SYM-PHYSIS

THE SYM-1 USERS' GROUP NEWSLETTER

VOLUME III, NUMBER 2 (ISSUE NO. 12) - SUMMER 1982 (APR/MAY/JUN)

SYM-PHYSIS is a quarterly publication of the SYM-1 Users' Group, P. O. Box 319, Chico, CA 95927. SYM-PHYSIS and the SYM-1 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.

Editor/Publisher: *H. R. "Lux" Luxenberg*
Business/Circulation: *Jean Luxenberg*
Office Staff: *Joyce Arnovick*
Associate Editors: *Dennis Hall, Jack Brown*
Tom Gettys, Jack Gieryic

SUBSCRIPTION RATES: (Volume III, 1982, Issues 11 - 14)

USA/Canada - $10.50 for a volume of four issues. Elsewhere - $14.00. Make checks payable in US dollars to "SYM-1 Users' Group", P. O. Box 319, Chico, CA 95927, Telephone (916) 895-8751.

BACK ISSUES ARE STILL AVAILABLE AS FOLLOWS:

Issue 0, the Introductory Issue (1979), and Issues 1 through 6 (Volume I, 1980), are available, as a package, for $12.00, US/Canada, and $16.00, First Class/Airmail, elsewhere.

Issues 7 through 10 (Volume II, 1981), are available for $10.50, US/Canada, and $14.00, First Class/Airmail, elsewhere.

RAM-BLINGS

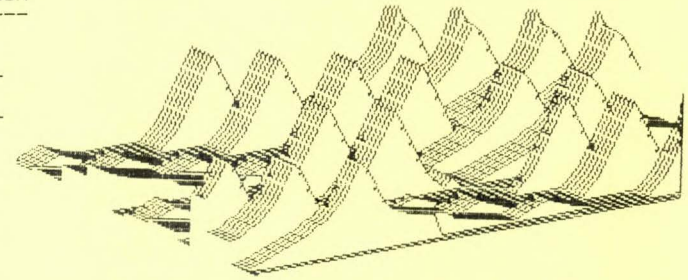
The past quarter has been an unusually busy one for us. First, there was the four week trip to Australia and New Zealand, then a four week effort to complete the documentation for the FDC-1, then several weeks in "reorganizing" our laboratory and production facilities and our ever growing paperwork storage system, this latter to increase the liklihood of finding needed information within a short enough time for it to be of use, both to ourselves and to those who call in or write for help.

We had SYMmers from Australia, France, Switzerland, and Oregon visit with us, for from one to four days. In addition, we received many excellent programs for both publication and distribution, all of which required the usual amount of editing and testing, several new hardware items which required installation and checkout, and several excellent books for review.

We taught a weekend microprocessor course at University of California, Davis during mid-June and are preparing for a one week course on display systems engineering at University of California, Los Angeles during mid-August. Our nine month sabbatical is nearly over, and we are preparing some new course material for the fall semester which begins at the end of August. Our writing speed and ability to read what we have written on a CRT have, unfortunately, both been diminished by cataracts developing in both eyes (one eye will be worked on at the end of August, the other in December). Thus, this issue is, as usual, later than it should be, and our correspondence and unfinished project files are as backlogged as ever!

GRAPHICS ON THE EPSON

An example of the output of "RADAR", a 3-D plotting program incorporating a hidden line algorithm, by Ian Dilworth, University of Essex. More on "RADAR" below, but first, an adaptation of Ian's Graphics Printer Driver for a seven bit interface.



See the "unhidden" lines on page 12-30!

While the program and examples presented below are specifically for the Epson MX-80 with Graftrax, and the MTU Visible Memory, the program is easily modifiable for use with any printer with point graphics capabilities, and for any visible display unit (VDU) in which each pixel is individually addressable. In fact, a VDU unit is not even required, although the absence of this capability will slow down the procedure, and waste lots of time and paper.

We received an Epson Printer/Visible Memory graphics printing routine from Ian Dillworth while we were still less than half-finished with our own version. His routine gave "strange" results because of a different method of interfacing, and we had to modify it to work with our system. Because we liked his approach, we borrowed heavily from it and give him full credit below. His page zero assignments conflicted with RAE-1, and even modified some of its parameters, causing interesting results on return to RAE after a plot!

To avoid this, and to make the routine universally callable from RAE, BAS, FORTH, PASCAL, tiny-C, etc., we added several useful features which you may wish to incorporate in those of your own programs which require extensive use of page zero and/or (temporary) modification of system vectors. These are the subroutines used on both entry into and exit from the main program to save and restore all page zero locations and vectors used by the main program. We also included a JSR INSTAT to permit aborting the printing with the BREAK key on the terminal, and the printer patch itself, for the sake of completeness.

Note that the printer patch is based on using only seven data lines to the printer, and an eighth line for the busy signal from the printer; thus only one port is needed. If your interface supports the eighth data line the necessary mods to the program should be obvious. The use of the eighth line will speed up the printing time, but at the expense of tying up a second port for the one busy signal bit.

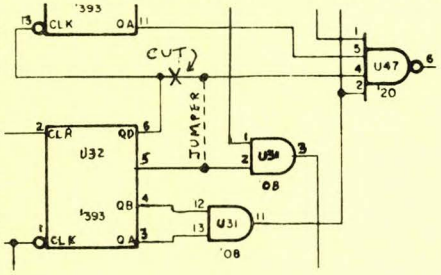
NOTE TO VISIBLE MEMORY USERS: A minor problem with the "7 bit" Epson interface (as compared to the more conventional 8 bit interface) is that three additional lines will be printed at the bottom of the picture, since 7 x 29 = 203, while 8 x 25 = 200. The obvious way around this is to fill the extra 120 bytes with either $00 or $FF to provide either a black or a white lower edge to the print.

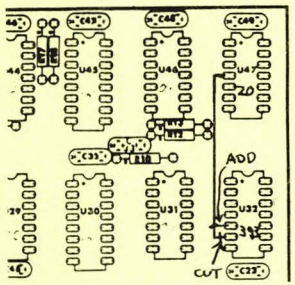
There is, however, a way to get an additional four lines on the screen by cutting one trace and adding one jumper, as illustrated in the two figures below. Now, instead of your Visible Memory running, say, from $2000-$3F3F, it will cover $2000-$3FDF, and, instead of 320 H x 200 V, you will now have 320 H x 204 V pixels. This provides significantly higher resolution than the Apple's 280 H x192 V black and white graphics mode.

Recall that only 8000 of the 8192 bytes are normally displayed, leaving 192 bytes as "ordinary" RAM. By making the mod you will have 8160 displayed bytes and still have a reserve of 32 "invisible" bytes to be used for such utilitarian functions as page zero swap locations, cursor storage, etc. The program given above will, of course, print three of these extra four lines. Previously written programs for the Visible Memory need not be modified, except for blanking out the lines, if necessary, prior to use.

The information on the Visible Memory modification came to us through Walter Glab from Dave Kemp, who alluded to it in his June 1980 MICRO article, "Slide Show for the SYM".

```
                0010 ;SCREEN DUMP OF VISIBLE MEMORY TO MX-80
                0020 ;BY IAN J. DILWORTH
                0030 ;DEPT E.E.S, UNIV. OF ESSEX, COLCHESTER, ESSEX. U.K.
                0040
                0050 ;MODIFIED BY LUX FOR 7 BIT INTERFACE,
                0060 ;PAGE ZERO RELOCATIONS, VECTOR SWAPS, ETC.
                0070
                0080 VM.START   .DE $2000
                0090 VM.END     .DE $3F3F     ;OR $3FDF, WITH MOD SHOWN BELOW
                0100
                0110 BEEP       .DE $8972
                0120 OUTCHR     .DE $8A47
                0130 INSTAT     .DE $8386
                0140 ACCESS     .DE $8B86
                0150
                0160 OUTVEC     .DE $A664
                0170
                0180 PAD        .DE $A801
                0190 PADHI      .DE $A80F
                0200 PADD       .DE $A803
                0210 PCR        .DE $A80C
                0220
                0230 ;PAGE ZERO
                0240
                0250 BOTTOM     .DE 0         ;OR ANYWHERE ON PAGE
                0260
                0270            .BA BOTTOM
                0280
0000-           0290 VISMEM     .DS 2
0002-           0300 VISORG     .DS 2
0004-           0310 BLKPNT     .DS 2
0006-           0320 CARRYSUM   .DS 1
0007-           0330 LINECOUNT  .DS 1
0008-           0340 COUNT1     .DS 1
0009-           0350 COUNT2     .DS 1
                0360 TOP
                0370
                0380 ;SCRATCH PAD MEMORY LOCATIONS
                0390 ;HIDE IN "INVISIBLE" MEMORY
                0400
                0410            .BA VM.END+1+160
                0420
3FE0-           0430 ZEROSAVE   .DS TOP-BOTTOM
3FEA-           0440 VECTORSAVE .DS 2
3FEC-           0450 BYTES      .DS 8
                0460
                0470 ;>>>>>>> MAINLINE STARTS HERE <<<<<<<
                0480
                0490            .BA $4000
                0500            .OS
                0510
                0520 ;INITIALIZE ROUTINES
                0530
```






```
4000- 20 D8 40  0540            JSR VECSWAP
4003- 20 C8 40  0550            JSR ZEROSWAP
4006- 20 0D 41  0560            JSR TURNON
4009- A2 07     0570            LDX #7
400B- 20 F4 40  0580            JSR SETSPC
                0590
                0600 ;INITIALIZE POINTERS
                0610
400E- A9 00     0620            LDA #0
4010- 85 06     0630            STA *CARRYSUM
4012- 85 00     0640            STA *VISMEM
4014- 85 02     0650            STA *VISORG
4016- A9 EC     0660            LDA #L,BYTES
4018- 85 04     0670            STA *BLKPNT
401A- A9 3F     0680            LDA #H,BYTES
401C- 85 05     0690            STA *BLKPNT+1
401E- A9 20     0700            LDA #H,VM.START
4020- 85 01     0710            STA *VISMEM+1
4022- 85 03     0720            STA *VISORG+1
                0730
                0740 ;GRAPHICS DUMPED 7 ROWS AT A TIME IN 8 BYTE BLOCKS
                0750
4024- A9 28     0760            LDA #40       ;SET LINE BYTE COUNTER
4026- 85 08     0770            STA *COUNT1
4028- A9 1D     0780            LDA #29       ;= 203/7
402A- 85 07     0790            STA *LINECOUNT
                0800
                0810 ;8 BYTE BLOCK TRANSFER LOOP
                0820
402C- A5 02     0830 START      LDA *VISORG
402E- 85 00     0840            STA *VISMEM
4030- A5 03     0850            LDA *VISORG+1
4032- 85 01     0860            STA *VISMEM+1
                0870
4034- A9 1B     0880 GRAPHICS   LDA #27       ;ESC NEW LINE SET UP
4036- 20 47 8A  0890            JSR OUTCHR
4039- A9 4B     0900            LDA #'K       ;FOR GRAPHICS MODE
403B- 20 47 8A  0910            JSR OUTCHR
403E- A9 40     0920            LDA #64       ;IE 320 - 256
4040- 20 47 8A  0930            JSR OUTCHR
4043- A9 01     0940            LDA #1        ;EQUIVALENT TO 256
4045- 20 47 8A  0950            JSR OUTCHR
4048- A0 00     0960            LDY #0
404A- B1 00     0970 BLOOP      LDA (VISMEM),Y
404C- 91 04     0980            STA (BLKPNT),Y
                0990 ;MOVE UP BY 40
404E- 18        1000            CLC
404F- A9 27     1010            LDA #39
4051- 65 00     1020            ADC *VISMEM
4053- 85 00     1030            STA *VISMEM
4055- 90 02     1040            BCC CONT
4057- E6 01     1050            INC *VISMEM+1
                1060
                1070 ;NOW 40 BYTES ON
                1080
4059- C8        1090 CONT       INY
405A- C0 07     1100            CPY #7
405C- D0 EC     1110            BNE BLOOP
                1120
                1130 ;NOW HAVE 7 BYTES IN BLKPNT
                1140 ;SEND BLOCK OF 8 TO PRINTER
                1150
405E- 20 A8 40  1160            JSR TRANSPOSE
                1170
```

(continued to page 12-6)

NOTES ON THE FDC-1

Since most (all?) FDC-1 owners read SYM-Physis, we'll communicate with them through these pages. First, some corrections to the documentation:

1) Chip U5 as supplied with the kit is a 74367 (non-inverting buffer). Correct Appendix F (Chip Functions) and the schematic to conform.

2) Jumpers J1 (-1793) and J2 (-2732) are already present as traces on the lower side of the board.

3) Chip U9 is an 82S129 (not 825129).

4) Replace " -$ØFFF. At $A62A- " with " -$ØFFF, at $A62A- " on p. 5-1.

5) Replace "1 for single density" with "1 for double density" on p. 3-1.

6) Replace "ABCD Ø" with "ABCD-Ø" on p. 5-3.

7) Move the "(default)"s to follow "Single" and "128" on p. 5-9.

We are ordering one specially burned 82S129 PROM with the 1791 registers on page $AE (not $FØ), and the control port on page $AF (not $F1), so that the 2K block from $FØØØ-$F7FF can be freed for better uses. If there is sufficient demand for relocation to these pages, we'll order up a batch of them. See below for how to use them. Is there anyone out there who has facilities to burn these PROMS for others on a production or custom basis, and would like to do so?

ADDING MORE I/O CAPABILITY

In Issue 5/6 we described a simple method for cutting the memory space assigned to VIAs #2 and #3 from 1K each (four pages) down to only two pages each. This was done to permit installing the HDE FODS controller at A88Ø. This is right in the middle of a page, and unfortunately so, since the FDC-1 assigns whole pages to each of two sets of registers.

We will shortly have a SYM system capable of supporting both FODS and FDC simultaneously. Either controller will be switchable between 5" and 8" drives. This will make it possible to distribute (*) software in all four formats. Additionally, we will use the FODS subsystem as a development tool for the FDC subsystem by placing SYMDOS in RAM and booting to it from FODS. This should be lots of fun!

The FODS boot is at $FØØØ, and we want to leave it there, so we'll relocate the FDC's registers to $AEØØ and $AFØØ. Here's how:

Cut the trace (on topside of board) from pin 6 of U1Ø to a pass-through hole. Pin 6 is $\overline{AA8}$. Mount a 74LS32 upside down between chips U11 and U1Ø. Solder its pin 6 into the pass-through hole. This leads to the two VIAs $\overline{CS2}$. Pin 6 is the output of one of the four ORs on the 74LS32. Solder its pin 5 (an input to that OR) to pin 6 of U1Ø. Solder pin 14 of the 74LS32 to pin 16 of U11 (+5V) and pin 7 of the 74LS32 to pin 8 of U1Ø (GND). Then bring A9 from pin 22 of any one of the nearby ROMs to pin 4 of the 74LS32 (the other input to the OR). This completes the job, and it looks much neater than it sounds.

Note that in Issue 5/6 it was A7 that was brought to the second input of the OR. Since there are three unused ORs left in the 74LS32, you may cascade them to generate A9+A8 or A9+A8+A7, if you wish, to cut each of the VIAs down to a single or to a half page. To avoid having to relocate our FODS VIA from $A88Ø, we will use A9+A7.

(*) Others systems will be used for CODOS 8" and cassette distribution.

