```
 SSSS  S   S S     S     SSSS  S   S S   S  SSSS  SSS  SSSS
S    S S   S SS   SS     S   S S   S S   S S    S  S  S    S
S      S   S S S S S     S   S S   S S   S S       S  S
 SSSS   SSS  S  S  S SSS SSSS  SSSSS  SSS   SSSS   S   SSSS
     S   S   S     S     S     S   S   S        S  S       S
S    S   S   S     S     S     S   S   S   S    S  S  S    S
 SSSS    S   S     S     S     S   S   S    SSSS  SSS  SSSS
```

T H E   S Y M - 1   U S E R S '   G R O U P   N E W S L E T T E R

ISSUE NUMBER 4 - JULY/AUGUST 1980

SYM-PHYSIS is a bimonthly publication of the SYM Users' Group, P. O. Box 315, Chico, CA. 95927. SYM-PHYSIS and the SYM Users' Group (SUG) are in no way associated with Synertek Systems Corporation (SSC), and SSC has no responsibility for the contents of SYM-PHYSIS. SYM is a registered trademark of SSC. SYM-PHYSIS, from the Greek, means the state of growing together, to make grow, to bring forth.

We welcome for publication all articles dealing with any aspect of the SYM-1, and its very close relatives. Authors retain all commercial copyrights. Portions of SYM-PHYSIS may be reproduced by clubs and educational institutions, and adaptations of programs for other computers may be freely published, with full credit given and complimentary copies provided to SYM-PHYSIS and the original author(s). Please include a self-addressed stamped envelope with all correspondence.

Editor/Publisher:	H. R. 'Lux' Luxenberg
Business/Circulation:	Jean Luxenberg
Associate Editors:	Thomas Gettys, Jack Brown

SUBSCRIPTION RATES:
------------ -----

USA/Canada $9.00 for a volume of 6 issues; [illegible] $12.50. Make checks payable in US dollars to 'SYM Users' [illegible] O. Box 315, Chico, CA 95927, Telephone (916) 895-8751.

PLEASE USE LATEST 'GREEN SHOPPING LIST'

The Introductory [illegible] umber 0, is not included as part of the first year's subscr[illegible] but is available separately, postpaid first class or air mail, for $1.50 in US/Canada and $2.00 elsewhere.

FROM THE EDITOR
---- --- ------

As many of you have discovered, by writing or phoning, the Users' Group is able to provide answers to most of your questions, and solutions to many of your problems with SYM and its peripherals. We try to get our mail answered within a week, and many days (weekends, too) we are on the phone for several hours with SYMmers with problems. It is actually getting easier to provide the help, because the problems have a large commonality factor, and it is still more fun than work, because of the information exchange, and because much of what we learn from SYM problems carries over to other areas of interest, and helps us better understand and improve our own three SYMs, a 'his' and a 'hers' and a 'theirs', which is at the campus for student use.

Thanks to the many readers who have sent in notes, hints, programs, tips, articles, modification suggestions, hardware recommendaions, etc., which we had no room to publish. Please rest assured that we have studied them all, and did find it very hard to make the selection. We have them in an 'easy access' file, and will pass on the information to callers and writers to whom your material would be of immediate value.

In Issue #5 we will describe, compare and evaluate the merits of the various higher level languages now available for the SYM-1 (in addition to BAS-1).

THE AUDIO-VISUAL SYM
--- ----------- ---

We checked out the Atari 400 and 800 the other day, and were impressed with Atari's approach to Computer Assisted Instruction (CAI). We pass on to you their audio-visual concept for implementation on your SYM.

After dumping your program to cassette tape, add music, sound effects, questions, answers, etc., using the mike and/or an audio patch cord. Your program can then start and stop the cassette with the commands X=USR(&'8DB9',0) and X=USR(&'8D4E',0). We have found it very impressive to provide a musical background of computer synthesized music with abstract graphics on our high resolution MTU Visible Memory. We feel that the addition of audio to the visual display can very much enhance the educational process, as well as provide 'mere' entertainment.

Now for the technical details: Atari 'plays' its cassette input through the TV receiver sound system. Unless your video monitor has audio input you will have to find another way. Our approach is to connect a small hi-fi amplifier to the Audio In pin on the Applications connector. This provides the added bonus of permitting audio monitoring of your cassette read-ins; very useful when adjusting your system to read other people's tapes.

Once program tapes contain audio as well as digital data, they cannot be duplicated through the SYM, for distribution, as we have been doing for the tapes we are marketing. As mentioned in an earlier issue, we found it practical to dub SYM cassettes from one recorder to another, using an attenuating patch cord available at Radio Shack. We have now acquired a 3M 10KHz sine wave head azimuth alignment tape. After we have precisely gotten our heads on straight, we will make some SYM-1 Synch Signal Head Alignment Tapes for quality control purposes.

Incidentally, if you need a good small hi-fi amplifier for the purpose described above, let us recommend the MTU DAC board, which contains the 8-bit Digital to Analog Converter, as well. The audio amplifier has several inputs, as well as scope outputs, so that you can also monitor your cassette input visually. See back page for special pricing information. The DAC board does not require any special bus, since it connects directly to the Applications connector. We are using two of them now for oscilloscope graphics (X-Y deflection), and soon for laser graphics.

APPLES VERSUS ORANGES - APPLE VERSUS SYM
------ ------ ------- - ----- ------ ---

The Apple II is a well engineered, highly cost-effective, small-system computer, designed with great imagination, and with easy to implement expansion capabilities. I purchased a copy of the new Apple II Reference Manual, to replace my old, worn-out, copy of the 1978 edition. Their quality of documentation has been so upgraded that the manual is now practically self-explanatory! It was a real pleasure to read, which I did to see which of their design approaches could be applied to the SYM-1.

The Apple II and the SYM-1 are both great, but should not be directly compared, because they are different kinds of systems. The Apple II is a small-system computer, the SYM-1 is a single board computer. They are intended for different end-user types. The SYM-1 is intended more for the do-it-yourselfer, who is willing to put in many hours of time to save the initial outlay of dollars (or marks, or pounds, or francs; we have many overseas readers). More importantly, it is for the person who can learn most rapidly through the self-doing. These were my reasons for choosing SYM-1 over Apple II.

(continued on page 25)

MORE FROM JACK GIERYIC

Following are portions of two letters, two articles, and a BASIC program written by Jack Gieryic, 2041 - 138th Ave. N. W., Andover, MN 55303.

Jack is one of our most prolific correspondents, and we have a large file of written material from him. Originally, he was sending printed listings of his BASIC programs (see the example below to see how time-consuming they could be for a two-fingered typist to enter!), until we suggested that he send cassettes, instead. Now that he has RAE-1 on his system, his letters and articles are also coming in on cassettes, which are immediately transcribed to disks for editing and storage. How much easier life is becoming with SYM-1, RAE-1, SWP-1 and FODS!

Dear Lux:

SUBJECT: Bill Gowans' Hi-Density Plot Routine

I have discovered the lower right character position should not be accessed by user calls to the "SET" entry point of Bill Gowans' routine. The reason for this is after the keyboard is positioned to and writes this character, the cursor is then positioned at the first character of the next line. Since this "next line" is the 25th one, this results in the entire screen scrolling up one line. Now Bill's memory map no longer reflects what the user sees.

Any program using Bill's routine should avoid the following X-Y coordinates where Y is the virtual row and X is the virtual column.

KTM-2

X	Y
78	46
79	46
78	47
79	47

KTM-2/80

X	Y
158	46
159	46
158	47
159	47

The following code inserts this fix into the demonstration program included with the article from issue 3. This fix will maximize the CRT usage. The problem doesn't occur in the demo because the lower right character position is not referenced. I only want to show how to fix the "bug" in the event others wish to write their own interface program.

```
192 IF Y < 46 THEN 200
194 IF X > 157 THEN 210
```

Your Avid Reader,

Jack Gieryic

Jack Gieryic

Dear Lux:

First a note on the 4 QUADRANT PLOT program I sent you on the previous tape. If possible, I would like to see Bill Gowans' code relocated to under 4K, and line 1 of my program adjusted accordingly. Copies of the program would then be made under the Monitor such that the BASIC and machine code would be in the same file. That way, the user would just do a single load under BASIC, and both portions would be loaded. Also, the program would fit in 4K. The user would also be given the correct value to use for MEMORY SIZE at log-on time. Is this too much to ask?

Now for this tape: I have included a copy of yet another JACK BUILT PROGRAM....my Graphics Demo Package 2. If you liked the random pattern generator of the first package, then you'll love this program. The only big difference with this program, as compared to all the others, is that it requires about 5K of memory. Maybe you also want to publish this one in the next issue of SYM-PHYSIS.

On that note, I have also included two articles I'd like you to take a look at. If you want to use them in SYM-PHYSIS, then feel free. If any changes are required, let me know. I hope they are acceptable. I now realize why you only publish every other month. This stuff is extremely time consuming.

Sincerely,

Jack

CURSOR POSITIONING/GRAPHICS PRIMER
with KTM-2 Examples
by Jack Gieryic

The hobbyist's CRT is probably the most under-used item of his/her computer system. Many program displays are very primitive, consisting mainly of data generated from print statements. With the aid of cursor positioning, some graphic characters, and a heavy helping of imagination, the user can make dramatic changes in the area of data presentation. This article addresses the subjects of terminal graphics and X-Y cursor positioning. It will aid the hobbyist who has a terminal with these capabilities, and enlighten those who are considering purchase of a terminal, or who are lacking knowledge on these subjects.

Cursor positioning can be thought of as the ability for a terminal to immediately pick up the cursor and place it at any character position on the CRT. This is done by sending a 'single' command to the terminal, which the terminal interprets as a special function. The command usually consists of several characters. It will not be displayed as text, but will instead cause the terminal to perform the selected function.

The first commamd of the string is a non-text character. The terminal is constantly interrogating everything sent to it. When this special non-text character is found, the terminal then gears up for receipt of one of its special function commmands. It knows the next few characters are to be commands even though these characters may be text (alphabet, numerics, punctuation).

In the case of a terminal with several different commands, the next character will tell it which command is selected. If this second character indicates cursor positioning then the next two characters may be the X position (which character on the line) and the Y position (which line on the CRT). At this point the terminal knows the command sequence is finished. No more information is needed to perform the function, nor is a special end-of-command character required. The cursor now appears at the selected position. If a text message follows, then the text appears on the CRT beginning at the selected cursor position. Unfortunately, all terminals may not operate in this EXACT sequence but the general concept of a 'special command' still holds.

