Mr. Robert Lock, Editor
COMPUTE

Dear Sir:

While browsing through the magazine rack of Minnesota Micro Systems in Minneapolis, I came across a new magazine, Compute. I thumbed through it and saw two articles on OSI computers. HURRAY. Someone knows that there are some OSI owners out here. That is more information in one month then other magazines have in a year.

As you might guess, I own an OSI computer. I am a newcomer to computing. I spent many hours researching the various microcomputers to enable me to make a somewhat intelligent decision if and when I bought a computer. There really isn't a whole lot of unbiased information available for the beginner. Much of the printed word is written for the experienced person and is not always understandable to those of us just learning the jargon.

Cost was the prime consideration. It is somewhat difficult to justify a purchase of over five hundred dollars for a hobby interest, especially a new hobby one knows next to nothing about.

I had decided to purchase an ATARI, having been very satisfied with their video game computer for three years. But no one had one for sale and could only say that they would be available shortly. APPLE was a bit expensive, PET didn't have color and had a strange keyboard. Radio Shacks sales people were not any more knowledgable then I was so not much information was gained from them and I didn't consider their system.

While walking through a Montgomery Wards store in Minneapolis I saw a sign pointing to a "Computer Store". There was a small area with five or six OSI computers. I discovered that Wards was test marketing personal computers in two areas. OSI in the Minneapolis-St. Paul area and another system in Houston, Texas. The salesmen were from the OSI factory. They were informative, answered questions, and demonstrated their equipment. It seemed that having the backing of Wards for a product might be somewhat more reliable then a computer store that had been in business for six months. So I purchased a Challenger 4P cassette input system.

I am pleased with my computer, but not so with the equipment documentation nor the support from the manufacturer. The operators manual is poor. It is written for a disc system. Only nine out of 238 pages deal with the cassette system. OSI downplays the cassette based system and suggests, "for a relatively small investment over the cost of a cassette system one can have the benefits of a mini-floppy disc system". That small investment happens to be a thousand dollars.

The manual is full of mistakes, mistakes even I have been able to identify. Capabilities such as the graphic, color, tone generator and DAC receive only passing mention, generally 1 small demonstration program, some of which contain errors and do not work. The manual is written on the assumption that all purchasers are knowledgeable and literate in computer terminology and technology. There is also a lack of software on cassettes that is listed in OSI's advertising material. I wrote to the factory for information about software and all I received was a copy of the advertisement I wrote about.

I guess it is the frustration of owning this marvelous machine and not having access to the information that is necessary to learn how to utilize it's capabilities. Your magazine is a ray of hope that help for me is on the way. I look forward to more OSI oriented articles in COMPUTE. I have been told by the salespersons at Wards that they have sold over fifty OSI systems since they began selling then in October. There are some interested persons in this area.

Thank you for a fine publication. I wish you much success for years to come. Remember, some of your readers are beginners. Define terminology, identify what system a program listing is for.

Herbert Crandall
3135 East 69th Street
Inver Grove Heights, Mn. 55075

Thanks for a good letter. All of you potential authors out there should note the last two sentences. We've tried to help with the expanded table of contents, new this issue.
Robert Lock

The Single-Board 6502

Eric Rehnke

Even though February in California is like spring and summer in most other places, I've still been able to get some useful things accomplished.

On March 14 through the 16th, I'll be in San Francisco for the West Coast Computer Faire (my first one!). Depending on the timing, I may have a report for my next column.

By the way, even though I don't publish the 6502 User Notes anymore, I still like to get letters, comments about my column, product announcements etc. These can be sent to me directly at 540-61 South Ranch View Cir, Anaheim CA 92807. Articles and programs for submission to COMPUTE should, of course, be sent to the magazine.

TINY C FOR THE 6502

After following up on an ad in one of the computer classifieds, I discovered that a TINY C interpreter was indeed available for the 6502. I even placed an order so there may be a product review coming up in the near future. A friend of mine with an 8080 system purchased TINY C so I got a chance to see what the documentation was like. It was excellent. Looked like an ideal system for the neophyte to learn programming. (Yes, I'm aware of BASIC, but since BASIC is not structured it's easy to get sloppy and become an on-the-fly programmer. I feel that people who intend to get serious about learning how to program should first learn some sort of structured language such as ALGOL, C or perhaps PASCAL before moving over to BASIC).

Oh yes, the source listing and a KIM cassette are available.

For more info, contact:

TINY C ASSOCIATES
PO Box 269
Holmdel, NJ 07733
(201) 671-2296

1 K ADVENTURE GAME FOR KIM

Will wonders never cease?

I've just been playing around with a new KIM game from Robert Leedom (he also wrote HEX-PAWN and BASEBALL). KIM ADVENTURE plays a surprisingly good game for its size (I haven't even been able to finish a game yet with any treasures).

There are 24 locations, a dragon, some wizardry, a magic wand, etc. and so on. ALL IN 1K OF RAM!!!

The game can literally span several sessions of one to two hours apiece before you become an experienced traveller. There are eleven skill classifications from unrated to Grandmaster.

All in all, a very exciting game concept for a basic KIM. Could keep you (or your kids) busy (enchanted) for hours!

For more info, contact:

Bob Leedom
14069 Stevens Valley Ct.
Glenwood, MD 21733

AIM 65 TEST PROGRAM AVAILABLE

Some of you AIM users will be happy to hear that Rockwell is making available the program that they use to check out AIM's before they are shipped.

The program is 5K bytes long and normally resides in a couple of EPROMS which gets installed in the BASIC ROM slots.

If you're interested, order the Test Manual (EA74-M800) and the Test Program listing (PL-EA74-J100) at $15.00 for the pair from:

Rockwell International
Spares Control
PO Box 3669, RC-48
Anaheim, CA 92803

OSI MONITOR LISTING NEEDED

A friend of mine has an OSI C2-4P computer. Now he's been pretty happy with the machine despite the rather meager documentation - but he needs to know more about the machine than OSI wants to divulge. He needs a listing of the monitor so he can use the built-in I/O devices in his own programs. (Actually, he wants to use his C2-4P as a serial terminal for his KIM-1. BRAVO!!!)

The fact that OSI (and one or two others) don't document the monitor routines for the user completely baffles me. Do they think that some other company will copy their code or do they not consider this a part of properly documenting their machines? Since nobody I'm aware of has "ripped-off" the Apple or KIM monitor programs, I'm sure this can't be the problem.

Does anyone know why OSI doesn't choose to document their hobbyist systems to the same degree as the Apple, KIM SYM, or AIM?

In the meantime, if any of you OSI freaks has generated a source listing of his 65V monitor PROM (as found in the early C2-4P) could you please help a fellow user in distress?

Send it to: Ron Regal W8GMH
5614 Alber Ave.
Parma, OH 44129

If you send me a copy, there's a good chance we could print it here in COMPUTE and assist other information hungry OSI users.

INDEPENDENT OSI USERS NEWSLETTER

I just received the first issue of a newsletter intended for OSI users. The publisher, Charles Curley, wrote all of the 8 page issue himself with the help of his C2-8P disk system.

Included in issue #1 were fixes for bugs in two of the OSI operating systems, a book review, a hardware review, information on an accessible time sharing system, and an interview with Alan Taylor who is the head of OSI's west coast office.

Looks like it could be useful if enough users contribute information. Subscription rate is $10 for six issues.

contact:

Charles Curley
6061 Lime Ave #2
Long Beach, CA 90805

SYSTEM SOON FROM HDE

Hudson Digital Electronics (POB 120, Allamuchy NJ 07820) will soon be announcing their first packaged system-the OMNI 65.

Based on KIM as the CPU, OMNI 65 will consist of two mini floppy drives, 32K of RAM, EPROM boot for the disk system and system software dependent on which version of OMNI 65 is ordered. Three versions will be available: the engineering version, the word processing version and the full development version. Full size disk drives will be optional.

Basic price for the system will be from $4000 to $4500 depending upon the configuration.

