-Punch along this edge for insertion into a binder-

THE USERS' GROUP OFFICE HOURS (INCLUDING TELEPHONE SERVICE)

ARE M-W-F 9:00 AM to 2:00 PM Pacific Time

SYM-PHYSIS

THE SYM USERS' GROUP NEWSLETTER

VOLUME IV, NUMBER 1 (ISSUE NO. 15) - SPRING 1983 (JAN/FEB/MAR/APR)

SYM-PHYSIS is a thrice yearly publication of the SYM Users' Group, P. O. Box 319, Chico, CA 95927. SYM-PHYSIS and the SYM Users' Group (SUG) are in no way associated with Synertek Systems Corporation (SSC), and SSC has no responsibility for the contents of SYM-PHYSIS. SYM is a registered trademark of SSC. SYM-PHYSIS, from the Greek, means the state of growing together, to make grow, to bring forth.

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

Editor/Publisher: *H. R. "Lux" Luxenberg*
Business/Circulation: *Jean Luxenberg*
Office Staff: *Joyce Arnovick, Denny Hall*

SUBSCRIPTION RATES: (Volume IV, 1983, Issues 15 - 17)

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

BACK ISSUES ARE AVAILABLE AS FOLLOWS:

Issues 0 through 6 (Volume I, 1979/80), are available for $12.00, US/Canada, and $16.00, First Class/Airmail, elsewhere.

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

Issues 11 through 14 (Volume III, 1982), are available for $10.50, US/Canada, and $14.00, First Class/Airmail, elsewhere.

AN OUTSTANDING OFFER TO THE SYM COMMUNITY

Our lead article (starting in the page 2 "slot") is the first of a series of guest columns by Jeff Lavin. Jeff is a relative newcomer to the microcomputer field. He got a SYM-1 as a learning tool, called us a few times when he had questions, then called us many times to give us answers to questions we had often wondered about ourselves, but had neither the time or expertise to answer for ourself.

Jeff is a prolific writer and highly inventive. He has come up with some extremely ingenious hardware and software for the SYM-1 (and other systems as well). Some of these have been described in previous issues. Several others, notably SYM/ELIZA, a truly efficient and versatile EPROM Programmer, and a DUAL ACIA Board, will be described in the NEW PRODUCTS section.

Jeff has two very useful software packages he wishes to contribute to the SYM user community. These are BASIC TERMINAL CONTROL PATCH and RAE TERMINAL CONTROL PATCH, both based largely on material published in earlier issues of SYM-PHYSIS, but greatly enhanced by him. We have tried them both, and they vastly improve the SYM's human interface, making it truly pleasurable to use. Both patches include FDC-1 links.

He will provide complete RAE source code for both, on either cassette or FDC-1 5 1/4" double density diskette (please state which!), to all who wish copies, asking only a nominal $10.00 to cover media, handling, and shipping charges. This is an offer you shouldn't refuse!

ADDRESS DECODING, POR, and the SUPER SYM

By Jeff Lavin - January 1983
P.O. Box 1019
Whittier, CA 90609

The purpose of this article is to explain how the SYM-1 uses partial address decoding to select different devices, and how the Power-on-reset (POR) circuitry operates. In the next installment this concept will be expanded to show how the SYM may be converted to the type of machine Lux described in the #13/14 issue of SYM-PHYSIS. If there are any topics the reader would like to see covered in this column, drop a line to Lux or myself.

When power is first applied to the SYM, a 555 timer, connected as a one-shot, applies a reset pulse to the processor and all the I/O. The I/O is left in a known condition; the processor must be initialized, however. Quoting from the MOS Technology Programming Manual:

> "... the only automatic operations of the microprocessor during reset are to turn on the interrupt disable bit and to force the program counter to the vector location specified in locations FFFC and FFFD and to load the first instruction from that location."

In the preceeding remark, the locations being referred to are called VECTORS. A vector consists of two consecutive memory locations containing the address of a routine in the format ADL - ADH, or three locations with the first being a JMP ($4C). Vectors are responsible for the power and flexibility of the SYM. If the ROM or EPROM containing the RESET routine were to be located at the top of memory, the vectors would be cast in silicon. In order to preserve the usefulness of being able to point vectors to new routines, the vectors in the ROM would need to point to vectors in RAM. At two bytes per vector, this would waste a lot of memory. There are other ways to accomplish this, however.

The ROM containing the RESET routine and RESET vector may be called at the top of memory on power-up or user reset, and later be replaced by RAM. This manipulation is the function of the POR circuitry.

Refering to the diagram, a NAND gate (U8) creates the signal $\overline{POR}$. The two inputs to U8 are $\overline{F8}$ and CA2 from VIA #1. The RESET signal causes CA2 to be HIGH. The inclusion of $\overline{POR}$ into U7, causes the 74LS145 decoder to select an output higher than #7 (U10 and U11 are actually BCD to DECIMAL decoders). The result is that $\overline{POR}$ is made LOW. U24 is an AND gate that controls the $\overline{CS}$ for the ROM, U20. An interesting point is that an AND gate acts as an OR gate for NEGATIVE LOGIC. I.E.: If both A AND B = Y, then $\overline{A}$ OR $\overline{B}$ = Y. Therefore, if either input of U24 is low, the ROM is selected. This causes the ROM to be selected at the RESET vector. The purpose of including $\overline{POR}$ in the address decoding done by U11 is to keep SYSTEM RAM from being addressed. Since the SYM uses only outputs 0 - 7 of U11, anything addressed at C000 or higher will not be selected while $\overline{POR}$ is active.

