H. R. LUXENBERG 8/29/80
EDITOR: SYM-PHYSIS ,SYM-1 USERS' GROUP NEWSLETTER
P. O. BOX 315 CHICO,CA 95927

LUX,
JUST A SHORT CASSETTE TO EXPRESS MY APPRECIATION OF SYMPHYSIS. YOUR MATERIAL HAS GREATLY HASTENED MY EVOLUTION OUT OF SIMPLESYMDOM. I AM COMPOSING THIS FILE USING MY SYM, A SURPLUS XEROX KEYBOARD FROM CALIFORNIA DIGITAL, AND PAIA ELECTRONICS' TVT 6-5/8 (CHEAP VIDEO). I REMEMBER SEEING IN ISSUE 3 OF SYM-PHYSIS A PROMISE TO DISCUSS 'CHEAP' VIDEO TERMINALS, WHICH I WAITED ANXIOUSLY FOR, BUT DID NOT SEE. I HAVE JUST ADAPTED CHEAP VIDEO TO THE SYM-1 AND THOUGHT OTHERS MIGHT BE INTERSTED IN MY EXPERIENCE.

CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

CHEAP VIDEO FOR THE SYM

JOHN MATTOX
102 NW 27 TERRACE
GAINESVILLE FL 32607
(904) 378-6768

PAIA ELECTRONICS SELLS A BARE BONES VIDEO DRIVER KIT (TVT6-5/8) FOR $43, WHICH INCLUDES DON LANCASTER'S CHEAP VIDEO COOKBOOK. TVT CONSISTS OF SEVEN INTEGRATED CIRCUITS; AN UPPER CASE CHARACTER GENERATOR IS SUPPLIED WITH THE KIT. THE IMPLEMENTATION OF AN UPPER-LOWER CASE GENERATOR IS DISCUSSED.

TVT IS DESIGNED FOR THE KIM. THE DESIGN REQUIRES THE DISPLAY MEMORY DATA BUS TO BE BUFFERED (TO PROVIDE DIRECT ACCESS BY TVT WHILE THE HOST MICROPROCESSOR EXECUTES A SUBROUTINE WHICH INCREMENTS THE ADDRESS BUS). THIS MEANS THAT ADDITIONAL OFF-BOARD MEMORY IS REQUIRED WITH TRI-STATE DRIVERS IN BOTH DIRECTIONS FOR THE SYM.

TVT USES THE 4 HIGHEST ADDRESS LINES AS DISPLAY INSTRUCTIONS. THIS MEANS THAT IF ONE USES THE INSTRUCTION DECODER PROM SUPPLIED WITH THE KIT, ADDRESS SPACE $6000-$DFFF MUST BE RESERVED FOR TVT. THIS IS CLEARLY UNACCEPTABLE. FOR $3 PAIA ELECTRONICS WILL TRADE THE PROM FOR ONE PROGRAMMED TO USE MEMORY SPACE $2000 THROUGH $9FFF. THE MONITOR ENABLE JUMPER MUST BE REPLACED WITH OFF BOARD LOGIC USING A 6522 VIA TO SWITCH BETWEEN ENABLING THE MONITOR AND ENABLING THE TVT. ADDITIONAL EXPANSIONS WILL ALSO NEED TO BE DESELECTABLE.

IN ADDITION TO USING 50% OF THE ADDRESS FIELD, TVT REQUIRES UP TO 95% OF THE CPU TIME (THERE AIN'T NO FREE LUNCH). ALSO, FOR DISPLAYS WITH MORE THAN 40 COLUMNS, THE HORIZONTAL FREQUENCY OF THE CRT MONITOR MUST BE REDUCED.

MY CURRENT IMPLEMENTATION IS TO DISPLAY WHILE WAITING FOR KEYBOARD INPUT (WHICH IS TIME OTHERWISE WASTED). WITH A 1 K DISPLAY MEMORY, I AM USING A 64X16 DISPLAY. I AM BUILDING AN 8 K DISPLAY MEMORY WHICH WILL ALLOW A 80X24 DISPLAY AND 256X256 BLACK & WHITE GRAPHICS.

CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

I INCLUDE HARDCOPY BECAUSE I AM NOT CERTAIN OF MY RECORDER. YOU MAY PUBLISH WHAT YOU WISH OF IT. IF I HAD IT TO DO OVER, I WOULD HAVE INSTEAD PURCHASED KTM-2/80.

SINCERELY,

JOHN

John: Thanks for the valuable writeup. Many of our readers will also find it helpful. Your recorder writes fine, reads easily! - Lux.

Dr. H. R. Luxenberg
Editor SYM-PHYSIS
P. O. Box 315
Chico, California 95927

August 14, 1980

Dear Dr. Luxenberg:

I have been using Jack Brown's "Super TCP" and "Ultrarenumber" along with a modified version of Tom Gettys' 'Merge/Delete' and find them to be a marvelous enhancement to my SYM. I've put them together with a graphics package for MTU's K-1008 board, a tape verify and tape directory segment (after the one of Jack Gieryic), and linkage to a printer, all on a pair of 2716 EPROMs (stacked, of course) with the result that I have a very nice overall operating system for the SYM. My primary use of the system right now is to write programs for lecture demonstration in Physics classes using the TV graphics.

In using Ultrarenumber I discovered that if it finds reference to a nonexistent line number following GOTO, GOSUB, etc. it gives it the number 65535 which is an illegal line number in SYM-1 BASIC. The This means that you cannot run a renumbered program until all the illegal references are individually corrected. The largest acceptable line number in SYM-1 BASIC is 63999 which is F9FF in hex. Therefore I changed lines 3220-3230 in Ultra-renumber as follows:

```
3220   STA *FACTO-1
3222   LDA #$F9
3230   STA *FACTO
```

Now when the above circumstance occurs, it recomputes the references to the nonexistent lines as 63999. References to this line number can then be trapped out by using a statement such as:

```
63999 GOTO____
```

which takes the program to some appropriate point. I have not found a situation in which I could not get a satisfactory renumbering job.

I certainly enjoy SYM-PHYSIS and find it generally very helpful. I think it is clearly the best publication going for "Symmers". I have found the work of Jack Brown and Tom Gettys very instructive, especially in understanding how some of the BASIC routines work. Keep up the good work.

Sincerely,

James G. Pengra
Department of Physics
Whitman College
Walla Walla, Washington 99362

SYM AS A PERSONAL COMPUTER & THE EDUCATIONAL/ACADEMIC MARKET

(continued from page 5/6-2)

Synertek appreciates, and are exploring the possibilities of OEMing a package built around this nucleus, with attractive discounts to individual faculty and students to encourage personal systems as well. We consider working engineers just breaking-in to the microcomputer field as being in the "student" category, for discount purposes, too.

We have purveyed several such systems locally, and would be interested in hearing your suggestions and comments about this class of systems.

HOW TO ADD MORE VIAS EASILY

The SYM-1 dedicates a full 4 K of address space ($A000-$AFFF) to the 6532 and the three on-board 6522s, 1 K to each. The 6532 needs 128 addresses for its RAM, and 32 addresses for its I/O-TIMERS, but the 6522s each need only 16 addresses of their assigned 1 K blocks.

When the problem arose of adding the HDE Disk Controller, which has its own on-board 6522, with full address decoding, we assigned it the addresses $A88X (16 bytes only), and temporarily removed the (user supplied) VIA #2 in U28, so that there would be no address conflicts. We could have broken the 1 K block at $A800-$ABFF in half, and assigned $AA00-$ABFF to any added VIAs, but decided to use only page $A8 for all I/O, since the upper half of this one page alone would allow the addition of eight more VIAs (with full decoding).

The first two purchasers of the HDE SYM-FODS came up with two very different hardware implementations of the simple logic to keep VIA #2 out of the upper halves of the four pages to which it has access. These are both presented elsewhere in this issue. It is interesting to note that, not only did each chose a different logical equation to implement, each selected a different technology. One chose the IC approach, adding a chip; the other chose DTL (Diode Transistor Logic) because it (the transistor) was already there.

As of now we have not added the additional VIAs we thought we would want. One reason is that we were able to "recover" the use of PB 6 on VIA #1. The second reason is that we would like to put all of our "extra" ports at the Auxilliary Application Connector, and not load our unbuffered Address and Data Busses at the Expansion Connector down any further, even though we have had no problems as yet. In fact, we just installed the Color-Mate color graphics system on the (unbuffered) expansion bus (addresses $9000-$9FFF) with no problems.

Since we fully intend to install the AY-3-8910 Programmable Sound Generator (one of these days!), we plan to make use of its "free" built-in pair of 8-bit I/O ports, which can be used independently of the sound generation function, to handle the control functions we wish. One of the ports could serve as the origin of a two-way data bus to a number of other VIAs, and the second port could provide the chip select and resister select functions. Obviously we have not thought this through in detail as yet, because we are still not yet certain of our requirements.

MORE FROM JOHN GIERYIC ABOUT JACK BUILT PROGRAMS

We told Jack that we didn't think there would be a great market for games for the SYM-1/KTM-2 system. This turned out to be true; most of our readers seem to want mostly "utilities." The demand for SWP-1, and BBE-1 (our new name for the Brown Basic Enhancements Package, given because the Purchasing Office people who place orders with us feel uncomfortable unless the product has an ordering number as well as a name), has been high; the Jack Built Programs have not shaken the earth with fast movement.

Well, Jack has generated some PLOT Utilities for BASIC, which work with the (built-in) Gowan Double Density Plot and Tris Patch machine language programs. These additions to the Jack-Built Programs line may be ordered through the Users' Group. The best way to describe these programs is to reprint the instruction sheet; this we do below. He also provided some bus exterminators and a well human-engineered BASIC EPROM Programmmer Program. We will therefore turn the next few pages over to Jack. But first, a few more editorial comments (to keep our editorial license current):

Jack is an extremely good BASIC programmer, and is using RAE to prepare his manuscripts for publication. We're glad about the latter, since it is so easy to "SWP" his manuscripts into camera-ready copy. His PLOT programs mix BASIC with MLC (Machine Language Code), which is, of course, generated by RAE. We hope we can persuade Jack to redo one of his PLOT programs in SYM-FORTH, and report back to us a (not necessarily objective) comparision of the two approaches.