SUPPRESSING THE "ECHO" AT $F8ØØ

As we know, the 6502 expects its NMI, RST, and IRQ vectors to reside at $FFFA-$FFFF. During power-on, or after the RST key on the SYM has been pressed, the RST vector is "fetched" from the third and fourth bytes from the top of whatever chip is in socket PØ/U2Ø. This is normally SUPERMON, resident at $8ØØØ-$8FFF. It is, of course, possible to power-on reset (POR) to any other ROM socket just by changing the jumpers to N, P, R, and S from 19 and 2Ø. One of the very important functions of any POR program written for SYM is its own disabling (see lines 1502-Ø3 in the SUPERMON source listing). After this, all interrupt requests use the actual $FFFX addresses.

Note that jumper U-22 enables the Monitor RAM (SYSRAM), as well as everything else resident in the 2K block at $AØXX (jumper T-21), whenever the 2K select line $F8XX is active (low). Thus, the NMI and IRQ vectors are now obtained from SYSRAM, to which the default vectors were copied down from the top of SUPERMON on reset. While it is definitely an advantage to have these vectors in RAM rather than ROM or EPROM, so that they may be dynamically changed under program control, it "hurts" to lose the entire 2K block to this "echo" of the system RAM at the top of memory.

DEAN GARTH, in a recent letter, showed how the echo may be supressed by cutting jumper U-22, while still retaining the advantages of interrupt vectors in RAM. The 2K block from $F8ØØ-$FFFF may (must!) then be filled with an EPROM, although RAM will do if your POR program transfers the default vectors to it. If you wish your RST vector at $FFFC-D to be different from that in SUPERMON, you must disable the POR signal at jumper N-19. Your new NMI and IRQ vectors must now point to addresses within your EPROM in which you have placed indirect jumps to the appropriate SYSRAM locations, i.e., the top of your EPROM should contain a program similiar to the following:

```
                9000 ;SAMPLE "TOP OF THE EPROM" PROGRAM
                9010
                9020            .BA $FFF4
                9030 ;          .OS
                9040
                9050 NMIRAM     .DE $A67A
                9060 RESET      .DE $8B4A     ;OR, DO YOUR OWN THING
                9070 IRQRAM     .DE $A67E
                9080
FFF4- 6C 7A A6  9090 RAMNMI     JMP (NMIRAM)
FFF7- 6C 7E A6  9100 RAMIRQ     JMP (IRQRAM)
                9110
FFFA- F4 FF     9120            .SI RAMNMI
FFFC- 4A 8B     9130            .SE RESET
FFFE- F7 FF     9140            .SI RAMIRQ
                9150
                9160            .EN
```

GRAPHICS ON THE EPSON (continued from page 12-4)

```
                1180 ;STEP TO NEXT BYTE ADDRESS ADD ONE TO VISMEM
                1190 ;ORIGIN, I.E., VISORG
                1200
4061- E6 Ø2     1210            INC *VISORG
4063- A5 Ø2     1220            LDA *VISORG
4065- DØ Ø2     1230            BNE PASS
4067- E6 Ø3     1240            INC *VISORG+1
4069- A5 Ø2     1250 PASS       LDA *VISORG
406B- 85 ØØ     1260            STA *VISMEM
406D- A5 Ø3     1270            LDA *VISORG+1
406F- 85 Ø1     1280            STA *VISMEM+1
```

```
                    1290 ;
4071- A0 00         1300 SKIP1      LDY #0
4073- C6 08         1310            DEC *COUNT1  ;ALL 40 DONE?
4075- D0 D3         1320            BNE BLOOP
                    1330
                    1340 ;AT THIS POINT DONE 40 X 8 =320 IMPACTS
                    1350 ;TERMINATE THIS O/P LINE
                    1360
4077- A9 0D         1370            LDA #13      ;CR
4079- 20 47 8A      1380            JSR OUTCHR
407C- A9 0A         1390            LDA #10      ;LF
407E- 20 47 8A      1400            JSR OUTCHR
4081- 20 72 89      1410            JSR BEEP
4084- 20 86 83      1420            JSR INSTAT
4087- B0 13         1430            BCS OUT
4089- A9 28         1440            LDA #40
408B- 85 08         1450            STA *COUNT1  ;RESET COUNT TO 40
                    1460
                    1470 ; SHIFT VISORG 40X7=280 ALONG
                    1480
408D- 38            1490            SEC
408E- A5 02         1500            LDA *VISORG
4090- 69 EF         1510            ADC #239
4092- 90 02         1520            BCC MISS
4094- E6 03         1530            INC *VISORG+1
4096- 85 02         1540 MISS       STA *VISORG
4098- C6 07         1550            DEC *LINECOUNT
409A- D0 90         1560            BNE START
409C- A2 0C         1570 OUT        LDX #12
409E- 20 F4 40      1580            JSR SETSPC
40A1- 20 D8 40      1590            JSR VECSWAP
40A4- 20 C8 40      1600            JSR ZEROSWAP
40A7- 60            1610            RTS
                    1620
                    1630 ;ROUTINE MANIPULATES AND SENDS BLOCK OF 8 BYTES TO MX
                    1640
40A8- A0 00         1650 TRANSPOSE  LDY #0
40AA- A9 08         1660            LDA #8
40AC- 85 09         1670            STA *COUNT2
40AE- B1 04         1680 LOOP       LDA (BLKPNT),Y
40B0- 2A            1690            ROL A
40B1- 91 04         1700            STA (BLKPNT),Y       ;STORE IT BACK SHIFTED
40B3- 26 06         1710            ROL *CARRYSUM
40B5- C8            1720            INY
40B6- C0 07         1730            CPY #7
40B8- D0 F4         1740            BNE LOOP
40BA- A5 06         1750            LDA *CARRYSUM
40BC- 49 FF         1760            EOR #$FF     ;OPTIONAL INVERSION
40BE- 20 47 8A      1770            JSR OUTCHR
40C1- A0 00         1780            LDY #0
40C3- C6 09         1790            DEC *COUNT2
40C5- D0 E7         1800            BNE LOOP
40C7- 60            1810            RTS
                    1820
40C8- A2 09         1830 ZEROSWAP   LDX #TOP-BOTTOM-1
40CA- B5 00         1840            LDA *BOTTOM,X
40CC- BC E0 3F      1850            LDY ZEROSAVE,X
40CF- 94 00         1860            STY *BOTTOM,X
40D1- 9D E0 3F      1870            STA ZEROSAVE,X
40D4- CA            1880            DEX
40D5- 10 F3         1890            BPL ZEROSWAP+2
40D7- 60            1900            RTS
                    1910
40D8- 20 86 8B      1920 VECSWAP    JSR ACCESS
                    1930
```



```
40DB- AD 64 A6      1940            LDA OUTVEC
40DE- AC EA 3F      1950            LDY VECTORSAVE
40E1- 8D EA 3F      1960            STA VECTORSAVE
40E4- 8C 64 A6      1970            STY OUTVEC
                    1980
40E7- AD 65 A6      1990            LDA OUTVEC+1
40EA- AC EB 3F      2000            LDY VECTORSAVE+1
40ED- 8D EB 3F      2010            STA VECTORSAVE+1
40F0- 8C 65 A6      2020            STY OUTVEC+1
                    2030
40F3- 60            2040            RTS
                    2050
                    2060 ;SET TO [X] POINT SPACING
                    2070
40F4- A9 1B         2080 SETSPC     LDA #27      ;ESC
40F6- 20 47 8A      2090            JSR OUTCHR
40F9- A9 41         2100            LDA #'A
40FB- 20 47 8A      2110            JSR OUTCHR
40FE- 8A            2120            TXA
40FF- 20 47 8A      2130            JSR OUTCHR
4102- A9 0D         2140            LDA #13      ;CR
4104- 20 47 8A      2150            JSR OUTCHR
4107- A9 0A         2160            LDA #10      ;LF
4109- 20 47 8A      2170            JSR OUTCHR
410C- 60            2180            RTS
                    2190
                    2200 ;NOTE 0: BITS 0 THRU 6 OF THE "A" PORT ARE
                    2210 ;        THE OUTPUTS TO THE 7 LSB'S OF THE
                    2220 ;        EPSON. SINCE BIT 7 OF THE A REGISTER
                    2230 ;        IS ALWAYS ZERO ON CALLS TO OUTCHR
                    2240 ;        WHY "WASTE" PA7, WHEN WE CAN PUT IT
                    2250 ;        TO GOOD USE ELSEWHERE?
                    2260
                    2270 ;        THE MSB LINE OF THE EPSON MUST BE
                    2280 ;        TIED TO GROUND, SINCE IT IS NOT
                    2290 ;        DRIVEN BY THE SYM.
                    2300
                    2310 ;        BIT 7 OF THE "A" PORT IS THE "BUSY"
                    2320 ;        SIGNAL INPUT.
                    2330
                    2340 ;        CA2 IS THE "STROBE" SIGNAL OUTPUT.
                    2350
410D- 20 86 8B      2360 TURNON     JSR ACCESS
4110- A9 30         2370            LDA #L,PRINT
4112- 8D 64 A6      2380            STA OUTVEC
4115- A9 41         2390            LDA #H,PRINT
4117- 8D 65 A6      2400            STA OUTVEC+1
411A- AD 0C A8      2410            LDA PCR
411D- 29 F0         2420            AND #%11110000
411F- 09 0A         2430            ORA #%00001010
4121- 8D 0C A8      2440            STA PCR      ;SET FOR ONE-SHOT "HAND-SHAKE"
4124- A9 7F         2450            LDA #%01111111
4126- 8D 03 A8      2460            STA PADD
4129- A9 11         2470            LDA #$11     ;CTRL Q
412B- 20 47 8A      2480            JSR OUTCHR
412E- 18            2490            CLC
412F- 60            2500            RTS
                    2510
4130- 2C 0F A8      2520 PRINT      BIT PADHI
4133- 30 FB         2530            BMI PRINT
4135- 8D 01 A8      2540            STA PAD
4138- 60            2550            RTS
                    2560
                    2570            .EN
```

B&W GRAPHICS ON THE SYM

The SYM can be used to generate "typewriter-style" graphics on even as simple a terminal as the ASR-33 TTY, 72 columns wide by as long as desired. Of course, any printing terminal can be used. The SYM-PHYSIS logo used on Issues 0 through 6 (all of Volume I) were produced in this way, on a decwriter II (LA 36) printer, until Chuck Lundgren did the artwork for our current logo.

Video terminals, such as the KTM-3 or KTM-3/80, will work in the same manner, but with 40 or 80 columns, respectively, and, of course, only 24 lines long, and only for "soft"-copy. The KTM-2 and KTM-2/80, with their added graphics font, can provide more interesting graphics, and the use of the 16 2x2 block symbols permits doubling the number of point-elements across the width of the screen to 80 and 160, respectively.

A CRT terminal such as the KTM-2/80, which can display some 80x24 characters on the screen, stores each of these characters in one byte of RAM, and has a built in character generator to convert from ASCII to picture elements (pixels) during the scanning process. Less than 2K of RAM is needed (80 x 24 = 1920 bytes).

For high resolution graphics more RAM is required, typically around 8K, since each pixel requires one bit of RAM. A hardware character generator is now not required, but the hardware to scan the CRT and display each bit must be present. With static RAM (SRAM) the scanning process must be handled on a DMA (direct memory access) basis; with dynamic RAM (DRAM) the scanning is combined with the refresh.

A memory board with built in video generation capabilities is called a VDU (video display unit). Many SYMmers, both in the USA and abroad, have designed and built their own VDUs, but the video standards differ. Several of these individuals are exploring the possibilities of marketing two versions of their VDUs, NTSC (USA/Canada), and PAL/SECAM (most other places). We should very shortly be receiving a sample of one such unit for evaluation.

Meanwhile, for NTSC systems, the 8K Visible Memory, made by MTU, and available through the Users' Group, is one of the best VDUs available, with lots of software around. Visible Memories can be, and have been, combined, with bank switching to permit assigning them all to the same address block, for generating RGB color, providing a gray scale, or allowing for off-screen (invisible) editing.

The Epson MX-80 now comes with the Graftrax option installed (to meet the competition!), and many other printers in the same price range also have inbuilt point graphics capabilities. Thus you can get high resolution, hard copy, point graphics even without a VDU on which to edit and preview. Tom Gettys did some beautiful work with a very inexpensive printer and no VDU.

We paid extra for the FT option on the MX-80, thinking that we would be using the friction feed option for handling single sheets of preprinted letterheads, but have never once used it for that purpose. Nor have we ever used any of the paper rolls on the FT (we had some around from our TTY days). We did receive some printouts from someone on a roll of paper towel stock, however!

Our answer to the letterhead problem was to first get the graphics printing patch going (that's now been done) and then to design a letterhead for the Epson to generate on an as-requested basis. We would also then do a new logo for SYM-PHYSIS. Perhaps we should have a contest for our readers, offering a free lifetime subscription to the winner?

We print below a reproduction of extracts from a letter sent by one of our readers showing a very nice computer generated letterhead, done on a Centronics 739 printer. We wrote Mr. Wuethrich asking for permission to reproduce it; rather than answering our letter, he dropped in (all the way from Switzerland!) to give us his OK in person. Dan and a friend were our overnight guests. While here he picked up an FDC-1 kit to carry back with him. He had it assembled and ready for checkout on our test system in about 1 1/2 hours; it worked immediately!

ibw

INGENIEURBÜRO WÜTHRICH BRUGG

Hardware Mikroprozessor-Software Prozesssteuerungen Prototyp-Entwicklungen Kleinserien

ibw
Ingenieurbüro Wüthrich
Zimmermannstrasse 29
5200 Brugg

Tel: 056 414365

SYM-PHYSIS
SYM-1 Users' Group
P. O. Box 319
Chico, CA 95927
United States of America

Postcheck: 80-153983
SBG Brugg : DK 586.855 L1 Q

5200 Brugg, 3.19.82

Dear Jean,

For Your information some remarks about my system:
- SYM-1 expanded Memory-Mate Expansion Board
- 36 k RAM, 24 k ROM/EPROM, 150 I/O lines
- Write protect and parity check (9 bit RAM)
- EPROM-Programmer
- Synertek KTM-3 with Leedex Video-100 monoitor
- Centronics 739 Printer
- Marantz-Tape-Deck SD 1020 (2 speeds)

I would like to attach a Floppy- or Winchester-Disk to my system. Can You please answer the following questions:
- What type of disk-drive ?
- What type of disk-controller ?
- Do You sell a Software-driver for the SYM-1, so that I can still use all the features of BASIC and RAE together with the disk ?

I would be very glad if could write the answer of these question as soon as possible.

Finally just 5 words about Your SYM-issues: KEEP ON GOING LIKE THIS !!!

SYM-cerly

Dan

Daniel Wuethrich

MORE VISITORS, MORE FDC-1

Just the week before Dan's visit, Olivier Garbe, from Paris, France, also dropped by, for just a few hours, to pick up his FDC-1 kit! And, just a few weeks earlier, Ken Curry, whom we visited in Australia, spent the 4th of July weekend with us, viewing our local fireworks show (we were in Australia on Anzac Day).

Ken took ten FDC-1 kits back to Australia with him for resale. and left a fully expanded AIM 65 with us so that we could adapt the FDC-1 software (SYMDOS to AIMDOS[?]) to it. The SYM-1 can easily talk to the AIM 65 either through the KIM-1 cassette format or through the TTY interface using the "DEMON" punched paper tape format common to both systems. This should be a fun project, and will certainly take longer than even our most pessimistic estimate.

