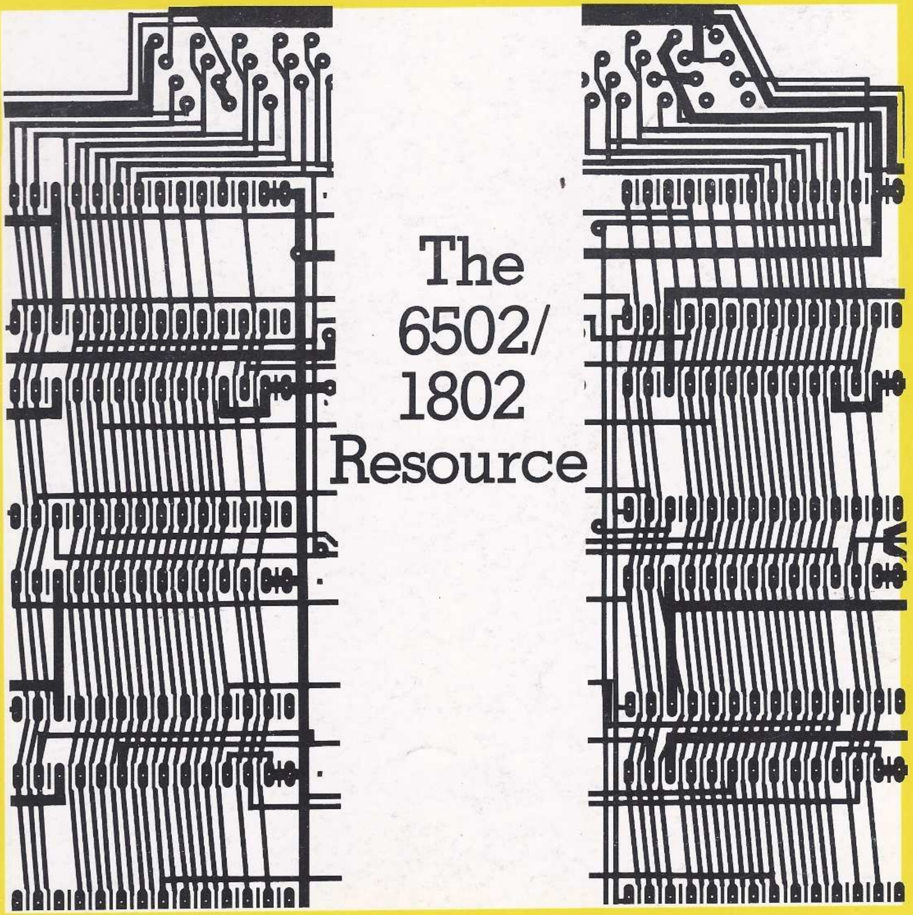
APRIL/MAY, 1980. ISSUE 1. $2.00

compute II.

The Single-Board COMPUTE™



The 6502/ 1802 Resource

Table of Contents

Advertiser's Index

Subscription Information:
U.S. $9.00/6 issue year

CANADA $12.00/6 issue year (U.S. funds)

Europe: Air Subscription $22.50 U.S./6 issue year

Other areas: Inquire for air rates.

In the **United Kingdom** and **Europe, new subscribers and dealers** should contact:

L.P. Enterprises
8-11 Cambridge House
Cambridge Road
Barking, Essex
England IG1 18NT
(United Kingdom and Northern Europe)

Ing. W. Hofacker GMBH
8 Munchen 75
Postfach 437
West Germany
(Germany, Switzerland, and Austria)

Staff of COMPUTE/compute II.:
Robert C. Lock, Editor/Publisher
Carol Holmquist Lock, Circulation Manager.
Larry Isaacs, Software/Hardware Lab.
Joretta Klepfer, Editorial Assistant.
Marlene Pratto, Educational Assistant.

COMPUTE. & compute II. receive continuing assistance from the following persons:
Harvey B. Herman, University of North Carolina at Greensboro, Dept. of Chemistry. Editorial Assistance.
Gary Dean, The Design Group, Greensboro, N.C., Art Direction/Design Consultation.
Jim Butterfield, Toronto. Editorial Assistance.

The following writers contribute to our publications on a regular basis as Associate Editors:
Gene Beals, 115 E. Stump Road, Montgomeryville, Pennsylvania 18936.
Len Lindsay, 1929 Northport Drive #6, Madison, Wisconsin 53704.
Roy O'Brien, Post Office Box 426, Beaumont, California 92223.
Eric Rehnke, #61, 540 S. Ranchview Circle, Anaheim Hills, California 92807.
COMPUTE. & compute II. are published by Small System Services, Inc., 900-902 Spring Garden Street, Greensboro, North Carolina 27403. Telephone: (919) 272-4867.
PET is a trademark of Commodore Business Machines, Inc.
Apple is a trademark of Apple Computer Company.
Atari is a trademark of Atari, Inc.

Authors of manuscripts warrant that all materials submitted to COMPUTE. are original materials with full ownership rights resident in said authors. By submitting articles to COMPUTE. authors acknowledge that such materials, upon acceptance for publication, become the exclusive property of Small System Services, Inc. Unsolicited manuscripts not accepted for publication by COMPUTE. will be returned if author provides a self-addressed, stamped envelope. Program listings should be provided in printed form as well as machine readable form. Articles should be furnished as typed copy with double spacing. Each page of your article should bear the title of the article, date and name of the author.

Address all manuscripts and correspondence to COMPUTE. Post Office Box 5119, Greensboro, N.C. 27403. Materials (advertising art work, hardware, etc.) should be addressed to COMPUTE., 900 Spring Garden Street, Greensboro, N.C. 27403.

We assume no liability for errors in articles or advertisements. Opinions expressed by authors are their own.

Address all articles, circulation questions and other inquiries to:
COMPUTE./compute II.
P.O. Box 5119
Greensboro, NC 27403
(919) 272-4867

The Editor's Notes

Robert Lock

Welcome to compute II

Well here it is. I'll cover the highlights of the reasons for **compute II** for those of you who didn't see the announcement in Issue #3 of COMPUTE. We were bursting at the seams after just three issues, and the logical split in the magazine seemed to be to move the single board machines into a magazine of your own. Everyone ends up with more magazine this way, and we can provide better coverage. compute II will be a bimonthly, with months of publication alternating with COMPUTE.

WRITE MORE!

One of the things that helped us make the move into compute II was the onslaught of quality material we were getting. Now that you have your own magazine, we'll be using material faster and faster. Please keep it coming. Our lead times are among the shortest in the industry. I'm interested not only in major ''feature'' type articles, but short programming notes, articles, and applications as well. One more comment regarding applications. You engineers out there should take the time to sit down and write up your ''routine'' applications. While they may be routine to you, if you'll take a page or two of time to explain in simple terms the design and implementation of an application on your sbc, you can provide a service for everyone.

We want to continue sections of the magazine that are devoted to resources, educational, industrial and business applications. Take the time to jot a note. It helps.

Major Features

Gene Zumchak's Column, Nuts and Volts, has now moved from COMPUTE to compute II. I think you'll find it extremely useful. In this Column, Gene continues his discussion on interfacing non-family devices to the 6502.

Larry Isaacs, of COMPUTE's staff, kicks off a series in this issue on the IEEE-488 instrument bus. He'll be implementing it on a SYM, under software control, during the next several issues. (An aside to you SYM-owners who wrote about recent comments in COMPUTE #2. . . we use 'em.)

Eric Rehnke continues his column, The Single-Board 6502, and I assume he's pleased that he's not stuck in the back of the magazine anymore.

On the Goals of compute II.

We expect this magazine to be a resource for readers... at all levels. We're as interested in beginning basic/machine language tutorials as in advanced applications articles. We're also seeking seminar announcements, club notes, grant announcements and so on. Take the time to drop me a note. compute II pays for all published articles, regardless of size. For more information, give us a call.

Educators: If you're running an SBC applications/learning lab, let me know. We'd like to start a series on the fine art of funding/teaching an applications lab. Please include a copy of your course bibliography. We'll try to keep a running summary of recommended texts.

The Fifth West Coast Computer Faire (March 14,15,16)

Well, just returned from the show and was most impressed with the ''state of our industry''. We'll have a full photo essay and comments on the show in issue 2. One significant highlight: Microsoft's new consumer products division introduced their first piece of hardware: **The Z-80 Softcard.** It plugs into the Apple, comes with Microsoft disk BASIC and CP/M and opens up whole new worlds. Price is $349.00 with June deliveries. Amazing.

Other notes. . .

Robert A. Peck (Datapath, P.O. Box 2231, Sunnyvale, CA 94087) has introduced his **SYM-1 Hardware theory of Operations Manual.** At $6.00 it looks good. Full review next time. Robert is author of the SYM/KIM Appendix to the First Book of KIM.

Sol Libes, a well-respected industry watcher, has introduced a new bimonthly: **S-100 Micro Systems.** Issue #1 includes a detailed presentation of the IEEE S-100 Standard. For more information contact S-100 Micro Systems, P.O. Box 1192, Mountainside, N.J. 07092. Pre-paid charter subscription price (6 issues) is $7.50 through April 30.

GPA Electronics, Inc. is introducing their **Universal Interface,** a double-sided PC board with on-board 6502, 2K ROM (or 1K), plus 1K of static RAM. Add to this the four parallel ports and RS-232 interface at the (already mounted) DB-25 connector, and you have an interesting applications tool. Without EPROM, the device is $249.95. If you can't wait for our review next time, you can obtain additional information from: GPA Electronics, Inc., P.O. Box 931, Laytonville, CA 95454 (707) 984-6270.

Tell them **compute II** sent you. ©

The Single-Board 6502

Eric Rehnke

You asked for it-you got it-your own magazine. How 'bout that?

Let's give Robert Lock a resounding 'Well Done' for giving KIM, AIM, SYM and OSI users our own magazine to expand into. Let's also understand that since we now have a greater vehicle for expressing ourselves *in,* we have an even greater responsibility for expressing ourselves. And since **compute II** pays for your time as well as putting your name up in lights, you have no excuse but to turn that writer on that lives and breathes inside every computer freaque. **compute II NEEDS YOU!!!**

EPROM SIMULATOR

Ever since I first saw an ad for the Pragmatic Design's 'Debug Memory' (DBM-1) board I became fascinated by the design concept of this rather unique RAM board.

Basically, the DBM-1 is a 2Kx8 RAM board for the S-100 bus which can simulate a 2K ROM or EPROM (or 2-1K ROMS) to a target system.

In other words, the same 2K block of RAM that appears to the development system as normal random-access memory, appears to the target system as a 2K ROM/EPROM through a cable plugged into that ROM socket.

It can achieve this nature by viture of its dual-port RAM design. This means that the same memory can be accessed from two separate systems each with its own address and data bus.

The benefit of this type of arrangement becomes obvious when you consider that everything that is written to the 2K block of RAM in the development system also appears in the ROM/EPROM address space of the target system.

Now, instead of developing a program inside your large development system, then burning an EPROM, and installing it in the target system to see if it works, you simply write the program into the RAM on the "Debug Memory" board.

It will then appear in the target system and can be tested immediately. If there's a problem, no EPROM to erase and reprogram, simply write the corrected software to the DBM-1. The updated software automatically appears in the target system.

Saves lots of time!!!

The biggest benefit of the EPROM simulator will be gained when developing software for small, dedicated controllers which have no built-in monitor software of their own. But, using the simulator will also save lots of time when writing programs for semi-smart single-board computers like AIM or SYM, for example.

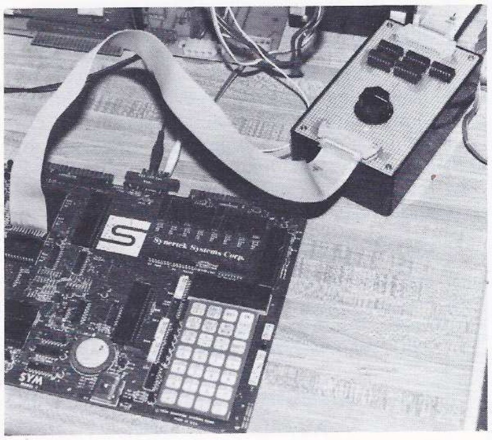
Another, not so obvious, use of the simulator is for developing a character generator for a video board. The video board can even be plugged into the development system itself.

(Most memory-mapped video boards also use this dual-port RAM concept in their design).

Of course, for the simulator to function most efficiently, the assembler in the development system must be able to assemble an object program to a location in memory other than specified by the program counter equate in the source code of that program. This is called 'assembling with offset'.

Say the EPROM socket in the target system resides at $F800-$FFFF in that system while the EPROM Simulator is addressed at $C000 in the development system. To be able to have a program assembled to run at $F800 while actually residing at $C000, it needs to be assembled with an offset. Get it?

My assembler (from HDE) has this capability sc I decided to design my own EPROM Simulator. (I couldn't use the DBM-1 since it was designed for the S-100 bus).



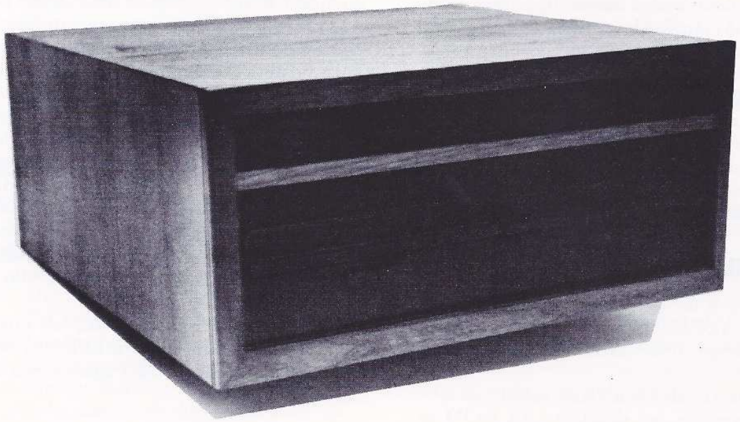
EPROM SIMULATOR shown plugged into a SYM for easy software development.

HDE inc.

BOX 120
ALLAMUCHY, N.J. 07820
201-362-6574

HUDSON DIGITAL ELECTRONICS INC.

ANNOUNCING THE HDE OMNIDISK 65/8



Now, you can "plug in" the latest in a successful series of flexible disk systems developed by HDE for the KIM, SYM and AIM microcomputers. The OMNIDISK 65/8 is a complete system, using 8 inch soft sectored diskettes with a formatted (IBM Standard) capacity of 256K. Of course, a disk formatting function is included as are system supporting utilities for file renaming, disk packing, copy (dual systems) and others.

TED, a full featured, line oriented editor is standard in KIM and SYM based versions to get you up and running on your project in a hurry. The AIM version uses the on-board editor. With the OMNIDISK 65/8 you can concentrate on your problem, the disk supports you all the way.

OMNIDISK 65/8 is available in an attractive walnut wood cabinet, or unpackaged for OEM applications in dual and single drive configurations. The HDE disk controller is a state-of-the-art 4½" by 6½" card electronically compatible with the 44-pin KIM-4 bus structure. The controller and disk-driver are designed to operate with the popular Shugart 801-R and compatible devices.

The OEM single drive is $1195, the dual, $1895 and the dual in the walnut cabinet, $2200. Price is another reason to step up to the proven quality of an HDE system.

HDE PRODUCTS – BUILT TO BE USED WITH CONFIDENCE
AVAILABLE DIRECT OR FROM THESE FINE DEALERS:

JOHNSON COMPUTER
Box 523
Medina, Ohio 44256
216-725-4560

ARESCO
P.O. Box 43
Audubon, Pa. 19407
215-631-9052

PLAINSMAN MICROSYSTEMS
Box 1712
Auburn, Ala. 36830
800-633-8724

LONE STAR ELECTRONICS
Box 488
Manchaca, Texas 78652
612-282-3570

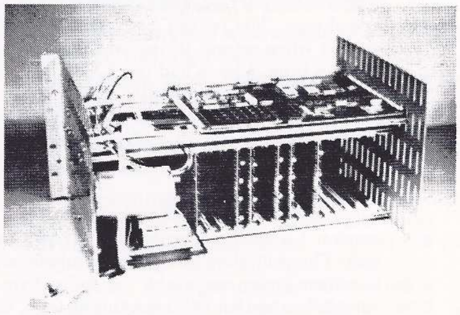
PERRY PERIPHERALS
P.O. Box 924
Miller Place, N.Y. 11764
516-744-6462

HDE inc.

BOX 120
ALLAMUCHY, N.J. 07820
201-362-6574

HUDSON DIGITAL ELECTRONICS INC.

THE HDE CARD CAGE



Shown With KIM-1 (not included)

VERSIONS

KIM*	**AVAILABLE**
AIM*	1st Qtr. 80
SYM*	1st Qtr. 80

$525.00
Complete With Power Supply

Now you can expand your 65XX single board microcomputer into a powerful microprocessor based system with the 19" (RETMA standard) HDE DM816-CC15 Card Cage. The DM816-CC15 has virtually all of the features you need for even the most demanding situations. Complete with power supply, backplane, card guides and supports, the HDE DM816-CC15 accepts state of the art 4½" wide cards permitting your system to remain a compact configuration, while expanding with a variety of functions.

HDE has developed the DM816-CC15 for the demanding industrial marketplace. Consequently, you can design your KIM*, AIM* or SYM* based installation using RETMA standard cabinet or rack components. Sufficient clearance has been included for custom front panel switches, lights and controls as well as cable and fan installation at the rear. The microcomputer is mounted to permit convection cooling in all but the most densely packed situations.

The self-contained power supply is ratea +8 VDC at 12 A and ±16 VDC at 3 A (both unreg.). The backplane, with the standard S44 bus, accepts up to 15 cards and has on board 5 VDC and 12 VDC regulators. In addition to power on reset, the backplane includes the logic connectors for remote reset.stop and single step as well as cassette and 20 mA loop terminal I/O. Provisions for data and address bus termination are included. Two 16 pin DIP pads are available for unique requirements and the microcomputer application and expansion connectors are extended to the backplane further increasing the utility of the total package.

Other HDE products include:

- 5¼" and 8" single/dual disk systems
- 8K static RAM memory
- Prototyping cards
- Software (disk and cassette)
  - Text Editor (TED)
  - Text Output Processing System (TOPS)
  - Assembler (ASM)
  - Comprehensive Memory Test (CMT)
  - Dynamic Debugging Tool (DDT)

Watch for announcements:
EPROM Card, RS232 Card, PIA Card, DAC Card

* KIM Is a Commodore product
* AIM is a Rockwell International product
* SYM is a Synertec product

HDE PRODUCTS – BUILT TO BE USED WITH CONFIDENCE

AVAILABLE DIRECT OR FROM THESE FINE DEALERS:

JOHNSON COMPUTER
Box 523
Medina, Ohio 44256
216-725-4560

ARESCO
P.O. Box 43
Audubon, Pa. 19407
215-631-9052

PLAINSMAN MICROSYSTEMS
Box 1712
Auburn, Ala. 36830
800-633-8724

LONE STAR ELECTRONICS
Box 488
Manchaca, Texas 78652
612-282-3570

PERRY PERIPHERALS
P.O. Box 924
Miller Place, N.Y. 11764
516-744-6462

Also, my version is rather more simplified than the DBM-1 since there were some features I didn't feel I needed - like trap address comparators, daisy chaining, etc.

I built it on an HDE prototyping card and designed it to simulate the 2708, the TI 2716 or the Intel 2716. This should satisfy most of the requirements. Later, when 4K EPROMS get cheaper, I'll build a version to simulate them.

So far, I've used the simulator to speed up software development for my SYM system, look forward to using it with my AIM system, and will use it to develop a PET character generator for an upcoming video board design.

As I have found, the EPROM Simulator is a development tool which can really save time and improve the efficiency of the system designer.

MORE FROM HDE

I have a difficult time keeping up with this company at times. They're surely not resting on their past accomplishments.

The first thing they've done is add conditional assembly directives to their already formidable 2-pass, disk-based assembler.

Besides the conditional assembly directives. IFZ (IFZERO), .IFN (IFNOTZERO), and .EIF (END IF), there are other special directives like .SOR (SORT SYMBOL TABLE), and .COV (INSTRUCTION COUNT) which enable the HDE Assembler to offer capability approaching that of the 6502 Cross Assembler available on time-sharing systems. HDE's Assembler actually comes out on top in the area of conditional assembly (that capability is not available in the Cross Assembler). The only noticeable features of the Cross Assembler which aren't available in the HDE counterpart are the symbol table cross referencing facility and the macro pre-processor. As soon as HDE adds these capabilities to their system (and they're in the works, according to Hudson Digital Electronics), they will have even more features than the Cross Assembler.

Besides that, HDE added the CHAINING function to their already much enhanced version of MICROSOFT Basic. This lets you run Basic programs which are actually too large to fit into your memory. According to HDE, a disk file system is next on the list for their not-so-basic BASIC.

I've been playing with the new conditional assembler for several days, their enhanced Basic for about a month, and continue to be impressed with the software (and hardware) this company produces.

As you may know, I have a KIM system with a couple of full-size HDE floppies and can wholeheartedly recommend this company's stuff to anyone who wants a darn good 6502 development system. (HDE, POB 120, Allamuchy, NJ 07820 PHONE 201-362-6574)

TIPS FROM READERS

From Christopher J. Flynn (2601 Claxton Dr., Herndon, VA 22070)...

KIMSI fans, if they don't already know, should get a copy of Forethought's application note A15. It describes the use of the popular EXPANDORAM dynamic memory board in the KIMSI. A two chip refresh controller is added to the KIMSI breadboard area. The refresh controller causes the 6502 to wait when refresh is needed thus slowing things down a little. I have had the EXPANDORAM running for two or three months now with no problems. It's a very good way to get a lot of RAM for a moderate price without requiring a huge power supply.

From Sam Sturgis, (86 Fisher St., Medway, MA 02053)...

Micro-Ade Assembler users can fix a bug in version 1.0 by changing location $2AF9 from $49 to $48. This will allow use of the symbolic argument modifiers /, +, and - with the define byte (=) psuedo instruction. Without this change, the modifiers are ignored by the assembler, making it extremely difficult (if not impossible) to construct symbolic tables.

From an anonymous reader. . .

AIM users who would like to have darker print from the printer can parallel R21 with an additional 4.7K resistor. I'm not sure how this affects the life of the printhead, but I'm sure Rockwell wouldn't approve of this mod.

HARDWARE REVIEW

SPEAK & SPELL INTERFACE (available from East Coast Micro Products, 1307 Beltram CT, Odenton, MD. 21113).

The thought of having your computer actually speak to you has probably crossed about every hobbyists mind at one time or another. The fact that very few of us ever hear a word uttered from our computers is probably due more to the cost of such speech output devices and not to their availability as most of them are in the $400-$600 range.

When Texas Instruments introduced the SPEAK & SPELL for about $50, I had a feeling that a truly "cheap" speech output device was just around the corner.

I forget just how long that "corner" turned out to be, but it must have been about a year before Dave Kemp of East Coast Micro Products introduced his SP-1 interface for for the SPEAK & SPELL.

Dave initially sent me his information package which acquianted me with the SP-1 interface and discussed some of the basics of the SPEAK & SPELL's method of speech synthesis.

The most important thing that I learned was that

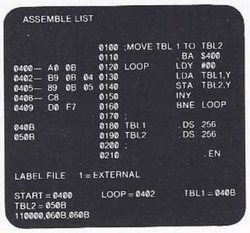
Skyles Electric Works

Presenting the Skyles MacroTeA

The Software Development System For the Professional Programmer and The Serious Amateur Software Hobbyist

...and for anyone who needs to automate dedicated industrial measurement and control applications.

The Macro TeA, designed for use with the **Commodore PET** to create a remarkable synergism: a complete, integrated **software development system** for the 6502...the only 6502 system not requiring a separate disk drive. With over 60 commands for your complete machine language programming.



```
ASSEMBLE LIST

                  Ø1ØØ :MOVE TBL 1 TO  TBL2
                  Ø11Ø             .BA  $4ØØ
Ø4ØØ— AØ ØB       Ø12Ø LOOP        LDY  #ØØ
Ø4Ø2— B9 ØB Ø4    Ø13Ø             LDA  TBL1,Y
Ø4Ø5— 89 ØB Ø5    Ø14Ø             STA  TBL2,Y
Ø4Ø8— C8          Ø15Ø             INY
Ø4Ø9  DØ F7       Ø16Ø             BNE  LOOP
                  Ø17Ø :
Ø4ØB              Ø18Ø TBL1        .DS  256
Ø5ØB              Ø19Ø TBL2        .DS  256
                  Ø2ØØ :
                  Ø21Ø                  .EN

LABEL FILE   1=EXTERNAL

START=Ø4ØØ        LOOP=Ø4Ø2        TBL1=Ø4ØB
TBL2=Ø5ØB
11ØØØØ,Ø6ØB,Ø6ØB
```

...a lightning fast...fast...fast.........
.....machine language assembler with true macro capabilities. Assemble 16K source text in less than ten seconds! A single name indentifies a whole body of lines. Macro and conditional assembly support. Automatic line numbering. Labels up to ten characters long; 10^{16} different labels.

Install permanently without tools and in less time than it takes to load an equivalent sized assembler/text editor program from tape. No tape loading ever. And no occupying of RAM memory space: the MacroTeA puts 10K of executable machine language code in ROM (98ØØ to BFFF) and 2K in RAM (9ØØØ to 97FF).

...a super powerful text editor. 26 commands with string search and replace capability. Manuscript feature. Test loading and storage on tape or disks. Supports tape drives, disks, CRT, printers and keyboard.

...an enhanced monitor with 11 powerful commands for program debugging and final polishing.

...with a warm-start button, on a 12 inch cable. Reset the PET but not the 1792 bytes of object code in the **Macro TeA** RAM memory.

...a completely solid state firmware system ...all in ROM and RAM. No tapes to load. The system is available from the time you turn on your PET to the time you shut it off.

15 chips on a single high quality printed circuit board; interfaces with PET's parallel address and data bus or with Skyles Memory Adapter. A comprehensive 170 page manual is included

Truly, there is simply no other system of this magnitude at anywhere near this price. **$395.00***

(With any Skyles Memory Expansion System, **$375.00**

**California residents: please add 6% or 6.5% sales tax as required.*

VISA, MASTERCHARGE ORDERS CALL (800) 538-3083 (except California residents)
CALIFORNIA ORDERS PLEASE CALL (408) 257-9140

Skyles Electric Works

231E South Whisman Road
Mountain View, CA 94041
(408) 735-7891

"...SP-1 does not turn SPEAK & SPELL into a black box speech synthesizer which can be used to add voice output to a users' chess, bridge, or Star Trek program. It is a tool which will allow the serious experimenter to investigate speech synthesis at a cost far below other commercially available synthesizer boards."

Later, I got the chance to review the SP-1 interface package.

The documentation for the SP-1 turned out to be quite a bit more than I expected. Only 2 of the 28 pages were devoted to assembling the interface board which indicated that this package was intended for the advanced experimenter. The software drivers were presented in well commented source listings as well as detailed explanations of how the SP-1 operates and a special section on SPEAK & SPELL theory of application. Software flow charts and a bibliography were also included.

I rate the documentation as excellent. Plenty of information here.

I hooked the interface to my SYM, loaded the hex dump software and shortly thereafter, SYM was giving me a vocal hex dump of its memory contents.

Shades of DEMON SEED. SYM was actually talking to me!!!

Other software was included to enable one to pull word data out of SPEAK & SPELL for analysis, but I never got around to using it. It would have been almost anti-climactic after hearing SYM actually talk.

The vocal hex dump software is the only piece of "black-box" software you get with the SP-1 that you can plug in and use. It's a great demo of the possibilities. The other software included is meant to aid the experimenter in working with the SPEAK & SPELL and developing things further.

By the way, no modifications were necessary to the SPEAK & SPELL as the SP-1 plugged into an internal edge connector.

The SP-1 interface package sells for $49.00 and is available directly from East Coast Micro Products.

If you want to tinker around with speech synthesis at a low price, check out the SP-1.

SOFTWARE REVIEW

Assembler/text Editor (sold by M.S.S. Inc., POB 2034, Marshall, TX 75670).

What can you get for $25.00 these days? Well, if you like to tinker with software, you could get the beginnings of an interesting little assembler/editor. Or, you could get an education in what makes an assembler tick.

Actually, you do get more than just the beginnings of an assembler. It's almost complete.

What's all this nonsense you ask? OK, I'll try to describe it.

For $25.00 you get a standard, no frills line editor and 2-pass assembler that resides in slightly over 2K of memory. The assembler mnemonics are close to MOS Technology's standard. How close? The actual mnemonics correspond perfectly. It's the operand formats that differ. I'm sure you could get used to it, though.

The best thing about the package is that you get a source listing for it included for no extra money. It's not very well commented, but it's better than nothing. A serious student of the 6502 dialect should have little problem figuring out what's going on.

The reason I said that the package is ALMOST complete is because there's no built-in means of saving or loading assembler source code to/from cassette (or other mass storage device). If you want to do that, you'll have to add it yourself. They do, however present an example of a routine to add a new command to the system so, if you know what you're doing, you could probably add the necessary cassette support routines.

The fact that the source listing is included opens a rather exciting possibility. It's possible to make the assembler into a cross-assembler for some other CPU-like the 1802, for example. Then you could get that 1802 Simulator program by Dann McCreary and have a complete 1802 development system on your 6502!!!

Does that turn you on?

This assembler and its associated documentation is NOT for beginners! But, could be a good value if you know what you're getting into.

BLUE SKY CORNER

If you also have an amateur radio operator's license, you have the opportunity to connect your computer to some radio gear and communicate with fellow amateurs through a rather sophisticated satellite-born radio repeater system.