MAKING BEAUTIFUL MUSIC

I've been mildly interested in computer music generation (synthesis?) for some time now. But it wasn't until Texas Instruments introduced their SN76477 Complex Sound Generator that I saw a real possibility for turning my computer into a real-time music generator (as opposed to the more usual method of first composing a song and entering it into the system for later play).

A study of the SN76477 chip, however, indicated that full digital control over the sound output wouldn't be a straight-forward task. A number of external resistors and capacitors are needed to set the sound parameters and it just ain't that easy to control these analog critters without getting into complex read relay or electronic switching.

So, the idea for a real time musical instrument was shelved for awhile until a more suitable method of digitally controlled music output.

Fortunately, I didn't have to wait too long to find a suitable candidate for my "instrument".

It came in the form of an article published in the July 79 issue of BYTE magazine. The article was written by Stephen Ciarcia and discussed the operation of the TI chip AND a new device on the market- the General Instruments AY3-8910. The GI chip has the same sound generating abilities as the older TI device but offers complete digital control over all the sound parameters. A significant advantage and a necessity for my particular application.

Basically, there are two ways of generating computer music. The first method is to have the computer do all the sound generation. This can range from the simple "kluge" harp which consists of a simple transistor "amplifier" hung on one bit of an output port to a more complex device such as the popular Micro Technology Unlimited K-1002 DAC music board and associated 4-port music software. (I feel that the MTU DAC board represents the pinnacle of achievement in computer generated music).

The second method is to use the computer as the controller of a sound generating device such as a top octave generator, a full synthesizer, or, as in this example, the GI AY-3-8910 Programmable Sound Generator chip.

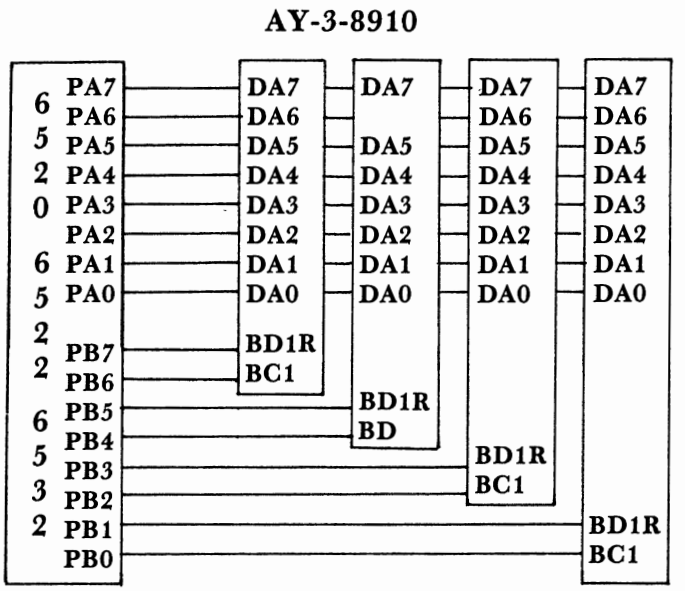
The main advantage to using the computer as a control element in a sound generating system and not as the sound generating device itself is that a lot of CPU time is freed up for other tasks such as handling the instrument keyboard interface and perhaps controlling a number of sound generating subsystems. The use of hardware subsystems could also simplify the system software generating task.

The AY-3-8910 has three audio output channels, each channel having its own tone generator, noise generator, a mixer to combine the outputs of the tone generator and noise generator, amplitude control with fixed or variable amplitude pattern, an envelope generator to control the variable amplitude pattern and a D/A converter that produces up to a 16 level output signal as determined by the amplitude control.

Pretty slick!

Since each chip has three separate audio outputs and I will be controlling the system with a three to six octave surplus electronic organ keyboard, it would be nice to have one audio channel for each of my ten fingers (assuming I could, or would, span 10 different keys at the same moment). That says we need at least 4 sound generator chips to handle the task. Well, it just so happens that 4 devices can be hooked to a 6522, or any of the PIA devices, rather easily since the 8 bit data port on the AY-3-8910

tri-states when unselected so can be paralleled together and each chip has 2 control lines. The interface will look something like this:



PRODUCT REVIEW

I've been using the MICRO TECHNOLOGY UNLIMITED K1000-5 power supply with my AIM 65 for several months now. The supply is fully enclosed with the 5 volt regulator and a large heat sink mounted on the outside of the box.

With a fully loaded AIM (4K RAM, Assembler and Basic ROMS) and an HDE 8K RAM card connected to the AIM expansion connector, the 5 volt regulator on the MTU supply got just a bit too warm to touch for more than a fraction of a second. Without the external 8K RAM board, the regulator barely got even warm.

This indicated to me that there was some reserve capacity built into the design of the supply. MTU has a very good reputation for having their engineering act together and I haven't found any reason to doubt this from what I've seen.

The one problem that I did have with the unit was a failure of the 5 volt regulator. Now this happened shortly after I soldered a LED across the output of the regulator so I could have inadvertently applied too much heat to the device. I'm not sure. However, MTU cheerfully repaired the unit and I haven't had any trouble since. (I even ran the heavy load experiment several more times to be sure).

The only thing I didn't like was the lack of a power switch and pilot light. I installed these at a cost of $1.50. When I mentioned this to MTU they indicated that they would consider adding these items to future units.

I liked the fact that the K1000-5 was fully enclosed and ready to use as opposed to some of the open-frame supplies available which need to have a power cord and fuse hooked-up and which are also unsafe because of exposed AC voltages.

The MTU supply retails for $80.00 and more information can be obtained from MTU, P.O. Box 4596, Manchester, NH 03103. ©

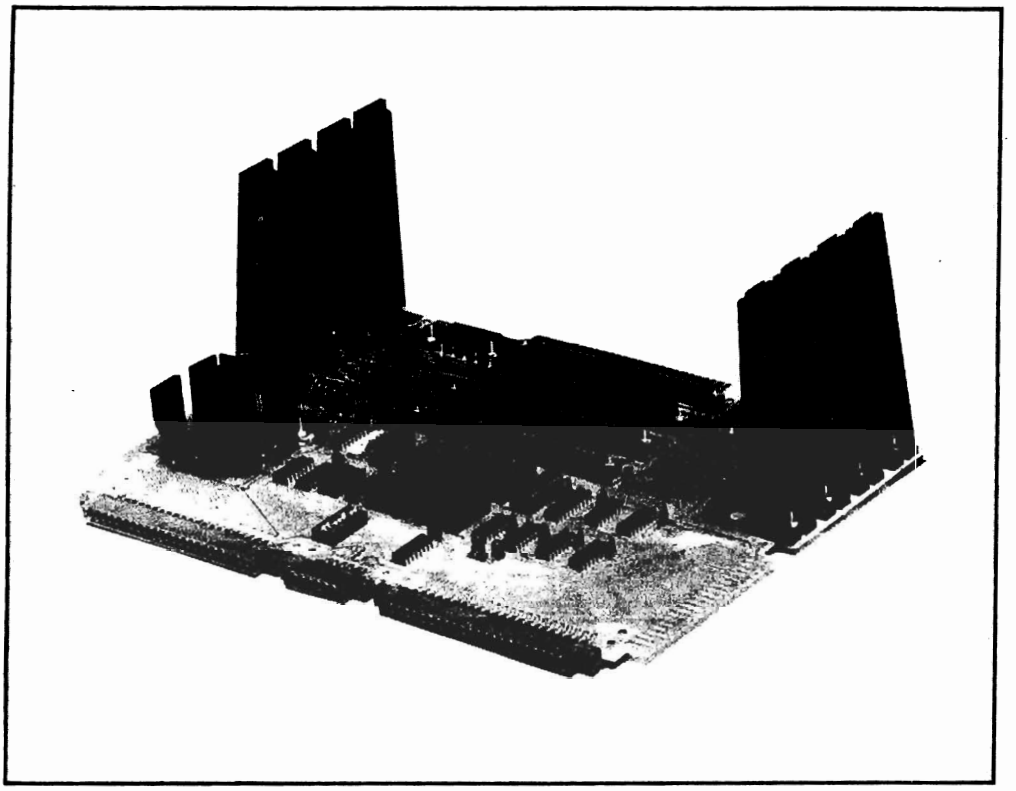
More about compute II

Robert Lock

This is your last issue of COMPUTE; next month you'll receive Issue #1 of **compute II.** The problem, as explained briefly in the Editor's Notes, is one of wanting COMPUTE to grow in too many directions at once. It's frustrating to me to have to limit the SBC Gazette to a handful of articles when you're sending them in as fast as the rest of the magazine readers put together.