Ken runs Energy Control, P. O. Box 6502, Goodna, Australia, Phone (07) 288 2757 (near Brisbane; note the box number!). Energy Control is a distributor for both Rockwell International and Synertek Incorporated, and his catalog prices for their products are lower than any other prices we saw in the Australian magazine advertisements. He understands the products he sells, and fully supports those products. We suggest that our readers in Australia/New Zealand check with him, first, for hardware products, and with us for software and those hardware items he does not carry.

HOW TO USE THE NEW EPROMS

Table 4.3 of the SYM-1 Reference Manual shows how to install 2K (2316), 4K (2332), and 8K (2364) ROMS, and 2K (2716) EPROMS into the 24 pin sockets at U21, U22, U23, and U24.

The following note and the accompanying figures, provided by Alan L. Foster, Granville Technical College, New South Wales, Australia, should help you in installing the newer 4K and 8K EPROMS in these same sockets.

Notice that the 2732 and the 2532 differ in the choice of which pin is used for the A11 address line and in the polarity to be applied to the pin not used for A11. They are definitely NOT interchangeable!

Note also that while the 2532 and 2332 both use pin 18 as the A11 line, they differ in the polarity applied to pin 21, as do the 2716 and the "standard" 2316 (2316s can be found in non-standard versions, e. g., the KTM-2 master 2316 ROM has an active high CS).

The upshot is that either a 2516 or a 2716 may be substituted for a (standard) 2316, and a 2532 for a 2332 if pin 21 is moved from GND for the ROMs to +5 V for the EPROMS, and an MCM68764 directly for a 2364, once programmed, of course. On the MCM68764, pins 18, 19, and 21 are A11, A10, and A12, while pin 20 is $\overline{E}$/Vpp (enable low).

We appreciate Alan providing us with this very helpful summary of the available EPROM options; we had not known of the Motorola chip before.

EPROM PROBLEMS AND SYM COMPATIBILITY

One of the features which makes the SYM an ideal single board computer is the presence of the four sockets U20 - U23. These are normally dedicated to such chips as MON1.1, BAS1, RAE1 etc, but (assuming that 8k versions or "piggy-backs" are used) one normally has at least one socket free for user applications. If a 2k EPROM is placed in U21 say, there is no problem with the commonly available EPROMS. In this case, the Intel 2716 and the TMS2516 are interchangable. All the relevant chip select pins and address pins require the same voltage levels (see fig. 1). The only EPROM (ROM?) that requires a slightly different configuration is the Synertek 2316, which requires that the Vpp line (pin 21) be at 0 volts for a read, as opposed to the 2516/2716 which require pin 21 to be at +5 volts for a read. The 4k versions of these chips are a slightly different problem. Intel have decided to retain their two chip select lines (pins 18 and 20), and place the extra address line required (A11) onto pin 21. Texas have adopted a different philosophy by dropping one of the chip select lines, and replacing it with A11. (see fig. 2). This is still really no great problem, as the jumper options available on the SYM allow us to use either philosophy. So, what is the point of this article ? Simply, in the upgrade from 4k to 8k, both Intel and Texas have decided to opt for 28 pin versions, and 28 pins don't fit very well into the 24 pin SYM user sockets. (It can be done by using flying leads, but it's messy). The two companies have chosen this path, because they have their eyes on 16k and even 32k EPROMS in 28 pin packages, and they wish to provide pin compatible upgrades from the 8k chips. Motorola, on the other hand have just produced an 8k EPROM which is called the MCM68764, which, thankfully, is in a 24 pin package. Even more thankfully, it is

upwards compatible with the Texas philosophy, so for upgrading the approach to use is 2516/2716 to 2532 to MCM68764. All these chips require the same programming voltage (+25 volts), however, the 68764 requires that this only be applied for two milliseconds instead of the normal 50 milliseconds. This is easy to accomodate using any of the timers on the 6522's or the 6532. Incidentally, the Intel 2732A EPROM must not have +25 volts applied to pin 21. It only requires a programming supply of 21 volts. Exceeding 21.5 volts will blast the chip, not the data. Occasionally, 2732A's have been known to accidentally slip into a batch of 2732's, with consequent disastrous results for the purchasers.

Pin Number	Function		Read		Program		Standby	
	2716	2516	2716	2516	2716	2516	2716	2516
18	$\overline{CE}$	PD/$\overline{PGM}$	ϕ_V	ϕ_V	Pulsed	$\phi_V \rightarrow +5_V$	$+5_V$	$+5_V$
19	A_{10}	A_{10}	–	–	–	–	–	–
20	$\overline{OE}$	$\overline{CS}$	ϕ_V	ϕ_V	$+5_V$	$+5_V$	Don't Care	
21	V_{PP}	V_{PP}	$+5_V$	$+5_V$	$+25_V$	$+25_V$	$+5_V$	$+5_V$

Figure 1

	2732	2532	2732	2532	2732	2532	2732	2532
18	$\overline{CE}$	A_{11}	ϕ_V	–	ϕ_V	–	$+5_V$	–
19	A_{10}	A_{10}	–	–	–	–	–	–
20	$\overline{OE}/V_{PP}$	PD/$\overline{PGM}$	ϕ_V	ϕ_V	$+25_V$	Pulsed $V_{IH} \rightarrow V_{IL}$	Don't Care	$+5V$
21	A_{11}	V_{PP}	–	$+5_V$	–	$+25_V$	–	$+5V$

Figure 2

A CASSETTE DATA HANDLER - BY JOE HOBART

Below is a very interesting approach to implementing a very useful cassette utility into BAS-1. We have not tried it ourselves because we have been working mainly with disks, but it looks like it should do the job, and we also are familiar with the original Blalock version, which we did try. Joe is also into disks, himself, now, as he received one of the first half-dozen or so prototypes of the FDC-1 for testing.

For those who are curious about the machine language portion of the program, we have appended a disassembly, done with Dessaintes' Disassembler (DESDIS). This disasembler automatically creates a sorted .DE file, inserts the proper .BA, adds the .EN, or if the new source is too long, a .CT, and ";" lines after branches, jumps, and returns. After each #$XX it provides the ASCII equivalent of the XX as a ";" comment. In these comments "." indicates the sign bit is set, and the up-arrow indicates a control character. The labels are made up of the actual hex address where they were found, preceeded with Z for zero page, J for jump, B for branch, S for subroutine, or A for absolute.

The original DESDIS did a .CT (continue to tape), and Tom Gettys added the capability of .CT XXXXX, where XXXXX is a five character filename, forcing a continue to disk. Ever since, we've been disassembling everything we see!

PUTTING A CASSETTE BASED DATA SAVE/RELOAD ROUTINE IN A BASIC PROGRAM

Here is a technique for putting a machine language data save and re-load routine inside a BASIC program. This technique will work for any other machine language program as well. The save/reload routine is a modified version of one by John Blalock that appeared in the April, 1980, issue of MICRO magazine. It works with SYM BASIC alone and also with Brown's Terminal Control Patch.

The following steps will incorporate the routine into a BASIC program:

A. Enter the following as the first three lines of the BASIC program: (There are 49 X's in each line.)

```
1 REMXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
2 REMXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
3 REMXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
```

B. Exit BASIC to the monitor and change the contents of memory location $0201 from $38 to $A6 so BASIC will skip over lines 2 and 3.

C. Enter the following code from $0206 to $02A3:

```
0206 20 86 8B 20 88 81 8D 4E,35
020E A6 A9 01 29 10 8D 4D A6,3E
0216 8D 4B A6 A9 65 8D 4C A6,49
021E A9 EA 8D 4A A6 20 87 8E,8E
0226 A9 2A 20 47 8A A0 80 A5,17
022E 7D 8D 4C A6 A5 7E 8D 4D,10
0236 A6 A5 81 8D 4A A6 A5 82,80
023E 8D 4B A6 EE 4E A6 20 87,87
0246 8E A9 2A 20 47 8A A0 80,F9
024E A5 83 8D 4C A6 A5 84 8D,56
0256 4D A6 A5 87 8D 4A A6 A5,97
025E 88 8D 4B A6 EE 4E A6 20,9F
0266 87 8E 4C C4 81 20 86 8B,76
026E 20 88 81 8D 4E A6 A5 D3,98
0276 85 EE A5 D4 85 F1 20 78,92
027E 8C A9 2A 20 47 8A A0 80,02
0286 EE 4E A6 20 78 8C A9 2A,DB
028E 20 47 8A A0 80 EE 4E A6,CE
0296 20 78 8C A5 EE 85 D3 A5,82
029E F1 85 D4 4C C4 81,5D
 4F5D
```

D. Verify the machine code to ensure accuracy.

E. Return to BASIC. A list of the program will show a long and unusual looking line number 1. Lines 2 and 3 will no longer exist.

I use the following BASIC subroutines to call the save/reload routine:

```
50000 REM  *CASSETTE DATA SAVE SUBROUTINE*
50010 Q=FRE(0)
50020 PRINT"START THE CASSETTE IN RECORD MODE AND PRESS ANY KEY   ";
50030 Q=USR(-30120,-11957,0) : PRINTCHR$(Q/256)
50040 Q=USR(&"0206",384)
50050 PRINT"DATA SAVED" : RETURN

60000 REM  *CASSETTE DATA RELOAD ROUTINE*
60010 PRINT"START CASSETTE PLAYBACK"
60020 Q=USR(&"026B,384)
60030 PRINT"DATA LOADED" : RETURN
```


A few comments and cautions are in order. The addresses in statements number 50040 and 60020 assume the machine code resides from $0206 to $02A3. Statement 50010 compresses the string storage area to eliminate superseded strings. Statement 50030 is a neat GETKEY and PRINT function that I use in almost all my programs. Once data has been saved from a BASIC program, the overall length of that program must not be changed if the data is to be reloaded successfully. This technique may be used with other machine code, but since BASIC uses $00 as a delimiter between each line, $00 cannot be used in code so saved.

The machine language is completely relocatable. It can be added to an existing program as well as used to begin a new one. I have had very good results using this save/reload routine with a Line Oriented Text Editor (COMPUTE for February, 1982) and with several adventure games. To save time, I recorded the machine code on tape ($0206-$02A3) and just load it in for step C above instead of having to type it in each time.

```
                    0010 Z7D    .DE $7D
                    0020 Z7E    .DE $7E
                    0030 Z81    .DE $81
                    0040 Z82    .DE $82
                    0050 Z83    .DE $83
                    0060 Z84    .DE $84
                    0070 Z87    .DE $87
                    0080 Z88    .DE $88
                    0090 ZD3    .DE $D3
                    0100 ZD4    .DE $D4
                    0110 ZEE    .DE $EE
                    0120 ZF1    .DE $F1
                    0130 S8188  .DE $8188
                    0140 J81C4  .DE $81C4
                    0150 S8A47  .DE $8A47
                    0160 S8B86  .DE $8B86
                    0170 S8C78  .DE $8C78
                    0180 S8E87  .DE $8E87
                    0190 AA64A  .DE $A64A
                    0200 AA64B  .DE $A64B
                    0210 AA64C  .DE $A64C
                    0220 AA64D  .DE $A64D
                    0230 AA64E  .DE $A64E
                    0240 ;
                    0250        .BA $0206
0206- 20 86 8B      0260        JSR S8B86
0209- 20 88 81      0270        JSR S8188
020C- 8D 4E A6      0280        STA AA64E
020F- A9 01         0290        LDA #$01
0211- 29 10         0300        AND #$10
0213- 8D 4D A6      0310        STA AA64D
0216- 8D 4B A6      0320        STA AA64B
0219- A9 65         0330        LDA #$65
021B- 8D 4C A6      0340        STA AA64C
021E- A9 EA         0350        LDA #$EA
0220- 8D 4A A6      0360        STA AA64A
0223- 20 87 8E      0370        JSR S8E87
0226- A9 2A         0380        LDA #$2A
0228- 20 47 8A      0390        JSR S8A47
022B- A0 80         0400        LDY #$80
022D- A5 7D         0410        LDA *Z7D
022F- 8D 4C A6      0420        STA AA64C
0232- A5 7E         0430        LDA *Z7E
0234- 8D 4D A6      0440        STA AA64D
0237- A5 81         0450        LDA *Z81
0239- 8D 4A A6      0460        STA AA64A
023C- A5 82         0470        LDA *Z82
023E- 8D 4B A6      0480        STA AA64B
0241- EE 4E A6      0490        INC AA64E
0244- 20 87 8E      0500        JSR S8E87
0247- A9 2A         0510        LDA #$2A
0249- 20 47 8A      0520        JSR S8A47
024C- A0 80         0530        LDY #$80
024E- A5 83         0540        LDA *Z83
0250- 8D 4C A6      0550        STA AA64C
0253- A5 84         0560        LDA *Z84
0255- 8D 4D A6      0570        STA AA64D
0258- A5 87         0580        LDA *Z87
025A- 8D 4A A6      0590        STA AA64A
025D- A5 88         0600        LDA *Z88
025F- 8D 4B A6      0610        STA AA64B
0262- EE 4E A6      0620        INC AA64E
0265- 20 87 8E      0630        JSR S8E87
0268- 4C C4 81      0640        JMP J81C4
                    0650 ;
026B- 20 86 8B      0660        JSR S8B86
026E- 20 88 81      0670        JSR S8188
0271- 8D 4E A6      0680        STA AA64E
0274- A5 D3         0690        LDA *ZD3
0276- 85 EE         0700        STA *ZEE
0278- A5 D4         0710        LDA *ZD4
027A- 85 F1         0720        STA *ZF1
027C- 20 78 8C      0730        JSR S8C78
027F- A9 2A         0740        LDA #$2A
0281- 20 47 8A      0750        JSR S8A47
0284- A0 80         0760        LDY #$80
0286- EE 4E A6      0770        INC AA64E
0289- 20 78 8C      0780        JSR S8C78
028C- A9 2A         0790        LDA #$2A
028E- 20 47 8A      0800        JSR S8A47
0291- A0 80         0810        LDY #$80
0293- EE 4E A6      0820        INC AA64E
0296- 20 78 8C      0830        JSR S8C78
0299- A5 EE         0840        LDA *ZEE
029B- 85 D3         0850        STA *ZD3
029D- A5 F1         0860        LDA *ZF1
029F- 85 D4         0870        STA *ZD4
02A1- 4C C4 81      0880        JMP J81C4
                    0890 ;
                    0900        .EN 01
```

THREE FROM AUSTRALIA

Dear Lux:

Enclosed are three programs which may be suitable for publication in SYM-PHYSIS.

First, there are two versions of a machine language program written by my colleague, Dr. M. A. Cusiter, which will sort BASIC string arrays by sorting the pointers, instead of the strings themselves. Hence it is an extremely fast sort. Note that if there are two or more arrays to be sorted, they must have the same dimensions.

The others are a program to provide BASIC with automatic line numbering, and one which will put a margin on the left of any printout.

Yours faithfully,

Alan Foster
28 Gavin Place, Kings Langley,
N.S.W., Australia, 2147

The following are two versions of an extremely fast machine language program for sorting BASIC strings.

In each case there is an example of the operation of the program followed by a listing of the program.

The first version allows a number of string arrays to be sorted independently of each other, while the second sorts a number of arrays according to the first array.

In each case the first array must be the array Z$(X), where X must be one greater than the number of elements to be sorted. The other arrays to be sorted must immediately follow Z$(X) in memory. The easiest way to ensure this is to use a DIM statement as in the examples.