Since the FCC has approved the use of ASCII for some satellite communications, the use of computers is a natural. And once you have a computer hooked up, all kinds of things are possible.

How 'bout a fully automatic, cross country communications network?

Is there anyone out there working along these lines? Or, do you know of any groups or publications dedicated to such an end? I'd sure like to hear from you.

Remember the Adventure Game for KIM? (Issue 3, COMPUTE)

It's $24.95 and is available from ARESCO, Box 1142, Columbia, MD 21044 ©

Here, in full, is Gene's Column 1. A portion of this column was printed in Issue 3 of COMPUTE. We thought we'd start at the beginning for Issue #1 of compute II. RCL

Nuts and Volts

Gene Zunchak
Niagara Micro Design, Inc.
1700 Niagara Street
Buffalo, N.Y. 14207

With this article, I hope to begin a series of dissertations on the art and joy of doing it yourself. But first let me introduce myself. I'm Gene Zumchak. I graduated with an M.E.E. from Cornell in 1968. I paddled across Cayuga Lake and got my first job with a likable maverick named Robert Moog of Moog Synthesizer fame. I left my first, and one of the only legitimate jobs I've had in 1970. I moved to Buffalo and since that time have worked for a series of little flake outfits trying to find a winner, but at the same time, avoid having to work for a living. So far I've managed neither, but sure have had a lot of fun. But alas I think I'm getting closer. I now call myself Niagara Micro Design, Inc., and although the pay isn't that hot, the boss let's me play golf whenever the whim takes me (between thaws in July).

Anyway, I started drooling over micros in the early seventies when they first appeared in the electronic design magazines. Then in '75 an unknown semi company called MOS Technology made a big two month splash in the magazines (never to be heard from again) prior to a Western show. I clipped a coupon, sent in $25 and became the proud owner of a 40-legged centipede called a 6502. Now to give you some idea of just how fantastic that was, the 8080 chip set (8080, 8224, 8228) was selling for over $200. A few days after the chip arrived, I got a flyer advertising a single-board computer for just a few bucks more than the naked 8080 chip set. I ran to the bank, withdrew my life savings and sent in coupon number two becoming an even prouder owner of KIM-1 serial no. 00005. Soon afterwards, three friends and myself contracted to design and build five smart printer systems for Honeywell, without a TTY or assembler, let alone a development system. With little more than that KIM and a Simpson VOM we delivered five microprocessor controlled printers. Today, the same job would be a lot easier, but my "development system" is still just an expanded KIM. Anyone, in fact, can put together a development system for only a couple hundred dollars over the price of a KIM, SYM, AIM or other system. Putting together that development system is one of the topics I hope to get to in the future.

Presently I am working on a book entitled "Microcomputer Design and Maintenance" with the guidance of Jon Titus of the "Blacksburg Group" who hopefully will get it published. The topics presented in this series of articles in Compute will touch on some of the material in that book, as well as different material.

A dedicated micro system can be put together for under a $100. Why tie up your Apple or Pet to turn the furnace on, run your electric train, or program your wife's loom? Why not use your system as a tool to crank out dedicated controllers? I hope to show you how. Enough B.S. Let's start talking about hardware.

Read/Write Timing

The most important consideration in hardware design is read/write timing. It is not a complicated topic, but many "designers" avoid confronting it by surrounding a CPU with family chips (usually expensive) using circuits right out of the manuals. There's nothing wrong with the fancy family chips if you really need them. Oftentimes the most attractive chip may belong to another family. If you understand read/write timing, however, you may indeed be able to use a foreign chip.

Write Timing

The terms "reading" and "writing" always reflect the direction of data flow from the perspective of the processor. Thus in a "write" operation, data is presented by the processor to some external device, memory or output, and locked into that device. A bit of memory or output is a flip-flop. In memory, the output of the flip-flop can be read back into the processor. In "output", the output of the flip-flop is connected to the world. (In some programmable devices, an output can often be read back into the processor.)

The usual type of flip-flop used with a processor is the "D type" flip-flop. A D flip-flop has a "D" or data input, and a clock (strobe) input which is an edge-sensitive function. That is, data is presented to the D input and is transferred to the output when the active edge occurs (usually positive going). An edge triggered flip-flop's output can change only on a clock edge. An example is the TTL 7474 dual D flip-flop. A variation of this is the transparent latch. It too has a D or data input, but a Gate input, instead of a strobe. When the gate is true, the output follows the data and is transparent to the data. Data is locked into the flip-flop on a false going gate edge. The 7475 is a quad transparent latch. In both types of flip-flops, data is locked in with a clock or gate edge.

In any latching operation, the following sequence occurs: Data is presented to an input, a locking edge occurs, and finally the data is removed. In general, the data to be written exists before and after the locking edge. We are now ready to define the important parameters of a write operation. The "set-up" time is the minimum time the data must be present before the locking clock edge occurs. The "hold" time is how long the data must remain after the locking edge has gone away. The set-up and hold times for a 7474 flip-flop are only 20 and 5 ns. respectively. Since these times are so short, TTL latches can always be assured of working with MOS processors.

6502 Write Timing

A 6502 clock cycle is read cycle when the R/W line is high, and a write cycle when the R/W line is low. A cycle is divided into two (more or less) symmetrical halves. In the first half, the ø1 clock is high. The R/W line and the address lines change ø1. In ø2, data transfers occur. According to the spec sheet, the delay from the fall of ø2 to the beginning of the rise of ø1 can be zero (no max spec given). For zero delay, the ø1 clock is approximately ø2. However for external use, ø2 and ø2 are generally used. ø1 should not be used as a substitute for ø2. Figure 1. shows write timing for the 6502.

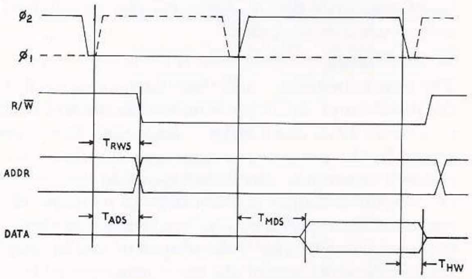


Figure 1. 6502 Write Timing

The R/W and address lines have a setup time (T_{RWS} and T_{ADS}) as 300 ns. maximum after the beginning of ø1. Data is available in a maximum of 200 ns. after the rise of ø2 (T_{MDS}). ø2 has a minimum width of 430 ns with a one micro second clock (1MHz). Thus data is available a minimum of 230 ns. before the fall (locking edge) of the ø2 clock. The data is held beyond the fall of ø2 for a minimum of 30 ns. (T_{HW}). Thus the 6502 is guaranteed to write successfully to any device with a set-up time requirement of 230 ns. or less, and a hold time requirement of 30ns. or less. Implicit in the timing is that the falling edge of ø2 is the locking edge. The 6502 generates no write strobe. A write strobe must be fabricated by NANDing ø2 with the inverted R/W signal, R/W. This gives a strobe that goes low during ø2 only for a write cycle. In family devices, ø2 and R/W are applied separately and the gating is performed internally.

6502 Read Timing

In a read operation, an external device puts its data on the data lines and it is locked internally into the 6502 at the end of ø2. This timing is shown in figure 2.

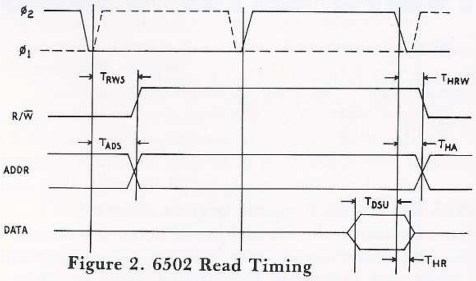


Figure 2. 6502 Read Timing

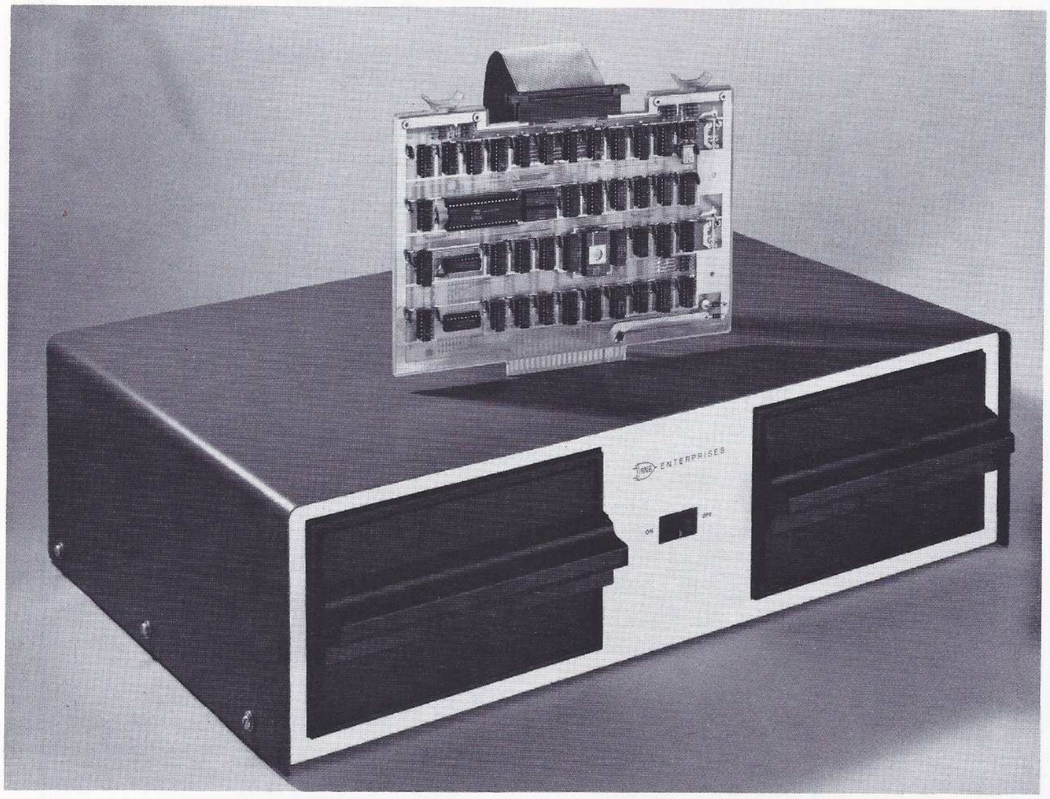
As in a write cycle, the address and R/W set-up time is a maximum of 300 ns. into ø1. The data set-up time is a minimum 100 ns. before ø2 (T_{DSU}). In addition, the data must be held a minimum of 10 ns. past ø2. Both the set-up and hold times for the 6502 in a read operation are quite short making it easy to read I/O devices from any MOS family.

When a device is read, the data is gated onto the bus by a read gate generated from the R/W line and a decoded address. The strobing is done internally by the processor. A gating signal allows the data to overlap the strobe. It will be seen in many 6502 systems that read gates often incorporate ø2. It would appear that this would cut off the data at strobe time, and violate the hold time requirement. First of all, the tri-state gate buffering the data probably has a delay of at least 15 ns. Secondly, if ø2 has gone through one or two gates of buffering, it will occur 15 to 30 ns. after the ø2 seen at the 6502. Thus the data will remain on the data bus 30 to 45 ns after ø2, even though ø2 appears to be used to cut data off.

Interfacing Non-family Devices

Meeting the read requirements of the 6502 is easy for almost any I/O device in any family. The problem occurs when the 6502 tries to write to the relatively sluggish devices of other families. Some of these chips have relatively long set-up and hold time requirements for a write operation. The set-up times are not often a problem, however. In order to be on the safe side, the chip manufacturer often quotes a rather conservative minimum hold time spec (like 100ns.) while quoting a typical spec of 30 or even 0 ns. To insure reliable performance, you must meet the minimum spec. The 6502 will guarantee only a 30 ns write hold time. What do you do when you want to write to a Western Digital 1771 floppy

VAK-7 FLOPPY



The VAK-7 Disk System was specifically designed for use with AIM-65, SYM-1, and KIM-1 Microcomputer Systems. The VAK-7 will plug directly into the VAK-1 Motherboard or with the addition of regulators into the KIM-4* Motherboard. The VAK-7 is a complete full size (8") FLOPPY DISK SYSTEM. This system will READ, WRITE, and FORMAT both IBM SINGLE and DUAL DENSITY diskettes. Single-Sided is standard and Dual-Sided is optional. Our Single-Sided drives are set up so they can be converted at a later date to Dual-Sided by the factory, if your storage needs increase.

The VAK-7 system occupies a 4K address space. The system has a 1K block of D.M.A. RAM as a transfer buffer. Also, a 1K block of RAM reserved for D.O.S. pointers, drive status, and catalog information. The remainder of the address is occupied by the resident 2K MINI-DOS. This MINI-DOS is a complete set of subroutines to Read, Write, and Format.

DISK SYSTEM

The MINI-DOS is not a high level Disk Operating System, but contains all the elementary subroutines for implementation of a high level DOS. Since all the functions are in subroutines, the implementation of this system into a dedicated system is simplified.

MINI-DOS SUBROUTINES

- Block Move
- Seek Track
- Recalibrate Disk
- Sense Interrupt Status
- Read/Write Data
- Read/Write Deleted Data
- Format Disk/Test For Bad Sectors
- Initialize Disk
- Physical Copy (Disk to Disk)
- Self Test

The VAK-7 is an interrupt driven system, which uses the $\overline{\text{IRQ}}$ vector. Since this is an interrupt driven system, your system processor is only used to move data into or out of the 1K of DMA RAM, issue the command, and check status at the end of the disk operation. Your system processor is free to do other functions, during disk operations because the intelligent disk controller will complete the operation without tying up valuable processor time.

The VAK-7 System comes complete with Disk Controller Board, Interconnecting Cable, a Cabinet with Power Supply (for two Disk Drives) and one Disk Drive. The VAK-7 Controller can handle up to Four Drives.

SPECIFICATIONS:

- Completely assembled, tested, and burned in.
- Occupies address $9000-$9FFF for AIM-65, $9000-$9FFF for SYM-1, or $E000-$EFFF for KIM-1.
- IBM Format; Single Density (128 bytes/sector); Dual Density (256, 512, or 1024 bytes/sector).
- All IC's are in sockets.
- Fully buffered address and data bus.
- Standard KIM-4*BUS (both electrical pin-out and card size).
- Designed for use with a regulated power supply, but has provisions for adding regulators for use with an unregulated power supply.
- Dimensions: Board—10" wide x 7" high (including card-edge)
  Cabinet—23.5" wide x 6.5" high x 16" deep.
- Power Requirements: +5V DC @ 2 Amps.
  117V AC 60Hz @ 2 Amps.

**KIM-4 is a product of MOS Technology/C.B.M.*

PRICE:

Single-drive, 1-sided	$1,299.00
Dual-drive, 1-sided	1,898.00
Single-drive, 2-sided	1,499.00
Dual-drive, 2-sided	2,398.00

Plus Shipping	UPS	Mail (APO, FPO)	International
Single Drive	12.00	32.00	Shipped Air Freight. Freight charges collect.
Dual Drive	16.00	44.00	

For Alaska and Hawaii, use mail rates.

We manufacture a complete line of high quality expansion boards. Use reader service card to be added to our mailing list, or U.S. residents send $1.00 (International send $3.00 U.S.) for airmail delivery of our complete catalog.

RNB ENTERPRISES INCORPORATED

2967 W. Fairmount Avenue
Phoenix AZ. 85017
(602)265-7564

master charge VISA

controller chip with a longer hold time requirement? There are a number of ways to overcome this problem. First, however, we should consider an additional complication.

Earlier we saw that a little delay in the ø2 clock when used in a read gate was not a bad thing. However, delay in ø2 when used to generate a write strobe could be bad news. The 6502 provides for only 30 ns beyond the ø2 that it sees. A strobe generated from a delayed ø2 may in fact occur after the data has actually gone away. In this case, delay in the data path is beneficial. Any delay in ø2 greater than the delay in the data path takes away from the effective write hold time. Since few designers understand, let alone consider read/write timing, it is a wonder that most microcomputer systems work at all. In a typical single-board computer system, ø2 is generated on the board, buffered on some motherboard, and further buffered on the individual boards which plug into the motherboard. It is really questionable whether this buffering is really helpful. The fortunate thing for us all is that the hold time of the 6502 chips and the hold time requirement for I/O chips is almost always very conservatively speced.

Let us consider some approaches for getting a little extra hold time. Figure 3. shows perhaps the safest way. The data bus write buffer consists of a transparent latch followed by a tri-state gate. The latch is gated with ø2 so that the data to be written is locked into the latch at the end of ø2. The R/W line ordinarily extends well into ø1 before changing. In fact the R/W line is not guaranteed to change for at least 300 ns. (TRWS). To be on the safe side, we can always insert several gate delays in R/W so that the latched data will extend well beyond the end of ø2. This approach costs the addition of a pair of quad transparent latch chips, which may be very cheap insurance for reliable operation.

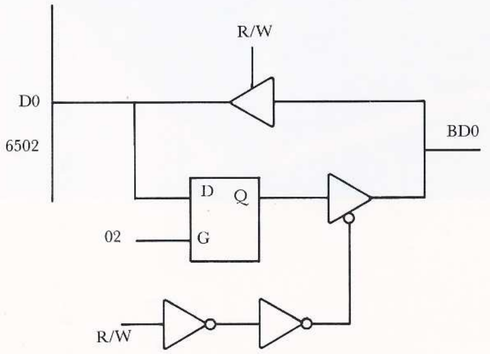


Figure 3. Write Hold Time Extender

Another approach to getting a longer write hold time is to generate a write strobe which finishes earlier than ø2. This can be accomplished by hitting a one-shot with the regular write strobe. In this method, however, the longer hold time is at the expense of set-up time. If you have lots of set-up time to waste there's no problem. The width of the pulse will have to be adjusted carefully. Figure 4. shows the shortened write strobe, W* and set-up time trade-off.

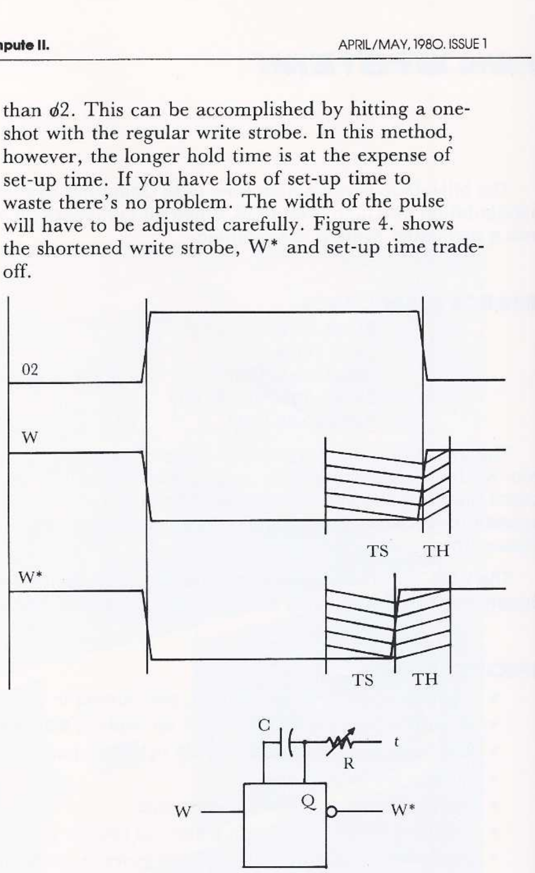
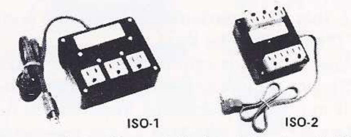


Figure 4. Shortened Write Strobe

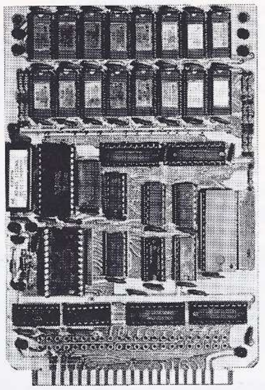
Access Time

When considering RAM or ROM, we need to consider access time, which is the time it takes the data to reach the output after the address is stable. For the 6502 at 1MHz, the addresses are good 300 ns. into ø1. The data must be ready 100 ns. before the end of ø2. This gives us about 600 ns. of available access time. (The spec sheet guarantees 575.) If we could only steal another 75 ns., we could use much cheaper 650ns. RAM and EPROM. If we pay a $5 premium for a 2MHz 6502, and run it at 1MHz, we can get an extra 150 ns. from a shortened address set-up time. The few extra dollars for the faster processor could save a lot more bucks ion a system with lots of memory.

In summary, no serious, or at least no creative design can be undertaken without an understanding of read/write timing. It requires accommodating the set-up and hold time requirements of the I/O devices with the corresponding times of the processor. Tricks like latching the write data can be used to overcome any discrepancies. The information is available from the device spec sheet. ©



OPTIMAL TECHNOLOGY, INC.
EP-2A-79
EPROM PROGRAMMER
ON
OFF
POWER
PERSONALITY MODULE
PROGRAMMING SOCKET
ON
OFF
PROGRAMMING VOLTAGE



Part 1: RS232 COMMUNICATIONS

Michael E. Day
2590 DeBok Road
West Linn, OR 97068

As more computer equipment is purchased by the small systems user, connecting this equipment together becomes a bigger problem, particularly when the equipment is made by different manufacturers.

One of the more common methods of connecting data communications equipment together is by way of the RS232 standard. However, even this has been cause for confusion, as there are various levels of implementation within the standard.

The purpose of this article is to provide sufficient information concerning the RS232 standard to allow proper implementation at the desired level.

The minimum level of RS232 consists of:
Pin 2 TXD (Transmitted Data -OUT-)
Pin 3 RSD (Received Data -IN-)
Pin 7 Logic Grnd

The 2nd level consists of the minimum level plus:
Pin 6 DSR (Data Set Ready)
Pin 8 DCD (Data Carrier Detect)
Pin 20 DTR (Data Terminal Ready)

The 3rd level consists of the other two levels plus:
Pin 4 RTS (Request to Send)
Pin 5 CTS (Clear to Send)
Pin 22 RI (Ring Indicator)

Pin 1 Protective Ground should be used at all levels; however, it is not required for proper operation.

Level 1 is normally used with equipment tied directly to each other, such as a terminal tied directly to a computer. Level 2 is normally used where some degree of handshaking is required, and is often found on accoustic couplers. The third level is used where a more detailed control of the information flow is required. This level will usually be found with auto answer modems.

This is a generalization of what will be encountered by the small systems user, and in no way implies that all equipment will follow these rules. Some equipment will need other special signals, or not use all of the signals within a specific level. Synchronous transmission will normally require additional special lines and will be described in detail later.

There have been three standards of RS232 produced--A, B, & C. RS232A is obsolete, and equipment using this standard is almost non-existent. RS232B is also obsolete; however, there is still some old equipment around that uses this standard. RS232B is basically the same as RS232C except that the Transmit Data and Receive Data signal levels are inverted; that is, a marking condition is a positive level rather than a negative level.

The following is a description of the full RS232C standard. It is not required that all signals be provided, and it may be implemented in part or in full.

Each data set has a standard 25-pin connector (Cinch or Cannon chassis-mount, female type DB-25S). The table below has the pin number, the circuit mnemonic, and description for each signal in the RS232-C interface. Unassigned pin may have a different function in each type of data set, so check the technical manual for pin assignments for each data set.

Pin Number	Mnemonic	Description
1	AA	Protective Ground
2	BA	Transmitted Data
3	BB	Received Data
4	CA	Request to Send
5	CB	Clear to Send
6	CC	Data Set Ready
7	AB	Signal Ground (Common Return)
8	CF	Received Line Signal Detector
9	--	(Reserved for Data Set Testing)
10	--	(Reserved for Data Set Testing)
11		Unassigned
12	SCF	Sec. Rec'd. Line Sig. Detector
13	SCB	Sec. Clear to Send
14	SBA	Secondary Transmitted Data
15	DB	Transmission Signal Element Timing (DCE Source)
16	SBB	Secondary Received Data
17	DD	Receiver Signal Element Timing (DCE Source)
18		Unassigned
19	SCA	Secondary Request to Send
20	CD	Data Terminal Ready
21	CG	Signal Quality Detector
22	CE	Ring Indicator
23	CH/CI	Data Signal Rate Selector (DTE/DCE Source)
24	DA	Transmit Signal Element Timing (DTE Source)
25		Unassigned

For timing and control interchange signals, the function will be ON when the voltage is more positive than plus three volts and OFF when the voltage is more negative than minus three volts. The table below illustrates the signal function voltage relationships.

INTERCHANGE VOLTAGE

	NEGATIVE -3 to -25	POSITIVE +3 to +25
Binary State	1	0
Signal Condition	Marking	Spacing
Function	OFF	ON

10-DAY FREE TRIAL

Send for our FREE Catalog

$100 FREE ACCESSORIES WITH 16K or 32K PET

When you buy a 16K or 32K PET, apply $100 toward PET accessories, FREE! Just indicate on your order that you have reduced the cost of your accessories by $100.

FREE **Terminal Package with 8K PETs**
See Special Below

SAVE $98

8K-Keyboard N	**$795**
16K-Keyboard B	**$995**
16K-Keyboard N	**$995**
32K-Keyboard B	**$1,295**
32K-Keyboard N	**$1,295**

B—large keyboard (graphics not on keys)
N—large keyboard with graphics symbols

PET ACCESSORIES

Commodore Dual Floppy Disk Drive	$1,295.00
Commodore Printer (friction feed)	$849.00
Commodore Printer (tractor feed)	$995.00
Second Cassette-from Commodore	$95.00
Commodore PET Service Kit	$30.00
Beeper-Tells when tape is loaded	$24.95
Petunia-Play music from PET	$29.95
Video Buffer-Attach another CRT	$29.95
Combo-Petunia and Video Buffer	$49.95
TNW Bi-dir. RS-232 printer X-face (see box below)	**$229**
KIM 1 (A Single Board Computer from Commodore)	$179.00

Lear Siegler's ADM-3A

The ADM-3A is industry's favorite dumb terminal for some very smart reasons:

- 12 in. diagonal screen
- Full or half duplex operation at 11 selectable data rates
- 1920 easy-to-read characters in 24 rows of 80 letters
- Typewriter-style keyboard
- RS-232 C interface extension port
- Direct cursor addressing

Our Low Sale Price $795

CompuMart has the following **Lear Siegler** products in-stock. Call for complete specs, options, & special OEM prices:

ADM-31 • High resolution • Full editing capabilities • 90 key keyboard. • All 128 ASCII characters.
ADM-31 SALE! $1,195 Reg. Price $1,450

ADM-42. The low-cost, semi-intelligent video display terminal that provides total flexibility of format, editing, interface & transmission. 2 pg. display expandable to 8. Detachable keyboard.
ADM-42 SALE! $1,595 Reg. Price $1,795

310 Ballistic Printer. Long-Life, Exceptional Quality.
SALE! $1,195 Reg. Price $2,045

SANYO MONITORS

SALE! **9-inch $169**

SALE! **15-inch $279**

Perfect for PET & APPLE Owners...

Sanyo cassette recorder	$55

apple II *plus*

$200 FREE ACCESSORIES

The new Apple II with Applesoft BASIC built-in! Eliminates the need for a $200 Firmware Card and includes new Autostart ROM for easy operation. This combined with the FREE accessories from NCE could save you up to $400 on a 48K Apple II system!

16K Apple II Plus—$1195 (take $100 in free accessories)
32K Apple II Plus—$1345 (take $150 in free accessories)
48K Apple II Plus—$1495 (take $200 in free accessories)

Apple II Accessories

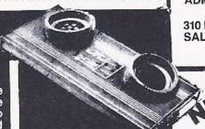
General Business	**$625**
PASCAL	**$495**
Integer BASIC ROM Card	**$200**
VISI-Calc	**$99**
Centronics Printer Interface	**$225**
Disk and Controller	**$595**
Parallel Printer Card	**$180**
Communications Card	**$225**
Hi-Speed Serial Card	**$195**
Firmware Card	**$200**

REMOTE TERMINAL

for only $98

A self-contained module and program cassette enables your PET to function as a 300 baud terminal. Supports Upper/Lower case, Rubout, Escape and all control functions. Output is TTL. Can be used with THE SOURCE. You will need the Terminal Option, Cat Coupler & EIA Cable to do this. Complete package which includes all 3 avail. from CompuMart for **$279.**

FREE **WITH PET PURCHASE**



CAT COUPLER

New 300 baud Originate/Answer Acoustic Coupler Looks good, works great

SALE $179

NEW! ZENITH COLOR VIDEO MONITOR

Zenith's first color video display designed specifically for computers.

This 13-inch monitor is Zenith's first color video display designed specifically for computers. Features include automatic color level, color processing and degaussing circuits.

Zenith Color Monitor $499.00

Excellent Performance—Looks Great

The TNW-2000

Bi-Directional RS-232 Serial Interface

Interface your PET to printers, modems, CRT terminals, plotters, paper tape readers and punches—or other computers. The TNW-2000 Serial Interface lets you connect your PET to RS-232 serial devices. Comes assembled and tested with cable to connect unit to an IEEE Standard or Commodore PET connector.

TNW Bi.-Dir. RS-232 printer X-face	$229

WE'VE GOT IT!

A Calculator, A System, A Whole New Standard.

HEWLETT-PACKARD'S HP-41C.

130 built-in functions • Alpha capability • 400 lines expandable up to 2,000 of program memory • Continuous memory • RPN logic.
"Customize it. Reassign any program or standard function to any keyboard location you want.
Enhanced programmability. Up to 6 levels of subroutines; 10 conditionals, 56 internal flags, specific loop control, indirect addressing, local & global branching. **HP-41C Calc. $288**

THE SYSTEM.