We are curious to hear what this Jack will have to say on FORTH versus BASIC, because our other Jack (Jack Brown, that is), also very well versed in BASIC, has sent us some graphics programs (for the MTU Visible Memory), and the classical "Towers of Hanoi" as a beautiful graphics demonstration on the KTM-2/80, written in FORTH, with its own built-in Tris Patch, and MLC portions, compiled with FORTH'S built-in ASSEMBLER Vocabulary. We find, as only "occasional" programmers (meaning only when we have to, because we couldn't con someone else into doing it for us!), that the FORTH program is actually easier to follow than a BASIC one, in spite of many more years of BASIC experience.

from JACK BUILT PROGRAMS

These three programs all require 8K of RAM, a KTM-2/80 and BAS-1. Enter BASIC with 6400 bytes free. Each program consists of a BASIC program part and a machine language part (19F0 thru 1FFF) containing Bill Gowans' plot package from SYM-PHYSIS Issue #3 plus the tris functions. Two loads are required in BASIC. The first loads the machine language portion and the second loads the BASIC program. Once this is done the user can go to one of the other two programs by loading only the BASIC part of that other program. The machine language portion is identical for all three programs.

```
**********************
*                    *
* DUAL Y-AXIS PLOT   *
*                    *
**********************
```

This package produces a plot of two equations of the form Y = F(X) over the same user-specified x range, while using a different user-specified y range for each equation. This concept is similar to the dual trace oscilloscope. It allows the user to view two widely differing plots superimposed on the same "piece of graph paper". Two equations are already in memory. Enterins the followins values in response to the program prompts will sive a "feel" for the program: -1, 1, 0, 720, 0, 720.

```
**********************
*                    *
* 4 QUADRANT PLOT    *
*                    *
**********************
```

Up to four independent equations of the form Y = F(X) are plotted on a four quadrant grid which utilizes the entire monitor screen. The user specifies the number of equations to plot, the maximum positive x value and the maximum positive y value. The program plots the equation(s) on a 160(h) by 48(v) "dot" grid. Enter the followins values in response to the program prompts: 2, 1, 720. This will produce four complete cycles of a sine wave and another 4 cycles of a cosine wave.

```
**************
*            *
* POLAR PLOT *
*            *
**************
```

Up to four independent equations of the form R=F(TH) are plotted on a polar grid which uses the entire monitor screen. The scaling is adjusted such that the plot of R = 5 does appear as a circle on the monitor. The program prompts the user for the maximum R value, the range of THETA (minimum and maximum), and the increment value. For example, if the range of THETA is 10 to 30 degrees with an increment of 4, then the program would plot a point every 4 degrees between 10 and 30. A minimum THETA of 0 cannot be entered due to error checking by the program. Try entering the following values to the program prompts: 4, 6, 1, 360, 4. This will show what the program can do. Now change line 20 to read R=TH and enter RUN. Enter the following values to the prompts: 1, 27, 1, 1440, 6. This should result in an impressive spiral.

JACK BUILT PROGRAM BUGS

Two of my programs have minor buss. I want to thank Don Full and Cap Teisen for bringing these to my attention. Lux already has the fixes incorporated into his source. To the many who have already purchased the two programs, check your source. If the following fixes are missing, then add them. Do not write over your original tape. Copy the corrected source to a new tape. If the new tape checks out OK, then, and only then, should you rewrite your original tape.

BAR GRAPH

Symptoms: Very short bars (less than 8 scan lines tall) are actually 8 scan lines taller than they should be. With this fix these short bars will be the correct length. If the bar is too short to appear as even a single scan line, that bar on the graph will be blank. That's the way it should be.

Fix: Insert the following line: 526 IFC=0THENQ=1:GOTO532

PLOT

Symptoms: Under certain conditions, a single point does not appear to "follow" the plot. Instead it appears at a point somewhere below its real position and the plot has a "hole" (missing plot point) at this position. This hole will appear at the very top line of the graph. The bug is a very common one; a relational operator in an IF statement did not include the "equal to" case.

Fix: Insert an "=" in line 153 so it reads as follows:

```
153 IFY>=YLTHENB=39:GOTO160
```

MORE ON THE EPROMMER

Dear Lux and all SYMmers:

I read with extreme interest Joe Hobart's article "An EPROM Programmer for the SYM-1" in SYM-PHYSIS Issue #4. His idea was elegant and, best of all, cheap! As a complement to his simple hardware design, I have written a very comprehensive software package which gives the user a great deal of power when programming the 2516 or 2716 EPROM.

I reassigned the ports on the 6522's such that NO modifications are necessary to the SYM. Simply wire the programmer per the directions in the enclosed article and you're ready to RUN. This allows you to use the four buffered bits on Port AC00 for other applications on the "AA BUS".

For you lucky users who own a KTM-2 or KTM-2/80, my program will utilize cursor positioning to make the man-machine interface a bit more pleasing. For you other terminal owners, all data and displays will be left justified on your screen.

An additional feature of this program is the ability to turn off all power to the EPROM when the EPROM is not being accessed. This enables the user to change EPROMs without turning off the SYM. While the option list is being displayed and the program is waiting for the user's selection, all port bits are logic 0. If the user installs a double pole single throw switch on the +5 volts to EPROM pin 24, and the +25 volts to EPROM pin 21, the user can turn this switch off and remove/replace/insert EPROMs. This switch need be turned on only for options 1,2,3,4 and 7.

Wouldn't it be nice if this switch could be automated? Well, I've provided such a signal on the AA connector pin S (3PB3). This pin will go high when the EPROM is accessed (options 1, 2, 3, 4, and 7). This signal can be used as a means to control a relay which would replace the manual dpst switch.

There is yet another method of implementing this automatic control of the +5 and +25 voltages. In my case, I used two SIGMA relays (part no. 191TE1A1-5S). These are dual-in-line packaged reed relays (14 pin DIP) with an internal suppression diode. Each relay has a single pole single throw switch. One relay is used to switch the +5 volts and the other to switch the +25 volts. The signal from pin S (3PB3) drives an inverter (7404), which, in turn, is used to sink current on the +5 volt control relay coil. Another signal is provided on the Y pin (3PB4) to drive another inverter, which is used to sink current on the +25 volt control relay coil.

This use of one of the buffered bits on the port at AC00 requires a change on the SYM. Buffer B4 (lower lefthand of the SYM board) should have point A jumpered to point 3 (refer to Figure 4-5a in your SYM Reference Manual) and point B jumpered to point 18. Also, the .47 capacitor to EPROM pin 24 is repositioned so it is always connected from ground to +5 volts. My schematic for this method is summed up below.

SYMcerely,

Jack Gieryic

From JACK BUILT PROGRAMS

```
********************************
*                              *
* 2516/2716 EPROM PROGRAMMER *
*                              *
********************************
```

Hardware requirements:

SYM-1
SYM BASIC
7K RAM
Terminal with at least 16 lines and
at least 32 characters per line

This program presents the user with a list of 8 options from which to choose. When the selected option is completed the program will ask the user if he/she wants to go back to the option display. If the response is YES, then the option list will again be presented. All addresses and limits can be entered in either decimal or hex. Hex numbers are simply preceded by the letter H, e.g., H2F5.

OPTIONS

- 1 - PROGRAM EPROM
- 2 - COMPARE EPROM TO MEMORY
- 3 - VERIFY EPROM IS CLEARED
- 4 - DISPLAY EPROM MEMORY
- 5 - DISPLAY MEMORY
- 6 - ENTER MEMORY DATA
- 7 - READ EPROM TO MEMORY
- 8 - MEMORY MOVE

OPTION 1 - This option is used to program the EPROM. It prompts the user for the EPROM starting address, the number of bytes to program, and the data starting address in memory. The EPROM starting address plus the number of bytes cannot extend beyond the 2K of the EPROM. If it does, the program will again prompt the user for all of the data. This option allows the user to program any number of bytes, anywhere within the EPROM, from any area of the SYM memory, without disturbing the remaining locations in the EPROM. After the programing is completed this option will verify the data just written, and any errors will be displayed in the following format:

ERROR "EPROM address" "EPROM data" "memory address" "memory data"

If the requested parameters were entered in decimal then the error data is displayed in decimal. If the parameters were entered in hex then the error data will be displayed in hex.

OPTION 2 - This option will compare any portion of the EPROM to memory. The option requests 3 parameters, as does option 1. Any errors are displayed as in option 1.

OPTION 3 - This option will verify that any part of the EPROM contains hex FF. It will request a starting address and the number of bytes to check. Any locations not containing FF will be printed, along with the data found in those locations. It is wise to verify if the EPROM is cleared, prior to programming, as this could save a lot of time.

OPTION 4 - This option is used to view the contents of any number of contiguous bytes in the EPROM without bringing the data into memory. The option asks for the starting address and the number of bytes to display. The data is displayed in hex regardless of the format used to enter the address and byte count (decimal or hex).