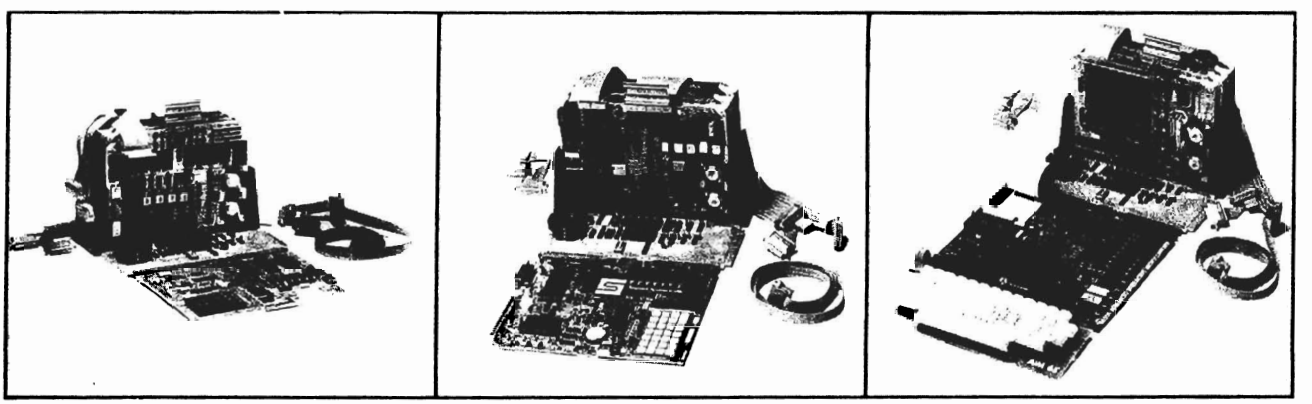
Our solution? Your "own" 56 - 64 page magazine. **compute II** will maintain all of the quality and enthusiasm of COMPUTE. I hope many of you will choose to subscribe to both, but regardless, let's get to work making **compute II** as healthy a resource magazine as COMPUTE has become. Please send me your comments, programming hints, articles and suggestions. One last note: we are currently redoing most of the artwork submitted with articles and columns. In many cases we are unable to redo some of the listings. Rather than use five or six pages of needed space for program listings, we're reducing them. A good example is Eric's Pet to Aim program listing in this issue. I would appreciate your comments on the readability. Materials should be sent to my attention, **compute II** magazine, P.O. Box 5119, Greensboro, N.C. 27403.

Introducing SEAWELL's



Little Buffered Mother

The ultimate Motherboard for any KIM-1, SYM-1, or AIM-65 system



Features:

- 4K Static RAM on board
- +5V, +12V, and −12V regulators on board
- 4 + 1 buffered expansion slots
- Accepts KIM-4 compatible boards
- Full access to application & expansion connector
- LED indicators for IRQ, NMI, and power-on
- Also compatible with SEA-1, SEA-16, the PROMMER, SEA-PROTO, SEA-ISDC, and more
- Onboard hardware for optional use of (128K addressing limit)
- Mounts like KIM-4 or with CPU board standing up
- 10 slot Motherboard expansion available - SEAWELL's Maxi Mother

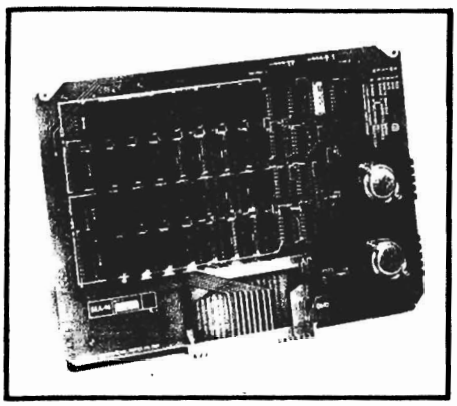
$199.00

Assembled Only

For further information contact:

SEAWELL Marketing Inc.
P.O. Box 30505
Seattle, WA 98103

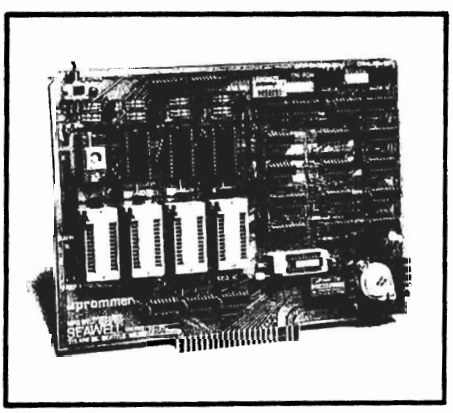
Little Buffered Mother's Compatible Hardware



SEA-16/16

SEA-16/16 is a 16K x 8 Static RAM Board. Two individually addressable 8K blocks of RAM, with individual WRITE PROTECT and optional Bank Switching. SEA-16/8 is the same as above except 8K of RAM not installed.
SEA-16/BLANK is the same as above, except no RAM is installed, thirty two 2114's makes it go.

SEA-16/16-45 $325.00

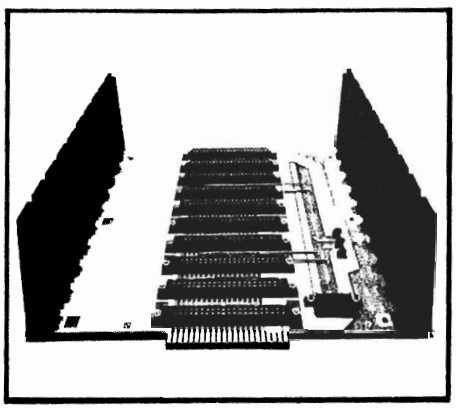


The Prommer

EpROM Programmer and EpROM/ROM Board. Memory mapped EpROM programming will program up to 16K per command, firmware in PROM included. Specify KIM, SYM or AIM.

(Presently in redesign. Available soon.)

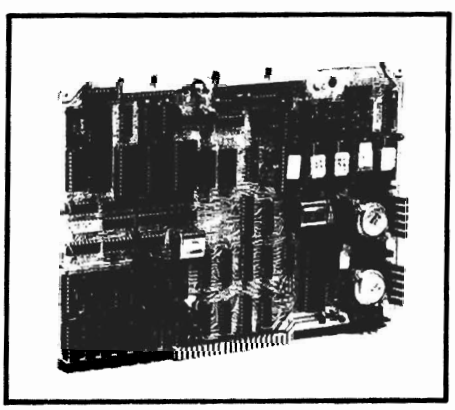
All products are assembled. No kits.



The Maxi-Mother

The Maxi-Mother® is a 10 slot Mother Board. Combined with the Little Buffered Mother®, it provides 14 slots or use it stand-alone with SEA-1.

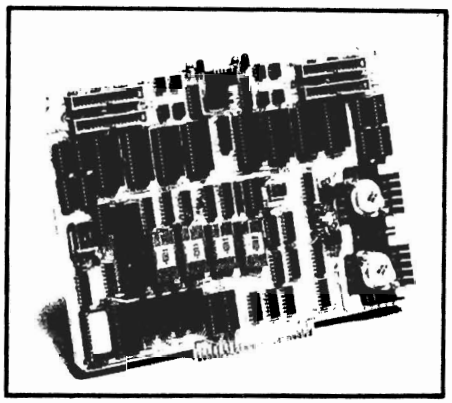
MAXI-MOTHER $135.00



SEA-1 Seawell's CPU Board

MCS 6512 CPU, 3 Hardware RS-232 Ports, one current-loop, programmable Audio Tape interface, 9K + Static RAM, up to 18K of ROM/EpROM, Single Step Hardware, Bank Switching, Interrupt driven Monitor, and more.