Memory Modules. Increase capacity to 319 registers or 2,000 lines. $45.
"Extra smart" Card Reader. Records programs & data onto blank mag-cards. $180.
The Printer. Portable, quiet, thermal operation. Upper & lower case; plotting routines. $320.

Application Modules **($45 ea.)** Standard Pac ROM, Statistics ROM, Math ROM, Finarcial Decisions ROM, Surveying ROM.

NCE/Compumart has the following quality Hewlett-Packard scientific and business calculators in stock. Call for complete product info and specs:
HP-31E-Scientific, $49.95 • HP-32E-Advanced scientific, $66.95 • HP-33E-Programmable scientific, $79.95 • HP-37E-Business management, $69.95 • HP-38E-Advanced financial, $108

CONTINUOUS MEMORY CALCULATORS
HP-33C-$114.95, HP-34C-$144.95, HP-38C-$144.95

Hazeltine

Super Terminal Sale!

1410 Reg. $900 **SALE! $749**

1500 Reg. $1,069 **SALE! $995**

PAPER TIGER 440SPE

The Graphics Printer for Apple II

Now you can print illustrations, block letters, charts, graphs, and more—all under software control. And with the expanded buffer, the Paper Tiger can hold the text from an entire 24-line-by-80-column CRT screen.

$1194.00 440 Reg. w/o graphics: $995

HEATH

The All-In-One Computer

Dual Z-80 Processors • Built-in 102K Floppy Disk • 16K to 48K RAM • 25 x 80 Character Display • Upper/Lower Case and Line Graphics • 80 Character Keyboard with Keypad • 8 User-definable Keys • Two BASIC's and Auto-Scribe Word Processing available • Can support CP/M

WH89 with 16K RAM	**$2,295**
WH89 with 32K RAM	**$2,445**
WH89 with 48K RAM	**$2,595**
WH17 Second Disk Drive	**$550**
Dual-port Serial Interface	**$85**
HDOS Operating System	**$100**
Microsoft BASIC	**$100**
Word Processing	**$395**

NCE/CompuMart SINCE 1971

DEPT **BY30**, 270 THIRD ST.,
CAMBRIDGE, MA 02142

To Order: 1 (800) 343-5504
In Mass.: 1 (617) 491-2700

VISA master charge

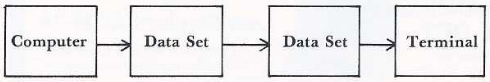
Member:
Computer Dealers
Association

IMPORTANT ORDERING INFORMATION
All orders must include 4% shipping and handling. Mass. residents add 5% sales tax; Mich. residents 4%, for sales tax.

Phones open from 8:30 a.m. to 5:30 p.m. EST Mon.-Fri. • P.O.'s accepted from D&B rated companies — shipment contingent upon receipt of signed purchase order • Sorry no C.O.D.s • All prices are subject to change without notice • Most items in stock for immediate shipment — call for delivery quotation • In the Ann Arbor area? Our retail store is open 11:00 a.m. to 7:00 p.m. Tues.-Fri., 10:00 a.m. to 5:00 pm. Saturdays (closed Sun. and Mon.)

If not satisfied, return your purchase with-in 10 days for full refund of purchase price!

The figure below illustrates a 2-wire, point-to-point, half duplex, and a telephone leased line which is always available to the customer.



Assume that the computer needs to transmit a message to the terminal. The computer's software brings up the "data terminal ready line" to its data set. If the data set is "ON" it will return "Data Set Ready" (interlock it) to the computer. When the computer wants to transmit, it raises the "request to send" level which tells the data set to turn on the carrier wave. The carrier wave is sent from the computer's data set over the telephone circuit to the terminal's data set. The terminal's data set, upon detecting the carrier, will raise the "Received Line Signal Detector" level to inform the terminal, in effect, that a message is about to be received. After a fixed delay time (strappable in some data sets) and after raising "Request ToSend," the computer's data set will return, "Clear To Send." The computer upon receiving the "Clear To Send" signal, can now start transmitting the message, as marks and spaces, on the "Transmitted Data" line to its data set. The data set converts the digital signals into frequency or phase-- shifted signals for transmission over the leased line to the terminal's data set.

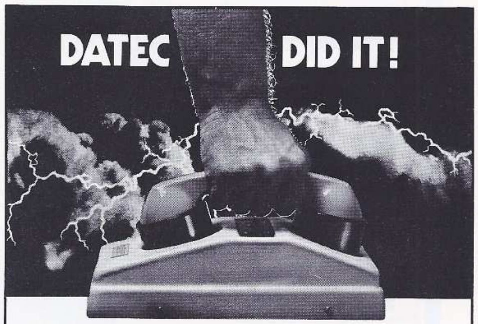
Most data sets contain a clamp circuit which clamps the "Received Data" line. "Received Line Signal Detector" level is not generated until after the carrier is detected. The clamp delay masks out all the possible noise on the line which occurs during the switching from either transmit to receive or receive to transmit.

If this was a synchronous operation, the clocking or synchronization of each bit would be done by the computer's data set. So that the computer knows when each bit must be placed on the "Transmitted Data" line, the data set sends clock to the computer on the "Transmission Signal Element Timing" line. This clock will be coincident with the leading adge of each data bit on the "Transmitted Data" line.

At the terminal end of the system, the computer's data set turns on its carrier; the terminal's data set detects it and sends "Carrier Detected" level to the terminals. Several milliseconds later (length determined by the "Clear To Send" delay in the transmitting data set), the first message bits arrive and are converted to a digital signal, which is passed from the data set to the terminal on the "Received Data" line. In synchronous operation, clocking for the data is generated by the receiving data set and is passed to the terminal on the "Receiver Signal Element Timing: line in order that the terminal can correctly clock the bits into its buffer or memory as they arrive. The clock pulse is timed to occur at the center of the data bit on the "Received Data" line.

RS232 Communications continues in Issue 2 of compute II with expanded definitions and comments on RS232-C signals. RCL

©



compute II needs you!

We're as interested in short applications/programming notes as in full-length feature articles.

An Upgrade for KIM MICROCHESS 1.0

Garold R. Stone
P.O. Box 153
Annapolis Junction, MD. 20701

If you have Peter Jennings' MICROCHESS program for the KIM-1 microcomputer you can teach it to play a significantly better game of chess without adding a single byte of expansion memory. This article describes a "patch" I have written for MICROCHESS which gives the computer a more flexible opening game and two new strategies for the middle and end game. Just load your copy of MICROCHESS, enter my code from the accompanying program listing along with the chess opening sample from table one, and play chess. There are no changes in the way you run the program. (For a description of the MICROCHESS program see KB, August 1978, page 74). For clarity I will use the term MICROCHESS only to refer to the original program as written by Peter Jennings. I will say "patch" to refer to the changes I am describing here.

Off the Shelf

The MICROCHESS I bought from Micro-ware Ltd. opens the game by playing from a pre-selected list of moves for a user chosen chess opening (Roy Lopez, French Defence, etc.). That opening list also contains one anticipated opponent move for each computer move. Things go well as long as the opponent makes the anticipated replies. But a human opponent seldom does that -- at least I don't. As soon as I make a novel move MICROCHESS permanently abandons the opening list. Whenever MICROCHESS is forced to quit the opening list too early, coherent development of pieces stops, the queen usually comes out too early, an ill-prepared attack is launched, and the computer loses its ability to castle (because castling is only possible from the opening list).

Compromises in 1.1K

Mr. Jennings points to these problems in his excellent documentation manual:

> "A major problem in the analysis is that there is only one strategy which is used for the opening, the middle game and the end game. This involves a considerable compromise of three different types of play."

The single strategy used by MICROCHESS is best suited for the middle game, where the capture of pieces dominates. In order to add a dynamic opening strategy which would emphasize the development and positioning of pieces, I had to settle for my own set of compromises, as you'll see. I should point out that Mr. Jennings seems to have surmounted this problem in the other versions of MICROCHESS he has written for microcomputers with more memory, such as the PET, TRS-80, and the APPLE.

The Opening

Table 1 shows my data format for eight opening development moves. Unlike in MICROCHESS, anticipated opponent replies are **not** listed. On each turn the **patched** program evaluates all of the computer's available moves. The available move which comes out with the highest evaluation is compared with the evaluation for the next legal move in my opening list and the higher of the two is selected as the computer's move for that turn. The development move is usually selected because its evaluation is always boosted by a threshold factor. I set the threshold factor high enough so that only moves with a significantly higher evaluation can override the development move. The higher the threshold, the more likely it is that the development move will be selected for that turn. Thus, the computer follows an opening game plan, responds to significant attack threats or capture opportunities, and then continues to carry out the opening game plan on the next turn by consulting the opening list again.

Books on chess openings and opening game strategy can serve as guides in writing new lists of development moves. Choose openings which are general in nature and do not depend on specific moves by the opponent. Specify each development move by giving the piece (variable DEVP), the square of origin (FROM), and the destination (TO), using the same notation as in MICROCHESS (see tables 2 and 3). Openings for white and black will require separate notation. Fill all unused locations in the opening list with the magic number 1F (hexadecimal), which causes those locations to be skipped because they are off the board.

Castling

As in MICROCHESS the computer's castling move must be completed for it by moving its rook after the computer signals castling by moving its King the necessary two squares. My added programming will prevent castling if the computer's King is off its starting square or if it would end up in check. The other rules for castling are not checked, however. If the computer castles illegally, then the move must be refereed. The simplest way is to use the "touch-move" rule -- once a player touches a piece it

Table 1
Opening Move Data

ADDR	VARIABLE	MOVE	WHITE	BLACK	COMMENT
00C3	.FACTOR		05	05	THRESHOLD FACTOR
00C4	.DEVP-1	N-KB3	06	06	PIECE
00C5	.FROM		01	06	ORIGIN
00C6	.TO		22	25	DESTINATION
00C7	.DEVP-2	P-KN3	0A	0A	PIECE
00C8	.FROM		11	16	ORIGIN
00C9	.TO		21	26	DESTINATION
00CA	.DEVP-3	B-KN2	04	04	PIECE
00CB	.FROM		02	05	ORIGIN
00CC	.TO		11	16	DESTINATION
00CD	.DEVP-4	P-K3	0F	0F	PIECE
00CE	.FROM		13	14	ORIGIN
00CF	.TO		23	24	DESTINATION
00D0	.DEVP-5	0-0	00	00	PIECE (KING SIDE CASTLE)
00D1	.FROM		03	04	ORIGIN
00D2	.TO		01	06	DESTINATION
00D3	.DEVP-6	K-QB3	07	07	PIECE
00D4	.FROM		06	01	ORIGIN
00D5	.TO		25	22	DESTINATION
00D6	.DEVP-7	P-Q4	0E	0E	PIECE
00D7	.FROM		14	13	ORIGIN
00D8	.TO		34	33	DESTINATION
00D9	.DEVP-8	(NO	1F	1F	
00DA	.FROM	(MOVE)	1F	1F	
00DB	.TO		1F	1F	

See Tables 2 and 3 for coding of Pieces and Squares

must be moved. Thus, the computer would have to move its King somewhere else, and you would enter that move for it. If there are no legal moves left for the King, then the computer must resign. This situation seldom comes up because I write openings which castle early enough to avoid the risk and annoyance of an illegal attempt.

Program Flow

What follows is a description of how the patched program works. MICROCHESS subroutines which are not defined in my accompanying program listing are in bold letters.

Whenever it is the computer's turn to move, MICROCHESS command loop **CHESS** calls my version of subroutine GO (see 03A2 in the program listing). MICROCHESS uses the value of a variable called STATE to keep track of what it's doing. State 4 guides the generation and evaluation of the computer's available moves. There are other states for generating potential opponent replies, etc. MICROCHESS subroutine **GNMX** (see 03AA) initializes some variables called "counts" for evaluating moves and then generates all moves available to the computer on that turn. **GNMX** calls MICROCHESS subroutine **JANUS** to calculate and evaluate the counts for each trial move. Based on the value in STATE, **JANUS** decides what to do next -- generate potential opponent replies for evaluation, calculate exchanges of pieces, etc. **JANUS** changes the value in STATE as it goes.

Table 2
Microchess Piece Notation and Storage

		MEMORY LOCATION	
CODE	PIECE	COMPUTER	OPPONENT
00	KING	0050	0060
01	QUEEN	0051	0061
02	KING ROOK	0052	0062
03	QUEEN ROOK	0053	0063
04	KING BISHOP	0054	0064
05	QUEEN BISHOP	0055	0065
06	KING KNIGHT	0056	0066
07	QUEEN KNIGHT	0057	0067
08	KR PAWN	0058	0068
09	QR PAWN	0059	0069
0A	KN PAWN	005A	006A
0B	QN PAWN	005B	006B
0C	KB PAWN	005C	006C
0D	QB PAWN	005D	006D
0E	Q PAWN	005E	006E
0F	K PAWN	005F	006F

Table 3
Board Notation

Computer

00	01	02	03	04	05	06	07
10	11	12	13	14	15	16	17
20	21	22	23	24	25	26	27
30	31	32	33	34	35	36	37
40	41	42	43	44	45	46	47
50	51	52	53	54	55	56	57
60	61	62	63	64	65	66	67
70	71	72	73	74	75	76	77

OPPONENT

Note: Whether playing White or Black, the Computer's starting squares are always 00 through 17. Be sure to orient the playing board so that the lower left corner is black. The White Queen should be on a white square and the Black Queen should be on a black square.

Table 4
New Variables Used

ADDR	VARIABLE	COMMENT
00C3	.FACTOR	Threshold factor for opening moves
00DC	.OMOVE	MICROCHESS opening move flag
00DC	.OMOVE	Base for opening move array
00EF	.BKMOB	Number of legal moves for Opponent King
00F0	.BIAS	Receives threshold factor for legal list move

JANUS and portions of **GNMX** call each other recursively, again and again, until all of the computer's available moves have been evaluated in the light of all possible opponent replies. By the time program control returns from that very first call to subroutine **GNMX,** one move has emerged with an evaluation higher than all the others.

Then my patch searches the opening move list from the beginning to find the first piece (variable DEVP) which is still where it is supposed to be (FROM) (see 03B1). The move by this piece to its destination (TO) is checked for legality by a call into the middle of MICROCHESS subroutine **CMOVE.**

If the list move is legal, then the threshold factor (FACTOR) is stored in the variable BIAS for later use (see 03D8). MICROCHESS subroutine **JANUS** is called to do the counts for this list move and for the opponent's potential replies.

To evaluate these counts **JANUS** calls up my version of subroutine STRATEGY (see 1780-17C1). This is where the evaluation of the list move is boosted by adding the threshold factor which was stored earlier in the variable BIAS. Actually, this same subroutine STRATEGY is used by **JANUS** to evaluate any trial move but BIAS is always zero except for legal list moves. If the selected list move is not legal, then **JANUS** is not called to evaluate it, and no more list moves will be tried for that turn. This ensures that moves from the opening list are made in the order you wrote them. After the last list move has actually been moved, the variable OMOVE is set to zero and the opening list is ignored for the rest of the game (see 03AF).

As you exit subroutine STRATEGY you enter that portion of MICROCHESS which compares the evaluation of the current trial move with that of the best move so far, saving the better of the two as the new best move so far. This is also where MICROCHESS tests for check or checkmate before returning to **JANUS.** Control then passes to the MICROCHESS subroutine which takes the best trial move and actually moves it (see 03E3). The computer's move is flashed on the KIM display and the program returns to the MICROCHESS command loop, ready for the opponent to enter his move.

Middle and End Game

MICROCHESS sees only one and a half moves ahead. With this limited horizon it has trouble finding and closing in on the opposing King. To compensate for this I give a bonus of two points for moves inside a zone which surrounds the opposing King and moves along with it. The computer's Pawns and King do not get the bonus (see 179D).

Another strategy encourages moves which hem in the opposing King, in preparation for checkmate. The value of any trial move is decreased by the number of safe moves it leaves for the opposing King. This is the same as adding a point for each square denied to the opposing King. Since MICROCHESS calls subroutine **JANUS** to evaluate only legal moves, it was easy enough to put a subroutine call inside **JANUS** which would increment a mobility count (BKMOB) for each legal move found for the opponent King when the computer is checking for opponent reply moves during state zero (see 0112, 17D9,179A).

Both strategies come into play only after the opening list has been emptied, so as not to interfere with the development of pieces during the opening game (see 1796).

Evaluation

I approached move evaluation in much the same way as in MICROCHESS -- adding and subtracting weighted counts representing captures, position, and mobility for both sides. I did not use some of the counts generated by MICROCHESS and I created the new ones I described above. Given the severe memory restrictions, my goal was an evaluation formula which emphasizes immediate and tangible factors, such as position and the values of pieces captureable during the current turn. Less immediate factors, such as overall attack strengths, are given fractional weighting. These become influential only after more significant factors have cancelled each other out.

For now I've had to be satisfied with just breaking MICROCHESS of its habit of throwing away its pieces by occasionally making bad decisions about captures where pieces are exchanged. In my patch any piece the computer wants to capture must be greater than or equal to the most valuable piece the computer would lose by making that move (variable BMAXC). Only trial moves which pass this admittedly simplistic test are given an extra 20 hex points (see 17B1). There is more that could be done, like making better use of the MICROCHESS counts for exchanges involving up to three captures per side.

I hope I've made my point. All you need is a shoe horn and you can slip just about any changes you want into the 1.1K KIM MICROCHESS. You may pinch a few toes in the process, but the result is a KIM that plays better chess. By trying to "upgrade" MICROCHESS I really learned to appreciate what an excellent piece of work it is.

MICROCHESS is available on KIM cassette with documentation manual from Micro-Ware Ltd., 496 Albert St., Suite 7, Waterloo, Ontario, Canada, N2L 3V4

Abbreviated Instructions for Loading and Running MICROCHESS 1.0 UPGRADE

Load:

Enter (RS) to reset KIM
Enter (AD) 00F1 (DA) 00 to reset decimal flag
Enter (AD) 17F9 (DA) C1 to enter tape ID for program segment
Enter (AD) 1873 (GO) to start read routine of KIM
Press "Play" on cassette player
STOP recorder when display shows: 0000
Enter (RS) (AD) 1873 (GO) to read second program segment (same label "C1")
STOP recorder when display shows: 0000
Enter (RS) (GO) to start program execution

Playing:

Enter (C) on KIM hexpad keyboard to reset program for new game
Enter (PC) (for "play chess") because KIM plays first
After KIM gives its move, enter your move as FROM-TO according to the board notations in table 3 of the article. Keep typing until your move shows correctly, then enter (F) (PC).

```
                   Ø11Ø              .BA $3A2
Ø3A2- A2 Ø4        Ø12Ø GO           LDX #$Ø4      ; RESET BEST EVALUATION
Ø3A4- 86 FA        Ø13Ø              STX *BESTV    ;   SO FAR
Ø3A6- 86 B5        Ø14Ø              STX *STATE    ; STATE = 4; TRAIL MOVES
Ø3A8- A2 12        Ø15Ø              LDX #$12      ; ZERO COUNTERS & BIAS
Ø3AA- 2Ø Ø2 Ø2     Ø16Ø              JSR GNMX      ; GENERATE TRAIL MOVES
Ø3AD- A4 DC        Ø17Ø              LDY *OMOVE    ; OPENING LIST DONE?
Ø3AF- 1Ø 32        Ø18Ø              BPL NODEVP    ;   - YES, MID-GAME
Ø3B1- AØ E6        Ø19Ø              LDY #$E6      ;   - NO, NEXT DEVP
Ø3B3- C8           Ø2ØØ NEXT         INY
Ø3B4- C8           Ø21Ø              INY  ;          INDEX OF DEVP
Ø3B5- 84 DC        Ø22Ø              STY *OMOVE    ; OPENING LIST EMPTY?
Ø3B7- 1Ø 2A        Ø23Ø              BPL NODEVP    ;   - YES, MID-GAME
Ø3B9- B6 DC        Ø24Ø              LDX *DEVP,Y   ;   -NO, NEXT DEVP
Ø3BB- 86 BØ        Ø25Ø              STX *PIECE
Ø3BD- B5 5Ø        Ø26Ø              LDA *BOARD,X       ; DEVP LOCATION
Ø3BF- C8           Ø27Ø              INY  ;          INDEX OF  FROM
Ø3CØ- 48           Ø28Ø              PHA  ;          (SAVE DEVP LOCATION)
Ø3C1- 98           Ø29Ø              TYA  ;          TRANSFER INDEX OF
Ø3C2- AA           Ø3ØØ              TAX  ;            FROM INTO X
Ø3C3- 68           Ø31Ø              PLA  ;          DEVP LOCATION IN ACCUM
Ø3C4- D5 DC        Ø32Ø              CMP *FROM,X   ; DEVP AT ORIGIN?
Ø3C6- DØ EB        Ø33Ø              BNE NEXT      ;   - NO, GET NEW DEVP
Ø3C8- E8           Ø34Ø              INX  ;          INDEX OF  TO
Ø3C9- B5 DC        Ø35Ø              LDA *TO,X     ; CHECK LEGALLITY OF DEVP
Ø3CB- 2Ø D1 Ø2     Ø36Ø              JSR CMOVE     ; FROM .FROM TO .TO
Ø3CE- 3Ø 13        Ø37Ø              BMI NODEVP    ; NEQ = ILLEGAL MOVE
Ø3DØ- A6 BØ        Ø38Ø              LDX *PIECE    ;   - LEGAL MOVE
Ø3D2- EØ Ø8        Ø39Ø              CPX #$Ø8      ; IS PIECE A PAWN
Ø3D4- 3Ø Ø2        Ø4ØØ              BMI LEGAL     ;   NEG = NOT PAWN
Ø3D6- 7Ø ØB        Ø41Ø              BVS NODEVP    ; SET = ILLEGAL PAWN CAPTURE
Ø3D8- A6 C3        Ø42Ø LEGAL        LDX *FACTOR   ; LEGAL OPENING MOVE!!
Ø3DA- 86 FØ        Ø43Ø              STX *BIAS     ; SET BIAS TO FACTOR
Ø3DC- A2 Ø4        Ø44Ø              LDX #$Ø4      ; EVALUATE OPENING MOVE
Ø3DE- 86 B5        Ø45Ø              STX *STATE    ;   AND PUT IT IN BESTV
Ø3EØ- 2Ø ØØ Ø1     Ø46Ø              JSR JANUS     ;   IF ITS THE BEST MOVE
Ø3E3- A6 FA        Ø47Ø NODEVP       LDX *BESTV    ;   SO FAR
Ø3E5- EØ ØF        Ø48Ø              CPX #$ØF      ; RESIGN OR STALEMATE IF
Ø3E7- 4C C2 17     Ø49Ø              JMP CONT      ;   BESTV TOO LOW
                   Ø5ØØ ;
                   Ø51Ø              .BA $17C2
17C2- 9Ø 12        Ø52Ø CONT         BCC MATE      ; (ORIGINAL MICROCHESS
17C4- A6 FB        Ø53Ø MV2          LDX *BESTP    ;   CODING)
17C6- B5 5Ø        Ø54Ø              LDA *BOARD,X       ; MOVE AND DISPLAY THE
17C8- 85 FA        Ø55Ø              STA *BESTV    ;   BEST MOVE
17CA- 86 BØ        Ø56Ø              STX *PIECE
17CC- A5 F9        Ø57Ø              LDA *BESTM
17CE- 85 B1        Ø58Ø              STA *SQUARE
17DØ- 2Ø 4B Ø3     Ø59Ø              JSR MOVE
17D3- 4C ØØ ØØ     Ø6ØØ              JMP CHESS     ; END COMPUTER'S TURN
17D6- A9 FF        Ø61Ø MATE         LDA #$FF      ; RESIGN OR
17D8- 6Ø           Ø62Ø              RTS
                   Ø63Ø ;
                   Ø64Ø              .BA $178Ø
178Ø- A9 8Ø        Ø65Ø STRATEGY     LDA #$8Ø      ; EVALUATION = 8Ø + OR - SCORE
1782- 18           Ø66Ø              CLC
1783- 65 EB        Ø67Ø              ADC *WMOB     ; COMPUTERS'S MOBILITY
1785- 4A           Ø68Ø              LSR A
1786- 18           Ø69Ø              CLC
1787- 69 4Ø        Ø7ØØ              ADC #$4Ø      ; RESET EVAL TO 8Ø +OR- SCORE
1789- 65 ED        Ø71Ø              ADC *WCC      ; COMPUTER'S ATTACK STRENGTH
```

```
178B- 38        Ø72Ø           SEC
178C- E5 E5     Ø73Ø           SBC *BCC      ; OPPONENT'S ATTACK STRENGTH
178E- 4A        Ø74Ø           LSR A
178F- 4A        Ø75Ø           LSR A         ; MOBILITY X 1/16
179Ø- 4A        Ø76Ø           LSR A         ; ATTACK STRENGTH X 1/8
1791- 18        Ø77Ø           CLC
1792- 69 7Ø     Ø78Ø           ADC #$7Ø      ; RESET EVAL TO 8Ø +OR- SCORE
1794- 65 FØ     Ø79Ø           ADC *BIAS     ; ZERO UNLESS DEVP MOVE
1796- A4 DC     Ø8ØØ           LDY *OMOVE    ; NEGATIVE IF STILL DEVP
1798- 3Ø 17     Ø81Ø           BMI CAPTEST   ; MID-GAME IF POSITIVE
179A- 38        Ø82Ø           SEC   ;         DEDUCT MOBILITY OF THE
179B- E5 EF     Ø83Ø           SBC *BKMOB    ;   OPPONENT'S KING
179D- A6 BØ     Ø84Ø           LDX *PIECE    ; BONUS FOR MOVE INTO
179F- CA        Ø85Ø           DEX   ;         OPPONENT'S KING ZONE
17AØ- EØ Ø7     Ø86Ø           CPX #$Ø7      ; NOT FOR COMPUTER'S KING
17A2- BØ ØD     Ø87Ø           BCS CAPTEST   ;   OR PAWNS
17A4- 48        Ø88Ø           PHA   ;        (SAVE EVALUATION)
17A5- A5 6Ø     Ø89Ø           LDA *BK       ; LOCATION OF OPPONENT'S KING
17A7- 38        Ø9ØØ           SEC
17A8- E9 38     Ø91Ø           SBC #$38      ; CALCULATE KING ZONE
17AA- C5 B1     Ø92Ø           CMP *SQUARE   ; MOVE INTO ZONE?
17AC- 68        Ø93Ø           PLA   ;        (RESTORE EVALUATION)
17AD- BØ Ø2     Ø94Ø           BCS CAPTEST   ; CARRY CLEAR IS IN ZONE
17AF- 69 Ø2     Ø95Ø           ADC #$Ø2      ;   ADD BONUS, NEAR KING
17B1- A6 DD     Ø96Ø CAPTEST   LDX *WCAPØ    ; IF COMPUTER'S CAPTURE
17B3- E4 E4     Ø97Ø           CPX *BMAXC    ;   IS NOT GREATER THAN
17B5- 9Ø Ø3     Ø98Ø           BCC QUIT      ;   OR EQUAL OPP, QUIT
17B7- 18        Ø99Ø MOVEOK    CLC   ;         PASSES CAPTURE TEST
17B8- 69 2Ø     1ØØØ           ADC #$2Ø      ; POINTS FOR GOOD MOVE
17BA- 65 DD     1Ø1Ø QUIT      ADC *WCAPØ    ; POINTS FOR CAPTURE
17BC- 38        1Ø2Ø           SEC   ;         POINTS FOR OPPONENT'S
17BD- E5 E4     1Ø3Ø           SBC *BMAXC    ;   MAX CAPTURE IN REPLY
17BF- 4C 77 Ø3  1Ø4Ø           JMP CKMATE    ; TEST FOR CHECKMATE
                1Ø5Ø ;
                1Ø6Ø           .BA $17D9
17D9- DØ Ø6     1Ø7Ø BKMOVE    BNE OUTBK     ; RTS IF STATE NOT ZERO
17DB- C9 ØØ     1Ø8Ø           CMP #$ØØ      ; RTS IF NOT OPP KING'S
17DD- DØ Ø2     1Ø9Ø           BNE OUTBK     ;   MOVE
17DF- E6 EF     11ØØ           INC *BKMOB    ; COUNT LEGAL OPP KING
17E1- 6Ø        111Ø OUTBK     RTS   ;           MOVES
                112Ø ;
                113Ø           .BA $Ø112
Ø112- EØ ØØ     114Ø           CPX #$ØØ      ;COUNT LEGAL REPLY MOVES
Ø114- 2Ø D9 17  115Ø           JSR BKMOVE    ;   FOR OPPONENT'S KING
Ø117- EA        116Ø           NOP
                117Ø ;
                118Ø           .BA $2ØØ
Ø2ØØ- A2 11     119Ø           LDX #$11      ; CLEAR COUNTERS, NOT BIAS     ©
                12ØØ           .EN
```

FREE! Up to $170 in merchandise with purchase of one of following PET-CBM items ! ! !

BETSI PET to S-100 Interface	$ 119.00
PET Connectors-Parallel or IEEE	$ 1.95
Cassette Port	$ 1.45
Hands on Basic with a PET	$ 10.15
Programming the 6502 (Zaks)	$ 10.45
6502 Applications Book (Zaks)	$ 10.45
6500 Manuals (MOS Technology)	$ 6.50
Programming a Microcomputer: 6502	$ 7.75
6502 Assembly Language (Osborne) NEW!	$ 9.90

Axiom EX-801 PET Printer (with graphics)	$ 475.00
Axiom EX-820 PET Plotter	$ 749.00
Anderson Jacobson 841 Selectric	$1015.00
Leedex Video 100 12" Monitor	$ 119.00
Heath WH19 Terminal (factory asm.)	$ 770.00
Heath WH14 Printer (factory asm.)	$ 735.00
IEEE-RS 232 Printer Adaptor for PET	$ 88.00

PET SPECIALS

		*FREE
PET 16N 16K full size graphics keyboard	$ 995	**$130**
PET 16B 16K full size business keyboard	$ 995	**$130**
PET 32N 32K full size graphics keyboard	$1295	**$170**
PET 32B 32K full size business keyboard	$1295	**$170**
PET 8N 8K full size graphics keyboard	$ 795	**$100**
PET 2040 DUAL DISK DRIVE-343,000 bytes	$1295	**$170**
PET 2022 Tractor Feed Printer	$ 795	**$100**
PET 2023 Pressure Feed Printer	$ 695	**$ 70**
PET C2N External Cassette Deck	$ 95	**$ 12**
Used 8K PETs (limited quantities)	$ 495	

★ **EDUCATIONAL DISCOUNTS** ★
Buy 2 PET Computers, get 1 FREE

*Free Merchandise with Purchase of PET-CBM Item.

