-Punch along this edge for insertion into a binder-

OUR NEW POST OFFICE BOX IS 319 PLEASE RENEW FOR EIGHTY-TWO

FOREIGN CHECKS MUST BE IN U.S. DOLLARS AND MUST BEAR MAGNETIC INK COMPUTER READABLE (MICR) CODING.

SYM-PHYSIS

THE SYM-1 USERS' GROUP NEWSLETTER

VOLUME II, NUMBER 4 (ISSUE NO. 10) - WINTER 1981 (OCT/NOV/DEC)

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

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

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 (1980), are available, as a package, for $12.00, US/Canada, and $16.00, First Class/Airmail, elsewhere.

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

THE MYSTERIES OF BAS-1 REVEALED! (OR AT LEAST SOME OF THEM)

The following questions asked by James Blackshear have been asked by many readers, and if not specifically asked, must at least have occurred to many others:

"I would like to know more about BAS-1. What is on page zero, how can we use it, and where are the useful routines in BASIC? And, is there a source listing (commented or not) available, and if so, how can we get ahold of it?"

To answer the second question first, the source code for BAS-1 is proprietary to Microsoft and its licensee Synertek Systems Corporation (the first time we visited Jared Larsen at SSC, we observed a listing of the source code on a desktop, and started to leaf through it, but it was removed and placed out of our reach!). Publishing a complete disassembly of BAS-1 would permit readers to by-pass purchasing the ROMs, thus reducing potential sales by the copyright owner. Thus the answer is NO!!!

On the other hand, an incomplete listing would still require the reader to purchase BAS-1 in order to obtain a working BASIC. So, to answer Jim's first question, we are publishing a partial disassembly of BAS-1, and, in the article "MEAN14 FOR THE SYM", information on entry points for most of the floating point arithmetic subroutines. This information should provide a useful starting point for owners of BAS-1 who wish to proceed with their own "reverse-engineering" of BAS-1.

HOW TO POWER-UP INTO A RUNNING BASIC PROGRAM

Another frequently asked question is "How can we arrange to have the SYM power-on-reset either to BASIC, or directly into a working BASIC program (i.e., in the RUN mode)? The answer to this extremely important question is the subject of our lead article:

```
                        0010 ; Example of a power-on-reset program to start up-
                        0020 ; and-running in a BASIC program, i.e., a "turnkey"
                        0030 ; system.  If you wish to "protect" your program,the
                        0040 ; easiest way is to POKE 42580,128 to lock-out inputs
                        0050 ; until they are called for, and write a "guarded"
                        0060 ; input routine to prevent return to the direct command
                        0070 ; mode on a <cr> reply to an input request.  The BREAK
                        0080 ; key will also cause a return to the direct command
                        0090 ; mode, but if the keyboard has been locked out by the
                        0100 ; POKE only RESET can restart the program.
                        0110
                        0120 ; IT MIGHT BE A GOOD IDEA FOR TROUBLE SHOOTING TO BUILD
                        0130 ; IN A "SECRET KEY" TO PERMIT EXIT TO SUPERMON.  ONE WAY
                        0140 ; IS TO ALLOW A SPECIAL INPUT SEQUENCE TO CAUSE A JUMP
                        0150 ; TO A USR(USRENT,0).
                        0160
                        0170 ; This program may be tested in RAM, at any address,
                        0180 ; from SUPERMON, by .G TURNKEY, to simulate reset.
                        0190
                        0200 ;        SUPERMON ADDRESSES
                        0210
                        0220 INCHR      .DE $8A1B    Input character
                        0230 OUTCHR     .DE $8A47    Output character
                        0240 TOUT       .DE $8AA0    Terminal character out
                        0250 ACCESS     .DE $8B86    Unprotect SYSRAM
                        0260 VECSW      .DE $8BB7    Default I/O terminal vectors
                        0270 DFTBLK     .DE $8FA0    Default table
                        0280 RIN        .DE $887E    RAM input
                        0290
                        0300 ;        SYSTEM RAM ADDRESSES
                        0310
                        0320 SYSRAM     .DE $A620    RAM above SCPBUF
                        0330 SDBYT      .DE $A651    Baud rate constant
                        0340 SCRA       .DE $A63A    Scratch
                        0350 TOUTFL     .DE $A654    Terminal I/O flags
                        0360
                        0370 ;        SYSTEM RAM VECTORS
                        0380
                        0390 INVEC      .DE $A660
                        0400 OUTVEC     .DE $A663
                        0410
                        0420 ;        SYSTEM I/O ADDRESSES
                        0430
                        0440 PCR1       .DE $A00C
                        0450
                        0460 ;AND NOW, HERE WE GO ! ! ! !
                        0470
                        0480            .BA $9800    Or wherever, in any EPROM
                        0490            .OS
                        0500
9800- A2 FF             0510 TURNKEY    LDX #$FF Initialize stack pointer
9802- 9A                0520            TXS
9803- A9 CC             0530            LDA #$CC
9805- 8D 0C A0          0540  .         STA PCR1     Disable POR, tape off, etc.
9808- A9 04             0550            LDA #$04     Zero flags, and disable IRQ
980A- 48                0560            PHA
980B- 28                0570            PLP
```

```
                    0580
980C- 20 86 8B      0590            JSR ACCESS
980F- A2 5F         0600            LDX #$5F      Init SYSRAM using defaults
                    0610
                    0620 ;As an alternative, your choices for default values could
                    0630 ;be stored at the "top" of this EPROM, and moved to SYSRAM
                    0640 ;from there
                    0650
9811- BD A0 8F      0660 XFER       LDA DFTBLK,X
9814- 9D 20 A6      0670            STA SYSRAM,X
9817- CA            0680            DEX
9818- 10 F7         0690            BPL XFER
                    0700
                    0710 ;Avoid necessity for log-on
                    0720
981A- A9 01         0730            LDA #$01
981C- 8D 51 A6      0740            STA SDBYT     Set baud rate to 4800
981F- 20 B7 8B      0750            JSR VECSW     Set default terminal vectors
                    0760
                    0770 ;Prepare for "EXECUTE"-type command
                    0780
9822- AD 62 A6      0790            LDA INVEC+2
9825- 8D 3B A6      0800            STA SCRA+1
9828- AD 61 A6      0810            LDA INVEC+1
982B- 8D 3A A6      0820            STA SCRA
                    0830
982E- A9 7E         0840            LDA #L,RIN
9830- 8D 61 A6      0850            STA INVEC+1
9833- A9 88         0860            LDA #H,RIN
9835- 8D 62 A6      0870            STA INVEC+2
                    0880
9838- A9 54         0890            LDA #L,EXEC
983A- 8D FA 00      0900            STA $FA
983D- A9 98         0910            LDA #H,EXEC
983F- 8D FB 00      0920            STA $FB
                    0930
                    0940 ;Change any other vectors or defaults desired here
                    0950 ;Clear screen, turn off I/O during reset
                    0960 ;Omit these lines if you want to observe the process
                    0970
9842- A9 1B         0980            LDA #27 ESC
9844- 20 47 8A      0990            JSR OUTCHR
9847- A9 45         1000            LDA #'E Clear screen for KTM-2
9849- 20 47 8A      1010            JSR OUTCHR
                    1020
984C- A9 00         1030            LDA #$00
984E- 8D 54 A6      1040            STA TOUTFL
                    1050
9851- 4C 00 C0      1060            JMP $C000
                    1070 EXEC       ;These values are for 8K systems, no reserve
9854- 38 31 39      1080            .BY '8192' $0D        ;MEMORY SIZE?
9857- 32 0D
9859- 38 30 0D      1090            .BY '80' $0D    ;WIDTH?
985C- 58 3D 55      1100            .BY 'X=USR(&"8035",0)' $0D    ;USRENT
985F- 53 52 28
9862- 26 22 38
9865- 30 33 35
9868- 22 2C 30
986B- 29 0D
                    1110
                    1120 ;Insert here any SUPERMON commands needed to
                    1130 ;modify BAS-1 default values, e.g., TRIGPATCH,
                    1140 ;Lower Case Patch, etc.
                    1150
```



```
986D- 47 30 0D      1160            .BY 'G0' $0D    ;WARM START
9870- 58 3D 55      1170            .BY 'X=USR(&"8B86",0)' $0D       ;ACCESS
9873- 53 52 28
9876- 26 22 38
9879- 42 3B 36
987C- 22 2C 30
987F- 29 0D
9881- 31 30 30      1180            .BY '1000?"PROGRAM STARTS HERE"' $0D
9884- 30 3F 22
9887- 50 52 4F
988A- 47 52 41
988D- 4D 20 53
9890- 54 41 52
9893- 54 53 20
9896- 48 45 52
9899- 45 22 0D
989C- 35 30 30      1190            .BY '5000?:?:?' $0D
989F- 30 3F 3A
98A2- 3F 3A 3F
98A5- 0D
98A6- 39 30 30      1200            .BY '9000?"AND ENDS HERE . . . . ."' $0D
98A9- 30 3F 22
98AC- 41 4E 44
98AF- 20 45 4E
98B2- 44 53 20
98B5- 48 45 52
98B8- 45 20 2E
98BB- 20 2E 20
98BE- 2E 20 2E
98C1- 20 2E 22
98C4- 0D
98C5- 30 50 4F      1210            .BY '0POKE 42580,144' $0D ;TOUTFL - CRT I/O
98C8- 4B 45 20
98CB- 34 32 35
98CE- 38 30 2C
98D1- 31 34 34
98D4- 0D
                    1220
98D5- 52 55 4E      1230            .BY 'RUN' $0D
98D8- 0D
98D9- 30 0D         1240            .BY '0' $0D  ;Delete Line No. 0 (the POKE)
98DB- 4C 49 53      1250            .BY 'LIST' $0D
98DE- 54 0D
98E0- 00            1260            .BY $00
                    1270
                    1280 ;The next two bytes MUST be in the POWER-ON-RESET socket!
                    1290 ;(and in byte positions three and four from the top, also)
                    1300
                    1310            .BA $9FFC
                    1320
9FFC- 00 98         1330            .SE TURNKEY
                    1340
                    1350            .EN
                    1360
                    1370 ; An alternate method of getting started is to move
                    1380 ; BASIC's page zero and the BASIC program into your
                    1390 ; EPROM, and down load it from SUPERMON, rather than
                    1400 ; from BASIC, as in the example below:
                    1410
                    1420 ; REPLACE JMP $C000 WITH JMP $8000 IN LINE 1060
                    1430
                    1440            .BA $9854
                    1450
```

```
9854- 42 30 2C  1460 EXEC       .BY 'B0,9000,90E7' $0D       ;PAGE ZERO
9857- 39 30 30
985A- 30 2C 39
985D- 30 45 37
9860- 0D
9861- 42 32 30  1470            .BY 'B200,9200,922F' $0D     ;BASIC PROGRAM
9864- 30 2C 39
9867- 32 30 30
986A- 2C 39 32
986D- 32 46 0D
9870- 47 30 0D  1480            .BY 'G0' $0D                 ;BASIC WARM START
9873- 58 3D 55  1490            .BY 'X=USR(&"8886",0)' $0D   ;ACCESS
9876- 53 52 28
9879- 26 22 38
987C- 42 38 36
987F- 22 2C 30
9882- 29 0D
9884- 52 55 4E  1500            .BY 'RUN' $0D $00            ;TURN OFF RIN
9887- 0D 00
                1510
                1520 ; The TOUTFL POKE should be in your BASIC program.
                1530 ; The ACCESS may also be in your BASIC program. It must
                1540 ; be executed before the first INPUT, so that subroutine
                1550 ; RIN can be turned off by the $00 following RUN.
                1560
                1570            .EN
```

MACHINE LANGUAGE FLOATING POINT ARITHMETIC

Perhaps the easiest way of learning to design Assemblers, Interpreters, Arithmetic Packages, Data Management Systems, Disk Operating Systems, Compilers, etc., is to study the "works of the masters" and learn by example. With this thought in mind, we have been using Hissink's DISARAE, and an even more powerful disassembler, to "reverse engineer" BAS-1, SYM/FODS, RAE-1, CODOS, and every other piece of 6502 object code from which we felt we could learn something new.

We have been asked if it were possible to write a floating point package for the SYM. The answer is yes, but rather than starting from scratch, we prefer to "research" to see how others have done the job, and, to paraphrase Newton, "stand on the shoulders of giants". A direct copy, or even a "paraphrase", without permission or acknowledgement, is plagiarism. An enhancement, a major modification, a synthesis of the works of others, a conversion to another system, published or marketed in such a way as not to injure potential sales of the original product is neither illegal, immoral, nor fattening! We will now answer the floating point question in our usual roundabout way:

Apple II, the one with Integer BASIC, but not the II + with the Applesoft (Microsoft!) BASIC, contains an interpreter known as SWEET16. Interpreters of this class accept programs written in a set of "pseudo-operation" codes designed to make programming easier for a specialized class of problems. The SWEET16 interpreter makes a 6502 system "behave" as if it were a 16 bit (integer) processor.

While Apple's Integer BASIC handles 16 bit integer arithmetic, it does so very slowly, compared to machine language (ML) equivalents. SWEET16 interprets at a rate closer to ML, but is easier to program, since the pseudo-ops are essentially "macro"-instructions. SWEET16 programs may be assembled by hand, or by a SWEET16 assembler. In fact RAE-1 has a "flag" at $0132 so that a SWEET16 (or any other assembler, for that matter) may be "patched" to RAE-1, making use of all of RAE's text editing capabilities. The LISA Assembler for the Apple also includes an assembler for SWEET16 mnemonics.

MEAN14, by R. M. Mottola, works with Applesoft BASIC to provide five byte floating point arithmetic at least ten times faster than the BASIC. We have been working off-and-on to rewrite MEAN14 for the SYM-1/BAS-1 system, but have not yet finished the task. For those who have asked, we print below our INCOMPLETE conversion, leaving the completion as an "exercise for the student", and refer the reader to Mr. Mottola's articles, cited below, for the necessary details. For "extra credit", why not add the trig functions as well? A good follow-up "exercise", of value to the entire 6502 community would be a MEAN14 Assembler. For SYM-1 users this should be "called" from RAE-1 by setting the flag at $0132.

Note that MEM (memory location) is passed to the BASIC subroutines through the A,Y register pair. We have not yet found a good entry point for converting INPUT ASCII to floating point but the subroutine at $DB9A converts the floating point number in FPAC1 to ASCII at **$0100-$010F**, with a terminator byte of $00, making it very simple to print out the results. Try running a program such as:

```
10 A = 123.456 : REM TRY VARIOUS VALUES
20 PRINT A
30 X = USR(&"8035",0)
RUN
```

From SUPERMON, .V 1000,101F <cr> to see the ASCII representation of A, and reenter BASIC with .G 0 <cr>. Try using values for A which will force the "E" representation of the value. Incidentally, these (in addition to the stack, of course) 16 bytes are the only useage of page one by BASIC. The remainder of page one is yours to use.

The combination of MEAN14 with the existing BAS-1 ROM will provide a very easy-to-use and easy-to-implement floating point capability operating at near ML speeds.

REFERENCES:

Mottola, R. M., "Applesoft Floating Point Routines", MICRO, August, 1980, pp. 27:53 - 27:55.

Mottola, R. M., "MEAN 14: A Pseudo-Machine Floating Point Processor for the Apple II", MICRO, September, 1980, pp. 28:67 - 28:71.