Graphics is a much broader subject to address, as each terminal has its own set of graphic characters, if it has any at all. One character may cause a terminal to display one particular graphic character while on another terminal the same character may result in a radically different character or none at all. There seems to be no uniformity in the industry concerning graphics. The number and set of characters varies from terminal to terminal.

A drawing of the various graphic characters will show a particular terminal's capabilities. The user's vivid immagination is the best means of evaluating the character set. Can fine lines (less than a character width) be drawn both vertically and horizontally? Can the character field be split into finer pieces? Can any special and graphic characters be mixed on the CRT? Questions of this type will help in evaluating a terminal's capabilities.

The remainder of this article uses commands and sequences specifically for Synertek's KTM-2/80 keyboard. The command codes for other terminals may be different, but the approach should be similar. The reader who does not own a KTM-2/80 should try to extract the approach out of the remainder of this article. For you other lucky ones, this can serve as your first lesson.

To enter the graphics mode of operation the escape character is first sent to the terminal, followed by the 'enter graphics' special function command. This first step is similar to that mentioned previously for cursor positioning. The following statement sets the terminal into graphic mode:

```
PRINT CHR$(27) + "G"
```

By itself, this statement will get the terminal into trouble, as now all characters are graphic. When a text key is hit, the corresponding letter will not be displayed, but instead the CRT will show the graphic character associated with that key. The statement must be made part of a BASIC program by assigning a line number to it. It is advisable to also make it into a subroutine, so that the final program can go into the graphic mode at any point, by simply doing a GOSUB instead of the longer PRINT statement. This subroutine will save memory and make programing easier. Enter this statement:

```
4 ?CHR$(27)+"G";:RETURN
```

The semicolon will prevent an automatic carriage return at the end of the PRINT statement. This is CRITICAL. If a program is printing a sequence of text, and needs a graphic character inserted immediately at the end of the text, then the program can print the text and do a GOSUB4 followed by a print of the graphic character. The graphic character will immediately follow the text. Without that semicolon the graphic character would appear at the first position on the next line.

The next step is to position the cursor at some spot on the CRT. The first part of this position command is the same, no matter where the cursor is placed. It makes sense to also make a subroutine out of this first part, as it can be used whenever cursor positioning is required in the program. This will save considerable memory. Enter the following:

```
2 ?CHR$(27)+"=";:RETURN
```

Again, the semicolon is extremely critical, as without it BASIC will automatically send a carriage return (0D) to the terminal, which will be interpreted as the Y cursor position. The semicolon MUST be there to prevent this. Enter the following line and then do a RUN:

```
1 GOSUB2:?CHR$(37)+CHR$(41)+"A":END
```

The "A" will now be positioned on the 6th line, 10th character. On cursor commands the Y position comes before the X position. This may seem backwards but that's the way it is. To get to the 6th line from the first line add 5 to the home value(32) to get 37. In a similar manner add 9 to 32 to get from the first to the 10th character on the line. The home position is the upper-leftmost character. It has a Y-X position value of 32,32.

Now let's try drawing a horizontal line. The program will use cursor positioning and graphics. When finished it will need to exit the graphic mode so the terminal will once again be conditioned to print text. This exit from the graphic mode is done by calling the following subroutine consisting of a print of an ESC character followed by a lower case "g". Again, the program can call this subroutine from any point, thereby saving memory if several calls are made. The purpose of the semicolon is to allow the program to print some graphic characters, exit the graphic mode (GOSUB6), and then print some text characters all on the same line. Without that semicolon you would get the graphic characters on one line and the text beginning at the left end of the next line.

```
6 ?CHR$(27)+CHR$(103);:RETURN
```

The following FOR...NEXT loop will print the selected graphic character (97 in this case) on line 6, character positions 11 thru 20. Enter and type RUN.

```
1 GO TO 100
100 GOSUB4
120 FOR A = 1 TO 10
130 GOSUB2
140 ?CHR$(37)+CHR$(A+41);
150 ?CHR$(97)
160 NEXT
170 GOSUB6
180 END
```

Wouldn't it be nice if the screen were cleared first? Enter the following two lines. Line 10 will put the cursor to the home position and also clear the entire screen. The empty FOR...NEXT loop will give the CRT time to clear the entire screen and then once again look for more characters from the SYM. Without that empty loop your terminal may miss the next character sent to it as the clear operation does take some time to do.

```
10 ?CHR$(27)+"H"+CHR$(27)+"J":FORA=1TO9:NEXT:RETURN
110 GOSUB10
```

A small change will result in a vertical line instead. Try this change and enter RUN. You should know enough at this point to figure out what I've done.

```
140 ?CHR$(A+37)+CHR$(41);
```

For your homework try to do the following:

1. Pick a different graphic character in line 150
2. Start at some other point on the screen. See what happens when you start too close to the bottom or right side.
3. Draw a diagonal line going from upper left to lower right beginning at a point of your choice. This line will only be 10 characters long.
4. Draw a diagonal line from lower left to upper right.
5. Put a program together to draw a square.

The two areas of memory efficiency and speed are not attempted in this article. First learn BASIC and then the ins-and-outs of your KTM-2. These other two areas will come naturally with experience.

I'd like to pass along a few programing hints to the adventerous KTM-2 owners:

1. If you do not want your screen to scroll, then avoid the rightmost character on the bottom line (Y=23,X=79). Writing in this position will cause your screen to scroll no matter what you do (as far as I know).
2. If your cursor is on the bottom line (Y=23) then any print should terminate with a semicolon or else BASIC will give you a carriage return which will cause your CRT to scroll. I assume you don't want to scroll.
3. BASIC keeps count of the number of characters on a line. If you use too many continuous print statements all ending with a semicolon then this count can reach the maximum. At this point BASIC will automatically insert a carriage return. This can be very irritating. Inserting a zero into location 25 (decimal) will reset this count but this is by no means the only solution to the problem.
4. If, during execution, your program has a 'bug' in the graphic area then your error message will come out in graphic characters. This is really tough to read. Eliminate your GOSUB4 and run your program again. This time you'll get a legible error message.

I hope this article will solve many of the initial problems for you new KTM-2 owners. Good luck.

COMPLEX SOUND GENERATOR CHIP - SN76477N

Jack Gieryic

The addition of sound to my SYM has been an intriguing idea which I have been experimenting with over the past year. One requirement for the sound generation is a minimum of processor dedication. Some of the ideas I have seen in the past make use of the tape cassette interface to generate tones. 'The First Book of KIM' has one such program which I converted over to my SYM. This program has good sound quality but it requires complete processor dedication in order to produce a note. Any process which operates in this mode is unacceptable to me.

An ideal 'sound system' would consist of some hardware which can be programed through a port or memory mapped I/O. I would like to just turn the device on and go on to something else, like updating the CRT or monitoring the keyboard. The less the processor needs to do the better. Such a requirement is met by the Complex Sound Generator chip available at Radio Shack (part number 276-1765) for around $3. The necessary support hardware for generating a particular sound or two may double that cost. A truly great bargain in this era of inflation! The 28-pin chip comes with very thorough documentation and a number of sample circuits for generating some of your more common sounds, such as an excellent phasor gun, a very good prop plane, a race car, a gun shot, an organ, and more.

At first glance the documentation included with the chip was so extensive that I began to wonder just what it was all about...attack, decay, envelope select, VCO, SLF. Fortunately, the documentation also suggested building a 'sound demonstrator box' from a supplied schematic. This turned out to be an excellent idea as it simplifies the task of exercising and learning about the various functions on the chip. I strongly urge you to build such a 'box' as it will give you a very clear picture of the capabilities and limitations of this chip. You will avoid a great deal of frustration and possible disenchantment with a rather good piece of hardware.

For my system I decided to build a heavy duty, stand alone 'box' capable of connecting to any stereo system. The reason being, I wanted

a toy for kids to use without necessarily interfacing with the SYM. Because of this, the cost was a bit high, but this made it rather easy to vary parameters during experimentation. I have seen an inexpensive version using dip switches and miniature variable resistors which required some patience when hunting for new sounds, although it did accomplish its goal. Furthermore, by using a stereo amplifier (only one channel) I was able to obtain great sound reproduction, especially in the low frequency range. This also eliminated the hardware connected to pins 11, 12, and 13 on the demonstrator box schematic. Pin 13 is instead connected to the tape recorder input jack of my stereo. A 9 volt battery is used for power to pin 14 during stand alone operation. Pin 15 then supplies the +5 volts for chip operation. The SYM +5 volts power to pin 15 is used when connected to the SYM.

Essentially the sound generator contains a super low frequency oscillator, a voltage controlled oscillator, and a noise generator. These three sound sources can be mixed in any combination with the addition of attack and decay when using the one-shot feature. A rather limited amount of hardware is required for any particular application. This will enable you to obtain a few sounds by means of the logic control lines to the chip. These logic control lines (pins 1, 9, 22, 25, 26, 27, and 28) are directly controllable thru one of the I/O ports on your SYM. No buffering is necessary.

If your particular application requires several sounds which cannot all be generated by one hardware configuration of a single sound generator chip, then I suggest several chips, each wired for one sound. The SYM then need only control pin 9 of each chip to activate or deactivate the chip. One port will be able to control 8 chips.

I have intentionally refrained from any detailed discussion of the chip's inner workings. The documentation supplied does a very adequate job. Instead I hope to have sparked interest in a piece of hardware which is very compatible with your SYM, can enhance man-machine interaction with the addition of sound, and has a low price tag.

SN76477N COMPLEX SOUND GENERATOR

ENVELOPE SELECT 1	1	28	ENVELOPE SELECT 2
GROUND	2	27	MIXER SELECT C
EXTERNAL NOISE CLOCK	3	26	MIXER SELECT A
NOISE CLOCK RES	4	25	MIXER SELECT B
NOISE FILTER CONTROL RES	5	24	ONE-SHOT CONTROL RES
NOISE FILTER CONTROL CAP	6	23	ONE-SHOT CONTROL CAP
DECAY CONTROL RES	7	22	VCO SELECT
ATTACK/DECAY TIMING CAP	8	21	SLF OSC CONTROL CAP
SYSTEM ENABLE	9	20	SLF OSC CONTROL RES
ATTACK CONTROL RES	10	19	PITCH CONTROL
AMPLITUDE CONTROL RES	11	18	VCO CONTROL RES
FEEDBACK RES	12	17	VCO CONTROL CAP
AUDIO OUTPUT	13	16	EXTERNAL VCO CONTROL
Vcc	14	15	Vreg

WORTH-WHILE READING

COMPUTE., the 6502 magazine recommended in Issue #3, will concentrate on the "packaged" 6502 systems, e. g., PET, Apple, Atari, and the larger OSI systems. A new magazine, compute II., will cover the Single Board Computers based on both the 6502 and the 1802 microprocessors. Incidentally, we have an RCA COSMAC VIP (1802-based) SBC, with tiny BASIC, ASCII keyboard, music system, and other "goodies," and it's loads of fun, especially for its graphics capabilities. But, of course, the SYM-1 is still our own personal favorite SBC.