KIM-1 $159 (add $30 for power supply) SYM-1	$209
BAS-1 Microsoft ROM Basic for SYM	$ 85
Memory Plus (KIM, SYM, AIM)	$195
SYM Assembler in ROM	$ 85
KIM 16K Static RAM	$284
Seawell Motherboard – 4K RAM Space	$139
KTM-2/80 Synertek Video Board	$349
S-100 16K Static RAM Kit **SALE**	$198
TIS PET Workbooks – set of 6	$ 21.50
Dust Cover for PET	$ 7.45

All Book and Software Prices are Discounted

PET Personal Computer Guide (Osborne)	$ 12.75
PET and the IEEE-488 Bus (Osborne)	$ 12.75
6502 Software Cookbook (Scelbi)	$ 9.45

Cassettes (all tapes guaranteed) AGFA PE611 Premium quality, high out put lownoise in 5 screw housing with labels:

C-10	10/5.65	50/25.00	100/48.00
C-30	10/6.90	50/30.00	100/57.00

2716 EPROM (5 volt)	$29.00	5@ 27.00	10@ 25.00
4116 RAM 16K x 1		$7.50	8 @ 7.10
2114 L 450	$5.35	20 @ 4.95	80 @ 4.45
2114 L 300	$6.10	20 @ 5.65	80 @ 5.05
6502	$8.35	10 @ 7.85	40 @ 7.35
6522	$7.15	10 @ 6.95	40 @ 6.45
6520	$5.15	10 @ 4.90	40 @ 4.45
6532	$7.90	10 @ 7.40	40 @ 6.85
6502A			$9.50
6504			$7.95
21L02			$ .90
6550 RAM (PET 8K)			$12.70
S-100 Wire Wrap		$2.85	10 @ 2.65
S-100 Solder Tail		$2.35	10 @ 2.15

ATARI – INTRODUCTORY SPECIAL

ATARI 400, Atari 800 and all Atari Modules 20% OFF.

Programmers Toolkit – PET ROM Utilities	$ 44.90
Sargon II (TRS-80 or Apple) NEW!	$ 24.90
Microchess for PET (Peter Jennings)	$ 17.90
Visible Music Monitor (4 Voice) for PET	$ 29.90
KL-4M 4 Voice Music Board (PET)	$ 34.90
CmC Word Processor (8K $25) (16K $34)	
Adventures by Scott Adams	15% Off

SALE

(write for quantity prices)

Scotch 8" Disks	10/$31.00
Scotch 5¼" Disks	10/$31.50
Verbatim 5¼" Disks	10/$26.50
Diskette Storage Pages	10/$ 3.95
Disk Library Cases 8" - $2.95	5" - $2.15
BASF 5¼" Disks	10/$28.00
BASF 8" Disks	10/$29.00

MINIMAX by COMPU/THINK

MINIMAX I (.8 Megabyte Disks)	$3395
MINIMAX II (2.4 Megabyte Disks)	$4495

SPECIAL—MINIMAX prices include Compu/Think PAGEMATE Database and Report Writer at no charge.

115 E. Stump Road
Montgomeryville, PA 18936

(215) 699-8386
699-5826

A B Computers

WRITE FOR COMPLETE CATALOG
Add $1.00 per order for shipping.
We pay balance of UPS surface charges on all prepaid orders.

FLOPPY DISK FOR PET, KIM,SYM,AIM - S100 EXPANSION

ALL IN ONE

PEDISK!

- ADD I/O,VOICE,MEMORY
- LOAD,SAVE INSTANTLY
- PLUG-IN HOOK UP
- UP TO FOUR DISK DRIVES

The **PEDISK SYSTEM** from CGRS Microtech provides both a high speed FLOPPY DISK and an S100 EXPANSION chassis all in one! The FLOPPY DISK provides the PET computer with means to instantly load and save programs and information. Tired of waiting for your cassette? The optional S100 EXPANSION chassis gives the power and versetility of the S100 bus to the PET. The extensive and diverse product offerings on the S100 bus include Memory, I/0, Telephone Interface Cards, IBM Selectric Interface, Analog I/0, Voice Generators and even Voice Input Cards.

The **PEDISK SYSTEM** offers an economical solution to the problem of expanding the PET, It offers complete expansion capability in a single package. The FLOPPY DISK uses the IBM 3740 format to promote compatibility with other systems (Radio Shack). The S100 EXPANSION uses the most widely supported microcomputer bus in the industry. Standard, Economical Expansion – The **PEDISK SYSTEM**

The KM³ Software is furnished with each package and is available with the EXS100 Board. This software allows the user to LOAD, SAVE, RUN, and UPDATE files on the disk. In addition, Rev. 2 software will allow the user to OPEN, CLOSE, INPUT, and PRINT a disk file. The commands will execute directly or from a program.

COMPLETE SYSTEMS.......

PACKAGE 1	EXS100 DISK CONTROLLER MINIFLOPPY DISK DRIVE CABINET & DISK POWER SUPPLY	$799.95
PACKAGE 2	EXS100 COMBINATION DISK/S100 BOARD MINIFLOPPY DISK DRIVE CABINET & S100/DISK POWER SUPPLY 3 SLOT S100 CHASSIS	$895.00

PARTS.......

EXS100 BOARD

Assembled as an S100 Adaptor only	$199.95
Assembled as an Disk Controller only	$299.95
Combination Disk/S100 Board	$349.95

MICROTECH
P.O. Box 368
SOUTHAMPTON, PA. 18966
(215) 757-0284

*PET is a trademark of Commodore.

PROGRAM TRANSFERS (PET TO KIM)

Joseph A. Dilts
Assoc. Prof. of Chemistry
Univ. of North Carolina
Harvey B. Herman
Prof. of Chemistry
Univ. of North Carolina

In a recent 6502 User Notes editorial (Issue #14, P. 27) you propose a method to transfer BASIC programs from PET to KIM. Your editorial prompted us to document our method for your readers, as we have been doing this for almost a year now.

There are several possible methods to communicate between computers. One way, memory to memory transfer using parallel ports, has already been published ("KIM-1 Talks to PET", PET User Notes, Vol. 1 #5, p 6). However, as you point out, tokens for various Micros of BASICs differ. Consequently, a simple transfer will not work for BASIC programs.

Another way, the way we picked, is to send an ASCII version of the program from the IEEE port of the PET to the serial interface of the KIM. This method requires both an IEEE/RS232 serial adapter and a RS232/KIM adapter. We used an IEEE adapter manufactured by Connecticut Microcomputer (PET ADA 1200 $170 assembled) which one of us (J.A.D.) uses with a word processing program (CMC). We constructed a modified version of the RS232/KIM adapter published in your journal (Issue #4, p 6) to complete the connection from PET to KIM.

It is very important to set the slide switches on the PET ADA 1200 properly. One combination that works for more than one application is:

1 parity bit
2 stop bits (10010 in switches 1-5)
7 characters even parity

The baud rate of the KIM serial line can be set with a serial terminal and the reset/rubout sequence or with the KIM keyboard (change locations $17F2/$17F3, see 6502 User Notes. Issue #6, p 11 for typical values.) We worked with a 300 baud terminal and the IEE/RS232 interface set at 300 baud.

The first method we tried was similar to the suggestion in your editorial. KIM BASIC was brought up as always using the terminal. A NEW command was given, the terminal was carefully disconnected from KIM and the PET/adapter substituted (pin 2 on the KIM to pin 3 on the PET/adapter). A PET program which we wished to transfer to KIM was loaded and listed to the IEEE bus using the following sequence (on the PET).

```
OPEN 6, 6
CMD 6
LIST
PRINT #6
CLOSE 6
```

The terminal was reconnected and the program listed on KIM. To our horror we found this method generally only transferred the **first** program line as apparently a delay or hand shaking is necessary after the first carriage return. One of us (HBH) uses an X-off/X-on sequence to properly load paper tapes on KIM thereby circumventing the same problem. We could not use this method here and we did not see an easy way to modify the PET ADA 1200 adapter for handshaking.

We tried another approach; one that eventually worked well. Since we saw the problem as necessitating a delay after carriage return, perhaps software, rather than hardware, could accomplish this. A BASIC program was written which could read an ASCII file from tape and send each character to the IEEE bus individually. The program adds a proper delay after the carriage return character is sent and gives KIM BASIC enough time to digest each line.

It is easy to make any PET BASIC program into a file on cassette tape after it is loaded.

```
OPEN 1, 1, 1
CMD 1
LIST
PRINT #1
CLOSE 1
```

The PET program, GET, (shown in the figure) which reads the file is loaded next. (Line 45 is incomplete, see below for reason, and needs GOSUB 60 after the colon). The KIM terminal is disconnected and PET connected to KIM. GET is executed on the PET and causes the program file, made previously, to be sent character for character over the IEEE bus to KIM, inserting delays when necessary. KIM's terminal is reconnected and if the PET program happens to be directly compatable with KIM's BASIC, it can be run immediately without modification.

Some minor problems may arise:

1. KIM's BASIC has a line buffer limitation of 72 characters. Lines longer than 71 characters are truncated by our program to fit this line length (c.f. line 45 above) and have to be reformulated after transfer.
2. KIM uses "   " and "@" as special characters for character and line delete. If the PET

program uses these characters it may be advantageous to temporarily change locations $2440 and $243C (in our version of 9 digit KIM BASIC) to other ASCII characters to avoid trouble.

3. KIM and PET do not have exactly the same BASIC language. Neither do they have the same hardware. Some commands in PET BASIC will have to be translated to properly operate on KIM. Locations specific to PET will have to be relocated to be compatable with KIM. It may not be practical to transfer all programs. Users will have to use their judgement.

Even though we recognized that problems exist we have successively moved very long PET programs (close to 8K) and made them operate on KIM. As a short, possibly bad, example we transferred the GET program itself to KIM and listed it on a KIM operated teletype (110 baud). This program cannot execute on KIM because of software and hardware differences. It also illustrates the line length buffer limitation (line 45). As long as users are aware of the potential trouble spots they should have no difficulty.

We have also used the PET ADA 1200 adapter in other applications. The adapter was originally designed to directly drive a printer off the IEEE bus. When the terminal is not in proximity to the PET but is near a telephone and modem it is possible to make listings over phone lines. We have used the Pennywhistle 103 Modem to transmit PET programs and output on a high channel frequency, to terminals whose modem is receiving on the same frequency. A simple list to the IEEE bus (with CMD) or data output of the bus is all that's required. If the terminal also requires an extra delay after carriage return more bits per character (9) could suffice or the GET program used. The remote device could also be a computer. However, we have not tried this. The best part of the remote printing operation is to watch the faces of the folks in the computer room when they spy a strange (to them) dialect of BASIC printing on their terminal.

GET PROGRAM

```
10 OPEN1,1,0
15 OPEN6,6
17 FORI=1to500:NEXTI
18 PRINT#6,CHR$(15)
20 GET #1,C$
30 IF (ST) AND 64 THEN 500
40 IFASC(C$)=13THEN PRINT L$:PRINT#6,L$:L$="":FOR
   I=1TO500:NEXTI:GOTO20
45 L$=L$+C$:IFLEN(L$)=71THENPRINTL$:PRINT#6,L$:L$=
   "":FORI=1TO500:NEXT:
50 GOTO20
60 GET#1,C$:IF(ST)AND64THEN500
70 IFASC(C$)=13THENRETURN
80 GOTO60
500 CLOSE1
OK
```

©

Part 1: Implementing the IEEE-488 Bus on a SYM-1

DESIGNING AN IEEE-488 RECEIVER WITH THE SYM

Larry Isaacs, COMPUTE. Staff

This article is the first in a series on the use of a single board computer as a dedicated interface. In this section I will describe the design of an interface connecting a Spinwriter to the PET IEEE-488 Port using a SYM-1. If you have a need for an interface or controller, but not much experience using single board computers, this series should provide some pointers on how to go about implementing one.

In this article, the discussion of the IEEE-488 Bus will be limited to that which is relevant to the PET, and to how the PET sends data to a printer. Also, when the software to be presented is too general to give the actual assembly language, it will be given in PASCAL. The listings should be readable, even if you haven't had much exposure to PASCAL. The names used in the PASCAL listings will correspond to the names used in the assembly language listings. The following notes should help if you haven't seen PASCAL before.

1. PASCAL uses ": = " for the assignment operator; " = " is used only for comparisons.
2. The ";" is used to separate statements.
3. When statements are enclosed between a "begin" and an "end;", it means that that block of statements may be treated as if they were one statement. The programs are indented to help show which "end;"'s go with which "begin"'s.

DIVIDE AND CONQUER: A STARTING POINT

One of the most effective ways to handle design problems is to successively divide the required functions into small sets of sub-functions. Once the complexity of a sub-function has been reduced to a manageable level, then it is implemented.

The first division of the PET-to-Spinwriter interface is shown in Listing 1.

Listing 1.

```
program PETTOSPINWRITERINTERFACE;

procedure INIT; begin...end; {initialization}
procedure PRINT; begin...end; {send chr. to Spinwriter}
procedure CYCLE; begin...end; {get byte from IEEE}
procedure INTERFACE; begin...end; {main interface software}
begin {PET to Spinwriter Interface}
   INIT;
   INTERFACE
end.
```

Here, the interface task has been divided into four sub-functions. The task of the INIT procedure will become apparent as the other parts of the software are written. The INTERFACE procedure will contain the intelligence of the interface. The exact function of INTERFACE can't be determined yet, so this sub-function will be dealt with later.

The PRINT and CYCLE sub-functions will be used by INTERFACE to communicate with the PET and the Spinwriter. Unlike the others, the functions of PRINT and CYCLE are sufficiently narrow in scope to be implimented at this point. Both will involve dealing with hardware as well as software. But once done, most of the hardware details will be taken care of.

PRINT

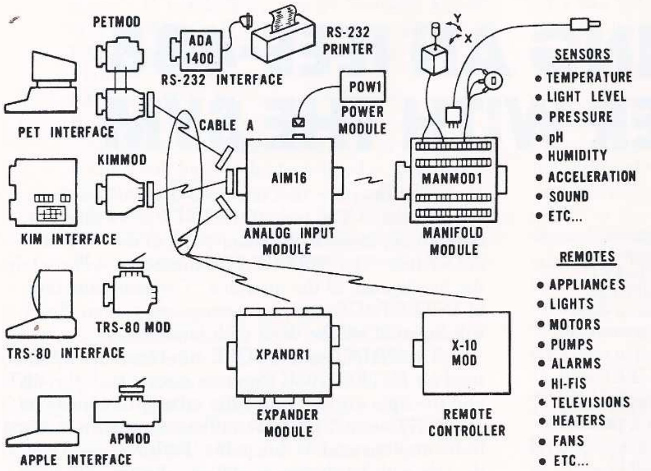
The purpose of this routine is to handle all of the requirements for communicating with the Spinwriter. To do this, one must first consult the Spinwriter and SYM-1 documentation:

Our Spinwriter has a serial interface. This means we can use the serial interface software provided in the SYM-1 Monitor to send characters to the Spinwriter. The Spinwriter Product Description manual reveals that CARRIER DETECT (pin 8 on the RS232 connector), DATA SET READY (pin 6), and CLEAR TO SEND (pin 5) must be high (between +3 and +12 volts) for the Spinwriter to operate. This was simple to take care of since the SYM-1 provides this voltage at the corresponding locations of the T connector.

The Product Description manual also reveals a way of increasing throughput by using the ETX/ACK protocol. This makes use of the 256 character receive buffer found in the Spinwriter. You use this protocol by sending data blocks of up to 254 characters followed by an ETX character (control C). When the Spinwriter withdraws the ETX character from the receive buffer, it transmits an ACK character (control F) to indicate the buffer is empty and ready for another block of characters. This will allow the SYM to transmit at 1200 baud, and let the Spinwriter print at its maximum speed. All of this leads to Figure 1 which shows how to attach the required RS232 connector to the SYM.

After the proper initialization, the OUTCHR subroutine in the SYM Monitor can be used to send characters to the Spinwriter, and the INCHR

MICROCOMPUTER MEASUREMENT and



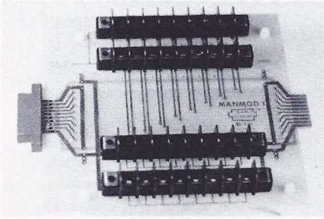
The world we live in is full of variables we want to measure. These include weight, temperature, pressure, humidity, speed and fluid level. These variables are continuous and their values may be represented by a voltage. This voltage is the analog of the physical variable. A device which converts a physical, mechanical or chemical quantity to a voltage is called a sensor.

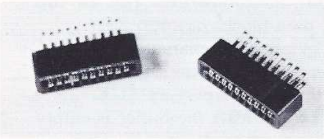
Computers do not understand voltages: They understand bits. Bits are digital signals. A device which converts voltages to bits is an analog-to-digital converter.

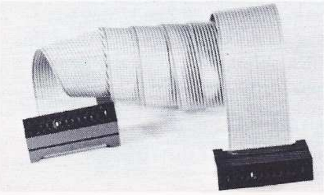
Our AIM 16 (Analog Input Module) is a 16 input analog-to-digital converter.

The goal of Connecticut microComputer in designing the uMAC SYSTEMS is to produce easy to use, low cost data acquisition and control modules for small computers. These acquisition and control modules will include digital input sensing (e.g. switches), analog input sensing (e.g. temperature, humidity), digital output control (e.g. lamps, motors, alarms), and analog output control (e.g. X-Y plotters, or oscilloscopes).

Connectors







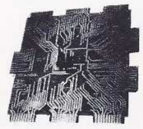
The AIM 16 requires connections to its input port (analog inputs) and its output port (computer interface). The ICON (Input CONnector) is a 20 pin, solder eyelet, edge connector for connecting inputs to each of the AIM16's 16 channels. The OCON (Output CONnector) is a 20 pin, solder eyelet edge connector for connecting the computer's input and output ports to the AIM16.

The MANMOD1 (MANifold MODule) replaces the ICON. It has screw terminals and barrier strips for all 16 inputs for connecting pots, joysticks, voltage sources, etc.

CABLE A24 (24 inch interconnect cable) has an interface connector on one end and an OCON equivalent on the other. This cable provides connections between the uMACSYSTEMS computer interfaces and the AIM 16 or XPANDR1 and between the XPANDR1 and up to eight AIM 16s.

XPANDR1

The XPANDR1 allows up to eight Input/Output modules to be connected to a computer at one time. The XPANDR1 is connected to the computer in place of the AIM16. Up to eight AIM16 modules are then connected to each of the eight ports provided using a CABLE A24 for each module. Power for the XPANDR1 is derived from the AIM16 connected to the first port.



Analog Input Module

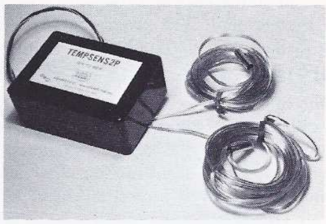
The AIM 16 is a 16 channel analog to digital converter designed to work with most microcomputers. The AIM16 is connected to the host computer through the computer's 8 bit input port and 8 bit output port, or through one of the uMAC SYSTEMS special interfaces.

The input voltage range is 0 to 5.12 volts. The input voltage is converted to a count between 0 and 255 (00 and FF hex). Resolution is 20 millivolts per count. Accuracy is 0.5% ± 1 bit. Conversion time is less than 100 microseconds per channel. All 16 channels can be scanned in less than 1.5 milliseconds.

Power requirements are 12 volts DC at 60 ma.

The POW1 is the power module for the AIM16. One POW1 supplies enough power for one AIM16, one MANMOD1, sixteen sensors, one XPANDR1 and one computer interface. The POW1 comes in an American version (POW1a) for 110 VAC and in a European version (POW1e) for 230 VAC.

TEMPSENS



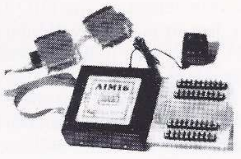
This module provides two temperature probes for use by the AIM16. This module should be used with the MANMOD1 for ease of hookup. The MANMOD1 will support up to 16 probes (eight TEMPSENS modules).

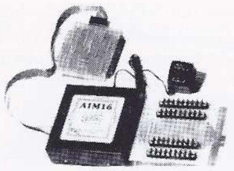
Resolution for each probe is 1°F.

CONTROL for PET, Apple, KIM, and AIM

μMAC SYSTEMS

Computer Interfaces and Sets





For your convenience the AIM16 comes as part of a number of sets. The minimum configuration for a usable system is the AIM16, one POW1, one ICON and one OCON. The AIM16 Starter Set 2 includes a MANMOD1 in place of the ICON. Both of these sets require that you have a hardware knowledge of your computer and of computer interfacing.

For simple plug compatible systems we also offer computer interfaces and sets for several home computers.

INTRODUCING
SUPER X-10 MODULE

Open a door or window and turn on a light, tape recorder, alarm!

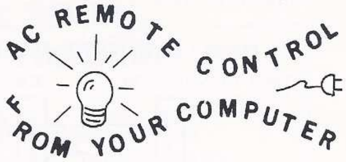
Control lab equipment. CLOSE THE LOOP on the real world.

AN INEXPENSIVE CONTROL SOLUTION FOR

HOME SECURITY
ENERGY CONSERVATION
GREENHOUSES
ENVIRONMENTAL CONTROL
INDUSTRIAL CONTROL
LABORATORIES

SUPER X-10 MOD SPECS

1. Remote controller
   Controls up to 256 different remote devices by sending signals over the house wiring to remote modules. Uses BSR remote modules available all over the USA (Sears, Radio Shack, etc.). Does not require BSR control module. Does not use sonic link.
2. Clock/calendar
   Time of day - hours, minutes, seconds
   Date - month, day - automatically corrects for 28,29,30 and 31 day months.
   Day of the week.
3. Digital inputs/outputs
   8 inputs - TTL levels or switch closures. Can be used as triggers for stored sequences.
   8 outputs - TTL levels
4. Computer interfaces
   S-100: Requires one 8-bit input port and one 8-bit output port.
   Requires cable assembly.
   PET, APPLE, TRS-80, KIM, SYM, AIM65: Plug-in sets available - no cable assembly required.
   Other: same as S-100
5. Self-contained module in metal case with its own power supply. Physical size approximately 5X6X2.



PLUS:
CLOCK, CALENDAR,
REMOTE SEQUENCE TRIGGERS

Price (until April 30, 1980): $199.00 (S-100), $249.00 (other)

All prices and specifications subject to change without notice. Our 30-day money back guarantee applies.

AIM16 (16 channel-8 bit Analog Input Module)	179.00
POW1a (POWer module-110 VAC)	14.95
POW1e (POWer module-230 VAC)	24.95
ICON (Input CONnector)	9.95
OCON (Output CONnector)	9.95
MANMOD1 (MANifold MODule)	59.95
CABLE A24 (24 inch interconnect cable)	19.95
XPANDR1 (allows up to 8 Input or Output modules to be connected to a computer at one time)	59.95
TEMPSENS2P1 (two temperature probes, -10°F to 160°F)	49.95
LIGHTSENS1P1 (light level probe)	59.95

The following sets include one AIM16, one POW1, one OCON and one ICON.

AIM16 Starter Set 1a (110 VAC)	189.00
AIM16 Starter Set 1e (230 VAC)	199.00

The following sets include one AIM16, one POW1, one OCON and one MANMOD1.

AIM16 Starter Set 2a (110 VAC)	239.00
AIM16 Starter Set 2e (230 VAC)	249.00

The following modules plug into their respective computers and, when used with a CABLE A24, eliminate the need for custom wiring of the computer interface.

PETMOD (Commodore PET)	49.95
KIMMOD (KIM, SYM, AIM65)	39.95
APMOD (APPLE II)	59.95
TRS-80 MOD (Radio Shack TRS-80)	59.95

μMAC SYSTEMS

Order Form

CONNECTICUT microCOMPUTER, Inc.
150 POCONO ROAD
BROOKFIELD, CONNECTICUT 06804
TEL: (203) 775-9659 TWX: 710-456-0052

The following sets include one AIM16, one POW1, one MANMOD1, one CABLE A24 and one computer interface module

PETSET1a (Commodore PET - 110 VAC)	295.00
PETSET1e (Commodore PET - 230 VAC)	305.00
KIMSET1a (KIM, SYM, AIM65 - 110 VAC)	285.00
KIMSET1e (KIM, SYM, AIM65 - 230 VAC)	295.00
APSET1a (APPLE II - 110 VAC)	295.00
APSET1e (APPLE II - 230 VAC)	305.00
TRS-80 SET1a (Radio Shack TRS-80 - 110 VAC)	295.00
TRS-80 SET1e (Radio Shack TRS-80 - 230 VAC)	305.00

QUANTITY	DESCRIPTION	PRICE	TOTAL

SUBTOTAL	
Handling and shipping — add per order	$3.00
Foreign orders add 10% for AIR postage	
add 7% sales tax	
TOTAL ENCLOSED	

NAME ________________

COMPANY ________________

ADDRESS ________________

CITY ________________

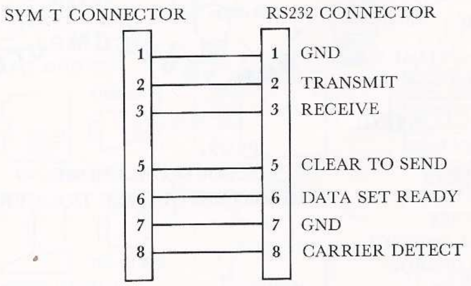
STATE ________ ZIP ________

VISA ○ M/C ○ Expiration date ________

Card number ________________

subroutine to receive the ACK character involved with the protocol.

Figure 1. SYM to Spinwriter Hardware



The assembly language for PRINT is shown in Listing 2.

Listing 2

```
0111- 20 47 8A  1570 PRINT   JSR OUTCHR  ;PRINT AND INC. COUNT
0114- E6 00     1580         INC *COUNT
0116- D0 0C     1590         BNE RETURN
0118- A9 03     1600 ACK     LDA #$03    ;ASCII ETX
011A- 20 47 8A  1610         JSR OUTCHR
011D- 20 58 8A  1620         JSR INCHR   ;WAIT FOR ACK
0120- A9 02     1630         LDA #$02
0122- 85 00     1640         STA *COUNT
0124- 60        1650 RETURN  RTS
```

CYCLE

The function of CYCLE is to read the byte on the data lines during a byte transfer cycle on the IEEE bus. In some cases, the INTERFACE sub-function will need to know the state of some of the other signals during the transfer. CYCLE should therefore sample the signal lines as well.

All the information needed for the IEEE part of the interface can be found in the Commodore CBM manual. The information in this section will deal only with the byte transfer cycle. The remaining information will be presented in the next part of this article. In the discussion below, reference is made to active and inactive devices. An active device is simply one which is participating in the current transfer cycle. Before continuing, you may want to refer to Table 1 which lists the IEEE signals and a brief description of their function. In this table, Listener refers to the receiving device, and Talker refers to the sending device.

The IEEE bus make use of three handshake signals. These are the NRFD, NDAC, and DAV lines. When the CYCLE routine is entered, both the PET, the SYM, and any other active devices are expecting a byte transfer to take place. This means that NRFD and NDAC are low, and NDAV is high. At this point CYCLE sets NRFD high, indicating the SYM is ready to proceed with the byte transfer. Since the NRFD signal line is Wire-ORed, any active device can hold the NRFD line low. This means the cycle doesn't proceed until all active devices indicate they are ready.

Once NRFD goes high, the PET responds by placing the byte to be transferred on the DIO lines and then setting DAV low to indicate valid data. When CYCLE sees DAV go low, it should read the data lines and then sample the signal lines. Now CYCLE sets NDAC high to indicate that the data has been accepted. The NDAC line is also Wire-ORed, so the other active devices must indicate they have accepted the data before the cycle can finish.

When the PET sees the NDAC line go high, it sets DAV low. Once CYCLE sees the DAV line go high, it resets NDAC to the low state completing the cycle. Now CYCLE returns to the calling software. Refer to Listing 3 for the assembly language for this routine.