Wozniak, Stephen, "SWEET16: The 6502 Dream Machine", BYTE, November, 1977, pp. 150 - 159.

```
0010 ;    >>>>>>> MEAN14 FOR THE SYM-1 <<<<<<<
0020
0030 ; SEE CITED ARTICLE(S) BY MOTTOLA FOR DETAILS
0040 ; NOT YET COMPLETE OR OPERABLE!!!!
0050 ; NO WARRANTY EXPRESS OR IMPLIED
0060 ; GOOD LUCK IN FINISHING THE JOB!
0070
0080            .BA $9000
0090
0100 ; Not yet sure where these should be placed,
0110 ; but they are not used by BAS-1
0120
0130 TMPL       .DE $E9
0140 TMPH       .DE $EA
0150 MPCL       .DE $4C
0160 MPCH       .DE $4D
0170
0180 ; START BYTES OF THE TWO FLOATING ACCUMULATORS
0190
0200 FPAC1      .DE $B1
0210 FPAC2      .DE $B9
0220
```

```
                    0230 ; ENTRY POINTS FOR SELECTED BAS-1 SUBROUTINES
                    0240
                    0250 ; MEM  = (A,Y) = (ADL,ADH)
                    0260 ; MEM2 = (X,Y) = (ADL,ADH)
                    0270
                    0280 LDAC1      .DE $D958    ; MEM          > FPAC1
                    0290 LDAC2      .DE $D842    ; MEM          > FPAC2
                    0300 FPADD      .DE $D61D    ; MEM + FPAC1  > FPAC1
                    0310 FPSUB      .DE $D60D    ; MEM - FPAC1  > FPAC1
                    0320 FPMUL      .DE $D7DE    ; MEM * FPAC1  > FPAC1
                    0330 FPDV1      .DE $D8C5    ; MEM / FPAC1  > FPAC1
                    0340 FPDV2      .DE $D8C5    ; FPAC2 / MEM  > FPAC1
                    0350 FPRND      .DE $D9D1    ; 0.5 + FPAC1  > FPAC1
                    0360 FPSTR      .DE $D98D    ;       FPAC1  > MEM2; [MEM2=(X,Y)]
                    0370 TR2>1      .DE $D9B2    ;       FPAC2  > FPAC1
                    0380 TR1>2      .DE $D9C5    ;       FPAC1  > FPAC2
                    0390 FPSGN      .DE $D9EF    ;  SGN(FPAC1)  > FPAC1
                    0400 FPABS      .DE $DA0E    ;  ABS(FPAC1)  > FPAC1
                    0410 FPINT      .DE $DA82    ;  INT(FPAC1)  > FPAC1
                    0420 FPSQR      .DE $DCF3    ;  SQR(FPAC1)  > FPAC1
                    0430 FPLOG      .DE $D7A0    ;  LOG(FPAC1)  > FPAC1
                    0440 FPEXP      .DE $DCFA    ; FPAC2 ^ MEM  > FPAC1
                    0450 INT>FP     .DE $D14C    ;     [(Y,A)]  > FPAC1
                    0460 FP>INT     .DE $CF79    ;  INT(FPAC1)  > ($B4,$B5) (HI, LO)
                    0470 ASCII.OUT  .DE $DB9A    ; ASC$(FPAC1)  > ($100-$11F)
                    0480
9000- 68            0490 MEAN14     PLA
9001- 85 4C         0500            STA $MPCL
9003- 68            0510            PLA
9004- 85 4D         0520            STA $MPCH
9006- 20 5F 90      0530            JSR INC.PC
9009- 20 0F 90      0540 GO.ON      JSR GET.DO
900C- 4C 09 90      0550            JMP GO.ON
                    0560 ;
900F- A0 00         0570 GET.DO     LDY #$00
9011- B1 4C         0580            LDA (MPCL),Y
9013- AA            0590            TAX
9014- 29 3F         0600            AND #$3F
9016- 0A            0610            ASL A
9017- A8            0620            TAY
9018- C8            0630            INY
9019- B9 A0 90      0640            LDA SUBTBL,Y
901C- 48            0650            PHA
901D- 88            0660            DEY
901E- B9 A0 90      0670            LDA SUBTBL,Y
9021- 48            0680            PHA
9022- 20 5F 90      0690            JSR INC.PC
9025- 8A            0700 FIND.MODE  TXA
9026- 29 C0         0710            AND #$C0
9028- F0 34         0720            BEQ IMPLIED
902A- 10 20         0730            BPL IMEDIATE
902C- 29 40         0740            AND #$40
902E- D0 13         0750            BNE ABSOLUTE
9030- B1 4C         0760 INDIRECT   LDA (MPCL),Y
9032- 85 E9         0770            STA $TMPL
9034- C8            0780            INY
9035- B1 4C         0790            LDA (MPCL),Y
9037- 85 EA         0800            STA $TMPH
9039- 88            0810            DEY
903A- B1 E9         0820            LDA (TMPL),Y
903C- 48            0830            PHA
903D- C8            0840            INY
903E- B1 E9         0850            LDA (TMPL),Y
9040- 48            0860            PHA
9041- 90 13         0870            BCC SET2COUNT       ;ALWAYS
```



```
                    0880 ;
9043- B1 4C         0890 ABSOLUTE   LDA (MPCL),Y
9045- 48            0900            PHA
9046- C8            0910            INY
9047- B1 4C         0920            LDA (MPCL),Y
9049- 48            0930            PHA
904A- 90 0A         0940            BCC SET2COUNT       ;ALWAYS
                    0950 ;
904C- A5 4C         0960 IMEDIATE   LDA $MPCL
904E- 48            0970            PHA
904F- A5 4D         0980            LDA $MPCH
9051- 48            0990            PHA
9052- A9 05         1000 SET5COUNT  LDA #$05
9054- 90 02         1010            BCC COUNT     ;ALWAYS
                    1020 ;
9056- A9 02         1030 SET2COUNT  LDA #$02
9058- 20 61 90      1040 COUNT      JSR COUNT.PC
905B- 68            1050            PLA
905C- A8            1060            TAY
905D- 68            1070            PLA
905E- 60            1080 IMPLIED    RTS
                    1090 ;
905F- A9 01         1100 INC.PC     LDA #$01
9061- 18            1110 COUNT.PC   CLC
9062- 65 4C         1120            ADC $MPCL
9064- 85 4C         1130            STA $MPCL
9066- 90 03         1140            BCC NO.CARRY
9068- E6 4D         1150            INC $MPCH
906A- 18            1160            CLC
906B- A0 00         1170 NO.CARRY   LDY #$00
906D- 60            1180            RTS
                    1190 ;
906E- AA            1200 STORE      TAX
906F- 4C 8D D9      1210            JMP FPBTR
                    1220 ;
9072- 85 E9         1230 CONV1      STA $TMPL
9074- 84 EA         1240            STY $TMPH
9076- A0 00         1250            LDY #$00
9078- B1 E9         1260            LDA (TMPL),Y
907A- 48            1270            PHA
907B- C8            1280            INY
907C- B1 E9         1290 CONV1A     LDA (TMPL),Y
907E- A8            1300            TAY
907F- 68            1310            PLA
9080- 20 4C D1      1320            JSR INT>FP
9083- A5 B6         1330            LDA $FPAC1+5
9085- 10 07         1340            BPL NO.OP
9087- A9 C4         1350            LDA #L,VALUE1
9089- A0 90         1360            LDY #H,VALUE1
908B- 20 1D D6      1370            JSR FPADD
908E- 60            1380 NO.OP      RTS
                    1390 ;
908F- 85 E9         1400 CONV2      STA $TMPL
9091- 84 EA         1410            STY $TMPH
9093- A0 01         1420            LDY #$01
9095- B1 E9         1430            LDA (TMPL),Y
9097- 48            1440            PHA
9098- 88            1450            DEY
9099- F0 E1         1460            BEQ CONV1A    ;ALWAYS!
                    1470 ;
909B- 68            1480 RETRN      PLA
909C- 68            1490            PLA
909D- 6C 4C 00      1500            JMP (MPCL)
                    1510 ;
```