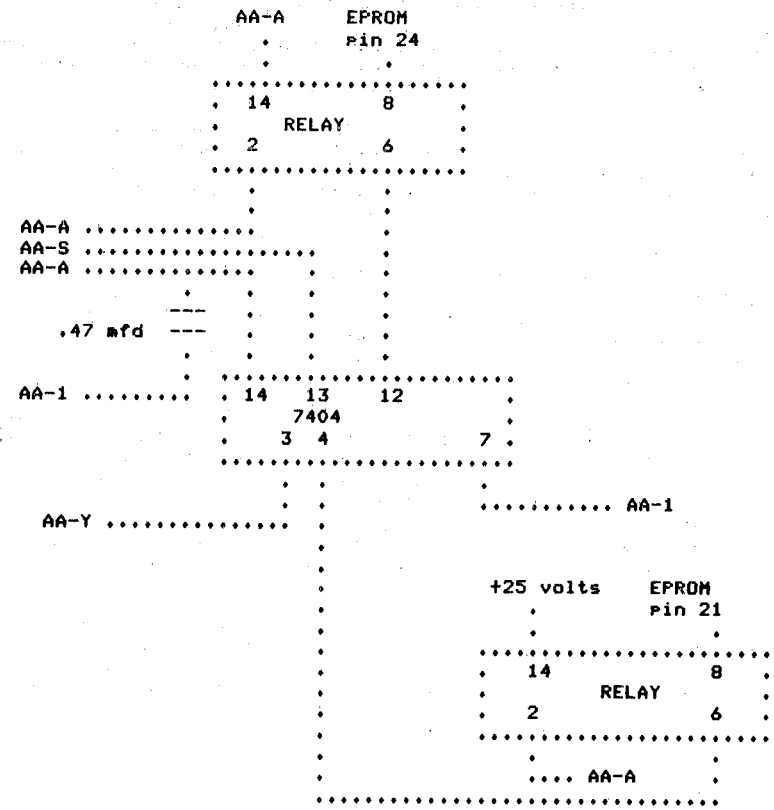
OPTION 5 - This option is used to view the buffer before burning it into the EPROM. Any area within memory may be viewed, and there is no limit on the number of bytes to display. The display format is identical to option 4.

OPTION 6 - This option will allow the user to hand-construct a buffer in memory for transfer to the EPROM via option 1. This option will also enable the user to change data before burning it into an EPROM. This option requests a starting address (decimal or hex), and then displays the address (decimal or hex), followed by the data (hex only). A dash appears after the data and the user now enters new hex data to change that location, or the same data as is displayed, to skip over that location. The next address and its data are now displayed, and so on, until the user enters the letters END, instead of data. This terminates the option.

OPTION 7 - This option allows the user to transfer any or all of an EPROM's data into the SYM's memory. This allows the user to change the data via option 6, if desired, and then burn a new EPROM, via option 1. The option requests the EPROM's starting address, the number of bytes to read in, and the beginning of the buffer in the SYM's memory. Make sure your memory buffer is large enough!

OPTION 8 - This option will allow the user to move any size block of memory into another area of memory. With this option, the user can move data from an EPROM already being used in his/her system into RAM, alter it via option 6, and then burn a new EPROM. All this without touching the EPROM being used in the system!

Note: Options 6, 7 and 8 will not allow the user to use page 0, page 1, or the memory occupied by the program. This prevents self-destruction. This program resides in the first 5K of memory.



Above is the schematic for Jack's Automatic EPROM Programmer Switcher. His revised wiring list for the EPROM Programmer is on page 5/6 -7. Please note that the BASIC listings for the EPROM PROGRAMMER and for the PLOT programs are NOT published in this issue! Only the instructions for these five programs are given here, so that you can decide whether or not they will be useful enough to purchase in cassette form from JACK BUILT PROGRAMS.

PRODUCT RECOMMENDATIONS

We have tested a number of new products since Issue No. 4, and can recommend all but one of them (that one will not be listed here). Some of these we will be distributing. Please note that we don't recommend because we are distributing. On the other hand, if we can recommend a product, we may try to get distribution rights, if possible.

The ColorMate

The easiest way to describe this newest product from MicroMate, P. O. Box 50111, Indianapolis, IN 46256, is to reprint the following extract from their brochure:

> ColorMate brings the flexibility of color video display to KIM, SYM and AIM microcomputer systems. Designed around the Motorola 6847 video display generator, ColorMate offers nine modes of operation, ranging from alphanumeric to full graphic. A 12-bit word format in the alphanumeric/semigraphic modes provides capability to mix alphanumeric with semigraphic characters. Two pages of video memory are implemented in the alphanumeric/semigraphic modes, providing added flexibility in many applications. Selection of the page to be displayed requires only a write to the ColorMate control register. The alphanumeric display format is 16 rows by 32 columns. Semigraphic modes provide colorful applications on a 48 by 64 element grid. Full graphics resolution ranges to 192 by 128.
>
> The ColorMate is memory mapped, and requires a 4K byte memory space. Decoding to 4K boundaries is provided onboard. It interfaces to the Expansion (E) connector of the KIM, SYM and AIM microcomputers. The single 5 volt supply is provided via the interface. The PC board is a standard 4½ by 6½ inch format, with plated through holes and gold contacts. The ColorMate may be configured to drive a standard TV (ch. 3 or 4).
>
> In addition to a complete description of the various modes of operation,the ColorMate User Manual includes example software for a color video terminal driver (ViewMate), a Fourier synthesis program, and various graphics plot subroutines.

ColorMate is available directly from MicroMate, either as the PC board and manual alone, for $50, or fully assembled and tested, but without the necessary IC's, at $95. In either case, you will have to hunt up the IC's somewhere; you can most easily obtain the complete set of IC's from MicroMate, for $125 additional. Add 2% to all prices, for shipping and handling. NOTE: The ColorMate will NOT work with PAL or SECAM color TV!!!!

While the maximum full graphics mode resolution of 128 H x 192 V is less than that of the Atari (320 H x 192 V), the Apple II (280 H x 192 V), or the TRS-80 Color Computer (256 H x 192 V), the color capability is still quite impressive, and can be very useful and effective. The ColorMate requires 4 K of address space (ours is at $9000-$9FFF), with 3 K of video mapped 2114 RAM on board (the remaining 1 K is not memory mapped, but is used as the "address" for the control register). We use a B/W monitor switched between the KTM-2/80 and the MTU Visible Memory, with a Radio Shack Coax Switch, for most purposes. The color graphics coexist simultaneously on a color TV set. The ColorMate has its own on-board RF Modulator, so that connection to the TV is through a 75 ohm to 300 ohm switching adaptor to the VHF antenna terminals on the set. When the TV is not available for SYM use, SYM has an extra 3 K of RAM with which to play!

Several subscribers have already written, or phoned, to let us know they are using, and are pleased with, ColorMate. Dick Turpin has provided plenty of software in the manual with which to get started, and will be supporting a User Group. The source code is in RAE format, and we are making arrangements with Dick to provide ColorMate Graphics software in cassette form. The board uses what we are calling the "Reverse KIM" pinout. See the next recommendation for installation suggestions.

The Quest Expansion Board

As many of you have discovered, not all 44-contact SYM/KIM/AIM expansion boards are compatible. With the exception of contacts E-16, -17, -18, -19, -20, and -X, 38 out of 44 of the SYM, KIM, and AIM expansion "Pin-out" assignments are identical. One group of expansion boards (MTU, for example) has, except for E-2, and E-3 (in addition to those listed above), the identical contact assignments. For this class of boards, any "mother-board" must have what we are calling the "KIM-1" Bus. All connector contacts, except for the ones listed above, are wired in parallel with the contacts on a connector into which the SYM is "plugged."

MOS Technology, producers of the KIM-1, devised, and marketed, for a very short time only, the KIM-4 Motherboard. All of the lettered, and two of the numbered, contacts were shifted by one position with respect to the KIM-1 Bus, and some signals were dropped, and others added. The Computerist, Hudson Digital Electronics, and RNB Enterprises, among others, support the KIM-4 Bus structure, with varying degrees of fidelity.

Other expansion boards are available, most notably the ColorMate and the Beta Computer Devices' 32 K Memory Board, which use the "Reverse KIM" Bus. These are designed to plug directly into a connector which "extends" and replaces the edge contacts on the SYM board. Alternately, a connector may be "reversed" and its solder-eyes or solder-tails soldered directly to the edge contacts on the expansion board, and the board/connector combination mounted directly onto the edge contacts of the SYM.

What's a fella to do, if, like me, he wants to add an HDE Disk Controller (KIM-4 Bus), ColorMate (Reverse KIM Bus), and an MTU Visible Memory Board (KIM-1 Bus) to the same SYM? We found that the 44-contact Expansion Board which was developed by Quest Electronics, P. O. Box 4430, Santa Clara, CA 95054, for use with their Super Elf (an excellent, RCA 1802 based, single board computer) System, when fitted with an extra, reverse mounted, 44-contact connector, fills the bill admirably. It can be fitted to the SYM straight out, or at a right angle. If fitted at a right angle, the three solder tail sockets can be mounted on either side of the board, to project either forward or backward. There are three rows of installation holes for each connector, so that the connectors can be installed in either of two positions, to provide either direct or reversed KIM-1 Bus.

There is plenty of room between the connectors to cut the appropriate traces, and the unused rows of installation holes make it easy to insert the necessary jumpers to convert one or more of the connectors to accept a KIM-4 board, e. g., the HDE Disk Controller. These boards can also be used on the Applications Connector, to mount a pair of DACs for stereo music or vector graphics, for example, and also on the Auxilliary Applications Connector, for all kinds of goodies. If you make your own application boards, you wire the contacts to match the application connector. If you use commercially available boards, such as MTU's DACs, you must, of course, cut and jumper the traces on the expansion boards to match.

While we are on the subject of busses, we should mention the SYM compatible "S-44" bus, and series of memory and other cards available from Kathryn Atwood Enterprises, P.O. Box 5203, Orange, CA 92667. This bus is ingeniously arranged so that no damage will be done if the boards are inserted backwards, and, in fact, many of the boards will work either way! This has to be the ultimate in "idiot-proofing" equipment.

The Beta Computer Devices 32 K Dynamic RAM Board

The SYM is expandable to 4 K of RAM on-board, and the Blalock Memory Board provides an easy expansion to 8 K. As many of our readers have discovered, with BASIC and/or RAE in ROM, an 8 K RAM SYM makes a very respectable system. Additional utilities are very easily put in EPROM.