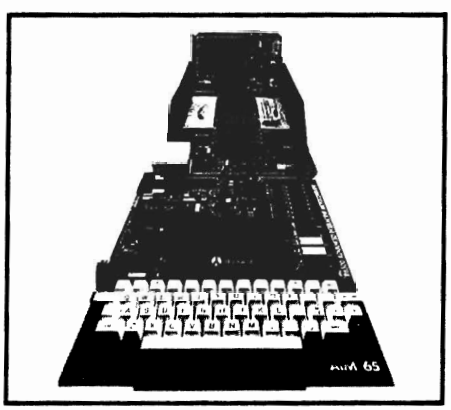
SEA-1, 1.5 MHz $595.00



SEA-ISDC Intelligent Serial Data Concentrator

MCS 6512 CPU, 4K of Dual Port RAM, 2K local RAM, 4K EpROM/ROM, 8 RS-232 Ports. The SEA-ISDC can operate as a stand-alone computer or as a buss compatible "Front End" I/O Processor for a larger system. The 4K of RAM can be accessed by either CPU without delay.

SEA-ISDC $595.00



Or go for the works!

AIM-65 with the Little Buffered Mother®, the Maxi-Mother® and SEA-16/16.

Available Soon!

Development System
Parallel I/O Board, Prototype Board and Floppy Disk Controller
CMOS/PROM/Real Time Clock Board

SEAWELL Marketing Inc.
315 N.W. 85th
Seattle, WA 98117
(206) 782-9480

Read Pet Tapes With Your Aim

Eric Rehnke
Mark Reardon

From now on, you AIM owners needn't feel like second class citizens. All those PET Basic tapes that you've been drooling over can now be fed into your AIM. Admittedly, you'll have to force feed them since AIM doesn't normally read PET cassette tapes, but, it's almost painless and just think of all the fun you'll have.

What this program that Mark and I have written **DOES do** is read a PET Basic program into memory and convert all the convertable token to those used by AIM Basic. There are, however, a number of PET tokens which cannot be converted to AIM and these special tokens are printed out together with the line number they are found on. The conversion program then replaces these offending tokens with a REM token so they can be easily found. The BASIC program can now be saved to AIM cassette using the normal Basic SAVE command.

What this program **DOES NOT do** is get the PET program to actually RUN on AIM. You'll have to change any PEEKS and POKES or use of Pet's graphics to look at AIM's memory locations and use AIM I/O devices.

Programs that use very little of PET's graphics or special memory locations will, of course, be the easiest to convert.

Since most PET programs are written for 8K systems, there should be at least 8K of RAM in your AIM. Also, you'll need to connect up the remote control to your cassette deck for reading PET tapes.

First, turn the printer on, then load the PET loader into your system and start it running at $0200 (or use the F1 key if its properly initialized). Now start reading a PET cassette. When the loader finds a program on the tape it will stop the recorder and prompt with the program name and the last address that it will occupy in memory.

Press 'Y' to load that program. When it's loaded, the conversion program will go through each program line printing out the line number and converting all the tokens in that line. If the printer was turned on, you'll now have a listing of the line numbers. Any tokens not found in AIM will be printed out and then replaced with the REM token so AIM BASIC doesn't choke.

When all the lines have been converted, the PET Loader will automatically initialize and jump into BASIC. The PET program can now be listed and saved in the normal fashion.

If any other key besides "Y" was pressed in response to the program name prompt, the program will skip forward to find the next program on the tape. Since the PET Loader does not mess around with anything contained within quotes and PET programs can have special cursor controls enclosed within quotes, funny things will get printed by AIM in place of the cursor controls. AIM Basic will interpret these cursor control characters as tokens and print out the equivalent AIM BASIC statement.

The cassette read portion of this program is a much modified version of a program originally published in MICRO magazine, page 61 of the June 1979 issue and written by Dave Kemp.

128 END	164 +
129 FOR	165 -
130 NEXT	166 *
131 DATA	167 /
132 INPUT	168 ^
133 DIM	169 AND
134 READ	170 OR
135 LET	171 >
136 GOTO	172 =
137 RUN	173 <
138 IF	174 SGN
139 RESTORE	175 INT
140 GOSUB	176 ABS
141 RETURN	177 USR
142 REM	178 FRE
143 STOP	179 POS
144 ON	180 SQR
145 NULL	181 RND
146 WAIT	182 LOG
147 LOAD	183 EXP
148 SAVE	184 COS
149 DEF	185 SIN
150 POKE	186 TAN
151 PRINT	187 ATN
152 CONT	188 PEEK
153 LIST	189 LEN
154 CLEAR	190 STR$
155 GET	191 VAL
156 NEW	192 ASC
157 TAB(	193 CHR$
158 TO	194 LEFT$
159 FN	195 RIGHT$
160 SPC(	196 MID$
161 THEN	
162 NOT	
163 STEP	