(continued to page 14)

```
0001 ;    >>>PARTIAL "SOURCE CODE" FOR BAS-1<<<
0002 ;
0003 ;COURTESY A. J. HISSINK, STEVE COLE, JACK BROWN.
0004 ;AND OH, SO MANY OTHERS!
0005 ;
0006              .BA $C000
0007              .MC *C000
0008              .OS
0009 ;
0010 JUMP0        .DE $00 JUMP TO WARM START
0011 JUMP3        .DE $03 JUMP TO MESSAGE SUB
0012 VECT6        .DE $06 VECTOR FLOATING TO INTEGER SUB
0013 VECT8        .DE $08 VECTOR INTEGER TO FLOATING SUB
0014 USRJMP       .DE $0A JUMP TO USER ROUTINE
0015 SEARCH       .DE $0D SEARCH CHARACTER
0016 SCAN         .DE $0E SCAN CHARACTER
0017 IBUPTR       .DE $0F INPUT BUFFER POINTER
0018 TYPE         .DE $11 FF=STRING 00=NUMERIC
0019 TYPEF        .DE $12 FF=INTEGER 00=FLOATINT PT
0020 FLG13        .DE $13 FLAG- DATA SCAN; LIST QUOTE
0021 FLG14        .DE $14 SUBSCRIPT FLG; FNX FLG
0022 TYPFLG       .DE $15 00-INPUT;40-GET;98-READ
0023 CEFLG        .DE $16
0024 INPFLG       .DE $17 INPUT FLG (SUPPRESS OUTPUT)
0025 NULLS        .DE $18
0026 PR.POS       .DE $19 POSITION ON PRINT LINE
0027 TWIDTH       .DE $1A TERMINAL WIDTH
0028 COLLIM       .DE $1B INPUT COLUMN LIMIT
0029 LINNUM       .DE $1C LINE NUMBER
0030 BUFF         .DE $1E OLD BUFFER POINTER
0031 DISSTK       .DE $66 POINTER TO DESCRIPTOR STACK
0032 ADRPTR1      .DE $72 ADDRESS PTR FOR TEXT INSERTION
0033 ADRPTR2      .DE $74 ADDRESS PTR FOR TEXT INSERTION
0034 PROD         .DE $76 PRODUCT AREA FOR MULTIPLICATION
0035 PSAD         .DE $7B START SOURCE TEXT
0036 VSAD         .DE $7D START SIMPLE VARIABLES
0037 ASAD         .DE $7F START ARRAY VARIABLES
0038 VEAD         .DE $81 VARIABLES END ADDRSSS
0039 SSAD         .DE $83 STRINGS START ADDRESS
0040 SEAD         .DE $85 STRINGS END ADDRESS
0041 HIMEM        .DE $87 LAST AVAILAVLE RAM LOC
0042 CURLIN       .DE $89 CURRENT BASIC LINE NUMBER
0043 LSTLIN       .DE $8B PREVIOUS LINE NUMBER
0044 CONTPTR      .DE $8D POINTER: STATEMENT FOR CONT
0045 DATALIN      .DE $8F CURRENT DATA LINE NUMBER
0046 DATADR       .DE $91 CURRENT DATA ADDR
0047 INPVEC       .DE $93 SOURCE OF INPUT
0048 CURVARNAM    .DE $95 CURRENT VARIABLE NAME
0049 CURVARADR    .DE $97 CURRENT VARIABLE ADDRESS
0050 VADPTR       .DE $99 VARIABLE AD POINTER
0051 STXPTR       .DE $9B SAVE TEXT POINTER
0052 FUNDIS       .DE $9E POINTER TO FUNCTION DESCRIPTION
0053 WORKPTR      .DE $A0 WORK POINTER
0054 GARBAG       .DE $A3
0055 FUNJMP       .DE $A4 JUMP VECT FOR FUNCTIONS
0056 WORKAREA     .DE $A7 MISC NUMERIC WORK AREA
0057 VECT1        .DE $A8
0058 VECT2        .DE $AA
0059 BLKPTR1      .DE $AD BLOCK TRANSFER POINTER
0060 BLKPTR2      .DE $AF BLOCK TRANSFER POINTER
0061 AC1.E        .DE $B1 ACCUM#1: EXPONENT
0062 AC1.M        .DE $B2 ACCUM#1: MANTISSA
0063 AC1.S        .DE $B6 ACCUM#1: SIGN
0064 SEREVAL      .DE $B7 SERIES EVALUATION CONSTANT PTR
```



```
0065 AC1.O        .DE $B8 ACCUM#1: OVERFLOW (HI-ORDER)
0066 AC2.E        .DE $B9 ACCUM#2: EXPONENT
0067 AC2.M        .DE $BA ACCUM#2: MANTISSA
0068 AC2.S        .DE $BE ACCUM#2: SIGN
0069 SGNCMP       .DE $BF SIGN COMPARISON AC1 vs AC2
0070 AC1.R        .DE $C0 ACCUM#1: ROUNDING (LO-ORDER)
0071 SERPTR       .DE $C1 SERIES POINTER
0072 TRIGP        .DE $C3 TRIG JUMP
0073 TAPESV       .DE $C6 TAPE SAVE JUMP
0074 TAPELD       .DE $C9 TAPE LOAD JUMP
0075 CHRGET       .DE $CC GET NEXT CHARACTER
0076 CHRGOT       .DE $D2 REGET LAST CHARACTER
0077 TXTPTR       .DE $D3 TEXT POINTER
0078 RANDOM       .DE $E4 RANDOM NUMBER SEED
0079 ;
0080 ;**      MEMORY ALLOCATIONS
0081 ;
0082 PGONE        .DE $100 POINTER TO PAGE ONE
0083 PROGRAM      .DE $200 START OF SOURCE PROGRAM
0084 ;
0085 ;**  MONITOR  ROUTINES  USED
0086 ;
0087 ASCNIB       .DE $8275 CONVERT ASCII TO LO 4 BITS
0088 INSTAT       .DE $8386 SEE IF KEY DOWN
0089 INCHR        .DE $8A1B INPUT CHAR
0090 OUTCHR       .DE $8A47 MON CHAR OUT
0091 ACCESS       .DE $8B86 UNWRITE PROTECT SYS RAM
0092 LOADT        .DE $8C78 MONITOR TAPE IN ROUTINE
0093 DUMPT        .DE $8E87 MONITOR TAPE OUT ROUTINE
0094 ;
0095 ;**     SYSTEM  RAM  LOCATIONS
0096 ;
0097 TEAD         .DE $A64A TAPE END ADDRESS
0098 TSAD         .DE $A64C TAPE START ADDRESS
0099 ID           .DE $A64E TAPE HEX ID
0100 ;
C000- 4C 6D DE  0101 BASCOLD     JMP INIT.ZPAGE
0102 ;
0103 ADRKEYWD    .SI END-1     ;action addr for primary keywords
C003- 23 C6     0104              .SI FOR-1
C005- 34 C5     0105              .SI NEXT-1
C007- D7 CA     0106              .SI DATA-1
C009- 81 C7     0107              .SI INPUT-1
C00B- B8 C9     0108              .SI DIM-1
C00D- 54 CE     0109              .SI READ-1
C00F- E4 C9     0110              .SI LET-1
C011- 2E C8     0111              .SI GOTO-1
C013- 2E C7     0112              .SI RUN-1
C015- 06 C7     0113              .SI IF-1
C017- B1 C7     0114              .SI RESTORE-1
C019- 09 C6     0115              .SI GOSUB-1
C01B- 11 C7     0116              .SI RETURN-1
C01D- 5B C7     0117              .SI REM-1
C01F- C4 C7     0118              .SI STOP-1
C021- 21 C6     0119              .SI ON-1
C023- D4 C7     0120              .SI NULL-1
C025- 64 C6     0121              .SI WAIT-1
C027- E2 D5     0122              .SI LOAD-1
C029- B6 C6     0123              .SI SAVE-1
C02B- 75 C6     0124              .SI DEF-1
C02D- 6B D1     0125              .SI POKE-1
C02F- D9 D5     0126              .SI PRINT-1
C031- BD C8     0127              .SI CONT-1
C033- 4A C6     0128              .SI LIST-1
C035- AC C4
```

```
C037- 71 C4      0129              .SI CLEAR-1
C039- 01 D0      0130              .SI FCERROR-1          ;GET not implemented
C03B- 55 C4      0131              .SI NEW-1
                 0132 ;
C03D- EF D9      0133 ADRFUN       .SI SGN        ;action addr for functions
C03F- 82 DA      0134              .SI INT
C041- 0E DA      0135              .SI ABS
C043- 0A 00      0136              .SI USRJMP     ;actually starts at $CDBD!
C045- 38 D1      0137              .SI FRE
C047- 59 D1      0138              .SI POS
C049- F3 DC      0139              .SI SQR
C04B- 14 DE      0140              .SI RND
C04D- A0 D7      0141              .SI LOG
C04F- 6F DD      0142              .SI EXP
C051- C3 00      0143              .SI TRIGP
C053- C3 00      0144              .SI TRIGP
C055- C3 00      0145              .SI TRIGP
C057- C3 00      0146              .SI TRIGP
C059- C3 D5      0147              .SI PEEK
C05B- 31 D5      0148              .SI LEN
C05D- 1E D2      0149              .SI STR
C05F- 62 D5      0150              .SI VAL
C061- 40 D5      0151              .SI ASC
C063- A1 D4      0152              .SI CHR
C065- B5 D4      0153              .SI LEFT
C067- E1 D4      0154              .SI RIGHT
C069- EC D4      0155              .SI MID
                 0156 ;
C06B- 79         0157 ADROPER      .BY $79        ;hierarchy and action addresses
C06C- 1F D6      0158              .SI ADD-1      ;for operators
C06E- 79         0159              .BY $79
C06F- 08 D6      0160              .SI MINUS-1
C071- 7B         0161              .BY $7B
C072- E0 D7      0162              .SI MULT-1
C074- 7B         0163              .BY $7B
C075- C7 D8      0164              .SI DIVIDE-1
C077- 7F         0165              .BY $7F
C078- FC DC      0166              .SI EXPON-1
C07A- 50         0167              .BY $50
C07B- 27 CD      0168              .SI AND-1
C07D- 46         0169              .BY $46
C07E- 24 CD      0170              .SI OR-1
C080- 7D         0171              .BY $7D
C081- 35 DD      0172              .SI GREATER-1
C083- 5A         0173              .BY $5A
C084- 73 CC      0174              .SI EQUAL-1
C086- 64         0175              .BY $64
C087- 54 CD      0176              .SI LESS-1
                 0177 ;
C089- 45 4E C4   0178 KEYWD        .BY 'EN' $C4           ;END
C08C- 46 4F D2   0179              .BY 'FO' $D2           ;FOR
C08F- 4E 45 58   0180              .BY 'NEX' $D4          ;NEXT
C092- D4
C093- 44 41 54   0181              .BY 'DAT' $C1          ;DATA
C096- C1
C097- 49 4E 50   0182              .BY 'INPU' $D4         ;INPUT
C09A- 55 D4
C09C- 44 49 CD   0183              .BY 'DI' $CD           ;DIM
C09F- 52 45 41   0184              .BY 'REA' $C4          ;READ
C0A2- C4
C0A3- 4C 45 D4   0185              .BY 'LE' $D4           ;LET
C0A6- 47 4F 54   0186              .BY 'GOT' $CF          ;GOTO
C0A9- CF
C0AA- 52 55 CE   0187              .BY 'RU' $CE           ;RUN
```



```
C0AD- 49 C6      0188              .BY 'I' $C6            ;IF
C0AF- 52 45 53   0189              .BY 'RESTOR' $C5       ;RESTORE
C0B2- 54 4F 52
C0B5- C5
C0B6- 47 4F 53   0190              .BY 'GOSU' $C2         ;GOSUB
C0B9- 55 C2
C0BB- 52 45 54   0191              .BY 'RETUR' $CE        ;RETURN
C0BE- 55 52 CE
C0C1- 52 45 CD   0192              .BY 'RE' $CD           ;REM
C0C4- 53 54 4F   0193              .BY 'STO' $D0          ;STOP
C0C7- D0
C0C8- 4F CE      0194              .BY 'O' $CE            ;ON
C0CA- 4E 55 4C   0195              .BY 'NUL' $CC          ;NULL
C0CD- CC
C0CE- 57 41 49   0196              .BY 'WAI' $D4          ;WAIT
C0D1- D4
C0D2- 4C 4F 41   0197              .BY 'LOA' $C4          ;LOAD
C0D5- C4
C0D6- 53 41 56   0198              .BY 'SAV' $C5          ;SAVE
C0D9- C5
C0DA- 44 45 C6   0199              .BY 'DE' $C6           ;DEF
C0DD- 50 4F 4B   0200              .BY 'POK' $C5          ;POKE
C0E0- C5
C0E1- 50 52 49   0201              .BY 'PRIN' $D4         ;PRINT
C0E4- 4E D4
C0E6- 43 4F 4E   0202              .BY 'CON' $D4          ;CONT
C0E9- D4
C0EA- 4C 49 53   0203              .BY 'LIS' $D4          ;LIST
C0ED- D4
C0EE- 43 4C 45   0204              .BY 'CLEA' $D2         ;CLEAR
C0F1- 41 D2
C0F3- 47 45 D4   0205              .BY 'GE' $D4           ;GET
C0F6- 4E 45 D7   0206              .BY 'NE' $D7           ;NEW
C0F9- 54 41 42   0207              .BY 'TAB' $A8          ;TAB(
C0FC- A8
C0FD- 54 CF      0208              .BY 'T' $CF            ;TO
C0FF- 46 CE      0209              .BY 'F' $CE            ;FN
C101- 53 50 43   0210              .BY 'SPC' $A8          ;SPC(
C104- A8
C105- 54 48 45   0211              .BY 'THE' $CE          ;THEN
C108- CE
C109- 4E 4F D4   0212              .BY 'NO' $D4           ;NOT
C10C- 53 54 45   0213              .BY 'STE' $D0          ;STEP
C10F- D0
C110- AB         0214              .BY $AB        ;+
C111- AD         0215              .BY $AD        ;-
C112- AA         0216              .BY $AA        ;*
C113- AF         0217              .BY $AF        ;/
C114- DE         0218              .BY $DE        ;^
C115- 41 4E C4   0219              .BY 'AN' $C4   ;AND
C118- 4F D2      0220              .BY 'O' $D2    ;OR
C11A- BE         0221              .BY $BE        ;>
C11B- BD         0222              .BY $BD        ;=
C11C- BC         0223              .BY $BC        ;<
C11D- 53 47 CE   0224              .BY 'SG' $CE           ;SGN
C120- 49 4E D4   0225              .BY 'IN' $D4           ;INT
C123- 41 42 D3   0226              .BY 'AB' $D3           ;ABS
C126- 55 53 D2   0227              .BY 'US' $D2           ;USR
C129- 46 52 C5   0228              .BY 'FR' $C5           ;FRE
C12C- 50 4F D3   0229              .BY 'PO' $D3           ;POS
C12F- 53 51 D2   0230              .BY 'SQ' $D2           ;SQR
C132- 52 4E C4   0231              .BY 'RN' $C4           ;RND
C135- 4C 4F C7   0232              .BY 'LO' $C7           ;LOG
C138- 45 58 D0   0233              .BY 'EX' $D0           ;EXP
```

```
C13B- 43 4F D3  0234              .BY 'CO' $D3          ;COS
C13E- 53 49 CE  0235              .BY 'SI' $CE          ;SIN
C141- 54 41 CE  0236              .BY 'TA' $CE          ;TAN
C144- 41 54 CE  0237              .BY 'AT' $CE          ;ATN
C147- 50 45 45  0238              .BY 'PEE' $CB         ;PEEK
C14A- CB
C14B- 4C 45 CE  0239              .BY 'LE' $CE          ;LEN
C14E- 53 54 52  0240              .BY 'STR' $A4         ;STR$
C151- A4
C152- 56 41 CC  0241              .BY 'VA' $CC          ;VAL
C155- 41 53 C3  0242              .BY 'AS' $C3          ;ASC
C158- 43 48 52  0243              .BY 'CHR' $A4         ;CHR$
C15B- A4
C15C- 4C 45 46  0244              .BY 'LEFT' $A4        ;LEFT$
C15F- 54 A4
C161- 52 49 47  0245              .BY 'RIGHT' $A4       ;RIGHT$
C164- 48 54 A4
C167- 4D 49 44  0246              .BY 'MID' $A4         ;MID$
C16A- A4
C16B- 47 CF     0247              .BY 'G' $CF           ;GO
C16D- 00        0248              .BY $00
                0249 ;
                0250 ;        ERROR CODES
                0251 ;
C16E- 4E C6     0252 BASMSG       .BY $4E $C6  ;NF
C170- 53 CE     0253              .BY $53 $CE  ;SN
C172- 52 C7     0254              .BY $52 $C7  ;RG
C174- 4F C4     0255              .BY $4F $C4  ;OD
C176- 46 C3     0256              .BY $46 $C3  ;FC
C178- 4F D6     0257              .BY $4F $D6  ;OV
C17A- 4F CD     0258              .BY $4F $CD  ;OM
C17C- 55 D3     0259              .BY $55 $D3  ;US
C17E- 42 D3     0260              .BY $42 $D3  ;BS
C180- 44 C4     0261              .BY $44 $C4  ;DD
C182- 2F B0     0262              .BY $2F $B0  ;/0
C184- 49 C4     0263              .BY $49 $C4  ;ID
C186- 54 CD     0264              .BY $54 $CD  ;TM
C188- 4C D3     0265              .BY $4C $D3  ;LS
C18A- 53 D4     0266              .BY $53 $D4  ;ST
C18C- 43 CE     0267              .BY $43 $CE  ;CN
C18E- 55 C6     0268              .BY $55 $C6  ;UF
                0269 ;
C190- 20 45 52  0270 ERRMSG       .BY ' ERROR' $00
C193- 52 4F 52
C196- 00
C197- 20 49 4E  0271 INMSG        .BY ' IN ' $00
C19A- 20 00
C19C- 0D 0A 4F  0272 OKMSG        .BY $0D $0A 'OK' $0D $0A $00
C19F- 4B 0D 0A
C1A2- 00
C1A3- 0D 0A 42  0273 BRKMSG       .BY $0D $0A 'BREAK' $00
C1A6- 52 45 41
C1A9- 4B 00
                0274 ;
                0275              .EN
```

(SYM-PASCAL - continued from page 24)

P. S. - We took a little "time-off" to follow the suggestion in the SYM-Pascal manual about which sections of the object code should be disassembled; examination of the thus-obtained source code indicates that "customization" to match your system requirements should be a very simple task.

>>>PLEASE CONTACT SATURN SOFTWARE LIMITED FOR ANY FURTHER INFORMATION<<<

(continued from page 8)

```
90A0- 57 D9     1520 SUBTBL       .SI LDAC1-1
90A2- 6D 90     1530              .SI STORE-1
90A4- C4 D9     1540              .SI TR1>2-1
90A6- B1 D9     1550              .SI TR2>1-1
90A8- 1C D6     1560              .SI FPADD-1
90AA- 0C D6     1570              .SI FPSUB-1
90AC- DD D7     1580              .SI FPMUL-1
90AE- C4 D8     1590              .SI FPDV1-1
90B0- 8D 90     1600              .SI NO.OP-1
90B2- F2 DC     1610              .SI FPSQR-1
90B4- F9 DC     1620              .SI FPEXP-1
90B6- 81 DA     1630              .SI FPINT-1
90B8- 0D DA     1640              .SI FPABS-1
90BA- EE D9     1650              .SI FPSGN-1
90BC- 9F D7     1660              .SI FPLOG-1
90BE- 71 90     1670              .SI CONV1-1
90C0- 8E 90     1680              .SI CONV2-1
90C2- 9A 90     1690              .SI RETRN-1
                1700
90C4- 90 00 00  1710 VALUE1       .BY $90 $00 $00
90C7- 00 00     1720              .BY $00 $00
                1730
                1740              .EN
```

THE BASIC USER FUNCTION

The BAS-1 USR function has more parameter passing capability than the USR function as implemented in most earlier Microsoft BASICs. Also, there is no need to "POKE" the subroutine location, as in most other Microsoft BASICs. If you wish to have the values in the A,Y register pair returned correctly, however, your machine language program must not end with the usual RTS, but instead with either a JMP $D14C or a JMP ($0008). The BASIC subroutine at $D14C converts an integer supplied to it in the A,Y register pair into a floating point number, and returns to the proper reentry point in BASIC with its own RTS. This floating point value is then assigned to the variable whose "name" was set equal to the USR function.

The purpose of this and the following brief notes is to point out alternate ways of passing parameters. First of all, the "final" parameter is also passed in the two page zero locations at $B4,$B5, as well as in A,Y. This may prove useful if the parameters are to be used several times. Remember that although BASIC stores its integers in hex format, the bytes are "reversed(?!)" from the usual 6502 format, with high byte first. The A,Y pair is passed as USR(LOCATION, 256$A+Y) on the first call, or as USR(256$A+Y) on succeeding calls to (the same) LOCATION.

Note that while Y may lie in the range 0 <= Y <= 255, you cannot pass values of A greater than 127. This is no real restriction, however, since the acceptable range for A is -128 <= A <= 127. To pass $FF, set A = -1, to pass $80, set A = -128, etc.

ON INTEGER VARIABLES

You may also pass parameters as BASIC integer values, P1%, P2%, ... If the first line in your BASIC program assigns dummy values to these variables, then after RUN is "executed" the values of these parameters are at known locations, from which your subroutines may pick them up. RUN the program:

```
1P1%=10:P2%=100:P3%=1000
```

Exit BASIC with X=USR(&"8035",0) <cr>. Reentry from SUPERMON is with .G

Ø <cr>. You will find the correct value pairs ØØ ØA, ØØ 64, Ø3 E8, in the memory at the end of your BASIC program, with the pairs separated by seven bytes. The location of the first pair (high byte first!) is at the address in locations $7D,$7E (low byte first!) + 2. This technique of passing paraneters as BASIC integers is very useful in graphics, music, and control applications. Your BASIC program can change the values of these parameters as desired, but the locations will remain fixed. AND NOW HERE COMES THE GOOD PART!!!!! You are not restricted to returning only a single pair of bytes through (A,Y). Return as many pairs as you wish, through P1%, P2%, ...

Just recently, one of our readers phoned to ask about the "%" sign he had seen following variable names, and it was suggested that he reread the BAS-1 Reference Manual section on integer variables!!!! He pointed out that the Reference Manual had absolutely no mention of integer variables; much to our surprise this is true. Those of us familiar with BASIC before getting BAS-1 must just have assumed BAS-1 would have integer variables and did not realize they were not documented in the Manual!

Incidentally, the use of integer variables instead of floating point variables saves no space, and very little time, if any. The major advantage of using integer variables is the space saving in arrays (matrices), e.g., use A%(I,J,K), instead of A(I,J,K), if the entries are integers. A second advantage of integer variables is the close relation to hexadecimal form as illustrated above. Has any reader studied Microsoft BASIC enough to let us know under what conditions the use of integer variables will reduce computation time?

A PATCH FOR DISARAE

Here is an improvement for Hissink's DISARAE, extracted from a letter from Dick Albers:

As originally published in SYM-PHYSIS, DISARAE wastes one, or, for certain instructions, two bytes per line, by inserting $A0 ($A0 = $80 + $20 = "negative space") as an end-of-line indicator. That can be a lot of bytes for a large program, such as RAE-1, or BAS-1. These new lines will correct the "problem" and save the bytes:

```
3900
3910 END.LINE    PHP
3912             STY *YSAVE
3914 DECPTR      LDY *PGM.PTR
3916             DEY
3918             STY *PGM.PTR
3920             CPY #$FF
3922             BNE MRKEND
3924             DEC *PGM.PTR+1
3926 MRKEND      LDY #$00
3928             LDA (PGM.PTR),Y
3930             CMP #$20
3932             BEQ DECPTR
3934             ORA #$80
3936             BMI A.STORE+3
3940
```

FLOATING POINT AND HUEY II

So that you may better understand how Microsoft BASIC does its arithmetic, including series evaluations, we publish below a "derived" source code for the BAS-1 TRIG PATCH; this is well worth studying.

We would also like to remind you of HUEY II, a free-standing floating-point program, available in Manual form only, and only from the 6502 Program Exchange. If you really want to understand floating point arithmetic, including conversions to-and-fro, sophisticated computational algorithms, and "macro-string" programming, the HUEY II Manual is a must!