The programs are called by J=USR(&"START",N) where N is the number of arrays to be sorted after the first, and START is the address assigned to the label START at the beginning of the machine language program.

```
0010             .BA $2000-$10E
0020 ; OS
0030 ;
0040 ;*****************************************
0050 ;*      BASIC STRING SORT PROGRAM        *
0060 ;*       WRITTEN BY M.A.CUSITER          *
0070 ;*          AND A.L.FOSTER               *
0080 ;*****************************************
0090 ;
0100 ;
0110 ;This program sorts a number of BASIC string arrays.
0120 ;First array to be sorted must be the matrix Z$(X)
0130 ;where X must ALWAYS be at least one greater
0140 ;than the number of array elements to be sorted.
0150 ;
0160 ;The number of arrays subsequent to Z$ to be sorted
0170 ;is passed to BASIC via the user command:
0180 ;       J=USR(&"START",N)
0190 ;where START is the start address of this program
0200 ;and N is the number of subsequent arrays.
0210 ;If there is only one array, then N=0.
0220 ;These arrays can have any name.
0230 ;
```



```
0240 ;NOTE:
0250 ;Subsequent arrays must have the same dimensions as Z$.
0260 ;The zero elements must be used.
0270 ;
0280 ;
0290              ;STORAGE IS BEHIND THE PROGRAM
0300 ZSTORE       .DI END+1
0310 TOUT         .DE $8AA0
0320 AVST         .DE $7F
0330 CURLEN       .DE $81
0340 CURSTRT      .DE $82
0350 NXTLEN       .DE $84
0360 NXTSTRT      .DE $85
0370 STRSTRT      .DE $87
0380 CHECKFL      .DE $89
0390 AVST1        .DE $8A
0400 COUNT        .DE $8C
0410 ;
1EF2- 0D 0A 53 0420 MESSAGE     .BY $0D $0A 'STRING NOT FOUND ' $0D $0A $00
1EF5- 54 52 49
1EF8- 4E 47 20
1EFB- 4E 4F 54
1EFE- 20 46 4F
1F01- 55 4E 44
1F04- 20 0D 0A
1F07- 00
                 0430              ;ENTRY POINT - COPY Z PAGE VECS
1F08- A2 0F      0440 START        LDX #15
1F0A- B5 7E      0450 COPY         LDA *AVST-1,X
1F0C- 9D EA 1F   0460              STA ZSTORE-1,X
1F0F- CA         0470              DEX
1F10- D0 F8      0480              BNE COPY
1F12- 86 89      0490              STX *CHECKFL
1F14- 84 8C      0500              STY *COUNT    ;GET No. OF STR TO SORT
1F16- A0 00      0510 FINDZ        LDY #0
1F18- B1 7F      0520              LDA (AVST),Y
1F1A- C9 5A      0530              CMP #'Z
1F1C- F0 4A      0540              BEQ SORTSTRT          ;FOUND Z$
1F1E- C8         0550              INY
1F1F- C8         0560              INY
1F20- B1 7F      0570              LDA (AVST),Y          ;GET LO STRT NXT STR
1F22- 18         0580              CLC
1F23- 65 7F      0590              ADC *AVST     ;ADD TO LAST ADDR
1F25- AA         0600              TAX
1F26- C8         0610              INY  ;         HI BYTE
1F27- B1 7F      0620              LDA (AVST),Y
1F29- 65 80      0630              ADC *AVST+1
1F2B- 85 80      0640              STA *AVST+1
1F2D- 86 7F      0650              STX *AVST     ;DONE
1F2F- A5 80      0660 CHECK        LDA *AVST+1   ;CHECK TO SEE IF AT
1F31- C5 82      0670              CMP *AVST+3   ; END OF STRINGS
1F33- D0 E1      0680              BNE FINDZ
1F35- A5 7F      0690              LDA *AVST
1F37- C5 81      0700              CMP *AVST+2
1F39- D0 DB      0710              BNE FINDZ
1F3B- A0 00      0720              LDY #0
1F3D- B9 F2 1E   0730 MESS         LDA MESSAGE,Y         ;STRING NOT FOUND
1F40- F0 19      0740              BEQ OUT
1F42- 20 A0 8A   0750              JSR TOUT
1F45- C8         0760              INY
1F46- D0 F5      0770              BNE MESS
1F48- F0 11      0780              BEQ OUT
1F4A- A5 8C      0790 OUT1         LDA *COUNT
1F4C- F0 0D      0800              BEQ OUT
```

```
1F4E- C6 8C     0810            DEC *COUNT
1F50- A5 8A     0820            LDA *AVST1
1F52- 85 7F     0830            STA *AVST
1F54- A5 8B     0840            LDA *AVST1+1
1F56- 85 80     0850            STA *AVST+1
1F58- 4C 68 1F  0860            JMP SORTSTRT
                0870 ;
1F5B- A2 0F     0880 OUT        LDX #15       ;RESTORE ZPAGE
1F5D- BD EA 1F  0890 PUTBACK    LDA ZSTORE-1,X
1F60- 95 7E     0900            STA *AVST-1,X
1F62- CA        0910            DEX
1F63- D0 F8     0920            BNE PUTBACK
1F65- 4C 4C D1  0930            JMP $D14C     ;BACK TO BASIC
                0940 ;
1F68- C8        0950 SORTSTRT   INY
1F69- C8        0960            INY
1F6A- B1 7F     0970            LDA (AVST),Y
1F6C- 18        0980            CLC
1F6D- 65 7F     0990            ADC *AVST
1F6F- 85 8A     1000            STA *AVST1
1F71- C8        1010            INY
1F72- B1 7F     1020            LDA (AVST),Y
1F74- 65 80     1030            ADC *AVST+1
1F76- 85 8B     1040            STA *AVST1+1
1F78- A2 07     1050            LDX #7
1F7A- 20 E1 1F  1060            JSR INCPTR
1F7D- A5 7F     1070            LDA *AVST
1F7F- A6 80     1080            LDX *AVST+1  ;         MOVE OVER CONTROL BYTES
1F81- 85 87     1090            STA *STRSTRT          ;AND STORE FIRST ELEMENT
1F83- 86 88     1100            STX *STRSTRT+1        ;ADDR. IN Z-PAGE
                1110 ;
1F85- A0 03     1120 SORT       LDY #3
1F87- B1 7F     1130            LDA (AVST),Y          ;GET NEXT ELEMENT LENGTH
1F89- D0 10     1140            BNE CONT      ;ZERO IF AT END
1F8B- A5 89     1150            LDA *CHECKFL
1F8D- F0 BB     1160            BEQ OUT1      ;FINISHED !
1F8F- A5 87     1170            LDA *STRSTRT          ;NO, ANOTHER PASS
1F91- A6 88     1180            LDX *STRSTRT+1
1F93- 85 7F     1190            STA *AVST     ;SETUP AVST FOR
1F95- 86 80     1200            STX *AVST+1   ;ANOTHER GO
1F97- A9 00     1210            LDA #0
1F99- 85 89     1220            STA *CHECKFL          ;RESET CHECKFL
1F9B- A0 FF     1230 CONT       LDY #$FF
1F9D- C8        1240 SETUP      INY
1F9E- B1 7F     1250            LDA (AVST),Y          ;SETUP TWO ELS. INTO Z
1FA0- 99 81 00  1260            STA AVST+2,Y          ;PAGE
1FA3- C0 05     1270            CPY #5
1FA5- D0 F6     1280            BNE SETUP
1FA7- A0 00     1290            LDY #0
1FA9- 18        1300            CLC
1FAA- B1 85     1310 COMPARE    LDA (NXTSTRT),Y
1FAC- D1 82     1320            CMP (CURSTRT),Y
1FAE- 90 12     1330            BCC EXCHANGE
1FB0- D0 09     1340            BNE NXTSTR    ;IN RIGHT ORDER
1FB2- C8        1350            INY  ;         NEXT STRING CHAR.
1FB3- C4 81     1360            CPY *CURLEN   ;END OF CURRENT STR?
1FB5- F0 04     1370            BEQ NXTSTR
1FB7- C4 84     1380            CPY *NXTLEN   ;END OF NEXT STR?
1FB9- D0 EF     1390            BNE COMPARE
1FBB- A2 03     1400 NXTSTR     LDX #3
1FBD- 20 E1 1F  1410            JSR INCPTR
1FC0- F0 C3     1420            BEQ SORT
1FC2- A2 01     1430 EXCHANGE   LDX #1
1FC4- A0 03     1440            LDY #3
1FC6- B5 80     1450 SHIFT1     LDA *AVST+1,X
```



```
1FC8- 91 7F     1460            STA (AVST),Y          ;PUT CURRENT PTRS
1FCA- E8        1470            INX  ;                 IN NEXT STR
1FCB- C8        1480            INY
1FCC- C0 06     1490            CPY #6
1FCE- D0 F6     1500            BNE SHIFT1
1FD0- A0 00     1510            LDY #0
1FD2- B5 80     1520 SHIFT2     LDA *AVST+1,X
1FD4- 91 7F     1530            STA (AVST),Y          ;PUT NXT STR. PTRS
1FD6- E8        1540            INX  ;                 IN CURRENT STR
1FD7- C8        1550            INY
1FD8- C0 03     1560            CPY #3
1FDA- D0 F6     1570            BNE SHIFT2
1FDC- 86 89     1580            STX *CHECKFL          ;SET CHECKFL
1FDE- 4C BB 1F  1590            JMP NXTSTR
                1600 ;
1FE1- E6 7F     1610 INCPTR     INC *AVST
1FE3- D0 02     1620            BNE NEXTX
1FE5- E6 80     1630            INC *AVST+1
1FE7- CA        1640 NEXTX      DEX
1FE8- D0 F7     1650            BNE INCPTR
1FEA- 60        1660 END        RTS
                1670            .EN

                0010            .BA $2000-$146
                0020 ; .OS
                0030 ;
                0040 ;*****************************************
                0050 ;*       BASIC STRING SORT PROGRAM       *
                0060 ;*        WRITTEN BY M.A.CUSITER         *
                0070 ;*          AND A.L.FOSTER               *
                0080 ;*****************************************
                0090 ;
                0100 ;
                0110 ;This program sorts a number of BASIC string arrays.
                0120 ;The first array, Z$(X), is sorted into alphabetical
                0130 ;sequence; subsequent arrays are sorted in the same
                0140 ;order as the elements of Z$(X).
                0150 ;where X must ALWAYS be at least one greater than
                0160 ;the number of elements to be sorted.
                0170 ;
                0180 ;The number of arrays subsequent to Z$ to be sorted
                0190 ;is passed to BASIC via the user command:
                0200 ;        J=USR(&"START",N)
                0210 ;where START is the start address of this program
                0220 ;and N is the number of subsequent arrays.
                0230 ;These arrays can have any name.
                0240 ;
                0250 ;NOTE:
                0260 ;Subsequent arrays must have the same dimensions as Z$.
                0270 ;The zero elements must be used.
                0280 ;
                0290 ;
                0300            ;STORAGE IS BEHIND PROGRAMME
                0310 ZSTORE     .DI END+1
                0320            ;
                0330            ;MONITOR ROUTINE USED
                0340            ;
                0350 TOUT       .DE $8AA0
                0360            ;
                0370            ; ZERO PAGE DEFINITIONS
                0380            ;
                0390            ;
                0400 AVST       .DE $7F
                0410 CURLEN     .DE $81
                0420 CURSTRT    .DE $82
```

```
                0430 NXTLEN     .DE $84
                0440 NXTSTRT    .DE $85
                0450 STRSTRT    .DE $87
                0460 COUNT      .DE $89
                0470 CHECKFL    .DE $8A
                0480 NXTSTRNG   .DE $8B
                0490 STORE      .DE $8D
                0500            ;
1EBA- 0D 0A 53  0510 MESSAGE    .BY $0D $0A 'STRING NOT FOUND ' $0A $0D $00
1EBD- 54 52 49
1EC0- 4E 47 20
1EC3- 4E 4F 54
1EC6- 20 46 4F
1EC9- 55 4E 44
1ECC- 20 0A 0D
1ECF- 00
                0520            ;ENTRY POINT - COPY Z PAGE VECS
1ED0- A2 1C     0530 START      LDX #28
1ED2- B5 7E     0540 COPY       LDA *AVST-1,X
1ED4- 9D D7 1F  0550            STA ZSTORE-1,X
1ED7- CA        0560            DEX
1ED8- D0 F8     0570            BNE COPY
1EDA- 86 8A     0580            STX *CHECKFL
1EDC- 84 89     0590            STY *COUNT    ;GET No. STR TO SORT
1EDE- A0 00     0600 FINDZ      LDY #0
1EE0- B1 7F     0610            LDA (AVST),Y
1EE2- C9 5A     0620            CMP #'Z
1EE4- F0 37     0630            BEQ SORTSTRT          ;FOUND Z$
1EE6- C8        0640            INY
1EE7- C8        0650            INY
1EE8- B1 7F     0660            LDA (AVST),Y          ;GET LO STRT NXT STR
1EEA- 18        0670            CLC
1EEB- 65 7F     0680            ADC *AVST     ;ADD TO LAST ADDR
1EED- AA        0690            TAX
1EEE- C8        0700            INY  ;         HI BYTE
1EEF- B1 7F     0710            LDA (AVST),Y
1EF1- 65 80     0720            ADC *AVST+1
1EF3- 85 80     0730            STA *AVST+1
1EF5- 86 7F     0740            STX *AVST     ;DONE
1EF7- A5 80     0750 CHECK      LDA *AVST+1   ;CHECK TO SEE IF AT
1EF9- C5 82     0760            CMP *AVST+3   ;END OF STRINGS
1EFB- D0 E1     0770            BNE FINDZ
1EFD- A5 7F     0780            LDA *AVST
1EFF- C5 81     0790            CMP *AVST+2
1F01- D0 DB     0800            BNE FINDZ
1F03- A0 00     0810            LDY #0
1F05- B9 BA 1E  0820 MESS       LDA MESSAGE,Y         ;STRING NOT FOUND
1F08- F0 06     0830            BEQ OUT
1F0A- 20 A0 8A  0840            JSR TOUT
1F0D- C8        0850            INY
1F0E- D0 F5     0860            BNE MESS
                0870 ;
1F10- A2 1C     0880 OUT        LDX #28       ;RESTORE ZPAGE
1F12- BD D7 1F  0890 PUTBACK    LDA ZSTORE-1,X
1F15- 95 7E     0900            STA *AVST-1,X
1F17- CA        0910            DEX
1F18- D0 F8     0920            BNE PUTBACK
1F1A- 4C 4C D1  0930            JMP $D14C     ;BACK TO BASIC
                0940 ;
1F1D- C8        0950 SORTSTRT   INY
1F1E- C8        0960            INY
1F1F- B1 7F     0970            LDA (AVST),Y          ;RECORD REL ADDR
1F21- 85 8B     0980            STA *NXTSTRNG         ; OF NXT STR
1F23- C8        0990            INY
1F24- B1 7F     1000            LDA (AVST),Y
```



```
1F26- 85 8C     1010            STA *NXTSTRNG+1       ;LO,HI
1F28- A2 07     1020            LDX #7
1F2A- 20 A4 1F  1030            JSR INCPTR
1F2D- A5 7F     1040            LDA *AVST
1F2F- A6 80     1050            LDX *AVST+1  ;         MOVE OVER CONTROL BYTES
1F31- 85 87     1060            STA *STRSTRT          ;AND STORE FIRST ELEMEMT
1F33- 86 88     1070            STX *STRSTRT+1        ;ADDR. IN Z-PAGE
                1080            ;
1F35- A0 03     1090 SORT       LDY #3
1F37- B1 7F     1100            LDA (AVST),Y          ;GET NEXT ELEMENT LENGTH
1F39- D0 10     1110            BNE CONT      ;ZERO IF AT END
1F3B- A5 8A     1120            LDA *CHECKFL
1F3D- F0 D1     1130            BEQ OUT       ;         FINISHED !!
1F3F- A5 87     1140            LDA *STRSTRT          ;NO, ANOTHER PASS
1F41- A6 88     1150            LDX *STRSTRT+1
1F43- 85 7F     1160            STA *AVST     ;SETUP AVST FOR
1F45- 86 80     1170            STX *AVST+1   ;ANOTHER GO
1F47- A9 00     1180            LDA #0
1F49- 85 8A     1190            STA *CHECKFL          ;RESET CHECKFL
1F4B- 20 CB 1F  1200 CONT       JSR SETUP
1F4E- A0 00     1210            LDY #0
1F50- 18        1220            CLC
1F51- B1 85     1230 COMPARE    LDA (NXTSTRT),Y
1F53- D1 82     1240            CMP (CURSTRT),Y
1F55- 90 12     1250            BCC CHECKCNT
1F57- D0 09     1260            BNE NXTSTR    ;IN RIGHT ORDER
1F59- C8        1270            INY  ;         NEXT STRING CHAR.
1F5A- C4 81     1280            CPY *CURLEN   ;END OF CURRENT STR?
1F5C- F0 04     1290            BEQ NXTSTR
1F5E- C4 84     1300            CPY *NXTLEN   ;END OF NEXT STR?
1F60- D0 EF     1310            BNE COMPARE
1F62- A2 03     1320 NXTSTR     LDX #3
1F64- 20 A4 1F  1330            JSR INCPTR
1F67- F0 CC     1340            BEQ SORT
                1350 ;
1F69- A5 89     1360 CHECKCNT   LDA *COUNT    ;MORE STRINGS?
1F6B- D0 06     1370            BNE SAVE.PTRS
1F6D- 20 AE 1F  1380            JSR EXCHANGE
1F70- 4C 62 1F  1390            JMP NXTSTR
                1400 ;
1F73- A2 0B     1410 SAVE.PTRS  LDX #11       ;         REMEMBER WHERE WE ARE
1F75- B5 7E     1420 SV.PTRS    LDA *AVST-1,X         ;WITH FIRST STRING
1F77- 95 8C     1430            STA *STORE-1,X        ;SO WE CAN RETURN
1F79- CA        1440            DEX
1F7A- D0 F9     1450            BNE SV.PTRS
1F7C- 20 AE 1F  1460            JSR EXCHANGE
1F7F- C6 89     1470 MORESTR    DEC *COUNT    ;ONE LESS TO GO
1F81- 18        1480            CLC
1F82- A5 7F     1490            LDA *AVST     ;YES, SO POINT TO IT
1F84- 65 8B     1500            ADC *NXTSTRNG
1F86- 85 7F     1510            STA *AVST
1F88- A5 80     1520            LDA *AVST+1
1F8A- 65 8C     1530            ADC *NXTSTRNG+1
1F8C- 85 80     1540            STA *AVST+1
1F8E- 20 CB 1F  1550            JSR SETUP
1F91- 20 AE 1F  1560            JSR EXCHANGE          ;FOR THIS ARRAY TOO!
1F94- A5 89     1570            LDA *COUNT    ;MORE ARRAYS?
1F96- D0 E7     1580            BNE MORESTR   ;YES
                1590 ;
1F98- A2 0B     1600 RESTORE    LDX #11
1F9A- B5 8C     1610 LOOP       LDA *STORE-1,X
1F9C- 95 7E     1620            STA *AVST-1,X
1F9E- CA        1630            DEX
1F9F- D0 F9     1640            BNE LOOP
1FA1- 4C 62 1F  1650            JMP NXTSTR
```

```
                    1660 ;
1FA4- E6 7F         1670 INCPTR     INC *AVST
1FA6- D0 02         1680            BNE NEXTX
1FA8- E6 80         1690            INC *AVST+1
1FAA- CA            1700 NEXTX      DEX
1FAB- D0 F7         1710            BNE INCPTR
1FAD- 60            1720            RTS
                    1730 ;
1FAE- A2 01         1740 EXCHANGE   LDX #1
1FB0- A0 03         1750            LDY #3
1FB2- B5 80         1760 SHIFT1     LDA *AVST+1,X       ;PUT CURRENT PTRS
1FB4- 91 7F         1770            STA (AVST),Y        ; IN NEXT STR
1FB6- E8            1780            INX
1FB7- C8            1790            INY
1FB8- C0 06         1800            CPY #6
1FBA- D0 F6         1810            BNE SHIFT1
1FBC- A0 00         1820            LDY #0
1FBE- B5 80         1830 SHIFT2     LDA *AVST+1,X       ;PUT NEXT STR PTRS
1FC0- 91 7F         1840            STA (AVST),Y        ; IN CURRENT STR
1FC2- E8            1850            INX
1FC3- C8            1860            INY
1FC4- C0 03         1870            CPY #3
1FC6- D0 F6         1880            BNE SHIFT2
1FC8- 86 8A         1890            STX *CHECKFL        ;SET CHECKFL
1FCA- 60            1900            RTS
                    1910 ;
1FCB- A0 FF         1920 SETUP      LDY #$FF
1FCD- C8            1930 SETUP.1    INY
1FCE- B1 7F         1940            LDA (AVST),Y        ;SETUP TWO ELS. INTO
1FD0- 99 81 00      1950            STA AVST+2,Y        ; Z PAGE
1FD3- C0 05         1960            CPY #5
1FD5- D0 F6         1970            BNE SETUP.1
1FD7- 60            1980 END        RTS
                    1990            .EN
```

