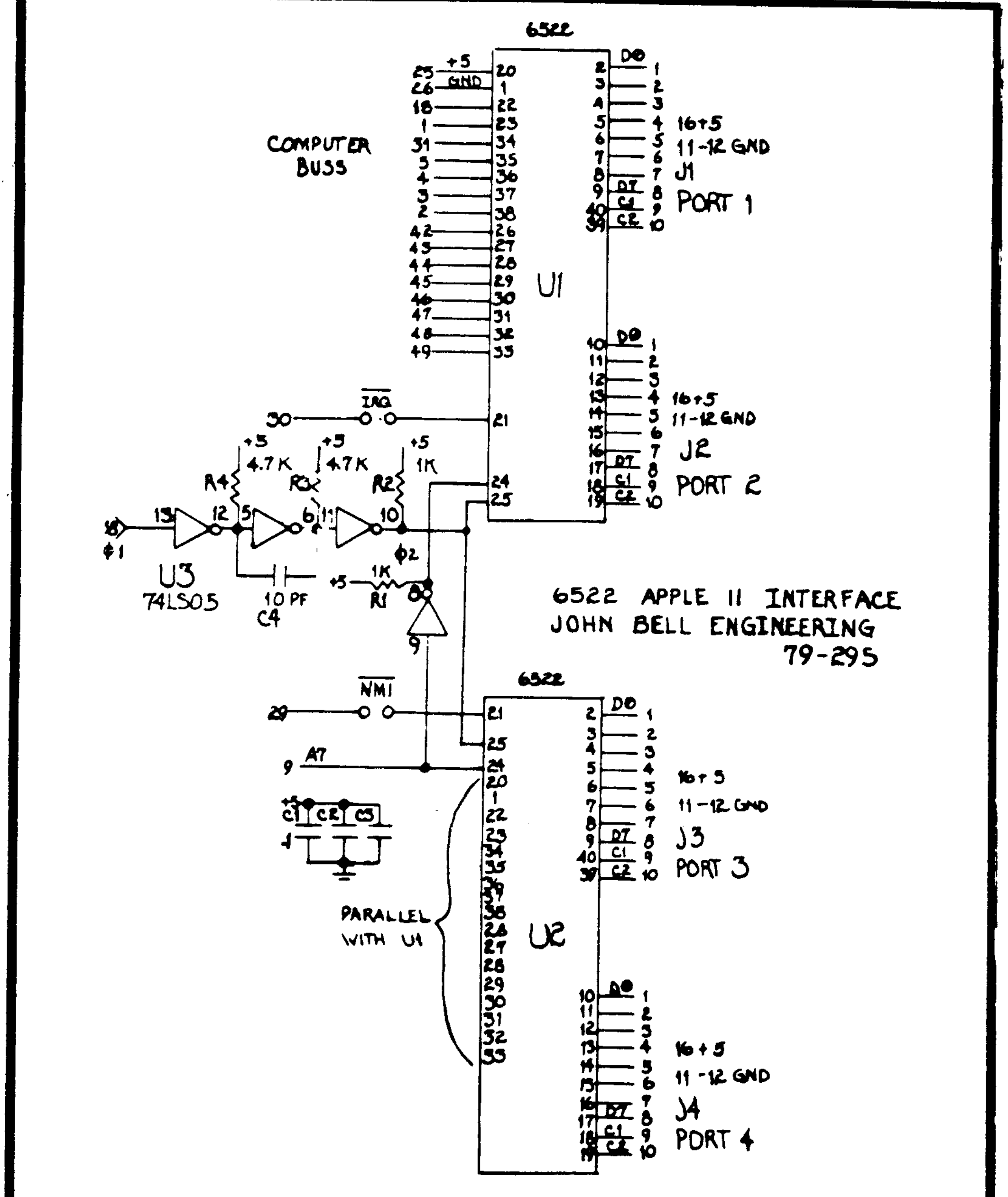


Figure 1

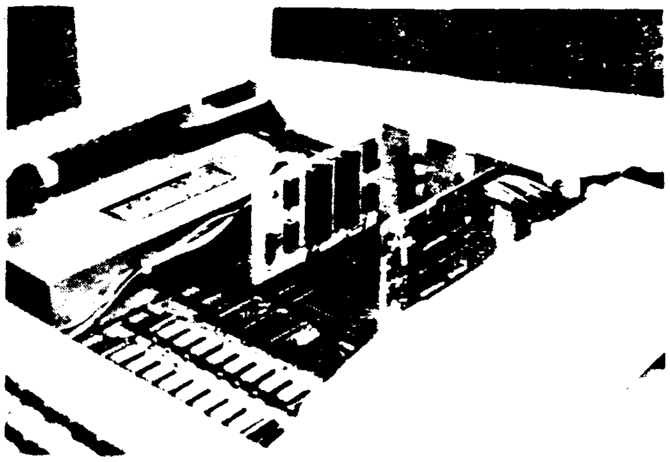
address you want, you would connect either pin 1 or pin 41 to CS2 on the 6522, and connect CS2 to the +5V power supply.

Now for the problem. We still have one more control signal to connect up. This is the phase two clock signal. All of the read and write functions of the 6502 processor are controlled by the phase two clock. However, the phase two clock signal from the 6502 processor in the Apple does not appear on the peripheral connector. The phase zero clock signal that is provided on the Apple][connector is the right polarity but occurs too early in the 6502 read-write cycles.

The solution. All that is needed is to recreate the phase two clock signal with the proper timing. By inverting the phase one clock and delaying both the rising and falling edges, we come up with a phase two clock. Figure 1 is a dual 6522 interface by John Bell Engineering. A 74LS05 is used to delay both the rising and falling edges of the phase one clock cycle and inverts it. This delayed inverted phase one clock circut can be used to interface other 6500 chips which require the phase two clock. This interface has four 8-bit parallel ports (32 lines). By the addition of proper external interface circuitry, (solid state switches and input protectors) this interface can be used to control robot arms, alarm systems for your home, lab experiments, and industrial machinery. If 32 I/O lines aren't enough, you can plug two circuit boards in and get 64 I/O lines, or plug four circuit boards in and get 128 lines.

In Conclusion

Since constructing this interface, I have found many uses for it such as a parallel centronics printer port and a real time clock, just to name a few. For more information on interfacing to the Apple][, Howard W. Sams and Co. Inc., publishes a book by Marvin L. DeJong called **Apple][Applications**. Also, a completely assembled and tested circuit board with two 6522s for interfacing to the Apple][or][e is available for $69.95 from John Bell Engineering, Inc., 400 Oxford Way, Bellmont, CA 94002. ■



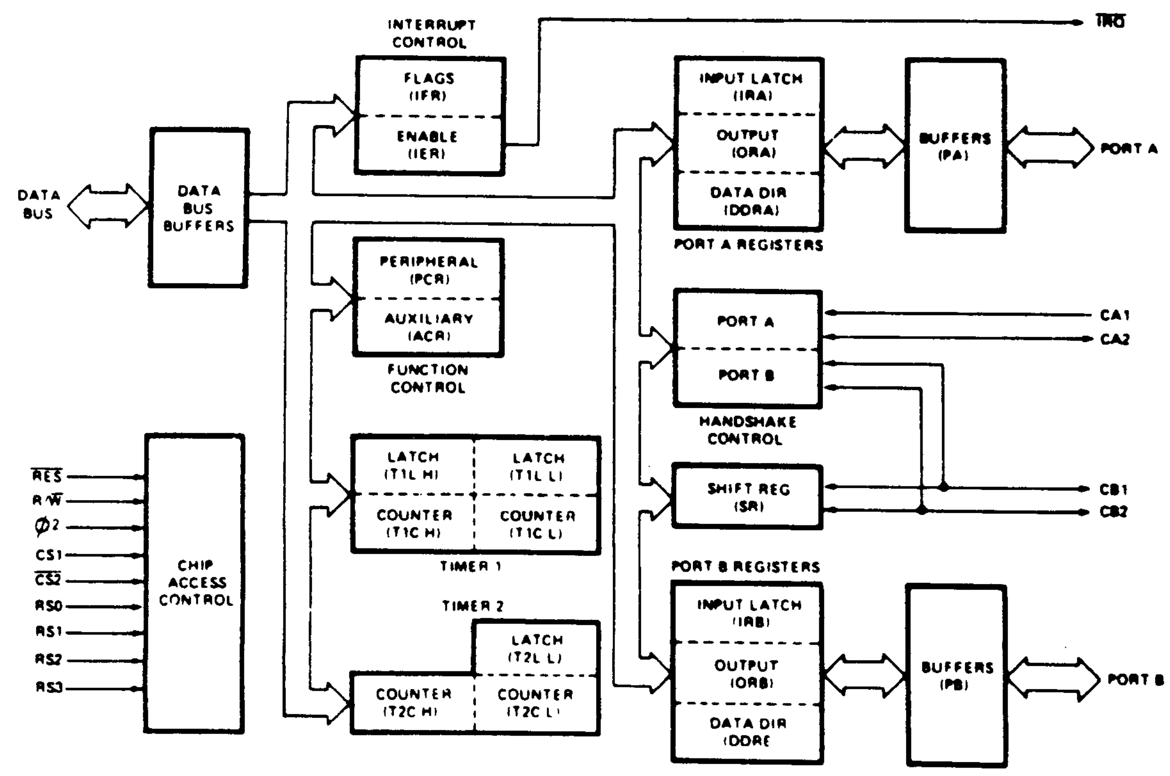


Figure 2: S46522 Block Diagram

CJ1

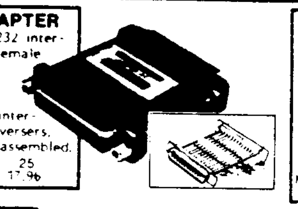
L-COM® Shopper's Guide

Products for the ELECTRONIC & COMPUTER Industry

RS-232 DATA JUMPER BOX ADAPTER

A Data Jumper Box used to customize RS-232 interface devices. All 25 pins of the Male and Female connectors terminate to 25 solder pads. The PC Board is already wired to the D-Subs. Supplied with 25 wires already stripped for permanent wiring to suit your needs. Many interfaces could be built - Null Modems, Pin Reversers, you name it! Supplied with all hardware unassembled.

MODEL	DESCRIPTION	1-9	10-24	25
DJB	Jumper Box	22.45	20.43	17.96

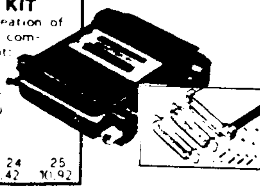


RS-232 DO IT YOURSELF KIT

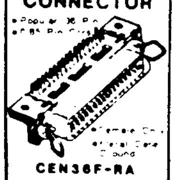
This Do It Yourself Kit allows the creation of many RS-232 variations. The following components are included in the assortment:

1ea.	Male 25 pin D-Sub Connector
1ea.	Female 25 pin D-Sub Connector
2ea.	Snap-on half covers (UL 94 VO)
1 set	Hardware for Male & Female Connectors

MODEL	DESCRIPTION	1-9	10-24	25
D-Y	Do It Yourself Kit	13.65	12.42	10.92



CENTRONICS RT ANGLE PC CONNECTOR



CEN36F-RA

1-9	10-24	25
8.95	7.88	8.50

You Need This!

RS-232 GENDER CHANGERS

Needed when RS-232 cables won't mate. Our DC Series are compact and feature trouble free PC Board construction and lower cost. Choice of shielded or unshielded. Keep some for emergency needs.

MODEL	GENDER	1-9	10-24	25
DC-25F	(F-F)Unshielded	16.95	15.42	13.56
DC-25M	(M-M)Unshielded	16.95	15.42	13.56
DCS-25F	(F-F)Shielded	18.65	16.97	14.92
DCS-25M	(M-M)Shielded	18.65	16.97	14.92

Rotary-Scale Clamp-On Instruments

INDISPENSABLE TOOL NEEDED BY ELECTRIC ELECTRONIC TECHNICIANS

MODEL SK7100

The most advanced Clamp-On Tester with many performance features • Range selection knob automatically advances the specific scale • Easy reading - no confusion • Accurate AC current measurements assured because of round clamp core with built in balanced coils • Most reliable taut band internal core magnet meter • Trigger lock device retains reading for use in hard to get places • Automatic Zero Adjust ohms scale • Overload protection up to 150 for one minute ON ALL RANGES • Test leads lock in for added safety • Accuracy ±3 full scale on all ranges • Accessories included: carry case, test leads and instructions • Size 8.25x3.25x1.4 • Reasonably priced

10 USEFUL RANGES	
AC Current	6 15 60 150 300 600A
AC Voltage	150 300 600V
Resistance	0-20kΩ (1kΩ center scale)

Price Cut 65%

57.95/ea	47.60/10
55.55/3	42.85/25

Reg. $119.00

THIS MONTH'S SPECIAL!

COAXIAL CABLES

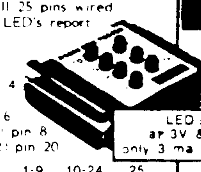
W/Molded **BNC** Connectors On Each End

MODEL ft	1-9	10-24	25-99
RG58C/U - for Instrumentation Use			
CC58C-1	3.55	3.23	3.05
CC58C-2.5	4.10	3.73	3.53
CC58C-5	4.65	4.23	4.00
CC58C-10	6.05	5.51	5.20
CC58C-15	7.45	6.78	6.41
CC58C-25	10.25	9.33	8.82
CC58C-50	17.25	15.70	14.84
CC58C-100	31.25	28.44	26.88
RG59B/U - for Video & General Use			
CC59B-1	3.65	3.32	3.14
CC59B-2.5	4.00	3.64	3.44
CC59B-5	4.75	4.32	4.09
CC59B-10	6.25	5.69	5.38
CC59B-15	7.75	7.05	6.67
CC59B-25	10.75	9.78	9.25
CC59B-50	18.25	16.61	15.70
CC59B-100	33.25	30.26	28.60
RG62A/U - for Computer Applications			
CC62A-1	3.60	3.28	3.10
CC62A-2.5	3.95	3.59	3.40
CC62A-5	4.70	4.28	4.04
CC62A-10	6.15	5.60	5.29
CC62A-15	7.60	6.92	6.54
CC62A-25	10.50	9.56	9.03
CC62A-50	17.75	16.15	15.27
CC62A-100	32.25	29.35	27.74
RG174U - for General Use			
CC174-1	3.60	3.28	3.10
CC174-1.5	3.75	3.41	3.23
CC174-2	3.90	3.55	3.35
CC174-2.5	4.05	3.69	3.48
CC174-4	4.50	4.10	3.87
CC174-5	4.80	4.37	4.13
CC174-10	6.35	5.78	5.46
CC174-15	7.90	7.19	6.79
CC174-25	11.00	10.01	9.46
CC174-30	12.55	11.42	10.79

RS-232 DATA LINE MONITOR

A miniature Data Line Monitor that allows the user to determine the status of the seven key signals of the RS-232 data path. All 25 pins wired through, and dedicated Red LED's report the status of:

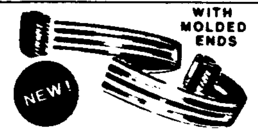
- Transmit Data (TD) pin 2
- Receive Data (RC) pin 3
- Request To Send (RTS) pin 4
- Clear To Send (CTS) pin 5
- Data Set Ready (DSR) pin 6
- Data Carrier Detect (DCD) pin 8
- Data Terminal Ready (DTR) pin 20



LED's glow at 3V & draw only 3 ma at 12V

MODEL	DESCRIPTION	1-9	10-24	25
DLM	Data Line Monitor	37.35	33.99	29.88

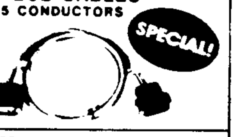
SUPERIOR GRADE DIP-16 FLAT CABLES



MODEL-ft	1-9	10-24	25
MLF16PP-0.5	3.30	3.00	2.84
MLF16PP-1	3.60	3.28	3.10
MLF16PP-2	4.20	3.82	3.61
MLF16PP-3	4.80	4.37	4.13
MLF16PP-4	5.40	4.91	4.64