While we are not officially dealers for HUEY II, we do have permission to distribute a RAE-1 Format Source Code Cassette for HUEY II to those who order the HUEY II Manual and Cassette as a package.


```
2000- A9 A0
2002- A0 0E
2004- 85 C4
2006- 84 C5
2008- 60
```

```
0010 ;SOURCE CODE FOR (MODIFIED) BAS-1 TRIG PATCH
0020
0030 ;The "original" source code appearing here was
0040 ;recreated by Tom Gettys from a study of the
0050 ;published object code.  It was slightly modi-
0060 ;fied and compacted by Jack Brown to generate
0070 ;a shorter object code than the published one.
0080
0090 ;Wherever you choose to relocate this patch,
0100 ;remember that it is the address of TRIGST
0110 ;that is to be installed in TRIGP, not the
0120 ;.BA address, i.e., include the following
0130 ;link in your initialization:
0140
0150             .BA $2000     ;OR WHEREVER
0160
0170 LINK        LDA #L,TRIGST
0180             LDY #H,TRIGST
0190             STA *TRIGP+1
0200             STY *TRIGP+2
0210             RTS
0220
0230 ;Jack Brown has given us permission to reprint
0240 ;here the following (slightly modified) extract
0250 ;from his copyrighted SYM-BASIC extensions.
0260
0270 ;**    COPYRIGHT 1980 BY J. W. BROWN     **
0280 ;**         ALL RIGHTS RESERVED          **
0290
0300 ;**       PAGE  ZERO  DEFINITIONS        **
0310
0320 CEFLG       .DE $16
0330 FUNDIS      .DE $9E POINTER TO FUNCTION DESCRIPTION
0340 FACTC       .DE $A7
0350 FACTA       .DE $B1 FLOATING PT REGISTER A
0360 FACTB       .DE $B9 FLOATING PT REGISTER B
0370 ASCFAC      .DE $C1 ASCII REP OF FACTA
0380 TRIGP       .DE $C3 TRIG JUMP
0390
0400 ;**      INTERNAL  BASIC  ROUTINES       **
0410
0420 KEYWD       .DE $C088 START KEYWORDS
0430 OPENUP      .DE $C1D9 OPEN UP SPACE FOR LINE
0440 INIT        .DE $C229 INITIALIZE
0450 ERRMES      .DE $C258 ERROR MES SUB
0460 BAWARM      .DE $C27E BASIC WARM START
0470 FIXLNK      .DE $C323 FIX LINE LINKS JUMP
0480 LNKFIX      .DE $C32C FIX LINE LINKS SUB
0490 FINDLN      .DE $C427 LOCATE LINE IN TEXT
0500 SCRATCH     .DE $C458
0510 RSTCLR      .DE $C46D RESET PTRS AND CLR
0520 PROCES      .DE $C5D1 EXECUTE LINE OF TEXT
0530 BADSAV      .DE $C6DD SAVE ERROR EXIT
0540 BADLOD      .DE $C6EF LOAD ERROR EXIT
0550 ENGOTO      .DE $C732 END GOTO
0560 LINGET      .DE $C7F5 ASCII TO BINARY
0570 BASRET      .DE $C8F6 HERE WITH FULL BUFF
0580 MESSUB      .DE $C954 SEND MESSAGE HI Y LO A
0590 ENDGET      .DE $CA2E RETURN FROM GET
0600 NUMERIC     .DE $CA46 NUMERIC GET
0610 EVARG       .DE $CB43 EVAL ARG AS 2 BYTE HEX
0620 CRBRAK      .DE $CCA5 CHK FOR RT BRACKET
0630 CLBRAK      .DE $CCA8 CHK FOR LFT BRACKET
0640 CCOMMA      .DE $CCAB CHK FOR COMMA
0650 GVARAD      .DE $CE5F GET VARIABLE ADDRESS
```

```
                        0660 FIXFLT     .DE $CF79 FIXED TO FLOAT
                        0670 FCERROR    .DE $D002 FC ERROR MESSAGE
                        0680 FLTFIX     .DE $D14C FLOAT TO FIX
                        0690 CHKDIR     .DE $D15F CHECK FOR DIRECT COMMAND
                        0700 EVARGX     .DE $D553 EVAL AS 1 BYTE
                        0710 INTFLT     .DE $D5AD FLOAT TO INTEGER
                        0720 ADDHALF    .DE $D5FF .5+FACTA>FACTA
                        0730 SUBMF      .DE $D606 MEM-FACTA > FACTA
                        0740 SUBAF      .DE $D609 FACTB-FACTA >FACTA
                        0750 ADDMF      .DE $D61D MEM+FACTA > FACTA
                        0760 DIVAM      .DE $D8BD FACTB/MEM > FACTA
                        0770 DIVMF      .DE $D8C5 MEM/FACTA > FACTA
                        0780 MEMFAC     .DE $D958 MEM TO FACTA
                        0790 FLOAT2     .DE $D980
                        0800 FLOAT3     .DE $D98A
                        0810 FLOAT4     .DE $D9C2
                        0820 FLOATC     .DE $D9FF BINARY TO FLOAT
                        0830 INTFAC     .DE $DA82 INT(FACTA) > FACTA
                        0840 FOUT       .DE $DB9A MAKE ASCII
                        0850 CHGSGN     .DE $DD36 -FACTA >FACTA
                        0860 FLOAT1     .DE $DDC2
                        0870 CHRCOD     .DE $DE50 CHRGET/GOT CODE
                        0880
                        0890 CONSTANT   .DE $D772 81 00 00 00 00
                        0900
                        0910
                        0920 ;**      TRIG PATCH SOURCE CODE        **
                        0930
                        0940            .BA $0E00     ;OR WHEREVER
                        0950
0E00- 0B                0960 CONST1     .BY $0B       ;12 CONSTANTS FOR TRIG
0E01- 76 B3 83          0970            .BY $76 $B3 $83 $BD $D3
0E04- BD D3
0E06- 79 1E F4          0980            .BY $79 $1E $F4 $A6 $F5
0E09- A6 F5
0E0B- 7B 83 FC          0990            .BY $7B $83 $FC $B0 $10
0E0E- B0 10
0E10- 7C 0C 1F          1000            .BY $7C $0C $1F $67 $CA
0E13- 67 CA
0E15- 7C DE 53          1010            .BY $7C $DE $53 $CB $C1
0E18- CB C1
0E1A- 7D 14 64          1020            .BY $7D $14 $64 $70 $4C
0E1D- 70 4C
0E1F- 7D B7 EA          1030            .BY $7D $B7 $EA $51 $7A
0E22- 51 7A
0E24- 7D 63 30          1040            .BY $7D $63 $30 $88 $7E
0E27- 88 7E
0E29- 7E 92 44          1050            .BY $7E $92 $44 $99 $3A
0E2C- 99 3A
0E2E- 7E 4C CC          1060            .BY $7E $4C $CC $91 $C7
0E31- 91 C7
0E33- 7F AA AA          1070            .BY $7F $AA $AA $AA $13
0E36- AA 13
0E38- 81 00 00          1080            .BY $81 $00 $00 $00 $00
0E3B- 00 00
                        1090
0E3D- A5 B6             1100 ATAN       LDA $FACTA+5 GET SIGN
0E3F- 48                1110            PHA SAVE ON STACK
0E40- 10 03             1120            BPL ATAN1
0E42- 20 36 DD          1130            JSR CHGSGN -FACTA > FACTA
0E45- A5 B1             1140 ATAN1      LDA $FACTA GET EXPONENT
0E47- 48                1150            PHA SAVE ON STACK
0E48- C9 81             1160            CMP #$81
0E4A- 90 07             1170            BCC ATAN2
0E4C- A9 72             1180            LDA #L,CONSTANT
```



```
0E4E- A0 D7             1190            LDY #H,CONSTANT
0E50- 20 C5 D8          1200            JSR DIVMF 1/FACTA > FACTA
0E53- A9 00             1210 ATAN2      LDA #L,CONST1
0E55- A0 0E             1220            LDY #H,CONST1
0E57- 20 C2 DD          1230            JSR FLOAT1
0E5A- 68                1240            PLA
0E5B- C9 81             1250            CMP #$81
0E5D- 90 07             1260            BCC ATAN3
0E5F- A9 6D             1270            LDA #L,CONST2
0E61- A0 0E             1280            LDY #H,CONST2
0E63- 20 06 D6          1290            JSR SUBMF MEM-FACTA > FACTA
0E66- 68                1300 ATAN3      PLA GET SIGN BACK
0E67- 10 03             1310            BPL ATAN4
0E69- 4C 36 DD          1320            JMP CHGSGN
0E6C- 60                1330 ATAN4      RTS
                        1340 ;
0E6D- 81 49 0F          1350 CONST2     .BY $81 $49 $0F $DA $A2
0E70- DA A2
0E72- 7F 00 00          1360 CONST3     .BY $7F $00 $00 $00 $00
0E75- 00 00
0E77- 05                1370 CONST4     .BY $05       ;FIVE CONSTANTS FOLLOW
0E78- 84 E6 1A          1380            .BY $84 $E6 $1A $2D $1B
0E7B- 2D 1B
0E7D- 86 28 07          1390            .BY $86 $28 $07 $FB $F8
0E80- FB F8
0E82- 87 99 68          1400            .BY $87 $99 $68 $89 $01
0E85- 89 01
0E87- 87 23 35          1410            .BY $87 $23 $35 $DF $E1
0E8A- DF E1
0E8C- 86 A5 5D          1420            .BY $86 $A5 $5D $E7 $28
0E8F- E7 28
0E91- 83 49 0F          1430 CONST5     .BY $83 $49 $0F $DA $A2
0E94- DA A2
0E96- A1 54 46          1440            .BY $A1 $54 $46 $8F $13
0E99- 8F 13
0E9B- 8F 52 43          1450            .BY $8F $52 $43 $89 $CD
0E9E- 89 CD
                        1460
0EA0- C0 72             1470 TRIGST     CPY #$72 SINE FUNCTION?
0EA2- F0 39             1480            BEQ SINE YES-
0EA4- 90 30             1490            BCC COSINE NO- IT'S COSINE
0EA6- C0 76             1500            CPY #$76 MAYBE- IT'S ATAN FUNCTION?
0EA8- F0 93             1510            BEQ ATAN YEP-
                        1520
0EAA- 20 80 D9          1530 TAN        JSR FLOAT2 NO- MUST BE TANGENT
0EAD- A9 00             1540            LDA #0
0EAF- 85 16             1550            STA $CEFLG COMPARE EVAL FLAG
0EB1- 20 DD 0E          1560            JSR SINE
0EB4- A2 9E             1570            LDX #L,FUNDIS
0EB6- A0 00             1580            LDY #H,FUNDIS
0EB8- 20 8A D9          1590            JSR FLOAT3
0EBB- A9 A7             1600            LDA #L,FACTC
0EBD- A0 00             1610            LDY #H,FACTC
0EBF- 20 58 D9          1620            JSR MEMFAC FACTC > FACTA
0EC2- A2 00             1630            LDX #0
0EC4- 85 B6             1640            STA $FACTA+5 CLEAR SIGN
0EC6- A5 16             1650            LDA $CEFLG
0EC8- 20 D2 0E          1660            JSR TANB
0ECB- A9 9E             1670            LDA #L,FUNDIS
0ECD- A0 00             1680            LDY #H,FUNDIS
0ECF- 4C C5 D8          1690            JMP DIVMF     ;MEM/FACTA > FACTA
0ED2- 48                1700 TANB       PHA
0ED3- 4C 0F 0F          1710            JMP SINEA
                        1720 ;
0ED6- A9 6D             1730 COSINE     LDA #L,CONST2
```

```
0ED8- A0 0E     1740          LDY #H,CONST2
0EDA- 20 1D D6  1750          JSR ADDMF MEM+FACTA > FACTA
0EDD- 20 C2 D9  1760 SINE     JSR FLOAT4
0EE0- A9 91     1770          LDA #L,CONST5
0EE2- A0 0E     1780          LDY #H,CONST5
0EE4- A6 BE     1790          LDX $FACTB+5 GET SIGN
0EE6- 20 BD D8  1800          JSR DIVAM FACTB/MEM > FACTA
0EE9- 20 C2 D9  1810          JSR FLOAT4
0EEC- 20 82 DA  1820          JSR INTFAC INT(FACTA) > FACTA
0EEF- A0 00     1830          LDY #0
0EF1- 85 BF     1840          STA $FACTB+6
0EF3- 20 09 D6  1850          JSR SUBAF FACTB-FACTA > FACTA
0EF6- A9 72     1860          LDA #L,CONST3
0EF8- A0 0E     1870          LDY #H,CONST3
0EFA- 20 06 D6  1880          JSR SUBMF MEM-FACTA > FACTA
0EFD- A5 B6     1890          LDA $FACTA+5 GET SIGN
0EFF- 48        1900          PHA SAVE ON STACK
0F00- 10 0D     1910          BPL SINEA
0F02- 20 FF D5  1920          JSR ADDHALF .5+FACTA > FACTA
0F05- A5 B6     1930          LDA $FACTA+5 GET SIGN
0F07- 30 09     1940          BMI SINEB
0F09- A5 16     1950          LDA $CEFLG
0F0B- 49 FF     1960          EOR #$FF
0F0D- 85 16     1970          STA $CEFLG
0F0F- 20 36 DD  1980 SINEA    JSR CHGSGN -FACTA > FACTA
0F12- A9 72     1990 SINEB    LDA #L,CONST3
0F14- A0 0E     2000          LDY #H,CONST3
0F16- 20 1D D6  2010          JSR ADDMF MEM+FACTA > FACTA
0F19- 68        2020          PLA GET SIGN BACK
0F1A- 10 03     2030          BPL SINEC
0F1C- 20 36 DD  2040          JSR CHGSGN
0F1F- A9 77     2050 SINEC    LDA #L,CONST4
0F21- A0 0E     2060          LDY #H,CONST4
0F23- 4C C2 DD  2070          JMP FLOAT1
                2080 ;
                2090          .EN
RAE CLOCK CORRECTIONS - DICK ALBERS
--- ----- -----------   ---- ------
```

13:09:53 WED DEC 30, 1981

Dear Lux,

Here are the last of the corrections to RAE CLOCK.

As you can see, the actual corrections are simple. The important ones are in line(s) 1050, and lines 3500 to 3800, since these portions do not run properly as published under many conditions. The other corrections are not as critical, since those problems only occur under certain unlikely conditions.

BRKFIX, INXIRQ, and OUTXIRQ are the subroutines that "should have been in SUPERMON" that you mentioned in the introduction to the program. These routines are general enough to be used any time IRQs may occur during I/O (unless you are using IRQs for interrupt driven input).

I too, would like to see an interrupt driven input program for SYM, but additionally, with the use of an ACIA or UART for serial/parallel, parallel/serial conversion.

The inspiration for this program was a similar one which you sent me that had appeared in an early issue of MICRO magazine (I found it in THE BEST OF MICRO, Vol. 3), written by Casmir J. Suchyta, III, and Paul W. Zitzewitz. That program was designed to be used with BASIC, and you commented how nice it would be to have something like it for RAE. That was received as a challenge, and RAE CLOCK is the result.

Thanks also to Jack Brown for pointing out an earlier mistake of using the same timer as SUPERMON uses for tape timing. Everything was fine until I did a tape read or write! It is the one at $A0XX, and was used by you in issue 2 of SYM-PHYSIS for a clock for the unexpanded SYM. I have been using various versions of that program as long as I have had a SYM, and I had never wondered "why?".

An easy way to make these changes without changing the original line numbers (so they can still be used for reference), is to enter the new lines after the end of the program, and then MOve them to their final location. If you really want to, you can renumber lines from MON, with ".M", to any sequence you desire. It's a lot of work though.