The following program provides BASIC with an automatic line numbering facility. It works fine as it is, however it should probably be seen as a starting point for an extended BASIC package, or perhaps it could be built into a BASIC control patch such as the one recently published in SYM-PHYSIS.

The program is patched to BASIC via INVEC. G 1B00 will cold start BASIC with the auto line numbering feature included.

To start auto line numbering type CONTROL Q. The start line and increment may then be chosen by giving values to the variables A% and B%. For example, A%=100:B%=5 will cause numbering to start at 100 with an increment of 5. Either or both of these values may be assigned, or CONTROL Q may be followed by a carriage return only. This results in default values of 10 for both start line and increment.

After the last program line has been typed, CONTROL R will feed a carriage return to BASIC and exit auto mode.

Other features are: CONTROL C allows exit to monitor; return to BASIC with G <cr> or G 0 <cr>. Lower case input is possible.

Note that there is a flag in page zero which is used to monitor the state of the program. There are five states:

State 0 - Not in auto mode.
State 1 - Partly set up - waiting for A%, B%.
State 2 - Almost set up - output first line number.
State 3 - Output line number.
State 4 - Type characters into line.


```
                    0010 ;**********************************************
                    0020 ;*                                            *
                    0030 ;*       AUTOMATIC LINE NUMBERING             *
                    0040 ;*                FOR BASIC                   *
                    0050 ;*                                            *
                    0060 ;*         WRITTEN BY A.L.FOSTER              *
                    0070 ;*            MARCH 1982                      *
                    0080 ;*                                            *
                    0090 ;**********************************************
                    0100 ;
                    0110 ACCESS     .DE $8B86
                    0120 CRLF       .DE $834D
                    0130 TOUT       .DE $8AA0
                    0140 INTCHR     .DE $8A58
                    0150 INVEC      .DE $A660
                    0160 BASCOLD    .DE $DE6D
                    0170 WARMVEC    .DE $0
                    0180 BASWARM    .DE $C27E
                    0190 ;
                    0200 BUF        .DE $1E
                    0210 V.PTR      .DE $7D
                    0220 FLAG       .DE $F0
                    0230 TEMP       .DE $F1
                    0240 ;
                    0250 LINE       .DE $122
                    0260 INC        .DE $129
                    0270 ;
                    0280            .BA $1B00
                    0290 ;
1B00- 20 D4 1B      0300 START      JSR CHANGE
1B03- 4C 6D DE      0310            JMP BASCOLD
                    0320 ;
1B06- A0 07         0330 AUTO       LDY #7          ;PULL STACK
1B08- 68            0340            PLA
1B09- 88            0350            DEY
1B0A- 10 FC         0360            BPL AUTO+2
1B0C- A9 01         0370            LDA #1
1B0E- 24 F0         0380            BIT *FLAG       ;FLAG IN STATE 2 OR 3?
1B10- 50 38         0390            BVC GETCHR      ;NO, BRANCH
                    0400 ;
1B12- D0 36         0410            BNE GETCHR      ;ENTER A% , B%
                    0420 ;
1B14- 10 08         0430            BPL LINENO      ;IF STATE 2
1B16- A5 F1         0440            LDA *TEMP       ;THEN RESTORE PTR
1B18- 85 7D         0450            STA *V.PTR
1B1A- A5 F2         0460            LDA *TEMP+1
1B1C- 85 7E         0470            STA *V.PTR+1
                    0480 ;
1B1E- A9 80         0490 LINENO     LDA #$80
1B20- 85 F0         0500            STA *FLAG       ;NOW IN STATE 4
1B22- AD 22 01      0510            LDA LINE        ;GET LINE NO.
1B25- AE 23 01      0520            LDX LINE+1
1B28- 20 8A DB      0530            JSR $DB8A       ;OUTPUT ASCII
1B2B- A2 00         0540            LDX #0
1B2D- BD 01 01      0550 GET.NO     LDA $101,X      ;GET ASCII FROM PGE 1
1B30- F0 05         0560            BEQ INCLNE
1B32- 95 1E         0570            STA *BUF,X      ;PUT IN BUFFER
1B34- E8            0580            INX
1B35- D0 F6         0590            BNE GET.NO
                    0600 ;
1B37- 18            0610 INCLNE     CLC
1B38- AD 23 01      0620            LDA LINE+1
1B3B- 6D 2A 01      0630            ADC INC+1       ;INC LINE NO.
1B3E- 8D 23 01      0640            STA LINE+1
1B41- AD 22 01      0650            LDA LINE
```

```
1B44- 6D 29 01  0660            ADC INC
1B47- 8D 22 01  0670            STA LINE
                0680 ;
                0690 ;
1B4A- 20 58 8A  0700 GETCHR     JSR INTCHR    ;INPUT A CHAR
1B4D- 29 7F     0710            AND #$7F
1B4F- C9 20     0720            CMP #$20
1B51- 90 01     0730            BCC ^Q
1B53- 60        0740            RTS  ;         RETURN IF NOT CTRL CHAR
                0750 ;
1B54- C9 11     0760 ^Q         CMP #$11      ;^Q
1B56- D0 3C     0770            BNE CR        ;NO, BRANCH
1B58- A9 41     0780            LDA #$41      ;FL IN STATE 1
1B5A- 85 F0     0790            STA *FLAG
1B5C- A5 7D     0800            LDA *V.PTR
1B5E- 85 F1     0810            STA *TEMP     ;SAVE PTR
1B60- A5 7E     0820            LDA *V.PTR+1
1B62- 85 F2     0830            STA *TEMP+1
1B64- A9 20     0840            LDA #L,LINE-2
1B66- 85 7D     0850            STA *V.PTR    ;CHANGE PTR
1B68- A9 01     0860            LDA #H,LINE-2
1B6A- 85 7E     0870            STA *V.PTR+1
                0880 ;
1B6C- A9 00     0890            LDA #0
1B6E- 8D 22 01  0900            STA LINE
1B71- 8D 29 01  0910            STA INC       ;SET DEFAULTS
1B74- A9 0A     0920            LDA #$A
1B76- 8D 23 01  0930            STA LINE+1
1B79- 8D 2A 01  0940            STA INC+1
1B7C- A9 C1     0950            LDA #$C1      ;ASCII A ,B7 SET
1B7E- 8D 20 01  0960            STA LINE-2
1B81- A9 C2     0970            LDA #$C2      ;ASCII B ,B7 SET
1B83- 8D 27 01  0980            STA INC-2
1B86- A9 80     0990            LDA #$80
1B88- 8D 21 01  1000            STA LINE-1
1B8B- 8D 28 01  1010            STA INC-1
                1020 ;
1B8E- 20 4D 83  1030            JSR CRLF
1B91- 4C 4A 1B  1040            JMP GETCHR
                1050 ;
                1060 ;
1B94- C9 0D     1070 CR         CMP #$D       ;CR
1B96- D0 10     1080            BNE ^R
1B98- 24 F0     1090            BIT *FLAG
1B9A- 30 03     1100            BMI STATE4
1B9C- 70 04     1110            BVS STATE3
1B9E- 60        1120            RTS  ;         RETURN IF STATE 0
                1130 ;
1B9F- 46 F0     1140 STATE4     LSR *FLAG     ;NOW IN STATE 3
1BA1- 60        1150            RTS
                1160 ;
1BA2- A9 C0     1170 STATE3     LDA #$C0
1BA4- 85 F0     1180            STA *FLAG     ;FL IN STATE 2
1BA6- D0 08     1190            BNE RET.CR
                1200 ;
                1210 ;
1BA8- C9 12     1220 ^R         CMP #$12      ;^R
1BAA- D0 09     1230            BNE ^C
1BAC- A9 20     1240 TURNOFF    LDA #$20
1BAE- 85 F0     1250            STA *FLAG     ;FL IN STATE 0
1BB0- A9 0D     1260 RET.CR     LDA #$D
1BB2- 4C A0 8A  1270            JMP TOUT
                1280 ;
1BB5- C9 03     1290 ^C         CMP #3        ;^C
1BB7- D0 2C     1300            BNE END
```

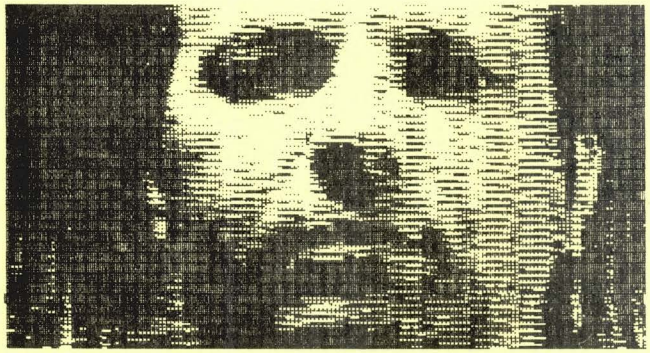


```
1BB9- A9 58     1310            LDA #L,INTCHR         ;RESTORE INVEC
1BBB- 8D 61 A6  1320            STA INVEC+1
1BBE- A9 8A     1330            LDA #H,INTCHR
1BC0- 8D 62 A6  1340            STA INVEC+2
1BC3- A9 CE     1350            LDA #L,WARM   ;SET WARM START
1BC5- 85 01     1360            STA *WARMVEC+1
1BC7- A9 1B     1370            LDA #H,WARM
1BC9- 85 02     1380            STA *WARMVEC+2
1BCB- 00        1390            BRK  ;         BREAK TO MON
1BCC- EA        1400            NOP
1BCD- EA        1410            NOP
1BCE- 20 D4 1B  1420 WARM       JSR CHANGE
1BD1- 4C 7E C2  1430            JMP BASWARM
                1440 ;
1BD4- 20 86 8B  1450 CHANGE     JSR ACCESS
1BD7- A9 06     1460            LDA #L,AUTO   ;CHANGE INVEC
1BD9- 8D 61 A6  1470            STA INVEC+1
1BDC- A9 1B     1480            LDA #H,AUTO
1BDE- 8D 62 A6  1490            STA INVEC+2
1BE1- A9 20     1500            LDA #$20
1BE3- 85 F0     1510            STA *FLAG     ;FL IN STATE 0
1BE5- 60        1520 END        RTS
                1530 ;
                1540            .EN

                0010 ;*****************************************
                0020 ;*                                       *
                0030 ;*            MARGIN PATCH               *
                0040 ;*                                       *
                0050 ;*        WRITTEN BY A.L.FOSTER          *
                0060 ;*           FEBRUARY 1982               *
                0070 ;*                                       *
                0080 ;*****************************************
                0090 ;
                0100 ;        TO INITIALISE:-
                0110 ;                        SD A600,A664
                0120 ;
                0130 TOUT       .DE $8AA0
                0140 PORTA      .DE $AC01
                0150
                0160            .BA $A600
                0170 ; .OS
                0180
A600- C9 0A     0190 PATCH      CMP #$0A      ;LINE FEED?
A602- F0 08     0200            BEQ TAB       ;YES, THEN BRANCH
                0210
A604- 2C 01 AC  0220 OUT        BIT PORTA     ;HANDSHAKE PRINTER
A607- 10 FB     0230            BPL OUT       ; VIA B7 OF PORTA
A609- 4C A0 8A  0240            JMP TOUT
                0250
A60C- 20 04 A6  0260 TAB        JSR OUT       ;OUTPUT LINE FEED
A60F- A9 20     0270            LDA #$20
A611- A2 08     0280            LDX #8
A613- 20 04 A6  0290 LOOP       JSR OUT       ;OUTPUT 8 SPACES
A616- CA        0300            DEX
A617- D0 FA     0310            BNE LOOP
A619- 60        0320            RTS
                0330
                0340 ;Note: Replace OUT with your own printer
                0350 ;      driver.  The Scope Buffer at $A600
                0360 ;      is a good place for such short patches,
                0370 ;      but many ot Jack Brown's programs also
                0380 ;      make use of it.  So does FDC-1 SYMDOS!
                0390
                0400            .EN
```