JACK BUILT PROGRAM - GRAPHICS DEMO PACKAGE NO. 2
---- ----- -------   -------- ---- ------- --- -

Here is the demonstration package Jack mentioned in his letter. If you don't feel quite up to keying it in yourself, you can order it on cassette from the Users' Group, for $6.00 postpaid anywhere.

```
1 E=27:S=124:LIM=2000:TH=32:GOTO100
2 PRINTCHR$(E)+"=";:RETURN
3 PRINTCHR$(E)+"R";:RETURN
4 PRINTCHR$(E)+"G";:RETURN
5 PRINTCHR$(E)+CHR$(114);:RETURN
6 PRINTCHR$(E)+CHR$(103);:RETURN
7 GOSUB2:PRINTCHR$(Y+TH)+CHR$(X+TH)+CHR$(S):RETURN
8 FORY=YSTOYS+YL:GOSUB7:NEXT:RETURN
9 FORX=XSTOXS+XL:GOSUB7:NEXT:RETURN
10 PRINTCHR$(E)+"H"+CHR$(E)+"J":FORA=1TO5:NEXT:RETURN
11 X=INT(77*RND(1)):Y=INT(23*RND(1)):GOSUB7:RETURN
12 GOSUB5:GOSUB6:S=124:RETURN
13 GOSUB10:GOSUB3:GOSUB4:RETURN
14 YL=INT(21*RND(1)):IFYL<3THEN14
15 RETURN
16 GOSUB3:GOSUB4:GOSUB20:GOSUB25:RETURN
17 PRINTCHR$(E)+"H";:FORA=1TO5000:NEXT:RETURN
18 FORA=1TO2000:NEXT:RETURN
19 S=63+INT(64*RND(1)):RETURN
20 XS=INT((79-XL)*RND(1)):YS=INT((21-YL)*RND(1)):RETURN
21 X=XS:GOSUB8:Y=YS:GOSUB9:RETURN
22 X=XS:GOSUB8:Y=YS+YL:GOSUB9:RETURN
23 Y=YS:GOSUB9:X=XS+XL:GOSUB8:RETURN
24 Y=YS+YL:GOSUB9:X=XS+XL:GOSUB8:RETURN
25 GOSUB22:GOSUB23:GOSUB5:GOSUB6:RETURN
26 Y=YS+YL:FORX=XSTOXS+XL:GOSUB7:Y=Y-1:NEXT:RETURN
27 Y=YS:FORX=XSTOXS+XL:GOSUB7:Y=Y+1:NEXT:RETURN
28 PRINTCHR$(Y+TH)+CHR$(X+TH);A:RETURN
30 PRINT"YOU PERVERT":RETURN
32 PRINT"HEY SICKO":RETURN
34 PRINT"YOU SHOULD MAKE AN APPOINTMENT WITH THE VET":RETURN
36 PRINT"ARE YOU FOR REAL?":RETURN
38 PRINT"TRY AGAIN DUMMY":RETURN
40 PRINT"GOOD GUESS":RETURN
42 PRINT"HELLO ";A$;" ARE YOU THERE?":RETURN
44 PRINT"BETTER LUCK NEXT TIME ";A$:RETURN
46 PRINT"ARE YOU NUTS?":RETURN
48 PRINT"THAT'S EXACTLY WHAT I THOUGHT":RETURN
52 GOSUB2:PRINTCHR$(Y+TH)+CHR$(79-X+TH)+CHR$(S):RETURN
54 GOSUB2:PRINTCHR$(23-Y+TH)+CHR$(X+TH)+CHR$(S);:RETURN
56 IFX>0THEN58
57 IFY=0THENRETURN
58 GOSUB2:PRINTCHR$(23-Y+TH)+CHR$(79-X+TH)+CHR$(S);:RETURN
80 GOSUB2:PRINTCHR$(Y+32)+CHR$(X+32);:FORA=1TO4:PRINTCHR$(S);:NEXT:RETU
RN
81 FORA=1TO4:GOSUB2:GOSUB83:NEXT:RETURN
82 FORA=1TO7:GOSUB2:GOSUB83:NEXT:RETURN
83 PRINTCHR$(Y+A+31)+CHR$(X+32)+CHR$(S);:GOSUB84:RETURN
84 PRINTCHR$(E)+"H":RETURN
85 X=XB:Y=YB:RETURN
100 GOSUB10:GOSUB3:GOSUB4:XB=38:YB=8:GOSUB600:GOSUB5:GOSUB6
107 FORA=1TO3:A(A-1)=A-2:B(A-1)=A-2:NEXT
110 GOSUB900:GOSUB2:PRINT"(91  RORSCHACH TEST"
116 GOSUB2:PRINT")92  RORSCHACH TEST - RANDOM GRAPHICS"
117 GOSUB2:PRINT"*93  SYMMETRY"
118 GOSUB2:PRINT"+94  SYMMETRY - RANDOM GRAPHICS"
```

```
120 GOSUB2:PRINT",95  KALEIDOSCOPE"
121 GOSUB2:PRINT"-96  PICK A CARD"
135 GOSUB2:PRINT"0>";:INPUT"YOUR SELECTION IS ";B
137 IFB<1THEN135
139 IFB>6THEN135
140 GOSUB10:ONBGOSUB400,400,200,200,700,800
153 GOSUB10:GOTO110
200 GOSUB3:GOSUB4:GOSUB19:X=20:Y=6
210 FORK=1TO125:IFB=4THENGOSUB19
212 A=INT(3*RND(1)):IFA=3THEN212
214 L=INT(3*RND(1)):IFL=3THEN214
216 IFA(A)<>0THEN240
217 IFB(L)=0THEN212
240 X=X+A(A):IFX<2THENX=37
242 IFX>39THENX=2
244 Y=Y+B(L):IFY<0THENY=11
246 IFY>11THENY=0
248 GOSUB7:GOSUB52:GOSUB54:GOSUB56
250 X=X+A(A):GOSUB7:GOSUB52:GOSUB54:GOSUB56
255 NEXT:GOSUB17:GOSUB5:GOSUB6:RETURN
400 GOSUB3:GOSUB4:GOSUB19:X=20:Y=12
410 FORK=1TO250:IFB=2THENGOSUB19
412 A=INT(3*RND(1)):IFA=3THEN412
414 L=INT(3*RND(1)):IFL=3THEN414
416 IFA(A)<>0THEN440
417 IFB(L)=0THEN412
440 X=X+A(A):IFX<2THENX=37
442 IFX>39THENX=2
444 Y=Y+B(L):IFY=-1THENY=22
446 IFY=23THENY=0
448 GOSUB7:GOSUB2000:X=X+A(A):GOSUB7:GOSUB2000:NEXT:GOSUB5:GOSUB6
449 GOSUB4000:RETURN
450 GOSUB3:GOSUB4:GOSUB19:X=40:Y=12
460 FORK=1TO500:IFB=9THENGOSUB19
462 A=INT(3*RND(1)):IFA=3THEN462
464 L=INT(3*RND(1)):IFL=3THEN464
466 IFA(A)<>0THEN490
467 IFB(L)=0THEN462
490 X=X+A(A):IFX<2THENX=77
492 IFX>79THENX=2
494 Y=Y+B(L):IFY=-1THENY=22
496 IFY=23THENY=0
498 GOSUB7:X=X+A(A):GOSUB7:NEXT:GOSUB5:GOSUB6:RETURN
501 X=XB+2:Y=YB:GOSUB82:RETURN
502 GOSUB85:GOSUB80:X=X+3:GOSUB81:X=X-3:Y=Y+3:GOSUB80:GOSUB81:Y=Y+3
503 GOSUB80:RETURN
504 GOSUB85:GOSUB80:X=X+3:GOSUB82:X=X-3:Y=Y+6:GOSUB80:GOSUB2
505 PRINTCHR$(YB+35)+CHR$(XB+33)+CHR$(S)+CHR$(S):RETURN
506 GOSUB85:GOSUB81:X=X+3:GOSUB82:X=X-3:Y=Y+3:GOSUB80:RETURN
507 GOSUB85:GOSUB80:GOSUB81:Y=Y+3:GOSUB80:X=X+3:GOSUB81:X=X-3
508 Y=Y+3:GOSUB80:RETURN
600 S=124:FORC=1TO5:B=6-C
610 ONBGOSUB501,502,504,506,507
615 PRINTCHR$(E)+"H"
620 FORA=1TO500:NEXTA
630 GOSUB10
640 FORA=1TO500:NEXTA
650 NEXTC:RETURN
700 GOSUB4:FORC=1TO25:GOSUB3:GOSUB19:A=INT(4*RND(1)):IFA=1THENGOSUB5
705 XN=INT(40*RND(1)):IFXN=40THEN705
710 YN=INT(12*RND(1)):IFYN=12THEN710
712 FORB=1TO2:X=XN:Y=YN:GOSUB7:GOSUB52:GOSUB54:GOSUB56
725 A=INT(3*RND(1)):IFA=3THEN725
730 YI=A(A)
```

```
735 A=INT(3*RND(1)):IFA=3THEN735
740 XI=A(A)
741 IFXI<>0THEN745
742 IFYI=0THEN725
745 X=X+XI:Y=Y+YI:IFX>39THEN770
752 IFY>11THEN770
754 IFX<0THEN770
756 IFY<0THEN770
758 GOSUB7:GOSUB52:GOSUB54:GOSUB56:GOTO745
770 NEXTB:NEXTC:GOSUB5:GOSUB6:GOSUB17:RETURN
800 X=INT(63*RND(1)):Y=INT(11*RND(1))
805 C=2+INT(9*RND(1)):IFC=11THEN805
810 A=INT(4*RND(1)):IFA=4THEN810
815 SU=123:IFA=0THENSU=110
816 IFA=1THENSU=122
817 IFA=2THENSU=125
820 GOSUB2:PRINTCHR$(Y+33)+CHR$(X+32);:PRINTC
822 GOSUB2:PRINTCHR$(Y+41)+CHR$(X+42);:PRINTC
824 GOSUB3:GOSUB4:GOSUB2:PRINTCHR$(Y+33)+CHR$(X+35)+CHR$(SU)
826 GOSUB2:PRINTCHR$(Y+41)+CHR$(X+45)+CHR$(SU)
828 GOSUB2:PRINTCHR$(Y+32)+CHR$(X+32)+"A0000000000000B"
830 GOSUB2:PRINTCHR$(Y+42)+CHR$(X+32)+"CPPPPPPPPPPPPPD"
832 FORA=1TO9:GOSUB2:PRINTCHR$(Y+A+32)+CHR$(X+32)+"V":NEXT
834 FORA=1TO9:GOSUB2:PRINTCHR$(Y+A+32)+CHR$(X+46)+"U":NEXT
836 ONCGOSUB852,852,853,854,855,856,857,858,859,860
840 GOSUB5:GOSUB6:PRINTCHR$(E)+"H";:GOSUB18:RETURN
852 GOSUB870:GOSUB872:RETURN
853 GOSUB852:GOSUB871:RETURN
854 GOSUB875:GOSUB877:RETURN
855 GOSUB854:GOSUB871:RETURN
856 GOSUB870:GOSUB875:GOSUB877:GOSUB872:RETURN
857 GOSUB854:GOSUB853:RETURN
858 GOSUB854:GOSUB873:GOSUB879:RETURN
859 GOSUB858:GOSUB870:RETURN
860 GOSUB859:GOSUB872:RETURN
870 GOSUB2:PRINTCHR$(Y+35)+CHR$(X+39)+CHR$(SU):RETURN
871 GOSUB2:PRINTCHR$(Y+37)+CHR$(X+39)+CHR$(SU):RETURN
872 GOSUB2:PRINTCHR$(Y+39)+CHR$(X+39)+CHR$(SU):RETURN
873 GOSUB2:PRINTCHR$(Y+34)+CHR$(X+37)+CHR$(SU)
874 GOSUB2:PRINTCHR$(Y+34)+CHR$(X+41)+CHR$(SU):RETURN
875 GOSUB2:PRINTCHR$(Y+36)+CHR$(X+37)+CHR$(SU)
876 GOSUB2:PRINTCHR$(Y+36)+CHR$(X+41)+CHR$(SU):RETURN
877 GOSUB2:PRINTCHR$(Y+38)+CHR$(X+37)+CHR$(SU)
878 GOSUB2:PRINTCHR$(Y+38)+CHR$(X+41)+CHR$(SU):RETURN
879 GOSUB2:PRINTCHR$(Y+40)+CHR$(X+37)+CHR$(SU)
880 GOSUB2:PRINTCHR$(Y+40)+CHR$(X+41)+CHR$(SU):RETURN
900 GOSUB2:PRINT"#<GRAPHICS DEMO PACKAGE 2"
910 GOSUB3:GOSUB4:GOSUB2:PRINT"!:";:FORA=1TO27:PRINTCHR$(108);:NEXT
920 FORA=1TO4:GOSUB2:PRINTCHR$(A+33)+"9"+CHR$(107):NEXT
930 GOSUB2:PRINT"%:";:FORA=1TO28:PRINTCHR$(106);:NEXT
940 FORA=1TO4:GOSUB2:PRINTCHR$(A+32)+"U"+CHR$(108):NEXT
950 GOSUB2:PRINT" 9";:FORA=1TO29:PRINTCHR$(113);:NEXT
960 FORA=1TO5:GOSUB2:PRINTCHR$(A+32)+"V"+CHR$(97):NEXT
970 GOSUB5:GOSUB2:PRINT"&9";:FORA=1TO29:PRINTCHR$(119);:NEXT
980 FORA=1TO5:GOSUB2:PRINTCHR$(A+32)+"8"+CHR$(103):NEXT:GOSUB6:RETURN
2000 GOSUB2:PRINTCHR$(Y+TH)+CHR$(TH+79-X)+CHR$(S):RETURN
4000 GOSUB2:PRINT"7 WHAT DO YOU THINK THIS IS ";A$;:INPUT"? ";B$
4010 GOSUB10
4015 A=INT(10*RND(1)):IFA=10THEN4015
4020 GOSUB2:PRINT"+>";
4030 ONAGOSUB30,32,34,36,38,40,42,44,46,48
4040 FORA=1TO2000:NEXT:RETURN
OK
```