After the ROM is selected, the processor forces the program counter to the address contained in the RESET vector, and loads the instruction found there. In the case of the SYM, when the ROM is selected at F000 instead of 8000, the two bytes normally addressed at 8FFC and 8FFD appear at FFFC and FFFD instead. The address of the reset routine is stored here, and the processor begins executing instuctions at 8B4A. Note that now, the ROM is being addressed at its normal location.

The first thing that happens in the reset routine after the stack and flag register is initialized, is to turn off $\overline{POR}$.

```
8B4A A2 FF      RESET  LDX #$FF     Initialize stack
8B4C 9A                TXS           point to $1FF
8B4D A9 CC      POR    LDA #$CC     (%11001100)
8B4F 8D 0C A0          STA PCR1     Disable POR, tape off
```

Bits 3 - 1 control CA2. Loading #$CC into the Peripheral Control Register (PCR, VIA #1; hence PCR1) utilizes the bits inside the brackets to control CA2: %1100[110]0. The VIA programming card states that this combination of bits sets CA2 LOW. This disables $\overline{POR}$ and returns normal addressing. This concludes discussion of the POR circuitry.

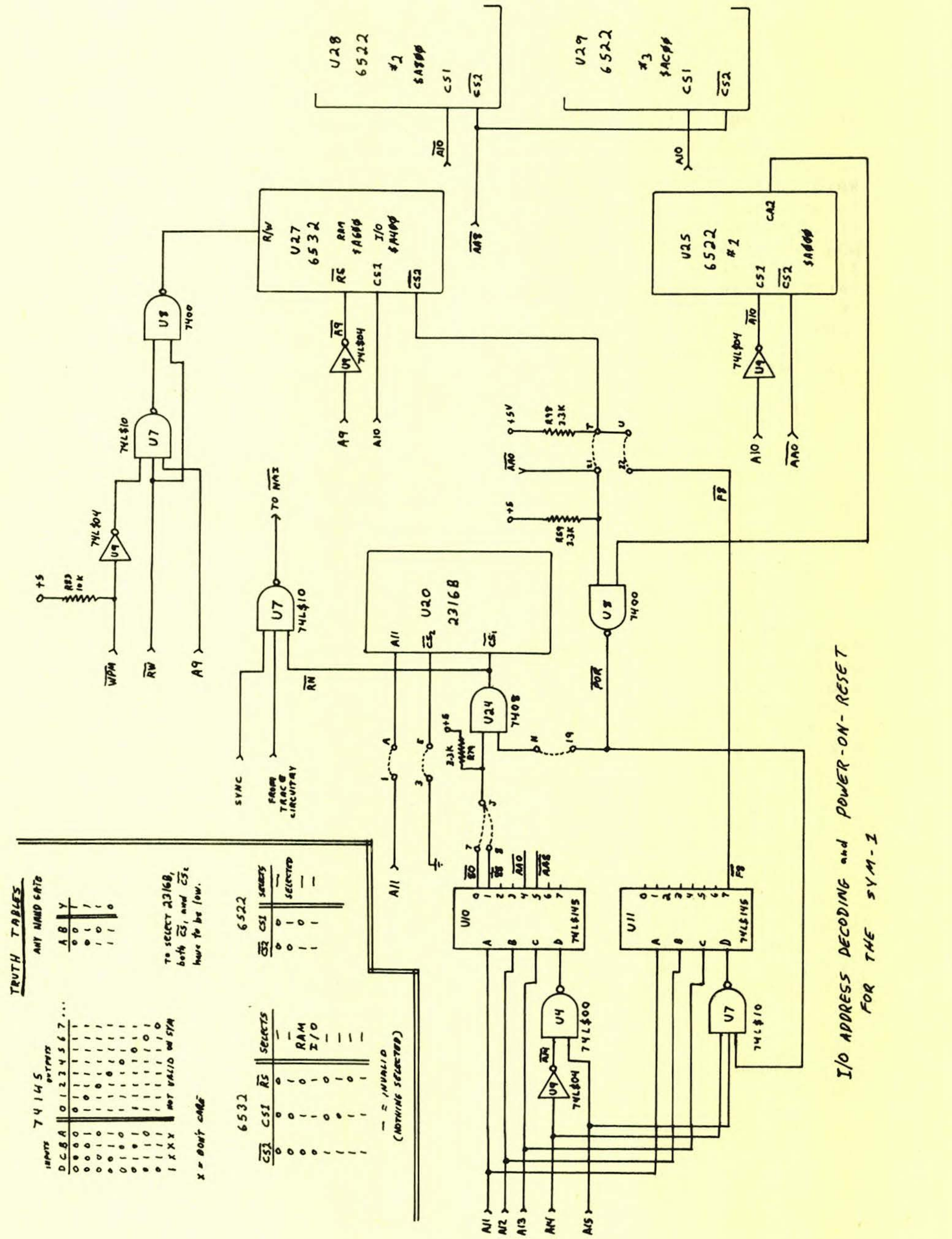
Note that the same signal used to select the ROM also is connected to U7. This insures that when the TRACE function is used, it will not operate in the Monitor and cause the system to crash.