Listing 3

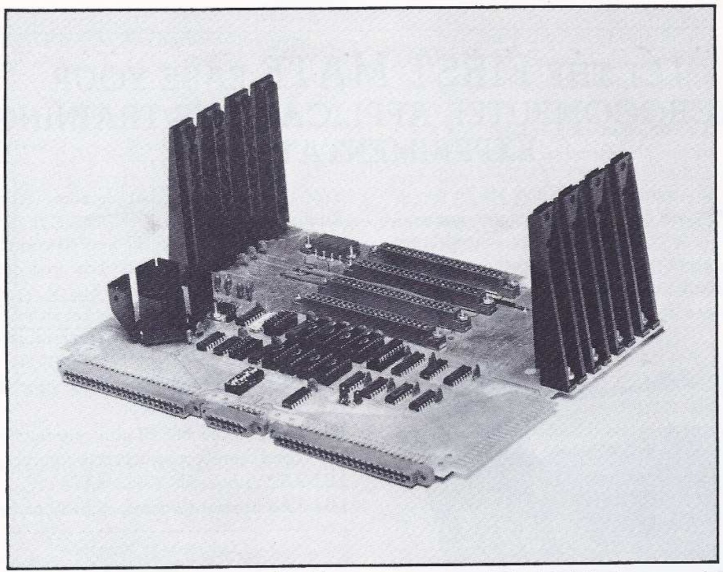
```
00E7- A9 03     1390 CYCLE     LDA #$03
00E9- 8D 00 A8  1400           STA @2IORB   ;NRFD=1 NDAC=0
00EC- 2C 00 A8  1410 @1        BIT @2IORB   ;TEST DAV
00EF- 70 FB     1420           BVS @1       ;BRANCH IF DAV=1
00F1- 6A        1430           ROR A
00F2- 8D 00 A8  1440           STA @2IORB   ;NRFD=0 NDAC=0
00F5- AD 01 A8  1450           LDA @2IORA
00F8- 49 FF     1460           EOR #$FF
00FA- 85 02     1470           STA *DATA
00FC- AD 00 A8  1480           LDA @2IORB
00FF- 85 01     1490           STA *SIGNALS
0101- A9 00     1500           LDA #$00
0103- 8D 00 A8  1510           STA @2IORB   ;NRFD=0 NDAC=1
0106- 2C 00 A8  1520 @2        BIT @2IORB
0109- 50 FB     1530           BVC @2       ;BRANCH IF DAV=0
010B- A9 01     1540           LDA #$01
010D- 8D 00 A8  1550           STA @2IORB   ;NRFD=0 NDAC=0
0110- 60        1560           RTS
```

TABLE 1

NAME	SET BY	DESCRIPTION
DI01- DI08	Talker	Data Input/Output. These lines carry the commands and data.
NRFD	Listener	Not Ready for Data. When low, it means the device is not ready to receive data. It is set high when the device is ready.
DAV	Talker	Data Valid. When high, it means the data on the data lines is not valid. It is set low once all NRFD goes high and valid data has been placed on the data lines.
NDAC	Listener	Not Data Accepted. When low, it means that the data has not been accepted. It is set low once DAV goes low and the data has been latched.
ATN	Talker	Attention. Signals that the byte on the DIO lines is a command.
EOI	Talker	End Or Identify. Signals that the last data byte is being transferred.
IFC		Interface Clear. Resets all devices.

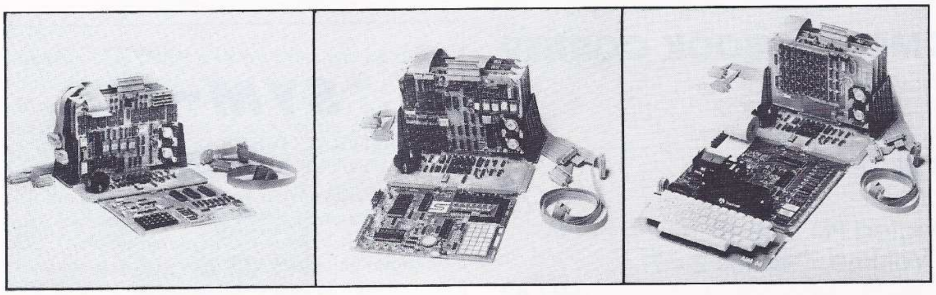
©

Introducing SEAWELL's



Little Buffered Mother

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



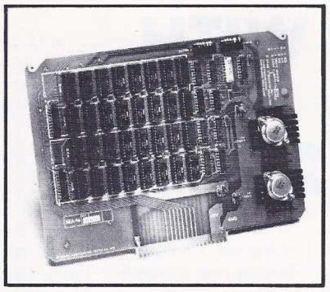
Features:

- 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:

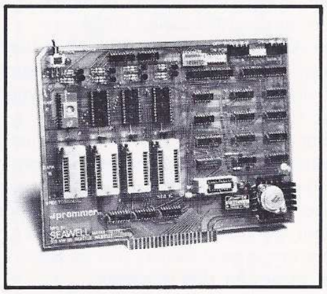
SEAWELL Marketing Inc.
P.O. Box 30505
Seattle, WA 98103



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



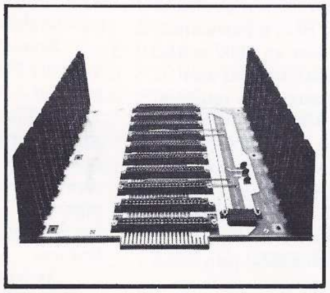
The Prommer

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.

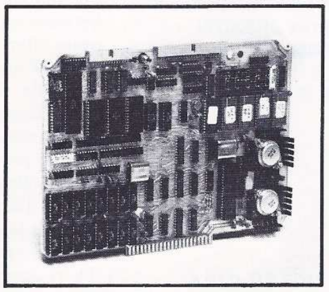
Little Buffered Mother's Compatible Hardware



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.

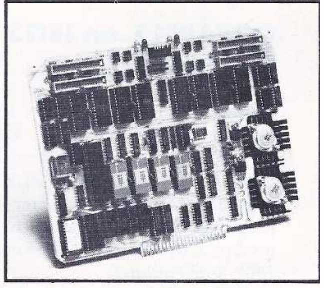
MAXI-MOTHER $135.00



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.

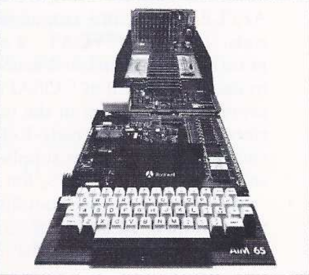
SEA-1, 1.5 MHz $595.00



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/0 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 1/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

COSAPPLE, an 1802 Simulator for The Apple II

From Dann McCreary (Box 16435-Y, San Diego, California 92116, (714) 281-4748): COSAPPLE is an 1802 Simulator and Debugger software package designed to run on an APPLE II 6502 based computer. Any 16K or larger APPLE II can run programs coded in 1802 machine language and can be used as a development aid for checking out modest 1802 applications.

The 1802 is the most widely used and available CMOS microprocessor, and is ideal for low-power or battery-based applications. COSAPPLE is particularly useful to anyone desiring to learn how an 1802 works. All of the 1802's internal registers are labeled and plainly displayed on the APPLE's screen. A touch of the space bar moves COSAPPLE a step at a time through your 1802 program as you observe data chainging in affected registers. COSAPPLE puts you in control of every facet of 1802 operation. In addition, COSAPPLE lets you call 6502 subroutines directly from your 1802 program and even lets you embed 1802 code in the middle of your 6502 programs.

The complete COSAPPLE package is in three parts. First, the COSAPPLE manual - an INTEGER BASIC program which provides the necessary documentation for using COSAPPLE. Second, COSAPPLE - a machine language interpreter of 1802 code. Third, COSYCAT - a complete 1802 game program with a commented source listing which helps to demonstrate all of COSAPPLE's features while providing a little fun in the bargain. A pocket-sized COSAPPLE Ready-Reference Card is also included. COSAPPLE is supplied on cassette tape and is shipped, FIRST CLASS, for the price of $20.00 + $1.50 shipping and handling. California residents must add 6% sales tax. ©

and KIM-1 1802 Simulator

Also from Dann: The COSMAC 1802 SIMULATOR, a program for the KIM-1 that lets it execute the entire 1802 instruction set. It provides features similar to his 8080 SIMULATOR, and allows for input of 1802 external flags and for the 1802 Q output flip-flop.

The simulator is well suited to all but time-sensitive applications. Each simulated instruction must be interpreted by many 6502 instructions. However, the simulator is capable of real-time operations of moderate speed. ©

FUN WITH THE 1802

Larry Sandlin
Drawer 900
P.O. Box 58170
Houston, TX 77058

Have you been looking for a small system that is loads of fun and doesn't cost a bundle? Then the 1802 system should be given strong consideration. The 1802's low price, power consumption, and expandability make it ideal.

Several companies offer an 1802 system starting at under $100.00 dollars. Also available are accessories to build a full blown system with Basic and the works. The 1802 is a fun processor, featuring cosmos circuitry for low power consumption. A single phase clock is all that is required and can run at slower speeds without damage to the processor or the program. The 1802 has been approved by NASA for the Space Shuttle program because of these features.

As with most of the low price systems available, the 1802 uses only 256 words bytes of memory. However this can be expanded to use a full 65535 words of Ram or Rom or any combination of the two. A special feature of the 1802 called DMA or Direct Memory Addressing allows the user to load a program into memory without the use of a monitor program which many other systems require. I wouldn't be so bold as to say that the 1802 has a serial port built in, but with the four flag lines and one Q line out, programming for serial communications between any I/0 device is not only possible but is quite often done. Software is not as plentiful as for some of the other systems but there is still a lot available. There are several basics, text editors, assemblers and games for this processor.

I, among many others, spend a considerable time writing programs and suspect that in the near future there will be an abundance of software on the market for the 1802. Programming the 1802 is not difficult. As with learning to program other systems, presistance and interest are the main requirements. There are several sources available but one of the best manuals for learning this processor is the users manual for the CDP 1802 Cosmac Microprocessor. This book is written by RCA and explains both the architecture and software in simple, easy to understand terms. There are several user groups and clubs that publish news letters on 1802 systems and software.

Just remember, whichever processor or system you choose, you're going to have lots of fun and a great learning experience.

Dice Toss

This game uses the hex displays to represent the two dice, and is written in machine language. Load the program in the Elf computer in location 0000 to 0030. The program will run when the run switch is turned on. To roll the dice, press the input switch and hold for a second. Release this switch and a random number will appear on the hex display, 1 thru 6. This represents your two dice. Example: 11 is snake eyes and a 66 is box cars.

```
0000 ;          0001           ..DICE TOSS
0000 ;          0002           STORE=  1
0000 ;          0003           COUNT1= 2
0000 ;          0004           COUNT2= 3
0000 ;          0005
0000 ;          0006           ORG #0000
0000 ;          0007           START=  0
0000 ;          0008
0000 90;        0009           GHI START
0001 B1;        0010           PHI STORE..SET UP STORAGE PAGE
0002 F82F;      0011           LDI HERE
0004 A1;        0012           PLO STORE..SETUP STORAGE LOC,
0005 E1;        0013           SEX STORE
0006 7A;        0014           REQ....TURN Q LAMP OFF
0007 12;        0015 LOOP1:    INC COUNT1
0008 82;        0016           GLO COUNT1
0009 FF07;      0017           SMI 7...INC COUNTER AND TEST
000B 3225;      0018           BZ RESET1..BRANCH IF GREATER THAN SIX
000D 3F07;      0019 RET1:     BN4 LOOP1
000F 7B;        0020           SEQ..TURN Q LAMP ON
0010 ;          0021
0010 13;        0022 LOOP2:    INC COUNT2..START INC SECOND DIE
0011 83;        0023           GLO COUNT2
0012 FF07;      0024           SMI 7..TEST AS ABOVE
0014 322A;      0025           BZ RESET2..IF IT'S OVER RESET IT
0016 ;          0026
0016 3710;      0027 RET2:     B4 LOOP2..WAIT FOR KEY TO BE RELEASED
0018 ;          0028
0018 82;        0029           GLO COUNT1
0019 51;        0030           STR STORE..STORE IT
001A 83;        0031           GLO COUNT2
001B FEFEFEFE;  0032           SHL;SHL;SHL;SHL
001F F1;        0033           OR..MIX THEM TOGETHER
0020 51;        0034           STR STORE..STORE FINAL RESULTS
0021 64;        0035           OUT 4..PUT ON HEX DISPLAY
0022 21;        0036           DEC STORE
0023 3006;      0037           BR LOOP1 -1
0025 ;          0038
0025 F801;      0039 RESET1:   LDI 1
0027 A2;        0040           PLO COUNT1
0028 300D;      0041           BR RET1
002A ;          0042
002A F801;      0043 RESET2:   LDI 1
002C A3;        0044           PLO COUNT2
002D 3016;      0045           BR RET2
002F ;          0046
002F 00;        0047 HERE:     IDL..STORAGE AREA..
0030 ;          0048           END...END OF PROGRAM
```

©

The Serious Side of the 1802

Robert Lock

As Larry points out here, the CMOS 1802 is approved for space shuttle use. With Issue #2 of compute II, we'll introduce a continuing column by Dann McCreary on applications of the 1802. ©

Improved Pulse Counting Software For The 6522 VIA

Marvin L. De Jong
Dept. of Mathematics-Physics
The School of the Ozarks
Pt. Lookout, MO 65726

Ever since I began playing with the 6522 I have been trying to find a program that would use the 6522 to count pulses for an *exact* one second interval. By exact I mean one million clock cycles, not one million plus or minus several instruction intervals. Of course, it should be noted that if the system clock frequency is not *exactly* one Megahertz then an error of several instruction intervals may not be particularly important. In this connection, the measurements I have made of clock frequencies on a few KIM-1s and one AIM 65 show that errors of several hundred parts per million are not unusual, so if your twenty-four hour clock runs slow or fast, do not be surprised.

In any case, assuming that the system clock frequency is precise to say one part per million, the program supplied in this note will count pulses for an interval that is as precise as the system clock frequency. The assembly language program to count pulses for exactly one second (one million clock cycles) is given in Table 1, and the simple interface circuit it requires is given in Figure 1. A BASIC program to convert the pulse count to decimal and display it is given in Table 2. This program works on my AIM 65, and it will probably have to be modified for other machines.

The assembly language program in Table 1 makes use of the T1 timer in its one-shot mode with PB7 enabled. That is, the T1 timer is programmed to produce a time interval of 50,000 clock cycles, and during that interval of 0.05 s PB7 is held at logic zero. Refer to Figure 1 and note that when PB7 is at logic zero the pulses from some external device will be gated to PB6, the pulse-counting pin for the T2 counter/timer. In order to produce pulse-counting intervals that are longer than 0.05 s, the T1 timer is reloaded and started N times, where N is an eight-bit number stored in a memory location labeled CNTR in Table 1. Thus, if N = 2 the counting interval is 0.1 s, if N = 20 the counting interval is 1.0 s, and if N = 200 the counting interval is 10 s. These numbers must be converted to hexadecimal numbers before using them in the program.

While T1 is timeing-out it is read continuously so that it may be reloaded and started after *exactly* 50,000 clock cycles. This prevents PB7 from reaching logic one any time during the N timing intervals. If we were to allow T1 to time-out and then reload and start it, PB7 would toggle from logic zero to logic one and back to logic zero, with the possibility of producing an extraneous count on PB6. Thus, the program loop starting from REPEAT in Table 1 and ending with DUMMY in the same listing is *tuned* to take exactly 50,000 clock cycles. Each time through the loop N is decremented, until it reaches zero at which time T1 is finally allowed to time-out for the last time.

When T1 times out for the last time, no more pulses will reach PB6. At this time the interrupt flag register (IFR) on the 6522 is read first. If the T2 flag is set, then the pulse count was greater than \$010000 (65536_{10}) because the T2 counter was initially loaded with \$FFFF. If the T2 interrupt flag (IFR5) is set, then the most-significant byte, PLUSHI, of the pulse-count storage locations is incremented. Otherwise it is cleared. After this operation, the T2 counter is read and the resulting pulse counts are loaded into PLSMI, the middle byte of the three-byte pulse-count storage locations, and PLSLO, the least-significant byte of the pulse-count storage locations. The program then uses a JMP instruction to return to the BASIC calling program given in Table 2. Other BASICs may use a different return technique.

The most obvious application of pulse counting is a simple frequency meter. The programs and interface described here will count at a maximum pulse count of 131,071 counts during whatever counting interval (0.1 s, 1.0 s, or 10 s) you choose. Note that 131,071 = \$01FFFF. Other applications include voltage-to-frequency converters and temperature-to-frequency converters. Commercial tachometer pickups produce a pulse rate that is proportional to the angular velocity (RPM) of a rotating shaft. The 6522 can be used to measure this pulse rate and the microcomputer can convert it to rotations per minute. The 6522 can also be interfaced to Geiger counters (GM tubes) or scintillation detectors to count nuclear events. There are a variety of new transducers appearing (temperature, light intensity, pressure) that can be used with a V/F converter to produce a pulse rate that is directly proportional to the physical quantity being measured. Although direct analog-to-digital (A/D) conversion is faster than pulse counting, it usually requires a much more sophisticated interface. In applications where speed is not a problem, investigate the possibility of using this simple program and interface.

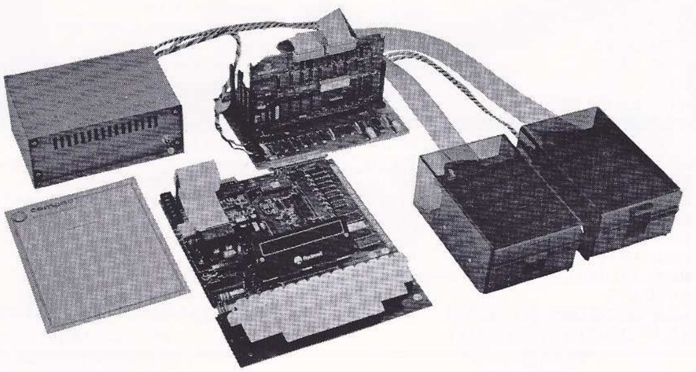


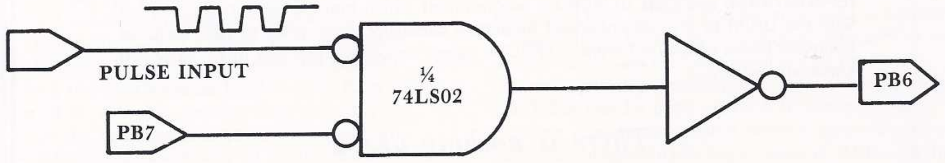
Table 1. Simple pulse counting program for the 6522.

```
$0F00 A9 01      START    LDA $80      Make PB7 an output pin by loading
$0F02 8D 02 A0            STA PBDD     one into the data direction register.
$0F05 A9 A0               LDA $A0      Set up the ACR so T1 runs once, PB7
$0F07 8D 0B A0            STA ACR      enabled, and T2 counts pulses.
$0F0A A9 14      HERE     LDA $14      Set up counter to do 20 ($14) intervals
$0F0C 85 30               STA CNTR     of 0.05s, totaling one second.
$0F0E A9 FF               LDA FF       Initialize T2 to start with
$0F10 8D 08 A0            STA T2LL     $FFFF and count down.
$0F13 8D 09 A0            STA T2CH     T2 is now ready to count when PB7
$0F16 A9 4F               LDA $4F      goes to logic zero.
$0F18 8D 04 A0            STA T1LL     Set up T1 to count 5000 clock
$0F1B A9 C3      REPEAT   LDA $C3      pulses. $C34F + 1 = 50000.
$0F1D 8D 05 A0            STA T1LH     Start T1, PB7 to logic zero.
$0F20 AD 05 A0   WAIT     LDA T1CH     Read the T1 counter, high-order byte.
$0F23 D0 FB               BNE WAIT     Wait until it is zero. These
$0F25 AD 04 A0   LOOP     LDA T1CL     instructions are part of a tuned
$0F28 C9 19               CMP $19      loop designed to wait exactly 50000
$0F2A B0 F9               BCS LOOP     cycles before starting T1 again.
$0F2C C6 30               DEC CNTR     The loop is repeated until the
$0F2E EA                  NOP          contents of CNTR = 0.
$0F2F 90 00               BCC DUMMY    These two dummy instructions tune
$0F31 D0 E8      DUMMY    BNE REPEAT   the loop.
$0F33 A9 00               LDA $00      Clear the most-significant byte of
$0F35 85 33               STA PLSHI    the pulses counted.
$0F37 AD 0D A0            LDA IFR      Read the IFR to see if count went
$0F3A 29 20               AND $20      through zero. Mask bits other than
$0F3C F0 02               BEQ OVER     T2 flag. If it was set, add $010000
$0F3E E6 33               INC PLSHI    to pulse counter.
$0F40 38         OVER     SEC          Otherwise, set carry flag and
$0F41 A9 FF               LDA $FF      perform subtraction to see how many
$0F43 ED 09 A0            SBC T2CH     pulses were counted.
$0F46 85 32               STA PLSMI    Result into middle byte of pulse
$0F48 A9 FF               LDA $FF      counter.
$0F4A ED 08 A0            SBC T2CL
$0F4D 85 31               STA PLSLO    Result into low-order byte of pulse.
$0F4F 4C D1 C0            JMP BASIC    Return to BASIC.
```

Table 2. Counting Pulses with a BASIC program.

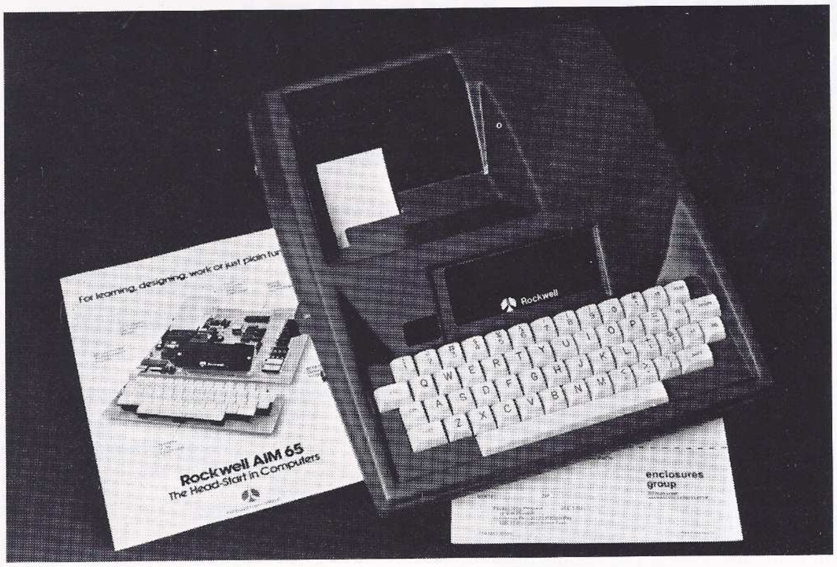
```
10 REM THIS PROGRAM REQUIRES THE MACHINE LANGUAGE ROUTINE IN TABLE 1.
20 POKE 04,00: POKE 05,15
30 Y = USR(0)
40 X = PEEK(49) + 256*PEEK(50) + 65536*PEEK(51)
50 PRINT X; ''PULSES PER SECOND''
60 GO TO 30
70 END
```

Figure 1.



Interface circuit for the pulse-counting program of Table 1. The inverter can be implemented with one of the other gates on the 74LS02 chip. The incoming pulse train must be at TTL logic levels. ©

PERFECT AIM


For learning, designing, work or just plain fun
Rockwell AIM 65
The Head-Start in Computers
Rockwell
enclosures group

ATTRACTIVE FUNCTIONAL PACKAGING FOR YOUR AIM-65 MICROCOMPUTER

- Professional Appearance
- Striking Grey and Black Color Combination
- Protects Vital Components

ENGINEERED SPECIFICALLY FOR THE ROCKWELL AIM-65

- All Switches Accessible
- Integral Reset Button Actuator
- Easy Paper Tape Replacement

EASILY ASSEMBLED

- Absolutely No Alteration of AIM-65 Required
- All Fasteners Provided
- Goes Together in Minutes

MADE OF HIGH IMPACT STRENGTH THERMOFORMED PLASTIC

- Kydex 100*
- Durable
- Molded-In Color
- Non-Conductive

AVAILABLE FROM STOCK

- Allow Three to Four Weeks for Processing and Delivery
- No COD's Please
- Dealer Inquiries Invited

TO ORDER: 1. Fill in this Coupon (Print or Type Please)
2. Attach Check or Money Order and Mail to:

NAME ______________________________

STREET ______________________________

CITY ______________________________

STATE ____________________ ZIP __________

SAE 1-1 PLEASE SHIP PREPAID ________ SAE 1-1(s)
@ $43.50 each
California Residents Please Pay
$46.33 (Includes Sales Tax)

SAE 1-2 PLEASE SHIP PREPAID ________ SAE 1-2(s)
@ $46.50 each
California Residents Please Pay
$49.52 (Includes Sales Tax)

enclosures group

771 bush street
san francisco, california 94108

*TM Rohm & Hass Patent Applied For

PRINTING A SYMBOL TABLE FOR THE AIM-65 ASSEMBLER

Richard F. Olivo
Biological Sciences, Smith College
Northampton, MA 01063

The assembler for Rockwell's AIM 65 makes assembly-language programming very convenient, particularly in conjunction with the excellent editor that is part of AIM 65's monitor. However, the assembler does not include an option to print the symbol table, although it does create such a table in memory. The following program is one way of decoding and printing the symbol table. In revising a program, a print-out of the symbol table can be very helpful.

On entering the AIM 65 assembler from the monitor, you are asked for the addresses that start and end the symbol table. The assembler places your answers in zero-page addresses 3A, 3B ("FROM") and 3E, 3F ("TO"). After assembly, the total number of symbols is available in addresses 0B, 0C (in high, low order). The symbol table itself consists of sequential eight-byte entries. The first six bytes of each entry are the symbol name, in ASCII characters (the assembler enters spaces if the symbol is less than six characters), and the last two bytes are the symbol's address, in hex notation.

The program to print the table reads through the table using indirect addressing indexed by Y. It establishes the variable ADDR (at locations 00 and 01), which provides the address of the first character of the current symbol. ADDR is initially set equal to the address in "FROM (3A, 3B); it is incremented by eight after each symbol is printed. For each symbol, the Y register is incremented from zero to seven to access the successive bytes of that symbol.

A second variable, COUNT (addresses 02 and 03), keeps track of the number of symbols that remained to be printed. COUNT is initially set equal to one less than the total number of symbols (from addresses 0B and 0C), and it is decremented by one after each symbol is printed. After COUNT reaches zero (the last symbol is numbered zero, which is why the initial count is one less than the total), the program exits and prints the total number of symbols in hex notation. The program uses AIM monitor subroutines to print the ASCII and hex characters. It also turns the AIM printer on and off at the start and end of the table, which I find very handy.

The listing given below places the program at locations 0200-027D, which are available on every AIM 65. The program could of course be placed in other memory locations, and it would be very convenient in a PROM. At the end of the listing, the program was run to list its own symbol table.

```
---------------------
==0000 BLANK=$E83E

==0000 CRLOW=$EA13

==0000 EQUAL=$E7D8

==0000 PRIASC=$E97A

==0000 PRIFLG=$A411

==0000 PRIHX2=$EA46

==0000 ADDR=0
==0000 COUNT=ADDR+2
==0000
       *=$0200
---------------------
INITL ADDR,COUNT,Y

==0200
;"FROM" = 3A,3B
==0200 SYMTBL
A53A   LDA $3A
8500   STA ADDR
A53B   LDA $3B
8501   STA ADDR+1
;ADDR ACCESSES TABLE

A50B   LDA $0B
8502   STA COUNT
A50C   LDA $0C
8503   STA COUNT+1
==0210
;COUNT=SYMBOLS TO GO
C603   DEC COUNT+1
;FIRST SYMB=0, NOT 1

A000   LDY #0
;INDX 8 BYTES/SYMBOL

A980   LDA #$80
8D11A4 STA PRIFLG
;TURN PRINTER ON
2013EA JSR CRLOW
2013EA JSR CRLOW
;SKIP 2 LINES AT TOP
---------------------
MAIN LOOP

==021F SYMLP
B100   LDA (ADDR),Y
C006   CPY #6
;BYTES 0-5 =ASCII
F007   BEQ SPACE
;PRINT 6 ASCII CHAR.
207AE9 JSR PRIASC
C8     INY
4C1F02 JMP SYMLP

;PRINT SPACE & EQUAL
==022C SPACE
48     PHA
203EE8 JSR BLANK
20D8E7 JSR EQUAL
203EE8 JSR BLANK
;NEXT 2 BYTES = HEX
68     PLA
2046EA JSR PRIHX2
C8     INY
B100   LDA (ADDR),Y
==023D
2046EA JSR PRIHX2
2013EA JSR CRLOW
;HAVE PRINTED 1 LINE
---------------------
DECR COUNT & TEST

C603   DEC COUNT+1
A9FF   LDA #$FF
C503   CMP COUNT+1
;FF = BORROW
D006   BNE NXTADR
C602   DEC COUNT
==024D
C502   CMP COUNT
;FF = DONE
F012   BEQ DONE
---------------------
UPDATE ADDRESS

==0251 NXTADR
18     CLC
A500   LDA ADDR
;LOW BYTE
6908   ADC #8
8500   STA ADDR
A501   LDA ADDR+1
;HIGH BYTE
6900   ADC #0
8501   STA ADDR+1
A000   LDY #0
4C1F02 JMP SYMLP
---------------------
PRINT TOTAL & EXIT

==0263 DONE
2013EA JSR CRLOW
A50B   LDA $0B
2046EA JSR PRIHX2
A50C   LDA $0C
2046EA JSR PRIHX2
;PRINT TOTAL,SKIP LN
2013EA JSR CRLOW
==0273
2013EA JSR CRLOW
A900   LDA #0
8D11A4 STA PRIFLG
;TURN PRINTER OFF
4C82E1 JMP $E182
;JUMP TO MONITOR

        .END

BLANK  = E83E
CRLOW  = EA13
EQUAL  = E7D8
PRIASC = E97A
PRIFLG = A411
PRIHX2 = EA46
ADDR   = 0000
COUNT  = 0002
SYMTBL = 0200
SYMLP  = 021F
SPACE  = 022C
NXTADR = 0251
DONE   = 0263

000D
```

©

EXCERT, INCORPORATED
*** AIM-65 ***

SPECIAL

A65-4AB	AIM-65 w/4K RAM Assembler & BASIC ROM	$595

P/N		QTY 1-9
A65-1	AIM-65 w/1K RAM	$375
A65-4	AIM-65 w/4K RAM	$450
A65-A	Assembler ROM	$85
A65-B	BASIC ROM	$100

SPARE PARTS (When Available)

A65-P	Printer	$40
A65-D	Complete Display Bd.	$65
	w/Exchange of Old Bd.	$40
A65-K	Keyboard	$40

ACCESSORIES

P/NO. **QTY 1-9**

Power Supplies (fully AIM-65 Compatible)

PRS3	+5V at 3A, +24V at 1A w/mtg hardware, cord, etc.	$65
PRS4	+5V at 2A, +24V at .5A w/mtg hardware, cord, etc.	$50

From The Enclosure Group

ENC1	AIM-65 case w/space for PRS3/PRS4	$45
ENC1A	AIM-65 case w/space for PRS3/PRS4 and one expansion board	$49

Cases with Power Supplies

ENC3	ENC1 w/PRS3 mounted inside	$115
ENC3A	ENC1A w/PRS3 mounted inside	$119
ENC4	ENC1 w/PRS4 mounted inside	$100
ENC4A	ENC1A w/PRS4 mounted inside	$104

From The Computerist, Inc.