RS-232 CABLES

25 CONDUCTORS

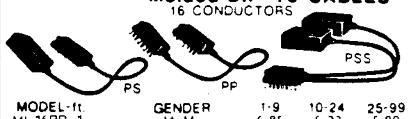


MODEL ft	1-9	10-24
C25-2.5	19.88	18.09
C25-5	21.16	19.26
C25-10	23.76	21.62
C25-15	26.36	23.99
C25-25	31.56	28.72

SPECIFY: Male-Male or Male-Fem

Molded DIP-16 CABLES

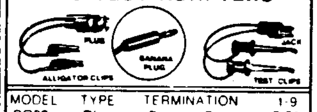
16 CONDUCTORS



MODEL-ft.	GENDER	1-9	10-24	25-99
ML16PP-1	M-M	6.85	6.23	5.89
ML16PP-2	M-M	7.15	6.51	6.15
ML16PP-3	M-M	7.45	6.78	6.41
ML16PS-1.5	M-F	6.95	6.32	5.98
ML16PSS-1.5	M-(2)F	11.45	10.42	9.85
ML16PSC-4*	M-F	8.95	8.14	7.70

*Coiled Cord 22" Unstretched

BNC TEST ADAPTERS



MODEL	TYPE	TERMINATION	1-9
BC30	Plug	Banana Plug	7.75
BC40	Plug	Test Clip	8.95
BC50	Plug	Alligator	7.10
BC60	Jack	Banana Plug	5.75
BC70	Jack	Test Clip	6.95
BC80	Jack	Alligator	5.75

Order 10 each type - DEDUCT 10%

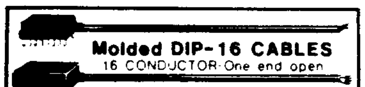
GENDER CHANGERS

MODEL	GENDER	1-9	10-24
MG-25F	F-F	23.95	21.79
MG-25M	M-M	23.95	21.79

ATTRACTIVE Prices for QTY Users

Molded DIP-16 CABLES

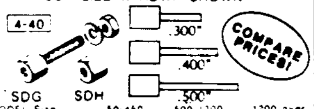
16 CONDUCTOR-One end open



MODEL-ft.	GENDER	1-9	10-24	25-99
ML16P-1	Male	3.95	3.59	3.40
ML16P-2	Male	4.25	3.87	3.66
ML16P-3	Male	4.55	4.14	3.91
ML16S-1	Female	3.95	3.59	3.40
ML16S-2	Female	4.25	3.87	3.66
ML16S-3	Female	4.55	4.14	3.91

RS-232 LOCK-DOWN SCREWS

SUPPLIED W/HDWR SHOWN



MODEL Size	50-450	500-1200	1200 over
SDC-300	.18	.17	.16
SDC-400	.19	.18	.17
SDC-500	.20	.19	.18
SDH	.11	.10	.09

NOTICE: Multiples of 50 Only

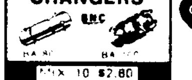
AC LINE TRANSIENT SURGE PROTECTOR

- Absorbs Spikes
- Safety Voltage Level
- Polarized Gnd'd Receptacle
- Rated: 15A 125VAC

MODEL	# taps	1-4	5-9	10-24
SS-1	Single	17.95	15.98	13.82
SS-3	Triple	19.95	17.56	15.36

Discounts offered in larger quantities

- POPULAR - GENDER CHANGERS



MIX 10 $2.80
MIX 25 $2.60

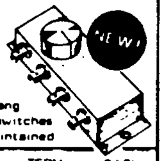
YOUR CHOICE 2.95

COMPARE

DEALER REQUESTS ATTENDED

DATA COAXIAL SWITCH

- Rugged Build
- Fully Shielded
- AB or ABC Types
- Low Contact Res
- Mounting Flanges
- Metal Construction
- BNC or TNC & Wang
- Center conductor switches
- Shield isolation maintained

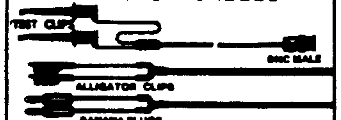


MODEL	FUNCTION	TERM	EACH
DS12B	SPDT AB	BNC	65.00
DS22B	DPDT AB	BNC	79.00
*DS22BT	DPDT AB	BNC-TNC	85.00
DS13B	SPDT ABC	BNC	75.00
DS23B	DPDT ABC	BNC	94.00
*DS23BT	DPDT ABC	BNC-TNC	99.00

*For Wang Devices-Switches Transmit & Receive Cables Simultaneously

Qty. Discounts & Custom Types Available

BNC TEST CABLES



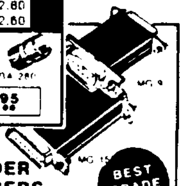
MODEL	TERMINATION	1-9	10-24
BC-01	Test Clip	8.95	8.08
BC-02	Alligator Clip	7.10	6.39
BC-03	Banana Plug	7.75	6.98

PARALLEL (Centronics) GENDER CHANGERS

Mates Centronics parallel printer cables that are of the wrong sex. All 36 pins are wired through. Bright metal construction. Mating Centronics cables are shown below.

MODEL	GENDER	1-9	10-24	25
DCS-36F	Female-Female	36.95	33.62	29.56
DCS-36M	Male-Male	36.95	33.62	29.56

GENDER CHANGERS



MODEL	GENDER	1-9	10-24
MG-9F	F-F	15.95	14.51
MG-9M	M-M	15.95	14.51
MG-15F	F-F	18.95	17.24
MG-15M	M-M	18.95	17.44

MINIMUM ORDER $25.00 • ADD $4.50 Shipping & Handling • You're Welcome to Visit Us

L-COM®

DATA PRODUCTS
1755 Osgood St. Rte 125
NO. Andover MA 01845

CHECK or M O

Toll Free Ordering
800-343-1455

MA Customers
617-682-6936
& Add 5% Sales Tax

Editor's Page, continued

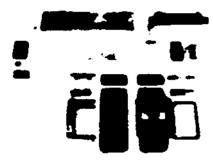
experiment with different parameters, and the response of the item being controlled is so slow that I add long delay loops. When driving printers or controlling high speed motors I'll switch to assembly language for high speed, real time control.

For several years there were few new developments in the area of assembly language, but as more people became aware of the limitations of the high level languages, the interest in assembly language increased. There are now a number of new books and utilities for the assembly language programmer. Some of the products which we are reviewing are Z80ASM (a high-speed assembler from SLR Systems), SMAL/80 (Structured Macro-Assembly Language from Chromod Associates), S-C Macro Assembler (Apple II assembler from S-C Software), and the book **Assembly Language Cookbook for the Apple II/IIe** by Don Lancaster (published by Howard Sams).

We believe that our readers should become familiar with several languages so that they can choose the best language for each project. However, we consider assembly language to be the foundation upon which all other languages are built, so we will encourage our readers to learn assembly language. Remember, high level language may make the program easier for the programmer to write, but assembly language may make the program easier for the user to use.

We will continue to carry information on other languages, and have discussed potential articles on FORTH, LISP, and SAVVY, among others. We would appreciate your letters and articles presenting arguments for your favorite language, especially those showing a comparision of how to accomplish the same task with different languages. One of the purposes of a journal is to present a forum for differing points of view. I doubt that any of our readers have agreed with everything that our authors have written, but we have received very little feedback. Take the time to tell us about your viewpoint and your experiences. It doesn't have to be fancy; we'll rewrite it if you have the information, and we'll withhold your name if requested. We would like to set up a bulletin board for your feedback as soon as we can afford it, but for now you'll have to spend the effort to write! ■

APROTEK 1000™ EPROM PROGRAMMER



TECHNICAL BREAKTHROUGH NOW ALLOWS A PRICE BREAKTHROUGH

only $250.00

A SIMPLE, INEXPENSIVE SOLUTION TO PROGRAMMING EPROMS

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

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

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

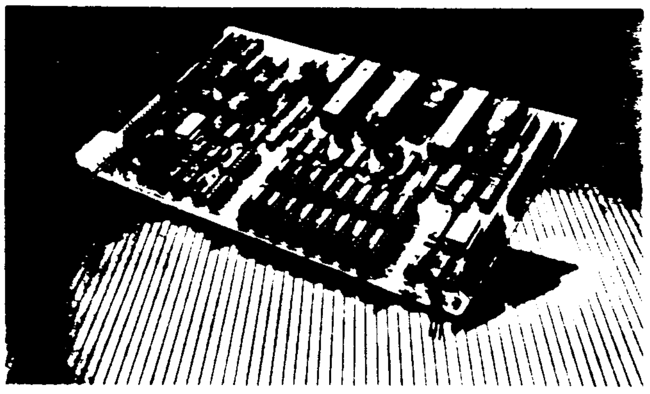
APROTEK 1000 is covered by a 1 year parts and labor warranty.

FINALLY — A Simple, Inexpensive Solution To Erasing EPROMS

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

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

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



POWER THAT GOES ANYWHERE!

Single Board Computer

FAST	— 6MHz Z80B* CPU
POWERFUL	— 64K to 256K RAM, 2K to 64K ROM
	— 5¼" and 8" Floppy Controller SASI
	— 2 RS-232, Centronics Port
FLEXIBLE	— 50-pin I/O Expansion Bus
SMALL	— 5¾" x 10"

DAVIDGE CORPORATION
292 East Highway 246
PO Box 1869
Buellton, CA 93427

(805) 688-9598

*Z80 is a registered trademark of Zilog

Interfacing Tips and Troubles

A Column by Neil Bungard

So far in "Interfacing Tips and Troubles" we have looked at a DC to DC converter for creating custom voltages, learned how to eliminate troublesome interfacing noise, and reviewed techniques which allow the inexpensive Sinclair computers to be used for real world control. In future installments of "Interfacing Tips and Troubles" we will keep the same instructive format, and vary our subject matter so that everyone will hopefully find something of interest in this column. In the next couple of installments I will present a few unique instruments that every hacker should own. I assure you that you will find these devices useful and interesting.

The first instrument, and the subject of this month's column, is a Poor-Man's Logic Analyzer (PLA). The PLA is actually a multichannel logic probe. Eight separate logic probes reside on a single PC board, and each probe can be switch selected to operate either in a "normal" mode or in a "memory" mode. When a channel of the PLA is connected to a logic line and the selector switch is in the normal mode, any logic changes on the line will appear exactly as they occur on the LEDs of the probe. The disadvantage of this, of course, is that if a fast pulse occurs on the line, your eye (being relatively slow) will not see the LEDs change. For this reason, each channel was given a switch selectable pulse memory. When a channel is selected for the memory mode, all pulses are effectively stretched so that your eye can see them. Pulses as short as 10 nanoseconds can actually be detected when the probe is in the memory mode.

Of course the PLA is not as versatile or as sophisticated as a real logic analyzer. A real logic analyzer actually creates a timing diagram on its CRT which shows the history of each logic line over a given span of time. The PLA, however, can provide some very useful information. First of all, eight different lines can be monitored simultaneously. This is a feature which is necessary when trying to analyze the condition of any bus structure. With the PLA the following questions can be answered:

- Is a line active or tristated?
- If active, is it a logic one or a logic zero?
- Is a line pulsing?
- If so, is it pulsing regularly, randomly, or intermittently?
- If there is a lot of activity on a line, is it predominantly low (logic zero) or is it predominantly high (logic one)?
- Are all eight lines active?
- What is the nature of the signals on each line (do the lines contain pulses, are they stable, are they switching states rapidly or slowly, etc.)?
- What changes occur on each line when some input on the computer is manipulated?

In nothing else, a multichannel logic probe eliminates the need to move your probe from checkpoint to checkpoint when attempting to locate a problem. If you are like me, that makes a big difference. One slip with a logic probe on a complicated PC board and you've caused more problems than you were hoping to solve.

Theory of Operation

Figure 1 shows a schematic drawing of one channel of the Poor-Man's Logic Analyzer. The bases of transistors T3 and T6 are connected to the circuit under test through resistor R3 and capacitor C3. Assume the selector switch is in the normal mode; this disconnects capacitors C1 and C2 from the PLA circuit. If the incoming signal is a logic zero, transistor T3 is turned off and transistor T6 is turned on. With transistor T6 turned on, transistors T5 and T4 are also turned on and the green LED is lit. If the incoming signal is a logic one, transistor T6 is turned off and transistor T3 is turned on. With transistor T3 turned on, transistors T2 and T1 are also turned on and the red LED is lit. When the selector switch is in the memory mode, capacitors C2 and C1 are connected to the bases of transistors T2 and T5 respectively. If a high-going pulse triggers the probe, T3 pulses on and off. When T3 turns on, it charges capacitor C2 to +5 volts. This charged capacitor keeps transistor T2 turned on until the voltage across the capacitor drops below transistor T2's turn off voltage. The current required to keep T2 turned on is very low, so the voltage across C2 will take a relatively long time to bleed off. This action effectively stretches the incoming pulse and keeps the red LED on long enough to detect the pulse's presence. A low

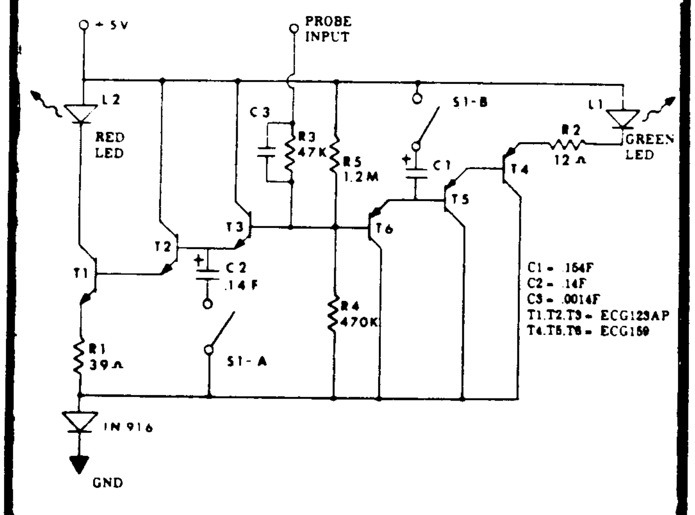


Figure 1

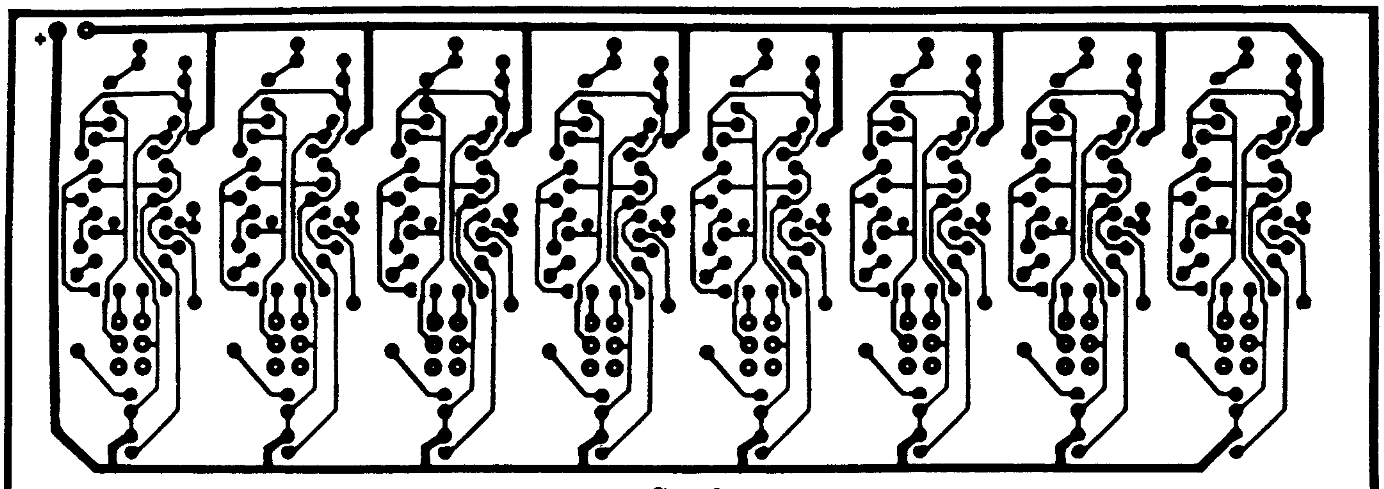


Figure 2

going pulse is determined in the same manner via capacitor C1 and transistor T5.