/s/ Richard Albers

```
0010
0020            ; RAE CLOCK CORRECTIONS
0030
0040            ;  By Richard R. Albers
0050
0060  ******    Insert the following definitions:
0070
0350  PBDA      .DE $A402     Terminal input port
0350  TOUTFL    .DE $A654     Terminal device flags
0350  INVEC     .DE $A660     Input vector
0350  OUTVEC    .DE $A663     Output vector
0350  UBRKVC    .DE $A676     User break vector
0350  SVBRK     .DE $804A     Save regs after break
0350  SAVER     .DE $8188     Register save routine
0350  TIN       .DE $8A6A     Part of INTCHR
0350  TOUT      .DE $8AA0     Terminal output subroutine
0470
0480  ******    Add the following lines to "LINK":
0490
0495            LDA #L,INXIRQ
0495            STA INVEC+1  Link input patch
0495            LDA #H,INXIRQ
0495            STA INVEC+2
0495            LDA #L,OUTXIRQ
0495            STA OUTVEC+1 Link output patch
0495            LDA #H,OUTXIRQ
0495            STA OUTVEC+2
0495            LDA #L,BRKFIX
0495            STA UBRKVC   Link BRK fix
0495            LDA #H,BRKFIX
0495            STA UBRKVC+1
0500
0510  ******    Fit the following in wherever you like:
0520
0520            ;  FIX RAE'S EXIT-TO-MON
0520
0520  BRKFIX    CLI           Clear IRQ inhibits
0520            JMP SVBRK     Continue as normal
0520
0520            ;  PREVENT GARBLING INPUT
0520
0520  INXIRQ    JSR INTX      Make INTCHR return here
0520            CLI            to allow IRQs, then
0520            RTS           Return to INVEC's caller
0520
0520  INTX      JSR SAVER     First part of INTCHR
0520            LDA #$00
0520            STA $F9
0520  LOOK1     LDA PBDA      Find leading edge
0520            AND TOUTFL
0520            SEC
0520            SBC #$40      Invert TTY polarity
```

```
0520              BCC LOOK1     Not yet
0520              SEI           Prevent garbled input
0520              JMP TIN       Let MON get char
0520              ;       (Returns via RESXAF to INXIRQ+3)
0520
0520              ;  PREVENT GARBLING OUTPUT
0520
0520  OUTXIRQ     SEI           Output char w/o IRQ
0520              JSR TOUT      Output char
0520              CLI
0520              RTS
0520
1030  ******      Replace line 1050 with the following:
1040
1050              CPX #$10      Test for Nov-Dec
1050              BCC GOTM      No
1050              BEQ NO
1050              LDX #$0B      It's Dec
1050              BNE GOTM      (Always)
1050  NO          LDX #$0A      It's Nov
1050  GOTM        LDA DAY/MO    Get current day
2490
2500  ******      Reverse lines 2520 & 2530:
2510
2520  CDONE       LDX SCRA      Get BUFF index
2530              CLI           Allow time changes
3430
3440  ******      Change line 3460 to:
3450
3460              JSR TOUT      Avoid CLI in OUTCHR patch
3470
3480  ******      Replace lines 3500 to 3800 with the following:
3490
3500  NXTR        JSR INBYTE    Get value for reg
3510              SEI           Minimize IRQ's
3520              BCS BAD       Non-hex not allowed
3530              PHA
3540              AND #$0F      Test low nibble
3550              CMP #$0A      Only 0-9 wanted
3560              PLA           Test high at STOT
3570              BCS BAD
3580              CPX #$04      Day of week?
3590              BEQ ADJUST
3600              CPX #$06      Month?
3605              BNE STOM
3610  ADJUST      SED           Month may be 10-12
3620              SEC
3630              SBC #$01      SUN or JAN=00
3635              CLD
3640              BCC BAD       Input of "00" n.g. here
3650  STOM        CPX #$05      Day of month?
3660              BNE STOT
3670              CMP #$32      Allow day of month = 31
3680              BCS BAD
3690              BCC STO       (Always)
3700  STOT        CMP TABL,X    Check range
3710              BCS BAD
3720  STO         STA TIMR,X
3730              DEX
3740              CPX #$07      Year needs 4 digits
3750              BEQ NYR        so skip CRLF if year
3760              JSR CRLF
3770  BAD         CPX #$07      Need re-prompt for year?
3780              BNE NYR       No
3785              INX           Yes, get all 4 digits
```



```
3790  NYR         LDA TSTAB,X   Index for prompt
3795              TAY
3800              BNE PMOR      If done, fall thru
3805
3850  ******      Line 3870 may be changed to:
3860
3870              JSR INVEC     Get any character
3880
3880              To allow any character to start the clock.
3880              (INBYTE requires a non-hex character.)
3880
3890  ******      Finally, you may delete lines 3890 and 3900,
3900              since both are included in START.
3910
3920
3930  !!!!!!      If you would like a more "standard" format:
3940
3950              ; 00:14:49 SUN  NOV 18, 1981
3960
3970  !!!!!!      Make these additional changes on the indicated lines:
3980
1615              .BY $2C       ;Comma
2570              LDA BUFF+$17
2610              STA BUFF+$17
2740              ADC #$1F      Add 31 bytes
2860              ADC #$1F
4140  DRTAB       .BY $C9 $CB $C9 $03 $CA $02 $CA $01
4150              .BY $84 $C9 $86 $05 $CC $C9 $00 $07 $00
```

OLD PROBLEMS AND NEW SOFTWARE POLICY

There is now far more quality software available for the SYM-1 than we (not editorial "we" this time, but us, you and me!) would ever have thought possible a year ago. Actually, we started the Users' Group with the selfish thought that it would provide us with "free" software, and the "noble" thought that we could expiate our selfishness by passing the software on to the rest of the SYM-1 community.

In practice however, we now have guilt-feelings about having become a bottleneck in the distribution system, since the better programs are, in general, much too long to publish in SYM-PHYSIS. The only alternatives are to publish them separately in book form, or distribute them on cassette and/or disk, and charge, not what the traffic will bear, but enough to pay for all actual costs incurred: media, printing, shipping, handling, outside labor, etc., plus at least minimum rate for the time we spend in editing, debugging, documenting, etc. After all, consulting would be much more remunerative!

We were semi-amused when one of our readers complained bitterly about the "exorbitant" prices Jack Brown was charging for his software (we felt much the same way about the prices of the 6502 Program Exchange, until we learned better, the hard way!). We were really amused when the same reader submitted a program for distribution at a far greater asking price for a very much lower quality program, explaining he had put so many hours into the work, and really should be remunerated for his time!

The publication of SYM-PHYSIS started as a hobby, and could easily continue as just that, a "spare time" activity, actually meeting the scheduled dates. The "slippages" are due to two unanticipated added responsibilities, neither one of which we would ever even have considered as a "hobby". One is answering the tremendous volume of mail asking for help with SYM-1 problems. Some problems are easily solved and many of the questions are easily answered; often a brief note suffices. Unfortunately, the more interesting questions and problems often require many

hours of "research" time, which are very frequently interrupted by telephone calls with still more problems and questions! This responsibility is one we cannot shirk; there would otherwise effectively be no source of help for SYMmers.

The second responsibility we shall give up, as explained here. To paraphrase Orson Welles (in a US TV wine commercial), our policy has been: "We shall release no program before its time". This meant that we examined each program carefully, understood its inner workings completely, exterminated as many of the bugs as we could find, tried to make it "crashproof", corrected any misspellings and grammatical errors, clarified any vague instructions, added missing documentation, and tried to give the package a "professional feel". This obviously has taken up much of our time, and has kept us from doing all of the "fun" things we got our computer for (this parenthetical expression is added only so that this sentence will not end with a preposition!).

Occasionally, as with SWP-1, since the demand for a word processor was so great, we released a package without the usual manual, to save time, hoping that the example provided, plus the fully commented source code, also provided, would make the package immediately usable anyway (besides, it would serve as a stimulus to seduce the reader into learning a new language, "source code"). Fortunately, most purchasers were equal to the task; very few had any real problems, and these few we helped get started by answering their mail or telephone inquiries.

After all, SYM users are "sharper" than the typical Apple user; their wits are honed by the exiguousness of SYM software, to mix a mean metaphor. Besides, manuals are expensive to prepare, print, handle, and mail, and the costs must be passed on to the purchaser. Since SWP-1 is an "accessory" to RAE-1, all purchasers had the ability to process the source code, which we would rather have than a manual, anyday! Some SYMmers have limited their SYMs to BAS-1, opting to omit RAE-1. We suggest they reconsider, since RAE-1 is one of the four major strong points of SYM-1 (see elsewhere for the other three). A whole brave new world will be opened up to them. Besides, RAE is needed for SYM-Pascal!

AND NOW, AT LAST, HERE IS OUR NEW POLICY ON NEW SOFTWARE:

To reduce the time delay which has up to now existed between the submission of software to be distributed by the SYM -1 Users' Group and the actual release date, all programs will be tested to see that they do work in at least a minimal acceptable way. Minor, non-catastrophic bugs which cannot be easily fixed will be annotated as such in the documentation; catastrophic bugs will, of course, be cause for rejection, and the program will not be distributed. Only a very minimum of editing and polishing will be done. RAE readable source code will be provided, and the purchaser should have RAE-1 or RAE-1/2 (both are functionally equivalent, one comes in a single 8K 2332, the other in two 4K 2316s) installed, and the skill to read and understand, and modify and reassemble the source code, if necessary, to eliminate any problem areas.

Only the minimum amount of hard copy documentation will be provided, i.e., just enough to get started. This will be at the very least just that provided by the author, and which was enough to get us started, and at the most we will add just enough to help the purchaser avoid all of the trial and error we had to go through to get the program working. These programs will be marked as NOT CERTIFIED. We suggest that only experienced users buy these versions. As feedback from early purchasers is received the program and documentation will be upgraded to make the package suitable for less skilled users, and the program will then be marked CERTIFIED.

We hope this new policy will help get useful programs out into the using community more rapidly, and help relieve the pressure on us. We prefer to ship data on SYM cassettes because the procedure for doing this is already "automated", and is handled by a relatively unskilled operator. Shipping on FODS 5 1/4 inch disks will require a medium surcharge. Shipping on CODOS 8 inch disks is not yet automated, and will require a surcharge for both medium cost and skilled labor. The same will hold true for FODS 8 inch disks when these are installed. The problem is that we currently do not have dual drives installed. Try making copies on a single drive system, and you'll understand!

When a program submitted to us is extremely useful but needs a major "overhaul" to bring it up to "commercial" standards we have been sending "review" copies to a few selected individuals who will either work with the authors, or do the necessary upgrading themselves, in exchange for the review copy. This is both bad news and good news. It is lots of work for them, but no work for us! Our most active reviewers have been Jack Brown and Dick Albers. Now that our new policy will give us the time to organize our time better, we hope to use more reviewers and have even more time for the "fun" things.

SYM-PASCAL IS READY

We just received in the mail from Saturn Software Limited (Jack Brown) a review copy of SYM-Pascal, Release 2.0 (16K RAM, cassette based), by Ralph Deane. We took a few minutes off to load the cassette, transfer the contents to disk, skim the manual enough to like what we saw, and entered the program with .G 200 <cr>. We promptly received the prompt:

```
SYM-PASCAL
COPYRIGHT RALPH DEANE 1981

25BD-35BD 35C1
25BD

]
```

If that reminds you a little of RAE-1's prompt, it should! SYM-Pascal requires that RAE-1 be installed, and makes extensive use of many of RAE's capabilities. 25BD-35BD is the area allocated to the Pascal source text; 35C1 is the start of the area where the P-code will be deposited during compilation. The 25BD on the line below is the current value of the text pointer. These values are all resettable.

To quote from the manual: "One of the reasons that SYM-Pascal is so compact is that it 'sits on top' of the RAE system and uses its editor and file system in the preparation of the Pascal source program. The compiler accepts RAE compatible text as its input." How very, very, elegant! Note that this is an Integer Pascal (hex as well as decimal).

We can say little more at this time for two reasons. First, the newsletter has highest priority; after the camera-ready copy has gone to the printer, we will have more time (?). Second, although most of Jack Brown's previous software will continue to be available through the Users' Group, SYM-Pascal is available only direct from Saturn Software; dealerships are not yet available.

We do not list prices here because some of you may wish FODS or CODOS linked versions, and since Release 2.0 does not include the source listing, you may prefer to have the linking done for you. Actually the linking should be easy, since all I/O is through RAE. SYM-Pascal recognizes PUt and GEt, but will give !ED errors on ENt, LOd, and DC. These are easily trapped, however, and your existing RAE links to FODS and CODOS used directly.

(continued on page 13)

BEHAVIORAL SCIENCES RESEARCH APPLICATIONS

Here is a letter we felt was well worth sharing with all SYMmers!

College of Physicians & Surgeons of Columbia University | *New York, N.Y. 10032*

DEPARTMENT OF PSYCHIATRY — 722 West 168th Street

January 6, 1982
Box 124 Psychology Department
Neurological Institute
710 W. 168th Street
New York, NY 10032
Telephone - 212-694-2214

Dr. H.R. Luxenberg
SYM Users' Group
P.O. Box 315
Chico, CA 95297

Dear Dr. Luxenberg:

Other SYM-1 users who are behavioral researchers will be interested to know that in our laboratory we have developed a group of programs for the 8-K SYM which present various kinds of visual and auditory stimuli on a CRT [or earphones in the case of auditory stimuli] and measure, store, and perform statistical analysis on subjects' manual reaction times (in milli seconds) to those stimuli. Stimuli are presented laterally; that is, to subjects' left and right visual fields [ears], as well as presented centrally [binaurally], in different conditions. The timing portion of these programs is written in assembly language. Programs for presentation of stimuli and for statistical analysis of reaction times are written in BASIC.

For the visual reaction-time experiments, the stimulus consists of a dot positioned about 10° to the right or left of a central fixation dot, or positioned in the center. The dot stays on for less than 180 milli seconds. In another experiment we present alphabet letters to left and right visual fields for a brief period and record subject reaction times. For the auditory experiments we are using the TI SN76488N complex sound generator chip to present simple tones (monaurally and binaurally in different conditions) through earphones.

The reaction time button is affixed directly to the SYM. A second button, operated by the experimenter, allows the experimenter to scratch "bad" subject responses (especially anticipatory responses).

In another ongoing project we are attempting to send our SYM-collected data over the phone lines to another, mainframe computer for large-scale statistical analyses. At present, this remains the ultimate Chinese (software) puzzle! We will inform you if we ever solve it!

Our laboratory is directed by myself and Rita G. Rudel, Ph.D. The programs and hardware modifications were made by Mr. Jack Harris. We feel we have only begun to scratch the surface of SYM-1 capabilities for driving sophisticated experimental and clinical diagnostic psychological procedures. We invite other behavioral researchers interested in learning more about our programs or exchanging ideas to contact us.