Aim Tokens

Program Listing

```
0002  1000=              RAMTOP=$1000
0003  EA46=              NUMA=$EA46
0004  E97A=              OUTPUT=$E97A
0005  E83E=              BLANK=$E83E
0006  E973=              REDOUT=$E973
0007  EA13=              CRLOW=$EA13
0008  E1A1=              COMIN=$E1A1
0009  E83E=              BLANK=$E83E
0010  CE85=              BASTBL=$CE85
0011  B27F=              BASIC6=$B27F
0012  00F0=              ADL=$F0
0013  00F1=              ADH=$F1
0014  00F2=              TCNT=$F2
0015  00F3=              TPAR=$F3
0016  00F4=              PECNT=$F4
0017  00F5=              CNT=$F5
0018  00F6=              PTR=$F6
0019  00F8=              LINLOW=$F8
0020  00F9=              LINHI=$F9
0021  00FA=              OLDL=$FA
0022  00FB=              OLDH=$FB
0023  00BE=              TABLE=$BE
0024  0073=              PGMST=$73
0025  00FE=              PAR=$FE
0026  A800=              TAPE=$A800

0028  0000                    *=$10C
0029  010C  4C 00 02          JMP START

0031  010F                    *=$200
0032  0200  A9 37       START LDA #$37        ;SET UP TAPE INPUT
0033  0202  8D 02 A8          STA TAPE+2
0034  0205  A9 EE             LDA #$EE
0035  0207  8D 0C A8          STA TAPE+12
0036  020A  20 94 03    NEXT  JSR PETCAS      ;READ PET TAPE
0037  020D  AD 6A 04          LDA FLAG        ;LEADER = $01
0038  0210  C9 01             CMP #01         ;LEADER?
0039  0212  20 13 EA          JSR CRLOW
0040  0215  20 13 EA          JSR CRLOW
0041  0218  D0 F0             BNE NEXT

0043  021A  20 89 03          JSR OFFON       ;TURN OFF TAPE
0044  021D  A2 00             LDX #00         ;OUTPUT NAME OF FILE
0045  021F  BD 6F 04    NAME  LDA FILE,X
0046  0222  C9 20             CMP #' '        ;LOOK FOR BLANK AT END
0047  0224  F0 06             BEQ LEN
0048  0226  20 7A E9          JSR OUTPUT
0049  0229  E8                INX
0050  022A  D0 F3             BNE NAME        ;GET NEXT LETTER
0051  022C  20 3E E8    LEN   JSR BLANK

0053  022F  18                CLC             ;OUTPUT NECESSARY MEMORY
0054  0230  AD 6D 04          LDA END         ;FOR PROGRAM
0055  0233  69 62             ADC #$62        ;ADD TO END
0056  0235  8D 6D 04          STA END         ;THE DIFFERENCE BETWEEN PET AND
0057  0238  AD 6E 04          LDA END+1       ;AIM BASIC START LOCATIONS
0058  023B  69 00             ADC #00
0059  023D  20 46 EA          JSR NUMA        ;OUTPUT IT
0060  0240  AD 6D 04          LDA END
0061  0243  20 46 EA          JSR NUMA
0062  0246  20 3E E8          JSR BLANK

0064  0249  20 73 E9          JSR REDOUT      ;GET A CHARACTER
0065  024C  20 89 03          JSR OFFON       ;TURN ON TAPE
0066  024F  C9 59             CMP #'Y         ;Y MEANS READ THIS FILE
0067  0251  F0 05             BEQ GO
0068  0253  8D 6A 04          STA FLAG        ;CHANGE FLAG'S VALUE
0069  0256  D0 B2             BNE NEXT        ;READ NEXT FILE ON TAPE

0071  0258  A9 08       GO    LDA #$08        ;DELAY PAST SECOND LEADER
0072  025A  85 F5             STA CNT
0073  025C  A2 FF       DELAY1 LDX #$FF
0074  025E  A0 FF       DELAY2 LDY #$FF
0075  0260  88          DELAY3 DEY
0076  0261  D0 FD             BNE DELAY3
0077  0263  CA                DEX
0078  0264  D0 F8             BNE DELAY2
0079  0266  C6 F5             DEC CNT
0080  0268  D0 F2             BNE DELAY1

0082  026A  20 94 03          JSR PETCAS      ;READ PROGRAM BODY
0083  026D  C8                INY             ;MAKE Y=0
0084  026E  8C 69 04          STY FLAG-1      ;PUT ZERO IN BASIC START
0085  0271  20 13 EA          JSR CRLOW
0086  0274  C6 F4             DEC PECNT       ;ONE ERROR WILL BE COUNTED
0087  0276  A5 F4             LDA PECNT       ;AT THE END OF THE PROGRAM
0088  0278  20 46 EA          JSR NUMA        ;OUTPUT NO. OF PARITY ERRORS
0089  027B  20 3E E8          JSR BLANK
0090  027E  B9 08 04    ERRS  LDA MSG,Y       ;OUTPUT ERROR MESSAGE
0091  0281  48                PHA
0092  0282  20 7A E9          JSR OUTPUT
0093  0285  C8                INY
0094  0286  68                PLA
0095  0287  10 F5             BPL ERRS

0097  0289  A2 04             LDX #>BASTRT    ;SET UP POINTER AND
0098  028B  86 F6             STX PTR         ;BASIC'S POINTER TO START
0099  028D  86 73             STX PGMST
0100  028F  A9 68             LDA #<BASTRT
0101  0291  85 F7             STA PTR+1
0102  0293  85 74             STA PGMST+1

0104  0295  20 13 EA    REENTR JSR CRLOW
0105  0298  D0 0F             BNE FIRST       ;Z=0 IF JUST STARTED
0106  029A  20 22 03          JSR INCPTR      ;BUMP IT TO NEXT LINE
0107  029D  A2 00             LDX #00         ;STORE PTR, PTR+1 IN LAST VECTOR
0108  029F  C8                INY             ;MAKE Y=1
0109  02A0  A5 F6             LDA PTR
0110  02A2  81 FA             STA (OLDL,X)
0111  02A4  AA                TAX             ;SAVE PTR IN X
0112  02A5  A5 F7             LDA PTR+1
0113  02A7  91 FA             STA (OLDL),Y

0115  02A9  85 FB       FIRST STA OLDH        ;SAVE PTR+1
0116  02AB  86 FA             STX OLDL        ;SAVE PTR
0117  02AD  20 22 03          JSR INCPTR
0118  02B0  D0 03             BNE MORE        ;LAST BYTE = 0
0119  02B2  4C 2D 03          JMP BASICS      ;FINISHED

0121  02B5  20 22 03    MORE  JSR INCPTR      ;CONVERT LINE NO. TO DECIMAL
0122  02B8  85 F8             STA LINLOW
0123  02BA  20 22 03          JSR INCPTR
0124  02BD  85 F9             STA LINHI       ;AND PRINT IT OUT
0125  02BF  A2 00       NXTDIG LDX #00        ;Y=0 FROM INCPTR SUBROUTINE
0126  02C1  A5 F8       SUBMEM LDA LINLOW     ;SUBTRACT MULTIPLES OF 10 FROM
0127  02C3  38                SEC             ;THE LINE NO. STARTING WITH 10000
0128  02C4  F9 0E 04          SBC SUBTBL,Y
0129  02C7  85 F8             STA LINLOW
0130  02C9  A5 F9             LDA LINHI
0131  02CB  C8                INY
0132  02CC  F9 0E 04          SBC SUBTBL,Y
0133  02CF  90 07             BCC ADBACK      ;SUBTRACTED ONE TOO MANY
0134  02D1  85 F9             STA LINHI
0135  02D3  E8                INX             ;COUNT NO. OF SUBTRACTIONS
0136  02D4  88                DEY
0137  02D5  4C C1 02          JMP SUBMEM      ;DO IT AGAIN WITH THE SAME VALUE

0139  02D8  88          ADBACK DEY            ;ADD BACK TO LSB ONLY
0140  02D9  A5 F8             LDA LINLOW
0141  02DB  79 0E 04          ADC SUBTBL,Y
0142  02DE  85 F8             STA LINLOW
0143  02E0  8A                TXA             ;CHANGE VALUE TO ASCII
0144  02E1  09 30             ORA #$30
0145  02E3  20 7A E9          JSR OUTPUT      ;PRINT IT
0146  02E6  C8                INY             ;SUBTRACT NEXT SMALLER POWER
0147  02E7  C8                INY             ;OF 10
0148  02E8  C0 08             CPY #08         ;END OF TABLE
0149  02EA  D0 D3             BNE NXTDIG

0151  02EC  A5 F8             LDA LINLOW      ;GET NO. OF ONES
0152  02EE  09 30             ORA #$30        ;MAKE IT ASCII
0153  02F0  20 7A E9          JSR OUTPUT
0154  02F3  20 3E E8          JSR BLANK

0156  02F6  20 22 03    LOOP1 JSR INCPTR      ;GET NEXT BYTE
0157  02F9  F0 9A             BEQ REENTR      ;END OF LINE
0158  02FB  48          CONT  PHA
0159  02FC  C9 22             CMP #'"         ;LOOK FOR STRINGS
0160  02FE  D0 0C             BNE CONT1       ;NOTHING IN QUOTES IS CHANGED
0161  0300  68                PLA             ;CLEAR STACK
0162  0301  20 22 03    LOOP1A JSR INCPTR     ;NEXT BYTE
0163  0304  F0 8F             BEQ REENTR      ;IF BYTE = 0
0164  0306  C9 22       CONT3 CMP #'"         ;LOOK FOR END
0165  0308  D0 F7             BNE LOOP1A
0166  030A  F0 EA             BEQ LOOP1       ;GO READ NEXT BYTE

0168  030C  68          CONT1 PLA             ;IGNORE ALL
0169  030D  10 E7             BPL LOOP1       ;CHAR HERE EXCEPT TOKENS
0170  030F  AA                TAX             ;MAKE TOKEN AN INDEX
0171  0310  BD 96 03          LDA TOKTBL-128,X ;FIND AIM'S TOKEN
0172  0313  30 09             BMI CONT4       ;FOR VALID TOKEN N=1
0173  0315  8A                TXA             ;OUTPUT INVALID TOKEN
0174  0316  20 46 EA          JSR NUMA
0175  0319  20 3E E8          JSR BLANK
0176  031C  A9 8E             LDA #$8E        ;MAKE TOKEN A REM
0177  031E  91 F6       CONT4 STA (PTR),Y
0178  0320  D0 D4             BNE LOOP1       ;GET NEXT BYTE

0180  0322  E6 F6       INCPTR INC PTR        ;GET NEXT LOCATIONS CONTENTS
0181  0324  D0 02             BNE INCPT1
0182  0326  E6 F7             INC PTR+1
0183  0328  A0 00       INCPT1 LDY #00
0184  032A  B1 F6             LDA (PTR),Y
0185  032C  60                RTS

0187  032D  20 22 03    BASICS JSR INCPTR     ;MAKE PTR LOOK PAST THE
0188  0330  A6 F6             LDX PTR         ;LAST THREE ZEROS AND
0189  0332  A5 F7             LDA PTR+1
0190  0334  86 75             STX $75         ;SAVE IN BASIC POINTERS
0191  0336  85 76             STA $76
0192  0338  A2 00             LDX #<RAMTOP    ;SET UP THE REST OF BASIC
0193  033A  A9 10             LDA #>RAMTOP
0194  033C  86 7F             STX $7F
0195  033E  85 80             STA $80
0196  0340  A2 FE             LDX #$FE
0197  0342  9A                TXS
0198  0343  A9 00             LDA #00
0199  0345  48                PHA
0200  0346  85 01             STA $01
0201  0348  85 10             STA $10
0202  034A  85 60             STA $60
0203  034C  85 80             STA $80
0204  034E  A2 1C             LDX #$1C
0205  0350  BD 85 CE    L1    LDA BASTBL,X
0206  0353  95 BE             STA TABLE,X
0207  0355  CA                DEX
0208  0356  D0 F8             BNE L1
0209  0358  A9 03             LDA #03
0210  035A  85 9B             STA $9B
0211  035C  A9 61             LDA #$61
0212  035E  85 5E             STA $5E
0213  0360  A9 B9             LDA #$B9
0214  0362  85 02             STA $02
0215  0364  A9 14             LDA #$14
0216  0366  85 12             STA $12
0217  0368  A9 0A             LDA #$0A
0218  036A  85 13             STA $13
0219  036C  A9 E1             LDA #$E1
0220  036E  85 82             STA $82
0221  0370  A9 4C             LDA #$4C
0222  0372  85 00             STA $00
0223  0374  85 03             STA $03
0224  0376  85 9C             STA $9C
0225  0378  85 BB             STA $BB
0226  037A  A2 87             LDX #$87
0227  037C  A9 8F             LDA #$8F
0228  037E  86 BC             STX $BC
0229  0380  85 BD             STA $BD
0230  0382  86 04             STX $04
0231  0384  85 05             STA $05
0232  0386  4C 7F B2          JMP BASIC6      ;JMP TO BASIC WARM START

0234  0389  48          OFFON PHA             ;TOGGLE OFF/ON THE TAPE
0235  038A  AD 00 A8          LDA TAPE
0236  038D  49 30             EOR #$30
0237  038F  8D 00 A8          STA TAPE
0238  0392  68                PLA
0239  0393  60                RTS

0241  0394  A9 04       PETCAS LDA #>LOAD     ;START SAVING AT LOAD
0242  0396  85 F1             STA ADH
0243  0398  A9 61             LDA #<LOAD
0244  039A  85 F0             STA ADL
0245  039C  A9 02             LDA #$02
0246  039E  85 F4             STA PECNT
0247  03A0  20 C5 03    PETCA JSR GBYTE       ;GET IN SYNCH WITH TAPE
0248  03A3  30 03             BMI PETX        ;READ SHORTS?
0249  03A5  4C 94 03          JMP PETCAS      ;NO, TRY AGAIN
0250  03A8  C6 F4       PETX  DEC PECNT       ;TWICE?
```