MORE ON 'TOPS'

In Issue #3, page 25, we mentioned a Tape Operating System for SYM, by Frank Winter, of the University of New South Wales, Australia, and promised more information in this issue. We, both Jack Brown and I, have studied the source code, and are much impressed with its capabilities. TOPS offers every advantage of a Disk System, except the speed, at a much lower cost. The source code is not sufficently commented to permit easy adaptation for other system configurations, and is therefore not quite ready for publication. In fact, when we tried to print his manuscript on our DECwriter, whatever character he was using for TAB printed as a lower case "a".

We did ask Frank to rewrite his handwritten description of TOPS on a RAE cassette; he was kind enough to do so, and we print it here for your information. If you want more details on TOPS we suggest you contact Frank directly (his full address is on page 3-25).

TOPS - a tape operating system for the SYM:

by Frank Winter

Most computing machines may be conceptualized to consist of the following parts:

- A central processor which performs some manipulations on information presented to it
- A device which permits presentation of data to the central processor
- A device which permits reception of the results of the manipulation to the data presenter

An advantage of defining the functional elements of a computer in this way is that the presenter may be a variety of 'devices' including RAM, ROM, tapes, disks or humans. With the exception of ROM, the same 'devices' can also act as receivers. For a more detailed exposition of these concepts, see C.E. Shannon & J. McCarthy (eds.) 'AUTOMATA STUDIES', Princeton Univesrsity Press, Princeton NJ:1956.

An applied implication is that, at least conceptually, all presenters and receivers are interchangeable. We may chose any convenient 'device' such as RAM and substitute it with another, say, tape.

The function of an operating system is to make this exchange process as simple as possible from the operators' points of view, who, for our purposes, are people who desire to achieve a specific objective through the application of a set of algorithms.

TOPS was designed to provide a wide range of commands, usually only found in DISK based operating systems. In its present form TOPS only only works in the BAS-1 environment but its basic design is such that it can be altered to run in any higher language environment, including RAE. Virtually an unlimited number of new commands may be added by users, replacing the BASIC USR function to call routines by name as well as passing any variety of parameters.

Presently, TOPS can use up to 2 tape units. One of these may have editing features such as remote fast-forward, fast rewind, record,and playback which are controlled through the resident 6522 VIA. Tapes may be formatted to permit creation and deletion of files (both program & data) on tape. With formatted tapes and using a transfer rate of 2400 baud, a C60 audio tape can store up to 254K bytes. We presently use 4K storage blocks to achieve this. If 1K storage blocks are used, the upper limit for C60 tape is 200K. We do not use C90 or C120 tapes due to their mechanical weaknesses.In its simplest form, TOPS may be used with one standard tape unit which permits the creation of sequential files such as those used in mailing lists. Updating such files in a minimal SYM system may be accomplished by using 2 standard recorders, one for reading data and one for writing data.

However, to take full advantage of the 'DISK like' features of TOPS, you do need a recorder with remote controls. The recorder must also be able to monitor the tape while it is being transported. TOPS counts the inter-record gaps to position the tape. We have inexpensively modified a portable cassette recorder with cue/review features by adding solenoids to perform the required functions. Total cost, including the recorder and our large 'junk box' was about $65. Hi-Fi decks are now being marketed in Australia with the required features.

TOPS allows you to save all or contigous parts of any BASIC program. The programs may be re-loaded or re-loaded and appended complete with interleaving. This part uses a technique sometimes called 'fast typist mode'. Presently all programs and data are stored as ASCII rather than binary. The appending option is a very powerful feature which permits self modification of programs which can be used to simulate batch processing for really large jobs.

Data files can use one of two buffers, one of two drives, and may be referenced by name. The options are set once for each file, usually at the beginning of a program segment and are then referenced by name only. That is, you 'SET' Charly.dat to the required option and then 'OPEN' and 'CLOSE' this file as Charly.dat' whenever you read or write to this file.

The file manager of TOPS permits any length file names as well as any printable symbol. The only limitation is the storage capacity and your willingness to use long file names. This makes it simple to select a descriptive file name to aid recall later. Files may be deleted to make room for new material, and the tape contents may be displayed any time.

There are a number of 'convenience' features attached to TOPS which include a rather neat line editor which permits easy change of BASIC source code. Any individual symbol in any line may be addressed and changed or deleted. A great time saver after you discover an error in a long line. Also new text may be inserted up to the legal line length. Line length is monitored and the SYM onboard beeper sounds whenever line length is over 60 characters. A null command entered in response to an input statement does not abort the program run. Listing of long programs may be interupted and resumed by pressing one key. The monitor may be accessed and control returned to BASIC, again by pressing one key. Conversion to upper case may be disabled and restored as required.

I would like to conclude with a BASIC source to demonstrate a typical program to create a data file:

```
1REM PROGRAM CHARLY: CREATE CHARLDAT
2PRINT"!SET:W:A:1:'CHARLY.DAT'" :REM write file,using buf A,drive#1
3PRINT"!OPEN'CHARLY.DAT'":REM print all output to tape
4PRINT"Uncle Charly's Data File"
5FOR J=1 TO 100
6PRINT J
7NEXT J
8PRINT"!CLOSE'CHARLY.DAT":REM restore to normal
9PRINT"!END'CHARLY.DAT":REM ensure all goes to tape
10END
```

Now save the program:

```
!PUT'CHARLY.BAS'
```

Now check the library:

```
!CAT
```

And SYM responds:

```
CHARLY.DAT  CHARLY.BAS
```

If you are finished, type

```
!BYE
```

and all additions and deletions to the tape are saved on the tape.

TOPS is not finalized. There are still some minor bugs due to the size of the project (about 4K plus buffers). Also the coding has not been optimized; our first priority was to get it working, and not to conserve memory, nor to display our skill in writing machine language programs. In fact I hope that TOPS will never be finished. It is intended to be a development tool. There is no doubt that TOPS, together with other software now available, puts the SYM head and shoulders above personal computers costing thousands of dollars more.