The address decoding for the I/O on the SYM is straightforward. Full decoding is not used. The 6532 and 6522 #1 are selected on their $\overline{CS2}$ inputs by $\overline{AA0}$, giving them 2K of address space (A000 - A7FF). For the RIOT, this is modified by CS1 being selected only when A10 is high, giving an address range of A400 - A7FF. The $\overline{RS}$ input is used to select RAM (L) or I/O (H). Connected to $\overline{A9}$, I/O is selected at A400 - A5FF, and RAM at A600 - A7FF. For the VIA, the CS1 input is connected to $\overline{A10}$, selecting this chip only when A10 is low (A000 - A3FF).

The $\overline{CS2}$ inputs of VIAs #2 and #3 are both selected by $\overline{AA8}$, also providing a 2K address space (A800 - AFFF). This is divided equally by A10. CS1 of VIA #2 is connected to $\overline{A10}$, giving the lower 1K, and A10 gives VIA #3 the upper 1K.

Earlier it was mentioned that SYSTEM RAM is selected at the top of memory, where the reset vector lives, but here it is stated that 6532 RAM is addressed at A600. How can this be? Well, SYSTEM RAM is actually addressed in both places. This is the famous ECHO. System RAM, located at A600 - A6FF is ECHOED at F800 - FFFF. This is accomplished by tying $\overline{AA0}$ to $\overline{F8}$ (Jumper T-21 and U-22). Anything selected by one is also enabled by the other. You may note that this wastes a lot of memory also. There is yet another way to provide vectors on power-up AND not use pointers to RAM, AND not waste memory!

In the next column, I will discuss modifying the things covered above in order to create a really useful 6502 based computer with 56K of contiguous RAM!!! Till then, happy computing.



I/O ADDRESS DECODING and POWER-ON-RESET FOR THE SYM-1

Dr. H.R. Luxenberg
SYM Users' Group
P.O. Box 319
Chico, CA 95927

January 25, 1983

Dear Lux,

A few weeks ago, I considered the problem of how to implement a CHAIN command on the SYM-1. The procedure I came up with is based on the power-on-reset to BASIC in Issue No. 10. It was only necessary to investigate some problems associated with USR calls to different points in the machine language program and with the transfer of data between program segments.

In your answer to a letter in Issue No. 13/14, you state that CHAIN is available as part of Jack Brown's enhancements. Hopefully, my method can be presented as a simple alternative.

The enclosed material consists of a printout of the assembly of the required machine language program, a printout of four short BASIC programs used to illustrate the procedure, and a short article explaining the problems, solutions, and constraints.

All material is duplicated on the cassette. First, the RAE file containing the source code is recorded twice, as F1 and F2. Next, the four BASIC programs are saved as A,B,C, and D. Finally, the RAE file containing the text of the article (with appropriate SWP-1 commands) is recorded as F3.

I have also enclosed a self-addressed stamped envelope for any reply you would care to make. Should you not consider the material suitable for publication, there is no need to return either the hard copy or the cassette. I am hoping that you will be able to use it.

Sincerely yours,

Ed

Dr. Edward Wysocki
P.O. Box 6257
Baltimore, MD 21206

CONTROLLED LOAD OF BASIC PROGRAMS FROM TAPE

Dr. Edward Wysocki
P.O. Box 6257
Baltimore, MD 21206

There may be times when you want to run a BASIC program which is too large to fit in your computer's memory. Some computer systems make use of a command called CHAIN , in which the segments of a large program can be automatically loaded into memory one after another. Such a provision appears to be lacking in the SYM.

The short machine language program presented here permits you to use the closing commands in one BASIC program to cause the loading of a new program. Of course, you must have computer control of the cassette recorder and it must be in PLAY. The program is adapted from the one in Issue No. 10 which permits power-on-reset into BASIC or a running BASIC program.

In each BASIC program, one uses the POKE to change the A in the ASCII string 'LOAD A' to the name of the program segment to be loaded. The subroutine to be called the first time is REPLY1. For each program segment that follows, use REPLY1 or REPLY2 according to a simple rule. If the present segment has involved any input from the keyboard, use REPLY1; otherwise use REPLY2.

The only problem which exists is the transfer of data from one program segment to the next. When the SYM executes a LOAD, it first performs a NEW. But the NEW does not cause the actual erasure of the program or variables. It only resets the pointers in locations $7B through $88 (See 'A Deductive Story', Issue No. 7). If the pointers are reset at the beginning of the program segment just loaded, the variables may be accessed.

There are rules to be followed in the transfer of data from one program to another:

1. The first program segment must be the longest.

2. All variables must be dimensioned and defined in the first segment, possibly with dummy values.

3. Simple strings, those defined by a pair of quotes, cannot be passed between segments.

4. Computed strings, those created by operations on other strings or by an INPUT, can be passed between program segments.

If the first two rules are not followed, the storage area for variables will be disturbed. One cannot pass simple strings since the pointer to such a string points back to the program storage area. With a new program segment there, you will get some characters, but not what you expected. Computed strings are stored eleswhere.

The transfer of data may be halted in any segment by not resetting the pointers at its start. Data transfer may be restarted in any segment which follows. In such a case, the rules regarding segment length and definition of variables apply to the new starting segment.