The ColorMate provides its own 3 K of RAM, if you want color graphics. If you want to add a disk system, you will need at least 8 K more of RAM, but once you have the disk system up, you will want more and more RAM, for all of the things the disk system will let you do.

One of our SYMs has been expanded to 32 K, by adding MTU's 16 K dynamic RAM, and MTU's 8 K Visible Memory (also dynamic). The 8 K Visible Memory provides the bonus of high resolution black and white graphics. The disadvantages include the necessity for adding additional voltages to the power supply, unregulated +8 V and +16 V, and the need to provide a card case to hold the added cards. The graphics capability more than makes up for these two minor disadvantages.

For a long time, we have been looking for a 32 K expansion board which would permit us to build a really portable, two piece system, which could travel with us, under an airplane seat, if necessary. One piece would be our 9" Sanyo Monitor, in a protective case; the other, about the size and shape of an attache case, would hold the SYM, the KTM, the power supply, and the cassette recorder.

Stephen Cole, whose letter appears elswhere in this issue, is using the Beta Computer Devices Model 6502DM Memory Board. At the same time we received Stephen's letter, another reader wrote in, asking our opinion of the board. Not wishing to recommend an item we had not yet tried, we ordered one from Beta, at 1230 West Collins, Orange, CA 92668. Here are our comments:

The board requires only a single +5 V (at 0.8 A max) supply. There is no heat generating regulator on-board, and the board generates its own regulated -5 V and +12 V. The board is 4" x 6" and has edge fingers designed for the S-44 Bus (see elsewhere for some comments on this). The board has holes for mounting a (supplied) right angled 44 contact connector to fit the Reverse KIM bus. This means that the connector fits directly on the Expansion Connector edge fingers of the SYM. The connector can be fitted to the memory board in two ways, so the the board may be mounted either extended out and away from the SYM, or "folded" neatly and compactly below it. The latter mounting style is what we will use for our portable system. We will probably bring out a second connector, so that a disk controller card can be added for home use. We might then bring along our disk system as a third unit, if we travel by private auto.

The board is dynamic RAM only. It contains no ROM sockets, additional VIAs, or EPROM programming system. We find that, for the SYM, these "omissions" are of no consequence. The SYM is "loaded" with its own on-board VIAs, and the Hobart EPROM Programmer works off the Auxilliary Application Connector. All 32 K of RAM may be freely assigned in 4 K blocks, anywhere in empty memory space, with absolutely no constraints. This means that, if you have the 4 K RAM sockets on board filled, you will have 36 K of RAM available. The extra 4 K can be used to fill the gap at $9000-$9FFF; this would be a good location for all of the MON/BAS utilities you are using. The only free memory now available is the 2 K block at $F000-$F7FF, or, if you inhibit the System RAM "echo" at $F800-$FFF, the whole top 4 K block. One of the four on-board ROM sockets can be freed to hold one or two (piggy-backed) EPROMS. If you inhibit the echo, the top six bytes in EPROM must contain "fixed" IRQ and NMI vectors. You will have to give up the flexibility provided by IRQVEC and NMIVEC in System RAM.

To summarize, we like the Beta Board, and recommend it highly, if you need no additional graphics capability beyond that provided by the KTM-2/80 (160 x 48). For our fully portable system we are willing to accept this restriction.

HUEY II

Many readers have asked about adding a floating-point arithmetic package to SYM, which can be called from assembly language programs, or "patched" to the various "tiny" languages, e. g., tiny-c, tiny basic, tiny pilot, etc.

One approach, if you have BAS-1 resident, is to call on its subroutines, as required. We have not done this, but refer you to an article by R. M. Mottola, "MEAN 14: A Pseudo-Machine Floating Point Processor for the Apple II", in MICRO No. 28, September, 1980. The name "MEAN 14" is a parody of the name "SWEET 16", for the pseudo-machine 16-bit processor package in the Apple II. MEAN 14 can be adapted to SYM by replacing the Applesoft subroutine call addresses by the corresponding BAS-1 addresses. We have not done this, mostly for lack of time, but also because the real problem is to provide a free-standing floating point package for use without BAS-1.

The real answer is provided by Don Rindsberg's "HUEY II", available from the 6502 Program Exchange (address elsewhere in this issue). We have long been fond of the original HUEY, but did not recommend it earlier because we knew that the new version was in the works, and because the old edition of the manual was incomplete, requiring the user to locate, somewhere, a copy of the December 1977 Kilobaud! The new manual is now a self-contained document.

Huey II may be used alone to make the SYM-1 act like a Reverse Polish calculator, or its subroutines may be called from other high or low level languages. We recommend it highly, even if you don't ever use it. We suggest that you study its structure and its numerical algorithms, particularly if you are new to programming and computational methods.

FOCAL, FAST FOCAL, XPLO, AND TEC 65

FOCAL (FORmula CALculator) is a close relative of BASIC, with a number of elegant enhancements, originally developed by DEC for the PDP systems. TEC 65 is a 6502 version of the Text Editor, TECO. XPLO is similar to Pascal and C, all three being descendants from ALGOL.

All are available, in SYM cassette format (we tested them all), from the 6502 Program Exchange, 2920 Moana, Reno, NV 89509. Please write them (our contact there is Dave Marsh), for information on memory requirements, prices, and additional supporting software availability. Source code listings are available, unfortunately not in RAE format (hence not on SYM readable cassette), for all of these languages, so that you can easily adapt them to your own system configurations.

If you are deeper into software than into hardware, all ot these are worth owning for study and comparision purposes. Each has its own unique set of good and bad features. If hardware is your major area of interest, you should know that all languages are "equivalent" in the sense that any language can be made to do any job, although some may be more convenient or/and faster than others, in certain applications.

Our own feeling about languages, based purely on personal experience, and, of course, personal bias, is that, in the microprocessor environment, FORTH might have the edge in speed and convenience, particularly for control applications. Of course others, with equal or greater experience, have their own personal biases. The only honest recommendation that can be made here, or anywhere, is to get and try them all, and come to your own conclusion.

tiny-c

C is a 'Pascal-like' language developed by the Bell Telephone Laboratories. It seems to have a reasonably wide usage, but nowhere near the mass popularity of Pascal. tiny-C is an integer only version, available from tiny-c associates, P.O. Box 269, Holmdel, NJ, 07733, as "A Home Computing Software System".

We tried tiny-c, and liked it, and found the manual and documentation to be of outstandingly high, truly professional quality. The SYM-1 readable cassette version presents a few problems in reading and organization to produce a "Load and Go" cassette. We have reported the results of our tests, and our suggested fixes, back to tiny-c associates (tca), and assume that the fixes will be incorporated into future versions. We hope to make arrangements to market the SYM version for tca, but it is too early to report further on this at present.

There is neither the time nor the space to describe tiny-c here, and we are not reporting on prices, because of a rise in overseas mailing costs early next year, and because the price of a new text on tiny-c has not yet been announced. We suggest you write tca directly for any additional information.

MISCELLANEA

The reason that the back of each sheet of SYM-PHYSIS may seem to be "upside-down" with respect to the front side is so that the issues may be punched along the LONG edge for insertion into a three hole binder. If the binder is then turned 90 degrees clockwise, the pages are then all right-side up. This is not our own idea; we borrowed it from the KIM-1/6502 USER NOTES, because we liked their format. The issues are not pre-punched for you because we are doing our best to keep all costs down. Besides, about twenty percent of our subscribers are in "metric" countries, and we are not sure of the standards for the metric three hole punch.

We decided against carrying a "Beginners' Column" in each issue, because SYM-PHYSIS is sold only by the volume, not by the issue. By the time a new reader has finished Issue No. 2, and read some of the recommended books and articles he is no longer a beginner. A number of new subscribers originally called us nearly every day, when they first set up their SYMs, to ask very elementary questions, to which they could have found the answers in the Reference Manual (admittedly, that's not always easy!). Their calls became less frequent, and their questions became much more sophisticated and challenging, as the weeks went by. Now they call or write only to report on some new or exciting application or expansion.

We will be teaching a weekend course at the University of California at Davis (about 20 miles west of Sacramento) on December 5-7, 1980, on "Microprocessor Fundamentals." The $475 fee includes a "free" SYM. If you already have all of the SYMs you need, the fee is reduced accordingly. Please write or phone Garrett Jones, University Extension, University of California, Davis, CA 95616, (916) 752-2177. We plan to offer one or more similar courses at Cal State Chico, next spring. Please write Prof. O. S. Madrigal, Department of Computer Science, California State University, Chico, CA 95929, for additional information.

John R. Robertson, of Portland, Oregon, advised us of a company in Hong Kong that makes enclosures for the SYM and the KTM-2s. We have written them concerning possible import arrangements. We will report further in the next issue. If you can't wait till then, contact us after mid-December.

Tom Evans, WA6WTA, 20501 Hatteras St., Woodland Hills, CA 91367, would like to hear from any hams doing RTTY with the SYM.

Norrito Giorgio, 1200 Levin Ave., Mt. View, CA 94040, is interfacing an Exatron Stringy Floppy to his SYM.

Bruce Thompson, Cornell University, has been using four SYMs in remote stations to sense the geomagnetic and geoelectric fields and record them on cassettes. These run unattended for three days on 6 V car batteries. The timing and synchronization are critical so he runs them with an external oscillator which is buried in the ground to reduce diurnal rate variations. Since the sample periods are between .001 and 10 secords, he has T1 feeding T2 via PB7 connected to PB6 externally in order to get the range necessary. He says that the SYMs operate with ne'er a problem despite the 100 degree temperatures and 100% humidity.

Shahrakh Ghaffari, Chemistry Department, Oregon State University, Corvallis, OR 97331, sent us a note describing how to transfer BASIC programs from KIM to SYM. Those readers with both KIM BASIC and SYM BASIC may wish to contact him for the technique.