A VERY INEXPENSIVE EPROM BURNER

Here is a portion of a letter from Joe Hobart, followed by the hardware description and software for a "prommer" which consists essentially of a socket, an edge connector, a board, and some wire! This is one more illustration of how much power is actually built directly into the SYM-1 itself. The design philosophy is elegant, too. Connection to the SYM-1 is only at the AA connector. Just install the board when you are going to use it; remove it otherwise. Then the AA connector is available for all the other things the VIAs can do. My Speak & Spell interface is now on the AA connector, and there are many other controller type devices I want to try out there.

I wonder if it would be feasible to build an "AA Bus" and plug in all sorts of devices simultaneously, switching between them at will? How should one select/deselect the devices? Would switching the +5V do it? Or grounding the +5V input point? The idea of time-sharing a number of VIAs appeals to me much more than adding more VIAs. Anyone out there have the answer?

Building the EPROM programmer on a separate board at its own edge connector leads to the next logical step - building another board just to hold EPROMS and ROMS. This board would just contain a bunch of 24-pin sockets wired in parallel. It could either plug into the Expansion Connector, or into one of the existing ROM sockets, haven't decided which yet. No additional decoding is necessary, since the 2K-block address lines, 90, 98, and F0, are already there, ready to be called into service. This would be a natural extension of John Blalock's two-socket adaptor for RAE-1/2. A further extension would provide for redundant sockets with program controlled switching between them! Switching between POR ROMS would then be easily implemented. Any comments or suggestions along these lines?

3465 North Andes Drive
Flagstaff, AZ 86001
19 April 1980

Dear Lux,

Here is an article on a 2516/2716 EPROM programmer for the SYM-1. The programmer is a joy of simplicity: U28 and U29 provide all signals through the AA connector. The only other connection required is to a regulated 25V supply. I have had excellent results putting Jack Brown's super terminal control patch, trig functions, ultrarenumber, and a tape directory at $F000. I presently have two versions: one for my H-19 and one for the XITEX video terminals at work.

My EPROM programming story is not all roses. I bought some surplus T.I. 2516 EPROMS in Phoenix. These IC's (all manufactured prior to 1979) showed leakage between adjacent memory cells. Apparently, TI did not test for this problem prior to 1979. I have had outstanding results with new TI units. Unfortunately, I cannot afford Intel IC's.

I am willing to program EPROMS for others. For you, there will be no charge; for others, I will charge five dollars for each EPROM programmed. I will need a tape of exactly what is to go on the EPROM. This could be a bit of a problem as it took me three tries to correct all the addresses for my TCP. The problem is that I just do not have any RAM at the F000 address where the EPROM goes. Fortunately, the real attraction of an EPROM is the letter E! The UV eraser sure is handy.

This programmer and associated program could be easily modified for 2532/2732 EPROM's. I do not think that the 32K EPROM's plug directly into the SYM, however, so I have no plans to do this here. I have sent information on this programmer to John Blalock and asked him to look into designing a board or possibly an entire unit for EPROM programming.

Best regards,
Joe Hobart
(602) 779-2110

AN EPROM PROGRAMMER FOR THE SYM-1 - JOE HOBART

The versatility of the SYM I/O ports provides an easy way to commit frequently used software to firmware. All the address and data signals necessary to program the 2516/2716 series of Erasable Programable ROM's are provided by U28 and U29 through the AA connector. The only external requirement is a regulated source of 25 volts at about 30 milliamperes. The only changes to the SYM are to add a third 6522 at U28 and to bypass the buffer transistors Q1 to Q4 by installing a wire from the input point A to the output point B on each buffer. Any other wires to these points should be removed. This modification allows port B of U29 to function as both an input and an output port on all eight lines.

I built the hardware in a few hours using a Radio Shack #276-156 digital breadboard, a 44 pin edge connector with the terminals soldered to the fingers of the breadboard (so the connector will plug into the AA connector on the SYM), an IC socket, two bypass capacitors, and some wire. The IC socket should be wired to the edge connector as follows:

EPROM PIN	AA CONNECTOR PIN	SIGNAL INVOLVED	
1	10	ADDR 7	(2 PA7)
2	M	ADDR 6	(2 PA6)
3	11	ADDR 5	(2 PA5)
4	N	ADDR 4	(2 PA4)
5	12	ADDR 3	(2 PA3)
6	C	ADDR 2	(2 PA2)
7	3	ADDR 1	(2 PA1)
8	D	ADDR 0	(2 PA0)
9	16	DATA 0	(3 PB0)
10	T	DATA 1	(3 PB1)
11	15	DATA 2	(3 PB2)
12	1	GROUND (ALSO FOR 25V SUPPLY)	
13	S	DATA 3	(3 PB3)
14	Y	DATA 4	(3 PB4)
15	21	DATA 5	(3 PB5)
16	Z	DATA 6	(3 PB6)
17	22	DATA 7	(3 PB7)
18	4	(NOT)CE/PGM	(2 CA2)
19	K	ADDR 10	(2 PB2)
20	5	(NOT)OE	(2 CB2)
21	NONE	+25 VOLTS (0.1 CAP TO GROUND)	
22	9	ADDR 9	(2 PB1)
23	L	ADDR 8	(2 PB0)
24	A	+5 VOLTS (0.47 CAP TO GROUND)	

ASSEMBLE LIST

```
              ;ENSURE THAT POWER(25V) IS REMOVED FROM EPROM
              ;PRIOR TO TURNING ON SYM AND LOADING PROGRAMS
                          .BA $1F00
                          .OS
              OUT         .DE $8A47
              OUTBYT      .DE $82FA
              CRLF        .DE $834D
              ACCESS      .DE $8B86
1F00- 20 86 8B            JSR ACCESS
1F03- A9 EC               LDA #$EC
1F05- 8D 0C A8            STA $A80C SETS CA2 LOW & CB2 HIGH
1F08- 00                  BRK
              ;TURN ON EPROM POWER AND ENTER G
1F09- 20 86 8B            JSR ACCESS
1F0C- A9 FF               LDA #$FF CONFIGURE PORTS AS OUTPUTS
1F0E- 8D 03 A8            STA $A803 U28 DDRA
1F11- 8D 02 A8            STA $A802 U28 DDRB
1F14- 8D 02 AC            STA $AC02 U29 DDRB
1F17- A2 00               LDX #$00
1F19- A0 10               LDY #$10 STARTING H,ADDR
1F1B- 8C 00 A8 PAGE       STY $A800 OUTPUT H,ADDR
1F1E- 8C 26 1F            STY HR+2
1F21- 8E 01 A8 BYTE       STX $A801 OUTPUT L,ADDR
1F24- BD 00 10 HR         LDA $1000,X
1F27- C9 FF               CMP #$FF
1F29- F0 17               BEQ SKIP
1F2B- 8D 00 AC            STA $AC00 OUTPUT DATA
1F2E- A9 31               LDA #$31 49 PERIODS
1F30- 8D 1F A4            STA $A41F SET TIMER
1F33- A9 EE               LDA #$EE
1F35- 8D 0C A8            STA $A80C SET CA2 HIGH
1F38- 2C 05 A4 WAIT       BIT $A405 TEST TIMER FLAG
1F3B- 10 FB               BPL WAIT
1F3D- A9 EC               LDA #$EC
1F3F- 8D 0C A8            STA $A80C SET CA2 LOW
```

```
1F42- E8         0350 SKIP       INX
1F43- D0 DC      0360            BNE BYTE
1F45- C8         0370            INY
1F46- C0 18      0380            CPY #$18 LAST PAGE + 1
1F48- D0 D1      0390            BNE PAGE
1F4A- 20 4D 83   0400 VERIFY     JSR CRLF
1F4D- A9 45      0410            LDA #$45 E
1F4F- 20 47 8A   0420            JSR OUT
1F52- A9 52      0430            LDA #$52 R
1F54- 20 47 8A   0440            JSR OUT
1F57- A9 52      0450            LDA #$52 R
1F59- 20 47 8A   0460            JSR OUT
1F5C- A9 4F      0470            LDA #$4F O
1F5E- 20 47 8A   0480            JSR OUT
1F61- A9 52      0490            LDA #$52 R
1F63- 20 47 8A   0500            JSR OUT
1F66- A9 53      0510            LDA #$53 S
1F68- 20 47 8A   0520            JSR OUT
1F6B- A9 3A      0530            LDA #$3A :
1F6D- 85 00      0540            STA *$00 SET FLAG
1F6F- 20 47 8A   0550            JSR OUT
1F72- 20 4D 83   0560            JSR CRLF
1F75- A2 00      0570            LDX #$00 CONFIGURE PORT AS INPUT
1F77- 8E 02 AC   0580            STX $AC02 U29 DDRB
1F7A- A9 CC      0590            LDA #$CC
1F7C- 8D 0C A8   0600            STA $A80C SET CA2 & CB2 LOW
1F7F- A0 10      0610            LDY #$10 STARTING H,ADDR
1F81- 8C 8C 1F   0620 READY      STY HE+2
1F84- 8C 00 A8   0630            STY $A800 OUTPUT H,ADDR
1F87- 8E 01 A8   0640 READX      STX $A801 OUTPUT L,ADDR
1F8A- BD 00 10   0650 HE         LDA $1000,X
1F8D- CD 00 AC   0660            CMP $AC00
1F90- F0 3B      0670            BEQ NEXT
1F92- 48         0680            PHA
1F93- 98         0690            TYA
1F94- 20 FA 82   0700            JSR OUTBYT H,ADDR
1F97- 8A         0710            TXA
1F98- 20 FA 82   0720            JSR OUTBYT L,ADDR
1F9B- A9 20      0730            LDA #$20 SPACE
1F9D- 20 47 8A   0740            JSR OUT
1FA0- AD 00 AC   0750            LDA $AC00
1FA3- 20 FA 82   0760            JSR OUTBYT EPROM DATA
1FA6- A9 20      0770            LDA #$20 SP
1FA8- 20 47 8A   0780            JSR OUT
1FAB- A9 56      0790            LDA #$56 V
1FAD- 20 47 8A   0800            JSR OUT
1FB0- A9 49      0810            LDA #$49 I
1FB2- 20 47 8A   0820            JSR OUT
1FB5- A9 43      0830            LDA #$43 C
1FB7- 20 47 8A   0840            JSR OUT
1FBA- A9 45      0850            LDA #$45 E
1FBC- 20 47 8A   0860            JSR OUT
1FBF- A9 20      0870            LDA #$20 SP
1FC1- 85 00      0880            STA *$00 CLEAR FLAG
1FC3- 20 47 8A   0890            JSR OUT
1FC6- 68         0900            PLA
1FC7- 20 FA 82   0910            JSR OUTBYT DATA
1FCA- 20 4D 83   0920            JSR CRLF
1FCD- E8         0930 NEXT       INX
1FCE- D0 B7      0940            BNE READX
1FD0- C8         0950            INY
1FD1- C0 18      0960            CPY #$18 LAST PAGE + 1
1FD3- D0 AC      0970            BNE READY
1FD5- A5 00      0980            LDA *$00
```

```
1FD7- C9 20      0990            CMP #$20 TEST FLAG
1FD9- F0 07      1000            BEQ DONE
1FDB- A9 4E      1010            LDA #$4E N
1FDD- 20 47 8A   1020            JSR OUT
1FE0- 10 05      1030            BPL ONE ALWAYS
1FE2- A9 44      1040 DONE       LDA #$44 D
1FE4- 20 47 8A   1050            JSR OUT
1FE7- A9 4F      1060 ONE        LDA #$4F O
1FE9- 20 47 8A   1070            JSR OUT
1FEC- A9 4E      1080            LDA #$4E N
1FEE- 20 47 8A   1090            JSR OUT
1FF1- A9 45      1100            LDA #$45 E
1FF3- 20 47 8A   1110            JSR OUT
1FF6- 20 4D 83   1120            JSR CRLF
1FF9- 00         1130            BRK
                 1140            .EN
```