MCP1	Mother Plus^{tm} - Dual 44 pin mother card takes MEB1, VIB1, PTC1, fully buffered, 5 expansion slots underneath the AIM	$80
MEB1	Memory Plus^{tm} - 8K RAM, 8K PROM sockets, 6522 I/O chip and programmer for 5V EPROMS **(w/cables $215)**	$200
PTC1	Proto Plus^{tm} - Prototype card same size as KIM-1 MEB1, VIB1	$40
VIB1	Video Plus^{tm} - Video bd w/128 char, 128 user char, up to 4K display RAM, light pen and ASCII keyboard interfaces w/cables	$245

P/NO. **QTY 1-9**

From Seawell Marketing, Inc.

MCP2	Little Buffered Mother^{TM}-Single 44 pin (KIM-4 style) mother card takes MEB2, PGR2, PTC2 and PI02. Has on board 5V regulator for AIM-65, 4 expansion slots. Routes A&E signals to duplicates on sides w/4K RAM	$199
MEB2	SEA 16^{TM}-16K static RAM bd takes 2114L w/regulators and address switches 16K	$325
PGR2	Prommer^{TM}-Programmer for 5V EPROMS w/ROM firmware, regulators, 4 textool sockets, up to 8 EPROMS simultanously, can execute after programming	$299
PI02	Parallel I/O Bd w/4-6522's	$260
PTC2	Proto/Blank^{TM}-Prototype card that fits MCP2	$49
PTC2A	Proto/Pop^{TM}-w/regulator, decoders, switches	$99

From Optimal Technology

ADC1	A/D: 8 channels; D/A: 2 channels. Requires ±12v to ±15 volts @ 100 ma and 2 I/0 ports from user 6522	$115.00

Miscellaneous

TPT2	Approved Thermal Paper Tape 5/165 rolls	$10
MEM6	6/2114 RAM Chips	$45

CLOSE-OUT!
From Beta Computer

MEB3	32K Dynamic Memory Card w/on bd DC to DC converters (5V only .8A max)	$399

SYSTEMS

We specialize in assembled and tested systems made from the above items. Normally, the price will be the total of the items, plus $5 for shipping, insurance and handling. Please call or write for exact prices or if questions arise.

Higher quantities quoted upon request.
COD's accepted.
Add $5 for shipping, insurance, and handling.
Minnesota residents add 4% sales tax.

Mail Check or Money Order To:
EXCERT, INC.
Educational Computer Division
P.O. BOX 8600
WHITE BEAR LAKE, MN. 55110
612-426-4114

Hard Copy Graphics For the Kim

Keith Sproul
1368 Noah Road
North Brunswick, New Jersey 08902

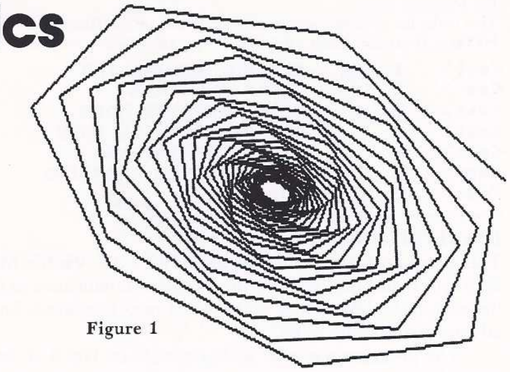


Figure 1

There are many different video boards out these days, each with its own advantages and each designed for a different system. Some video boards are 'Byte Mapped', that is they display an ASCII character for every byte in memory. The other type of video board is a 'Bit Mapped' screen, this type displays one 'pixel', a dot on the television screen, for each bit in memory. Both of these types have advantages. The byte mapped screens display a character at a time and are good for high speed text applications and sometimes rough graphics. The bit mapped screens by definition give you higher resolution graphics. With a bit mapped video board you can do professional quality graphics. Characters are still possible on a bit mapped screen, but they have to be 'drawn'.

Micro Technology Unlimited of Manchester, New Hampshire makes a 'VISIBLE MEMORY' video board that is of the second type. This video board displays 200 rows of 40 bytes across. At 8 bits to the byte, this produces a graphic display of 320 by 200 pixels. That is fairly high resolution, even higher than 'High Res' on the APPLE. Besides being a good graphics board, this 'Visible Memory' doubles as 8K of memory when the graphics is not needed.

Hal Chamberlin, who is well known in the microcomputer industry and also works for M.T.U., wrote a 'V.M. Support Package' for this board. This package does everything from plotting points to drawing characters on the screen. This 'software character generator' enables the user to display text in any format, including sub-scripted and super-scripted characters. The user can also redefine his own characters or the entire character set. This enables the use of languages other than English to be displayed on the screen, a feature that is very rarely seen.

All of this is fine and pretty on the screen, but what if hard copy is needed? Plotters can be used to produce hard copy, but they are extremely expensive and are dedicated to plotting. Plotters also require extensive software to run properly. A different alternative is to use a DIABLO HyTerm II terminal. This terminal produces about the best print quality available and has built-in graphics capabilities. By using the graphics mode, the image from a bit mapped screen can be 'plotted' on the terminal, bit for bit, producing the exact image that was on the screen. The program at the end of this article is a 6502 assembler routine to print the entire contents of the 'Visible Memory' screen onto a DIABLO HyTerm II. The ideas involved can be adapted to other graphics boards as well, but will only work with 'bit mapped' video boards such as the M.T.U. 'Visible Memory'. Figure 1 was drawn on a M.T.U. 'Visible Memory' video board and then 'plotted' on a DIABLO using this program.

This program has been somewhat optimized for speed because of the length of time that is required for printing the images. Further optimization is possible such as adding reverse printing, and using other specialized features of the DIABLO. This was omitted to keep the program small and simple.

The screen is processed in four steps, the Page, the Line, the Byte, and the Bit. The program prints a Page by calling the Line subroutine 200 times. This routine prints a Line by calling the Byte routine 40 times, which in turn prints a byte by calling the Bit or Dot routine 8 times. The Line routine checks for the end of the line, only printing up to the last non-zero byte. It also skips any completely blank line, immediately going onto the next line to decrease printing time. The Byte routine shifts the byte to the left to determine if each bit is a 'one' or a 'zero'. The program can take up to an hour to print a complicated design. The average picture takes approximately 25 minutes. The reason for the big difference in time is that to print a 'one', a 'period' and a 'space' have to be printed, and to print a 'zero', just a space has to be printed, so a 'one' takes twice as long as a 'zero'.

This same set of ideas can be used to digitize pictures using a light sensitive diode fastened to the printing mechanism of the DIABLO. Instead of reading from the Visible Memory and writing to the plotter, the process is reversed, reading from the DIABLO and writing to the Visible Memory. This method was discussed in October 1979 issue of 'Dr. Dobb's Journal'. I hope to experiment with this idea in the future but have not had time to do it yet.

Of course this is not the cheapest way to go, but a DIABLO can be used for a lot more than just plotting. It can also be used as a normal hardcopy device for text processing or just listings whereas a plotter can only be used for one thing, plotting.

For reference only:
This is the list of escape sequences understood by the Diablo HyTerm II printer. The one(s) with '*' are used in this program.

<esc>	1		SET HORIZONTAL TAB
<esc>	2		CLEAR ALL TABS
<esc>	3	*	SETS GRAPHICS MODE
<esc>	4		CLEARS GRAPHICS MODE
<esc>	5		FORWARD PRINTING
<esc>	6		SETS REVERSE PRINTING
<esc>	7		not used
<esc>	8		CLEAR INDIVIDUAL TAB
<esc>	9		SET LEFT MARGIN
<esc>	0		SET RIGHT MARGIN
<esc>	A		PRINT IN RED
<esc>	B		PRINT IN BLACK
<esc>	D		NEGATIVE 1/2 LINE FEED
<esc>	U		1/2 LINE FEED
<esc>	<ht>(n)		ABSOLUTE HORIZONTAL TAB
<esc>	<vt>(n)		ABSOLUTE VERTICAL TAB
<esc>	<lf>		NEGATIVE LINE FEED

DIABLO Plot Routine

This program takes the image on an M.T.U. Visible Memory Board and plots that image on a DIABLO HyTerm II printer. The routine has some intelligence in that it checks for the end of each line and only plots up to the end of the line, going on with the next line when finished. If the entire line is blank, it skips that line and advances to the next line.

The program waits for a character to be typed at the keyboard of the Diablo so that you can straighten the paper before it starts printing, be sure to type a character that does not print on the terminal (i.e. a space).

NOTE: graphics mode (and some of the other modes) are cleared when the Diablo receives a (cr) (hex $0D), so graphics mode is reset at the beginning of every line.

```
      00E0                  LINADR  =       $00E0          ; ADDR OF CURRENT LINE
      00E2                  BCOUNT  =       $00E2          ; BYTE POINTER
      00E4                  ACC     =       $00E4

      2000                  VM.ORG  =       $2000          ; Visible Memory Start Address
                             SBTTL  Plot Program

     0200                           =       $0200

0200   4C   09   02         START:  JMP     P.PAGE         ; TAILORING VECTORS
0203   4C   A0   1E         OUTCHR: JMP     $1EA0          ; KIM PRINT CHAR ROUTINE (OR YOUR OWN)
0206   4C   5A   1E         GETCHR: JMP     $1E5A          ; KIM GET CHAR ROUTINE (OR YOUR OWN)

0209                        P.PAGE:                        ; PLOT PAGE
0209   20   06   02                 JSR     GETCHR         ; WAIT FOR CHAR TO ALLOW SETTING UP OF PAPER
020C   20   B8   02                 JSR     CRLF           ; Put Print Wheel at LEFT Column
020F   A2   08                      LDX     #8             ; 8 Lines down the Page (CHANGE IF DESIRED)
0211   A9   0A                LINFD: LDA    #$0A           ; <lf>
0213   20   A6   02                 JSR     PRTCHR
0216   CA                           DEX
0217   D0   F8                      BNE     LINFD

0219   A9   00                      LDA     #VM.ORG\
021B   85   E0                      STA     LINADR         ; INIT LINADR to VM.ORG
021D   A9   20                      LDA     #VM.ORG^
021F   85   E1                      STA     LINADR+1

0221   20   6F   02                 JSR     INIPLT         ; INIT Printer for PLOTTING
0224   A2   C8                      LDX     #200           ; 200 LINES/PAGE

0226                        PLOT:                          ; PLOT PAGE
0226   A0   27                      LDY     #39            ; INIT INDEX Pointer
0228   B1   E0              CHKBLK: LDA     (LINADR),Y     ; TEST FOR END OF LINE
022A   D0   05                      BNE     EOL
022C   88                           DEY
022D   D0   F9                      BNE     CHKBLK         ; CHECK NEXT BYTE
022F   F0   06                      BEQ     NXTLIN         ; Advance to next LINE
                                                           ; IF ALL 40 BYTES = $00
0231   C8                   EOL:    INY                    ; MAKE Y = NUMBER OF BYTES TO PRINT
0232   84   E2                      STY     BCOUNT         ; SAVE # OF BYTES IN LINE (THAT AREN'T ZERO)
0234   20   7A   02                 JSR     P.LINE         ; PLOT the LINE at (LINADR)

0237   18                   NXTLIN: CLC
0238   A5   E0                      LDA     LINADR
023A   69   28                      ADC     #40            ; 40 BYTES/LINE
023C   85   E0                      STA     LINADR
023E   A5   E1                      LDA     LINADR+1
0240   69   00                      ADC     #0
0242   85   E1                      STA     LINADR+1
0244   CA                           DEX
0245   F0   1A                      BEQ     EXIT
                                                           ; End of Line
0247   A9   0D                      LDA     #$0D           ; <cr>
0249   20   A6   02                 JSR     PRTCHR
```

```
024C   A9  00                LDA    #$00          ; <nul>
024E   20  A6  02            JSR    PRTCHR
0251   A9  00                LDA    #$00          ; <nul>
0253   20  A6  02            JSR    PRTCHR
0256   20  6F  02            JSR    INIPLT        ; INITIALIZE TO GRAPHICS MODE
0259   A9  0A                LDA    #$0A          ; <lf>
025B   20  A6  02            JSR    PRTCHR        ; Advance Paper 1/48 IN.
025E   4C  26  02            JMP    PLOT          ; Go, BACK & Do NEXT LINE

0261   A9  0D        EXIT:   LDA    #$0D          ; <cr>
0263   20  A6  02            JSR    PRTCHR        ; Clear GRAPHICS Mode
0266   A9  0C                LDA    #$0C          ; <ff>
0268   20  A6  02            JSR    PRTCHR        ; Advance to top of next page
026B   60                    RTS                  ; IF USED AS A SUBROUTINE
026C   4C  4F  1C            JMP    $1C4F         ; OTHERWISE EXIT TO SYSTEM MONITOR
                                                  ; (KIM WARM START)

                     .SBTTL  Plot Subroutines

                                                  ; INIT the DIABLO to PLOT MODE
026F   A9  1B        INIPLT: LDA    #$1B          ; <esc> 3
0271   20  A6  02            JSR    PRTCHR        ; Escape Sequence for PLOT MODE
0274   A9  33                LDA    #'3
0276   20  A6  02            JSR    PRTCHR
0279   60                    RTS


027A                 P.LINE:                      ; PLOT 1 LINE (320 BITS or 40 BYTES)
                                                  ; (or the # of BYTES in BCOUNT)
027A   A0  00                LDY    #0            ; INIT Index Pointer
027C   98            PLINE1: TYA                  ; SAVE Y
027D   48                    PHA
027E   B1  E0                LDA    (LINADR),Y
0280   20  8B  02            JSR    P.BYTE        ; PLOT the BYTE at (LINADR),Y
0283   68                    PLA                  ; RESTORE Y
0284   A8                    TAY
0285   C8                    INY                  ; Advance Pointer to next BYTE
0286   C6  E2                DEC    BCOUNT
0288   D0  F2                BNE    PLINE1
028A   60                    RTS                  ; Return when finished


028B                 P.BYTE:                      ; PLOT 1 BYTE
028B   48                    PHA                  ; SAVE ACC
028C   A0  08                LDY    #8            ; 8 BITS/BYTE
028E   68            PBYTE1: PLA                  ; RESTORE ACC
028F   0A                    ASL    A             ; SHIFT BIT into CARRY
0290   48                    PHA                  ; RE-SAVE ACC
0291   20  99  02            JSR    P.BIT         ; PLOT THE BIT
0294   88                    DEY                  ; KEEP TRACK OF BITS DONE
0295   D0  F7                BNE    PBYTE1        ; Do all 8 BITS
0297   68                    PLA                  ; RESTORE STACK
0298   60                    RTS


0299   90  05        P.BIT:  BCC    PBLANK        ; IF BIT = 0
029B   A9  2E                LDA    #'.
029D   20  A6  02            JSR    PRTCHR        ; IF BIT = 1
02A0   20  A4  02    PBLANK: JSR    OUTSPA        ; Advance Print wheel 1/60 IN.
02A3   60                    RTS

                      SBTTL  System Subroutines

02A4   A9  20        OUTSPA: LDA    #'            ; PRINT A <space>
                                                  ; FALL THROUGH TO PRTCHR

                                                  ; PRINT ASCII CHAR SAVING A, X, & Y
02A6   85  E4        PRTCHR: STA    ACC           ; SAVE ACC
02A8   48                    PHA
02A9   8A                    TXA                  ; SAVE X
02AA   48                    PHA
02AB   98                    TYA                  ; SAVE Y
02AC   48                    PHA
02AD   A5  E4                LDA    ACC
02AF   20  03  02            JSR    OUTCHR        ; KIM OUTPUT ROUTINE (OR SYSTEM OUTPUT)
02B2   68                    PLA
02B3   A8                    TAY                  ; RESTORE Y
02B4   68                    PLA
02B5   AA                    TAX                  ; RESTORE X
02B6   68                    PLA                  ; RESTORE ACC
02B7   60                    RTS

```

```
195  02B8   48               CRLF:   PHA                   ; SAVE ACC
196  02B9   8A                       TXA                   ; SAVE X
197  02BA   48                       PHA
198  02BB   A9   0D                  LDA     #$0D          ; SUBROUTINE TO
199  02BD   20   03   02             JSR     OUTCHR        ; PRINT <cr>,<lf>
200  02C0   A9   0A                  LDA     #$0A
201  02C2   20   A6   02             JSR     PRTCHR
202  02C5   A2   04                  LDX     #4            ; OUTPUT 4
203  02C7   A9   00          NULL:   LDA     #$00          ; <nul> ($00)
204  02C9   20   03   02             JSR     OUTCHR
205  02CC   CA                       DEX
206  02CD   D0   F8                  BNE     NULL
207  02CF   68                       PLA                   ; RESTORE X
208  02D0   AA                       TAX
209  02D1   68                       PLA                   ; RESTORE ACC
210  02D2   60                       RTS                   ; RTS X = X  Y = Y  A = A
213       0000                       . END
```

©

24 Hour Clock for SYM-1 BASIC

A. M. Mackay
600 Sixth Avenue West,
Owen Sound, Ontario, Canada
N4K 5E7

Load this program in your SYM-1 and enter G 0FB8. It will start a clock, display the memory you should enter to protect the program, then automatically transfer you to BASIC. Be sure to enter the amount of memory shown on the CRT. If you don't, you will lose the program.

The clock sits at the very top of your memory, and will not interfere with your BASIC - in fact, you won't even know it's there unless you call it. But it will be there when you want it for control operations, time delays for games, or whatever. You can even use your CRT as a time-of-day clock. It will keep ticking away until you hit "reset" or turn your SYM off.

As written, it is a 24 hour clock. If you want a 12 hour clock, change the contents of location 0FAD to "C0".

This clock uses timer 1 of U29 to interrupt the program every 50 ms. "COUNT" totals these interrupts, and after each 20 interrupts one is added to "SECS". Timer 2 and the input ports of U29 are not affected, and can be used for other purposes.

The program is written for a SYM-1 with 4K memory. It can be relocated upwards by changing all the "0F" bytes to "1F" or whatever. Also, change the bytes in locations 0FF9 to 0FFC. For example, for 8K, change these bytes to 37 35 34 38. The formula is 131 less than the number of free bytes usually displayed after going to BASIC, with a "3" before each digit.

The program as written keeps the time, starting at 0h, 0m, 0s from the time you enter G 0FB8. If you prefer time of day, enter the following (for 4K):

POKE &"0F7D", hour (in 24 hour time), CR
POKE &"0F7E", minute, CR
POKE &"0F7F", second, then hit CR on the exact second.

To use the clock in a BASIC program, enter the following BASIC command:

```
10 H = PEEK(&"0F7D"):M = PEEK(&"0F7E")
:S = PEEK(&"0F7F")
```

Then use the variables H, M and S as you require them. To use your CRT as a clock, use the following program:

```
10 - as in the paragraph above.
20 PRINT CHR$(12)
30 PRINT H; ":";M;":";S
40 FOR X  =1 TO 785: NEXT
50 GOTO 10
60 END
```

If you can turn cursor off, the clock will look much better.

The actual clock program ends at 0FDE. The rest of the program puts out the memory requirements on the CRT. If you don't want to transfer to BASIC, or if you don't have a CRT, put 4C 00 80 in locations 0FDF - 0FE1. You can then access the time by going to M 0F7D (hour), 0F7E (mins) and 0F7F (secs).

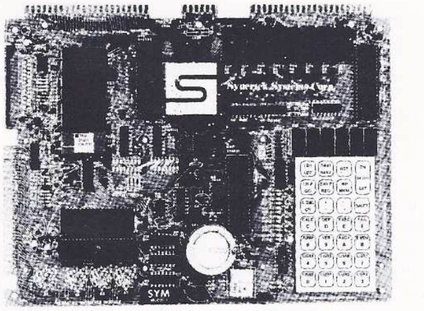
If anyone wants it, perhaps I'll write a routine for a future issue that will display the time on the SYM-1 LED readouts.

```
ØØ1Ø            .BA $ØF7C    ;THIS PROGRAM STARTS A
ØØ2Ø ACCESS     .DE $8B86    ;24 HOUR CLOCK AND
ØØ3Ø OUTVEC     .DE $A663    ;DISPLAYS THE AMOUNT OF
ØØ4Ø IRQVEC     .DE $A67E    ;REMAINING MEMORY TO BE
ØØ5Ø CLRINT     .DE $ACØ4    ;ENTERED WHEN REQUESTED
ØØ6Ø T1CH       .DE $ACØ5    ;THEN AUTOMATICALLY
ØØ7Ø T1LL       .DE $ACØ6    ;TRANFERS TO BASIC.
```

SYM-1, 6502-BASED MICROCOMPUTER

- FULLY-ASSEMBLED AND COMPLETELY INTEGRATED SYSTEM that's ready-to-use
- ALL LSI IC'S ARE IN SOCKETS
- 28 DOUBLE-FUNCTION KEYPAD INCLUDING UP TO 24 "SPECIAL" FUNCTIONS
- EASY-TO-VIEW 6-DIGIT HEX LED DISPLAY
- KIM-1* HARDWARE COMPATIBILITY

  The powerful 6502 8-Bit MICROPROCESSOR whose advanced architectural features have made it one of the largest selling "micros" on the market today.
- THREE ON-BOARD PROGRAMMABLE INTERVAL TIMERS available to the user, expandable to five on-board.
- 4K BYTE ROM RESIDENT MONITOR and Operating Programs.
- Single 5 Volt power supply is all that is required.
- 1K BYTES OF 2114 STATIC RAM onboard with sockets provided for immediate expansion to 4K bytes onboard, with total memory expansion to 65, 536 bytes.
- USER PROM/ROM: The system is equipped with 3 PROM/ROM expansion sockets for 2316/2332 ROMs or 2716 EPROMs
- ENHANCED SOFTWARE with simplified user interface
- STANDARD INTERFACES INCLUDE:
  - —Audio Cassette Recorder Interface with Remote Control (Two modes: 135 Baud KIM-1* compatible, Hi-Speed 1500 Baud)
  - —Full duplex 20mA Teletype Interface
  - —System Expansion Bus Interface
  - —TV Controller Board Interface
  - —CRT Compatible Interface (RS-232)
- APPLICATION PORT: 15 Bi-directional TTL Lines for user applications with expansion capability for added lines
- EXPANSION PORT FOR ADD-ON MODULES (51 I/O Lines included in the basic system)
- SEPARATE POWER SUPPLY connector for easy disconnect of the d-c power
- AUDIBLE RESPONSE KEYPAD



Synertek has enhanced KIM-1* software as well as the hardware. The software has simplified the user interface. The basic SYM-1 system is programmed in machine language. Monitor status is easily accessible, and the monitor gives the keypad user the same full functional capability of the TTY user. The SYM-1 has everything the KIM-1* has to offer, plus so much more that we cannot begin to tell you here. So, if you want to know more, the SYM-1 User Manual is available, separately.

SYM-1 Complete w/manuals	**$229.00**
SYM-1 User Manual Only	**7.00**
SYM-1 Expansion Kit	**60.00**

Expansion includes 3K of 2114 RAM chips and 1-6522 I/O chip.

SYM-1 Manuals: The well organized documentation package is complete and easy-to-understand.

SYM-1 CAN GROW AS YOU GROW. It's the system to BUILD-ON. Expansion features that are available:

BAS-1 8K Basic ROM (Microsoft)	**$ 89.00**
KTM-2 (Complete terminal less monitor)	**319.00**

QUALITY EXPANSION BOARDS DESIGNED SPECIFICALLY FOR KIM-1, SYM-1 & AIM 65

These boards are set up for use with a regulated power supply such as the one below, but, provisions have been made so that you can add onboard regulators for use with an unregulated power supply. But, because of unreliability, we do not recommend the use of onboard regulators. All I.C.'s are socketed for ease of maintenance. *All boards carry full 90-day warranty.*

All products that we manufacture are designed to meet or exceed industrial standards. All components are first qualtiy and meet full manufacturer's specifications. All this and an extended burn-in is done to reduce the normal percentage of field failures by up to 75%. To you, this means the chance of inconvenience and lost time due to a failure is very rare; but, if it should happen, we guarantee a turn-around time of less than forty-eight hours for repair.

Our money back guarantee: If, for any reason you wish to return any board that you have purchased directly from us within ten (10) days after receipt, complete, in original condition, and in original shipping carton; we will give you a complete credit or refund less a $10.00 restocking charge per board.

VAK-1 8-SLOT MOTHERBOARD

This motherboard uses the KIM-4* bus structure. It provides eight (8) expansion board sockets with rigid card cage. Separate jacks for audio cassette, TTY and power supply are provided. Fully buffered bus.

VAK-1 Motherboard	**$129.00**

VAK-2/4 16K STATIC RAM BOARD

This board using 2114 RAMs is configured in two (2) separately addressable 8K blocks with individual write-protect switches.

VAK-2 16K RAM Board with only 8K of RAM (½ populated)	**$239.00**
VAK-3 Complete set of chips to expand above board to 16K	**125.00**
VAK-4 Fully populated 16K RAM	**325.00**

VAK-5 2708 EPROM PROGRAMMER

This board requires a +5 VDC and ±12 VDC, but has a DC to DC multiplyer so there is no need for an additional power supply. All software is resident in on-board ROM, and has a zero-insertion socket.

VAK-5 EPROM Programmer w/2708 adapter	**$249.00**
VAK-5A Single voltage 2716 adapter	**45.00**

VAK-6 EPROM BOARD

This board will hold 8K of 2708 or 2758, or 16K of 2716 or 2516 EPROMs. *EPROMs not included.*

VAK-6 EPROM Board	**$119.00**

VAK-7 COMPLETE FLOPPY-DISK SYSTEM (Oct '79)

VAK-8 PROTYPING BOARD

This board allows you to create your own interfaces to plug into the motherboard. Etched circuitry is provided for regulators, address and data bus drivers; with a large area for either wire-wrapped or soldered IC circuitry.

VAK-8 Protyping Board	**$39.00**

POWER SUPPLIES

ALL POWER SUPPLIES are totally enclosed with grounded enclosures for safety, AC power cord, and carry a full 2-year warranty.

FULL SYSTEM POWER SUPPLIES

This power supply will handle a microcomputer and up to 65K of our VAK-4 RAM. ADDITIONAL FEATURES ARE: Over voltage Protection on 5 volts, fused, AC on/off switch. Equivalent to units selling for $225.00 or more.