John Blalock asked us to mention that the prices for the W7AAY 4 K RAM Board, and the W7AAY RAE-1/2 ROM Board are now $8.00 plus a 15 cent (why isn't there a cent sign on an ASCII keyboard?) self-addressed, stamped envelope, and $16.00 postpaid, in the USA. Please order directly from him, P. O. Box 39356, Phoenix, AZ 85069. As a courtesy to our foreign subscribers, and for the convenience of those ordering other items from us at the same time, we will keep a small stock of both on hand at all times. Overseas, please add postage costs for one ounce, and three ounces, respectively.

PILOT is an extremely easy-to-learn CAI (Computer Assisted Instruction) Language, which younsters can learn to use nearly as soon as they are able to read and write. It has been placed in the public domain, and a number of "Tiny" Pilot versions are available for microcomputer systems use. Recent issues of MICRO contain a number of articles on 6502 Tiny Pilots, beginning with a SYM version by Nick Vrtis. Nick sent us a cassette version of a SYM Tiny Pilot, together with a new instruction manual, greatly enhanced and improved over the version originally published. We hope that Nick decides to market his new version. It will be of great value to those with young SYMmers in their households.

SUPER-SYMS?

At least one of our readers is working along the following lines:

1. Relocate SUPERMON to $F000-$FFFF
2. Reassign the I/O, etc., to $E000-$EFFF
3. Obtain from Carl Moser a relocated RAE at $C000-$DFFF (same as BASIC)
4. Use a second Beta Memory Board with only 16 K of RAM to fill $A000-$DFFF
5. Use a Blalock Memory Board to free 4 K from the first Beta Memory Board to be assigned to $8000-$8FFF
6. Call BASIC, RAE, FORTH, Pascal, and all other higher level languages in from disk as needed

This approach will provide a 56 K RAM/4 K ROM SYM system. To provide more RAM, memory bank switching is the next step.

One of our Computer Science graduate students is adding a Z-80 board, similar, in function, to that made for the Apple II by Microsoft, so that he can run CP/M and UCSD Pascal on his SYM. His system expansion bus will be S-100. We'll keep you posted on this one!

TAKE A 'BREAK'

Here are some 'trivia' on the BRK instruction and the B status bit in the status register. While BRK is usually considered a one byte operation in the 6502, in some ways it actually is a two byte instruction, with the second byte being ignored. If you look up the specs on BRK in the 6502 Programming Manual, you will see that BRK causes the Program Counter to advance by 2.

It is important in programming for KIM, if you wish to continue after a BRK stop, to put in a 'dummy' byte. In the SYM, the 'saved' value of the Program Counter is decremented by 1 in the SAVINT subroutine, so that a dummy byte is not needed. If you have both a KIM and a SYM, it would not hurt to follow each BRK with a NOP to make the programs more transportable.

You will learn a lot about the 6502 interrupt capabilities, and see that BRK, in effect, generates an IRQ in software, by studying the monitor interrupt subroutines from $800F to $80AC. You will see how MON makes returning from a BRK identical to returning from an interrupt. During the BRK, you may examine and, if you wish, modify memory and/or registers. Whether you got to this point by NMI, IRQ, BRK, or USRENT, reentry is via the zero parameter .G, through $83F3, which eventually returns you to the 'interrupted' program with an RTI, after restoring the departing conditions. Very elegant programming here. And we have not even discussed how program trace (for all non-MON instructions) is implemented through NMI! The trace program is also worth studying, to see how NMI is used.

IRQ and BRK treat the Program Counter differently. BRK increments by 2, since its instruction has been completed. IRQ does not increment, since it occurs just before the next instruction is to start. Otherwise, with but one minor, but important, exception, the 6502 handles IRQ and BRK nearly the same way. For both, three bytes, PCH, PCL, and P (the status register, i.e., flags) are popped on the stack. During IRQ the B flag is popped as a zero. Any other transfer of P to the stack, as with PHP, for example, or with BRK, pops B as a one. As was pointed out by a reader (can't remember whom; will credit him in the next issue, if he reminds me) the only place where this bit of trivia is documented is in Table 4-5 of the Reference Manual, where it is explicitly stated that PHP sets B to 1.

The 'expansion' bit (bit 5) also pops, and is therefore pulled, as a 1. Has anyone found a use for this bit, or some other way to set it (it clears on RST)? The overflow bit (bit 6) can be set from the outside world, if desired, by a negative going edge at the S.O. (also called RD) input (pin 38) on the 6502. This is brought out at E-5. Has anyone out there made use of this input?

HOW TO MAKE SYM EXECUTE YOUR COMMANDS

We have automated our cassette production by giving SYM the necessary commands to LOD selected files from disk and .S2 them to tape. This is done by making extensive use of SUPERMON's .E (Execute) command. For those of you who are not familiar with the .E command, we give two simple examples:

EXAMPLE 1: Suppose you wish to dump multiple copies of a program from $0200 to $0347 to cassette with ID = 01. Using either .M or .D (we prefer .D), and the ':' for ASCII input feature, enter the following sequence at, say, $0100:


```
.D 0100
0100 :S :2 :0 :1 :, :0 :2 :0
0108 :0 :, :0 :3 :4 :7 0D :E
0110 :0 :1 :0 :0 0D 00
```

The 0D is ASCII Hex for CR; the 00 is the terminator for any .E sequence. After entering the above, start your recorder going, hit RETURN, enter .E 0100, and hit RETURN again. Stop the recorder, and stop the recording with RST when you have made enough copies. You can obviously modify the sequence to dump selected blocks a fixed number of times. Before we put our Disk Bootstrap into EPROM, we had it on cassette. We used this technique to fill the full lengths of both sides of the cassette tape with the Bootstrap, so that we never had to waste time rewinding, or look to see which side was up.

EXAMPLE 2: A cassette save and load cannot be made over the top of page 0 (the cassette loader pointer is at FE,FF). Save and load over the top of page 1 is not a good idea, since it clobbers the stack. Thus, programs which require initialization in pages 0 and 1 should contain their own initialization subroutines. Two alternatives are: 1) Dump the program in three sections, and 2) Dump the program in one section, with the pages 0 & 1 blocks in higher memory, and move these blocks down with .B's before the .G .

The first alternative is 'automated' by writing a loading sequence including :L :2 0D :L :2 0D :L :2 0D 00 , and dumping it to casette preceding the three sections of the program. .L2 in the loading sequence, .E to its address, and it will bootstrap in the three sections of the program.

The second alternative is 'automated' by appending a move-and-go 'Execute' sequence to the program, which contains the necessary :B's to do the moving, and the :G to start the program. After the tape is read in, start the program, not with a .G to its starting address, but with a .E to the starting address of the 'Execute' sequence. Don't forget to terminate the sequence with "00", and to use "0D" for CR.

Now that you have seen examples of how to use .E, study its source code carefully, and you may discover how to extend its capabilities greatly, by writing your own version, and setting EXEVEC at $A672 to point to your version.

Note that .E can accept up to three parameters, but that the version in SUPERMON uses only one. Your version can use the other parameters to set vectors, print out messages, call subroutines, etc. Note that EXEVEC normally points to RIN, and that Execute essentially replaces INVEC(+1) with RIN, so that 'inputs' will come from RAM (or ROM, if you wish). You can do something similar with OUTVEC to steer outputs to RAM rather than to the terminal. You may now wish to reexamine the MERGE/DELETE Program for SYM BASIC on page 1-13 to see how this was done by Tom Gettys.

DISKS AND TAPES AND GRAPHICS AND APPLES

As mentioned elsewhere in this issue, the SYM can be taught to read Apple II generated tapes. In fact, on the 'other' SYM (the 'fun-one'), right now, there is a high resolution (B/W) picture of 'Hopalong' Cassidy, with an excellently simulated, quasi-halftone, gray scale. This was transfered to the SYM from one of the school's Apples, by dumping to cassette the memory-mapped image. The Apple II memory mapping is not, as we mathematicians say, a continuous one-to-one transformation, but the cassette read-in program makes the proper transformation. The Apple tape sounds different from a SYM tape; when we get the time, we'll compute and compare the data rates.

This little experiment was done to check out the read-in program, while waiting for two Apple Disks to be delivered to us by an extremely slow Apple Dealer, from whom we ordered them, over a month ago. These disks contain two "Magic Lantern" graphics packages for the Apple. When they arrive, we will read them into the Apple, dump the "frames" onto cassette, read these frames into the SYM, and dump them onto cassette (later onto disk, when our graphics SYM gets its own disk drives), for future enjoyment on the SYM. We paid the asking price for these disks and will be using them only on one system. The intermediate copies will be solely for the purpose of making the machine-readable medium readable on our machine.

The thought naturally occurs: Why not bypass the double cassette transfer, and fix up things so that our SYM can directly read the Apple Disks? We realize that the Disk II System is copyrighted, but then, so too, is the Apple's Cassette Firmware; yet a variant of this was published in the open literature. If we buy Apple Software to use on our SYM, it is obviously to Apple's advantage to let us be able to read the purchased software. Suppose we buy software written for the Apple by others; Apple likes to have others market software for the Apple, since the existence of such software helps to sell Apples. In our case, however, we would be using a modification of Apple's disk software, but with no advantage to Apple. This is rather a sticky problem area, no?

I'm not sure any one can really advise on the new (1978) copyright law, because there have been very few test cases on the principle of "fair use" of copyrighted material. Fair use implies non-commercial use, but only in ways which do not injure potential sales by the owner of copyright. Commercial use, i.e., piracy, would be easy to prosecute. It would be very difficult to prove that "wholesale" copying for distribution to club members, say, hurts the sales of a product, because the club members might not buy, if they had to pay.