One simple method for providing the +25 volts is three 9 volt batteries in series with silicon diodes (0.6 volt drops each).

```
LABEL FILE:  [ / = EXTERNAL ]


/OUT=8A47               /OUTBYT=82FA            /CRLF=834D
/ACCESS=8B86            PAGE=1F1B               BYTE=1F21
HR=1F24                 WAIT=1F38               SKIP=1F42
VERIFY=1F4A             READY=1F81              READX=1F87
HE=1F8A                 NEXT=1FCD               DONE=1FE2
ONE=1FE7
```

Lines 80-110 of the program set the PGM line (U28 CA2) to protect the EPROM, and then breaks to allow the operator to turn on the +25 volt power supply. When G is entered, lines 130-390 transfer the data stored in memory locations $1000-$17FF to the EPROM. Lines 400-1130 verify the contents of the EPROM against the original program and provide a listing of any errors.

This programmer takes 50 milliseconds per byte or about 102 seconds to completely program a 2K EPROM. A fully erased EPROM has all the bits high so this program skips any location in the data that contains FF. I routinely fill all spaces in my data with FF, so I can later add to the EPROM without having to erase.

I have had excellent results with new T.I. 2516 EPROMS. As far as I know, they are among the least expensive units available. I do strongly recommend that all power be turned off when the IC is either installed in or removed from the programmer socket.

```
.
1F00 20 86 8B A9 EC 8D 0C A8,07
1F08 00 20 86 8B A9 FF 8D 03,70
1F10 A8 8D 02 A8 8D 02 AC A2,2C
1F18 00 A0 10 8C 00 A8 8C 26,C2
1F20 1F 8E 01 A8 BD 00 10 C9,AE
1F28 FF F0 17 8D 00 AC A9 31,C7
1F30 8D 1F A4 A9 EE 8D 0C A8,EF
1F38 2C 05 A4 10 FB A9 EC 8D,F1
1F40 0C A8 E8 D0 DC C8 C0 18,D9
1F48 D0 D1 20 4D 83 A9 45 20,78
1F50 47 8A A9 52 20 47 8A A9,DE
1F58 52 20 47 8A A9 4F 20 47,80
1F60 8A A9 52 20 47 8A A9 53,F2
1F68 20 47 8A A9 3A 85 00 20,6B
1F70 47 8A 20 4D 83 A2 00 8E,5C
1F78 02 AC A9 CC 8D 0C A8 A0,60
1F80 10 8C 8C 1F 8C 00 A8 8E,69
```

```
1F88 01 A8 BD 00 10 CD 00 AC,58
1F90 F0 3B 48 98 20 FA 82 8A,89
1F98 20 FA 82 A9 20 20 47 8A,DF
1FA0 AD 00 AC 20 FA 82 A9 20,9D
1FA8 20 47 8A A9 56 20 47 8A,7E
1FB0 A9 49 20 47 8A A9 43 20,6D
1FB8 47 8A A9 45 20 47 8A A9,C6
1FC0 20 85 00 20 47 8A 68 20,E4
1FC8 FA 82 20 4D 83 E8 D0 B7,BF
1FD0 C8 C0 18 D0 AC A5 00 C9,49
1FD8 20 F0 07 A9 4E 20 47 8A,48
1FE0 10 05 A9 44 20 47 8A A9,E4
1FE8 4F 20 47 8A A9 4E 20 47,82
1FF0 8A A9 45 20 47 8A 20 4D,58
1FF8 83 00,DB
 69DB
```

HANDLING BASIC DATA FILES & MULTIPARAMETER USR FUNCTIONS

The following letter and BASIC program describe a novel method of handling data files in BASIC. You will have to work out your own methods of deletion and updating. This should be not too difficult with a two recorder system. The demonstration program should give you many ideas along these lines.

An added bonus are the excellent examples of how to make good use of the multiparameter passing capabilities built into BAS-1. (I have seen no other Microsoft BASICs with this feature; do you know of any which do?) Also shown is a very ingenious method for ensuring that USR will return the correct value (to make up for a 'bug' in BAS-1).

Here are some parameter values which may help you in your analysis of the program:

DEC		HEX	LABEL
42570	=	$A64A	P3L
-31245	=	$85F3	L21B+4
-32400	=	$8170	ERMSG-1
-11957	=	$D14B	RETURN-1
-29049	=	$8E87	DUMPT

Note that POKE requires positive integers above $8000, while USR requires negative integers.

April 27,1980

Dear Dr. Luxenberg,

Ever since I got BASIC running on my SYM, I wondered how I could best handle Data Files. John Blalock now provides a partial answer by showing how to save and load all data residing in SYM at one time (s. MICRO 23:21). However, what I am still after is a convenient way in which one can manipulate data records under control of BASIC programs.

By now I have tried several approaches and - even though still looking for a better way - find that a few BASIC statements supported only by existing MON and BAS subroutines can provide a halfway decent HS tape handling capability. For production runs you can do wtih only six program lines; the attached listing is expanded by remarks.

The basic approach is this: The top page in RAM serves as I/O Buffer. USR-calls to the Monitor save or load the buffer, while BASIC programs access the area with PEEK or POKE. All this is accomplished by two short subroutines. The Tape Write Subroutine beginning at line 1000 takes the string T$ handed to it by the main program, transfers its contents to the buffer and calls the monitor to write it on tape with an ID equal to TI (determined by the main program). The Tape Read subroutine starting at line 2000 takes the ID (TI), causes the appropriate record to be read by the monitor and hands its contents to the main program as string T$.

The commented version of the program should explain itself, except for the USR call in line 2000: In principle, the call is to location -31245 or hex 85F3 which gets us to the LOAD, HS FMT, 1 PARM entry of MON 1.1. The first parameter of the call (0) is not used. When trying to return, SYM finds the second parameter (-32400) still in the stack, increases it by one and takes that as 'return address'. This gets us to hex 8171, the ERMSG entry of MON 1.1. Now if there has been an error during read, the carry is still set and an appropriate error message will be printed. Upon return the third parameter is found and control goes to hex D14C, a subroutine in BAS which makes the contents of A,Y available to

subsequent BASIC program steps (Note that there is this flaw in BASIC V1.1 which requires you to JMP or JSR to D14C when you want to pass a parameter from a machine language subroutine to a BASIC program). Finally the return to BASIC is made.

The value of USR at that time happens to be positive if the tape-read was successful (CHKH, the HI part of the checksum of a short record is always positive), while practically all error conditions result in a negative value (exception: 2F, the 'last character not '/' error', which I have not seen happen yet and which -should it ever happen - lets you by way of ERMSG at least know that something is wrong). In case of error then, line 2010 is executed fully; normally we proceed to 2020.

I'd hope that all this or at least the trick with the extra parameter -11957 can be of help to someone. It would have saved me a few hours had I known earlier about it.

I'd be interested in hearing from others trying to implement BASIC Data Files.

Sincerely,

Hans W. Gschwind

BASIC DATA FILE HANDLER - HANS W. GSCHWIND
Sindelfinger Weg 30
7250 Leonberg 7
West Germany