Provides +5 VDC @ 10 Amps & ±12 VDC @ 1 Amp

VAK-EPS Power Supply	**$119.00**
VAK-EPS/AIM provides the same as VAK-EPS plus 24V unreg.	**149.00**

KIM-1* Custom P.S. provides 5 VDC @ 1.2 Amps and +12 VDC @ .1 Amps

KCP-1 Power Supply	**$39.00**

SYM-1 Custom P.S. provides 5 VDC @ 1.4 Amps

VCP-1 Power Supply	**$39.00**

*KIM is a product of MOS Technology

RNB ENTERPRISES
INCORPORATED

2967 W. Fairmount Avenue
Phoenix AZ 85017
(602)265-7564

master charge VISA

Add $2.50 for shipping and handling per order.

```
                    0080 ACR        .DE $AC0B     ;TIME IS NOT DISPLAYED UNTIL
                    0090 IFR        .DE $AC0D     ;REQUESTED BY BASIC PROGRAM.
                    0100 IER        .DE $AC0E
                    0110 BASIC      .DE $C000
0F7C- 14            0120 COUNT      .BY $14       ;SET COUNT TO 20
0F7D- 00            0130 HOUR       .BY $00       ;START TIME AT 00 HOURS
0F7E- 00            0140 MINS       .BY $00       ;              00 MINUTES
0F7F- 00            0150 SECS       .BY $00       ;              00 SECONDS
                    0170 ;***INTERRUPT ROUTINE***
0F80- 48            0190 CLOCK      PHA
0F81- CE 7C 0F      0200            DEC COUNT     ;SEE IF 1 SEC HAS PASSED
0F84- D0 2D         0210            BNE EXIT      ;IF NO, EXIT
0F86- A9 14         0220            LDA #20       ;IF YES, -
0F88- 8D 7C 0F      0230            STA COUNT     ; RESTORE COUNT AND
0F8B- EE 7F 0F      0240            INC SECS      ; ADD 1 TO SECS.
0F8E- AD 7F 0F      0250            LDA SECS      ;SEE IF 60 SECS HAS PASSED
0F91- 38            0260            SEC
0F92- E9 3C         0270            SBC #60
0F94- D0 1D         0280            BNE EXIT      ;IF NO, EXIT
0F96- 8D 7F 0F      0290            STA SECS      ;IF YES RESET SECS TO 0 AND
0F99- EE 7E 0F      0300            INC MINS      ;ADD 1 TO MINS
0F9C- AD 7E 0F      0310            LDA MINS      ;SEE IF 60 MINS HAS PASSED
0F9F- E9 3C         0320            SBC #60
0FA1- D0 10         0330            BNE EXIT      ;IF NO, EXIT
0FA3- 8D 7E 0F      0340            STA MINS      ;IF YES RESET MINS TO 0 AND
0FA6- EE 7D 0F      0350            INC HOUR      ;ADD 1 TO HOUR
0FA9- AD 7D 0F      0360            LDA HOUR      ;SEE IF 24 HOURS HAS PASSED
0FAC- E9 18         0370            SBC #24       ;IF NO,
0FAE- D0 03         0380            BNE EXIT      ;   EXIT
0FB0- 8D 7D 0F      0390            STA HOUR      ;IF YES, RESET HOUR TO 0
0FB3- AD 04 AC      0400 EXIT       LDA CLRINT    ;ENABLE TIMER INTERRUPT
0FB6- 68            0410            PLA
0FB7- 40            0420            RTI
                    0440 ;***INITIATE TIMER***
0FB8- 20 86 8B      0460 START      JSR ACCESS    ;DISABLE WRITE PROTECT SYS. RAM
0FBB- A9 80         0470            LDA #$80      ;SET IRQ
0FBD- 8D 7E A6      0480            STA IRQVEC    ;  VECTOR
0FC0- A9 0F         0490            LDA #$0F      ;   TO
0FC2- 8D 7F A6      0500            STA IRQVEC+$1       ;  "CLOCK"
0FC5- A9 C0         0510            LDA #$C0      ;SET BITS 6 AND 7
0FC7- 8D 0B AC      0520            STA ACR       ;FOR FREE-RUNNING MODE
0FCA- 8D 0E AC      0530            STA IER       ;AND T1 INTERRUPT ENABLE
0FCD- AD 0D AC      0540            LDA IFR       ;CLEAR T1 FLAG BIT 6 BUT
0FD0- 29 BF         0550            AND #$BF      ;DON'T DISTURB OTHER
0FD2- 8D 0D AC      0560            STA IFR       ;IFR BITS
0FD5- A9 50         0570            LDA #$50      ;SET
0FD7- 8D 06 AC      0580            STA T1LL      ; TIMER
0FDA- A9 C3         0590            LDA #$C3      ;  FOR 1/20 SEC AND
0FDC- 8D 05 AC      0600            STA T1CH      ;   START TIMER
                    0620 ;***BASIC MEMORY OUT***
0FDF- A2 00         0640 DISPMEM    LDX #$00      ;START OF
0FE1- BD F2 0F      0650 OUTMEM     LDA TABLE,X   ; ROUTINE
0FE4- 20 63 A6      0660            JSR OUTVEC ;
0FE7- E8            0670            INX  ;          TO
0FE8- C9 04         0680            CMP #$04 ;
0FEA- F0 03         0690            BEQ GOBAS     ; DISPLAY
0FEC- 4C E1 0F      0700            JMP OUTMEM    ; REMAINING
0FEF- 4C 00 C0      0710 GOBAS      JMP BASIC     ; MEMORY
0FF2- 0D 0A 4D      0720 TABLE      .BY $0D $0A 'MEM. 3965' $0D $0A $04
0FF5- 45 4D 2E
0FF8- 20 33 39
0FFB- 36 35 0D
0FFE- 0A 04
                    0730            .EN  ;END OF PROGRAM
```

©

SCREEN CLEAR ROUTINES FOR THE OSI C1P

Charles L. Stanford
2903 Georgetown Road
Cinnaminson, NJ, 08077

Every OSI user eventually reaches a level of programming skill which requires an effective screen clear routine, whether for graphics, charts and graphs, or curve plotting. But many of the available programs in BASIC leave a lot to be desired. This article will demonstrate the more common screen clear methods, both BASIC and BASIC-addressed machine language, explain some of the advantages and disadvantages of each, and finally, list a really efficient subroutine.

The basic BASIC screen clear is, of course, "10 FOR S = 0 TO 31:PRINT:NEXT". It is simple and easy to remember. But, the scrolling is slow and distracting, and it won't clear POKE's below the cursor unless you add "20 FOR S = 54144 TO 54271:POKE S,32:NEXT". That's even slower and more distracting.

Another common BASIC clear is "10 FOR S = 53248 TO 54271: POKE S,32:NEXT". But you sure can't be in a hurry!

OSI, in their wisdom, has listed a USR function machine language program in the Graphics Reference Manual which really works pretty well. I have shown a slightly altered version of it for 8K machines in Program Listing I, lines 50000 through 50050. Unfortunately, it has some significant drawbacks. First, it is "address sensitive". If you examine Program Listing II, you will note that the instruction at address $1FFC performs a "Store Accumulator" on address $1FF0. Since you will want to store the routine at top of RAM to save program storage space, the program must be changed to reflect your 4, 8, 12, or more K of RAM. The POKE's to memory locations #11 and 12 of the USR start address, as well as the addresses in line 50020, must also be changed to match the storage location. In addition, the program must be protected by limiting memory on Cold Start, or string storage will wipe out the USR program (or vice versa).

And finally, it uses two substantial chunks of RAM, one for the BASIC program, and one for the machine language program.

There is a way around the protection requirement, which I only recently verified through a User Group newsletter; there is a unit of RAM in page 3, from about $0222 to $02FA, which is not addressed by any of the ROM functions. Thus, you can store a machine language routine there and eliminate two problems: excessive use of storage memory, and the need to protect upper RAM. Unfortunately, you still have the disadvantage of keying in a long and error prone sequence of loops and DATA.

So why not use one of the several routines in either the BASIC or Utility ROMs? I have found three so far, and I'll bet there are more. The problem is, they are routines. They are not subroutines. So if you jump to them, they will clear the screen, but they won't return to your program. However, there's a way around that. If you closely examine the two routines starting at 51000 and 52000 in Program Listing I, you will see that they are merely copies from ROM to RAM, then a $60 (#96) or RTS instruction is appended.

I have shown both routines in the demonstration program and in Program Listings III and IV only to demonstrate the different approaches to machine language programming. Firstly, these two are completely independent of location. The one at $FE0C uses "Zero Page" addressing to increment locations. The other one, at $FF24, uses Indirect, Y addressing with four successive pages to cover the 1K of screen refresh RAM.

In addition, there is a great difference in running time. The first takes only about 6400 machine cycles. (.0064 sec) while the second requires over 11,300 (.0113 sec). While we're not really in that much of a hurry, it demonstrates the efficiencies which can be achieved by different programming styles.

On the minus side, the program at line 51000 starts with the Y register at 0 only when addressed by the Monitor. So we have to add "POKE 546,160: POKE 547,0" to set this register equal to zero. To verify this, change the POKEs to "POKE 546,234: POKE 547,234" which substitutes NOP NOP for STY 00. This will very graphically prove that the Y register is **not** 0.

Because of this, the routine at line 52000 is probably preferable as it uses no BASIC storage RAM for Machine Language storage, only 63 Bytes of RAM for BASIC storage, and is relatively easy to remember and enter.

I would be interested in hearing from readers with other ideas or suggestions for screen clear, screen POKE, etc., as I am currently drafting an article on Machine Language Graphics subroutines for BASIC game programs.

Program Listing I Machine language demonstration program

```
   1Ø REM-MACH SCR CLR DEMO
   2Ø REM-GOSUB 5ØØØØ OR 51ØØØ OR 52ØØØ
   3Ø REM-USE THIS LINE FOR GOSUB
   4Ø FOR X = 53248 TO 54271: POKE x,187: NEXT
   5Ø X = USR(X)
   6Ø STOP
5ØØØØ REM-POOR ROUTINE-LIM MEM 8167
5ØØ1Ø RESTORE: POKE 11,232: POKE 12,31
5ØØ2Ø FOR X = 8168 TO 8191: READ M: POKE X,M: NEXT X
5ØØ3Ø DATA 169,32,16Ø,4,162,Ø,157,Ø,2Ø8,232,2Ø8,25Ø,238
5ØØ4Ø DATA 24Ø,31,136,2Ø8,244,169,2Ø8,141,24Ø,31,96
5ØØ5Ø RETURN
51ØØØ GOOD ROUTINE
51Ø1Ø POKE 11,34: POKE 12,2: POKE 546,16Ø: POKE547,Ø: POKE 565,96
51Ø2Ø FOR X = Ø TO 16: Y = PEEK(65316 + X): POKE548 + X, Y: NEXT
51Ø3Ø RETURN
52ØØØ REM-BETTER ROUTINE
52Ø1Ø POKE 11,34: POKE 12,2: POKE 574,96
52Ø2Ø FOR X = Ø TO 27: Y = PEEK(65Ø36 + X): POKE 546 + X, Y: NEXT
52Ø3Ø RETURN
```

Program Listing II OSI Graphics Manual Screen Clear

```
$1FE8  A9 2Ø      #169  32      LDA,IMM   $2Ø
 1FEA  AØ Ø4       16Ø   4      LDY,IMM   $Ø4
 1FEC  A2 ØØ       162   Ø      LDX,IMM   $ØØ
 1FEE  9D ØØ DØ    157   Ø 2Ø8  STA,ABS-X
 1FF1  E8          232          INX
 1FF2  DØ FA       2Ø8 25Ø      BNE
 1FF4  EE FØ 1F    238 24Ø  31  INC
 1FF7  88          136          DEY
 1FF8  DØ F4       2Ø8 244      BNE
 1FFA  A9 DØ       169 2Ø8      LDA,IMM   $DØ
 1FFC  8D FØ 1F    141 24Ø  31  STA,ABS
 1FFF  6Ø           96          RTS
```

Program Listing III First ROM Monitor Routine

```
$XXXX  AØ ØØ      #16Ø   Ø      LDY,IMM   $ØØ
 FF24  A9 2Ø       169  32      LDA,IMM   $2Ø
 FF26  99 ØØ D3    153   Ø 211  STA,Y
 FF29  99 ØØ D2    153   Ø 21Ø  STA,Y
 FF2C  99 ØØ D1    153   Ø 2Ø9  STA,Y
 FF2F  99 ØØ DØ    153   Ø 2Ø8  STA,Y
 FF32  C8          2ØØ          INY
 FF33  DØ F1       2Ø8 241      BNE
 XXXX  6Ø           96          RTS
```

Program Listing IV Second ROM Monitor Routine

```
$FEØC  A2 D4       162 212      LDX,IMM   $D4
 FEØE  A9 DØ       169 2Ø8      LDA,IMM   $DØ
 FE1Ø  85 FF       133 255      STA,Z
 FE12  A9 ØØ       169   Ø      LDA,IMM   $ØØ
 FE14  85 FE       133 254      STA,Z
 FE16  85 FB       133 251      STA,Z
 FE18  A8          168          TAY
 FE19  A9 2Ø       169  32      LDA,IMM   $2Ø
 FE1B  91 FE       145 254      STA,I,Y
 FE1D  C8          2ØØ          INY
 FE1E  DØ FB       2Ø8 251      BNE
 FE2Ø  E6 FF       23Ø 255      INC,Z
 FE22  E4 FF       228 255      CPX,Z
 FE24  DØ F5       2Ø8 245      BNE
 FE26  84 FF       132 255      STY,Z
 XXXX  6Ø           96          RTS
```

©

Personal Computer Breakthrough

The Ohio Scientific Challenger 1 is a Dramatic Demonstration of Price and Performance in a Single Package

Announcing the microcomputer that scored a dramatic breakthrough in price and performance at a terrific system price!

Price/Performance Leader

The development of the Ohio Scientific Challenger 1P marked a price revolution for microcomputers. Consider the following features:

(1) Microsoft 8K Basic-in-ROM
(2) 4K RAM--Expandable to 8K on board
(3) Full 53 Key keyboard with upper and lower case
(4) Elaborate graphic display capability
(5) Uses a standard cassette for input
(6) Uses a standard television with an RF modulator for video output

All of these features makes the Challenger 1P a best buy for a beginner or hobbyist on a limited budget wishing to get involved with microcomputer, such as students & their educators, and various other professionals.

Already Looking Ahead??

Of course, we are not beginners forever (hopefully!!). What happens to the C1P then? Is it obsolete? That answer is a definite no!!

Consider the specification facts on the **expandability** of the Challenger 1P:

(1) A fully expanded Challenger 1P can support:
(a) Basic-in-ROM
(b) 32K of RAM memory
(c) Dual Mini-floppies
(d) cassette
(e) printer
(f) modem
(g) full BUS expansion capability via the OSI 48 line BUS through which over 40 accessories can be added (A/D,D/A, voice output, I/O re memory, etc.)

As you can see, the Challenger 1P leaves more than adequate room for expansion. Thus, the system is not one you will have to 'get rid' of as you progress to more sophisticated computer functions.

Challenger 1P with Disk Capabilities

If you are familiar enough with computers to realize the importance of disk storage, then the Challenger 1P Mini Floppy is a prime consideration.

At $995, it is the first microcomputer system to offer disk storage capacility under $1000, another breakthrough. The C1P MF offers the advantage of using either Basic-in-Rom or Pico-Dos, a compact DOS. The unit comes standard with 12K of RAM.

Additional memory can be added anytime by simply plugging in additional RAM into the socketed memory board. Upon adding 8K additional RAM to get to 20K, you may run OS65DV3.1, a full-fledged DOS allowing named program and data files as well as random and sequential access capabilities to these files.

Why System Packages?

Microcomputers are sophisticated devices that require more than the initial purchase. Once the purchase is made, a quick second, third, and even fourth trip is necessary to secure the accessories necessary to operate properly. To save you the additional trips, we have made two system packages available to the public at great savings.

System Package #1: Challenger 1P

(1) Challenger 1P w/4KRAM	349.00
(2) Additional 4K RAM(installed)	69.00
(3) Challenger 1P workbook	7.00
(4) Howard Sams schematics	8.00
(5) 3 Software tapes	22.00
(6) 10 blank data cassettes	15.00
Total List price	470.00
CCSB System 1 Price	**399.00**

System Package #2: Challenger 1P MF

(1) Challenger 1P MF w/12K RAM	995.00
(2) Additional 8K RAM(installed)	138.00
(3) Challenger 1P workbook	7.00
(4) Howard Sams schematics	8.00
(5) 0S65D V3.1 operating system	50.00
(6) 3 Software Tapes	22.00
(7) 10 blank diskettes	35.00
Total List Price	1255.00
CCSB System 2 Price	**1088.00**

You will notice the system price reflects a **15% savings** over typical list price.

Interested??

Here are the specifics for ordering the systems:

(1) Freight and insurance are added to the cost of each package. Add $6 for #1 and $9 for #2 for delivery anywhere in the continental US. Overseas deliveries are available at a higher freight rate.

(2) Funds transfers are handled easiest by money order or certified check, as this insures the quickest turnaround. Personal checks require 10-14 days to clear. Also, Visa & Mastercharge are accepted for your convenience.

(3) All deliveries are made via UPS.

(4) Warranty: A standard 60-day parts and labor warranty is in effect when purchased. After this period, parts are guaranteed for 1 year from purchase. Service work is done in-house whenever possible for a quick turnaround.

(5) Want more information? Write for our OSI catalogs. Enclose $2.00 for brochures & handling.

COMPUTER CENTER OF SOUTH BEND

51591 US 31 NORTH SOUTH BEND IN 46637 (219) 277-4655

Store Hours Mon-Saturday 12-8

Machine Language Tapes for OSI Challengers

Daniel Schwartz
2160 Matthews Ave. Apt. 7P
Bronx, N.Y. 10462

When I first saw COMPUTE on the stand at my local computer store, I thought to myself "What? Another new computer magazine? Aren't there enough of them already?" The sub-title calling it "The PET Gazette with a new name" really confused me, since the store in question doesn't even carry the Commodore PET. But when I opened the magazine I was quite pleasantly surprised.

While I liked the whole magazine, what pleased me in particular was that there were actually four whole pages devoted to the system that I own, the OSI C1P. Not that those four pages contained any fantastic new information, but I had pretty much despaired of ever finding anything in a major magazine for Ohio Scientific users. Everything always seemed to be for either the TRS-80, the Apple II, or the PET (although I have found that most PET programs will run on the Challenger with minor modifications).

Anyway I would like to share with your readers a potentially useful program I have written, along with some other information I have found about my machine, in the hope that a continuing exchange of such information can be established in your magazine.

Specifically, this article concerns a point raised by W. Keith Russell in his review of the C1 -- the lack of a method for storing machine-language programs on tape. I have had my computer for about five months now, and have been dabbling with machine language for the past two. I have written a program in BASIC which will take a machine language program entered through the monitor and save it on tape in a format which can again be entered through the monitor using the "L" command.

First, I'll give the program listing (it's short), and then discuss how it works.

```
1Ø SAVE:PRINT ".Ø222/";
2Ø FOR A=546 TO 63Ø
3Ø OP=PEEK(A)
4Ø H=INT(OP/16):L=OP-16*H
5Ø H=H+48-7*(H>9):L=L+48-7*(L>9)
6Ø WAIT 6144Ø,2:POKE 61441,H
7Ø WAIT 6144Ø,2:POKE 61441,L
8Ø WAIT 6144Ø,2:POKE 61441,13
9Ø NEXT A
1ØØ PRINT ".Ø222G"
```

This program, and all procedures given in this article, should work equally well on the C1P or the C2-4P, except where noted.

The hex address in line 10 is the address of the first byte to be saved. Note that the string to be printed begins with a period to put the monitor into address mode, ends with a slash to put the monitor in data mode, and is followed by a semicolon to supress a carriage return. 0222 is a good address to start at if the program is short enough to fit in the space from 0222 to 02FF, because then it can run without interfering with BASIC, and there will be no need to "reserve" special space for it. If the program is too long for this, start it at 0400 if it will run on its own. In the event that the program is to be called from BASIC but is too long for the 0222-02FF niche, you must reserve space for it when cold starting BASIC, and the program should start at the address which is numerically equal to the answer you give for "MEMORY SIZE". (More on how to call machine programs from BASIC later). The BASIC program to generate the machine language tape will reside in 0300-03FF.

In line 20, the lower limit of the FOR-NEXT loop is the same starting address as in line 10, but this time expressed in decimal. The upper limit of the loop is of course the last byte of the program to be saved. The value 630 is given only as a typical example.

Line 30 gets the contents of the memory location to be saved, and line 40 breaks it down into high and low four-bit hex digits.

Line 50 converts the four-bit numbers H and L into the corresponding ASCII characters 0-9 or A-F. Note the use of the logical expressions H>9 and L>9 as if they were numerical values. This is a convenient feature of BASIC which is not mentioned in any "official" documentation. A true expression has the value of -1 in OSI BASIC (rather than a value of 1 as you might expect), and a false expression has a value of zero. Thus, if H-9, the value of H>9 is 0, and H + 48-7*(H>9) = 9 + 48 = 57, which is the ASCII code for "9". If H = 10, then the value of H>9 is -1, and H + 48-7* (H>9) = 10 + 48-)-7) = 65, which is the ASCII code for "A".

Lines 60-80 output the high-order character H, the low order character L, and carriage return (ASCII)13) to the cassette port, as required by the 65V monitor. The cassette port is located at hex F000 and F001, which is decimal 61440 and 61441. 61440 is the "status" byte: a one or high in the ones-bit means that a character can now be received (a zero or low here means that a character is in the process of being received and is not yet ready); a high in the twos-bit position means that a character can be sent (a low here means that a character is in the process of being sent, so don't send another one yet). The WAIT statements in lines 60-80 wait for this second bit to go high, indicating that the previous character has been completed, before sending the next character. Sending the charater is accomplished by simply POKEing location 61441, the "data" byte of the port, with the ASCII value to be sent.

The same location address, F001 or 61441, is also used to receive data from the cassette port. However, the input and output locations are actually wholly separate registers (its like the keyboard at 57088 in this respect - the value PEEKed will not match the value POKEd.) Note: the values given here as the port locations are for the C1P (this section is supposed to be for single board computers, remember?) I am not certain, but I believe that on the C2-4P the locations are FC00 and FC01, which would be 64512 and 64513 in decimal. If so (check the memory map in your manual), just change 61440 and 61441 in the program to 64512 and 64513, respectively. The rest stays the same.

After line 80, we have finished recording one memory byte on tape and prepared the computer to accept the next one. Line 90 loops back to get the next byte, and the process continues.

When the last byte specified in line 20 has been output, we end up at line 100. This line starts the program going with the "G" command. The hex address given here as the execution address may or may not be the same as the address of the first recorded byte given in line 10. In my example, it is. Again, note that the address is preceded by a period to put the monitor into address mode. If the program is one which is going to be called as a USR routine from BASIC, you can use A274 as the execution address. That is the warm-start address for BASIC, and will therefore automatically warm-start BASIC for you at the end of the load (to cold-start, use BD11).

That's about it, really. To load the tape, enter the Monitor, turn on the tape, and press "L" when the leader tape has passed over the heads. (pressing L too soon could cause the first few bytes of page zero, which contain the jump vector for warm-starting BASIC, to be destroyed, but even this is not too serious. If it does happen, just set bytes 0,1 and 2 back to 4C/74/A2 by hand if you want to use the "W" key to warm-start.)

Now, a few words about calling machine-language routines from BASIC. As you have surely heard, this is done with the USR function. When BASIC encounters a statement containing the USR function, such as X = USR(X), it goes to execute a machine language routine until an RTS instruction is encountered, at which point you are returned to BASIC. What isn't made clear by OSI is how to set up the system to call a routine at a specific memory address. While I'm sure some readers have figured this out, I'm equally sure that others haven't, so here is a brief run-down.

If you try to call a USR function without first setting up the USR "dispatcher", or vector, all you will get will be an "F/ ERROR" message. This is a protective feature to save the system from a crash if a jump were made to a random location. When BASIC is cold started, the USR vector is pre-set to call the routine inside of the BASIC interpreter which outputs the "F/ ERROR" message.

The USR vector or "dispatcher" is located in locations 11(low) and 12(high) or 000B and 000C in hex. These two locations specify the memory location at which the USR routine begins. For example, to call a routine starting at 0222 (hex), you would first execute the statements POKE 11,34 and POKE 12,2. (or you could set 000B and 000C to 22 and 02 via the monitor.) Then each time the USR function was encountered by BASIC, the routine starting at 0222 would be executed. If there was more than one routine that you wanted to call from the same BASIC program, you would simply need to POKE the proper low and high addresses into locations 11 and 12 each time before calling the routine with X = USR(X).

Finally, there is a way that you can make use of all this even if you don't know the first thing about machine language. By far the most common use of USR in my programs is to call the polled keyboard routine in ROM in order to accomplish a single-character input statement, similar to the GET statement on the PET or INKEY$ on the TRS-80. To do this, the beginning of your BASIC program should contain the line POKE 11,0:POKE 12,253. This sets up USR to call the polled keyboard routine. Then, to input a character, you use the statement

X = USR(X) :I = PEEK(531) or X = USR(X)
:I$ = CHR$(PEEK(531))

When one of these statements is encountered, the program will stop and wait for a key to be pressed. As soon as a key is pressed, the ASCII value of that key will be placed in location 531 and BASIC will resume. The second half of the statement then places this value in either the numeric variable I or the string variable I$. It can then be printed without waiting for a carriage return, or it can be checked for legality in the situation, printed if legal, and if not a loop can be made back to the USR statement to wait for a legal character. For example, In a yes/no situation, you could accept only "Y" and "N" as legal, rejecting anything else. You could then avoid the need to output something like "WHAT? ANSWER YES OR NO!!" in the case where someone gives an ambiguous answer, or to take something like "GLOP" as meaning "NO". I am sure you can think of many other uses for this function. ©

Songs in the Key of KIM

George W. Hawkins
200 Old Country Road
Melville, NY 11746

I was fascinated by Richard Martin's Four part Harmony (Cheap) program in issue 16 and decided to try coding some other songs. I have made no attempt to code to conserve memory so far. Note that I have 2K of additional memory on my KIM-1 starting at hex 0400. The tempo and note addresses for my four songs are:

Song	0007-000F
Jingle Bells	15 FF 01 97 03 8F 04 87 05
Deck the Halls	13 FF 01 15 03 23 04 31 05
Shangri-La (only 2 parts)	26 FF 01 FF 01 D5 02 D5 02
Love is Blue	26 FF 01 AD 03 BB 04 C9 05

Jingle Bells

```
200   5B 0F 00 01 36 0F 00 01 3C 0F 00 01 44 0F 00 01
210   5B 2F 00 01 5B 07 00 01 5B 07 00 01 5B 0F 00 01
220   36 0F 00 01 3C 0F 00 01 44 0F 00 01 51 2F 00 01
230   51 0F 00 01 51 0F 00 01 33 0F 00 01 36 0F 00 01
240   3C 0F 00 01 48 2F 00 01 2D 0F 00 01 28 0F 00 01
250   2D 0F 00 01 33 0F 00 01 3C 0F 00 01 36 2F 00 01
260   5B 0F 00 01 5B 0F 00 01 36 0F 00 01 3C 0F 00 01
270   44 0F 00 01 5B 2F 00 01 5B 0F 00 01 5B 0F 00 01
280   36 0F 00 01 3C 0F 00 01 44 0F 00 01 51 2F 00 01
290   51 0F 00 01 51 0F 00 01 33 0F 00 01 36 0F 00 01
2A0   3C 0F 00 01 2D 0F 00 01 2D 0F 00 01 2D 0F 00 01
2B0   2D 0F 00 01 28 0F 00 01 2D 0F 00 01 33 0F 00 01
2C0   3C 0F 00 01 44 3F 00 01 36 0F 00 01 36 0F 00 01
2D0   36 1F 00 01 36 0F 00 01 36 0F 00 01 36 1F 00 01
2E0   36 0F 00 01 2D 0F 00 01 44 0F 00 01 3C 0F 00 01
2F0   36 3F 00 01 33 0F 00 01 33 0F 00 01 33 17 00 01

300   33 07 00 01 33 0F 00 01 36 0F 00 01 36 0F 00 01
310   36 07 00 01 36 07 00 01 36 0F 00 01 3C 0F 00 01
320   3C 0F 00 01 36 0F 00 01 3C 20 2D 1F 00 01 36 0F
330   00 01 36 0F 00 01 36 1F 00 01 36 0F 00 01 36 0F
340   00 01 36 1F 00 01 36 0F 00 01 2D 0F 00 01 44 0F
350   00 01 3C 0F 00 01 36 3F 00 01 33 0F 00 01 33 0F
360   00 01 33 17 00 01 33 07 00 01 33 0F 00 01 36 0F
370   00 01 36 0F 00 01 36 07 00 01 36 07 00 01 2D 0F
380   00 01 2D 0F 00 01 33 0F 00 01 3C 0F 00 01 44 30
390   00 40 00 40 00 C0 FF 01 00 10 36 10 00 10 36 10
3A0   00 10 36 10 00 10 36 10 00 10 36 10 00 10 36 10
3B0   00 10 33 10 00 10 33 10 00 10 33 10 00 10 33 10
3C0   00 10 33 10 00 10 33 10 00 10 33 10 00 10 33 10
3D0   00 10 36 10 00 30 36 10 00 10 36 10 00 10 36 10
3E0   00 10 36 10 00 10 36 10 00 10 36 10 00 10 33 10
3F0   00 10 33 10 00 10 33 10 00 10 33 10 00 10 33 10

400   00 10 33 0F 00 01 33 40 00 10 36 10 00 30 2D 10
410   00 10 28 10 00 10 24 10 00 10 28 10 00 10 2D 10
420   00 10 28 10 00 10 24 10 00 10 28 10 00 10 28 10
430   00 10 2D 10 00 10 2D 10 00 10 28 10 00 10 28 10
440   00 10 28 10 00 10 28 0F 00 01 33 10 00 20 2D 10
450   00 10 28 10 00 10 24 10 00 10 28 10 00 10 2D 10
460   00 10 28 10 00 10 24 10 00 10 28 10 00 10 28 10
470   00 10 2D 10 00 10 2D 10 00 10 30 0F 00 01 28 1F
480   00 01 2D 1F 00 01 2D 30 00 40 00 40 00 C0 97 03
490   00 10 44 10 00 10 44 10 00 10 44 10 00 10 44 10
4A0   00 10 44 10 00 10 44 10 00 10 44 10 00 10 44 10
4B0   00 10 3C 10 00 10 3C 10 00 10 51 10 00 10 51 10
4C0   00 10 51 10 00 10 51 10 00 10 44 10 00 30 44 10
4D0   00 10 44 10 00 10 44 10 00 10 44 10 00 10 44 10
4E0   00 10 44 10 00 10 44 10 00 10 44 10 00 10 3C 10
4F0   00 10 3C 10 00 10 51 10 00 10 51 0F 00 01 5B 40

500   00 10 44 10 00 30 36 10 00 10 36 10 00 10 36 10
510   00 10 36 10 00 10 36 10 00 10 36 10 00 10 36 10
520   00 10 36 10 00 10 33 10 00 10 33 10 00 10 36 10
530   00 10 36 10 00 10 36 10 00 10 30 10 00 10 33 0F
540   00 01 2D 10 00 20 36 10 00 10 36 10 00 10 36 10
550   00 10 36 10 00 10 36 10 00 10 36 10 00 10 36 10
560   00 10 36 10 00 10 33 10 00 10 33 10 00 10 36 10
570   00 10 36 0F 00 01 33 1F 00 01 33 1F 00 01 36 30
580   00 40 00 40 00 C0 8F 04 44 0F 00 01 5B 0F 00 01
590   5B 0F 00 01 5B 0F 00 01 44 0F 00 01 5B 0F 00 01
5A0   5B 0F 00 01 5B 0F 00 01 44 0F 00 01 5B 0F 00 01
5B0   44 0F 00 01 5B 0F 00 01 66 0F 00 01 51 0F 00 01
5C0   66 0F 00 01 51 0F 00 01 79 0F 00 01 51 0F 00 01
5D0   79 0F 00 01 51 0F 00 01 5B 0F 00 01 5B 0F 00 01
5E0   5B 0F 00 01 5B 0F 00 01 5B 0F 00 01 5B 0F 00 01
5F0   5B 0F 00 01 5B 0F 00 01 44 0F 00 01 5B 0F 00 01

600   44 10 00 10 44 0F 00 01 5B 0F 00 01 5B 0F 00 01
610   5B 0F 00 01 44 0F 00 01 5B 0F 00 01 5B 0F 00 01
620   5B 0F 00 01 44 0F 00 01 5B 0F 00 01 44 0F 00 01
630   5B 0F 00 01 66 0F 00 01 51 0F 00 01 66 0F 00 01
640   51 0F 00 01 79 0F 00 01 51 0F 00 01 79 0F 00 01
650   51 0F 00 01 5B 0F 00 01 5B 0F 00 01 5B 0F 00 01
660   5B 0F 00 01 51 1F 00 01 48 1F 00 01 44 0F 00 01
670   5B 0F 00 01 44 10 00 10 44 0F 00 01 44 0F 00 01
680   44 0F 00 01 44 0F 00 01 44 0F 00 01 44 0F 00 01
690   44 0F 00 01 44 0F 00 01 44 0F 00 01 44 0F 00 01
6A0   44 0F 00 01 44 0F 00 01 44 0F 00 01 44 0F 00 01
6B0   44 0F 00 01 44 0F 00 01 79 0F 00 01 44 0F 00 01
6C0   5B 0F 00 01 48 0F 00 01 44 0F 00 01 44 0F 00 01
6D0   51 0F 00 01 44 0F 00 01 79 0F 00 01 44 0F 00 01
6E0   79 0F 00 01 40 0F 00 01 5B 0F 00 01 44 0F 00 01
6F0   48 10 00 10 44 0F 00 01 44 0F 00 01 44 0F 00 01

700   44 0F 00 01 44 0F 00 01 44 0F 00 01 44 0F 00 01
710   44 0F 00 01 44 0F 00 01 44 0F 00 01 44 0F 00 01
720   44 0F 00 01 44 0F 00 01 44 0F 00 01 44 0F 00 01
730   44 0F 00 01 79 0F 00 01 44 0F 00 01 5B 0F 00 01
740   48 0F 00 01 44 0F 00 01 44 0F 00 01 51 0F 00 01
750   40 0F 00 01 44 1F 00 01 48 1F 00 01 44 30 00 40
760   00 40 00 C0 87 05
```