Our own standpoint on the use of Apple software for the SYM is that we will buy such Apple (or Pet, or OSI, or whatever) software, for our personal use, if a) we want it, and b) we can figure out a way to read the purchased media with the SYM, and modify the software, as required, for SYM use. This in spite of, and possibly because of, the fact that some vendors of Apple disk software take advantage of the Apple Monitor and the DOS capability to cause the disk record to "self-destruct", if the purchaser should attempt to make a back-up copy.

Thus, in the very near future, we shall try to teach our old SYM a few new tricks, such as how to read an Apple disk.

REMARKS RE RAE-1

Carl Moser has provided us with a listing of the source code for RAE-1, so that we can more easily answer your RAE questions, and so that RAE NOTES NO. 3 can be more definitive about certain points. There is only one real "bug" that we have discovered in RAE. The pseudo opcode .EJ (essentially a "form-feed") falls one line short of the 66 lines necessary for an 11 inch form.

Many of you have objected to the "//" and the ">" prompt at the end of a manuscript. These can be suppressed by pointing OUTVEC to a patch which watches for these characters, and replaces them with nulls, before calling OUTCHR. If you want a (single) slash to appear as part of the text, your patch should store slashes each time they are sent out for printing, and wait for the following character to appear. If the following character is a second slash, suppress both; otherwise print both. If you wish the > to appear as a prompt on the CRT, but not on the hard copy, your patch should examine the "Hard Copy Flag" at $011F.

COMMENTS ON SWP-1

We have modified our own version of SWP-1 (with Tom Gettys' help) to suppress the word "PAGE" (if you want the word, put it in as part of the title or footer, as we often do, in lower case), and the leading zeroes in the page numbers.

Tom has also added the ability to continue a manuscript from either tape or disk. We are now editing Jack Brown's SYM-FORTH Manual with SWP-1, and it is still like magic to us to see SYM print out a 75 page manual with no human intervention, after the initial call to SWP.

We are not too happy with SWP's lack of a simple way to "TAB". The current way is to force spaces with a sequence of "up-arrows". Be careful with the use of spaces before and after up-arrows (best not to use them) since SWP "kills" spaces before them, and "transfers" all spaces after them, to the right-hand end of the line. SWP collapses all strings of spaces to a single space, except after a ".", where it prints a double space, to mark the end of a sentence. If you only want one space after a ".", use an up-arrow instead of a space. We leave it to the user to figure out how to get SWP to put two spaces between the "?" at the end of a sentence, and the first word of the following sentence.

We are studying the source code of RAE to see how it handles the tabs (only in steps of eight columns) to see if this method may be incorporated, or improved on for SWP. When this is done we will issue a SWP-2. To keep the faith with owners of SWP-1, the price of SWP-2 will be increased by the same amount that SWP-1 owners will be asked to send in to cover the costs of printing and mailing a listing of the enhancements to their current version. OK?

RECOMMENDED READING

Ever since we first begain using our KIM-1, we have looked forward to reading each of Professor Marvin L. De Jong's articles, first in the KIM-1/6502 User Notes, then in MICRO and in COMPUTE. We have long considered him to be "Mr. 6502", or, rather, "Prof. 6502". We were, then, very pleased to hear of his new book on the 6502. We are pleased, too, that Bob Peck agreed to review the book for all of us. We will add only one comment to Bob's review: If we could have only one book to go with our SYM, this would be the one.

PROGRAMMING AND INTERFACING THE 6502
with Experiments, Marvin DeJong
Howard W. Sams, 1980, $13.95

Robert A. Peck
DATAPATH, P.O. BOX 2231
Sunnyvale, CA 94087

I have been teaching assembly language programming for the past year at a local engineering college. As part of this teaching experience, I have collected quite a large number of books on the 6502 and other processors to try to determine the best approach to teaching assembly language programming. This book appears to have taken a different approach than any of the others I have seen.

Almost all other microprocessor books take the idea of introducing all of the addressing modes, then introduce the whole instruction set next. Then show subroutines, I/O devices, interrupts and so on. This information may occupy, as it does in many texts, as little as one chapter in the book with the rest dedicated in some way to applications of one kind or another. As an instructor, I have tried in the past to follow the outline of the texts I have used in this way but have come to realize this is a lot of data to throw at a student in one blob. I came away from earlier experiences believing that these other texts could serve the student as reference material once the student had been explained the techniques of this

type of programming but there was no text available which could lead the beginning student through assembly language programming without tossing everything at the student at once, making the concept itself even more difficult to understand.

Marvin has placed an understanding of the nature of the instruction stream as the prime goal of his book. He introduces the instruction set a few related instructions at a time. Each group is shown with as few basic addressing modes as needed for understanding of the function itself. Additional addressing modes are only shown as the progression of the instruction set explanation requires.

By a careful selection of the order in which the instructions are discussed, he is able to begin illustrating the functions of each by means of typical assembly language programs from the second chapter onward. By this means he is allowing the student immediate familiarity with the techniques the student will be using to generate his own programs.

Each grouping is then thoroughly discussed and illustrated by example and experiments which the student can do on a SYM or other 6502 unit. After the entire instruction set has been fully examined in this manner, he includes a section on hardware interfacing to the 6502 processor, continuing the emphasis on the programming aspects of this interface.

For anyone with an interest in learning the 6502 assembly language programming, with or without an instructor, I recommend this book highly. As an invitation to make your own judgement on the merits of this book, look at the table of contents of this book alongside that of any other 6502 programming book available today. You will see that it is more intensively dedicated to exactly the business at hand, that of showing the techniques of assembly language programming than the others. It does not include some of the filler material which is 'nice' to have at times but for a student to be able to judge what is filler and what is really necessary makes this book well worth its price in that it is all of value.

I intend to continue to use this as a text for the hardware and the programming courses in which I have future occasion to be involved. Based on Marvin's approach and the completeness of this text, I only hope that if someday I am called on to teach the Z-80, Z-8000, the 68000, or the 8086, I would hope that, by that time, Mr. DeJong might have had the opportunity to produce a text of this quality for this set of processors as well.

A BASIC WORD PROCESSING SYSTEM

Here are two BASIC programs submitted by KIN-PING KWOK, 22 Tung Choi St., Tat Ming Bldg., 10/F, Flat A, Kowloon, Hong Kong. We have not had the opportunity to test them, but the explanation of the programs, which appears following the programs, shows the right justified text which they produce. For those users who prefer BASIC to Assembly Language, or who, for any other reason, do not have RAE-1 installed, and thus cannot use SWP-1, these programs will provide a word processing capability.

```
10 ST=&"1000":DIM X(2):X(2)=8190+ST
20 GOTO 200
100 CH=PEEK(CS):IF CH=0 THEN NU=0:GOTO 200
110 IF CH=13 AND NU<>0 THEN CN=CA:GOTO 200
120 IF CH=13 THEN NU=-1:PRINT
130 DI=USR(&"8AA0",CH*256)
140 CS=CS+1:NU=NU+1:CA=CA+1:IF NU<400 THEN 100
150 CN=CA
200 CH=INT(USR(&"8A58",&"D14B",0)/256) AND 127
```



```
210 IF CH<32 THEN 500
220 POKE ST+CA,CH
230 CA=CA+1
240 IF CA>8000 THEN PRINT CHR$(7);
250 IF CA>8190 THEN CA=8190:PRINT CHR$(8);
260 GOTO 200
500 IF CH>7 AND CH<14 THEN ON CH-7 GOTO 1000,1050,1100,1150,200,1200
510 IF CH=5 THEN 2000
520 IF CH=17 THEN IF NU=0 THEN 200
525 IF CH=17 THEN NU=0:CA=CN:GOTO 100
530 IF CH=6 THEN X(0)=CA+1+ST:X(1)=X(0)-1:CN=CN+1:CS=CS+1:GOTO 550
540 IF CH<>2 THEN 200
545 X(0)=CA+ST:X(1)=X(0)+1:CN=CN-1:CS=CS-1
550 FOR DI=0 TO 2
570 CH=INT(X(DI)/256)
580 POKE 42574-2*DI,X(DI)-CH*256:POKE 42575-2*DI,CH
590 NEXT
595 DI=USR(&"8740",0)
600 X(0)=CN-CA:IF X(0)<1 THEN 200
610 FOR CH=0 TO X(0)-1
620 DI=USR(&"8AA0",256*PEEK(CA+ST+CH))
630 NEXT
635 PRINT " "+CHR$(8);:POKE 25,0
640 PRINT CHR$(27)+"+";:X(1)=INT((960-X(0))/40)
650 PRINT CHR$(32+X(1))+" ";
653 DI=960-X(0)-X(1)*40:IF DI=0 THEN 200
655 FOR CH=0 TO DI-1:PRINT CHR$(9);:NEXT
660 GOTO 200
1000 CA=CA-1:IF CA<0 THEN PRINT CHR$(9);:CA=0
1010 GOTO 240
1050 CA=CA+1:GOTO 240
1100 CA=CA+40:GOTO 240
1150 CA=CA-40:IF CA<0 THEN PRINT CHR$(10);:CA=0
1160 GOTO 240
1200 PRINT:POKE ST+CA,CH:GOTO 230
2000 IF NU=0 AND CA>0 THEN POKE ST+CA,0
2005 PRINT
2010 INPUT "COMMAND ";IN$
2020 IF IN$<>"L" THEN 2010
2030 CA=0:CS=&"1000":NU=0:GOTO 100
OK
```

```
10 N=70
11 L=10
12 A=0:A$="":B$="":B=0:CS=&"1000"
13 I=1:C$=A$:C=0:P=0:Z=0:T=2
18 X8=59:X9=63:X0=13:X1=32:X3=43:X4=48:X5=33:X6=41:X7=58:E$="  "
30 FOR A=LEN(A$)+I TO N
40 B=PEEK(CS):A$=A$+CHR$(B):CS=CS+I:IF B=Z THEN END
45 IF B=X0 THEN C$="":GOTO 500
47 NEXT
48 A=A-I:B=ASC(MID$(A$,A,I))
50 IF(B>X3ANDB<X4)ORB=X7ORB=X8ORB=X9ORB=X5ORB=X6THENP=A:GOTO 80
60 IF B=X1 THEN P=A-I:GOTO 80
70 GOTO 48
80 BL=Z:B1=Z:C$=""
90 B=PEEK(CS):C$=C$+CHR$(B):B1=B1+I:CS=CS+I:IF B=Z THEN END
96 IF B=X1 OR (B>X3 AND B<X4) OR B=X5 OR B=X6 THEN 110
100 IF B<>X7 AND B<>X8 AND B<>X9 THEN BL=I:GOTO 90
110 IF B=X1 AND BL=Z THEN P=N:GOTO 90
120 IF P=N THEN 500
130 D$=A$
140 C=N-P
145 A=I
150 B$=MID$(A$,A,I):IF B$="." OR B$="," OR B$=";" THEN 160
155 IF B$<>":" AND B$<>"!" AND B$<>"?" THEN 180
```

```
160 IFMID$(A$,A+I,T)<>E$THENA$=LEFT$(A$,A)+" "+MID$(A$,A+I):C=C-I:P=P+I
163 IF C=Z THEN 170
165 IFMID$(A$,A+I,T)<>E$THENA$=LEFT$(A$,A)+" "+MID$(A$,A+I):C=C-I:P=P+I
170 IF C=Z THEN C$=MID$(A$,N+I)+C$:A$=LEFT$(A$,N):GOTO 500
180 A=A+I:IF A<P THEN 150
200 A=I
210 IF MID$(A$,A,I)<>" " THEN A=A+I:GOTO 250
220 IFMID$(A$,A,T)<>"  "THEN A$=LEFT$(A$,A)+" "+MID$(A$,A+I):C=C-I:P=P+I
230 IF C=Z THEN 170
240 A=A+I:IF MID$(A$,A,I)=" " THEN 240
250 IF A<P THEN 210
260 A$=D$
270 IF B1=I THEN C$=RIGHT$(A$,T)+C$:A$=LEFT$(A$,N-T)+"-":C=I:GOTO 145
280 C$=RIGHT$(A$,I)+C$:A$=LEFT$(A$,N-I)+"-"
500 PRINT TAB(5)+A$
510 A$=C$
520 IF LEFT$(A$,I)=" " OR LEFT$(A$,I)=CHR$(X0)THEN A$=MID$(A$,T):GOTO520
530 GOTO 30
OK
RUN
```