+B0

```
0251  03AA  D0 F4           BNE PETCA       ;NO. GO DO AGAIN
0252  03AC  20 C5 03  PETCB  JSR GBYTE      ;READ THE REST OF THE LEADER
0253  03AF  30 FB           BMI PETCB       ;STILL SHORTS
0254  03B1  B0 04     PETCC  BCS PETCD       ;IF CARRY IS SET THEN NO
0255  03B3  A9 FE           LDA #PAR        ;PARITY ERROR
0256  03B5  E6 F4           INC PECNT
0257  03B7  91 F0     PETCD  STA (ADL),Y     ;Y=0
0258  03B9  E6 F0           INC ADL         ;INC INDIRECT ADDRESS
0259  03BB  D0 02           BNE PETCE
0260  03BD  E6 F1           INC ADH
0261  03BF  20 C5 03  PETCE  JSR GBYTE      ;READ NEXT BYTE
0262  03C2  10 ED           BPL PETCC       ;PLUS MEANS THAT IT'S NOT SHORTS
0263  03C4  60               RTS             ;SHORTS MEAN FILE IS DONE

0265  03C5  A0 11     GBYTE  LDY #$11        ;NO. OF SHORTS
0266  03C7  20 F9 03  GBA    JSR GETTR       ;GET TRANSITIONS
0267  03CA  E0 40           CPX #$40        ;IS IT A LONG?
0268  03CC  B0 08           BCS GBB         ;YES, GO READ BYTE
0269  03CE  E0 2C           CPX #$2C        ;IS IT A SHORT?
0270  03D0  B0 F3           BCS GBYTE       ;NO, GET IN SYNCH
0271  03D2  88              DEY             ;COUNT SHORTS
0272  03D3  10 F2           BPL GBA
0273  03D5  60               RTS

0275  03D6  A0 09     GBB    LDY #$09        ;BIT COUNT PLUS PARITY
0276  03D8  84 F3           STY TPAR        ;PARITY COUNTER
0277  03DA  20 F9 03        JSR GETTR       ;WASTE LAST HALF OF START BIT
0278  03DD  20 EE 03  GBC    JSR GETBIT      ;BIT VALUE IN C
0279  03E0  90 02           BCC GBD         ;0?
0280  03E2  E6 F3           INC TPAR        ;=1 SO INC PARITY COUNT
0281  03E4  6A        GBD    ROR A           ;ROLL INTO BYTE AT MSB
0282  03E5  88              DEY             ;9 YET
0283  03E6  D0 F5           BNE GBC         ;NO
0284  03E8  2A              ROL A           ;LOSE PARITY BIT
0285  03E9  49 FF           EOR #$FF        ;INVERT
0286  03EB  46 F3           LSR TPAR        ;PUT PARITY IN C
0287  03ED  60               RTS

0289  03EE  20 F9 03  GETBIT JSR GETTR       ;GET FIRST TRANSITION
0290  03F1  86 F2           STX TCNT
0291  03F3  20 F9 03        JSR GETTR       ;GET SECOND TRANSITION
0292  03F6  E4 F2           CPX TCNT        ;COMPARE, C= BIT
0293  03F8  60               RTS

0295  03F9              ;THE PET CASSETTE FORMAT
0296  03F9              ;IS PHASE DEPENDENT.  SOME RECORDERS MAY
0297  03F9              ;INADVERTENTLY INVERT THIS SIGNAL.  TO
0298  03F9              ;CORRECT THIS, CHANGE THE FOLLOWING INSTRUCTIONS:
0299  03F9              ;      'BMI GETA'   TO  'BPL GETA'
0300  03F9              ;      'BPL GETB'   TO  'BMI GETB'
0301  03F9              ;
0302  03F9  A2 00     GETTR  LDX #00         ;X= NO. OF CYCLES THRU LOOPS
0303  03FB  E8        GETA   INX
0304  03FC  2C 00 A8        BIT TAPE        ;CHECK PB7
0305  03FF  30 FA           BMI GETA        ;LOOP UNTIL LOW
0306  0401  E8        GETB   INX
0307  0402  2C 00 A8        BIT TAPE
0308  0405  10 FA           BPL GETB        ;LOOP UNTIL HIGH AGAIN
0309  0407  60               RTS
0310  0408  45 52     MSG    BYT 'ERROR',$D3
0310  040D  D3
0311  040E  10 27     SUBTBL .WOR 10000      ;POWERS OF 10 TABLE
0312  0410  E8 03            WOR 1000
0313  0412  64 00            WOR 100
0314  0414  0A 00            WOR 10
0315  0416  80        TOKTBL .BYT 128,129,130,131,132
0315  0417  81
0315  0418  82
0315  0419  83
0315  041A  84
0316  041B  00               BYT 0,133,134,135,136,137,138
0316  041C  85
0316  041D  86
0316  041E  87
0316  041F  88
0316  0420  89
0316  0421  8A
0317  0422  8B               BYT 139,140,141,142,143,144,146
0317  0423  8C
0317  0424  8D
0317  0425  8E
0317  0426  8F
0317  0427  90
0317  0428  92
0318  0429  93               BYT 147,148,0,149,150,0,151
0318  042A  94
0318  042B  00
0318  042C  95
0318  042D  96
0318  042E  00
0318  042F  97
0319  0430  98               BYT 152,153,154,0,0,0,0
0319  0431  99
0319  0432  9A
0319  0433  00
0319  0434  00
0319  0435  00
0319  0436  00
0320  0437  9B               BYT 155,156,157,158,159,160
0320  0438  9C
0320  0439  9D
0320  043A  9E
0320  043B  9F
0320  043C  A0
0321  043D  A1               BYT 161,162,163,164,165,166
0321  043E  A2
0321  043F  A3
0321  0440  A4
0321  0441  A5
0321  0442  A6
0322  0443  A7               BYT 167,168,169,170,171,172,173,174
0322  0444  A8
0322  0445  A9
0322  0446  AA
0322  0447  AB
0322  0448  AC
0322  0449  AD
0322  044A  AE
0323  044B  AF               BYT 175,176,177,178,179,180
0323  044C  B0
0323  044D  B1
0323  044E  B2
0323  044F  B3
0323  0450  B4
0324  0451  B5               BYT 181,182,183,184,185,186,187
0324  0452  B6
0324  0453  B7
0324  0454  B8
0324  0455  B9
0324  0456  BA
0324  0457  BB
0325  0458  BC               BYT 188,189,190,191,192,193,194,195,196
0325  0459  BD
0325  045A  BE
0325  045B  BF
0325  045C  C0
0325  045D  C1
0325  045E  C2
0325  045F  C3
0325  0460  C4

0327  0461              ;TO CHANGE THE LOCATION THAT
0328  0461              ;THE BASIC PROGRAM IS
0329  0461              ;STORED IN, CHANGE THE
0330  0461              ;LOCATION OF THESE FLAGS.
0331  0461              ;IE. *=$2000

0333  0461              LOAD   *=*+9
0334  046A              FLAG   *=*+1
0335  046B              BASTRT *=*+2
0336  046D              END    *=*+2
0337  046F              FILE   *=*+1
0338  0470              ZZZ    END

ERRORS=0000 <0000>


SYMBOL TABLE

ADBACK  02D8  ADH     00F1  ADL     00F0  BASIC6  B27F
BASICS  032D  BASTBL  CE85  BASTRT  046B  BLANK   E83E
CNT     00F5  COMIN   E1A1  CONT    02FB  CONT1   030C
CONT3   0306  CONT4   031E  CRLOW   EA13  DELAY1  025C
DELAY2  025E  DELAY3  0260  END     046D  ERRS    027E
FILE    046F  FIRST   02A9  FLAG    046A  GBA     03C7
GBB     03D6  GBC     03DD  GBD     03E4  GBYTE   03C5
GETA    03FB  GETB    0401  GETBIT  03EE  GETTR   03F9
GO      0258  INCPT1  0328  INCPTR  0322  L1      0350
LEN     022C  LINHI   00F9  LINLOW  00F8  LOAD    0461
LOOP1   02F6  LOOP1A  0301  MORE    02B5  MSG     0408
NAME    021F  NEXT    020A  NUMA    EA46  NXTDIG  02BF
OFFON   0389  OLDH    00FB  OLDL    00FA  OUTPUT  E97A
PAR     00FE  PECNT   00F4  PETCA   03A0  PETCAS  0394
PETCB   03AC  PETCC   03B1  PETCD   03B7  PETCE   03BF
PETX    03A8  PGMST   0073  PTR     00F6  RAMTOP  1000
REDOUT  E973  REENTR  0295  START   0200  SUBMEM  02C1
SUBTBL  040E  TABLE   00BE  TAPE    A800  TCNT    00F2
TOKTBL  0416  TPAR    00F3  ZZZ     0470
END OF ASSEMBLY
```