The four sample BASIC programs should be placed on tape as A, B, C, and D. When used with the machine language program, they should illustrate most of the principles involved. If you relocate the program, don't forget to change the POKE of the program name as well as the USR calls to REPLY1 and REPLY2.

```
               0010 ; CONTROLLED LOAD OF A BASIC PROGRAM BY
               0020 ; ANOTHER BASIC PROGRAM
               0030 ; EDWARD WYSOCKI - JANUARY 1,1983
               0040 ;
               0050 ;
               0060 RIN          .DE $887E
               0070 INVEC        .DE $A660
               0080 SCRA         .DE $A63A
               0090 TOUTFL       .DE $A654
               0100              .BA $1E00
               0110              .OS
1E00- AD 62 A6 0120 REPLY1       LDA INVEC+2
1E03- 8D 3B A6 0130              STA SCRA+1
1E06- AD 61 A6 0140              LDA INVEC+1
1E09- 8D 3A A6 0150              STA SCRA
1E0C- A9 7E    0160              LDA #L,RIN
1E0E- 8D 61 A6 0170              STA INVEC+1
1E11- A9 88    0180              LDA #H,RIN
1E13- 8D 62 A6 0190              STA INVEC+2
1E16- A9 28    0200 REPLY2       LDA #L,EXEC
1E18- 8D FA 00 0210              STA $FA
```

```
1E1B- A9 1E      0220            LDA #H,EXEC
1E1D- 8D FB 00   0230            STA $FB
1E20- A9 00      0240            LDA #$00
1E22- 8D 54 A6   0250            STA TOUTFL
1E25- 4C 4C D1   0260            JMP $D14C
1E28- 4C 4F 41   0270 EXEC       .BY 'LOAD A' $0D
1E2B- 44 20 41
1E2E- 0D
1E2F- 30 50 4F   0280            .BY '0POKE 42580,144' $0D
1E32- 4B 45 20
1E35- 34 32 35
1E38- 38 30 2C
1E3B- 31 34 34
1E3E- 0D
1E3F- 52 55 4E   0290            .BY 'RUN' $0D
1E42- 0D
1E43- 00         0300            .BY $00
                 0310            .EN
```

```
 10 REM PROGRAM TO START LOAD SEQUENCE
 20 X=0
 30 DIM A(5)
 40 B=6666
 50 C$="ABCDEF":D$="12345"
 60 E$=C$+D$
 70 FOR I=1 TO 5
 80 INPUT A(I)
 90 NEXT I
 100 REM SAVE VARIABLE POINTERS
 110 POKE 8000,PEEK(125):POKE 8001,PEEK(126)
 120 POKE 8002,PEEK(127):POKE 8003,PEEK(128)
 130 POKE 8004,PEEK(129):POKE 8005,PEEK(130)
 140 POKE 8006,PEEK(131):POKE 8007,PEEK(132)
 150 POKE 8008,PEEK(133):POKE 8009,PEEK(134)
 160 REM POKE PROGRAM NAME
 170 POKE 7725,66
 180 X=USR(&"1E00",&"0000")
 190 END
OK

LOADED
OK


 10 REM FIRST AUTO LOAD
 20 REM RESTORE
 30 POKE 125,PEEK(8000):POKE 126,PEEK(8001)
 40 POKE 127,PEEK(8002):POKE 128,PEEK(8003)
 50 POKE 129,PEEK(8004):POKE 130,PEEK(8005)
 60 POKE 131,PEEK(8006):POKE 132,PEEK(8007)
 70 POKE 133,PEEK(8008):POKE 134,PEEK(8009)
 80 REM OUTPUT
 90 FOR I=1 TO 5
 100 PRINT A(I)
 110 NEXT I
 120 PRINT B
 130 PRINT C$,D$,E$
 140 POKE 7725,67
 150 X=USR(&"1E16",&"0000")
 160 END
OK
```



```
LOADED
OK


 10 REM SECOND AUTO LOAD
 20 INPUT A$
 30 POKE 7725,68
 40 X=USR(&"1E00",&"0000")
 50 END
OK

LOADED
OK


 10 REM THIRD AND FINAL AUTO LOAD
 20 PRINT "SO YOU SEE,IT WORKS!!"
 30 END
OK
```

NEW CMOS 65XX CPUS

The following information originally appeared in the February 1983 issue of UPDATE ANNOUNCEMENTS, a monthly publication of the Professional Update Committee of the IEEE Philadelphia Section, and is reprinted here, with permission, for your general information.

Do any of our readers have "hands-on" experience with some of these new chips which they would like to share with others?

* * * * * * * * * * * * * * * * * * *

MICROCOMPUTER PRODUCTS OF INTEREST

1. From GTE Microcuits:

GTE is making CMOS versions of 6500 CPUs. They are making 18 CPUs, ten that are pin compatiable with NMOS CPUs and eight that are new. The CMOS 6500 CPUs have ten new instructions and two new addressing modes. The CPUs do not have the glitches the NMOS CPUs have, for example invalid op codes cause the NMOS CPUs to hang up while the CMOS CPUs treat them as NOPs. The new CPUs which are pin compatible with the NMOS CPUs are:

G65SC02	G65SC04	G65SC06	G65SC12	G65SC14
G65SC03	G65SC05	G65SC07	G65SC13	G65SC15

The new CMOS CPUs which have DMA and multi-processor interfaces are:

G65SC102	G65SC104	G65SC106	G65SC112
G65SC103	G65SC105	G65SC107	G65SC115

For more information please contact:

SEE ALSO PAGES 15-33,34 FOR ADDITIONAL INFORMATION

Harry Nash Associates
P.O. Box 188
Willow Grove, PA 19090
(215) 657-2213

* * * * * * * * * * * * * *

```
           0010           ;    EXPANDED MEMORY SEARCH FOR SYM-1
           0020           ;       By Richard R. Albers c 1981
           0030           ;
           0040           ; To use this program, enter: G (LINK)CR.
           0050           ; Then enter: F (start addr)-(end addr)CR.
           0060           ; The program will prompt for the data to be
           0070           ;  matched before entering the memory examine/
           0080           ;  modify mode.  Enter hex bytes, or ":" plus
           0090           ;  ASCII characters, or combinations.
           0100           ; Enter 1 to 255 bytes.  End input with a <CR>.
           0110           ; The program acts like .M with 3 parameters,
           0120           ;  but needs a match of all the pattern bytes
           0130           ;  before entering memory examine/modify mode.
           0140
           0150           .BA $0200
           0160 ; .OS
           0170 ACCESS    .DE $8B86
           0180 CRLF      .DE $834D
           0190 INCCMP    .DE $82B2
           0200 INBYTE    .DE $81D9
           0210 NEWLOC    .DE $8517
           0220 OUTCHR    .DE $8A47
           0230 P2SCR     .DE $829C
           0240 PARNR     .DE $A649
           0250 SPACE     .DE $8342
           0260 URCVEC    .DE $A66C
           0270
0200- 20 86 8B  0280 LINK      JSR ACCESS   Link to unrec. cmd vector
0203- A9 0E     0290           LDA #L,START
0205- 8D 6D A6  0300           STA URCVEC+1
0208- A9 02     0310           LDA #H,START
020A- 8D 6E A6  0320           STA URCVEC+2
020D- 60        0330           RTS
                0340
020E- C9 46     0350 START     CMP #'F      Our command?
0210- D0 51     0360           BNE ERROR
0212- AD 49 A6  0370           LDA PARNR
0215- C9 02     0380           CMP #$02
0217- D0 4A     0390           BNE ERROR
                0400
0219- A0 00     0410           LDY #$00     Clear index register
021B- 20 4D 83  0420           JSR CRLF     Clear line
021E- B9 65 02  0430 OUTP      LDA TABL,Y   Print "PAT "
0221- F0 06     0440           BEQ GETPAT
0223- 20 47 8A  0450           JSR OUTCHR
0226- C8        0460           INY
0227- D0 F5     0470           BNE OUTP
                0480
0229- A0 00     0490 GETPAT    LDY #$00     Clear index register
022B- 20 D9 81  0500 GETP      JSR INBYTE   Get a pattern byte
022E- F0 0B     0510           BEQ STOLNG   CR ends input
0230- B0 31     0520           BCS ERROR    Non-hex not allowed
0232- 99 6B 02  0530           STA PATRN,Y  Store pattern
0235- 20 42 83  0540           JSR SPACE    Separate bytes
0238- C8        0550           INY          Count bytes
0239- D0 F0     0560           BNE GETP     Force end at 256
                0570
023B- 8C 6A 02  0580 STOLNG    STY NBYTES
023E- 20 86 8B  0590           JSR ACCESS
0241- 20 9C 82  0600           JSR P2SCR    Move P2 to FE,FF
                0610
0244- A0 00     0620 COMP1     LDY #$00     Clear index register
0246- B1 FE     0630 COMP2     LDA ($FE),Y  Get a byte
0248- D9 6B 02  0640           CMP PATRN,Y  And compare to pattern
024B- D0 0F     0650           BNE NEXT     No match
```



```
024D- C8        0660           INY          Match
024E- CC 6A 02  0670           CPY NBYTES   End of pattern?
0251- 90 F3     0680           BCC COMP2    No; continue matching
                0690
0253- 20 17 85  0700           JSR NEWLOC   Enter mem examine/modify mode
0256- 90 09     0710           BCC QUIT     CR means return to SUPERMON
0258- C9 47     0720           CMP #'G      Go to next matched locn?
025A- D0 07     0730           BNE ERROR    Only "CR" or "G" allowed
                0740
025C- 20 B2 82  0750 NEXT      JSR INCCMP   Increment FE,FF
025F- 90 E3     0760           BCC COMP1    Not at end address yet
0261- 18        0770 QUIT      CLC          End of search
0262- 60        0780           RTS
                0790
0263- 38        0800 ERROR     SEC          Error; return
0264- 60        0810 BACK      RTS          ReTurn to SUPERMON
0265- 50 41 54  0820 TABL      .BY 'PAT ' $00
0268- 20 00
026A-           0830 NBYTES    .DS 1
026B-           0840 PATRN     .DS 255
                0850           .EN
```