COMPUTER IMAGING

Below are portions of a recent letter from Jack Gieryic, including a computer "portrait" of him. We'll have some additional comments to make, following the extracts:



JACK BUILT PROGRAMS

JACK GIERYIC
2041 138TH AVE N W
ANDOVER, MN 55303
USA

****

May 27,1982

Dear Jean and Lux,

Now for an explanation of the picture above. That's me. Well there really is more. It was done with the Digisector DS-65 from MICROWORKS, P. O. Box 1110, Del Mar, CA 92014.

The Digisector can digitize a video picture into a 256 by 256 dot array with 64 grey levels for each dot. It requires a few seconds to do this (about 10 for the above picture) and hence is not suitable for motion.

The above picture is a 160h by 100v consecutive dot digitization. Only 4 grey levels show up in the picture resulting in a very unfair demo of the Digisector's capabilities. I plan to take the data and pull out more grey levels to get a better idea of what can be done.

I am looking into the possibility of using the Digisector for inspection of printed circuit boards. The aim is to detect missing parts.

One thing very critical to the Digisector is light level. I'm sure this is no surprise to you. The video input is NTSC composite video. Consumer video tape players and video cameras work very well. My camera only has a 240 line resolution so I cannot use the full 256 vertical resolution but can still get 256 horizontal resolution.

The Digisector interfaces very easily to the SYM. I'm using two ports on one of the VIA's on the AA connector. I removed the 6821 on the Digisector and wired from the AA connector directly to the 6821's socket. The software provided gives good examples of how to program from the SYM as you can figure what's going on and do the same thing with your own assembly language program.

If anyone out there wants to try the DS-65 then I'd be willing to send them a copy of my software and wiring diagram in order to help them get started. The DS-65 requires +5, +12 and -5 volts.

SYMcerely,

Jack Gieryic

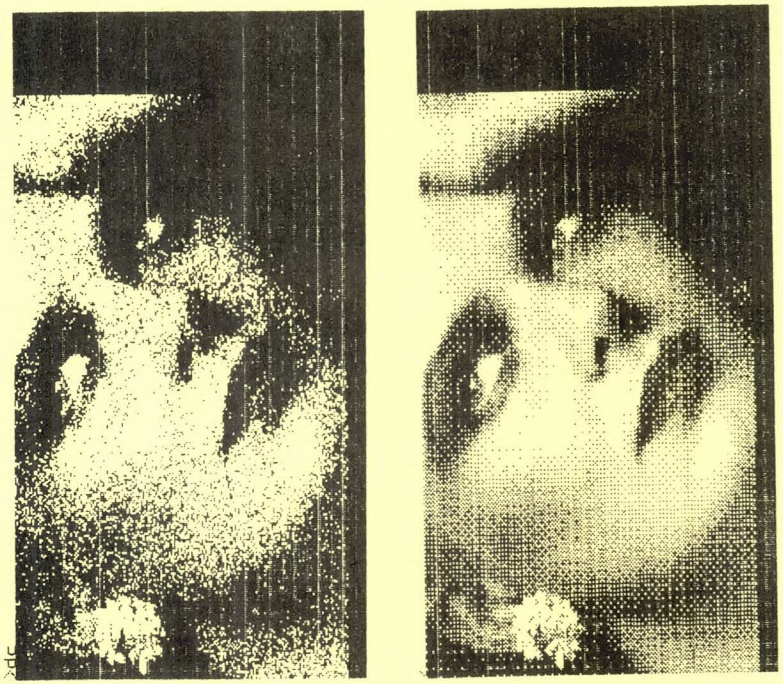
Jack Gieryic

Our area of interest, before we left industry, in 1970, to return to Academia, was in the area of what we called "Image Technology". We bought our KIM-1 in 1978, in the hopes that someday "soon" we could, somehow or other, do some experimental image processing on our very own computer, since the University's equipment could not be used for this purpose. This has not yet come to pass, but the time is coming closer!

Jack's portrait appears rather coarse and crude (not him, the image!) because of his method of emulating half-tone images. We show below two other methods of emulating half-tone images which have been transfered from Apple II to SYM. Denny Hall has a Digisector; we'll either borrow his, or get one of our own, and take advantage of Jack's offer of the software. We'll also try to figure out the algoriths used by Apple II for handling the gray scale.

We envy Jack for his being able to find the time to have so much fun with his SYM! And with his children, too! Here's another extract from his letter:

> Note 4 - I would like to buy the RCA VP3301 data terminal. Let me know if this is possible and how much. My two kids really enjoy typing on it. They are 16 months and 3 years old. Never too young!! The 3 year old can find the keys to spell her name. She'll actually be 3 on July 25th.



ASPECT RATIOS

Visible Memory	1.60:1 (320H/200V)
35mm slides	1.50:1 (36mm/24mm)
Apple II	1.46:1 (280H/192V)
TV and movies	1.33:1 (specified)

EXAMPLES OF HALF-TONE EMULATION FROM THE APPLE II SOFTWARE LIBRARY
[Done on SYM-1 with MTU Visible Memory, Epson MX-80/FT - Graftrax 80]

(The black borders on the bottom and right edges are due to the Visible Memory having 320 H x 204 V pixels vs the Apple's 280 H x 192 V pixels.)

REVERSE VIDEO ON THE KTM-2

The normal mode for KTM-2 video is bright characters on a dark background. We have a Sinclair ZX-81 around to show to non-technical people who ask about a "cheap" way to learn something about computers. The ZX-81 display is dark characters on a bright background. Which is better? We do have some opinions on the subject but will not mention them at this time, except to point out that the Sinclair generates RF (channel 3 or 4) for input through the antenna terminals of a TV receiver, and any TVI (television interference) produces an unpleasant shimmering in the bright background. This would probably not be a problem with a direct video input monitor.

The shimmering might not be so noticeable on the longer persistence green phosphors which are so popular, but we don't really like to use a green phosphor at 4800 baud, nor do we like dynamic graphics on a green phosphor. Incidentally, if you do use a green phosphor monitor (not a piece of green cellophane), you might try setting the interlace option on the KTM-2.

Anyway, if you wish to experiment with reverse video on the KTM-2, possibly with an RF modulator (but not with the KTM-2/80), with or without the interlace option, with either a green or a white phosphor, here's how to do it, according to F. H. Lassiter, of Olin Chemicals Group:

Cut the foil trace on the back of the board to pin 6 of U31 and solder a jumper from pin 5 to the foil trace you have just cut (you might consider installing a SPDT switch here). Pins 5 and 6 are the input and output of 1/6 of a 7404 hex inverter between the video output, pin 13 of U27, a 74166, and the input to 1/6 of a 74S05 open collector hex inverter, pin 11 of U41.

We have checked several recent model KTMs and could not locate a trace on the bottom of the board from pin 6 of U31. The trace from pin 6 apparently is (now?) closed and above-board, hidden underneath the soldered-in chip itself. Since the desired trace cannot easily be found by visual inspection, and we were too busy (lazy?) to use a continuity checker, we cheated, and looked at a schematic. The J3 end of jumper J3-A goes to pin 11 of U41 and the A end of the jumper goes to pin 6 of U31. So, just remove the installed jumper. It is worth noting here that where hand-installed jumper wires were used on earlier KTMs and SYMs, the current production models use the more cost effective printed circuit traces.

We would be interested in hearing reasons and reactions from those who make this reverse video modification.

A BETTER BELL FOR THE KTM-2

We have installed a bell on Jean's KTM-2/80, because she's a skilled typist and needs to know when she gets near the end of the line. We have no bells on our own KTMs because we would rather not have anyone else in the room hear the bells which accompany error messages, so we have not tried the following suggestion sent in by Steven G. Beuret of Millbourne, PA:

> One quick solution to the need for a nice bell on the KTM-2 is to cut the trace, as you've described earlier, and add a piezo beeper at the connector (the Sonalert has nice tonal quality). A spiffy improvement is had by adding a 10 microfarad capacitor and a 1.5 megohm resistor as follows;
>
> The positive end of the piezo connects to BELL.
>
> The negative end of the piezo connects to both the positive end of the capacitor, and one end of the resistor

> The free ends of the R/C pair connect to ground.
>
> This results in a pleasant beep which has a decay not unlike a real bell. The reason for this, is that the capacitor is being charged up while the beeper sounds, reducing the voltage across the beeper. The resistor slowly leaks the charge off the capacitor, such that activating BELL repeatedly, results in quieter beeps.

More news later. Thank you for all the supportive symmering.

Stephen G Beuret
3/27/82 SGB

FDC-1 SERVICE AND REPAIR

We are not prepared to troubleshoot FDC-1 kits which do not work properly upon initial assembly, to assemble kits on a production basis, or to repair boards which have failed after a period of useful service. We can only replace those components which are found (by the user, and verified by us) to be defective on receipt.

The following two SYMmers have indicated their willingness to provide such services, and we will provide them with components for warranty replacements. Others will be added to this listing as more users obtain the necessary experience with the system. Please contact them directly.

JOSEPH R. HOBART

3465 North Andes Drive, Flagstaff, AZ 8600l. Joe should be familiar to many of you through his articles in this and previous issues of SYM-PHYSIS, especially the original EPROM burner. Joe has had extensive experience with 8" FDC-1 systems.

JEFF LAVIN

Alternative Energy Products, P. O. Box 1019, Whittier, CA, 90609. We are publishing one of Jeff's many program submissions in this issue, reviewing some of his new products for the SYM in this issue, and becoming a dealer for his product line. Jeff is a long time SYMmer, but as of now, we know him only through telephone conversations, letters, and his products. He will be spending a week with us very shortly, getting briefed on troubleshooting the FDC-1.

THREE NEW SOFTWARE ITEMS

We try to publish the best of the programs which are submitted each quarter, but, obviously, there is not enough room to publish them all. A few of the submissions are so bug-ridden that they are best forgotten. Some need only minor patchup or are near-perfect. These we do publish, if they are short, useful, instructive, of general interest, etc.

We used to have time to personally try out all of the programs submitted, but not any more! If we have previously established the credibility of the author, we do take a chance, and publish them without a thorough shakedown. If the author is unknown to us, we at least try them out in a casual manner prior to publication, but cannot guarantee them to be totally bug-free.

Very long programs, those which would occupy more than eight pages, would almost "monopolize" a single issue. If they are really good and of general interest we will offer them for sale, but only after a really thorough shakedown. For others that are good, but of less general

interest, we cannot afford the time for thorough testing. These we will review, in the NOTES section, and suggest you contact the authors directly.

And, now, here is a description of the three new items:

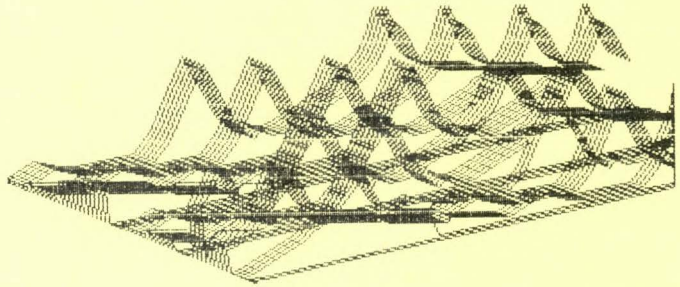
RADAR, by IAN DILWORTH

Ian Dilworth sent us an interesting program which begins thus:

```
                0010            .LS
                0020 ; EXTRACT FROM IAN DILWORTH'S "RADAR"
                0030 ; PARTIALLY EDITED AND TESTED BY LUX
                0040 ; 5 AUGUST 1982
                0050
                0060 ;*****************************************
                0070 ;RADAR PLOTTING ROUTINE (2) DEC 1981 I.J.D
                0080 ;THIS (2) ALLOWS HIDDEN LINE BLANKING...
                0090 ;INCLUDES VISIBLE MEMORY SOURCE CODE.......
                0100 ;USES VIA I/P'S TO SELECT MODE OF OPERATION
                0110 ;BIT 7 CONTROLS HIDDENLINE SELECTION (FAST/SLOW)
                0120 ;BIT 1 CONTROLS ASPECT OF PLOT, BIT 2 CONTROLS
                0130 ;WHETHER WE WANT TO PRINT DATA FROM THE FIRST
                0140 ;DATA ELEMENT OR FROM THE LAST I.E. WE CAN
                0150 ;EITHER DECREMENT MEMORY (DATA) OR INCREMENT
                0160 ;THROUGH IT......
                0170 ;PRESS BREAK KEY TO CONTINUE PLOTTING.
                0180 ;*****************************************
                0190
                0200            .BA $CA
00CA-           0210 MEM        .DS 2
                0220
                0230            .BA $B6
00B6-           0240 ADP1       .DS 2
                0250
                0260            .BA $C4
00C4-           0270 ADP2       .DS 2
                0280
                0290 IORA       .DE $A001
                0300 DDRA       .DE $A003
                0310
                0320 VMORG      .DE $2000
                0330 DATA       .DE $9000
                0340
                0350            .BA $4000
                0360
```

It calls a DATA file at $9000-$9FFF which apparently contains simulated terrain data. Ian uses manually operated switches on VIA #1 to control the processing as described above. We have two MTU DACs (for stereo music) on that VIA and didn't have time to make any mods either to the VIA or to the software, preferably the latter, so we started the program running, letting "fate" provide the "switch" signals. Open input lines ride high, we don't know what the DACs do to input lines.

Our Visible Memory is on the CODOS system, not on the FODS system on which we were testing the program, so we had to run "blind". Thus, we ran RADAR with no VIA switches and no Vis Mem, then ran the Graftrax Printer on the portion of RAM where the results were stored, to get the figure reproduced below. The output looks similar to that shown on page 12-2, but the "hidden lines" are not hidden. The program looks like it would be very exciting when run interactively, so we'll transfer it over to the CODOS/VM system, after first rewriting that section of the program involving the use of the VIA.