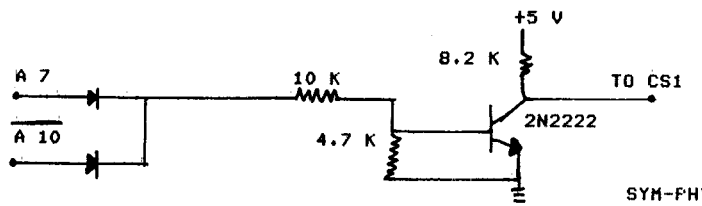
The first program is for input and edit. It store ASCII start from $1000. You can use cursor control to edit the input passage. Type CTL-E to end the input. If you type L to reply COMMAND, the passage will list for you to edit. It stops when it meet a CR or zero or it had list 400 characters. Use cursor control and CTL-F and CTL-B to edit, insert and delete. Varaible N in the second program is the number of characters per line. Since the programs are written in BASIC, the speed of the program is not very fast. You can change them into machine code programs to increase the speed and that is not very difficult. At last, the programs are written for KTM-2/40, change 40 to 80 in line 640,653 in the first program for KTM-2/80.

HOW TO 'REWIRE' THE VIA AT $A800

As mentioned elswhere in this issue, additional VIAs may be addressed in the $A800 block by inhibiting the enabling of U28 when address bit A7 is high.

Lev Barshay, of Nestle, elected to cut the existing $\overline{AA8}$ trace between pin 6 of U10 and pin 23 of U28 (VIA #2). Note that the latter is the $\overline{CS2}$ input, which is active low. The new input to pin 23 is obtained by "or"-ing A7 and $\overline{AA8}$ (active low) in 1/4 of an open collector 74LS32, and using the existing R60 as the pull-up resistor. Note that if this is done properly, VIA $\overline{\#3}$ at $AC00 is also inhibited, since its pin 23 shares the same AA8 and pull-up resistor. Lev mounted the 74LS32 by cementing it, pins up, to his SYM, near U24. He obtained $\overline{AA8}$ from pin 6 of U10, and A7 from pin 1 of U20, although other convenient sources of these signals could have been used. The output of the "or" circuit goes to the "low" end of R60.

Darrell Johansen, of Sense Modular, decided to work with pin 24 of U28. This is CS1 for VIA #2. He cut the existing trace to pin 24, and used one of the four buffers in the lower left corner of SYM to fashion the following "nor" circuit. Note that he now uses $\overline{A10}$, not A10, as previously, because of the inversion provided by the "nor".



THOUGHTS ON SMALL SYSTEMS AND MONITORS [MOSTLY THE SYM-1]

The title above is not ours; rather it is the title of an article by H. T. Gordon, in Dr. Dobb's Journal of Computer Calisthenics and Orthodontia, Number 48, September (?), 1980.

We personally read each issue of Kilobaud Microcomputing, MICRO, BYTE, and COMPUTE., and call to your attention any SYM relevant material. Tom Gettys reads many of the other computer magazines, and called the above mentioned article to our attention. Some of Professor Gordon's thoughts are worth quoting here; we shall do so, but recommend that you read the entire article, if you can:

> "The concept that has always appealed to me is more that of the "hi-fi" enthusiasts, where a system consists of several interacting but independently-replaceable components, from many competing sources. True competition, however abhorrent to industrial giants, makes possible a maximum of progressive change at the minimum cost to society. This eccentric, unbundled-components point of view enabled me to resist the lure of the increasingly powerful packaged systems that have entered the market in recent years. However flexible, they must be designed for some least-common-denominator purpose and tend to allocate large resources to things like BASIC interpreters (for me a turn-off). They are like low-cost smoothly-paved roads leading where everyone else wants to go."
>
> "Even the minority of programmers who have read the ROMS created by someone else do not fully appreciate the problem until they toy with the idea of creating their own. It's the chasm between critic and artist, or rather between a builder of sand-castles and a sculptor in marble."

Professor Gordon goes on to discuss monitor ROMS, and terminals, and related topics. His article inspires me to disassemble the KTM-2/80 ROM to see what kinds of enhancements could be added by replacing the ROM with my own EPROM. He closes with the following thought:

> "A brief afterthought on FORTH. To me, FORTH has been the most tantalizing of the existing HLLs. Perhaps the word is infuriating, since the FORTH enthusiasts - like the Rosicrucians or the initiates to the ancient Eleusynian mystery - won't tell you what it is. In comparison, my own much more miniscule programs come with a surfeit of explanatory comment (tinseled with allusions). When I glanced at a listing of FORTH, its most striking quality was the virtual absence of comment. Adam Osborne recently observed (InfoWorld 2(8):7, 1980) that the success of an HLL depends less on its intrinsic merit than on how hard it's pushed. Whatever the demerits of BASIC - and they are legion - being unexplained is not one of them. Dozens of books expound it in great detail, and some are brilliant. Where is the book that describes how FORTH works, from the ground up, in a painstakingly detailed, translucent and vivid way?"

By HLL, Dr. Gordon means higher level languages. We also very much like his use of the term "machine-linguists" for machine language programmers. Our answer to his closing question is Jack Brown's Manual for SYM-FORTH. Jack Brown's FORTH programs are more heavily commented than most of the BASIC programs we have seen lately. We will send Professor Gordon a preliminary copy of the SYM-FORTH Manual for his review and comments.

MORE ON TEXT EDITORS AND WORD PROCESSORS

We had just this morning delivered the camera-ready copy for a 44-page issue of SYM-PHYSIS to our printer, when we received a cassette in the mail from Frank Winter, whose article on TOPS we published in Issue No. 4. (See pages 3-25 and 4-12.) After looking over his word processing program on cassette, in RAE format, we decided to "stop the presses", and publish his letter and these comments. He is therefore responsible for this issue containing an extra four pages more than we had planned.

We like SYM mainly because of its RAE and SUPERMON firmware, and we use RAE as our text editor, because it is there!!! With SWP-1 added, our requirements for a word processor are fully satisfied. That doesn't stop us from examining and appreciating others, however.

We realize that many of you have no need for an assembler, so you have no need for the "A" part of RAE (Resident Assembler Editor). What other options are open, if you just need an Editor? If you have BAS-1, you can use BASIC for your word processing needs. That is why the BASIC word processing program by Mr. Kwok appears in this issue.

But what do you do for a word processor, if you don't have either BASIC or RAE? Very simple; just get a copy of TEC 65 from the 6502 Program Exchange (address is elsewhere in this issue). TEC 65 is a really fine text editor. We publish Frank's letter to show you how TEC 65 can be greatly extended. We will send a listing of the source code to Dave Marsh of the Program Exchange. Perhaps we can work out a three way arrangement to provide our readers with Frank's Enhanced SYM Version of TEC 65. Are any of you interested? Here is Frank's letter:

Dear Lux,

I enclose a copy of some modifications to the TEC 65 text editor, which I understand you purchased some time ago.

I was quite impressed with the capabilites of this language, but found the lack of formatting a problem. This arises when you alter the original text, and you still want a neat printout.

The enclosed program was specifically written to link up with TOPS (by the way thanks for publishing my comments). I don't think that it would be very difficult to change it to suit your disk operating system.