The first PLA that was built was a wire wrap version. The wire wrap version worked well, but to ensure long term reliability, I created a circuit board for the project. With the schematic in Figure 1, you can build a wire wrap version of the PLA. For those of you who know how to lift a figure and etch your own boards, the foil pattern of the PLA is provided in Figure 2. For those of you who don't know how, or don't care to make your own circuit boards, a board can be purchased from Interface Consultants (address in Figure 3). The price of a bare PLA circuit board is $15.00. To help you locate all of the parts needed for the PLA, a parts list including manufacturers and part numbers is provided in Figure 3. To aid in construction of the PLA, a parts placement diagram is provided in Figure 4.

In conclusion, I have found the PLA to be an indispensable instrument for the hacker's tool box—especially for the price! The PLA can be built for about $6.25 per channel (including the circuit board), which is less than any single logic probe that I have seen on the market. If you have any questions or comments concerning the PLA, drop me a line here at *The Computer Journal*. ■

CIRCUIT LABEL	QTY	DESCRIPTION	PART NUMBER	MANUFACTURER
T1,T2,T3	3	NPN TRANSISTOR	ECG 123AP	INT.CON.
T4,T5,T6	3	PNP TRANSISTOR	ECG 159	INT.CON.
S1	1	DPDT SWITCH	SW102-ND	DIGI-KEY
C1	1	.15 MFD TANTALUM CAP	551-0.15M35	MOUSER
C2	1	.1MFD TANTALUM CAP	551-0.15M35	MOUSER
C3	1	.001 MFD DISC CAP	21EE001	MOUSER
D1	1	1N916 DIODE	581-1N916	MOUSER
L2	1	3MM RED LED	35BL504	MOUSER
L1	1	3MM GREEN LED	35BL505	MOUSER
R2	1	12 OHM RESISTOR	29SJ250	MOUSER
R1	1	39 OHM RESISTOR	29SJ250	MOUSER
R3	1	47K OHM RESISTOR	29SJ250	MOUSER
R4	1	470K OHM RESISTOR	29SJ250	MOUSER
R5	1	1.2M OHM RESISTOR	29SJ250	MOUSER
	2	ALLIGATOR CLIPS	548-30	MOUSER
	1	PLA CIRCUIT BOARD	PLA-1	INT.CON.
	8	IC TEST CLIP	13IC308-994	MOUSER

```
INT.CON.--Interface consultants
          342-1 Armstrong Hills Rd.
          Surgoinsville, TN 37873

DIGI-KEY--DIGI-KEY Corporation
          Highway 32 South
          P.O. Box 667
          Thief River Falls, MN 56701

MOUSER--Mouser Electronics
          11433 Woodside Ave.
          Santee, CA 92071
```

Figure 3

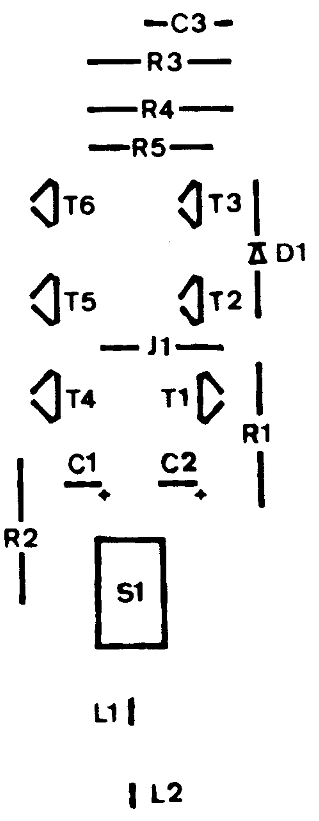


Figure 4

Controlling the Hayes Micromodem II From Assembly Language, Part Two

by Jan Eugenides

Last month I presented you with the first part of this program, which consisted of routines to dial the phone, gather incoming data, display it on the screen, store it in memory, and send data from the keyboard (with upper and lower case conversion). I also explained the workings of the command handler. This time, I'll show you the disk access routines, develop a way to save the buffer to disk, and show how to upload files directly from the disk. I'll add the input and print routines and a configuration program, and voila!, a working terminal program for the Apple II.

If you missed last month's article, or if you hate typing, I can supply you with the complete source code, the configuration program, and a binary file of the program itself on disk for $10.00 (plus $1.50 postage). Just send a check to me at 11601 N.W 18th St., Pembroke Pines, FL 33026.

Examine Listing 3. These are the lines I left out of Listing 1 last time. You can type them in as they appear, and they will fit into the proper spaces. Just watch the line numbers carefully.

The Equates

In lines 1180-1450 you'll find the equates, the various addresses the program uses or calls. You will remember from last time that CR1, CR2, CHAR, and OUTA are part of the Micromodem firmware. In addition, there are a few routines from the Apple monitor, and a couple of zero-page pointers and flags. Rather than go into each one separately, I will explain the function of each of them along with their associated routines.

Initialization

Lines 1500-1904 are the initialization part of the program. First, a zero goes in the capture flag, which means the program starts up with capture "on." Next, the monitor HOME routine is called to clear the screen, and then the window top edge is set to line four by simply storing a four in location $22. This will prevent the title lines from scrolling off the screen. Next, the capture buffer top address is set to $800, which is also the bottom address. In other words, the buffer is empty.

Lines 1650-1690 are necessary because DOS 3.3 does not allow direct file commands. Unless there is an Applesoft program running, you cannot OPEN, READ, or WRITE to a text file. So, our program adjusts the values of a few locations to fool DOS into thinking an Applesoft program is running, and allows us to issue normal DOS file commands from within our binary program. You'll see how this is used a little later.

Then, in lines 1710-1760, the ONERR flag is set up, and the address of our file closing routine is placed in the DOS error handler locations, $9D5A and $9D5B. This way, when DOS encounters an error, instead of a horrible crash it jumps to our routine, which closes any open files and continues with the program. This will come in handy in the uploading routine.

Finally, the titles are printed on the screen, and the main program loop is entered. (The main loop was presented last time.)

Saving The Buffer

When you press "◄ESC► S" the command handler passes control to the SAVEBUF routine in lines 4060-4660, after first signaling the host to stop sending data (X-off).

In order to save the buffer to disk, you need to know three things:

1. The start of the buffer.
2. The end of the buffer.
3. The filename to save it under.

The start of the buffer is always $800, so in lines 4110-4130 the pointer LNGTH is set to $800. Next, in lines 4170-4250, a filename is requested from the attending human, and stored in a safe place pointed to by IPTR. More about this in the discussion of the input routine a bit later. Once that's done, you need to OPEN the file. Here's where the direct DOS commands come in. Because DOS thinks an Applesoft program is running, all that you have to do now is issue a normal DOS command, by outputting a CTRL-D ($84) followed by the command. DOS only looks for commands at the beginning of a line, so you must precede the CTRL-D with a ◄CR► ($8D). Thus in lines 4270-4480, the program first OPENs the file, and then issues the WRITE command. Notice how a short loop is used to output the filename which is stored at IPTR.

Now, DOS is set up to WRITE to a text file, and all you have to do is spit out the buffer through COUT. This occurs in lines 4520-4610. As each byte is output, LNGTH is incremented and compared against PTR. When the two are the same, the end of the buffer has been reached. Then in lines 4360-4640, a final ◄CR► is output. Line 4660 CLOSES the file and returns to the main loop.

To save memory to disk as a text file:
1. Put $40 at $AAB6 and $76
2. Put $06 at $33 (PROMPT)
This allows direct file commands.
Then, output the data byte by byte through COUT

Cataloging The Disk

Knowing what you do about issuing DOS commands, this should be easy. Just output ◄CR► CTRL-D CATALOG ◄CR►. This happens in lines 4700-4750.

Reviewing The Buffer

When you're on-line, it is often useful to be able to see something that has already scrolled off the screen. A way is needed to display the contents of the buffer on the screen without disconnecting. Lines 4800-5060 accomplish this.

This routine works almost identically to the SAVEBUF routine, except that no file is open, and output goes to the screen. Also, for the sake of Apples without lower case, the characters are converted to upper case before display. The conversion to upper case can be defeated by the configuration program, for those of you with Apples which can display lower case. Notice that even if your Apple cannot DISPLAY lower case, you can still CAPTURE lower case material, and later load it into your word-processor, or print it on your printer. I haven't seen any other terminal program which does this! In line 4940 I have selected a delay to slow down the display of the buffer. Otherwise it scrolls by mighty fast. CTRL-S will of course stop the scrolling, but you may wish to change this delay setting to suit your taste.

Filling Up The Buffer

It is necessary for the user to know how much space is left in the buffer. As you saw last time, when the buffer fills up, the X-off is sent and a message is printed to the screen. You have the option at that point of saving the buffer, or clearing it. For some reason, the buffer always seems to fill up at the most inopportune times (Murphy's Law, right?). Rather than allow this to happen, you can monitor the buffer as it fills.

As each character is stored in the buffer, PTR is incremented. Therefore, PTR always points to the next available memory location. You can display this on the screen with a simple HEX to ASCII conversion. This is accomplished in lines 5100-5620.

There are lots of ways to convert HEX to ASCII, and there are certainly more elegant ways than the one I have used here. Unfortunately, the more elegant, the more difficult to understand, and this method works well. Let's examine it.

In line 5110 the high-order nibble is masked off (A nibble is 4 bits, one-half of a byte). You have to deal with each nibble separately, because each one becomes one ASCII character. In other words, $AF is only one byte in hex, but the letters AF require two bytes as ASCII.

If the nibble is between $00 and $09 then all that is necessary is to add $B0 to get the ASCII equivalent (Remember, we're dealing with "high" ASCII on the Apple, the eighth bit is set). If the nibble is between $0A and $0F, you must add $B7. Look it up on the ASCII chart I gave you last time, nd you'll see why. Between 9 and A in ASCII there are seven other character, :, ;, >, =, <, and ?. The additional seven is necessary to skip over them.

Once the nibble is converted, it is stored directly in screen memory. This is the quickest way to put something on the screen! And you always know right where it will show up. $424-$427 corresponds to the upper right corner of the screen, where "BUFFER TOP = $" has previously been printed. Each nibble is converted and displayed and then the main loop is reentered. It takes a while to explain, but it all happens very fast.

To convert HEX to ASCII:
1. Separate each nibble.
2. If 0-9 add $B0
3. If A-F add $B7

The Input Routine

This routine is used to get filenames, either to save or to upload. It uses the monitor GETLN routine, at $FD6F. This routine gets a line of input, and stores it at $200. It returns with the length of the string in the X register. In line 5910 the program checks for a null string, that is, one with length of zero. This would mean that only ◄RETURN► was pressed. If this is the case, lines 6010-6100 check to see if you're on-line (just like you did with the carrier detect routine last month). If you are, the main loop is reentered, if not, the program is terminated.

Assuming a filename has been entered, our routine takes the string from $200 and moves it to $9600 where it won't be bothered. Actually, in this case this is probably not necessary, but it's not a good practice to leave valuable information in the input buffer at $200, because it could easily be overwritten. The DOS buffer at $9600 won't be used, so that's a safe place. (Be careful, though, it might not be safe for other programs you may write.)

The Print Routine

This routine is a pretty standard way of printing embedded ASCII data. It allows you to put the ASCII right in the code, rather than having to structure it all in a table somewhere. Makes changes easier, too. It works like this:

In line 6408-6414 the address of the calling routine is retrieved from the stack, and stored in CTR. The next character is then loaded into the accumulator (it will be the first ASCII character). A loop is entered, and each character is output through COUT until a zero is reached. You MUST put a zero after the embedded ASCII data! Also, there cannot be more than 255 ASCII characters printed with any one call to this routine.

Once the end of the data is found, the Y register is then added to the address stored in CTR. This new address is then pushed back on the stack, and the RTS results in a return to the next instruction after the ASCII data! A little tricky, but nice.

Uploading From Disk

Most terminal programs I have seen upload text from memory. This is DUMB. First of all, you waste all that time reading the file in from the disk to memory, and secondly you limit the size of the file you can upload to the size of the available memory. Why not simply read in the file byte by byte, and upload it that way? Then, you can upload a file of any size that will fit on a floppy!

The upload routine is contained in lines 6560-7080, and begins in much the same way as the SAVEBUF routine. The same input routine is used, and the DOS commands OPEN and READ are issued in the same way. The actual reading of the file is done in lines 7120-7240.

The easiest way to read from an OPEN file is to call the monitor routine RDKEY at $FD0C. This routine gets one character from the currently selected input device, in this case the disk drive. Then, in line 7150, the character is stored in CHAR, which you will remember is the location the Micromodem uses for characters which are to be sent out through the modem. Next, to insure normal video, the character is ORed with $80 to set the high bit, and another monitor routine, STORADV is called. This routine displays the character on the screen, and advances the cursor position. Slick, eh? If the character is zero, the end of the file must have been reached, so the program jumps to the CLOSE routine. In practice, DOS usually issues an "END OF DATA" error before this happens, and jumps to the address we so cleverly set up way back in initialization. Since this is the address of the CLOSE routine, it has the same effect. Otherwise, OUTA is called to send the character. In line 7210-7220, a delay is introduced to slow things down to 300 baud. Without this delay, the routine runs too fast and characters will be lost.

Finally, in lines 6930-7080 the file is CLOSEd and the word "DONE" is put on the screen in inverse.

I should point out that if you are uploading a file which contains lower case, and your Apple cannot display lower case, what you see on the screen will be garbage. The file will be uploaded correctly, however. This is your chance to try out what you learned before about upper-lower case conversions, and modify the program to display only upper case when uploading. Consider it your homework assignment!

To upload a text file:
1. Open the file for READing using normal DOS commands
2. Call RDKEY ($FD0C) to get the characters
3. On END OF DATA error, close file

That pretty much wraps up the blow-by-blow description. Now all you need to know is how to USE the program!

How To Use Big Buff

Big Buff is a breeze to use, but first you must configure it for the slot your modem is in, and for the type of machine you own. Type in and run the CONFIGURE program in Listing 4. This program will allow you to select the slot, and whether you have lower case display and/or keyboard. The file BIG.BUFF must be on the disk when you run the CONFIGURE program. A new file named BIG.BUFF.CONFIGURED will be produced. BRUN this file, or BLOAD it and, from the monitor, type 9100G.

You'll be asked for a phone number to dial. Just type it in and hit ◄RETURN►. The modem will pick up the phone and dial. If a carrier is detected, you'll see:

MICROMODEM II: CONN

on your screen. Log on to the host in the usual way (each host is different, so it's up to you to know the procedure.) Text will start being displayed on the screen, and you'll see the BUFFER TOP = $0800 display changing rapidly as data is stored in the buffer. You have seven commands available to you, as I discussed last month. They are:

◄ESC► Z	Zero the buffer (clear it)
◄ESC► X	Hang up the phone
◄ESC► S	Save the buffer to disk
◄ESC► B	Toggle capture on and off
◄ESC► U	Upload a file from disk
◄ESC► C	Catalog the disk
◄ESC► R	Review the buffer (display on screen)

Try them out. They all work just like you would expect them to. I have noticed that occasionally, if you try to hang up the phone with ◄ESC► X during some log-off sequences (notably Compuserve), DOS picks up the incoming characters as a command, and issues a "SYNTAX ERROR." This is of course trapped by our program, and control passes to the CLOSE routine. No damage done, but you'll see an inverse "DONE" on the screen. Just press ◄ESC► X again, and this time it will work normally. You are given one last chance to save your buffer, ◄RETURN► alone exits the program.