We do like Ian's idea of using the VIA in this manner. What he has really implemented is a whole set of Option Switches which may be interrogated by any program. Provide a removable overlay on which the switches are labeled with the functions implemented for each program that uses them, and you have greatly improved the man-machine interface. We will build such a control box after we have added more VIAs to our main system (see elsewhere).

Ian asked us to market this for him, if there seems to be an interest. So, if you have a Visible Memory, or any other type of VDU, let us know and we'll send you a copy of RADAR (RAE source code) and the DATA file which goes with it, relocated in low RAM, in case you do not have RAM at $9000. When we get around to final editing we'll also change the page zero addresses and include a zero page swap, as was done in the MX-80 Graftrax Printer earlier in this issue.

We will ask Ian where the DATA file came from, and how others may be generated. This looks like the most "fun" program we have seen for SYM in a long time!

TECO, by DALE HOLT

and

FORMATTER, by GERHARD STRUBE

We have been using SWP-1, much augmented, as our word processor, for as long as we can remember. Apparently word processing is a very popular application for the SYM, since so many word processors have been written for it.

We did a cost comparision on SYM vs Apple as word processors, and a word processing SYM cost about 2/3 as much as a word processing Apple. With the FDC-1 now available, the SYM's cost advantage is even more favorable. Be that as it may, here are two really great word processors for SYM.

TECO has been described in previous issues, and is very popular with dec's PDP systems. Holt's version is quite compatible with those written for other systems, but is tailored to the SYM cassette I/O. When rewritten to I/O to an 8" FDC-1 floppy disk system, the disks could be interchanged between SYM and these other systems. TECO is "free-standing", i. e., it does not require BASIC or RAE, but since it is supplied in RAE source code you should have RAE installed, at least until you have generated the TECO object code.

FORMATTER formats RAE edited text, and is, by far, the most sophisticated word processor we have seen for small systems. Here are some samples of its "input" and "output":

```
0000  >REM @4 and @3 control underlining for my printer.
0010  >C>^'#
0020  >S12,60
0030  >M
0040  @4F O R M A T T E R@3
0050  >>1
0060  by Gerhard Strube
0070  >>3
0080  A Survey of FORMATTER commands
0090  >>5
0100  >V
0110  $A:                 from here on, text is to be printed
0120  $B:                 from here on, justify both margins
0130  $C,CC,SEP,BL,CNT:  define special characters
0140                     CC = control char., SEP =
0150                     separator, BL = blank char.,
0160                     CNT = continuation char.
0170  $D,(nn):           define density of lines
0180  $E:                 from here on, skip text
0190  $F:                 no justification of right margin
0200  $I,(n or 8n):      indent first lines of paragraphs by
0210                     n spaces, or (8n) second and following
0220                     lines by n spaces
0230  $K,(string):       define chapter heading
0240  $L or $LP:         clear or set proportional spacing
0250  $M:                 from here on, center lines
0260  $N,(nnnn):         advance to next page (and set
0270                     page number to new value)
0280                     $N0 will inhibit page numbering
0290  $P:                 note delimiter before and after notes
0300  $Q,(nn):           if less than nn lines free, advance
0310                     to top of next page
```

F O R M A T T E R

by Gerhard Strube

A Survey of FORMATTER commands

```
$A:                 from here on, text is to be printed
$B:                 from here on, justify both margins
$C,CC,SEP,BL,CNT:   define special characters
                    CC = control char., SEP =
                    separator, BL = blank char.,
                    CNT = continuation char.
$D,(nn):            define density of lines
$E:                 from here on, skip text
$F:                 no justification of right margin
$I,(n or 8n):       indent first lines of paragraphs by
                    n spaces, or (8n) second and following
                    lines by n spaces
$K,(string):        define chapter heading
$L or $LP:          clear or set proportional spacing
$M:                 from here on, center lines
$N,(nnnn):          advance to next page (and set
                    page number to new value)
                    $N0 will inhibit page numbering
$P:                 note delimiter before and after notes
$Q,(nn):            if less than nn lines free, advance
                    to top of next page
```


SOFTWARE PRICES

Getting a major program ready for distribution might take around 40 hours, or so, to test, document, prepare the "automated" reproduction program for cassette and/or any of several disk formats, etc. If you do this, yourself, as a hobby, the time was paid for by the fun of the job. On the other hand, if you pay someone else a couple of hundred dollars to do the job, you don't even have the fun!

After guessing how many copies might be sold, and how much money will be tied up in printed manuals sitting on the shelf, and for how long, etc., etc., etc., we came up with an average price of $36.00 per major item of quality software, including shipping, with 50% of the profit going in royalties to the authors, the other 50% to pay for the costs of editing and the labor and materials cost for reproduction, invoice handling, packing, and shipping. Software distribution frequently works out to be a low profit, largely labor-of-love, deal for low sales volume items.

A BASIC PROGRAM ADAPTED BY JEFF LAVIN

We first "met" Jeff Lavin by telephone when he called to ask a few questions about getting his new SYM going. Very shortly thereafter, he sent us a bunch of CAI (Computer Aided Instruction) BASIC programs he had adapted to the SYM. We had no room to publish any of them till now.

This is one of the shorter programs he sent. We include part of a sample RUN. Sorry you can't see our answers, but our particular printer patch doesn't echo inputs, it only prints outputs. We think you will enjoy working on and extending this one. If you like it, drop us a note, and we'll print another next issue.

```
10 PRINT"       CCCC  LL       0000   VV   VV  EEEEE"
11 PRINT"      CC     LL      00  00  VV   VV  EE"
12 PRINT"      CC     LL      00  00   VV VV   EEEE"
13 PRINT"      CC     LL      00  00    VVV    EE"
14 PRINT"       CCCC  LLLLL    0000      V     EEEEE"
15 PRINT
16 PRINT"      BY ELLEN NOLD AND SALLIE CANNOM 8/73"
17 PRINT"           ADAPTED BY JEFF LAVIN 11/81"
18 PRINT
19 PRINT
20 INPUT "HI. WHAT'S YOUR NAME ? ";N$
21 PRINT
22 PRINTN$", ARE YOU A MAN OR A WOMAN? (TYPE ONE WORD) ";:INPUT S$
23 PRINT
24 PRINT"THANKS. NOW WE'RE READY TO GO."
25 PRINT
26 PRINT"LANGUAGE AND MOST ORDINARY KINDS OF THOUGHT PROCESSES"
27 PRINT"ARE BASED ON CLASSIFICATION."
28 PRINT"WHEN I SAY 'CAT',WHAT DO YOU THINK OF?
29 INPUT CT$
30 PRINT
31 PRINT"YOU THINK OF "CT$"? THAT'S INTERESTING."
32 PRINT"I THINK OF SOMETHING FOUR-LEGGED, WHISKERY, AND FURRY."
33 PRINT"CLOSE TO WHAT YOU PICTURED?
34 PRINT
35 PRINT"WHEN I SAY 'ANIMAL', WHAT DO YOU THINK OF?
36 INPUT Z$
37 PRINT
38 PRINT"I BET WE'RE MUCH FURTHER APART ON THAT. I WAS THINKING"
39 PRINT"OF SOMETHING BULBOUS, SLIMY, AND STICKY-TONGUED."
40 PRINT
41 PRINT"'ANIMAL' IS A MORE GENERAL LABEL THAN 'CAT', OR CONVERSELY,"
42 PRINT"'CAT' IS MORE SPECIFIC THAN 'ANIMAL'."
43 PRINT"CAN YOU ADD A WORD TO 'CAT' TO MAKE IT EVEN MORE SPECIFIC?"
44 INPUT "(TYPE IT IN) ";SP$
```

```
 45 PRINT
 46 PRINT"YOU COULD SAY TABBY CAT, OR MY CAT, OR JUNGLE CAT, OR ANY-"
 47 PRINT"THING THAT FURTHER DEFINES CAT."
 48 PRINT
 49 PRINT"NOW TYPE IN A MORE GENERAL TERM FOR ANIMAL.":INPUT GL$
 50 PRINT
 51 PRINT"THAT'S HARDER. WHAT OCCURS TO ME"
 52 PRINT"IS SOMETHING LIKE 'LIVING THINGS'."
 54 PRINT
 55 PRINT,GL$
 56 PRINT,"ANIMAL"
 57 PRINT,"CAT"
 58 PRINT,SP$" CAT"
 59 PRINT
 60 PRINT"YOU HAVE DONE MORE THAN JUST CLASSIFY 'BIG' TO 'LITTLE', YOU"
 61 PRINT"HAVE ORDERED A UNIVERSE ON FOUR LEVELS."
 62 PRINT"ADD TWO MORE LEVELS. YOU MAY CHANGE YOUR LABELS COMPLETELY,"
 63 PRINT"BUT MAKE SURE YOU HAVE SIX LEVELS - FROM MOST GENERAL TO"
 64 PRINT"MOST SPECIFIC. USE THE NEXT SIX LINES."
 65 PRINT
 66 INPUT "(LINE 1) ";Z$
 67 INPUT "(LINE 2) ";Z$
 68 INPUT "(LINE 3) ";Z$
 69 INPUT "(LINE 4) ";Z$
 70 INPUT "(LINE 5) ";Z$
 71 INPUT "(LINE 6) ";Z$
 72 PRINT
 73 PRINT"BY CLASSIFYING THESE NOTIONS IN YOUR HEAD, YOU HAVE AGAIN"
 74 PRINT"CREATED A PARTICULAR KIND OF UNIVERSE. YOU CAN DESTROY IT AND"
 75 PRINT"CREATE ANOTHER SIMPLY BY RE-ORDERING THOSE SAME IDEAS"
 76 PRINT"IN A DIFFERENT WAY.";:INPUT "SOUND WEIRD ? ";A$
 77 PRINT
 78 IF LEFT$(A$,2)="NO" THEN 81
 79 IF LEFT$(A$,2)="UH" THEN 81
 80 PRINT"WELL, YOU PERFORM THIS EXERCISE DAILY.":PRINT:GOTO 82
 81 PRINT"GOOD. YOU UNDERSTAND THE INFLUENCE LANGUAGE HAS ON REALITY.":P
RINT
 82 PRINT"OF COURSE, YOU'RE PART OF A UNIVERSE TOO, "N$", . . . MINE."
 83 PRINT
 84 PRINT,"GALAXY 1501"
 85 PRINT,"SOLAR SYSTEM 10"
 86 PRINT,"EARTH"
 87 PRINT,"UNITED STATES"
 88 PRINT,"CALIFORNIA"
 89 PRINT,"STANFORD"
 90 PRINT,"STANFORD "S$
 91 PRINT,N$
 92 PRINT,N$",S NOSE"
 93 PRINT
 94 INPUT "LIKE YOUR PLACE IN MY UNIVERSE ? ";Z$
 95 PRINT
 96 INPUT "WHY ? ";Z$
 97 PRINT
 98 PRINT"FAIR ENOUGH."
 99 PRINT"YOU CAN MAKE YOUR UNIVERSE TO INCLUDE OR EXCLUDE"
100 PRINT"WHATEVER YOU WANT."
101 INPUT "WANT TO DO YOUR OWN ? ";A$
102 PRINT
103 IF LEFT$(A$,2)="NO" THEN 120
104 IF LEFT$(A$,2)="UH" THEN 120
105 PRINT"FINE. USE THE NEXT EIGHT LINES."
106 PRINT
107 INPUT "(LINE 1) ";Z$
108 INPUT "(LINE 2) ";Z$
```



```
109 INPUT "(LINE 3) ";Z$
110 INPUT "(LINE 4) ";Z$
111 INPUT "(LINE 5) ";Z$
112 INPUT "(LINE 6) ";Z$
113 INPUT "(LINE 7) ";Z$
114 INPUT "(LINE 8) ";Z$
115 PRINT
120 PRINT"WELL "N$", WE'VE MADE SOME UNIVERSES."
121 PRINT"WE DO SHAPE REALITY BY OUR MENTAL GYRATIONS"
122 PRINT"AND OUR CHOICE OF WORDS."
123 PRINT"DOES THAT SEEM OVERSTATED? IF SO, YOU MIGHT"
124 PRINT"WANT TO PAY FURTHER ATTENTION TO THE INTERRELATIONSHIP"
125 PRINT"BETWEEN LANGUAGE/THOUGHT/REALITY."
126 PRINT
127 PRINT
128 PRINT"THAT'S ALL FOR NOW, "N$
129 PRINT
130 PRINT"'BYE"
131 END
OK
```

```
  CCCC  LL      OOOO   VV   VV  EEEEE
 CC     LL     OO  OO  VV   VV  EE
 CC     LL     OO  OO   VV VV   EEEE
 CC     LL     OO  OO    VVV    EE
  CCCC  LLLLL   OOOO      V     EEEEE

 BY ELLEN NOLD AND SALLIE CANNOM 8/73
      ADAPTED BY JEFF LAVIN 11/81


HI. WHAT'S YOUR NAME ?

LUX, ARE YOU A MAN OR A WOMAN? (TYPE ONE WORD) ?

THANKS. NOW WE'RE READY TO GO.

LANGUAGE AND MOST ORDINARY KINDS OF THOUGHT PROCESSES
ARE BASED ON CLASSIFICATION.
WHEN I SAY 'CAT',WHAT DO YOU THINK OF?
?

YOU THINK OF AN ANIMAL? THAT'S INTERESTING.
I THINK OF SOMETHING FOUR-LEGGED, WHISKERY, AND FURRY.
CLOSE TO WHAT YOU PICTURED?

WHEN I SAY 'ANIMAL', WHAT DO YOU THINK OF?
?

I BET WE'RE MUCH FURTHER APART ON THAT. I WAS THINKING
OF SOMETHING BULBOUS, SLIMY, AND STICKY-TONGUED.

'ANIMAL' IS A MORE GENERAL LABEL THAN 'CAT', OR CONVERSELY,
'CAT' IS MORE SPECIFIC THAN 'ANIMAL'.
CAN YOU ADD A WORD TO 'CAT' TO MAKE IT EVEN MORE SPECIFIC?
(TYPE IT IN)

YOU COULD SAY TABBY CAT, OR MY CAT, OR JUNGLE CAT, OR ANY-
THING THAT FURTHER DEFINES CAT.

NOW TYPE IN A MORE GENERAL TERM FOR ANIMAL.
?

THAT'S HARDER. WHAT OCCURS TO ME
IS SOMETHING LIKE 'LIVING THINGS'.
```

```
LIVING THING
ANIMAL
CAT
BLACK CAT

YOU HAVE DONE MORE THAN JUST CLASSIFY 'BIG' TO 'LITTLE', YOU
HAVE ORDERED A UNIVERSE ON FOUR LEVELS.
ADD TWO MORE LEVELS. YOU MAY CHANGE YOUR LABELS COMPLETELY,
BUT MAKE SURE YOU HAVE SIX LEVELS - FROM MOST GENERAL TO
MOST SPECIFIC. USE THE NEXT SIX LINES.

(LINE 1)

OK
```

TWO NEW HARDWARE PRODUCTS

THE AEP-1 32K CMOS RAM BOARD

We have long recommended the Beta 32K Dynamic RAM Board, and still continue to do so, especially for those using the HDE FODS disk controller, since the Beta DRAM Board, while requiring only a +5 V supply, generates its own +12 V and -5 V on-board, and there is enough extra capacity in these two supplies to also power the HDE controller, which requires these two voltages to be supplied, in addition to the usual +5 V.