EDITOR'S NOTE: The Expanded Memory Search Program listed below is more "powerful" than that listed above in that "wild cards" are allowed. This could be helpful, for example, in finding all JSRs to a given page.

```
           0010           ;    EXPANDED MEMORY SEARCH FOR SYM-1
           0020           ;       By Richard R. Albers c 1983
           0030           ;
           0040           ; To use this program, enter: G (LINK)CR
           0050           ; Then enter: F (start addr)-(end addr)CR.
           0060           ; The program will prompt for the data to be
           0070           ;  matched before entering the memory examine/
           0080           ;  modify mode.  Enter hex bytes, or ":" and
           0090           ;  an ASCII character, or "?" as a wild card.
           0095           ;  (a thru z ASCII is made upper case by MON).
           0100           ; Enter 1 to 255 bytes (wild cards = 2 bytes).
           0101           ; End input with a <CR>.
           0110           ; The program acts like .M with 3 parameters,
           0120           ;  but needs a match of all the pattern bytes
           0125           ;  (a wild card is a guaranteed match)
           0130           ;  before entering memory examine/modify mode.
           0140
           0150           .BA $0200
           0160 ; .OS
           0170 ACCESS    .DE $8B86
           0180 CRLF      .DE $834D
           0190 INCCMP    .DE $82B2
           0200 INBYTE    .DE $81D9
           0210 NEWLOC    .DE $8517
           0220 OUTCHR    .DE $8A47
           0230 P2SCR     .DE $829C
           0240 PARNR     .DE $A649
           0250 SPACE     .DE $8342
           0260 URCVEC    .DE $A66C
           0270
0200- 20 86 8B  0280 LINK      JSR ACCESS   Link to unrec. cmd vector
0203- A9 0E     0290           LDA #L,START
0205- 8D 6D A6  0300           STA URCVEC+1
0208- A9 02     0310           LDA #H,START
020A- 8D 6E A6  0320           STA URCVEC+2
020D- 60        0330           RTS
                0340
020E- C9 46     0350 START     CMP #'F      Our command?
```

```
0210- D0 76     0360          BNE ERROR
0212- AD 49 A6  0370          LDA PARNR
0215- C9 02     0380          CMP #$02
0217- D0 6F     0390          BNE ERROR
                0400
0219- A0 00     0410          LDY #$00      Clear index register
021B- 20 4D 83  0420          JSR CRLF      Clear line
021E- B9 8A 02  0430 OUTP     LDA TABL,Y    Print "PAT "
0221- F0 16     0440          BEQ GETPAT
0223- 20 47 8A  0450          JSR OUTCHR
0226- C8        0460          INY
0227- D0 F5     0470          BNE OUTP      (Always)
                0480
0229- C9 3F     0490 GETW     CMP #'?       Wild card?
022B- D0 5B     0500          BNE ERROR
022D- A9 00     0510          LDA #$00      Yes, indicate it
022F- 99 90 02  0520          STA PATRN,Y
0232- C8        0530          INY
0233- F0 53     0540          BEQ ERROR     If PATRN too big
0235- A9 01     0550          LDA #$01
0237- D0 09     0560          BNE STO1      (Always)
                0570
0239- A0 00     0580 GETPAT   LDY #$00      Clear index register
023B- 20 D9 81  0590 GETP     JSR INBYTE    Get a pattern byte
023E- F0 15     0600          BEQ STOLNG    CR ends input
0240- B0 E7     0610          BCS GETW      Non-hex maybe wild
0242- 99 90 02  0620 STO1     STA PATRN,Y   Store pattern
0245- C9 00     0630          CMP #$00      Zero is special
0247- D0 06     0640          BNE GET1
0249- C8        0650          INY
024A- F0 3C     0660          BEQ ERROR     If PATRN too big
024C- 99 90 02  0670          STA PATRN,Y   Double zero matches zero only
024F- 20 42 83  0680 GET1     JSR SPACE     Separate bytes on CRT
0252- C8        0690          INY           Count bytes
0253- D0 E6     0700          BNE GETP      Force end at 256
                0710
0255- 8C 8F 02  0720 STOLNG   STY NBYTES    Store number of bytes
0258- 20 86 8B  0730          JSR ACCESS     in PATRN
025B- 20 9C 82  0740          JSR P2SCR     Move P2 to FE,FF
                0750
025E- A0 00     0760 COMP1    LDY #$00      Clear index registers
0260- A2 00     0770          LDX #$00
0262- BD 90 02  0780 COMP2    LDA PATRN,X   Get a byte of pattern
0265- D0 06     0790          BNE COMP3
0267- E8        0800          INX           Check for wild card
0268- BD 90 02  0810          LDA PATRN,X
026B- D0 04     0830          BNE COMP4     Wild, skip match attempt