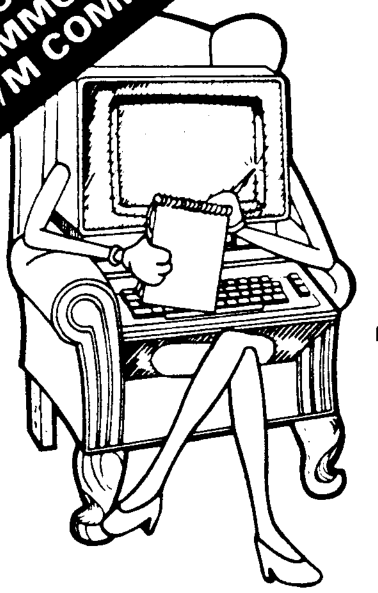
Possible Uses

Big Buff is useful just as it is for day to day networking, chatting on Compuserve or whatever. Many bells and whistles could be added, and there is much room for customized applications. An easy mod would be to allow macros of some kind. All you would need would be a data table, and a routine to output the macros using OUTA. I am considering putting together a package for the E-COM system (Electronic Computer Originated Mail), combining the basic routines I have presented here with a specialized data base and word processsing program to make using the E-COM system easy for Apple users. If you haven't heard of E-COM, get in touch with the U.S Postal Service for more info. It's neat! Or, set up your own BBS (bulletin board system) (you'll have to add auto-answer capabilities), or develop a local network of Apple users. Whatever. Now you know how, right? As always, I welcome your questions and comments. If you come up with a good use for Big Buff, or any nifty modifications, let me know. Or just write to say "HI!"

Next Time

In the works is a serial/parallel interface which connects to the game port and allows you to run printers, modems, speech synthesizers, whatever, out of the game port! As of

FOR YOUR APPLE
IBM, COMMODORE 64
OR CP/M COMPUTER



Problems with your Lover?

Tell ELIZA!

Meet ELIZA,
the computer psychotherapist.

Created at MIT in 1966 to run on a large mainframe, ELIZA has become the world's most celebrated artificial intelligence program. ELIZA will analyze any statement you enter and respond in true Rogerian fashion - and her remarks are often amazingly appropriate!

Unlike the stripped down versions you may have seen, our ELIZA has retained the full power and range of expression of the original mainframe program.

Best of all, ELIZA comes with the complete Source Program (written in BASIC) ... anyone, even a beginner, can easily customize ELIZA's responses.

So next time people ask you what your computer can do, bring out ELIZA and show them!

"Much more than a mere game ... you'll be impressed with Eliza ... a convincing demonstration of artificial intelligence" - PC magazine

"Delightful entertainment ... an ideal medium for showing off your system." - Microcomputing magazine

ELIZA is only $45 and available in the following formats:

- Apple II, II plus, IIe, IIc, III
- IBM PC and all compatibles
- Commodore 64 (disk or cassette)
- 8" disk for all CP/M systems
- 5¼" disk for most CP/M systems (specify make and model)

Please add $3.00 for shipping and handling
(California residents, add 6½% sales tax.)

Checks, MasterCard, and VISA accepted

Artificial Intelligence
RESEARCH GROUP

921 N. La Jolla Ave., Dept. N, Los Angeles, CA 90046 • (213) 656-7368 • (213) 654-2214

this writing, I've got it working with my Epson MX-80. Watch this magazine for it! In addition, I'll be showing you how to write a printer driver for the interface, and who knows what else will follow?! ■

Listing 3: To be combined with Listing 1 from last month (see article for details)

```
1010 *S.BIG.BUFF
1030 *----------------------------
1040 *BIG BUFF terminal program for
1050 *Apple II+ with Micro Modem II
1060 *
1070 *
1080 *
1090 *(C) 1984 Jan G. Eugenides
1100 *11601 N.W. 18 ST.
1110 *Pembroke Pines FL 33026
1120 *305-431-6892
1130 *12/5/84
1140 *----------------------------
1150 *CONFIGURED FOR SLOT 2
1160 *----------------------------
1170 *
1180 PTR     .EQ $06
1190 CR1     .EQ $C0A6
1200 CR2     .EQ $C0A5               modem control port
1210 CHAR    .EQ $778                temp char storage
1220 RDKEY   .EQ $FD0C
1230 OUTA    .EQ $C202
1240 COUT    .EQ $FDED
1250 HOME    .EQ $FC58
1260 GETLN   .EQ $FD6F
1270 BUFF    .EQ $200
1280 VTAB    .EQ $FC22
1290 CH      .EQ $24                cursor horiz. position
1300 BASL    .EQ $28
1310 STORADV .EQ $FBF0
1320 CV      .EQ $25                cursor vert. position
1330 PROMPT  .EQ $33
1340 CURLIN  .EQ $75
1350 LANG    .EQ $AAB6
1360 IPBUF   .EQ $9600              input buffer
1370 IPTR    .EQ $F9
1380 CAPFLAG .EQ $FB
1390 TEMP    .EQ $FC
1400 LNGTH   .EQ $19
1410 INVFLG  .EQ $32
1420 WAIT    .EQ $FCA8
1430 VTEMP   .EQ $09
1440 HTEMP   .EQ $0A
1450 CTR     .EQ $0B
1460 *
1470         .OR $9100
1480         .TF BIG.BUFF
1490 *
1500 *----------------------------
1510 *Initialization
1520 *----------------------------
1530         LDA #$0
1540         STA CAPFLAG    Capture flag
1550         JSR HOME       Clear screen
1560         LDA #4
1570         STA $22        Window top edge
1580         LDA #$00
1590         STA PTR        Set buffer top to $800
1600         LDA #$08
1610         STA PTR+1
1620 *----------------------------
1630 *Allow direct file commands
1640 *----------------------------
1650         LDA #$40
1660         STA LANG
1670         STA CURLIN+1
1680         LDA #$06
1690         STA PROMPT
1700 *---------------------------------
1710         LDA #$40
1720         STA $D8             Set errflag so that errors
1730         LDA #CLOSER         go to file close routine
1740         STA $9D5A
1750         LDA /CLOSER
1760         STA $9D5B
1770
1780 *----------------------------
1790 *Print header
1800 *----------------------------
1810         JSR PRINT
1820         .AS -"BIG BUFF TERM PROGRAM---BUFFER TOP=$"
1830         .HS 8D8D
1840         .AS -"    CAPTURE: ON     (C)JAN G. EUGENIDES"
1850         .HS 8D00
1860         LDA #$03
1870         STA CV         VTAB 3
1880         JSR VTAB
1890         JSR PRINT
1900         .AS -"---------------------"
1902         .AS -"---------------------"
1904         .HS 00

Lines from last month go here...

4060 *-------------------------------
4070 *Save buffer to text file
4080 *----------------------------
4090 SAVEBUF LDA #$8D
4100         JSR COUT           Start new line
4110         LDA #0             Set beginning of file to $800
4120         STA LNGTH
4130         LDA #$08
4140         STA LNGTH+1
4150 **
4160 *
4170         LDA #$3F
4180         STA INVFLG         Inverse
4190         JSR PRINT
4200         .AS -"FILE NAME TO SAVE?"
4210         .HS 8D00
4220         LDA #$FF           Normal
4230         STA INVFLG
4240 **
4250         JSR INPUT          Get file name
4260 *
4270         JSR PRINT
4280         .HS 8D8D84 ;<CR>+CTRL-D
4290         .AS -"OPEN "
4300         .HS 00
4310         LDY #$00
4320 BLOOP   LDA (IPTR),Y       Get file name from IPBUFFER
4330         BEQ BL1            Zero means end of name
4340         JSR COUT
4350         INY
4360         JMP BLOOP
4370 BL1     JSR PRINT
4380         .HS 8D84
4390         .AS -"WRITE"
4400         .HS 00
4410         LDY #$00
4420 BLOOP2  LDA (IPTR),Y
4430         BEQ BL2
4440         JSR COUT
4450         INY
4460         JMP BLOOP2
4470 BL2     JSR PRINT
4480        .HS 8D00
4490 *
4500
4510        LDY #0
4520 STLOOP LDA (LNGTH),Y       Write buffer to disk
4530         JSR COUT
4540         INC LNGTH
4550         BNE .2
4560         INC LNGTH+1
4570 .2      LDA LNGTH
4580         CMP PTR
4590         LDA LNGTH+1
4600         SBC PTR+1
4610         BCC STLOOP
4620 *
4630         LDA #$8D           Put <CR> at end
4640         JSR COUT
4650 *
4660         JMP CLOSER
4670 *-------------------------------
4680 *Catalog disk
4690 *-------------------------------
4700 CATALOG JSR PRINT
4710          .HS 8D8D84
4720          .AS -"CATALOG"
4730          .HS 8D00
4740          JSR XON
4750          JMP MLOOP
4760 *
4770 *-------------------------------
4780 *Review
4790 *-------------------------------
4800 REVIEW  LDA #$8D
4810         JSR COUT           Start new line
4820         LDA #0             Set beginning of file to $800
4830         STA LNGTH
4840         LDA #$08
4850         STA LNGTH+1
4860 *
4870         LDY #0
4880 RWLOOP  LDA (LNGTH),Y      Get character from buffer
4890         CMP #$E0           Lowercase?
4900         BCC .3             No
4910         SEC
4920         SBC #$20            Yes-convert to upper
4930 .3      JSR COUT            Display on screen
4940         LDA #40
4950         JSR WAIT            Slow it down a bit!
4960         INC LNGTH           Increment pointer
4970         BNE .2
4980         INC LNGTH+1
4990 .2      LDA LNGTH
5000         CMP PTR
5010         LDA LNGTH+1
5020         SBC PTR+1
5030         BCC RWLOOP          More to go
5040 *
5050         JSR XON
5060         JMP MLOOP
5070 *-------------------------------
5080 *Show buffer high address      *
5090 *-------------------------------
5100 SHOWBUF LDA PTR
5110         AND #$0F            Get low nibble
5120         CMP #$0A            Convert to screen ascii
5130         BCS LETTER1         It's A-F
5140         ORA #$B0            It's 0-9
5150         STA $427            Print to screen
5160         JMP HINIB1
5170 LETTER1 ORA #$B0            Convert to screen ascii
5180         CLC
5190         ADC #7
5200         STA $427            Print to screen
5210 HINIB1  LDA PTR            Repeat for hi nibble
5220         LSR
5230         LSR
```

```
5240         LSR
5250         LSR
5260         CMP #$0A
5270         BCS LETTER2
5280         ORA #$B0
5290         STA $426
5300         JMP HINIB2
5310 LETTER2 ORA #$B0
5320         CLC
5330         ADC #7
5340         STA $426
5350 *
5360
5370 HINIB2 LDA PTR+1          Now do the hi byte of the address
5380         AND #$0F          Get low nibble
5390         CMP #$0A
5400         BCS LETTER3
5410         ORA #$B0
5420         STA $425
5430         JMP HINIB3
5440 LETTER3 ORA #$B0
5450         CLC
5460         ADC #7
5470         STA $425
5480 HINIB3 LDA PTR+1           Repeat for hi nibble
5490         LSR
5500         LSR
5510         LSR
5520         LSR
5530         CMP #$0A
5540         BCS LETTER4
5550         ORA #$B0
5560         STA $424
5570        JMP HINIB4
5580 LETTER4 ORA #$B0
5590         CLC
5600         ADC #7
5610         STA $424
5620 HINIB4 RTS
```

Lines from last month go here...

```
5820 *--------------------------------
5822 *INPUT
5824 *--------------------------------
5826 INPUT  LDA #$80
5830        STA $D8
5840        LDA #IPBUF          Store at $9600
5850        STA IPTR
5860        LDA /IPBUF
5870        STA IPTR+1
5880        LDX #0
5890        JSR GETLN           Get a line of input
5900        TXA                 X holds length of line
5910        BEQ EXIT            If <CR>, just exit
5920        TAY
5930        LDA #$0
5940        STA BUFF,Y          Put zero at end of line
5950 .2     LDA BUFF,Y
5960        STA (IPTR),Y        Move to $9600 for safekeeping
5970        DEY
5980        CPY #$FF
5990        BNE .2
6000        RTS
6010 EXIT   LDA CR1             On line?
6020        CMP #128
6030        BCC .1
6040        LDA #$40
6050        STA $D8
6060        PLA
6070        PLA
6080        JSR XON
6090        JMP MLOOP
6100 .1     JMP $3D0            Return to basic if off-line
```

Lines from last month go here...

```
6400 *--------------------------------
6402 *Print routine
6404 *
6406 *--------------------------------
6408 PRINT  PLA                 Get address
6410        STA CTR
6412        PLA
6414        STA CTR+1
6416        LDY #$01
6418 .1     LDA (CTR),Y         Get data to print
6420        BEQ .3              Zero? then end
6422        JSR COUT            Print character
6424        INY
6426        BNE .1
6430 .3     CLC                 Return to next instruction after data
6440        TYA
6450        ADC CTR
6460        STA CTR
6470        LDA CTR+1
6480        ADC #$00
6490        PHA
6500        LDA CTR
6510        PHA
6520        RTS
6530 *---------------------------------
6540 *Read text file
6550 *---------------------------------
6560 READ
6580        LDA #$8D
6590        JSR COUT
6600 *
6610        LDA #$3F
6620        STA INVFLG          INVERSE
6630        JSR PRINT
6640        .AS -"FILE NAME TO UPLOAD?"
6650        .HS 8D00
6660        LDA #$FF            Normal
6670        STA INVFLG
6680 **
6690        JSR INPUT           Get file name
6700        JSR PRINT
6710        .HS 8D8D84
6720        .AS -"OPEN"
6730        .HS 00
6740        LDY #$00
6750 RBLOOP LDA (IPTR),Y
6760        BEQ RL1
6770        JSR COUT
6780        INY
6790        JMP RBLOOP
6800 RL1    JSR PRINT
6810        .HS 8D8D84
6820        .AS -"READ"
6830        .HS 00
6840        LDY #$00
6850 RBLOP2 LDA (IPTR),Y
6860        BEQ RDLOOP
6870        JSR COUT
6880        INY
6890        JMP RBLOP2
6900 RDLOOP JSR PRINT
6910        .HS 8D00
6920        JMP IO       Get data
6930 CLOSER JSR PRINT
6940        .HS 8D84
6950        .AS -"CLOSE"     Close files
6960        .HS 8D00
6990        LDA #$3F
7000        STA INVFLG
7010        JSR PRINT
7020        .HS 8D
7030        .AS -/DONE/
7040        .HS 8D00
7050        LDA #$FF
7060        STA INVFLG
7070        JSR XON
7080        JMP MLOOP
7090 *--------------------------------
7100 *Input from file
7110 *--------------------------------
7120 IO     JSR XON
7130 *
7140 IOLOOP JSR RDKEY           Get character
7150        STA CHAR            Set up to ouput from modem
7160        ORA #$80              Insure normal video
7170 .1     JSR STORADV          Display on screen
7180        LDA CHAR
7190        BEQ IODONE          Zero? then done
7200        JSR OUTA            Ouput character thru modem
7210        LDA #$70
7220        JSR WAIT            Slow down to 300 baud
7230        JMP IOLOOP          Do it again
```

Listing 4

```
5  REM *********************
10  REM *CONFIGURE BIG BUFF *
20  REM *     PROGRAM       *
30  REM *     (C)1984       *
40  REM * JAN G. EUGENIDES  *
50  REM *1543 N.E. 123 ST.  *
60  REM *N. MIAMI, FL 33161 *
70  REM *  305-891-2355     *
75  REM *********************
80 D$ =  CHR$ (13) +  CHR$ (4)
90  PRINT D$;"BLOAD BIG.BUFF"
100  HOME : PRINT "     BIG BUFF CONFIGURATION PROGRAM"
110  FOR N = 1 TO 40: PRINT "-";: NEXT
120  VTAB 7: PRINT "WHAT SLOT IS YOUR MODEM IN?";: GET A$: PRINT
A$:SLOT =  VAL (A$): IF SLOT < 1 OR SLOT > 7 THEN 120
125  VTAB 9: PRINT "DO YOU HAVE A IIE?";: GET B$: PRINT B$: IF B$ <  >
"Y" AND B$ <  > "N" THEN 125
126  IF B$ = "Y" THEN 200
130  VTAB 11: PRINT "DO YOU HAVE A LOWERCASE CHIP AND SHIFT  KEY
MODIFICATION?";: GET A$: PRINT A$: IF A$ <  > "Y" AND A$ <  > "N" THEN
130
150 N = 176 + SLOT:C1 = 134 + (SLOT * 16):C2 = 133 + (SLOT * 16):M =
120 + SLOT:C3 = 192 + SLOT
160  POKE 37310,N: POKE 37317,C1: POKE 37407,C3: POKE 37412,C1 + 1:
POKE 37507,C3: POKE 37516,C3
170  POKE 37596,C2: POKE 38063,C1: POKE 38093,N: POKE 38310,C3: POKE
38323,C1
180  IF A$ = "N" THEN 500
190  FOR X = 37328 TO 37338: READ OP: POKE X,OP: NEXT X
200  FOR X = 37451 TO 37453: POKE X,234: NEXT
210  FOR X = 37431 TO 37434: POKE X,234: NEXT
220  FOR X = 37853 TO 37855: POKE X,234: NEXT X
500  PRINT D$;"BSAVE BIG.BUFF.CONFIGURED,A$9100,L$4C5"
510  END
1000  DATA    48,3,76,193,145,44,16,192,174,99,192
```

Books of Interest

Assembly Language Cookbook
by Don Lancaster
Published by Howard W. Sams & Co., Inc.
4300 West 62nd Street
Indianapolis, IN 46268
408 pages, 8½" x 11", $21.95

Don has an unusual, refreshing, easy to read writing style, and he starts this book by telling you "Why You Gotta Learn Assembly Language." He certainly says what he means with outright statements such as "The only little thing wrong with BASIC or Pascal is that it is categorically impossible to write a decent Apple®][or //e program with either of them!" He then backs up his statement with the fact that every one of the top thirty best selling Apple programs are written in machine language or incorporate machine language modules. Don follows with several pages of arguments for assembly language including speed, program size, economy, and protection of your source code.