We now are adding another RAM board for the SYM to our product line, the AEP-1 32K CMOS RAM Board. This board fits directly onto the SYM's Expansion Conector, "folded" beneath it, with a right-angled 44 contact edge connector, and its free edge fingers are an extension of the SYM's Expansion Connector. It uses the new 2Kx8 CMOS static RAMs for low power consumption, is easily bank-switched to provide essentially unlimited memory, and will also hold 2716s as well.

We have been using an early prototype at $0000-$7FFF for some time now, and are thinking of adding a second one at $8000-$FFFF in the near future, for a very much customized and far-from-standard, highly personalized, system which will be a SYM in name only, since we will be relocating a customized SUPERMON, booting up to a DOS, at $F000-$FFFF, and moving all of the I/O up to $E000-$FFFF. This will be our dream 6502 system, and this is the expansion board around which we will build it. The disk controller will, of course be the FDC-1. Since all these boards use the standard KIM-1 (SYM-1) bus, we will install the system in an MTU card cage, together with a bank-switched Visible Memory.

The new RAM board is a product of Alternative Energy Products (Jeff Lavin), and permits almost complete freedom in memory address selection, within either the lower or upper 32K of memory space. Here are some extracts from Jeff's spec sheet:

200NS LOW POWER CMOS STATIC RAM - 32K draws less than 0.6 A enabling the KTM-2 and the SYM-1 with 32K of memory to run on a single 3 A power supply. Also has greater noise immunity.

EXPANSION CONNECTOR EXTENDED - instead of worrying with other buses, the Expansion connector is available for use.

FIRST 8K ARE JUMPER SELECTABLE - this means you may keep either 4 or 8K of 2114 RAM on board, and select the unused blocks somewhere else (at $9000 and $9800 for example). All memory is addressed on 2K boundaries.

COMPATIBLE WITH 2716 EPROMS - 2716 EPROMS may be substituted for RAM at any position and will operate in the power down mode.

MAY BE BANK SWITCHED - a jumper is provided for use in bank switching boards for greater memory.

STANDARD ADDRESSING - $0000-$7FFF on 2K boundaries. May be optionally addressed at $8000-$FFFF by using an inverted A15 address line provided externally.

G-10 EPOXY/GLASS, FULL SOLDER MASK, GOLD FINGERS

FULL 1-YEAR LIMITED WARRANTY

THE AEP-2 I/O EXPANSION BOARD

Despite all of the I/O capability already built onto the SYM, we have already run out of ports! We have two DACs (Digital to Analog Converters) permanently on VIA #1, and our Epson Printer uses half of VIA #2. Whenever we wish to burn an EPROM, or demonstrate the Speak & Spell, we have to power down and change connectors between devices. It would be nicer if all three of these devices were always on-line.

We would also like to have a real time hardware clock with battery back-up, and a multiplexed ADC (Analog to Digital Converter) always on-line. We also are now thinking of adding an Option Select Unit (see the review on Dilworth's "RADAR" program).

We told Jeff that we needed four ADDITIONAL VIAs on the SYM, and a few weeks later he shipped us a prototype AEP-2. This is a 4 1/2 inch square board with sockets for five 6522s, and a 74LS154 4-line to 16-line decoder/demultiplexer chip. Remove VIA #2 from the SYM, mount it on the board, and plug the board directly into the now empty socket. The VIA functions as before, with its I/O at the AA connector.

By bringing three additional address lines from the SYM board to holes on the I/O board waiting to receive them and send them to the decoder, you can get eight VIAs into the memory space assigned to VIA #2. The board holds only five, but the necessary signals are passed out of the board at a 44 pin edge connector for further expansion.

Actually four address lines come to the expansion board, so that if you are willing to give up VIA #3's assigned functions (think how seldom you really use them!) this board will let you address 16 VIAs.

P. S.: Jeff will soon be announcing his real time hardware clock card and a communications module to be used with this I/O adaptor. All VIAs (other than the "original" one) interface to the outside world through 20 pin in-line connectors adjacent to the VIA sockets.

THE RADIO SHACK LINE PRINTER VIII

Here's a brief extract from a letter showing some of the versatility of this printer which lists in the latest catalog at $799.00:

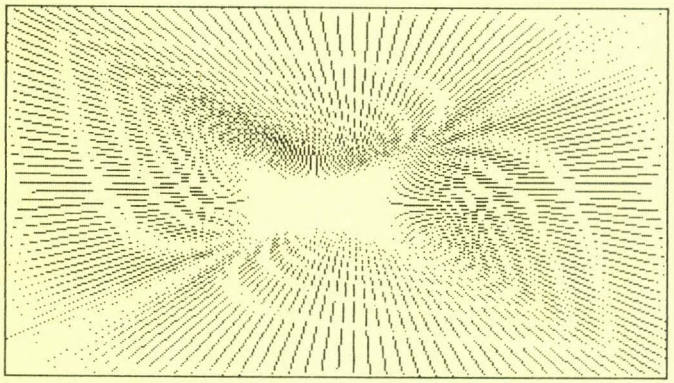
> I'm writing this letter using an editor/word processor I've written in FORTH, that takes advantage of the features of the Radio Shack Line Printer VIII. This printer features a proportional (variable pitch) character set, proportional spacing commands (move the print head 1 to 9 dots) and dot addressable graphics. It also has block grahpic characters and a European character set. I wrote the word processor in FORTH since it looked like a fairly massive task to modify SWP to use the proportional character set. In fact, it looked like I would have to re-write SWP from the top down, since the line justification algorithm would be totally different, line lengths would be specified in dots, not characters, and so on. Here are some example of what the LPVIII can do.

Various fonts -- proportional, elongated, condensed, condensed elongated
Special characters -- SYM™, ©copyright, £2.40, accents áçäöüéè, etc.
super$_{\text{scripting}}$ and sub$_{\text{scripting}}$.
Enclosed is a sample of a Visible Memory dump to the LPVIII.

That's about it for now. Good luck with Volume III.

Bill Wharrie

Bill Wharrie
272 Erb St W
Waterloo, Ontario
CANADA N2L 1W2



TWO MORE RECOMMENDED BOOKS

One of the "perks" (perquisites) of being a university professor is the scores of free books we get from the publishers to review for possible adoption in our classes. Of course we can adopt only two or three a year, and for most of my advanced courses, the art is changing so rapidly that I don't adopt a textbook at all.

Most of the books go into the bookshelves, and are donated, in batches, to collection drives for underdeveloped countries. Very few are worth lending to students as recommended reading.

During this past quarter two SYMmers had their publishers send us copies of their newly published books for review. The books are entirely different in scope and intended for different audiences. Unfortunately, there is neither sufficient time nor space here to review them in the depth they deserve, so we'll do it rather informally.

Microprocessor Systems - Interfacing and Applications
Robert J. Bibbero, P.E., and David M. Stern

This John Wiley and Sons book is not intended as a text, but could be assigned as required reading at the senior or graduate student level in seminar or independent study courses. Since its main area of concern is the increasing interdependence between computer technology and communications technology, and since it does present a good introduction to both fields, it could be read, with profit, by a computer engineer assigned to a communication project, or a communications engineer being faced with having to learn, and learn fast, computer technology.

It's the kind of book I used to look for, back when I was doing consulting work in an area that was new to me, one which would introduce me quickly to the basic concepts and technical jargon of the people I would be interfacing with.

Yes, I would recommend it to those of my students who are alert enough to recognize that they had better find out more about how communications technology is affecting their future in the computer field. We do not have a graduate engineering program here at California State University, Chico, but the Engineering Division will be introducing a new Computer Engineering program at the undergraduate level, and I will commend this book to those involved.

Microcomputer Design and Troubleshooting
Eugene M. Zumchak

This Howard W. Sams Company book is up there in a class with De Jong's book, which we think all SYM users need right next to the SYM-1 Reference Manual. Put a copy of Zumchak there, too.

While De Jong emphasizes how to apply an existing system, Zumchak helps you to find out why your existing system is giving you problems, and shows you how to build a better one.

We only had a few hours to study our review copy; Denny Hall borrowed it and won't return it until we get him another copy. Because of that, and because we feel that all SYMmers will find it useful, we're ordering a big batch for resale.

THE DVORAK KEYBOARD

We print below part of a brief note by Jim Mott, Code 3109, Naval Weapons Center, China Lake, CA 93555, on providing a special keyboard for the KTM-3. We do not print his table because the KTM-3 ROM is not the same as the KTM-2 ROM.

Two keyboards are compared in an article "Dvorak vs. Qwerty: Will Tradition Win Again?" by Shirley Boes Neill in the June 1980 issue of Phi Delta Kappan. The keyboards are also compared in "Change Comes Slowly" by Albert C. Kolb in the February 1979 issue of CTS Journal. A Dvorak All-Electric Portable Typewriter is available from the Typewriting Institute for the Handicapped, 3102 West Augusta Ave., Phoenix, Arizona 85021 (602) 939-5344.

A special manual has been prepared by Dr. Dvorak in collaboration with Ruth Ben 'Ary. The textbook can be used in a regular classroom setting or as a self teaching aid. Two KTM-3 computer terminals were available for me to try to make a Dvorak keyboard and try it. Both units have been changed and are now under evaluation. The KTM-3 is made by Synertek Systems.

The Dvorak keyboard is as follows, where the format is upper case/lower case [Editor's note - not all keys and/or symbols are shown here]:

```
!/1 "/2 #/3 $/4 %/5 &/6 '/7 (/8 )/9 @/0 */: =/- ?/
?// </, >/. P/p Y/y F/f G/g C/c R/r L/l LF CR
A/a O/o E/e U/u I/i D/d H/h T/t N/n S/s /@  /[  /]
+/; Q/q J/j K/k X/x B/b M/m W/w V/v Z/z
```

The read-only-memory at U10 was modified as follows, based on the PROM's first location being assigned a value of $0000. [Editor's note - table omitted] After writing the new PROM I rearranged the key caps to show the Dvorak keyboard. These terminals can be made available for demonstrations.

[Editor's note to all contributors: Please! Either use a fresh ribbon, or send your material on cassette!]

MISCELLANEA

LEE H. LONGSTREET, JR. sent us a copy of Sol Libes' BYTELINES column from the May 1982 BYTE (which we do get, but had not yet gotten around to reading!). We quote: "Commodore is expected to finally release its 16-bit microprocessor that will be software compatible with the 6502." . . . .

C. DAVID STRITT would like to see a series entitled "Biography of a System, or, See How My SYM Grew!", wherein users describe how they went about expanding their systems. Perhaps we could find room for one or two such articles???

LEE H. LONGSTREET, JR. has been sending us "goodies" at a rate faster than we can incorporate them into our CODOS system. Many months ago he sent us SUPERMON in EPROM, relocated at $F000, and a new CODOS master disk to go with it. We hesitated to install the new system, partly because of the time involved, but mostly because we would then have problems with software interchangeability with others. Yesterday we received from him a new CODOS disk, and a listing which begins as follows (note the 6800 cross-assembler!):

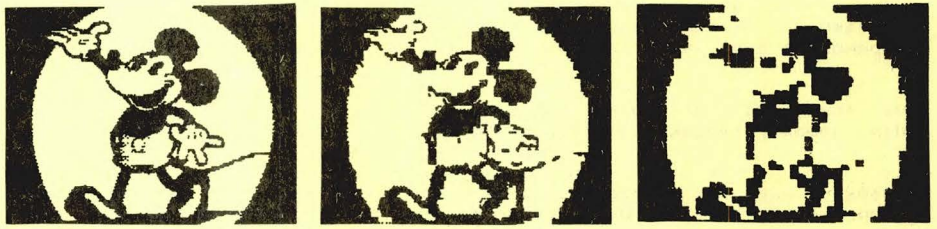
```
0002 ;*****************************************************
0004 ;**                                                 **
0006 ;**    CODOS/X-RAY, AN INTEGRATED OPERATING SYSTEM  **
0008 ;**      COMPOSED OF MTU CODOS, SYM RAE, & SATURN   **
0010 ;**      SOFTWARE'S X-RAY - OTHER USER FRIENDLY     **
0012 ;**      ENHANCEMENTS AND CODOS INTERGRATION        **
0014 ;**            BY LEE H LONGSTREET JR               **
0016 ;**                                                 **
0018 ;**    X-RAY PORTION (C) 1982 BY SATURN SOFTWARE-   **
0020 ;**    *MUST BE PURCHASED FROM SATURN SOFTWARE*     **
0022 ;**                                                 **
0024 ;**    SYM / RAE MAY BE OBTAINED IN SEVERAL FORMS:  **
0026 ;**    SYM-RAE IN ROM FORM, SYNERTEK SYSTEMS CORP.  **
0028 ;**        OCCUPIES B000-BFFF & E000-EFFF           **
0030 ;**    ASM/TED 6502 AND/OR CROSSASM/TED 6800 BY     **
0032 ;**        EASTERN HOUSE SOFTWARE, ON CASSETTE      **
0034 ;**        OCCUPIES APPX. 2000-4100                 **
0036 ;**                                                 **
0038 ;**    CODOS IS AN ADVANCED DISK OPERATING SYSTEM   **
0040 ;**     AND HDWE. PKG, MICRO TECHNOLOGY UNLIMITED   **
0042 ;**                                                 **
0044 ;**                                                 **
0046 ;**                DISK FILE XRAY1                  **
0048 ;**             LAST REVISION - 07/07/82            **
0050 ;**                                                 **
0052 ;*****************************************************
0054 ;
0056 ;NOTE: TO USE ON SYM WITH STD. MONITOR LOCATION, CHANGE
0058 ;MOST SIGNIFICANT BYTE (PAGE) IN SUPERMON ROM ROUTINES
0060 ;LISTING TO 8XXX, IE. F035 = 8035 . . . . IN THIS CASE,
0062 ;RAE CODE CAN BE USED AS IS, OTHERWISE USE THE RAEXXXX.J
0064 ;CODOS JOB FILE WHICH WILL AUTOMATICALLY MAKE THE CHANGES
0066 ;NECESSARY FOR RAE TO RUN WITH SYM MONITOR RELOCATED
0068 ;TO F000 - FFFF.
0070 ;
0072 ;
0074 ;ASSEMBLY DATA
0076 ;
0078             .CE
0080             .ES
0082             .BA $9000
0084             .MC $C000
0086             .OS
0088 VERSION     .DE 0          ;0=RAE6502, 1=RAE6800
```


PHIL KOHL, too, has been keeping us busy with cassettes containing revised versions of Jack Gieryic's EPROM Burner which will handle 2732s and 2764s. Now all we need do is rebuild the hardware. Actually, we will start from scratch with a totally new board.

We are sitting here going through the large file of material we had planned to include, or at least mention briefly in this issue, but we see that we are now on page 40 (why do printers say -30- to mean the end, when for us it is -40-; inflation, maybe?), and we must leave within the hour to catch a flight to Los Angeles (UCLA). So, the material will just have to wait. Apologies to those whose material is being delayed. We'll start on Issue #13, the JUL/AUG/SEP issue, in mid-September, and you should be getting your copy before the end of October, if all goes as planned.

We will be be at UCLA for half a week, then fly back for a lens implant on Friday the 13th. DICK ALBERS and JEFF LAVIN will be visiting shortly thereafter, and we'll be working together on a multi-DOS system using many of Jeff's new products. Fall semester classes will start at CSU, Chico on 30 August.

From the Apple II Software Library - "Here's lookin' at you . . ."



"So long, folks, . . . "

" -40- "