The following commands are available when you activate the formatter:

\T10\ sets the top margin to 10 lines
\B10\ sets the bottom margin to 10 lines
\L5\ sets the left margin to 5 character spaces
\R58\ sets the line length to 58 characters
\P\ sets the page length to 60 lines
\I10\ breaks and indents 10 characters
\E1\ ejects TWO pages. \E\ ejects to the end of the current page
\M1\ breaks and sets line spacing to two (ie one blank between lines)
\S2\ breaks and spaces two lines regardless of the invoked linespacing

When the formatter is not activated the text is printed exactly as it resides in the buffer.

My doctoral thesis is entering its final stage, and I plan to submit it

by February 1981. My next project is a text for Operations Research, which will emphasise the use of personal computers.

I hope you are over the major hurdles of getting your book published, and I look forward to the next issue of SYM-PHYSIS. In the meantime, kind regards from down under. It is getting bloody hot now - I really should get my office air conditioned!

Frank....

[Editor's note: Winter is enjoying (!) summer in Australia right now!]

MYSTERY PROGRAM

Here is a BASIC program which looks interesting, and quite useful. We don't know who submitted it, because it somehow got separated from its transmittal letter. We vaguely remember writing the author, to ask if he would mind resubmitting on cassette, because we were too busy to key it in for test and reproduction. Our filing system is such that the original letter has been misfiled forever, and we don't recall ever getting a cassette. If the author will let us know, we'll give him full credit in the next issue. Incidentally, here is a good example of incomplete documentation. Every program you write should be "signed and dated" in a comment line (and even copyrighted, maybe).

NOTE: This program must be used with BBE-1 (Brown's BASIC Enhancements).

```
100 CLEAR
110 PRINTCHR$(27)+"E"
120 GOSUB 680:DIM A$(50):DIM N$(50):DIM P$(50)
130 PRINTCHR$(27)+"E"
140 GOSUB 680:PRINTTAB(10);"**MENU**":PRINT:PRINT
150 PRINT"TO BUILD A FILE TYPE 1"
160 PRINT"TO SEE FILE-TYPE 2"
170 PRINT"TO SEE INDIVIDUAL NAME TYPE 3"
180 PRINT"TO CORRECT-TYPE 4"
190 PRINT"TO SAVE FILE-TYPE 5"
200 PRINT"TO GET FILE FROM TAPE TYPE 6"
210 INPUT Q:ON Q GOTO 220,320,350,430,580,640
220 INPUT"WHEN READY HIT RET (TO CLOSE THE FILE TYPE END FOR NAME)";X
230 FOR I=1 TO 50:PRINTCHR$(27)+"E":GOSUB 680:PRINT"ENT NAME"
240 PRINT"HIT 'RETURN' KEY";:INPUT N$(I)
250 IF N$(I)="END" THEN P1=I:GOTO 300
260 INPUT"ENT ADDR";A$(I)
270 INPUT"ENT PHONE #";P$(I)
280 IF FRE(X$)<100 GOTO 300
290 NEXT
300 PRINT"FILE CLOSED--":INPUT"TO SEE MENU,HIT 'RETURN'";X
310 GOTO 130
320 PRINTCHR$(27)+"E":GOSUB 680
330 FOR I=1 TO P1:PRINTI,TAB(7)N$(I),A$(I),P$(I):NEXT
340 INPUT"TO SEE MENU HIT 'RETURN'";X:GOTO 130
350 PRINTCHR$(27)+"E":FOR E=1 TO 10:NEXT:INPUT"ENT NAME";N$
360 FOR I=1 TO P1:IF N$(I)=N$ THEN 390
370 NEXT
380 PRINT"NAME NOT IN FILE":GOTO 400
390 PRINTN$(I),A$(I),P$(I)
400 PRINT:PRINT"FOR CONT. TYPE 1, TO STOP TYPE 0";:INPUT X
410 IF X=1 GOTO 350
420 GOTO 130
430 PRINTCHR$(27)+"E":GOSUB 680
440 PRINT"ENTER THE LINE'S NAME TO BE CHANGE":INPUT N$
450 FOR I=1 TO P1:IF N$=N$(I) GOTO 480
```

```
 460 NEXT
 470 PRINT"NOT IN FILE":GOTO 550
 480 PRINT"ENTER CORRECTED INFO."
 490 INPUT N$(I),A$(I),P$(I)
 500 FOR T=1 TO P1
 510 IF T=I THEN T=T+1
 520 IF N$(I)=N$(T) THEN PRINT"EXIST ON LINE";T:PRINT
 530 NEXT
 540 PRINT:PRINT: PRINT"THE LINE NOW IS:":PRINTN$(I),A$(I),P$(I)
 550 INPUT"FOR CONT. TYPE 1, TO STOP TYPE 0";X
 560 IF X=1 GOTO 430
 570 GOTO 130
 580 PRINTCHR$(27)+"E":GOSUB 680
 590 INPUT"MAKE PREP. FOR CASSETTE, WHEN READY HIT RETURN";X
 600 PRINT"COPYING"
 610 .SAVEV 1
 620 PRINT"DONE"
 630 INPUT"TO SEE MENU, HIT RETURN";X:GOTO 130
 640 PRINTCHR$(27)+"E":GOSUB 680:INPUT"WHEN READY, HIT RETURN";X
 650 PRINT"LOADING DATA"
 660 .LOADV 1
 670 PRINT"DATA LOADED":INPUT"TO SEE MENU, HIT RETURN";X:GOTO 130
 680 REM *** TIME DELAY RUTINE FOR CLEAR SCREEN***
 690 FOR E=1 TO 10:NEXT:RETURN
OK
```

MORE ON SOUNDS AND MUSIC

We see from the 1981 Radio Shack Catalog, not only that TI's SN76477 has gone up in price, but more importantly, that TI has introduced a new sound generation chip, the SN76488, more amenable to computer control, at $6.99 (RS 276-1766). If any of you try it, please let the rest of us know your results.

We used to think that the MTU Advanced Music Package was the greatest thing in the computer music business since the invention of the square wave. But, not any more! Now we feel that the MTU Advanced Real-Time Music Synthesis Techniques Package has taken its place. We suggest you read Hal Chamberlin's article (with the above title) in BYTE, April 1980, and, if this interests you, to send for the stereo audio demonstration cassette, which we have available. The MTU package, with full source code listing, and three demonstration scores, is available from the Users' Group in SYM readable format. The program is memory intensive. Only one of the three demos will work in a 8 K system; the other two require 16 K and 32 K, respectively.

KTM-2 TO KTM-2/80 CONVERSION

Bob Myers called today, just in time to get into this extra page, to tell us that he is now ready to start shipping the KTM-2 Upgrade Kit. The Kit includes two Synertek ROMS, full instructions, and 'artwork' to show where to make the trace and jumper modifications. The cost for the Kit is $65, plus shipping charges. You will need to buy sockets, a pair of 2114s, and a pair of other ICs, in addition to the parts supplied with the kit.

Bob asks us to advise those who have written and received no answer from him, that, while he was on an extended business trip, his office was moved from one building to the next, and that many of his papers got "lost" during the short haul. Please write him again with any questions, or to place your order. His address is on page 4-23.

>

AND SOME WORDS ABOUT WORDS

If you have interfaced TI's "Speak & Spell" to your SYM, you will be interested in the Phoneme Software Package, available from S.PEE.K uP SOFTWARE, 6710 Forest Bend, San Antonio, TX, 78240.

MORE ON DISK SYSTEMS

Quite a few of our readers are beginning to add HDE File Oriented Disk Systems (FODS) to their SYMS. We (Tom Gettys and I) provided Dick Grabowsky of Hudson Digital Electronics, Inc. with Version 1.0 of our SYM-FODS software package. Since that time SYM-FODS has been extended considerably. The major extensions have included adding .CT (name) to RAE-1 to permit Continue on Disk, and .DISK (filename) to SWP-1 to permit concatenating discrete files into one long sequentially page numbered document (fortunately, just in time to handle Jack Brown's very thorough (and very long) SYM-FORTH Manual).

SYM-FORTH has been designed from the first to work with a simulated (cassette/RAM) Disk System, and can easily be patched to "for-real" Disk System. We are making arrangements with HDE to provide SYM-FODS users with all extensions on a timely, non-profit basis.

CHECKOUT TIME

May we wish all of you the appropriate Season's Greetings, and a Happy New Year, even though a little early? Issue No. 7 will reach all re-subscribers in late January or early February. Meanwhile, we will concentrate first on answering the pile of old letters, some nearly a month old. They come in at the rate of 2-3 per day, so there must be perhaps 70 or so. Next, RAE NOTES NO. 3.

Teachers glory vicariously in the achievements of their students, in the same way that parents do with their offspring. Two years ago, in our KIM days, Steve Crescenti developed the software for a laser graphics system, tested on an oscilloscope. During the past year, Tom Gettys developed the foundations for a SUPERMON Extension Package, and Paul Close implemented a Voice Recognition System (12 word vocabulary, cooperative speaker, 95% recognition). These tasks were parts of their Master's Degree projects. Two have gone into industry; Tom is teaching at Cal State, Chico.

This current year, Hamid Kahansi is well into his project of Apple II/SYM-1 Complete Information Interchange (ASCII?), involving cassette, disk, and RS-232 subsystems. Peggy Leung, our first woman student to become interested in micros, is using the SYM as an intelligent controller interface between an H-P System and a surplus incremental plotter. Also, several students, both men and women, in our Industrial Technology major, with strong electronics backgrounds, are beginning to think micro-digital! During the next year there should be many interesting results to report to you.

We know that we ourselves will not be able to advance the state-of-the-art of Computer Music, Computer Speech, Pattern Recognition, and Image Processing significantly; we hope that some of our students may. We do hope to become very skilled in FORTH, because of our feeling that FORTH will enable us to accomplish our system design goals more rapidly than any other programming tool (to us languages are merely tools for communication).

We will close with the same words that Jack Brown used in a recent letter to us: "May the FORTH be with you!"