026D- D1 FE     0850 COMP3    CMP ($FE),Y   Compare to memory
026F- D0 10     0860          BNE NEXT      No match
0271- E8        0870 COMP4    INX
0272- C8        0880          INY           Match
0273- EC 8F 02  0890          CPX NBYTES    End of pattern?
0276- 90 EA     0900          BCC COMP2     No; continue matching
                0910
0278- 20 17 85  0920          JSR NEWLOC    Yes; examine memory
027B- 90 09     0930          BCC QUIT      CR means return to SUPERMON
027D- C9 47     0940          CMP #'G       Go to next matched locn?
027F- D0 07     0950          BNE ERROR     Only "CR" or "G" allowed
                0960
0281- 20 B2 82  0970 NEXT     JSR INCCMP    Increment FE,FF
0284- 90 D8     0980          BCC COMP1     Not at end address yet
0286- 18        0990 QUIT     CLC           End of search
0287- 60        1000          RTS
                1010
0288- 38        1020 ERROR    SEC           Error; return
```



```
0289- 60        1030 BACK     RTS           Return to SUPERMON
                1040
028A- 50 41 54  1050 TABL     .BY 'PAT ' $00
028D- 20 00
                1060
028F-           1070 NBYTES   .DS 1
0290-           1080 PATRN    .DS 255       May be less; don't overflow!
                1090          .EN
                0010          ;  HEX TO DEC & DEC TO HEX CONVERTER
                0015          ;    By Richard Albers
                0020
                0040          ;     Uses modifications of routines
                0050          ;     from Leo J. Scanlon's
                0060          ;    "6502 Software Design".
                0070
                0080          ;    .G 200 to start a conversion.
                0090          ;    Prefix hex with "$" ("+" from hexpad).
                0100          ;      Input decimal with no prefix.
                0110          ;    Output uses same hex indicator.
                0120          ;    Limit is $FFFF or decimal 65535.
                0130          ;    Test for overflow is only on decimal
                0140          ;     input; hex input uses PARM.
                0150          ;    End each number input with "CR".
                0160          ;    Exit to MON with "M" (MEM from hexpad)
                0170          ;     after prompt (?) for input.
                0180          ;    See coments for changes for use with
                0190          ;     hex keypad & LEDs.
                0200
                0210 ACCESS   .DE $8B86
                0220 BEEP     .DE $8972
                0230 CRLF     .DE $834D
                0240 ERMSG    .DE $8171
                0250 INCHR    .DE $8A1B
                0260 OUTBYT   .DE $82FA
                0270 OUTCHR   .DE $8A47
                0280 OUTQM    .DE $8320
                0290 PARM     .DE $8220
                0300 P3L      .DE $A64A
                0310 P3H      .DE $A64B
                0320 SPACE    .DE $8342
                0330 WARM     .DE $8003
                0340
                0350          .BA $200
                0360 ; .OS
                0370
0200- 20 86 8B  0380 START    JSR ACCESS
0203- D8        0390          CLD           Just in case ...
0204- A9 00     0400 RESTRT   LDA #$00      Clear storage & flag
0206- 8D 4A A6  0410          STA P3L
0209- 8D 4B A6  0420          STA P3H
020C- 8D FE 02  0430          STA ZFLAG
020F- 20 4D 83  0440          JSR CRLF      Indicate ready
0212- 20 20 83  0450          JSR OUTQM     Prompt : ?
0215- 20 42 83  0460          JSR SPACE
0218- 20 1B 8A  0470          JSR INCHR     Get first char
021B- C9 24     0480          CMP #'$       Hex to dec? (+ ($2B) for hexpad)
021D- F0 38     0490          BEQ H2D       Yes
021F- C9 4D     0500          CMP #'M       Return to MON?
0221- D0 03     0510          BNE D2H       No, must be dec to hex
0223- 4C 03 80  0520          JMP WARM      Yes, return
                0530
                0540          ; Convert decimal to hexadecimal
                0550
0226- C9 30     0560 D2H      CMP #'0       Test for valid decimal digit
0228- 90 23     0570          BCC ERROR
```

```
Ø22A- C9 3A    Ø58Ø          CMP #':       (ASCII "9"+1)
Ø22C- BØ 1F    Ø59Ø          BCS ERROR
Ø22E- 2Ø 65 Ø2 Ø6ØØ          JSR CONA2H    OK; convert to hex
Ø231- 2Ø 1B 8A Ø61Ø          JSR INCHR     Get next character
Ø234- DØ FØ    Ø62Ø          BNE D2H       Not CR; continue
Ø236- 2Ø 4D 83 Ø63Ø          JSR CRLF
Ø239- A9 24    Ø64Ø          LDA #'$       Print "$"("+"($2B) for LEDs)