MR. GSCHWIND'S PRINTER USED A TWO COLOR RIBBON, SHOWING USER ENTRIES IN RED. WE HAVE USED UNDERLINES TO SHOW USER ENTRIES.

```
.J 0
MEMORY SIZE? 3840
WIDTH?

 3327 BYTES FREE

BASIC V1.1
COPYRIGHT 1978 SYNERTEK SYSTEMS CORP.

OK
LOAD R        THESE LINES ARE THE 'KEY' TO THE PROGRAM:
LOADED        UNDERLINES INDICATE PARAMETERS TO BE
OK            CHANGED IF MORE MEMORY IS AVAILABLE
LIST

 1000 FORI=1TOLEN(T$):POKE3839+I,ASC(MID$(T$,I,1)):NEXT
 1010 POKE42574,TI:POKE42573,15:POKE42572,0:POKE42571,15
 1020 POKE42570,LEN(T$):Q=USR(-29049,128):RETURN
 2000 T$="":POKE42570,TI:Q=USR(-31245,-11957,-32400,0)
 2010 IFQ<0THENPRINT". BACK TAPE AND CONT ":STOP:GOTO2000
 2020 FORI=3840TO3839+PEEK(254):T$=T$+CHR$(PEEK(I)):NEXT:RETURN


OK
LOAD S
LOADED
OK
LIST   REMARKS HAVE BEEN ADDED TO THE 'KEY' BY WAY OF EXPLANATION:

 991 REM
 992 REM ***** TAPE HANDLING SUBROUTINES FOLLOW *****
 993 REM LAST PAGE OF 4K SYM SERVES AS I/O BUFFER
 994 REM LIMIT MEMORY SIZE TO 3840 AT START UP
 995 REM
```

```
996 REM ******** TAPE WRITE SUBROUTINE ********
997 REM ENTER WITH TI = RECORD ID AND
998 REM T$ = CONTENTS (UP TO 255 CHARACTERS)
999 REM FIRST DUPLICATE T$ IN I/O BUFFER:
1000 FORI=1TOLEN(T$):POKE3839+I,ASC(MID$(T$,I,1)):NEXT
1008 REM NOW POKE RECORD ID, START- AND END ADDRESS
1009 REM THEN CALL MONITOR TO SAVE BUFFER. FINALLY RETURN:
1010 POKE42574,TI:POKE42573,15:POKE42572,0:POKE42571,15
1020 POKE42570,LEN(T$):Q=USR(-29049,128):RETURN
1995 REM
1996 REM ******** TAPE READ SUBROUTINE ********
1997 REM ENTER WITH TI=RECORD ID TO BE READ
1998 REM SUBROUTINE RETURNS T$ TO MAIN PROGRAM
1999 REM CLEAR STRING T$, POKE RECORD ID, CALL MON TO LOAD:
2000 T$="":POKE42570,TI:Q=USR(-31245,-11957,-32400,0)
2009 REM NOW PROVIDE FOR RECOVERY FROM READ ERROR:
2010 IFQ<OTHENPRINT". BACK TAPE AND CONT ":STOP:GOTO2000
2019 REM FINALLY RECONSRUCT STRING T$ AND RETURN:
2020 FORI=3840TO3839+PEEK(254):T$=T$+CHR$(PEEK(I)):NEXT:RETURN
OK
LOAD T
LOADED
OK        HERE IS A DEMONSTRATION PROGRAM ILLUSTRATING HOW THE
LIST      "KEY" MAY BE USED:
```

```
10 PRINT:PRINT"ADDRESS FILE":PRINT
15 REM INTENDED ONLY AS EXERCISE IN
20 REM HANDLING OF MAG TAPE IN SYM BASIC
24 REM
25 REM ******** GENERATE FILE ********
30 PRINT"SET UP TAPE FOR RECORDING":PRINT:PRINT
40 A$=CHR$(013)+CHR$(010):REM CRLF (USED REPEATEDLY)
50 INPUT"ACCOUNT? (* TERMINATES FILE) ";A1$
60 IFA1$="*"GOTO120:REM GO TERMINATE FILE
70 INPUT"NAME? ";A2$
80 INPUT"STREET? ";A3$
90 INPUT"TOWN, STATE? ";A4$,A5$
100 PRINT:T$=A1$+A$+A2$+A$+A3$+A$+A4$+", "+A5$:REM COMPOSE T$
110 TI=2:GOSUB1000:GOTO50:REM WRITE T$ ONTO TAPE AND GO BACK
120 T$="*":GOSUB1000:REM WRITE EOF
130 PRINT:PRINT"FILE TERMINATED"
134 REM
135 REM ******** PRINT FILE ********
140 INPUT"SET UP TAPE FOR PLAYBACK. THEN TYPE GO ";A1$
150 TI=0:GOSUB2000:REM READ NEXT RECORD (I.E.T$) FROM TAPE
160 IFT$="*"THENPRINT:PRINT"END OF FILE":END
170 PRINT:PRINTT$:PRINT:GOTO150
991 REM
992 REM ***** TAPE HANDLING SUBROUTINES FOLLOW *****
993 REM LAST PAGE OF 4K SYM SERVES AS I/O BUFFER
994 REM LIMIT MEMORY SIZE TO 3840 AT START UP
995 REM
996 REM ******** TAPE WRITE SUBROUTINE ********
997 REM ENTER WITH TI = RECORD ID AND
998 REM T$ = CONTENTS (UP TO 255 CHARACTERS)
999 REM FIRST DUPLICATE T$ IN I/O BUFFER:
1000 FORI=1TOLEN(T$):POKE3839+I,ASC(MID$(T$,I,1)):NEXT
1008 REM NOW POKE RECORD ID, START- AND END ADDRESS
1009 REM THEN CALL MONITOR TO SAVE BUFFER. FINALLY RETURN:
1010 POKE42574,TI:POKE42573,15:POKE42572,0:POKE42571,15
1020 POKE42570,LEN(T$):Q=USR(-29049,128):RETURN
```

```
1995 REM
1996 REM ******** TAPE READ SUBROUTINE ********
1997 REM ENTER WITH TI=RECORD ID TO BE READ
1998 REM SUBROUTINE RETURNS T$ TO MAIN PROGRAM
1999 REM CLEAR STRING T$, POKE RECORD ID, CALL MON TO LOAD:
2000 T$="":POKE42570,TI:Q=USR(-31245,-11957,-32400,0)
2009 REM NOW PROVIDE FOR RECOVERY FROM READ ERROR:
2010 IFQ<OTHENPRINT". BACK TAPE AND CONT ":STOP:GOTO2000
2019 REM FINALLY RECONSRUCT STRING T$ AND RETURN:
2020 FORI=3840TO3839+PEEK(254):T$=T$+CHR$(PEEK(I)):NEXT:RETURN
OK
RUN

ADDRESS FILE

SET UP TAPE FOR RECORDING


ACCOUNT? (* TERMINATES FILE) 001
NAME? SYM-1 USERS' GROUP
STREET? P. O. BOX 315
TOWN, STATE? CHICO,CA 95927

ACCOUNT? (* TERMINATES FILE) 002
NAME? SYNERTEK SYSTEMS CORPORATION
STREET? 150 SOUTH WOLFE ROAD
TOWN, STATE? SUNNYVALE,CA 94086

ACCOUNT? (* TERMINATES FILE) 003
NAME? MICRO TECHNOLOGY UNLIMITED
STREET? P. O. BOX 12106
TOWN, STATE? RALEIGH,NC 27605

ACCOUNT? (* TERMINATES FILE) *

FILE TERMINATED
SET UP TAPE FOR PLAYBACK. THEN TYPE GO GO

001
SYM-1 USERS' GROUP
P. O. BOX 315
CHICO, CA 95927
                                             A TAPE READ ERROR WAS
ER FF. BACK TAPE AND CONT                 ←  DELIBERATELY FORCED AT
                                             THIS POINT, BY STOPPING
BREAK IN  2010                               THE CASSETTE READER, TO
OK                                           ILLUSTRATE HOW TAPE
CONT                                         READ ERRORS ARE HANDLED


002
SYNERTEK SYSTEMS CORPORATION
150 SOUTH WOLFE ROAD
SUNNYVALE, CA 94086


003
MICRO TECHNOLOGY UNLIMITED
P. O. BOX 12106
RALEIGH, NC 27605


END OF FILE

OK
```

UPGRADING OF KTM-2 TO KTM-2/80

KTM-2's after Serial No. 0733 can be upgraded to 80-column capabilities. The process involves clearing out plugged solder holes, cutting one trace, wiring one jumper, soldering in five dip sockets, and installing one ROM chip, two RAM chips, and two TTL chips. If you want to make the modification, contact Bob Myers (address below) for additional information and prices on the upgrade kits. Incidentally, the -2/80 can be "switched" to a 40 column display whenever you need, or wish, to use a standard TV receiver through an RF modulator.

We have been in telephone touch with Bob about this and other matters, these past many months, and he has sent us a number of BASIC subroutines which would be useful in a "bookkeeping" package. Instead of just giving his address, we're publishing his last letter to us, which contains an interesting free offer:

Bob Myers
109 Fire Lane
North Cape May, NJ 08204
USA
Home 609 884-0422
Work 609 522-7781 Ext. 250

15 May 1980

Dear Lux,

A freebie for the SUG. Because I learned to count in Octal way back when, I still think "HEX" is a curse. To that end I have done something useful with my SYM-1. I took an old number base conversion program and added some parts to make it print up a list of the decimal and hex boundaries for each 1-K byte block of memory.

The size has been reduced so that it will fit inside the front cover of the SYM-1 Reference Manual (or anywhere else the same size).

Any Octalmaticians (we that can count without using our thumbs) or other SUG members who would like a copy or two can have them by sending me a self-addressed and stamped envelope. If they want a copy of the number base conversion program, make it a legal size envelope and two fifteen cents stamps (or the equivalent in international postage).

The whole works is on four sheets of 8 1/2 by 11 paper for one chart and the program.

Sincerely,

Bob Myers

P.S. A couple of post scripts:
1. If anyone wants an example of how not to program, they get that free with the program. It was a kluge job to begin with. When I got done adding all the neat touches it brought new meaning to the Basic "wend while" command. (Or perhaps "wend all the while").
2. Ain't Carl Moser's SWP-1 GREAT!

RAM

(Editor's note: That "RAM" at the end of the letter is Bob's initials!)

MORE ON SWP-1

Some of our contributors are beginning to submit their articles on cassettes, in RAE-1 format. This makes the task of editing for publication so very much simpler. We transfer from cassettes to diskettes for greater convenience, and then edit their "manuscripts" for SWP processing.

So that you can see how this is done, we print below, in "rough draft" form, a portion of the introductory paragraphs to Mr. Gschwind's article (which appears on page 4-19). We have thrown in some comments (following the macros ".;") explaining how SWP-1 may be used more effectively. Now, when our contributors SWP their manuscripts themselves, editing will become a very minor task!

```
.M 0 73 65 1
.NOFILL
HANDLING BASIC DATA FILES  &  MULTIPARAMETER USR FUNCTIONS
-------- ----- ---- -----  -  -------------- --- ---------
.JUST
The following letter and BASIC program describe a novel method of
handling data files in BASIC.
You will have to work out your own methods of deletion and updating.
This should not be too difficult with a two recorder system.
The demonstration program should give you many ideas along these lines.
.L2
Here are some parameter values which may help you in your analysis of
the program:
.;Note how the "tabulating" mode is entered and exited here.
.;Ordinary text is entered as you would ordinarily enter it, and the
.;editing "macros" are entered later.
.;In data which is to be tabulated use "^" instead of space.
.;Note that there is no need to try to make line lengths balance, and it
.;is a good idea to begin each sentence on a new line.
.;This makes for easier editing, and permits you to use the RAE >MOve
.;command to rearrange sentence order if desired.
.;One of the enhancements most needed in SWP-1 (SWP-2?) is the addition
.;of a TAB macro of the form .TAB N1 N2 N3 N4, responding to the TAB
.;key (CONTROL I).
.M 15
.L2
.NOFILL
^^DEC^^^^^HEX^^^^^^LABEL
^^---^^^^^---^^^^^^-----
^42570 = $A64A^^^^^P3L
-29049 = $8E87^^^^^^DUMPT
.M 0
.L2
.JUST
Note that POKE requires positive integers above $8000, while USR
requires negative integers.
```

PRINTER INTERFACING

The three most common types of printer interfaces are through the RS-232 port, the 20 mA loop, and the Centronics Parallel Interface. The SYM-1 implements the first two interfaces, providing both the necessary hardware and software, at least for that class of printers which print each character as it is received, and the SYM allows for providing "pad" bits with each carriage return transmitted (the default value for PADBIT in $A650 is $01) to allow for the carriage return time.