He does spend two pages discussing some of the disadvantages of machine language, but counters it with the question "Have you ever seen a machine language program that was *improved* by rewriting it in BASIC or Pascal?" There are many examples of programs originally written in higher level languages which were not commercially successful until they were rewitten in machine language. On page 20, Don says, "For a SIXTH rate program—Write it in Pascal. For a FIFTH rate program-Write it in BASIC. For a FOURTH rate program—Write it in BASIC, but use a few PEEKS and POKES. For a THIRD rate program—Write it in BASIC, but use the CALL command to link a few short machine language code segments. For a SECOND rate program—Write it in BASIC, but use the "&" command to link several longer blocks of machine language code. For a FIRST rate program—Do the whole thing in machine language like you should have done in the first place."

The first half of the book is the theory part that explains assemblers and how to use them. Some of the topics covered in part one are: types of assemblers, how assemblers work, mnemonics, the miniassembler, macros, labels, disassemblers, sorce code file formats, operands, pseudo-ops, programming style, assembler commands, assembly listings, and debugging. The second part of the book contains practice modules of working assembly language code for a file based printer, imbedded string printer, monitor time delay, multiple sound effects generator, musical songs, how to use menu options, random numbers, and a fast random exchange method of rearranging an array of numbers.

In both parts of the book Don includes thorough explanations and tells you why certain things should be done instead of just giving you rules to follow blindly. Each person will find different parts of the book more helpful than others because of differing interests, but I especially appreciated the information on the COUT, COUT1, and WAIT monitor routines. I do disagree with Don on the choice of an assembler, as he likes the new EDASM in the Apple Tool Kit while I prefer the S-C Macro Assembler, but this is a matter of personal style and taste. Many books on Assembly Language contain numerous pages of op codes which duplicate those found in all the other books, but this book concentrates on how to use the op codes and you'll have to refer to some other manual for the listings.

This book should be very helpful to anyone interested in learning to program in assembly language or in improving their assembly language programming skills.

CBASIC User Guide
By Osborne, Eubanks, and Mcniff
Published by Osborne McGraw-Hill
630 Bancroft Way
Berkeley, CA 94710
215 Pages, 7" x 9", $17.95

Although I am interested in learning and using many different programming languages, I felt that one BASIC was enough for anyone. There are minor differences between the various BASIC dialects but they all perform about the same, and since I received Applesoft® with my Apple II® and MBASIC® with my CP/M® system, I wasn't interested in another variation of the language. When Ernie Brooner started the BASE series I argued that it should be written in MBASIC, but Ernie insisted that CBASIC® was a better language for this application—and he convinced me that it has many capabilities not found in other versions of BASIC and is more suitable for writing a database program. So, I started searching for a book that had the nuts-and-bolts information that I needed.

The **CBASIC User Guide** starts with a simple explanation of the language, which is written with an editor, compiled, and then interpreted when run. It gives step-by-step examples of entering a program with the ED text editor on the assumption that everyone who uses CP/M also has ED, but most of us will use a more complete word processor such as WordStar if we have it available. Next is a chapter on numerical data, string data, constants, variables, and arrays, with a very good explanation of the various data types.

Subsequent chapters continue with additional useful information on how the functions and statements are used in programming, which is much more useful than just a description of the functions and statements. The contents of the book are as follows:

- **Chapter 1 Beginning With CBASIC.** CBASIC—How it Works and What it Does, First Session: Editing a CBASIC Program.
- **Chapter 2 Data Types.** Numeric Data, String Data, Constants and Variables, Assigning Values to Variables, Arrays.
- **Chapter 3 CBASIC Program Organization.** Modular Programming, Flowcharting, Program Structure, CBASIC Statement Format, Line Numbers, The REMARK Statement, The STOP and END Statements.
- **Chapter 4 Simple Data Input And Output.** The INPUT Statement, The PRINT Statement, The DATA Statement, The READ Statement, The RESTORE Statement.
- **Chapter 5 Arithmetic and Numeric Operations.** Arithmetic Calculations, Comparisons and Boolean Operations, Summary of Expression Evaluation Hierarchy, Numeric Functions, User-Defined Functions.
- **Chapter 6 String Operations.** String Concatenation, Relational Operators, String Functions, User-Defined String Functions.
- **Chapter 7 Program Logic.** The GOTO Statement, The Computed GOTO Statement, The IF Statement, The FOR and NEXT Statements, The WHILE and WEND Statements.
- **Chapter 8 Modular Programming Features.** Subroutines, Multiple Line Functions, Program Overlays and Chaining, CBASIC Library Features.
- **Chapter 9 Input Programming.** Structuring Input Programs, A Mailing List Data Entry Program, More Advanced Input Programming, The CONSTAT% and CONCHAR% Functions.
- **Chapter 10 Output Programming.** The LPRINTER Statement, The CONSOLE Statement, The TAB Function, The POS Function, Formatted Printing: the PRINT USING Statement, Console Output Programming, Printer Output Programming.
- **Chapter 11 File Handling.** CBASIC File Organization, File Names, Using Files in a CBASIC Program, Accessing Open Files, PRINT# and PRINT USING# Statements with Files, Other Disk Statements and Functions, File Handling Program Examples, Appending Data to Sequential Files, CBASIC Data File Conventions, A Mailing List File Handling Program.
- **Chapter 12 CBASIC Runtime Organization.** The Structure of Memory at Runtime, Format of Variables, Interfacing with CP/M, Chaining.
- **Chapter 13 The Machine Interface.** The POKE Statement, The PEEK Statement, The CALL Statement, Locating Variables in Memory, the SADD and VARPTR Functions, Interfacing Assembly Language Routines with CBASIC Programs.
- **Chapter 14 CBASIC Statements.** Terms and Abbreviations used, Alphabetical List of CBASIC Statements.
- **Chapter 15 CBASIC Functions.** Alphabetical List of CBASIC Functions.
- **Chapter 16 Using CBASIC.** Command Line Options, Command Line Toggles, Compiler Directives, Tracing Execution of a CBASIC Program.
- **Appendix A.** Error Messages and Codes.
- **Appendix B.** Conversion Tables.
- **Appendix C.** Key Words.

This book covers a lot of ground, considering its modest size. Some of the descriptions are rather brief—it is not a programming textbook, but it does contain a wealth of useful information on how to use the language in addition to the usual lists of definitions. The book itself is so helpful that I'll probably use CBASIC instead of MBASIC for some programs just because of it. ■

COMPUTER® TRADER MAGAZINE

★ ★ ★ **LIMITED TIME OFFER** ★ ★ ★

BAKER'S DOZEN SPECIAL!

$12.00 for 13 Issues

Regular Subscription $15.00 Year

Foreign Subscription: $55.00 (air mail)
$35.00 (surface)

Articles on MOST Home Computers, HAM Radio, hardware & software reviews, programs, computer languages and construction, plus much more!!!

Classified Ads for Computer & Ham Radio Equipment

FREE CLASSIFIED ADS
for subscribers

Excellent Display and Classified Ad Rates
Full National Coverage

CHET LAMBERT, W4WDR
1704 Sam Drive • Birmingham, AL 35235
(205) 854-0271
Sample Copy $2.50

THE STATE OF THE INDUSTRY

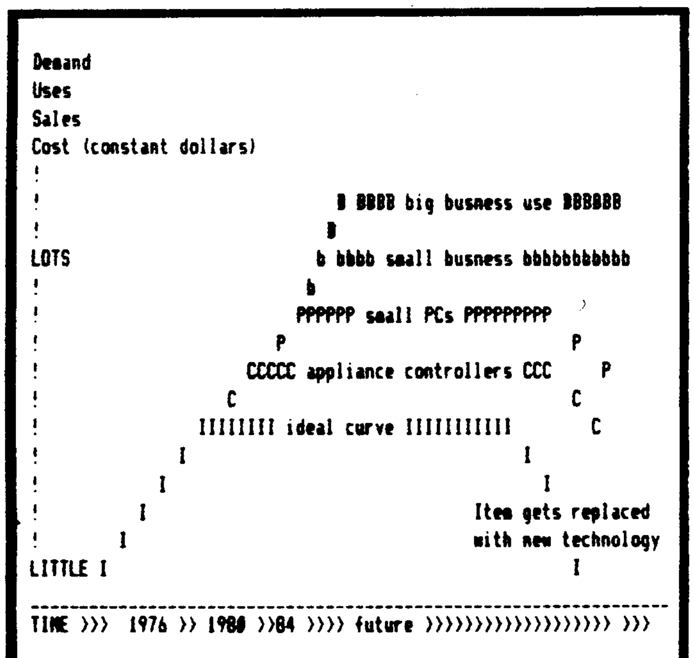
by Bill Kibler

Every once in a while I like to depart from my usual detailed discussions of hardware and system integration and cover some different ground. In this article I will take a look at some of the things I see happening in the computer industry. A lot seems to be going on with hardware and software that could affect how people will be doing computing in the years to come.

IBM Number One?

One company with which I do *not* have a love affair is IBM. There are many reasons for this position, and I guess the most important one is their lack of understanding of the common user. They sent the industry into upheaval with their PC, and have done it again with their newest upgraded unit. To understand what is happening, a little history is needed, so sit back and read on.

The computer industry started with the Intel 8080 processor used by MITS and IMSAI. These first units to hit mass acceptance started hobbyists and professionals thinking small, while IBM and others were still thinking "bigger is best." The Z80 came along and machines became faster and smarter, while still staying in 64K of memory. The 80868 was already on the market, but so much energy was being put into making the 8 bit systems run better that most people were ignoring 16 bit machines. Most users at this time were the brave and experimenting type. Potential business users were waiting for IBM to come out with a 16 bit system, even when the machines and software needed to do their work were already available.



```
Demand
Uses
Sales
Cost (constant dollars)
!
!                              B BBBB big busness use BBBBBB
!                             B
LOTS                         b bbbb small busness bbbbbbbbbbb
!                           b
!                          PPPPPP small PCs PPPPPPPPP
!                        P                             P
!                     CCCCC appliance controllers CCC     P
!                   C                                  C
!                IIIIIIII ideal curve IIIIIIIIIII       C
!              I                                  I
!            I                                      I
!          I                                 Item gets replaced
!        I                                   with new technology
LITTLE I                                               I
-----------------------------------------------------------------
TIME >>>  1976 >> 1980 >>84 >>>> future >>>>>>>>>>>>>>>>>>>> >>>
```

Figure 1

When IBM finally realized that a large part of the business community was waiting for them, they hired some people to produce a new machine, one not compatible with anything. I personally think it was pretty much a joke on us, the users. I believe IBM really wanted to sell big systems and thought that a poorly designed PC would convince the big buyer to stick with BIG BLUE. The trouble is, the joke is on everyone in the industry; the IBM PC has been a big success and has created another standard. After laughing all the way to the bank for so long, IBM has finally realized what they created, and are now being forced to regroup and try to correct their design problems.

Their new AT unit is a halfway step toward a machine that can perform the functions for which people really want a computer. This is not an endorsement of the AT, in fact it is just the opposite. The AT will not run most of the older PC's software or hardware because of the original PC's poor design and its lack of upgradability. The old expansion slots could not address enough memory, so a new bus has been added to the old one, making most of the PC expansion cards unusable with the AT. Software ROMs are different, so most of the old programs which had to make direct calls around the operating system to work are not usable. Major changes are needed for IBM to meet the needs and promises of the automated office concept which they and other companies have been working toward for so long.

What Technology Next?

My personal feeling is that the next new technology will proably have an IBM name on it. As much as I dislike it, big buyers and industrial users are now controlling most of what gets produced and sold. The automated office concept requires a lot of memory and interfacing abilities, and special processors like the 80286 in the AT. These sophisticated interfaces require a lot of software and expensive hardware—look out for the big price tags. I personally do not see the trend changing from the current emphasis on business users. By this I mean that most of the published demands and directions of the industry will be in the office area and will ignore the small user whose demand has leveled off. Don't get me wrong—the small user will continue to buy systems, but not at the increasing rates we have seen in the past (expect constant 5 or 10% growth, while some business systems may see 50 to 150% growth).

For the small user or specialist this will mean that prices on single user Z80 based units will continue to drop. More super machines like the AMPRO single board unit will become available. One can now find Sinclair 1000s going for under $20, with a new 68000 unit soon to be released for $500. Soon there will be many systems for $800 that have dual drives, 64K or more of memory, and lots of software. Some of this is due to market changes, but much of it is caused by the failures of several companies. There are many reason why companies are going under, but one of the main ones is poor management. Started by people without any previous sales or management skills, they were shaky at their best. What IBM did to these companies was to push them over the edge into disaster. However, what they did not do is change the market's positioning. There will always be room for good, small companies that can produce a product at a reasonable cost for a special need. An example would be a controller card that has a $100 profit margin. If you sold only 300 a year, $30,000 a year income for one person is not bad!

Marketing

When I studied marketing in college there were several positions that I had trouble agreeing with, but have since found to be true. One was the different phases which a product goes through, commonly called "product life cycle" (see Figure 1). This cycle is composed of different levels at which demand and consumer awareness will affect the product. I feel we have just started peaking or flattening out in the product cycle. When a product is first introduced, there is a limited market consisting of the brave or experimenters. This is followed by the "me-toos" who buy because they do not want to be left behind. Other names being used for these two first groups are "nerds" and "yuppies" respectively. That leaves the average American in the middle. This average group is the largest by far, but is also the hardest to sell to. There is another group which is seldom considered because of their stance of absolute fear and reluctance. They can be classified as the "never under any circumstance" group.

The nerds bought MITS and IMSAI, and are still building the leading edge systems. They like buying used parts and making systems of them, and are always trying some new software technique. If you have not guessed by now, this description matches me to a tee. Oh well, I guess being a nerd isn't so bad. The yuppies now buy IBM PCs or clones and usually get canned software. They like playing with the machines, but use them mostly for money oriented tasks. This group is looking more at what the machine can do for them than how they can use it, but are still not seeing it as much more than a fancy toy.

The current group of buyers is by far the largest, and can be broken down into several subgroups. These smaller groups are really just variations on the reasoning or buying power they have. I guess the most descriptive name for this group would be the Missouri state motto, "show me." This group wants hard, clear facts that prove that a system will not only meet their needs but also save them money. It is at this point that the future of the industry is being decided. If marketing can convince the user of the true worth of the machine, many more will be sold, and we will maintain a plateau. Should the viability be questioned too much, this level will become a peak and demand will start falling. This plateau is usually described as the "long hard pull," where the routines of sales and production stabilize (you hope).