REVIEW
KIMEX-1

PROM, RAM and I/O Expansion for the KIM

Harvey B. Herman

Digital Engineering Associates $139.95
P. O. Box 207
Bethlehem, PA 18016

Those of us who have cut our computer baby teeth on the KIM have longed to have some of the capabilities of SYM (a newer, single-board computer) without, heaven forbid, having to throw out our first love. Digital Engineering Associates has come to our rescue with their product KIMEX-1. They are marketing a single-board add-on module which plugs into the KIM expansion interface and requires 6 wires to be soldered to the KIM application connector. The following features are standard:

1. Sockets for 4K of RAM (4118) contiguous with KIM's 1K RAM.
2. A 6522 VIA with I/O lines brought out to a separate connector.
3. Sockets for four 2716 5VEPROMs which can be selectively vectored to on power up.

The last item is really neat as this should greatly simplify operation of applications programs in EPROM by users unfamiliar with KIM.

The module appears to my eye very well designed and professionally constructed. It was trivial to connect to a basic KIM (15 minutes or less). For review purposes only, the company provided a clock program on EPROM which is described as an example in their 19-page manual. I turned on power (an extra 300 mamp from the 5V supply is necessary) and I was into the clock program and counting. Their program makes use of the 6522 VIA on board (a data sheet on the 6522 is also included with the manual). I am only just beginning to appreciate the "versatility" of the VIA chip and missed having one on the original KIM. Now's my chance.

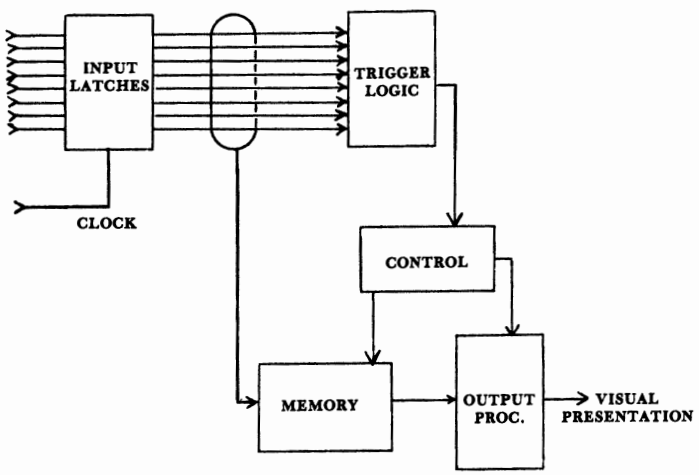
The negative points are minor. I believe it may be more difficult and/or expensive to obtain a MOSTEK 4118 (1K x 8) than a 2114 (1K x 4), for example. Furthermore, it might have been helpful in some systems to address the 4K of RAM anywhere in memory. Other than that, I think the module is a pretty good deal for KIM owners who need its features, and I recommend it to them. ©

Applications Review: Logic Analyzer For Kim

Roy Flacco
Physics Dept.
Drexel University
32nd and Chestnut
Phila. PA 19104

I have been using a logic analyzer on the KIM for some time and have become reasonably familiar with its applications (at least the simpler stuff) and appreciate the opportunity to share my experiences.

In its simplest form a Logic Analyzer (LA) consists of a memory, triggering logic, control circuitry, input latches, and output processing:



Basically, the LA latches a set of inputs, usually the Address and Data Buses every clock cycle (02) and stores them in a memory. Also, the trigger logic compares the logic state of these input lines with preset levels which are user-set. When the inputs match the presets, the LA "triggers" and does one of two things. If the LA is set to positive time, it stores in memory the logic levels present at the inputs for each clock cycle until the memory is full; or if the LA is set to negative time, it stops storing the inputs in memory (remember it has been storing all the previous states up till the trigger occurs, too.) Therefore, the memory now contains some number of data words representing successive words on the bus lines.