Shangri-La

```
200   00 10 51 10 4C 10 44 10 51 10 5B 30 00 10 51 10
210   4C 10 44 10 3C 40 00 10 51 10 4C 10 44 10 51 10
220   5B 1F 00 01 4C 10 3C 10 39 10 44 10 3C 10 39 10
230   33 10 30 18 33 08 33 40 33 0F 00 01 33 0F 00 01
240   33 0F 00 01 33 0F 00 01 33 10 2B 1F 00 01 33 0F
250   00 01 33 10 2B 1F 00 01 33 10 39 10 30 20 39 10
260   3C 2F 00 01 3C 0F 00 01 3C 10 33 1F 00 01 3C 0F
270   00 01 3C 10 33 1F 00 01 39 10 44 30 5B 10 3C 40
280   00 10 51 10 4C 10 44 10 51 10 5B 30 00 10 51 10
290   4C 10 44 10 3C 40 00 10 51 10 4C 10 44 10 51 10
2A0   5B 1F 00 01 4C 10 3C 10 39 10 44 10 3C 10 39 10
2B0   33 10 30 18 33 08 33 40 33 2F 00 01 5B 10 3C 10
2C0   39 10 44 10 3C 0F 00 01 39 10 33 10 30 10 2B 10
2D0   33 40 00 C0 FF 01 79 3F 00 01 66 2F 00 01 79 10
2E0   72 3F 00 01 4C 1F 00 01 51 1F 00 01 79 3F 00 01
2F0   66 3F 00 01 5B 1F 00 01 4C 1F 00 01 51 1F 00 01

300   40 1F 00 01 3C 40 3C 0F 00 01 3C 0F 00 01 3C 0F
310   00 01 3C 0F 00 01 39 3F 00 01 40 3F 00 01 40 3F
320   00 01 44 20 4C 1F 00 01 4C 3F 00 01 4C 3F 00 01
330   5B 2F 00 01 5B 0F 00 01 60 3F 00 01 79 3F 00 01
340   66 2F 00 01 79 10 72 3F 00 01 4C 1F 00 01 51 1F
350   00 01 79 3F 00 01 66 3F 00 01 5B 1F 00 01 4C 1F
360   00 01 51 1F 00 01 40 1F 00 01 44 40 51 30 00 10
370   5B 1F 00 01 4C 1F 00 01 51 1F 00 01 40 1F 00 01
380   3C 2F 00 01 3C 07 00 01 33 07 00 01 2D 0F 00 01
390   39 0F 00 01 48 0F 00 01 44 0F 00 31 D5 02
```

Love is Blue

```
200   36 1F 00 01 3C 17 00 01 44 07 00 01 48 0F 00 01
210   51 0F 00 01 5B 1F 00 01 51 0F 00 01 44 07 00 01
220   48 07 00 01 44 17 00 01 51 07 00 01 5B 07 00 01
230   51 07 00 01 5B 0F 00 01 6C 1F 00 01 36 1F 00 01
240   3C 17 00 01 44 07 00 01 48 0F 00 01 51 0F 00 01
250   5B 1F 00 01 51 0F 00 01 44 07 00 01 48 07 00 01
260   44 17 00 01 51 07 00 01 56 07 00 01 60 07 00 01
270   56 0F 00 01 51 20 00 40 36 1F 00 01 3C 17 00 01
280   44 07 00 01 48 0F 00 01 51 0F 00 01 5B 1F 00 01
290   51 0F 00 01 44 07 00 01 48 07 00 01 44 17 00 01
2A0   51 07 00 01 5B 07 00 01 51 07 00 01 5B 0F 00 01
2B0   6C 1F 00 01 36 1F 00 01 3C 17 00 01 44 07 00 01
2C0   48 0F 00 01 51 0F 00 01 5B 1F 00 01 51 0F 00 01
2D0   44 07 00 01 48 07 00 01 44 17 00 01 51 07 00 01
2E0   56 07 00 01 60 07 00 01 56 0F 00 01 51 20 00 40
2F0   40 1F 00 01 3C 1F 00 01 36 2F 00 01 40 07 00 01

300   51 07 00 01 51 1F 00 01 30 1F 00 01 36 3F 00 01
310   40 20 36 1F 00 01 48 2F 00 01 40 07 00 01 3C 07
320   00 01 3C 2F 00 01 48 07 00 01 56 07 00 01 51 30
330   00 10 36 1F 00 01 3C 17 00 01 44 07 00 01 48 0F
340   00 01 51 0F 00 01 5B 1F 00 01 51 0F 00 01 44 07
350   00 01 48 07 00 01 44 17 00 01 51 07 00 01 5B 07
360   00 01 51 07 00 01 5B 0F 00 01 6C 1F 00 01 36 1F
370   00 01 3C 17 00 01 44 07 00 01 48 0F 00 01 51 0F
380   00 01 5B 1F 00 01 51 0F 00 01 44 07 00 01 48 07
390   00 01 44 17 00 01 51 07 00 01 56 07 00 01 60 07
3A0   00 01 56 0F 00 01 51 20 51 20 00 A0 FF 01 00 10
3B0   36 0F 00 01 30 1F 00 01 2D 20 2D 1F 00 01 28 20
3C0   28 1F 00 01 2D 20 2D 1F 00 01 00 10 36 0F 00 01
3D0   30 1F 00 01 2D 20 2D 1F 00 01 28 20 28 1F 00 01
3E0   36 20 36 17 00 01 28 07 00 01 2B 07 00 01 30 07
3F0   00 01 2B 0F 00 01 28 10 00 10 00 10 36 0F 00 01

400   30 1F 00 01 2D 20 2D 1F 00 01 28 20 28 1F 00 01
410   2D 20 2D 20 00 10 36 0F 00 01 30 1F 00 01 2D 20
420   2D 1F 00 01 28 20 28 1F 00 01 36 20 36 17 00 01
430   28 07 00 01 2B 07 00 01 30 07 00 01 2B 0F 00 01
440   28 1F 00 01 51 1F 00 01 48 1F 00 01 40 2F 00 01
450   51 10 00 20 3C 1F 00 01 40 40 00 20 40 20 00 10
460   3C 10 30 10 24 10 00 10 28 0F 00 01 2B 20 00 10
470   40 10 36 10 28 10 00 10 36 0F 00 01 30 1F 00 01
480   2D 20 2D 1F 00 01 28 20 28 1F 00 01 2D 20 2D 1F
490   00 01 00 10 36 0F 00 01 30 1F 00 01 2D 20 2D 1F
4A0   00 01 28 20 28 1F 00 01 36 20 36 1F 00 01 40 0F
4B0   00 01 36 0F 00 01 36 1F 00 81 AD 03 00 10 44 0F
4C0   00 01 3C 20 3C 1F 00 01 36 1F 00 01 36 20 33 1F
4D0   00 01 3C 1F 00 01 36 20 00 10 44 0F 00 01 3C 20
4E0   3C 1F 00 01 36 1F 00 01 36 20 33 1F 00 01 3C 1F
4F0   00 01 44 17 00 01 00 08 3C 1F 00 01 44 10 00 10

500   00 10 44 0F 00 01 3C 20 3C 1F 00 01 36 1F 00 01
510   36 20 33 1F 00 01 3C 20 36 1F 00 01 00 10 44 0F
520   00 01 3C 20 3C 1F 00 01 36 1F 00 01 36 20 33 1F
530   00 01 3C 20 44 17 00 01 00 08 3C 1F 00 01 44 0F
540   00 01 36 0F 00 01 00 10 28 10 00 10 28 10 00 10
550   40 10 36 10 36 10 00 10 3C 10 3C 10 30 10 00 10
560   40 10 40 10 36 10 00 10 40 10 40 10 36 10 00 10
570   3C 10 3C 10 30 10 00 10 3C 10 3C 20 00 10 00 10
580   40 10 36 10 00 10 44 0F 00 01 3C 20 3C 1F 00 01
590   36 1F 00 01 36 20 33 1F 00 01 3C 1F 00 01 36 1F
5A0   00 01 00 10 44 0F 00 01 3C 20 3C 1F 00 01 36 1F
5B0   00 01 36 20 33 1F 00 01 3C 20 40 1F 00 01 51 1F
5C0   00 01 48 1F 00 81 BB 04 FF FF 51 1F 00 01 79 1F
5D0   00 01 5B 1F 00 01 44 1F 00 01 51 17 00 01 5B 07
5E0   00 01 66 1F 00 01 5B 1F 00 01 44 1F 00 01 51 1F
5F0   00 01 79 1F 00 01 5B 1F 00 01 44 1F 00 01 51 17

600   00 01 5B 07 00 01 66 1F 00 01 36 1F 00 01 51 1F
610   00 01 36 1F 00 01 51 1F 00 01 51 1F 00 01 79 1F
620   00 01 5B 1F 00 01 44 1F 00 01 51 17 00 01 5B 07
630   00 01 66 1F 00 01 5B 1F 00 01 44 1F 00 01 51 1F
640   00 01 79 1F 00 01 5B 1F 00 01 44 1F 00 01 51 17
650   00 01 5B 07 00 01 66 1F 00 01 36 1F 00 01 51 1F
660   00 01 36 1F 00 01 51 1F 00 01 51 1F 00 01 48 1F
670   00 01 40 2F 00 01 51 0F 00 01 3C 3F 00 01 51 3F
680   00 01 40 3F 00 01 3C 3F 00 01 36 2F 00 01 36 0F
690   00 01 51 3F 00 01 51 1F 00 01 79 1F 00 01 5B 1F
6A0   00 01 44 1F 00 01 51 17 00 01 5B 07 00 01 66 1F
6B0   00 01 5B 1F 00 01 44 1F 00 01 51 1F 00 01 79 1F
6C0   00 01 5B 1F 00 01 44 1F 00 01 51 17 00 01 5B 07
6D0   00 01 66 1F 00 01 36 1F 00 01 51 1F 00 01 51 1F
6E0   00 01 48 0F 00 01 36 0F 00 81 C9 05
```

Deck the Halls

```
200   33 17 00 01 39 07 00 01 3C 0F 00 01 44 0F 00 01
210   4C 0F 00 01 44 0F 00 01 3C 0F 00 01 4C 0F 00 01
220   44 07 00 01 3C 07 00 01 39 07 00 01 44 07 00 01
230   3C 17 00 01 44 07 00 01 4C 0F 00 01 51 0F 00 01
240   4C 1F 00 01 33 17 00 01 39 07 00 01 3C 0F 00 01
250   44 0F 00 01 4C 0F 00 01 44 0F 00 01 3C 0F 00 01
260   4C 0F 00 01 44 07 00 01 3C 07 00 01 39 07 00 01
270   44 07 00 01 3C 17 00 01 44 07 00 01 4C 0F 00 01
280   51 0F 00 01 4C 1F 00 01 44 17 00 01 3C 07 00 01
290   39 0F 00 01 44 0F 00 01 3C 17 00 01 39 07 00 01
2A0   33 0F 00 01 44 0F 00 01 3C 07 00 01 36 07 00 01
2B0   33 0F 00 01 2D 07 00 01 28 07 00 01 26 0F 00 01
2C0   28 0F 00 01 2D 0F 00 01 33 1F 00 01 33 17 00 01
2D0   39 07 00 01 3C 0F 00 01 44 0F 00 01 4C 0F 00 01
2E0   44 0F 00 01 3C 0F 00 01 4C 0F 00 01 2D 07 00 01
2F0   2D 07 00 01 2D 07 00 01 2D 07 00 01 33 17 00 01

300   39 07 00 01 3C 0F 00 01 44 0F 00 01 4C 20 00 40
310   00 40 00 C0 FF 01 3C 17 00 01 44 07 00 01 4C 0F
320   00 01 51 0F 00 01 5B 0F 00 01 51 0F 00 01 4C 0F
330   00 01 66 0F 00 01 51 07 00 01 4C 07 00 01 44 07
340   00 01 51 07 00 01 4C 17 00 01 5B 07 00 01 66 0F
350   00 01 66 0F 00 01 66 1F 00 01 3C 17 00 01 44 07
360   00 01 4C 0F 00 01 51 0F 00 01 5B 0F 00 01 51 0F
370   00 01 4C 0F 00 01 66 0F 00 01 51 07 00 01 4C 07
380   00 01 44 07 00 01 51 07 00 01 4C 17 00 01 5B 07
390   00 01 66 0F 00 01 66 0F 00 01 66 1F 00 01 51 17
3A0   00 01 4C 07 00 01 44 0F 00 01 51 0F 00 01 4C 17
3B0   00 01 4C 07 00 01 4C 0F 00 01 44 0F 00 01 4C 0F
3C0   00 01 4C 0F 00 01 3C 0F 00 01 3C 0F 00 01 44 0F
3D0   00 01 4C 0F 00 01 51 1F 00 01 3C 17 00 01 44 07
3E0   00 01 4C 0F 00 01 51 0F 00 01 5B 0F 00 01 51 0F
3F0   00 01 4C 0F 00 01 66 0F 00 01 4C 07 00 01 4C 07
400   00 01 4C 07 00 01 4C 07 00 01 4C 17 00 01 44 07
410   00 01 4C 0F 00 01 51 0F 00 01 4C 20 00 40 00 40
420   00 C0 15 03 33 17 00 01 33 07 00 01 33 0F 00 01
430   39 0F 00 01 3C 0F 00 01 33 0F 00 01 33 0F 00 01
440   3C 0F 00 01 33 07 00 01 33 07 00 01 33 07 00 01
450   33 07 00 01 33 17 00 01 39 07 00 01 3C 0F 00 01
460   44 0F 00 01 3C 1F 00 01 33 17 00 01 33 07 00 01
470   33 0F 00 01 39 0F 00 01 3C 0F 00 01 33 0F 00 01
480   33 0F 00 01 3C 0F 00 01 33 07 00 01 33 07 00 01
490   33 07 00 01 33 07 00 01 33 17 00 01 39 07 00 01
4A0   3C 0F 00 01 44 0F 00 01 3C 1F 00 01 33 17 00 01
4B0   33 07 00 01 33 0F 00 01 33 0F 00 01 33 17 00 01
4C0   44 07 00 01 3C 0F 00 01 33 0F 00 01 33 0F 00 01
4D0   33 0F 00 01 33 0F 00 01 33 0F 00 01 33 0F 00 01
4E0   36 0F 00 01 33 1F 00 01 33 17 00 01 33 07 00 01
4F0   33 0F 00 01 39 0F 00 01 3C 0F 00 01 33 0F 00 01

500   33 0F 00 01 3C 0F 00 01 39 07 00 01 39 07 00 01
510   39 07 00 01 39 07 00 01 33 17 00 01 2D 07 00 01
520   33 0F 00 01 39 0F 00 01 3C 20 00 40 00 40 00 C0
530   23 04 4C 17 00 01 51 07 00 01 4C 0F 00 01 66 0F
540   00 01 5B 0F 00 01 66 0F 00 01 4C 0F 00 01 4C 0F
550   00 01 66 07 00 01 66 07 00 01 66 07 00 01 66 07
560   00 01 4C 17 00 01 72 07 00 01 66 0F 00 01 66 0F
570   00 01 4C 1F 00 01 4C 17 00 01 51 07 00 01 4C 0F
580   00 01 66 0F 00 01 5B 0F 00 01 66 0F 00 01 4C 0F
590   00 01 4C 0F 00 01 66 07 00 01 66 07 00 01 66 07
5A0   00 01 66 07 00 01 4C 17 00 01 72 07 00 01 66 0F
5B0   00 01 66 0F 00 01 4C 1F 00 01 66 17 00 01 66 07
5C0   00 01 66 0F 00 01 66 0F 00 01 4C 17 00 01 4C 07
5D0   00 01 4C 0F 00 01 51 0F 00 01 4C 0F 00 01 3C 0F
5E0   00 01 4C 0F 00 01 5B 0F 00 01 44 0F 00 01 44 0F
5F0   00 01 66 1F 00 01 4C 17 00 01 51 07 00 01 4C 0F

600   00 01 66 07 00 01 60 07 00 01 5B 0F 00 01 66 0F
610   00 01 4C 0F 00 01 4C 0F 00 01 39 07 00 01 39 07
620   00 01 39 07 00 01 39 07 00 01 3C 17 00 01 39 07
630   00 01 33 0F 00 01 39 0F 00 01 4C 20 00 40 00 40
640   00 C0 31 05
```

©

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
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.

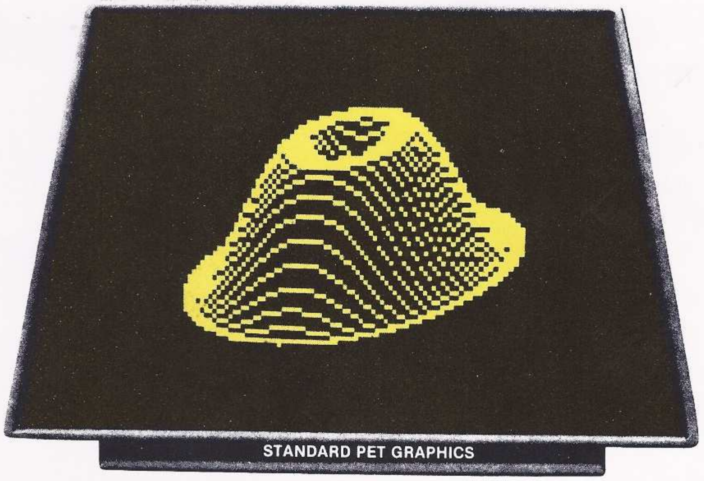
The module appears to my eye very well designed and professionally constructed. It was trivial to connect to a basic KIM (15 minutes or less). For review purposes only, the company provided a clock program on EPROM which is described as an example in their 19-page manual. I turned on power (an extra 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. ©

Editor's Note: If this review seems familiar to you, you may have read it in Issue 3 of COMPUTE. The blank half page in that issue was supposed to be the company's ad. Hopefully it's in this issue. We're reprinting the review as a service to you and them. My apologies to Edward H. Carlson, author of ***Fast Tape Read/Write Programs for your OSI*** *(Issue 3, COMPUTE, p. 115). Here, in full, is Listing 3. Oh well... RCL*

```
 10;        FAST KC TAPE READ
 20;
 30 LEADER =$0F     LEADER CHARACTER, $0F
 40 SCREEN =$D100   LOCATION ON MONITOR SCREEN
 50 ACIA   =$FC00   6850 ACIA TAPE PORT
 60 START  =$00     HOLDS ADDRESS OF 1ST BYTE OF TEXT
 70 END    =$02     HOLDS ADDRESS OF LAST BYTE OF TEXT
 80 EXECUT =$04     CONTAINS ADDRESS OF PROGRAM START
 90 CURENT =$06     HOLDS ADDRESS OF CURRENT TEXT BYTE
100 CHKSUM =$08     CHECK SUM FROM TAPE STORED HERE
110 COUNT  =$09     COMPUTED CHECK SUM AND OTHER STUFF
120        *=$C700
130        LDA #'N        READING NOISE BEFORE LEADER
140        STA SCREEN+2
150 MAIN   LDY #0         READ LEADER, $0F 0F 0F
160        STY COUNT
170 M1     JSR RT         READ TAPE BYTE
180        STA SCREEN
190        CMP #LEADER    IS IT A LEADER BYTE?
200        BNE MAIN       NO, READ ANOTHER BYTE
210        INC COUNT      YES, INCREMENT
220        LDA #'L        PRINT L FOR EVERY $0F READ
230        STA SCREEN+4,Y
240        INY
250        LDA #3         READ 3 OF THEM?
260        CMP COUNT
270        BNE M1         NOT YET, READ ANOTHER
280 ADDR   LDY #0         LEADER OVER.  READ START,
290        STY COUNT        END, EXECUTE ADDRESSES
300        LDA #'A
310        STA SCREEN+8
320 A1     JSR RT
330        STA START,Y
340        STA SCREEN
350        INY
360        CPY #6
370        BNE A1         BRANCH TO CONTINUE READING A
380        LDA START      SET INITIAL ADDRESS
390        STA CURENT+1
400        LDA START+1
410        STA CURENT
420 TEXT   LDY #0
430        STY COUNT      CLEAR FOR CALC. CHECK SUM
440        LDA #'T
450        STA SCREEN+10
460 RBT    JSR RT         READ A BYTE OF TEXT
470        STA (CURENT),Y
480        STA SCREEN
490        CLC
500        ADC COUNT      COUNT ACCUMULATES CHECK SUM
510        STA COUNT
520        LDA CURENT     TEST FOR END OF TEXT, LO
530        CMP END+1
540        BNE M3         NOT EQUAL, INC AND READ BYTE
550        LDA CURENT+1   LO EQUAL, TEST HI
560        CMP END
570        BEQ M6         BRANCH IF TEXT IS ALL READ
580 M3     INC CURENT     INCREMENT CURRENT ADDRESS
590        BNE M4
600        INC CURENT+1
610 M4     JMP RBT        GO READ NEXT BYTE
620 M6     JSR RT         READ CHECK SUM BYTE
630        STA CHKSUM
640        CMP COUNT      TEST CHECK SUM
650        BEQ GO         IF OK, BRANCH AND EXECUTE
660        LDA #'E        IF NOT. PRINT ERROR MESSAGE
670        STA SCREEN+12
680        JSR $CB48      BELL
690        BRK
700 GO     LDA EXECUT
710        STA CURENT+1
720        LDA EXECUT+1
730        STA CURENT
740        JSR $CB48      BELL
750        JMP (CURENT)   EXECUTE
760;
770:                      TAPE READ SUBROUTINE
780 RT     LDA ACIA       READ A BYTE FROM 6850
790        LSR A
800        BCC RT
810        LDA ACIA+1
820        RTS
```

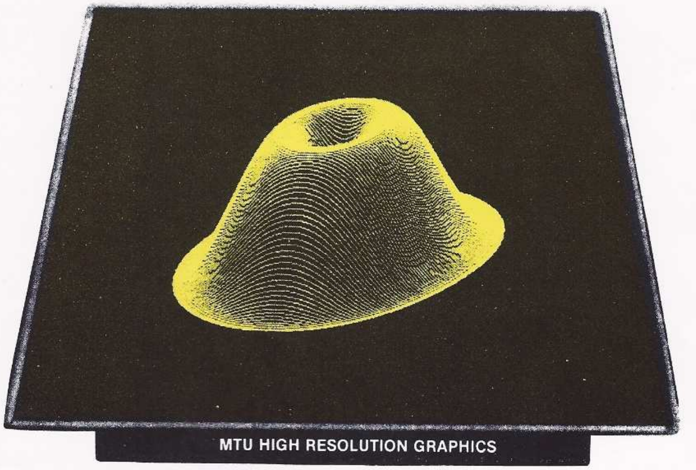
©



STANDARD PET GRAPHICS

HIGH RESOLUTION GRAPHICS

LOOK TO MTU. WE SUPPORT HIGH RESOLUTION GRAPHICS ON:
PET — AIM — KIM — SYM



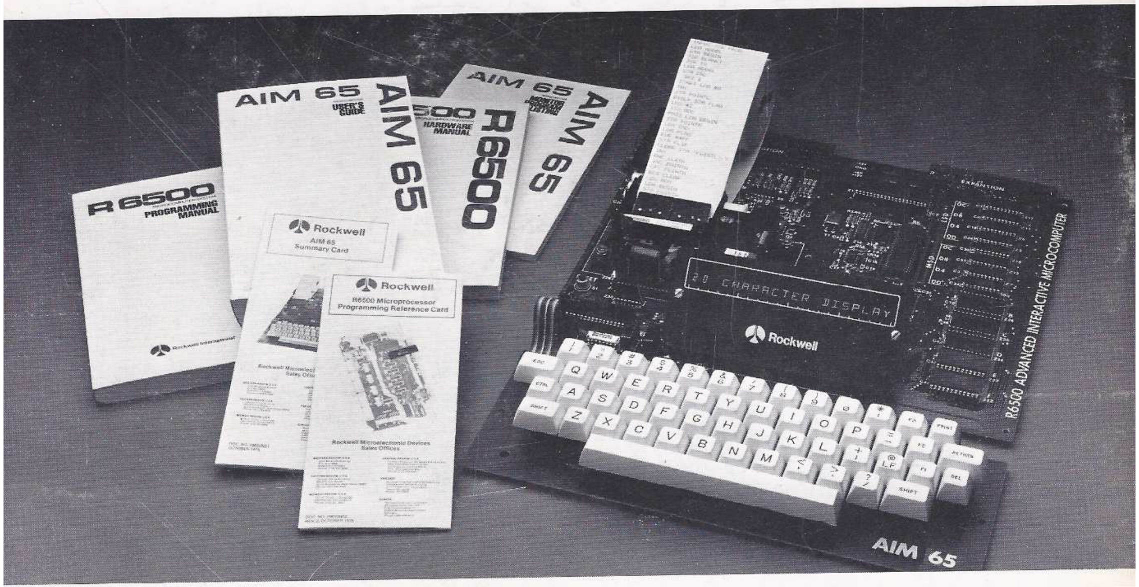
MTU HIGH RESOLUTION GRAPHICS

MTU

Micro Technology Unlimited
P.O. Box 4596, 841 Galaxy Way
Manchester, N.H. 03108
603-627-1464

Call Or Write For Our Full Line Catalog

AIM 65. The head start in educational microcomputers.


R6500 PROGRAMMING MANUAL
AIM 65 USER'S GUIDE
R6500 HARDWARE MANUAL
AIM 65 MONITOR PROGRAM LISTING
Rockwell AIM 65 Summary Card
Rockwell R6500 Microprocessor Programming Reference Card
20 CHARACTER DISPLAY
Rockwell
R6500 ADVANCED INTERACTIVE MICROCOMPUTER
AIM 65

On-Board Printer, Advanced R6502 CPU, Versatile I/O — It's the Honors Candidate for Microprocessor Learning

Micro Power

It's tops in its class because it's expressly designed for microprocessor learning. Rockwell's AIM 65 is a fully-assembled microcomputer system with special educational features at a low price school budgets can afford.

AIM 65's on-board thermal printer — unique in its price range — produces hard copies of exercises for easy checking by student and instructor. On-board I/Os provide dual cassette, TTY and general purpose interfaces. Bus and system expansion is built in. Same for PROM, ROM and RAM expansion.

AIM 65's Interactive Monitor prompts students each step of the way in hands-on learning of microprocessor fundamentals. It includes a Text Editor, Interactive Mnemonic Assembler, Debugger (Trace, Breakpoints), and more!

An optional fully symbolic Assembler program makes AIM 65 a powerful hands-on learning system for microcomputer development and prototyping. Advanced students can explore high level languages with an optional ROM-resident BASIC Interpreter. There's even a college textbook available.

And you'll find AIM 65 is ideal for equipment control and other laboratory computer applications.

Discover how with one low investment you can combine several AIM 65s for hands-on, high-productivity microprocessor learning in classes where students don't have to wait in line. Check the high features and low prices of Rockwell's AIM 65 printing microcomputer.

Contact your local distributor or write or call AIM 65 Marketing, Electronic Devices Division, Rockwell International, P.O. Box 3669, D727, Anaheim, CA 92803, (714) 632-3824.

Rockwell International

...where science gets down to business