Sincerely,

Melinda Broman, Ph.D.
Assistant Clinical Professor
of Medical Psychology

MB:ms


```
0010 ;   ↑                                ↑
0020 ;  ↑↑↑    A CHRISTMAS GIFT!        ↑↑↑
0030 ; ↑↑↑↑↑                           ↑↑↑↑↑
0040 ;   I   FOR ALL BASIC SYMMERS       I
0050 ;  ---                             ---
0060 ;   ******************************
0070 ;   THE ULTIMATE B A S I C PATCH.
0080 ;   ******************************
0090 ;       E. DE LE COURT
0100 ;       AV.DES ERABLES 41
0110 ;       1640 RHODE S.G.
0120 ;       BELGIUM
0130 ;       TEL: 2-358 48 13
0140 ;WITH IDEAS FROM C.MOSER (S.P. #0)
0150 ;-----------------------------------
0160 ;THIS PROGRAM ALLOWS TO RECALL ANY WRITTEN LINE
0170 ;FOR CORRECTION,DELETION,ADDITION OF ANY CHR(S)
0180 ;W I T H O U T  THE NEED OF  R E T Y P I N G
0190 ;       THE  W H O L E  LINE
0200 ;IT ALLOWS STRINGS WRITTEN IN LOWER CASE
0210 ;IT ALLOWS $ FOR HEX NUMBERS
0220 ;ALL OPERATIONS ARE INMEDIATELY VISIBLE
0230 ;       O N  T H E  S C R E E N
0240 ;AN ATTEMPT TO WRITE MORE THAN 72 CHR IS LOCKED.
0250 ;YOU CAN NOW WRITE YOUR BASIC PROGAMS WITH
0260 ;THE  T R I A L  AND  E R R O R  METHOD
0270 ;
0280 ;EPROM THIS PROGRAM AT $F000 ALONG WITH TRIG AT $F6C7
0290         .BA $F000        OR ELSEWHERE BUT TAKE CARE
0300         .MC $0800        OF THE ##F0 AND ##F1 REFERENCES
0310         .OS
0320 BASVEC .DE $01
0330 TRIGL  .DE $C4
0340 EDFLG  .DE $F5
0350 SAVEY  .DE $F6
0360 INDEY  .DE $F7
0370 ;BASIC USES $00 TO $EF
0380 BUFFER .DE $100
0390 MONW   .DE $8003
0400 ACCESS .DE $8B86
0410 RESXAF .DE $81B8
0420 INTCHR .DE $8A58
0430 TOUT   .DE $8AA0        OR ANOTHER OUTPUT ROUTINE
0440 CRLF   .DE $834D
0450 TECHO  .DE $A653
0460 INVEC  .DE $A661
0470 OUTVEC .DE $A664
0480 BASIN  .DE $DE6D
0490 BASWR  .DE $C27E
0500 ;START LINKS
0510 ;-----------
F000- 20 0C F0  0520 BASCLD JSR SWIVEC       START HERE BASIC COLD
F003- 4C 6D DE  0530         JMP BASIN
F006- 20 0C F0  0540 BASWRM JSR SWIVEC       RETURN HERE WITH .G 0
F009- 4C 7E C2  0550         JMP BASWR
F00C- 20 86 8B  0560 SWIVEC JSR ACCESS       SWITCH VECTORS FOR BASIC
F00F- A9 2D     0570         LDA #L,INPUT
F011- 8D 61 A6  0580         STA INVEC
F014- A9 F0     0590         LDA #H,INPUT
F016- 8D 62 A6  0600         STA INVEC+1
F019- A9 68     0610         LDA #L,TRIGIN
F01B- 85 C4     0620         STA *TRIGL
F01D- A9 F7     0630         LDA #H,TRIGIN
F01F- 85 C5     0640         STA *TRIGL+1
F021- A9 00     0650         LDA #0
```

```
F023- 85 F7     0660         STA *INDEV      LET BUFFER INDEX = 0
F025- 8D 53 A6  0670         STA TECHO       INHIBIT ECHO
F028- A9 20     0680         LDA #$20
F02A- 85 F5     0690         STA *EDFLG      NORMAL FLAG
F02C- 60        0700         RTS
                0710 ;INPUT OUTPUT LINKS
                0720 ;-----------------
F02D- 68        0730 INPUT   PLA             DISCARD A AND F
F02E- 68        0740         PLA
F02F- A4 F7     0750         LDY *INDEV
F031- F0 0E     0760         BEQ GETCHR      BUFFER EMPTY
F033- 24 F5     0770         BIT *EDFLG
F035- 30 03     0780         BMI =+4         FLAG = EDIT MODE
F037- 4C 16 F1  0790         JMP READBU      BASIC READ THE BUFFER
F03A- A9 20     0800         LDA #$20        RESET EDIT FLAG
F03C- 85 F5     0810         STA *EDFLG
F03E- 20 4A F1  0820         JSR RESOVC      RESET OUTPUT VECTORS
F041- 20 58 8A  0830 GETCHR  JSR INTCHR      INPUT 1 CHR
F044- 29 7F     0840         AND #$7F
                0850 ;ESCAPE ln CR RECALLS LINE ln FOR EDITION.
                0860 ;THEN ESCAPE RETURNS THE WHOLE LINE TO BASIC
                0870 ;OR CR RETURNS THE PART AT THE LEFT OF THE CURSOR.
F046- C9 1B     0880 ESCAPE  CMP #$1B        EDIT 1 LINE
F048- D0 24     0890         BNE CTRLH
F04A- C0 00     0900         CPY #0          VALID ONLY AS FIRST CHR
F04C- F0 06     0910         BEQ =+7
F04E- 20 4D 83  0920         JSR CRLF
F051- 4C 14 F1  0930         JMP READBU-2    OTHERWISE RETURN
F054- A9 2D     0940         LDA #L,WRITBU   CHANGE OUTVEC TO
F056- 8D 64 A6  0950         STA OUTVEC      WRITE IN BUFFER FROM LIST
F059- A9 F1     0960         LDA #H,WRITBU
F05B- 8D 65 A6  0970         STA OUTVEC+1
F05E- B9 46 F1  0980 CMDLST  LDA LIST,Y      PUT"LIST" IN BUFFER
F061- 99 00 01  0990         STA BUFFER,Y
F064- 20 A0 8A  1000         JSR TOUT
F067- C8        1010         INY
F068- C0 04     1020         CPY #4
F06A- D0 F2     1030         BNE CMDLST
F06C- F0 D3     1040         BEQ GETCHR      PUT LINE NUM. IN BUFFER
                1050 ;CTRL H MOVES THE CURSOR LEFT IN THE BUFFER
F06E- C9 08     1060 CTRLH   CMP #8           LEFT TAB
F070- D0 06     1070         BNE CTRLI
F072- 88        1080         DEY
F073- 10 08     1090         BPL JTOUT       VALID ONLY IF Y>=0
F075- C8        1100         INY
F076- 10 C9     1110         BPL GETCHR
                1120 ;CTRL I MOVES THE CURSOR RIGHT IN THE BUFFER
F078- C9 09     1130 CTRLI   CMP #9           RIGHT TAB
F07A- D0 06     1140         BNE CTRLZ
F07C- C8        1150         INY
F07D- 20 A0 8A  1160 JTOUT   JSR TOUT
F080- B0 BF     1170         BCS GETCHR
                1180 ;CTRL Z DELETES THE CHR AT THE CURSOR
F082- 84 F7     1190 CTRLZ   STY *INDEV
F084- C9 1A     1200         CMP #$1A        DELETE A CHR
F086- D0 0D     1210         BNE CTRLQ
F088- B9 01 01  1220 MOVDWN  LDA BUFFER+1,Y  MOVING BUFFER 1 LEFT
F08B- 99 00 01  1230         STA BUFFER,Y
F08E- C8        1240         INY
F08F- C0 48     1250         CPY #72         TILL END OF BUFFER
F091- D0 F5     1260         BNE MOVDWN
F093- F0 11     1270         BEQ PRTBUF
                1280 ;CTRL Q ALLOWS INSERTION OF A CHR AT THE CURSOR
F095- C9 11     1290 CTRLQ   CMP #$11        OPEN A HOLE
F097- D0 2E     1300         BNE CTRLY
```



```
F099- A0 47     1310         LDY #71         FROM END OF BUFFER
F09B- 88        1320 MOVUP   DEY             MOVING BUFFER 1 RIGHT
F09C- B9 00 01  1330         LDA BUFFER,Y
F09F- 99 01 01  1340         STA BUFFER+1,Y
F0A2- C4 F7     1350         CPY *INDEV
F0A4- D0 F5     1360         BNE MOVUP
F0A6- A4 F7     1370 PRTBUF  LDY *INDEV      REFRESH THE DISPLAY
F0A8- B9 00 01  1380         LDA BUFFER,Y
F0AB- C8        1390         INY
F0AC- C9 0D     1400         CMP #$0D
F0AE- F0 05     1410         BEQ MOVCUR
F0B0- 20 A0 8A  1420         JSR TOUT
F0B3- D0 F3     1430         BNE PRTBUF+2
F0B5- A9 20     1440 MOVCUR  LDA #$20        RESET THE CURSOR
F0B7- 20 A0 8A  1450         JSR TOUT
F0BA- A9 08     1460         LDA #8          GOING BACK
F0BC- 20 A0 8A  1470         JSR TOUT
F0BF- 88        1480         DEY
F0C0- C4 F7     1490         CPY *INDEV
F0C2- D0 F8     1500         BNE MOVCUR+7
F0C4- 4C 41 F0  1510 JGETCH  JMP GETCHR
                1520 ;CTRL Y JUMPS TO MONITOR.RETURN WITH .G 0 CR
F0C7- C9 19     1530 CTRLY   CMP #$19        RETURN TO MON
F0C9- D0 21     1540         BNE STOCHR
F0CB- A9 4C     1550         LDA #$4C
F0CD- 85 00     1560         STA *BASVEC-1
F0CF- A9 06     1570         LDA #L,BASWRM
F0D1- 85 01     1580         STA *BASVEC     SET RETURN VECTOR
F0D3- A9 F0     1590         LDA #H,BASWRM
F0D5- 85 02     1600         STA *BASVEC+1
F0D7- A9 58     1610         LDA #L,INTCHR   RESET INPUT VECTORS
F0D9- 8D 61 A6  1620         STA INVEC
F0DC- A9 8A     1630         LDA #H,INTCHR
F0DE- 8D 62 A6  1640         STA INVEC+1
F0E1- A9 80     1650         LDA #$80        ALLOW ECHO
F0E3- 8D 53 A6  1660         STA TECHO
F0E6- 20 4A F1  1670         JSR RESOVC      RESET OUTPUT VECTORS
F0E9- 4C 03 80  1680         JMP MONW
F0EC- C0 48     1690 STOCHR  CPY #72         STORE NOT TOO FAR
F0EE- F0 D4     1700         BEQ JGETCH
F0F0- 99 00 01  1710         STA BUFFER,Y
F0F3- C8        1720         INY
F0F4- 20 A0 8A  1730         JSR TOUT
F0F7- C9 24     1740         CMP #'$         ALLOW $ FOR HEX
F0F9- D0 15     1750         BNE NOTHEX
F0FB- 20 58 8A  1760         JSR INTCHR
F0FE- 29 7F     1770         AND #$7F
F100- C9 22     1780         CMP #'"
F102- F0 03     1790         BEQ =+4
F104- 4C 46 F0  1800         JMP ESCAPE      NOT ",THEN PROCEED
F107- A9 26     1810         LDA #'&
F109- 99 FF 00  1820         STA BUFFER-1,Y
F10C- A9 22     1830         LDA #'"
F10E- D0 DC     1840         BNE STOCHR
F110- C9 0D     1850 NOTHEX  CMP #$0D
F112- D0 B0     1860         BNE JGETCH
F114- A0 00     1870         LDY #0
F116- B9 00 01  1880 READBU  LDA BUFFER,Y    WHEN CR SEND BUFFER
F119- C8        1890         INY             TO BASIC
F11A- C9 0D     1900         CMP #$0D
F11C- D0 08     1910         BNE NEXTCH
F11E- A0 4F     1920         LDY #79
F120- 99 00 01  1930         STA BUFFER,Y    FILL BUFFER WITH CR
F123- 88        1940         DEY
F124- D0 FA     1950         BNE =-5
F126- 84 F7     1960 NEXTCH  STY *INDEV
```

```
F128- C9 0D     1970        CMP #$0D
F12A- 4C B8 81  1980        JMP RESXAF      ALLOW FOR LOWER CASE CHR
F12D- 24 F5     1990 WRITBU BIT *EDFLG      LIST WRITES HERE WHEN EDIT
F12F- 30 0A     2000        BMI NOPRT       DONT WRITE AFTER 2ND CR
F131- C9 0A     2010        CMP #$0A        IGNORE LF
F133- F0 06     2020        BEQ NOPRT
F135- C9 0D     2030        CMP #$0D        IGNORE CR
F137- D0 03     2040        BNE =+4
F139- 06 F5     2050        ASL *EDFLG      BUT ADJUST FLAG
F13B- 60        2060 NOPRT  RTS
F13C- A4 F7     2070        LDY *INDEY
F13E- 99 00 01  2080        STA BUFFER,Y    WRITE LINE IN BUFFER
F141- E6 F7     2090        INC *INDEY
F143- 4C A0 8A  2100        JMP TOUT        AND PRINT
F146- 4C 49 53  2110 LIST   .BY 'LIST'
F149- 54
F14A- A9 A0     2120 RESOVC LDA #L,TOUT     RESET VECTORS
F14C- 8D 64 A6  2130        STA OUTVEC
F14F- A9 8A     2140        LDA #H,TOUT
F151- 8D 65 A6  2150        STA OUTVEC+1
F154- 60        2160        RTS
                2170 ;BASIC TRIG PROGAM
                2180 ;-----------------
                2190 TRIG   .DI BASCLD+$6C7
                2200 TRIGIN .DI TRIG+$A1
```

INCIDENTALLY, THANKS TO DR. G. STRUBBE (S.P.#9),
I IMPLEMENTED 8 CONTROL KEYS AT THE 8 FREE PLACES
OF MY KTM 40 BOARD. IT IS WUNDERBAR!!
THE KEYBOARD ENCODING TABLE STARTS AT $7FA UP TO $879
I FOUND IT THANKS TO HISSINK'S DISASSEMBLER(S.P.#8)
THAT I HAVE, REWORKED A LOT FOR EASIER USE
AND MORE FUN(IT TRACES ALSO).
HEREAFTER MY SUBSCRIBTION RENEWAL.
H A P P Y N E W Y E A R

E. de la Court

NOTE THAT PERSONALLY I DONT LIKE BASIC
THIS PROGRAM WAS MADE FOR MY DAUGHTER.

A SHORT, BUT VERY SWEET, COMPUTER ASSISTED INSTRUCTION PROGRAM

Dear M. de la Court:

Thank you for the Christmas gift (above) for all BASIC SYMmers. While I have not tried your program out personally, it looks great, and most BASIC SYMmers will especially appreciate its line editing capability. This, plus Jack Brown's earlier published line renumbering program should allow one to really polish up the appearance of their BASIC programs.

Below is a program for your daughter, and all SYM BASICers, which I did not edit or "polish" (except to correct one misspelled word) in any way, so that SYMmers can practice using your line editor on it.

Sincerely,

Lup

EDITORIAL NOTES TO CRISWELL'S CAI BASIC PROGRAM

This, as yet unnamed, modest, mild, unassuming, little program is far

more powerful and versatile than at first glance it appears to be, and I strongly recommend it to all BASIC users. It was submitted by Dr. Hugh E. Criswell, another behavioral researcher, whom I hereby formally introduce to Dr. Broman (see her letter on page 25).

Dr. Criswell also submitted the necessary object code to permit saving and recalling data files (see p. 7:17), and a set of sample questions, with answers, as a "saved" data file. The questions so well illustrate the versatility of the program that we provide them, following the LISTing, as a "puzzle" for the reader. We used the Alphanumeric Verify published in an earlier issue to "dump" the data file in ASCII format. Readers might wish to see if they can figure out the questions and answers from this dump.

Studying this string file dump will certainly help provide an understanding of how BASIC processes and stores its strings. Remember that BASIC stores all generated strings and string arrays from the top of the memory downwards.