The DECwriter II, which we are using, handles the carriage return delay by storing characters in a buffer, then prints them at a higher rate, in a catch-up mode, returning to the lower rate when it has caught up. The PADBIT value of $01 is large enough so that the DECwriter never uses the catch-up mode. By a simple modification, costing around $3.00 for parts, the DECwriter can be persuaded to operate always at the catch-up rate, with no damage, since it was actually designed to handle the higher rate. Thus the new "normal" rate for the DECwriter is now 600 baud, not 300. We gave up the 150 rate, and still have the 110 and 300 rates for use with our timeshare modem.

The SYM-1 can drive the KTM-2 (or any other terminal) at 4800 baud. If your printer has a lower maximum baud rate, it can not be driven from the printer auxilliary port on the KTM-2, unless you lower the rate on the KTM-2; this is not too convenient to do. Our choice was to drive the printer through the 20 mA loop, which can be set to its own baud rate independently. We have written a simple patch which permits driving either the CRT alone or both printer and CRT.

Another class of printer can be driven at any baud rate up to 9600 baud, but all characters are buffered, and not printed until a carriage return ($0D) is received. An additional line between SYM and the printer is required to monitor the "printer ready" status, and handshaking software is required to delay further data transmission until the printer signals that it has completed its carriage return and is ready for more data. We have software for this type of printer too.

The Centronics interface is easily inplemented through one of the "spare" 6522 VIAs, and, yes, we also have the software driver for this. We are not making any "brand name" recommendations, nor do we prefer any one type of interface to another. Many of the printers on the market may provide two, or all three interface options, and you can select the most convenient option. Whichever printer or interface option you select, if you have any software questions, phone or write (please include stamps), and we will be happy to provide you with the necessary data and/or software at no cost.

APPLES VERSUS ORANGES (continued from page 2)

I am sure many of us, during periods when SYM has frustrated us, wish we had gotten the Apple instead. But, when my SYM is doing its thing, and doing it well (which is better then 99.9% of the time, once the early cassette problems were solved), I am pleased with my choice, and the added pride of having raised my SYM to be what I wanted it to be.

Because friends occasionally do ask us for a comparision, or a recommendation, we did compare the two systems for cost and effectiveness. If there is room left in this issue (highly unlikely) we will publish the study results. In any event, the final conclusion was that, for the same amount of money output, the SYM-1 can lead to a better "customized" system, and, if you credit the value of the hours you have spent to the cost of education, you may be better off with the SYM, from the hardware cost standpoint. We'll discuss the software situation in the next issue.

RANDOM FILES

WE WILL be teaching a weekend course on Microprocessor Systems for the University of California at Davis (just west of Sacramento), on the weekend of December 5-7, 1980. All participants will receive a SYM-1, plus some useful software on cassette, with source listings. The course fee will be around $475; for further information, contact Garrett Jones, UC Davis, Davis, CA 95616, (916) 752-2177.

WE HEAR that a really nice enclosure for the KTM-2 (and /80) will very soon be available from an independent source. We'll publish price and delivery information in Issue #5.

WE HAVE a very heavy teaching, traveling, research, and lecture schedule planned for this fall; also we are committed to delivering the manuscript for the second edition of our McGraw-Hill (1968) book (with R. L. Kuehn), "Display Systems Engineering," to our new publisher by year's end. This may delay Issue #5 (September/October) for awhile. If this should happen, count on a combined double-sized Issue #5/6 (September/October/November/December) somewhere in the middle of that four month period!

ONCE THIS issue goes to press (and then to the post office) we will have enough time (?) to install and evaluate FORTH and tiny c (a subset of C, a Pascal-like language). We have studied the manuals and are anxious to implement and use both languages; each for its own unique advantages. Jack Brown has nearly completed his own enhanced version of SYM FORTH. Both languages have very effective Users' Groups for program exchanges.

THERE ARE at least a few score SYM users in Australia and New Zealand. John F. Newman, 1/14 Marine Pde, St. Kilda, Vic, 3182, Australia, has expressed an interest in forming a "local" SYM Users' Group.

OUR COMPUTER room wall has a gallery of photos sent in by readers, showing how they have packaged their SYM systems. What diversity, what imagination, what ingenuity! And some of the SYM applications described to us are almost unbelievable.

WE PLAN to get out RAE NOTES No. 3 by the end of July. These notes will provide the detailed pages zero and one memory maps, as promised. We hope to include information on how .CT (continue on tape), and the Relocating Loader can be modified for Disk Operating Systems.

WE ARE very much interested, and so would our readers be, in hearing more from you about interfacing SYM to Disk Systems.

HARDWARE RECOMMENDATIONS

We are using the "First Mate" prototyping system, designed by Richard Turpin, and marketed through MicroMate, P. O. Box 50111, Indianapolis, IN 46256, for all of our experimental interfaces to SYM. The board mounts rigidly to SYM on nylon standoffs fitting into existing holes on the SYM board, without blocking the keypad and display. We do like it!

Dick's newest product, which we have not seen yet, is the ColorMate PC Board, a color video board for the SYM.The ColorMate Board, with full documentation, will sell for $50.00. The ColorMate is designed around the Motorola MC6847 video display generator. It is built on a 4 1/2 by 6 1/2 inch circuit board, requires a single 5 volt supply and interfaces directly to the Expansion connector. It provides nine modes of operation ranging from 192x128 full graphics to 16 row by 32 column alphanumerics, and up to nine colors plus reverse video. Write MicroMate for more details. Shades of the Apple, full color from SYM!

SOFTWARE RECOMMENDATIONS

We have been reviewing a number of programs distributed by THE 6502 PROGRAM EXCHANGE, 2920 Moana, Reno, NV 89509, and suggest that you write for their newest catalog. Please send $1.00, US/Canada, $2.00 overseas. Two programs, in particular should interest many of you. These are a SYM version of Peter Jennings' Microchess for KIM, upgraded to use a terminal display (much nicer then the keypad), and HUEY. The latter is a revised version of a program first published in KILOBAUD, December 1977. With HUEY, your SYM/Terminal combination can be made to emulate a programmable pocket calculator. That is, of course, quite a step down and backwards from BAS-1, but HUEY is worth having for one important reason. Haven't you ever wondered how those pocket calculators are programmed, and what kinds of algorithms they use internally for arctan, log, x to the y power, etc.? HUEY will fully educate you along those lines. Well worth the cost.

The EXCHANGE also offers a number of higher level languages for SYM. These include FOCAL, XPLO, and TEC (a Text Editor). We plan to review these in the next issue; our preliminary tests showed them to have some interesting features and applications.

SHOPPING LIST ADDENDUM

All prices given below are now obsolete. Please use prices on the most recent issued 'Shopping List'.

WE HAVE A LARGE NUMBER OF KTM-2/80S ON HAND FOR IMMEDIATE DELIVERY CONTACT US FOR SPECIAL SUMMER SALE PRICES.

WE HAVE MADE A SPECIAL BUY ON 2114 (450 NS) MEMORY CHIPS. SALE PRICE $5.00 EACH POSTPAID US/CANADA. OVERSEAS PLEASE ADD $1.00 TO EACH TOTAL ORDER FOR AIRMAIL.

WE MADE A LARGE PURCHASE OF MTU DAC MUSIC BOARDS (INCLUDING HARDWARE MANUAL, SOFTWARE, AND CASSETTE) AND ARE OFFERING THE FOLLOWING REDUCED PRICES, FIRST CLASS/AIRMAIL:
$46.00 US/CANADA, $47.00 EUROPE, $48.00 ASIA/PACIFIC.

NEW ITEMS (OLD ITEMS ARE ON THE BACK PAGES OF ISSUES 1, 2, AND 3)

BOB PECK, AUTHOR OF THE 'SYM/KIM APPENDIX TO THE FIRST BOOK OF KIM,' AND THE 'SYM-1 HARDWARE THEORY OF OPERATIONS MANUAL,' REVIEWED IN EARLIER ISSUES, HAS A NEW AND EVEN MORE USEFUL BOOK, THE 'SYM-1 MONITOR THEORY OF OPERATIONS MANUAL,' WHICH DESCRIBES THE FEATURES OF MON 1.1, AND THE USE OF ITS SUBROUTINES. THE 'MONITOR MANUAL' IS AVAILABLE THROUGH THE USERS' GROUP FOR $8.00 POSTPAID US/CANADA, $9.50 AIRMAIL OVERSEAS.

COPYRIGHTS AND COPYING 'RIGHTS'

As noted in the masthead, clubs, educational institutions, and other nonprofit organizations may freely reproduce and distribute any portions of SYM-PHYSIS.

It is probably safe to assume that this is being done, not only with SYM-PHYSIS, but with the software offered by the SYM-1 Users' Group. Now, two or three or four copies seem 'fair' under the copyright laws. We ourselves share purchased software occasionally with at most one or two very close colleagues, but such exchanges are not planned at time-of-purchase. They occur spontaneously, during technical discussions, when we are actually working together with the software, and problems arise on its use. On the other hand, purchasing one copy, with the intent of making ten or twenty or fifty duplicates for distribution does make it seem like the organization or group is in the republishing business on the side.

May we suggest that if you plan to make five or more copies of software cassettes and listings, and are a club devoted to the furtherance of SYM-1 Users, why not ask for a group rate, as you would do for hardware, or buy at least one copy for each five members? That way you get an 80% discount, and still can feel that you are doing your bit to encourage the production of high-quality, low-cost software for the SYM-1.

BLALOCK PRICE INCREASE

The prices for the 4K Memory Board and the Double ROM Chip Holder for RAE-1/2 have been increased to $8 and $16 respectively. Blalock's correct address is on page 3-27.

ERROR CORRECTION

JOHN VALENTE of Marlboro, VT, was the first to point out that we had mislabeled the pins of the 4050 chip on page 3-24. To correct the drawing, interchange the pin number pairs as follows: 7 and 6, 5 and 4, 3 and 2. Sorry about that!

BULK RATE
U.S. Postage
PAID
Chico, CA 95927
Permit # 430

TIME VALUE PRINTED MATTER

SYM-PHYSIS
SYM-1 Users' Group
P.O. Box 315
Chico, CA 95927

Address Correction Requested