I feel that the reality of the situation is that the market is now going in many different directions. In the beginning part of the curve, all uses of computers were being helped by each other. An advance in robotics helped sell more work stations and vice versa. Now, the public and the users are much more educated and can see the difference between the system's intended use and their own needs. It is no longer a general belief that one computer will solve all problems, which makes me very happy. There are now several uses for computers, and each has its own future on the horizon. The very small processors are going into all electrical appliances and will continue to do so forever. Industrial process controllers are becoming more stabilized, but some more defining of standards is needed. Look for more growth yet in the

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

THREE TOUGH QUESTIONS

WITH ONE EASY ANSWER:

1. **WHEN IS A COMPUTER LANGUAGE NOT A LANGUAGE?**
   MMSFORTH includes DOS, Assembler and high level commands and extraordinary utilities, extends to become any other language (or application), is an interpreter and a compiler, and is remarkably fast and compact!

2. **WHICH SOFTWARE RUNS THE SAME DISKS IN IBM PC AND TRS-80 MODEL 4?**
   MMSFORTH disks run on those and Compaq, and TRS-80 Model 3, and Tandy 1200, and TRS-80 Model 1, and AT&T 6300, etc., with your choice of formats up to 200K single-sided or 400K double-sided!

3. **WHO OFFERS SOURCE CODE WITH ITS LANGUAGE, UTILITIES, DATABASE, WORD PROCESSOR AND COMMUNICATIONS SOFTWARE?**
   Nearly all MMSFORTH software includes source code.

MMSFORTH

All the software your computer may ever need.

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

- Personal License (required):

MMSFORTH System Disk (IBM PC)	$249.95
MMSFORTH System Disk (TRS-80 1, 3 or 4)	129.95

- Personal License (optional modules):

FORTHCOM communications module	$ 39.95
UTILITIES	39.95
GAMES	39.95
EXPERT-2 expert system	69.95
DATAHANDLER	59.95
DATAHANDLER-PLUS (PC only, 128K req.)	99.95
FORTHWRITE word processor	175.00

- Corporate Site License Extensions .... from $1,000
- Some recommended Forth books:

UNDERSTANDING FORTH (overview) ...	$ 2.95
STARTING FORTH (programming)	18.95
THINKING FORTH (technique)	15.95
BEGINNING FORTH (re MMSFORTH) ...	16.95

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

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

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

industrial sector.

Data processing systems are going three ways; personal, business, and professional. The personal units are, and will be for some time, either Z80, Apple, or IBM PC clones below the $2000 mark for full systems. These units will be characterized as single user systems, combined with software for either personal or small business use. I feel that a leader and a perfect example of such a system is the Kaypro, which still holds an excellent market position. The true business group is represented by systems like the Comupro 10 and Teletek multiuser units. Although there are many other manufacturers of these styles of systems, Compupro and Teletek represent a concept that will continue to grow as the market matures. These units provide a growth path for a business, where they can add more power and users without affecting other user's speed or performance. This is done by giving each user their own processor and then sharing only disk access between the units. This concept works great within the needs of medium size businesses and will be more cost effective as time continues.

This last group of big business and professional users has large data needs. In fact, their needs are so large that previously only large mainframes or minis fit their bill. Now, however, the tasks are getting so varied and involve so many users that even the bigger systems are slowing down under the load. This calls for off-loading some of the tasks, like word processing, onto smaller independent work stations, and leaving the main system for large data bank problems or message handling. This group could be the largest buyer yet, and is the market for which most companies are building new products. Along with technical quality, these buyers also expect support, the real hand holding type, which takes large business structures to do. This is definitely the home of IBM, DEC, and WANG.

The Survivors

The next year or two will see some shaking-out in the industry. I really hate to say this, as a lot of media time has been spent on the subject already. This shakeout, however, is probably not what most expect, as IBM will not be a clear victor. The survivors will be those that have established a good relationship with their select group of users. Apple and Kaypro are two such companies, and will keep the small (personal) user. Some S-100 makers who have their products in specialized systems and are not over-capitalized will also be stable. These companies have good products with excellent support, a long track record, and most importantly, a good public image.

I don't think that the market for medium and large users is anywhere near as stable. The medium user market should settle down over the next year or two as those companies on shaky ground disappear. The products will improve, but mostly in options or software, with little change in their single user approach to multiuser configurations.

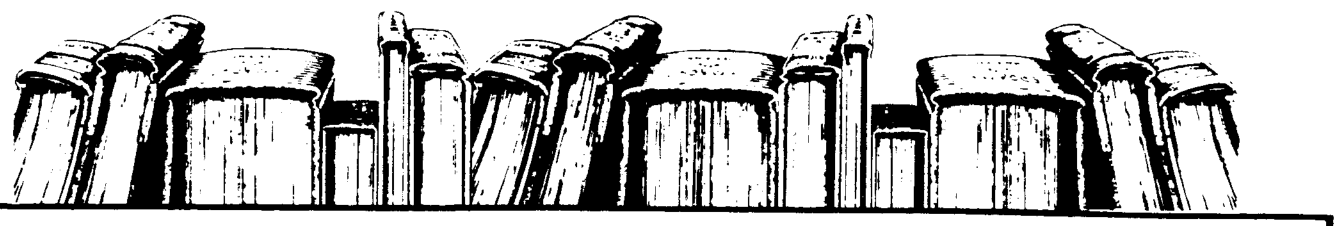
The major user area is just starting the long slow growth pattern which is so typical of this market. This group would usually be called the "Big Boys", with IBM at the top. They have to be big to be able to sustain long development projects. Large, complex software and hardware projects may take two to three years of development and testing before they see public exposure. The technology is just now beginning to appear for some of the needed interfaces. The lack of standards is slowing things down and is keeping many of the smaller firms from participating. Right now, this and the middle group are somewhat together, but look for a more clear separation to occur over the next year. IBM just raised their document prices for the XT from $50 to $285, a jump that will force small business to look elsewhere for support. The biggies know the true cost of support and will start charging accordingly. Becuse of this change the smaller companies that have shifted support to retail outlets and clubs will pick up considerable business (and lose the corporate accounts).

Where To Next?

Let's start bringing this article to a close by talking about what you, the user, should consider doing. My first advice is to keep your Z80 machines, especially if they have been meeting your needs. The new 16 bit units will not improve your abilities by a large enough margin to warrant their cost. If you are buying for the first time and want something new, consider a dual processor so you can run public domain software and the newer programs. Let's not forget the used systems, as real bargains can be had there. I dont feel that IBM PC compatiblity will be a big issue much longer. New DOSs will soon start requiring that all software go through it and thus make any MSDOS system compatible. DRI is not to be forgotten, as their concurrent CP 86 may yet gain some real interest. The basic designs of both industrial and business systems will not really change much, but will continue to stabilize and decrease in size. These changes will appear to the average user as major strides in technology, but in reality, only amount to combining older products in newer packages. The use of LSI or special chips is definitely on the rise, which makes me feel strongly that size will be the major change in future systems.

I must admit that one of the reasons for writing this article was my recent decision to sell my Z-100 computer. After six months of use, I still did over 95% 8 bit processing, mainly writing articles. I knew full well that I could buy a used system that would meet my needs for far less than the $2000 plus I paid, but orginally I had thoughts of big changes in the 16 bit software market. It now appears to me that 16 bit is going big time and is not for the little guys like me. I also decided to sell before the market started shifting and prevented me from getting most of my money back. A friend has agreed to buy it for just under full price, and I have a system in line for $500. The Z-100 is probably one of the better systems, and I still recommend it highly. What I recommend, however, is that you try it before you buy it. Several places are offering rental systems, and I would try that before you invest money in a system. For business users, get a list of other customers, and visit them to see how their system is working before you put your money down.

In summary, I would like to say that the industry has become more aware of what they must do in order to succeed.The best buyer is an educated one, and educating users is where the market is headed. ■



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 outer-layer utilities. By Mitchell Waite. Approximately 160pages, 8x9½, comb. ©1983. . . . . $18.95

The Programmer's CP/M Handbook

An exhaustive coverage of CP/M-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, 7½x9¼, softbound. $21.95

Interfacing to S-100 (IEEE 696) Microcomputers

This book is a must if you want to design a custom interface 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, 6½x9¼, 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. Auslander and P. Sagues, 310 pages, 7 3/8x9 1/4, softbound. $16.95

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, 5½x8½, 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, 6¼x9¼, 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 III and Richard Shoemaker, 288 pages, 6¼x9¼, 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 I/O, 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. $16.95

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 5¼x8¼, softbound. $20.00

TTL Cookbook

Popular Sams author Dan Lancaster gives you a complete look at TTL logic circuits, the most inexpensive, most widely applicable form of electronic logic. In no-nonsense language, he spells out just what TTL is, how it works, and how you can use it. Many practical TTL applications are examined, including digital counters, electronic stopwatches, digital voltmeters, and digital tachometers. By Don Lancaster, 386 pages, 5½x8½, soft. ©1974. . . . . $12.95

The Computer Journal

PO Box 1697 Kalispell, MT 59903

Order Date: ________________________

Print Name ________________________

Address ________________________

City ____________ State ______ Zip ______

☐ Check ☐ Mastercard ☐ 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	

Lowering Power Consumption In 8″ Floppy Disk Drives

by Tony Gambacurta and Jim Chamberlain

Introduction

In most computer systems, the floppy drives spend much of their time doing nothing but consuming power and generating heat while the user at the keyboard is editing text or using a spreadsheet. The result of this is heat buildup, noise, and unnecessary wear of the drive and media.

This article deals with reducing power consumption by shutting off the power to the stepper motor and the spindle motor. Our modification should substantially reduce power usage, increase system reliability and provide a welcome relief to the constant sound of needlessly spinning disks. On a system with two Qume Data Track 8″ drives, the power may be reduced by over 90 watts when the drives are not being used. If you don't think this is a significant amount of heat, hold your hand near a 75-100 watt light bulb. The constant spinning of the spindle motor causes bearing wear as well as degradation of media.

Example System

The system we will discuss consists of two Qume Data Track 8″ drives and an InterSystems FDC-II floppy controller which uses the NEC upd765 (Intel 8272) controller chip. The methods used should apply to other systems such as the CompuPro Disk 1 with DT-8″ drives.

Stepper Motor

The stepper motor moves the head from track to track. Most drives allow the stepper motor to be energized only when needed by powering it up either when the drive is selected or when the head is loaded (assuming that the controller loads the head before stepping during a seek). If, when the FDC (Floppy Disk Controller) is going to issue step pulses, it selects the drive and keeps it selected for some period (15-25 milliseconds) after the last step pulse, then the stepper motor may be enabled by the drive select. The NEC upd765 doesn't do any such thing because the NEC controller allows simultaneous stepping of up to four drives at once. As a result, it selects the drive, does a step, then may select other drives to issue step pulses to them. Another consideration might be to use the head-load signal to enable the stepper motor, but according to the NEC document tion this cannot be done because the upd765 unloads the head during seeks. Some controller designs, (the InterSystems in particular) do not use the head-load signal that the chip alone supplies. The head-load signal runs into a circuit that keeps the head loaded some time after it has been activated. This doesn't solve the problem of controlling the stepper motor, because if the drive has been doing nothing and the heads are not loaded, when a seek occurs it will do so without loading the heads first.

An undocumented "feature" exists with respect to the use of a signal called RW/SEEK on the NEC upd765 pin 39. This pin is used to multiplex eight signals into four to save pins on the chip. It so happens that this sgnal only goes to SEEK when a seek is performed. All other times, during reading, writing, checking ready status or doing nothing, this signal remains in the RW state.

Logic for minimum motor activity:

1. Combine RW/SEEK with the head-load signal.

2. Put the combined signal through a circuit that stays true for a period after its input goes talse to create a new head-load signal for the drive.

3. Set the options on the drive to enable the stepper motor only when heads are loaded.

This assumes that head loading is in radial mode, which means that the heads are loaded on all drives independent of drive select.

InterSystems FDC-2 Modification

This example uses a spare gate on the floppy disk controller board. It involves one trace cut and three wires. On the NEC upd765, pin 36, HLD (head-load) goes high when the heads are to be loaded. Pin 39, RW/SEEK goes high when a seek is performed. As delivered, pin 36 goes to a 96LS02 one-shot. Going to the same one-shot is a 2mhz clock to keep the one-shot re-triggering while HLD is true. In order to combine HLD and SEEK, we need an OR gate. InterSystems was kind enough to provide a spare OR gate in its FDC-II (rev A) on U10 pins 1,2, and 3.

To perform the modification, cut the trace running from U-36 pin 36 to U-24 pin 5. You will find the trace near U36 on the solder side of the board so the trace cut may be made on the solder side. If you don't feel confident doing trace cuts and adding wires to the board, don't do it. These kinds of changes are for experienced technicians.

After cutting the trace, wire:

U36 pin 39	.. to ..	U10 pin 1
U36 pin 36	.. to ..	U10 pin 2
U24 pin 5	.. to ..	U10 pin 3

The schematic in Figure 1 shows the result of these changes.

With these changes on the controller, the drive must be changed to enable the stepper only when head-load is true. This will involve reconnecting the shunt HL that was previously broken at position E1 on the main printed circuit board. To reconnect a broken jumper, it is best to replace it with an eight position DIP switch or a component to IC socket adaptor. Programmable shunts are not designed to be soldered to, and melt very easily.

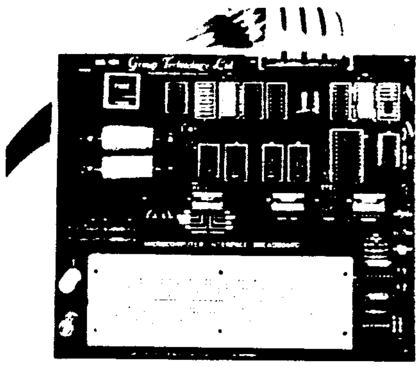
LEARN MICROCOMPUTER INTERFACING VISUALIZE SCIENCE PRINCIPLES

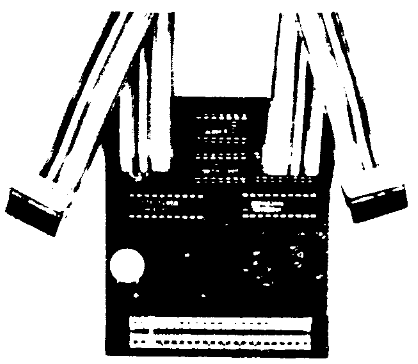
Using GROUP TECHNOLOGY BREADBOARDS with your
APPLE® ...COMMODORE 64® ...TRS-80® ...TIMEX-SINCLAIR® ...VIC-20®

Versatile breadboards and clearly written texts with detailed experiments provide basic instruction in interfacing microcomputers to external devices for control and information exchange. They can be used to provide vivid illustrations of science principles or to design interface circuits for specific applications. Fully buffered address, data, and control buses assure safe access to decoded addresses. Signals brought out to the breadboards let you see how microcomputer signals flow and how they can be used under BASIC program control to accomplish many useful tasks.

Texts for these breadboards have been written by experienced scientists and instructors well-versed in conveying ideas clearly and simply. They proceed step-by-step from initial concepts to advanced constructions and are equally useful for classroom or individual instruction. No previous knowledge of electronics is assumed, but the ability to program in BASIC is important.

The breadboards are available as kits or assembled. Experiment component packages include most of the parts needed to do the experiments in the books. Connecting cables and other accessory and design aids available make for additional convenience in applying the boards for classroom and circuit design objectives. Breadboard prices range from $34.95 to $350.00





The INNOVATOR® BG-Boards designed by the producers of the highly acclaimed Blacksburg Series of books have gained wide acceptance for teaching microcomputer interfacing as well as for industrial and personal applications. Detailed, step-by-step instructions guide the user from the construction of device address decoders and input/output ports to the generation of voltage and current signals for controlling servo motors and driving high-current, high-voltage loads. BG-Boards are available for the Apple II, II +, IIe; Commodore 64 and VIC-20; TRS-80 Model 1 with Level II BASIC and at least 4K read/write memory, Models III and 4. The books, *Apple Interfacing* (No. 21862) and *TRS-80 Interfacing Books 1 and 2* (21633, 21739) are available separately.