Although the program is far from self-explanatory, it is sufficiently simple in structure to be easily understandable by beginners, and is highly recommended to them as a useful way to learn how to use string arrays.

```
10 DIM A$(20,20),B(20)
11 REM A$ IS THE QUESTION MATRIX;A IS # OF LINES, B IS ANSWER.
15 DATA WRONG! TRY AGAIN,NOPE! REDO IT YOU TURKEY,INCORRECT! REPEAT IT
16 DATA "POOR ANSWER TRY AGAIN!"
20 L=1
21 PRINT"S=STORE,R=RETRIEVE"
30 PRINT"TYPE W TO WRITE A QUESTION OR L TO LOOK AT ONE":GOSUB5000
40 IF X$="W"THEN GOSUB1000
50 IF X$="L"THEN GOSUB2000
51 IF X$="S"THEN X=USR(&"1C20",&"0000")
52 IF X$="R"THEN X=USR(&"1C7B",&"0000")
60 GOTO30
1000 J=0
1001 PRINT"QUESTION #";L,FRE(0)
1010 PRINT"TYPE A QUESTION AND END IT WITH A ! AS THE LAST LINE"
1015 J=J+1
1020 GOSUB5000:A$(L,J)=X$
1030 IF A$(L,J)<>"!"GOTO1015
1040 PRINT"THE RIGHT ANSWER IS":GOSUB5000:B(L)=X
1050 L=L+1
1055 IFL>20THENGOSUB3000
1060 RETURN
2000 PRINT CHR$(27)+"E";
2001 I=INT((L-1)*RND(1)+1)
2005 REM THIS ENTERS A QUESTION INTO THE POOL
2010 J=1:X=SGR(2)
2012 PRINT"QUESTION #";I,FRE(0)
2020 PRINT A$(I,J)
2030 J=J+1
2040 IF A$(I,J)<>"!"GOTO2020
2050 PRINT"WHAT'S YOUR ANSWER?"
2060 GOSUB5000:A=X
2070 IF A=B(I)THENPRINT"RIGHT! GOOD JOB":RETURN
2085 READ C$
2090 IFC$="POOR ANSWER TRY AGAIN!"THENRESTORE
2100 PRINTC$:GOTO2060
3000 PRINT"JUST A SECOND, I'M MEMORIZING THESE QUESTIONS"
3001 X=USR(&"1C20",&"0000")
3010 L=5:RETURN
5000 X$="":X=0:W=1
5001 REM GUARDED INPUT AND BACKSPACE
```

```
5020 Y=USR(-30120,0):Y=127ANDNOTPEEK(249)
5021 IFW>254THENRETURN
5022 W=W+1
5030 IFY=13THENPRINTCHR$(13):RETURN
5035 X$=X$+CHR$(Y)
5040 IFY=95ANDLEN(X$)>0THENGOSUB5100
5060 IFY>46ANDY<58THENX=VAL(X$)
5070 GOTO5020
5100 X$=LEFT$(X$,LEN(X$)-2):PRINTCHR$(8);CHR$(8);:RETURN
```

ASCII DUMP OF DATA FILE (FROM $0CD0 - $114F)

```
.....SS44L!5,
2,4, and 5 are c
orrect,    you f
inish 20 questio
ns.4,  Typing lo
ng questions may
 run you out of
memory before3,
 Control c will
cause a crash.
  crash.2,  Typi
ng a question mo
re than 20 lines
 long will cause
 a    crash.1,
Typing return wi
thout first ente
ring data will c
ause a  What bug
s are left in th
is program?!ques
tion is 17.as lo
ng as the answer
 is a number.  T
he rightanswer t
o this     Note:
  You do not hav
e to use a multi
ple choice forma
t!    in an occa
sional " sign or
 use bad grammar
.)    answers or
 just use one li
ne for them, and
 that you can th
row6,  (this is
just to show tha
t you do not hav
e to give 4 poss
able5,  none of
the above4,  all
 of the above
 of the bugs are
 worked out.3,
# o times the pr
ogram can be exp
ected to bomb be
fore all2,  # of
 hours it took t
o write this !"@
##! program1,  #
 of bytes of mem
ory remainingnum
ber?     What is
 the strange num
ber that you see
 after the quest
ion!4,  20 lines
 or less..allrig
ht, so it's a ba
d question.3, 30
 lines2, 20 line
s1,  10 linesrun
ning this progra
m; How many line
s can a question
 have?     Just
in case you didn
't look at the D
IM statements be
fores.2.V.j.i.X.
```

ON SYNERTEK'S FLOPPY DISK CONTROLLER (FDC-1)

Published below are extracts from a letter from Joe Hobart, and a manuscript describing his experiences with the FDC-1. Joe and I are two of the six individuals outside of Synertek who have been "lent" pre-production samples for evaluation (don't know the other four, since they're not SYMmers!). The "Jared" referred to in the letter is Jared "Jerry" Larsen, of Synertek Systems, a great person to work with.

My review follows Joe's, but first let me say this: I used to assert that there were THREE major "selling" points for the SYM; SUPERMON (with all its vectoring and user available utilities), RAE-1 (with or without its extensions, SWP, DISARAE, XREF, etc.), and SYM's richness in I/O resources (3-6522s, 1-6532). There will soon be a FOURTH major strength: the FDC-1 disk system!

3465 North Andes Dr
Flagstaff, AZ 86001
(602) 779-2110

January 15, 1982

Hi Lux:

Here is the manuscript of my first experiences with Synertek's new FDC-1. I hope it is suitable and not too late for inclusion in the next issue of SYM-PHYSIS. I do not know when Synertek will have the production versions available for sale, but this information seems very timely.

This is my first disk experience and I must say that there were a few frustrating moments. I zapped the controller with static electricity or something and spent a week trouble-shooting it without any kind of a schematic. I also spent some additional time getting all the jumpers in the Shugart set to the normal configuration. My brother must have had a rather unusual controller when he was using this drive. It was really a great relief when I studied the disassembled listing for the FDC software, found addresses at $F000-F100, removed my EPROM, and heard the drive find track 00.

I hope that Synertek does supply the cables to go between the drive and the controller. I had to modify a standard Shugart cable; those press-on 50 pin connectors require a LOT of force to seat properly. I had to use a large bench vise to do the job.

The manuscript only hints of some of the problems I encountered in getting the disk system running. I attribute most of the problems to inexperience with the system. I remember similar problems when I first got SYM-BASIC running (especially with the TCP). As I became used to the system, the number of problems decreased very significantly. I am sure the same thing will happen with the disk system.

One of the most difficult areas is in getting the system to work in RAE with both disk and cassette. Jared suggested toggling $EE to get the cassette to work, but this does not always work. I have not fully checked all possibilities, but it seems that the cassette will not work if a disk save (ENter) has been done since RAE initialization. This will require some experimenting. I still do not understand what was happening to prevent me from reloading BASIC programs, but that problem seems to have gone away since I put TCP back in EPROM. Speaking of TCP, I just do not know enough about the SYM to be able to make TCP and SYM-DOS work together. Perhaps one of the users will have some ideas.

Jared did ask me to pass along that you should not get an error indication when you verify data on a disk that was taken from a changing source. Apparently you were saving data from a clock register and then expecting to see an error on the verify. From my understanding of disk systems in general, I think that the verify only compares the data on the disk with the CRC that was derived from the data sent to the disk. In the IBM format this CRC is recorded in each sector near the data. I have two known bad disks from our PDP-11 at work. The verify does detect the bad areas on these disks. I have not been able to use either disk past the bad area yet, but I think several tracks have been damaged by a poorly aligned drive.

Jared did pass along some information on how the R/W routines work. My next project is to modify my fig-FORTH to work with the disk. I did send Jared a copy of my fig-FORTH. After some problems getting it to load on his MDT 1000, he reports it works fine. By the way, Sandy McKay tried my DRAM fix of an extra ground wire between the 6502 ground and the memory ground bus with excellent results. Apparently all his memory problems and the system crashes when he turned on his teletype just vanished. Sandy used a long piece of wire (6 inches) too.

Enough rambling. I really do like having a disk system at last.

Best regards,

/s/ Joe Hobart

EXPERIENCES WITH SYNERTEK'S NEW FLOPPY DISK CONTROLLER

For the past two weeks, I have been using an engineering prototype of Synertek's new floppy disk controller. During this time the controller has performed very well, providing fast and reliable mass data storage by interfacing an eight inch Shugart drive to my SYM-1. Here are some

observations and suggestions for using this new controller with expanded systems:

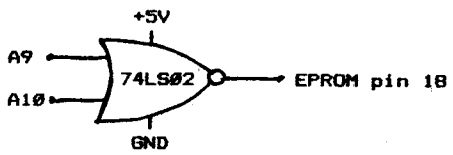
Since I have had difficulty using dynamic RAMs with my SYMs, I was very apprehensive about the physical separation between SYM and memory caused by putting a six inch controller between them. My fears were unfounded; the system works beautifully with the controller attached to the expansion port and either a 32K Beta memory board, a 32K static memory board, or a 4K static memory, attached to the controller board. Based on experience with my dynamic memories, an additional ground wire between one of the SYM's 6502 ground pins and the memory ground bus would help cure any problems that develop.

The interface between SYMDOS and SUPERMON is very smooth. A simple .8 9006 links the two and adds disk save, load, format, and list directory commands to those of the monitor. It is really nice to be able to load my 7300 byte FORTH in about 1 and 1/2 seconds as compared to about one minute required for a cassette tape load. The FDC-1 was designed to work immediately with a 4K SYM. For use with larger memory, it is convenient to relocate the disk buffer/workspace away from the $0E80-$0FFF default location. This is accomplished by changing the contents of $A62A-$A62B to point at the new buffer location.

The FDC also works smoothly with RAE-1. Once in RAE, a simple >RUN $9003 links SYMDOS to RAE and provides disk ENter, LOad, LOad and Append, and assemble multiple source files. It is while assembling multiple source files that the speed and reliability of a disk over a cassette system become especially apparent. I vividly remember some strong frustrations with my cassette system while repeatedly assembling fig-FORTH during the debugging stage. Assembling these same source files from disk was a very pleasant change.

To take full advantage of a memory larger than 4K, either the buffer location must be changed or the RAE memory allocations must be juggled to avoid this buffer area. A minor annoyance that will decrease as a user becomes more disk based, is that the cassette interface is not readily available once SYMDOS is linked to RAE. This problem can be partially circumvented by toggling the contents of memory location $00EE from 01 when using disk to 00 when using the cassette interface.

Interfacing SYMDOS with my BASIC was not as convenient as with SUPERMON or with RAE. The FDC uses $F000 - $F1FF which conflicted with my EPROM containing Brown's Super TCP and Renumber and the Synertek Trig Patch. To resolve this conflict, I modified the contents of the EPROM so that it occupies only $F200-$F7FF and disabled the EPROM whenever $F000-$F1FF is addressed by using 1/4 of a 74LS02 NOR gate in the circuit below:



EPROM pin 18 (chip enable) must be disconnected from ground. I connected wires to A9 and A10 at pins 19 and 22 of the SYM ROM sockets.

I mounted the 74LS02 over U24, and soldered pins 7 (GND) and 14 (+5V) to the same pins of U24. All other pins are bent upward and away from U 24. A9 and A10 go to pins 11 and 12 and the output is taken from pin 13, but there are three other combinations, as this is a quad NOR chip!

Once in BASIC, an X=USR(&"9000",0) adds disk SAVE and LOAD commands.

Synertek also added 4M to allow easy escape to the monitor so that disk format and list directory commands are accessible. Since BASIC is very inflexible in memory addressing, the disk buffer must be moved to take advantage of a memory larger than 4K. BASIC's top of memory and string pointers at $0083 and $0087 must be modified to protect the disk buffer area. While working on the modification to my TCP, I encountered some problems with not being able to reload programs that had been saved with different BASIC top of memory and string pointers. Once I finished the modifications to my TCP, I had no further problems. I recommend that no serious program saves be done until memory parameters are stable.

I saw no easy way to have SYMDOS and Brown's TCP operate simultaneously. To simplify access to disk-linked BASIC, I added a CONTROL D function to TCP. The source listing shows the changes required to Brown's TCP to enable the escape to SYMDOS. Return to TCP is by either X=USR(0,0) from BASIC or .G0 from SUPERMON. This approach has the merit of being simple to implement. I would like to see how others handle this interface. My TCP resets all necessary vectors, et cetera, each time it is entered.

Synertek should be congratulated for making available a relatively low cost disk controller that works with SUPERMON, RAE-1, and BAS-1 without requiring any modifications to the SYM-1. It is delightful to be able to load programs like my "Pirates' Adventure" in seconds instead of minutes required by cassette tape.

LUX'S COMMENTS ON THE FDC-1

The FDC-1 software will handle one or two, single or double sided, 5 1/4" or 8", drives. 5 1/4" systems may be single or double density; 8" systems are restricted to single density (the on-board crystal is 16.0 MHz; this is part of the restriction).

The hardware is factory-jumpered for 5 1/4" drives; instructions for re-jumpering for 8" systems are provided. Joe Hobart is working with 8" drives, Lux with 5 1/4" drives. Both report success and satisfaction.

5 1/4" systems use 34 wire cables from controllers to drives, 8" systems use 50 wire cables; separate connectors for both sizes of cables are provided on the controller. Actually only a single, either 50 or 34, pin connector is required. HDE uses a 34 wire cable even for their 8" systems! It turns out that the 26 wire pinout used for the Apple's Disk II, and the "Shugart Standard" 34 wire pinout for 5 1/4" drives, are contiguous subsets of the "Shugart Standard" 50 wire pinout for 8" drives, with at most one exception, which is easily taken care of with jumpers. The "oddball" signal is related to differences between hard and soft sector systems, e.g., Shugart 800 vs 801, or 850 vs 851.

The software is contained in a (4K) 2732 EPROM (remember this is not the same as the 2532!); I must "upgrade" my EPROMmer to handle these! This software resides at $9000, and might be considered as the long-awaited SUPERMON "Future Expansion" plus "extensions" for BAS-1 and RAE-1. From SUPERMON a .G 9006 modifies URCVEC to point to the commands .S3, .L3, .S9, and .L7, for saving, loading, formatting, and obtaining file listings. The parameters permit using up to ten character filenames, in addition to the usual numerical parameters.

The default values for the disk buffer area are at the top of the first 4K of RAM; no off-board expansion is required! The defaults could be lowered to fit a 1K system, if you would really want to! As Joe pointed out, the defaults are in SYSRAM, and should be raised for BAS and RAE. Since the program is in EPROM the best bet is to burn your own, with your own choice of defaults. Only a preliminary manual with neither schematics nor source code is as yet available.

Installation was extremely simple; the whole job took only five minutes.

The test board came with a 44-pin connector soldered to one of its sets of edge "fingers" (the other set passes the signals straight through for further expansion). We read the manual, soldered two leads (red and black coded) to terminals on the board for direct connection to the +5 V power supply (that's all!), mounted it on the expansion connector of an 8K SYM, installed a known-to-be-working 5 1/4" drive system (just by connecting the drive cable), and went on to test the system.

Then came two hours of increasing fustration with peculiar responses! We finally gave up with the hardware and began on the software. We disassembled the program and almost immediately found the solution to our problem. The controller's control registers were assigned addresses at $F000 and up, where we already had an EPROM installed. Removing the EPROM did the trick; the system worked as described in the preliminary manual.

The operating system treats the disk as a long serial medium and stores successive files sequentially. If a file name is reused the previous file with that name is not deleted; it is just no longer accessible, last-in-only-out! Utility programs for disk compacting and copying are promised, but did not come with the preliminary documentation we received.

The save-with-verify mode did not operate as we had come to expect it should. FODS includes a compare-disk-to-memory after each read or write. Several times this feature has informed us that our RAM was flakey because of a poor socket contact; we like the security of the compare.

We do not like the "wasting" of the 2K block at $F000 for just four register addresses $F000-$F003 and the page $F100-$F1FF. These could have been assigned within the $9000 block, with on-board logic to permit these addresses to "over-ride" the EPROM. Joe's fix is fine, but we will also investigate the possibility of reassigning these controller registers to the $A88X or $AC8X area (lots of "wasted" space there now), where all such I/O devices should go, when we get around to reburning an EPROM with more suitable default values, most likely for an 8K system.

The program now occupies $9000-$9770 and $9800-$9FFF; there is still lots of room available for more niceties in there. This would be a good place to patch in the now unused USR0 through USR7 as well as modifying the "useless" Jump Entries, J5 - J7, available in the existing portion of SUPERMON.

We have asked some of our product "reviewers" to examine the existing software to see what modifications would be required to permit reading and writing Apple II and/or CP/M (at least ASCII files) compatible disks. Since no higher level file management capabilities (even the simpler utilities such as COPY, PACK, etc., are not yet ready) exist in SYM-DOS, the opportunities exist for adapting an existing DOS (with built-in compatibility) or creating a "superior" one by incorporating the best features of various existing DOSes.