The LA now proceeds to display the data in the memory in some visually acceptable form. This has taken the form of strings of LED's, timing-diagram scope displays, etc., but the most useful by far is a numeric display (ones and zeroes) on a scope, at least to the software-debugging, hardware-soldering, wild-eyed amateurs of which group I am proudly a member. Vastly more handy than a logic probe, the LA gives you a history of exactly what your machine did, cycle by cycle, in a form like the source listing it is hopefully trying to follow.

At present there is only one logic analyzer priced within the reach of the typical KIM owner: the Model 100A from Paratronics, Inc. This is available for $229 from Jameco Electronics, (1355 Shoreway Rd. Belmont, CA 94002). It has an 8-bit by 16-word memory, can run at clock rates up to 8 MHz (mine just starts failing at 10 MHz so this is a conservative rating), works in both positive and negative time on positive-or negative-going clocks, will run from TTL, DTL, CMOS, MOS (input range from 0 to +15 volts) and outputs to the vertical and horizontal inputs of your scope. There are also a number of nice features such as an adjustable blanking signal to clarify the display and intensify the trigger word, and your choice of Hex or Octal display format. For example: 0000 0000 can be displayed as 00 000 000 if your code is written in octal.

The Model 100A, by itself, is useful but limited. First and most obviously, you are limited to 8 bits of triggering, and must move your input probes around if you wish to view data and address busses (which add up to 24 bits). The main application would be monitoring the data bus and checking that the proper data shows up on it in the proper sequence, or using it as a multi-input logic probe which can show you just how long that pulse lasted (the one that was stretched to 1/10 second on your hand-held model). But the thing to do is add the Model 10 Trigger Expander.

This lovely thing (also $229 from James) sits happily across all the Address and Data lines (it presents roughly one LS-TTL load to the bus line), the 02 clock, and two other auxillary lines of your choosing (more later about these). Now see what you can do:

A Hexadecimal-coded thumbwheel switch (0000-FFFF) lets you choose the address trigger; if you desire you can also trigger from the data bus (00-FF using 0-X-1 paddle switches); you choose either the Data (DB) or the high or low order Address (ABH,ABL) to be stored in the memory via a paddle-switch controlled multiplexer (MUX). Along with all that, you now have DELAY capability, which allows you to examine the buses from 0 to 999 (decimal) clock cycles or program loop reps after the initial trigger. The closest approximation to this capability is a very elegant de-bugging program that allows you to set breakpoints within loops. But even so, the breakpoint routines don't allow you to see what's actually happening in a live run, and they depend on having all your hardware running well. The LA can be used

to find faults in the hardware such as stuck bits (a line that's always at logic 1 or 0), or other such problems that keep you from running at all.

The two auxillary lines are gated with the clock and the address trigger to allow you to "qualify" the trigger or the clock. This may be from the Read/Write line so that you only view those events that occur during a write to memory, or from the SYNC signal so that only OP CODE fetches are recorded. These signals may be derived from any convenient point in the KIM where they occur at the desired time to control the LA.

The Model 10 Trigger Expander connects to the Model 100A LA via a piece of ribbon cable, and to the Expansion Connector of the KIM (I recommend wiring a 44-pin edge connector just for the LA; it greatly simplifies use). Your scope hangs off the 100A. By the way, there is a TTL signal which lets you know when the LA has triggered; this is available at a BNC connector and may be used to trigger the sweep of your scope if you want to view voltage levels somewhere in the computer at a time defined by the program. It is terrific for catching glitches, slow rise times, overlapping clock pulses, and so on. When all the voltage levels are essentially the same it is much easier to trigger from the logic levels in combinations.

Now, insofar as the unit itself is concerned, mine was purchased as two kits (100A first, then 10). Delivery was prompt (James has always been outstanding in my experience) and the kits were in good order: all parts present and accounted-for, all ICs carefully packed in foam, resistors etc. in bags, nothing loose. The assembly instructions and other documentation, something more than 250 pages with diagrams, pictorials, schematics and all, were superb. There is plenty of tutorial material, applications suggestions, a real professional job.

The kits were together easily in about 8 hours total. I have had some kit-building experience (a few Heathkits and an Eico scope) so I won't suggest that this is typical of a total novice, but with taking only reasonable care in assembly I managed to get it together without any real trouble. I will mention that the paddle switches were the only problem since they have to line up pretty exactly for proper mechanical alignment of the case. But then the real test:

It worked the first time I turned it on!

Now, those of us who only know KIMs don't appreciate the meaning of that statement, since KIMs are notoriously well-designed and seem to be darned hard to screw up. But for the rest of us, anything that has 40-odd ICs, about 100 resistors, a raft of capacitors, switches, and assorted oddities, plus three PC boards (glass epoxy with plated-thru holes by the way) is just about guaranteed not to function perfectly the

first time. Well, it did.

Moreover, it does everything it claims to do. Granted, there are so many different ways to use it, different combinations of controls and inputs, that it is difficult to say that every possible combination of functions is without flaw. I can say that I have used it for months, just about every day, without finding any failures.

Enough plaudits; what about drawbacks? OK, the display is in ones and zeroes. . .why not true hexadecimal. Cost, probably, and also versatility (remember it also wants to be able to display in octal) since it would require another ROM to do the character generation. There is one advantage to ones and zeros, tho', when you're looking for hardware problems. For instance, if you see the sequence: E37FA2B6 it is not obvious that bit 1 is stuck at logic 1, is it? It's a little bit easier like this:

```
1 1 1 0
0 0 1 1
0 1 1 1
1 1 1 1
1 0 1 0
0 0 1 0
1 0 1 1
0 1 1 0
```

but then again there are those of us who don't remember that 1 0 1 0 1 1 0 0 1 1 1 0 actually spells ACE. Well, if I can learn to convert from binary to hex, anybody can. It's definitely worth it.

The cost seem high? We have spent at least $245 to get started with KIM, and before we're done I'll wager the average home system runs well over a grand, and probably more than that. What you must consider is that the LA is not limited to just examining the bus lines on your micro. It will do a fine job of general monitoring of any logic system, check out your incoming parts for proper functioning (it sure beats 1k resistors and LEDs for this), is a tremendous assist in the design of asynchronous logic (like TV typewriters and I/O devices in general), and the ability to have a record of what happened just before your 6502 flew off into upper memory and hid from you is really nice. There's just no other device that will do it. Oh, pardon me, there are other LAs on the market, but for one with the 100A/10's capabilities expect to spend 4 figures, not 3, and the first one won't be a "1" either. ©

try PSUEDO CODE

Eric Rehnke

Just got done reading an article called "Sketchcode" which was written up in the May-June issue of PEOPLES COMPUTERS. The purpose of the article (authored by Todd Voros) is to define a sort of meta-language which itself can be used for designing structured programs and provide the programmer with a clear view of the flow of control within a program. This meta-language must be as applicable at high level (Basic, Fortran) as well as low level (assembler) programmers. Structured programming is sometimes known as GOTO-less programming.

Voros goes a long way in explaining the concepts and providing plenty of examples for this psuedo-code, which he dubs "sketchcode". Rather than try to describe the technique, here's an example:

```
COUNT = 1
DO WHILE (COUNT less than 11);
  PROCESS
  COUNT = COUNT + 1
END;
```

How's that for machine independent programming? Can you imagine how helpful it would be if all the magazines published this sort of documentation with every program that was published? By the way, an example program was presented in the article and converted from Sketchcode into both 8080 and 6502 assembly language just to be certain the reader has a complete understanding of this meta-language in a real implementation situation.

I have a couple of programs that I will try to rewrite using Sketchcode to clean up some very kludgey control flow. Maybe I'll present them in a later issue. ©