The FD-ZX1 I/O board provides access to the Timex-Sinclair microcomputer for use in automated measurement, data acquisition, and instrument control applications. A number of science experiments have been developed to aid teachers in illustrating scientific principles. The operating manual contains instructions for constructing input and output ports. A complete text of the experiments will be available later in 1984. The FD-ZX1 can be used with Models 1000, 1500, 2068, ZX81, and Spectrum.

The Color Computer Expansion Connector Breadboard (not shown) for the **TRS-80 Color Computer** makes it possible to connect external devices to the expansion connector signals of the computer. Combined with a solderless breadboard and the book *TRS-80 Color Computer Interfacing, With Experiments* (No. 21893), it forms our Model CoCo-100 Interface Breadboard providing basic interfacing instructions for this versatile computer. Experiments in the book show how to construct and use a peripheral interface adapter interface, how to input and output data, and how digital-to-analog and analog-to-digital conversion is performed.

Our new Spring Catalog describes the interface breadboards, dozens of books on microcomputer interfacing, programming, and related topics including the famous Blacksburg Continuing Education Series, a resource handbook for microcomputers in education, and a comprehensive guide to educational software; utility software for the TRS-80, scientific software for the Apple II, and other topics. We give special discounts to educational institutions and instructors. Write for the catalog today.

PUTTING HANDS AND MINDS TOGETHER

Group Technology, Ltd.
P.O. Box 87N
Check, VA 24072
703-651-3153

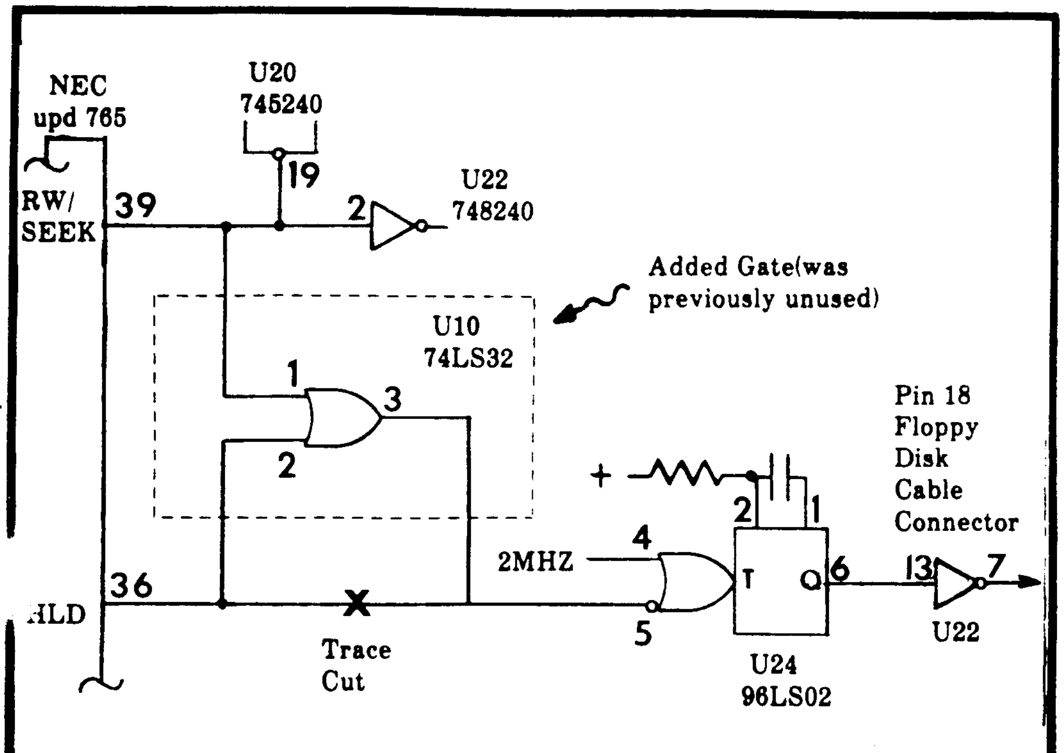


Figure 1: Schematic of section of InterSystems FDC-II showing changes required to combine head-load with SEEK signal.

On the sample system, the Qume DT-8 jumpers relating to this are:

B	=	open
X	=	shorted
C	=	shorted
DS	=	open
HL	=	shorted

The net result should be that the drives now load the heads in radial mode (all drives together) and the stepper motor is enabled when the head is loaded.

Spindle Motor

The spindle motor consumes a lot of power when running. It must, of course, be turning when the disk drive is accessed. The spindle motor should also be turned on when the door is open so that it will be spinning when the door is closed to insure proper centering of the diskette on the spindle. The spindle motor is typically an AC motor and therefore is much more difficult to control with simple logic signals. Optronics Technology makes a small board called the DCU (Drive Control Unit) that takes care of the hardest part, controlling an AC motor. The DCU uses a triac, an opto-isolator and circuitry to control the motor on zero-crossings (of the AC power to minimize surges and eliminate electromagnetic interference). With the DCU, all that is needed is to supply low level control signals to enable the AC spindle motor. The DCU also contains circuitry to allow the spindle motor to continue running for a while after it is enabled, reducing the amount of starting and stopping the drive motor will be required to do. This reduces repetitive start/stop cycles during operations such as copying files.

With Shugart SA-800 and SA-851 drives, the ready signal is latched in the true state after the initial rotations of the disk, and is reset only by door opening or power on. Shugart also provides a door-open signal for turning the motor on when the door is open. The door-open and the head-load signals are all that are required to use the DCU to enable the spindle motor as needed.

Things are not quite as simple with the Qume Data Track 8" drive because it does not supply a door-open signal. There is no switch to detect the door opening. In order to detect disk eject, the write-protect sensor is used. Using the write-protect sensor works fine to turn the motor on, but the motor will remain on if a write-protected disk is installed. There is also the problem of the ready signal going false when the motor turns off. Ready goes false because of the lack of index pulses detected for a period of time. This may be a problem with a system that uses this signal to detect a disk change. For example, CPM+ may be configured to allow an interrupt to set the media flag which BDOS will check on the next disk I/O operation. This is a nice feature because it allows CPM+ to automatically re-log in a disk when it has changed, and as a result, know when to invalidate disk buffers. When this feature is used (it is optional) CPM+ will spend less time checking the disk directory for a changed disk.

In summary, we don't want the system to treat turning off the spindle motors as though a disk has changed.

Qume Modification

To eliminate the problem of the ready signal, the write protect feature is no longer used and we modify the Qume Data Track 8 to use the write-protect sensor for the ready signal. This requires one trace cut and one wire added per drive. This change was made on two DT-8 drives, serial numbers 63157 and 68634. To disable the existing ready signal, cut the trace running from U3F pin 4 to U2E pin 4. This may be done near U3F where the trace connects to pin 4 on the component side of the board. The board does not need to be removed to cut the trace. Connect a wire from U3G pin 5 (or pin 4) to U2E pin 4. This makes the ready signal become a "not write-protect." The schematic in Figure 2 shows the result of these changes.

The next step is to wire the DCU to the drives. In the sample system, one DCU was used to control two drives. The DCU was built from a kit for negative logic. Two inputs are used, one from head-load, and one from write-protect. These signals may be tapped from the cable going to the

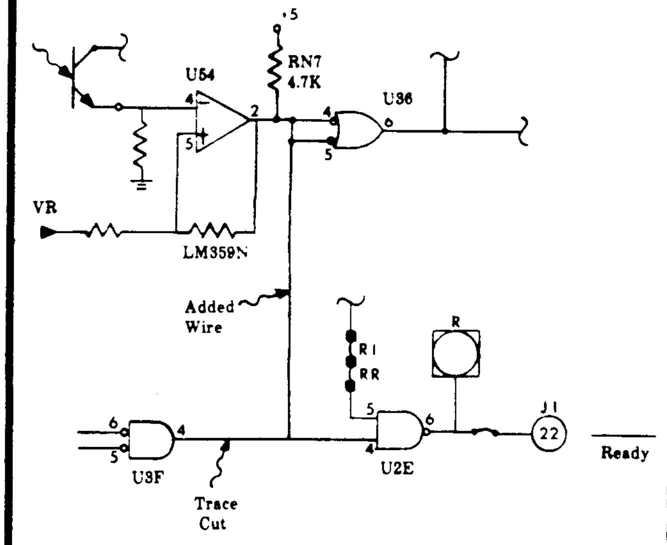


Figure 2
Upper portion can be found in QUME DT-8 schematics page 3.
Lower portion can be found in QUME DT-8 schematics page 2.

drives or on the drives.

Conclusion

The system described here now has two drives that draw 47 watts when idle, and 144 watts when the motors are running with steppers enabled. The power consumed depends on whether the spindle motor is on, whether the heads are loaded (which adds more drag to the spindle motor), and whether the stepper motor is enabled. Prior to making these changes, the fan on the drive cabinet sounded like an F-15 Eagle and it ran HOT. Now the system runs cool, and the jet engine has been removed. The system modification was relatively easy to do and didn't require any additional wires between the drives and the rest of the system. The only other change needed to make the system more quiet and reliable is to replace the daisy wheel printer with a laser printer. ■

Tony Gambacurta is V.P. of Engineering at Applied Research and Technology in Rochester, New York, and Jim Chamberlain is President of Optronics Technology in Pittsford, New York.

The Drive control Unit mentioned is available from Optronics Technology, P.O. Box 81, Pittsford, New York, 14534. The cost is $29.95 for a kit with documentation and $49.95 assembled and tested.

FREE SOFTWARE

RENT FROM THE PUBLIC DOMAIN!

User Group Software isn't copyrighted, so there are no fees to pay! 1000's of CP/M and IBM software programs in .COM and source code to copy yourself! Games, business, utilities! All FREE!

CP/M USERS GROUP LIBRARY
Volumes 1-92, 46 disks rental—$45

SIG/M USERS GROUP LIBRARY
Volumes 1-90, 46 disks rental—$45
Volumes 91-176, 44 disks rental—$50
SPECIAL! Rent all SIG/M volumes for $90

K.U.G. (Charlottesville) 25 Volumes—$25

IBM PC-SIG (PC-DOS) LIBRARY
Volumes 1-200, 5¼" disks $200

174 FORMATS AVAILABLE! SPECIFY.

Public Domain User Group Catalog Disk $5 pp. (CP/M only) (payment in advance, please). Rental is for 7 days after receipt, 3 days grace to return. Use credit card, no disk deposit. Shipping, handling & insurance—$7.50 per library.

(619) 914-0925 information,(9-5)
(619) 727-1015 anytime order machine
Have your credit card ready! VISA, MasterCard, Am. Exp.

Public Domain Software Center
1533 Avohill Dr.
Vista, CA 92083

BASE
A Series on How To Design and Write Your Own Database
By E.G. Brooner

The last column ended with a listing (and brief explanation) of the sample program's opening module. In that section the program checks the directory to see what bases, if any, have been created on that particular disk, and prints their names and the number of records in each. The program then asks the user to choose which base to operate on, and offers a choice of activities. The first time the program runs, of course, there are no bases in existence, and the program creates filespace in which to record those which will be created later. In either case, it then presents the main menu.

The first choice offered by the menu is the option of creating a new collection of data or deleting an existing one. If creation is chosen the user is then permitted to define the file structure from additional sub-menus. We should take this up first, as it is fundamental to the entire program and will help explain the techniques used from here on. Noting the listing provided with the last issue, then, assume that we have chosen option 1 and are proceeding to design a brand new database file structure.

The job of this section is to name, define, and create a related set of files that will make up a custom-designed filing system, painlessly and easily—in other words, a database.

As this section opens, you are asked to choose a unique name for the base you are creating. This name must be five characters or less in length. The program then runs through its directory to make sure that this is not an existing file name. If it truly is a new database, a set of blank files will be named and created automatically.

The next step is to define the file structure. You will first be asked to state the number of fields that are required, and then to name them and specify a length for each. You might, for example, have named the base "PRSNS" (for Persons) and name the fields "last name," "first name," "address," "city," and "zip." The field lengths have been limited to 20 characters or less in our example, and the number of fields to 12. This was arbitrary on my part and can easily be changed if not satisfactory. There should, though, be some limit to each parameter. After making your parameters known, you will have a chance to modify the structure, which is displayed for your approval. You can also quit at this point if you decide to think it over some more.

When you have approved or corrected your format, the program proceeds to generate a directory for the base on which you are working. You can then make entries. But first, let's talk about this automatic generation of the file names and files. After all, this is the major difference between a program of this kind, and a simple file handling program that is restricted to a single pre-determined structure.

There are several ways to do this; defining parameters from a menu is the commonest method used in software for micros. There are also a number of ways to set up the files themselves. Here (for no particular reason), we chose to put each field in a separate file. You could also create a single file for each database and combine the individual fields into a single record. This might be necessary with some BASICs. It would also require extensive changes in the rest of our sample program.

But, as we are doing it this way, we will have to have a named file for each field of each separate database. If we specify five fields, there will be five data files associated with that base in addition to some other miscellaneous entries that we'll mention later.

If you called your base PRSNS, it will become BPRSNS on the directory. The first field will be stored in BPRSNS1.DAT, the second in BPRSNS2.DAT, and so on. BPRSNS.EXT will keep track of the number of entries, BPRSNS.SRT will later be designated as a key file (there may be several for each base) and BPRSNS.PRT will eventually be formatted to direct the printout of the data. The new base will also be added to the main sub-directory for the disk under the file BBASE.DEF. The important thing to note is that all of these files and entries are done automatically from your few simple menu selections.

At this point, too, we had best explain CBASIC's® syntax for those not familiar with it. It does differ from some of the more common BASICs, and I personally like the differences. I beg the indulgence of those readers already familiar with these things.

The lack of line numbers was mentioned earlier—we use numbers only if a line will be referenced, as in a GOTO statement.

Long variable names might also seem strange. CBASIC variables can be up to 32 characters in length. In a long program this is an aid in remembering what you are doing. A total of dollars, for example, can be called DOLLAR.AMOUNT rather than X or D.

We also distinguish between integers (X%) and real numbers (X). This is not necessary, but does speed execution of such integer-based systems as loop indices. Thus we refer to NBR.FLD% when counting the number of fields, and NAME$ when talking about a series of string variables.