MORE ON DISKS AND MEMORY EXPANSION

The FDC-1 is a 4 1/4 x 5 1/2 inch "card" with a "Reverse KIM/SYM" pinout, intended to be installed extending outwards from the expansion connector. Our test board had a 44-pin connector soldered to it which was used for our first tests. This was later desoldered and removed, and a pair of 44-pin connectors soldered together, back-to-back, was used instead, to permit testing several alternate configurations.

Three of these alternate configurations we plan to try are as follows: First, a 4K SYM with the 32 K Beta Board "tucked under" and the FDC-1 also mounted underneath the SYM with a short cable attaching it to the

"free" set of edge fingers on the Beta Board, in a very compact package. This package, suitably encased, will be mounted atop a pair of 5 1/4 inch drives, also suitably encased. This configuration would be similar to our current FODS based system.

Second, an 8K (Blalock Expander Kit) SYM, plus the Quest "Motherboard" Expander Kit, in which will be installed the FDC-1 and the Turpin ColorMate Board (also Reverse KIM/SYM pinout) for color graphics. The third is the obvious one, which is too "spaced-out", in our opinion, mounting the FDC-1 directly on the expansion connector with the Beta Board extending outward from it. This approach we will try only to confirm that it can be done, as per Joe Hobart's report above.

Joe seems to have found an easy solution to the problem so many of us ran into when we tried to add memories such as the 32K Beta Board with too long an extension cable. While the Beta Manual recommended bringing in a direct +5 V line because of possible DC drop and heat in the narrow printed traces, it did not explicitly mention the same situation would occur in the GND traces! In addition there certainly would be "AC" drops, i.e., ground loops, for transient signals, which could, and did cause sub-marginal performance. Thanks, Joe! The FDC-1 has separate terminals to which both power leads are to be brought; apparently a very good engineering practice to follow in multi-board systems.

WHEN AND WHENCE THE FDC-1?

Synertek Systems Corporation has offered the SYM-1 Users' Group exclusive rights to manufacture and distribute the Floppy Disk Controller, FDC-1, in consideration for our completing all of the necessary supporting documentation, and providing all required Customer Service and Support for the FDC-1, on a long-term, continuing basis.

The latter we can do; it would merely be an extension of what we have been doing for SYM-1, KTM-2, BAS-1, RAE-1, etc., these past two years. To be rather immodest about the matter, SSC is offering us the FDC-1 product line because they feel we have the ability to support it in the manner to which they have become accustomed!

Software and System Support are our areas. The idea of setting up a hardware assembly line facility never entered our thoughts, and is still something we are not yet ready for. On the other hand, the sorting and gathering of the component parts into "build it yourself" kit packages we could easily do. The assembly and checkout of "ready-to-go" boards is something we would have to grow towards.

We plan to place a firm order with SSC by 1 April for a specific initial number of boards and components for delivery to us by 1 June 1982. We will also place an order for both 5 1/4 and 8 inch dual-drive cables with another vendor. Our aim is to produce a complete package, including all connectors and dual-drive cables (these alone are normally $35.00 and up!); all you need add are the drives (with power supplies, and power cables) to have a fully operational system.

Remember, though, that the FDC-1 will be available at first only in complete kit form. It should be easy to assemble, since in addition to the sockets and connectors, there are only five resistors and six capacitors to be soldered in. Schematic and layout drawings will be provided.

The kit price should be around $175.00 (US funds), including shipping, USA or Canada, with an additional $6.00 for airmail to Europe, or $8.00 for airmail elsewhere. If this interests you, please drop us a note; your feedback will let us know how large an initial order to place, and what ratio of 8 inch to 5 1/4 inch cables to order. Issue No. 11, which will reach you by mid-April, will give firm prices, detailed specs, and full ordering information. Deliveries should begin in mid-June.

ANNOUNCEMENTS

We have accepted an invitation from the Department of Electrical Engineering of the Queensland Institute of Technology to be the keynote speaker at "A Two Day Design Workshop on 6502/6809 Microcomputer Systems", to be held 14-15 April 1982, in Brisbane, Australia. Papers will be presented by Bob Tripp (MICRO), Rodnay Zaks (SYBEX), a Rockwell representative, and a number of Australian researchers. We are looking forward to the traveling, the workshop, the "vacation", meeting many of our Australia/New Zealand/Tasmania friends, etc.

From 10 April to 10 May, approximately (see above), the Users' Group office will be "partially open" on Mondays, Wednesdays, and Fridays, from 10AM to 4PM, Pacific Coast Time, to handle telephoned and mailed "business" matters. Unfortunately, there will be no one available to answer technical questions, or to help solve technical problems. These will be back-logged till our return.

Since we are OEMing a number of special purpose SYM based systems, we regularly buy various selected items at OEM prices in quantities sufficient to get good price breaks. These include Epsons (all models), hard-to-get connectors, special purpose chips, etc., and always have a few around. Because prices and stocks on hand are variable these are not listed in our flyers. Call or write if you have special needs; we may be able to help.

ADVERTISING POLICY

We have been asked many times if we would accept advertising, and, if so, what our rates were. Up to now our answer has been "No, but if you will lend us the equipment for test, or lend us a copy of the manual, we will review the product in an 'up-coming' issue." In most cases we so liked the product that we purchased it for our own use, and even became dealers for it, if possible. The testing did take time, however, and the reviews were often delayed.

The main reason for rejecting ads, as you will see from the analysis below, is that at least four "major" advertisers are required just to break even, postage rates being what they are. We think the rates proposed below will attract enough advertisers, especially since a one-time ad will have "multiple-exposure" in a newsletter in which back issues are regularly reread.

We feel that advertising conveys useful information to the using community, and is actually an added service, provided that the editorial content is not diluted, and that the product advertised is indeed worthwhile. We will therefore begin accepting advertising for publication with future issues. The rates and the analysis leading to these rates follows:

Our present mailings come right up to the two ounce limit; any additional material would require additional postage. Paying for one additional ounce of postage would permit the mailing of up to six additional sheets of printed matter. Our added mailing cost per issue would consist of domestic first class postage for one additional ounce ($0.17) to approximately 1000 subscribers, and overseas airmail/printed matter postage for one additional ounce ($0.36 - $0.46) to approximately 300 subscribers. This works out to about $300 whether we insert a half-sheet or six full sheets; never mind the cost of labor for the added folding and stuffing!

Beginning with Issue No. 11 (Vol. 3, No. 1) we will handle advertising on the following basis: We will accept "ready-to-stuff" 8 1/2 x 11 inch sheets of printed matter, and insert them in the mailing envelope with

SYM-PHYSIS, which will retain its present format and size. The rate will be $125 per sheet. We reserve the right to delay publication until multiples of four to six sheets may be mailed with each issue (if only four sheets of advertising material are available we further reserve the right to publish up to two more sheets, i. e., eight more pages, of SYM-PHYSIS, in their place!). We will also accept camera-ready 8 1/2 x 11 inch pages to be reduced to 5 1/2 x 8 1/2 inches and "batched" with three others on a single sheet, to be printed (by us) in SYM-PHYSIS format. The rate will be $50.00 per page. We reserve the right to delay publication until four such pages are available for batching.

We will still continue to publish PRODUCT RECOMMENDATIONS, where warranted, in the following format.

PRODUCT RECOMMENDATIONS

NOTE: There is neither enough time nor space left in this issue to do justice to the product lines of the following three sources, all of whom have enquired about advertising rates, sort of "forcing" us into making our decision re accepting advertising. It is very likely that one or more of them will be advertising their wares in the next issue, but if you need additional information before then, please contact them directly. Until the next issue, then, the following brief reviews are the best we can do:

COLUMBUS INSTRUMENTS

Columbus Instruments International Corporation, 900 N. Hague Ave., Columbus, OH 43204, (614) 488-6176 (Dr. Jan Czekajewski), is primarily a manufacturer of bio-medical instrumentation, using the AIM-65 in their products. Their so-called Universal AIM-65 Interface Card, with 16 channels of 12 bit (very high speed) A/D conversion, a battery backed-up calendar/clock, and a 16 K RAM/ROM expansion space, could just as well be called a Universal SYM-1 Interface Card! Only a single jumper need be added to the SYM-1 to fully utilize all of its capabilities. Even the BASIC program (written for the AIM) requires absolutely no modification for the SYM.

The calendar/clock, converters, and multiplexer are assigned addresses in the $9000 block. About the only difference is that the BASIC program will have to be keyed in "by hand" for the SYM, no big deal. We read their ads, asked to "borrow" a manual for review, and became dealers. There are at least two SYM users who are as impressed with the interface as we are, and we do not know of any other board as versatile.

R. J. BRACHMAN ASSOCIATES, INC.

R. J. Brachman Associates, Inc., P. O. Box 1077, Havertown, PA 19083-0077, (215) 622-5495 (Dr. Michael "Mike" Brachman), recently introduced the MICROsport MicroComputer (MMC) in advertisements in several of the computer magazines. They enquired about advertising in SYM-Physis, and we offered to review the product instead. Mike sent us samples of the entire product line and piles of documentation. We will get around to actual testing after the newsletter is out, and will let you know the results in the next issue. Part of the review loan agreement included the stipulation that we communicate our findings back to RJB prior to publication or release of data. That's fair enough.

We might mention in passing, however, that we have an upcoming application where we will be wanting to program a large number of 2732 EPROMs for the FDC-1s, and we were hoping that the MMC's EPROM Programming Adaptor (EPA) option would do the job. Unfortunately, while the EPA will program both TI 2516s and Intel 2716s with the software provided, and with minimal software changes will also program the TI

2532s, the Intel 2732s and the EPA are incompatible. If the EPA can be modified to handle 2732s, even at the cost of giving up the ability to handle 2716s, I'd buy that. Actually, we're ordering a complete system anyway; we want to try developing a stand-alone dedicated process control system. Pasted-up below are excerpts from the Product Description brochure:

The MICROsport MicroComputer (MMC), in addition to being a complete microcomputer on a 4½" x 6½" pc board, is the nucleus of a full hardware/software development system. Software can be developed for the MMC using any 6502-based computer such as the KIM-1, AIM-65, SYM, Apple II, PET, Ohio Scientific and others. The In-Circuit Emulator (ICE) permits full MMC software/hardware debugging,then adding the EPROM Programmer Adaptor, any single +5V EPROM such as the Intel-type 2716 or 2758 can be programmed without additional equipment or disconnecting the ICE/MMC. Of course any of these EPROM's can be programmed using other equipment and still operate in the MMC. The MMC is the ideal dedicated controller for use in control/monitor systems, laboratory experiments, timing, intelligent interfaces, security systems, and other applications requiring a low cost controller.

Features:

CPU.....MPS6503, operating at 1 MHz.
RAM.....1 Kbytes, static (MPS2114).
EPROM....Socket for Intel-type 2716 or 2758.

I/O . .2 MPS6522 VIAs for a total of:
32 input/output lines,
8 edge detector/control lines,
4-16 bit timer/counter/pulse generators,
2 shift registers, interrupt flag registers, and more.
8 user-defined pins on 44 pin edge connector.

Serial.....20 mA loop circuitry on-board.
Interrupts...Power-on and manual reset, non-maskable and maskable interrupts.
Power .....+5V regulated or 9 - 20V unregulated AC or DC.
Also.....All ICs socketed for easy maintenance.
LED power on indicator.
Adaptor socket for expanded memory functions such as CMOS-RAM w/battery back-up.

Options....In-Circuit Emulator; EPROM Programming Adaptor; MMC Development Model MMC/03D with zero insertion force sockets (3); EPROM Programming Services; and application software development.

Prices.....MMC/03D Development unit $149.00
MMC/03A Application unit $119.00
MMC/03S Complete development system including MMC/03D; MMC/03ICE; MMC/03EPA; and software
Kits available from $89.00.

CGS MICROTECH

CGRS Microtech, P. O. Box 102, Langhorne, PA 19047, (215) 757-0284 (Joseph T. Swope), sent us detailed information and operating manuals for the 6502PDS PETDISK Disk Operating System. This system can be adapted to work with the SYM and would be worth investigating by SYM owners who have access to PETs, or would like to be able to swap software with PET owners. It is a quite good operating system.

MISCELLANIA

JOHN R. MC DANIEL, 5557E Homestead Drive, Columbus, IN 47201, would like to communicate with others working with the Votrax SC-01 speech synthesis chip.

JEFF LAVIN, P. O. Box 1019, Whittier, CA 90609, sent us a batch of excellent CAI (Computer Assisted Instruction) programs in BASIC (we'll try to get at least one of them printed in Issue No. 11). We discussed the idea of forming SIGs (Special Interest Groups) and he has offered to prepare the questionnaire for mailing with Issue No. 11, and to help process the returns. The idea is that if you want to get in touch with SYMmers living, say, in Ohio, who are using the ColorMate for generating animated cartoons, we can sort through the Special Interest DataBank and put you in touch with them. Jeff has prepared an excellent "Selectric/Microcomputer Interface Manual", printed on the IBM Selectric

he has interfaced to his SYM. This manual is the definitive source on the subject. Copies may be ordered from him for $10.00. Add $2.00 for postage North America, $4.00 elsewhere.

JOHN BLALOCK, Blalock & Associates, P. O. Box 39356, Phoenix, AZ 85069, has been working on some really elegant hardware and software items which will become part of his expanding product line. One software product for which many SYMmers have been longing is now ready for release. This is his BAS-RAE/RAE-BAS transfer program, which allows BAS-1 programs to be written and edited in RAE-1! Drop him a self-addressed, stamped envelope for pricing information on this, and his other fine products.

SOFTWARE DATABANK

The following new programs, reviewed in Issue No. 9 are now available on cassette (RAE-1 Source Code format) with just enough hard copy documentation to get you started and over possible hurdles:

Kwok's Cross Reference Lister, XRF-1 95% "CERTIFIED"
Thuring's Structured Assembler Macros, MAC65 95% "CERTIFIED"

A new disassembler program is now available which provides a table of definitions for all external addresses, and automatically generates .CT and cassette dump for extra long source codes, is now available. We have tested all but the .CT feature (we must first modify to permit dump and continue on disk). If you liked Hissink's DISARAE, you will find this program even more loveable! Minimal hard copy, but beautifully commented source code, on cassette.

Dessaintes' Disassembler, DESDIS-1 95% "CERTIFIED"

Price for each of the above programs, and for all future programs of similar utility, size, and complexity, is $36.00 postpaid anywhere.

Not yet ready, with prices not yet set, but perhaps by next issue, are:

Holt's TECO, a "free-standing" Text Editor and Word Processor, fully compatible with DEC's PDP-11 version.
Kwok's BASIC WORD PROCESSOR (BWP-1), a BAS-1 based word processor, for those without RAE-1, who cannot use SWP-1.

Jack Brown has authorized us to "close-out" the present 16K SYM-FORTH package at a $90.00 price (Object Code Cassette, Reference Manual, and Source Code Listing). Purchasers of this package can then obtain from Saturn Software the expanded 79-Standard Version (requiring at least 24K of RAM) for an additional $50.00.

Bob Peck has authorized us to reduce the price of the FBOK Appendix Cassette to $10.00. Because of its popularity, we will be stocking the First Book of KIM, a "must" for unexpanded SYMs; the games are really great if you have youngsters around. Price of the FBOK is $11.75 postpaid US/Canada. Overseas add $3.00 for SURFACE mailing (weight is one pound). You also will need Peck's FBOK Appendix and/or the Appendix Cassette.

FINI!

That's it for now. We already have more than enough material and ideas on hand for Issue No. 11; actually this could have been a "double" issue! We'll now "rest" a day or two, working out the details of our South Pacific trip. Next, we will spend about two weeks answering all of your accumulated letters, seeing what is on the pile of cassettes and disks still waiting for us, and rearranging the clutter of papers piled high around each SYM. Then, we'll start working on Vol. 3, No. 1. You can expect the Jan/Feb/Mar issue to be in your hands by early April!