Finally, we can create a file name as follows: we input the name PRSNS, concatenate that with "B" and .PRT and end up with a file name (for example) of BPRSNS.PRT. A file name so created by the program will result in an active directory entry of that name into which we can deposit data.

```
REM     ####### ######## #######
REM     #CREATE/DELETE MODULE#
REM     #######################

2000    INPUT "NAME OF DATABASE, 5 CHAR OR LESS";BASE$
        IF LEN (BASE$)>5 THEN BASE$=LEFT$(BASE$,5)
REM     CHECK IF FILENAME EXISTS
        DELETED%=0
        FOR X%=1 TO 100                     REM CHECK DIRECTORY
                IF END #17 THEN 2050        REM FOR DUPLICATION
                READ#17,X%;BBASE$
                IF BBASE$=BASE$ THEN GOSUB 2500
2040    NEXT X%

        IF DELETED%=1 THEN 1300
2050    GOSUB 9999                    REM THEN DESCRIBE THE RECORD
        INPUT "NEW DATA BASE. CONTINUE? (Y/N)";NEW$
        IF NEW$<>"Y" THEN 1300        REM STILL TIME TO BACK OUT
        PRINT "YOU CAN DEFINE UP TO 12 FIELDS PER RECORD"
        INPUT "HOW MANY FIELDS? ";NBR.FIELD%
        IF NBR.FIELD%>12 THEN 2050
        PRINT:PRINT "NEW DATABASE ";BASE$;" WITH ";NBR.FIELD%;" FIELDS"
        INPUT "IS THIS CORRECT (Y/N)";CORRECT$
        IF CORRECT$<>"Y" THEN 2000
        FOR X%=1 TO NBR.FIELD%        REM DETAIL OF FIELDS
                PRINT "FIELD NUMBER ";X%
2100            INPUT "FIELD NAME, UP TO 12 CHAR ";FIELD.NAME$(X%)
                IF LEN(FIELD.NAME$(X%))>12 THEN 2100
2200            INPUT " FIELD LENGTH (20 OR LESS) ";L%
                IF L%>20 OR L%=0  THEN 2200
                FIELD.LENGTH%(X%)=L%
        NEXT X%

REM     DISPLAY THE FILE FORMAT FOR APPROVAL
        PRINT"DATABASE "; BASE$
        PRINT
        FOR X%=1 TO NBR.FIELD%
                PRINT FIELD.NAME$(X%), FIELD.LENGTH%(X%)
        NEXT X%
        PRINT
        INPUT "Y= APPROVED, N= NOT, Q= ABANDON ";APPROVE$
        IF APPROVE$="Q" THEN 1300               REM NO HARM DONE
        IF APPROVE$<>"Y" THEN 2000              REM TO MAKE CHANGES
                                                REM OR ELSE PROCEED
        FILE16$="B"+BASE$+".PRT"
        PRINT BASE$; ".PRT FILE"
        CREATE FILE16$ RECL 6 AS 16             REM CREATING FILES
        FOR X%=1 TO 12
                PRINT #16;0
        NEXT X%
        CLOSE 16
        FILE19$="B"+BASE$+".DEF"                REM NAMING THEM
        FILE20$="B"+BASE$+".EXT"
        CREATE FILE19$ AS 19
        PRINT BASE$;".DEF"
        CREATE FILE20$ RECL 5 AS 20
        PRINT BASE$;".EXT"
        PRINT #20,1;0

        FOR X%=1 TO NBR.FIELD%
                PRINT #19;FIELD.NAME$(X%),FIELD.LENGTH%(X%)
        NEXT X%

        FOR X%=1 TO NBR.FIELD%
                PRINT BASE$+STR$(X%);".DAT"
                FILE$="B"+BASE$+STR$(X%)+".DAT"
                RECL%=LEN(FIELD.NAME$(X%))+5
                CREATE FILE$ RECL RECL% AS X%      REM A DATA FILE
                CLOSE X%
        NEXT X%

        READ #18,1;LAST%
        PRINT #18,1;LAST%+1
        PRINT #17,LAST%+1;BASE$
        CLOSE 17,18,19,20
        GOTO 1000                               REM NOW READY FOR USE

2500    PRINT "DELETING DATABASE ";BASE$        REM HERE DELETING ONE
        INPUT "(Y/N) ";DELETE$
        IF DELETE$<>"Y" THEN RETURN
REM     DELETE FILES ASSOCIATED W/THIS BASE
REM     FILE 17 IS BASES.DEF
        IF END #17 THEN 2600
        FOR X%=1 TO 100
                READ #17,X%;DATUM$
                IF DATUM$=BASE$ THEN DATUM$=""
                PRINT #17,X%;DATUM$
        NEXT X%
        PRINT "BASE NAME ";BASE$;" REMOVED FROM DIRECTORY"
2600    OPEN "B"+BASE$+".DEF" AS 1
        IF END #1 THEN 2700
        FOR X%=1 TO 12
                READ #1;DUMMY$,DUMMY%
                END.OF.LIST%=X%
        NEXT X%
2700    PRINT"DELETING ";BASE$;" .DEF FILE":DELETE 1
        OPEN "B"+BASE$+".EXT" AS 1
        PRINT "DELETING ";BASE$;".EXT FILE":DELETE 1
        OPEN "B"+BASE$+".PRT" AS 1
        PRINT "DELETING ";BASE$;" .PRT FILE":DELETE 1

        FOR F%=1 TO END.OF.LIST%
                IF END #F% THEN 2800
                OPEN "B"+NAME$+STR$(F%)+".DAT"  AS F%
                PRINT "DELETING DATA FILE ";F%
                DELETE F%
        NEXT F%
2800    RETURN              REM THAT BASE NO LONGER EXISTS
```

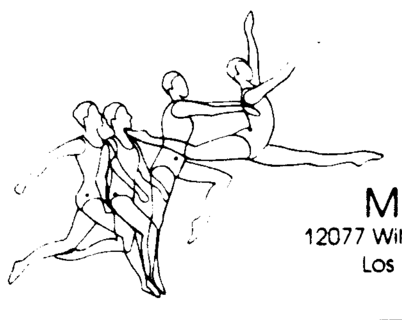
■

MicroMotion

MasterFORTH

It's here — the next generation of MicroMotion Forth.

- Meets all provisions, extensions and experimental proposals of the FORTH-83 International Standard.
- Uses the host operating system file structure (APPLE DOS 3.3 & CP/M 2.x).
- Built-in micro-assembler with numeric local labels
- A full screen editor is provided which includes 16 x 64 format, can push & pop more than one line, user definable controls, upper/lower case keyboard entry, A COPY utility moves screens within & between lines, line stack, redefinable control keys, and search & replace commands.
- Includes all file primitives described in Kernigan and Plauger's Software Tools.
- The input and output streams are fully redirectable.
- The editor, assembler and screen copy utilities are provided as relocatable object modules. They are brought into the dictionary on demand and may be released with a single command.
- Many key nucleus commands are vectored. Error handling, number parsing, keyboard translation and so on can be redefined as needed by user programs. They are automatically returned to their previous definitions when the program is forgotten.
- The string-handling package is the finest and most complete available.
- A listing of the nucleus is provided as part of the documentation.
- The language implementation exactly matches the one described in FORTH TOOLS, by Anderson & Tracy. This 200 page tutorial and reference manual is included with MasterFORTH
- Floating Point & HIRES options available.
- Available for APPLE II/II+/IIe & CP/M 2.x users.
- MasterFORTH – $100.00. FP & HIRES – $40.00 each
- Publications
  - FORTH TOOLS – $20.00
  - 83 International Standard – $15.00
  - FORTH-83 Source Listing 6502, 8080, 8086 – $20.00 each.



Contact:
MicroMotion
12077 Wilshire Blvd., Ste. 506
Los Angeles, CA 90025
(213) 821-4340

New Products

Speech Recognition for Under $200

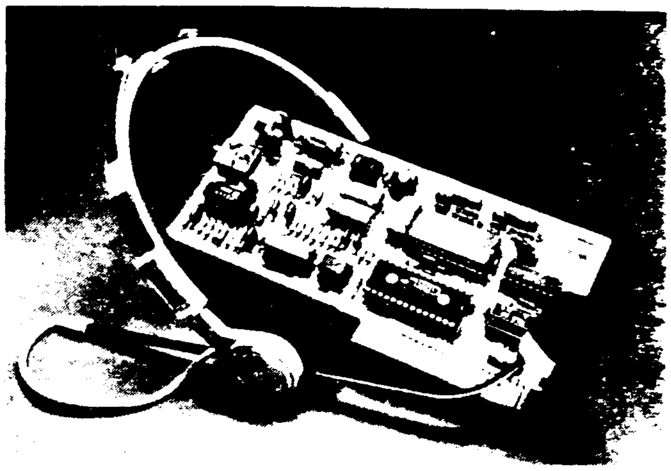
Micromint's "LIS'NER 1000," featuring the SP1000 LPC voice recognition chip from General Instruments, is both a voice recognition and voice synthesizer board for the Apple][and Commodore 64 computers. It can also be used with other 6502-based systems. The LIS'NER 1000 is designed for speaker dependent unconnected speech applications.

LIS'NER 1000 functions as a parallel voice entry device to the keyboard and is transparent to the operation of the computer. When LIS'NER 1000 hears a word it recognizes, it sends a preprogrammed sequence of characters to the keyboard input handler as if it has been typed. Saying the word "catalog" for example, will result in the directory being displayed just as if you had typed C-A-T-A-L-O-G and RETURN. DOS words, keyboard commands, and numbers can be spoken rather than typed.

In addition to speech recognition, the LIS'NER 1000 is capable of LPC (Linear Predictive Coding) speech output from a precoded word dictionary. The Apple][LIS'NER 1000 optionally accomodates an SSI 263 phonetic speech synthesizer chip with text to speech algorithm. The combination facilitates true hands-off speech I/O.

LIS'NER 1000 analyzes the voice input in real time using advanced techniques which operate directly on the incoming data stream. Each word is condensed and sorted as a unique template. The LIS'NER 1000 is supported by the host microcomputer and driver software which compares what LIS'NER "hears" with the speech templates stored in memory.

The LIS'NER 1000 comes with an ultra-lightweight, headset-style electret microphone. The price is $149 for the Commodore 64, and $189 for the Apple][. The Apple][LIS'NER 1000 with phonetic speech synthesizer chip is $259. To place an order, call 1-800-635-3355, or write to Micromint, Inc., 25 Terrace Drive, Vernon, CT. ■



3.5" DRIVES

WHY GO 3.5" ? THEY ARE SMALL, FAST, LIGHT, LOW POWER, COMPATIBLE WITH 5.25" DRIVES (ON THE SAME CABLE), AND THE DISKS AND DRIVES ARE MORE RUGGED AND RELIABLE. MITSUBISHI HAS A RECORD FOR BUILDING THE FINEST QUALITY DISK DRIVES

COMPATIBLE WITH IBM PC, PC COMPATIBLES, RADIO SHACK, or ANY SYSTEM NOW USING STANDARD TANDON OR SHUGART TYPE 5.25" DRIVES

- ☐MF351 SS 360K w/manual,connectors,disk $215
- ☐MF353 DS 720K w/manual,connectors,disk avail soon
- ☐**Case and Power supply** (built in spike protection) avail soon
- ☐Cable (two drive ribbon cable) $15

MITSUBISHI HALF HEIGHT 5.25" DRIVES ALSO AVAILABLE

☐M4853 DS 720K $225 ☐M4854 DS 1.2 Meg $245

FREE SHIPPING, QTY DISCOUNTS, CA RESIDENTS 6%tax
TECHNICAL QUESTIONS WELCOMED!

MANZANA
935 Camino Del Sur
Isla Vista, CA 93117

CHECK, M.O., VISA, M.C.
1-805-968-1387

Temperature Measurement Software

Interactive Microware announces the availability of TEMPSENSE, a thermal data acquisition software package for Apple][series microcomputers and IMI's ADALAB interface card. The TEMPSENSE program is capable of handling up to 64 input channels and is designed to calibrate, compensate, and linearize signals from any of the seven common thermocouple types (B,E,J,K,R,S,T) as well as calibrate and read IC type temperature transducers. Users can also define and acquire data from non-thermal sensor types when needed.

Data values are displayed in either Celsius or Fahrenheit degrees both on the computer monitor and in hardcopy format. Any datum which exceeds the user defined limits for that channel will sound an alarm. Acquired data is automatically buffered to disk in batch files for later recall and manipulation.

TEMPSENSE software is designed to be used with a 48K apple][+ or][e microcomputer equipped with an ADALAB interface card and one to eight 8-chanel ADA-MUX input multiplexers. When thermocouples or other very low-level signals are read, IMI's ADA-AMP instrumentation amplifier must also be purchased. A dot-matrix printer is recommended for hardcopy output.

TEMPSENSE software may be purchased for $100, including disk and instruction manual. The manual is available separately for $5 and is fully deductible if the software is purchased later. A complete, factory-tested 8-channel TEMPSENSE system is available for $2455, including an Apple][e computer, apple disk drive and monitor, EPSON dot matrix printer with interface card, ADALAB A/D-D/A interface card, 8-channel ADA-MUX multiplexer, and TEMPSENSE software.

For additional information, write to Interactive Microware, PO Box 139, State College, PA 16804, or call the sales department at 814-238-8294. ■

"Wiremaster" Software Design Tool

Afterthought Enginering announces Version 5.15 of "Wiremaster," a software tool for design, layout, and construction of electronic hardware. The new Wiremaster runs on most computers, including IBM PC, CP/M, DEC VAX, and MSDOS machines. It generates net lists, wire lists, parts lists, cross-references and checklists to ensure a perfect job. A companion program, "Changemaster," keeps track of modifications and updates, and "Plotboard" produces pictures of the board on a variety of graphics output devices, including most dot matrix printers.

New features of Version 5.15 include location accuracy of 0.001", provision for twisted-pair and coax wiring, input language expansions to easily handle special components such a connectors, and "traveling salesman" path optimization.

Post-processors are available for driving numerically controlled wiring machines, such as Multiwire, Gardner-Denver, Contact systems, and Standard Logic. A PC board layout system with auto-router is under development for first quarter release.

The basic Wiremaster package is available for $195 from Afterthought Engineering, 7266 Courtney Drive, San diego, CA 92111. Updates are available for $25 to owners of earlier versions. ■

Interface Board for the Apple IIe

MetraByte's model APM-08 is an 8 channel 12 bit high speed A/D converter, timer/counter, and 2 channel 12 bit D/A smart board for the Apple® personal computer.

The A/D is a 12 bit successive approximation converter with sample hold and a throughput of 30,000 samples per second in assembly language programming. The 8 channels are single ended with a full scale input voltage of ± 5V with a resolution of 0.00244 mV/bit, and is overvoltage protected to 30 Vdc.

Two channels of 12 bit D/A output, 7 bits of digital I/O, a precision + 10V reference voltage output, external interrupt input, and Apple bus power (+ 5V, + 12V, - 12V) are provided for user designed interface circuits.

The interface can operate in foreground/background, analog and digital I/O and can be used for event counting, pulse and waveform generation, or frequency and period measurements.

The software, located in firmware, is included in the board's purchase price and comes complete with data linearization, graphics, calibration, test, example programs, and a machine language input/output driver to control all board functions.

The model APM-08 comes with a full one year warranty. It is priced at $395, with quantity pricing available. In stock for immediate delivery from MetraByte Corporation, 254 Tosca Drive, Stoughton, MA 02072. ■

QUALITY SOFTWARE AT REASONABLE PRICES
CP/M Software by
Poor Person Software

Poor Person's Spooler $49.95
All the function of a hardware print buffer at a fraction of the cost. Keyboard control. Spools and prints simultaneously.

Poor Person's Spread Sheet $29.95
Flexible screen formats and BASIC-like language. Pre-programmed applications include Real Estate Evaluation.

Poor Person's Spelling Checker $29.95
Simple and fast! 33,000 word dictionary. Checks any CP/M text file.

aMAZEing Game $29.95
Arcade action for CP/M! Evade goblins and collect treasure.

Crossword Game $39.95
Teach spelling and build vocabulary. Fun and challenging.

Mailing Label Printer $29.95
Select and print labels in many formats.

Window System $29.95
Application control of independent virtual screens.

All products require 56k CP/M 2.2 and are available on 8" IBM and 5" Northstar formats, other 5" formats add $5 handling charge. California residents include sales tax.

Poor Person Software
3721 Starr King Circle
Palo Alto, CA 94306
tel 415-493-3735

CP/M is a registered trademark of Digital Research

For Sale or Trade

The following are excess or duplicate programs acquired with our various systems. They are all the original 8" SSSD CP/M disks and documentation.

• **MP/M II** *This is the Digital Research Multiple user operating system to replace the CP/M system. The program is complete with two disks, three manuals, and the registration card. This is an unconfigured system which must be configured for your system.* *$275*

• **Micronix** *This is the Morrow version of UNIX which is configured to run on the Morrow Decision One with a hard disk and 256K RAM. Morrow states that "it is compatible with Bell Laboratories' UNIX Version 6 and users will find many of the same features of the Bell sytems in Micronix." Includes the manual and four disks.* *$300*

• **dBase II** *This is version 2.3B of the famous Ashton-Tate data base program, including the manual, program disk, and demo disk.* *$225*

• **Accounting Plus** *General Ledger and Inventory Version 3.02 with two disks, the reference manual, a training manual, and the registration card. It requires a few minor steps in order to configure it for your system.* *$250*

We are interested in possible trades for these programs. Some of the items we need are: ZSID on 8" SSSD, an 8" double sided drive, an EPROM burner, and a terminal.

The Computer Journal
1103 South Main
Kalispell, Montana 59901