Ø23B- 2Ø 47 8A Ø65Ø          JSR OUTCHR
Ø23E- AD 4B A6 Ø66Ø          LDA P3H
Ø241- 2Ø FA 82 Ø67Ø          JSR OUTBYT    Output hex value
Ø244- AD 4A A6 Ø68Ø          LDA P3L
Ø247- 2Ø FA 82 Ø69Ø          JSR OUTBYT
Ø24A- 4C Ø4 Ø2 Ø7ØØ          JMP RESTRT
               Ø71Ø
Ø24D- 38       Ø72Ø ERROR    SEC           Let monitor print
Ø24E- 2Ø 72 89 Ø73Ø          JSR BEEP       error message
Ø251- 2Ø 71 81 Ø74Ø          JSR ERMSG
Ø254- 4C Ø4 Ø2 Ø75Ø          JMP RESTRT    And try again
               Ø76Ø
               Ø77Ø          ; Convert hexadecimal to decimal
               Ø78Ø
Ø257- 2Ø 2Ø 82 Ø79Ø H2D      JSR PARM      Get hex to convert
Ø25A- DØ F1    Ø8ØØ          BNE ERROR     Non-hex is not allowed
Ø25C- 2Ø 4D 83 Ø81Ø          JSR CRLF      New line
Ø25F- 2Ø B1 Ø2 Ø82Ø          JSR CONH2A    And conveRt it
Ø262- 4C Ø4 Ø2 Ø83Ø          JMP RESTRT    Get next # to convert
               Ø84Ø
               Ø85Ø          ; Store ASCII-coded decimal as hex
               Ø86Ø
Ø265- 29 ØF    Ø87Ø CONA2H   AND #$ØF      Convert to BCD
Ø267- 48       Ø88Ø          PHA           Save new digit
Ø268- ØE 4A A6 Ø89Ø          ASL P3L       Multiply current hex by 2
Ø26B- 2E 4B A6 Ø9ØØ          ROL P3H
Ø26E- BØ 3A    Ø91Ø          BCS OVFLO
Ø27Ø- AD 4B A6 Ø92Ø          LDA P3H
Ø273- 48       Ø93Ø          PHA           Save 2 X current value
Ø274- AD 4A A6 Ø94Ø          LDA P3L
Ø277- 48       Ø95Ø          PHA
Ø278- ØE 4A A6 Ø96Ø          ASL P3L       4 X current
Ø27B- 2E 4B A6 Ø97Ø          ROL P3H
Ø27E- BØ 2A    Ø98Ø          BCS OVFLO
Ø28Ø- ØE 4A A6 Ø99Ø          ASL P3L       8 X current
Ø283- 2E 4B A6 1ØØØ          ROL P3H
Ø286- BØ 22    1Ø1Ø          BCS OVFLO
Ø288- 68       1Ø2Ø          PLA           Retrieve 2 X
Ø289- 6D 4A A6 1Ø3Ø          ADC P3L       2 X + 8 X = 1Ø X
Ø28C- 8D 4A A6 1Ø4Ø          STA P3L        current value
Ø28F- 68       1Ø5Ø          PLA
Ø29Ø- 6D 4B A6 1Ø6Ø          ADC P3H
Ø293- BØ 15    1Ø7Ø          BCS OVFLO
Ø295- 8D 4B A6 1Ø8Ø          STA P3H
Ø298- 68       1Ø9Ø          PLA           Now, get new digit
Ø299- 6D 4A A6 11ØØ          ADC P3L       And add to 1Ø X
Ø29C- 8D 4A A6 111Ø          STA P3L        current value
Ø29F- A9 ØØ    113Ø          LDA #$ØØ
Ø2A1- 6D 4B A6 114Ø          ADC P3H       Pick up any carry
Ø2A4- 8D 4B A6 115Ø          STA P3H
Ø2A7- BØ Ø1    116Ø          BCS OVFLO
Ø2A9- 6Ø       117Ø          RTS
               118Ø
Ø2AA- A2 FD    119Ø OVFLO    LDX #$FD      Overflow; n > 65535
Ø2AC- 9A       12ØØ          TXS           Clean-up stack
Ø2AD- A9 ØF    121Ø          LDA #$ØF      Code for "OVERFLOW"
Ø2AF- DØ 9C    122Ø          BNE ERROR     (Always)
               123Ø
```



```
               124Ø          ; Output hex as ASCII-coded decimal
               125Ø
Ø2B1- AØ Ø8    126Ø CONH2A   LDY #$Ø8      Table pointer
Ø2B3- A2 ØØ    127Ø NEXTD    LDX #$ØØ      Decimal-to-be
Ø2B5- AD 4A A6 128Ø SUBT     LDA P3L       Hex value to convert
Ø2B8- 38       129Ø          SEC
Ø2B9- F9 F5 Ø2 13ØØ          SBC TABL-1,Y  Subtract decimal value of
Ø2BC- 8D 4A A6 131Ø          STA P3L        this digit from the hex
Ø2BF- AD 4B A6 132Ø          LDA P3H        value stored in P3
Ø2C2- 88       133Ø          DEY            until we exceed the hex
Ø2C3- F9 F5 Ø2 134Ø          SBC TABL-1,Y   value, then add-back one
Ø2C6- 9Ø Ø7    135Ø          BCC ADJOUT
Ø2C8- 8D 4B A6 136Ø          STA P3H
Ø2CB- C8       137Ø          INY           Here is where we count
Ø2CC- E8       138Ø          INX            in decimal ( Ø <= X <= 9 )
Ø2CD- DØ E6    139Ø          BNE SUBT      (Always)
               14ØØ
Ø2CF- C8       141Ø ADJOUT   INY           Add back 1 to P3L;
Ø2DØ- AD 4A A6 142Ø          LDA P3L        P3H was not changed
Ø2D3- 79 F5 Ø2 143Ø          ADC TABL-1,Y
Ø2D6- 8D 4A A6 144Ø          STA P3L
Ø2D9- 8A       145Ø          TXA           Move decimal to A
Ø2DA- DØ Ø5    146Ø          BNE ASCOUT    Suppress leading zeros
Ø2DC- AE FE Ø2 147Ø          LDX ZFLAG     Test leading zero flag
Ø2DF- FØ Ø8    148Ø          BEQ NOPR
Ø2E1- EE FE Ø2 149Ø ASCOUT   INC ZFLAG     Indicate found non-zero
Ø2E4- Ø9 3Ø    15ØØ          ORA #$3Ø      Convert it to ASCII
Ø2E6- 2Ø 47 8A 151Ø          JSR OUTCHR    And print it
Ø2E9- 88       152Ø NOPR     DEY           Now adjust for next
Ø2EA- 88       153Ø          DEY            decimal value
Ø2EB- DØ C6    154Ø          BNE NEXTD     Unless this is the last?
Ø2ED- AD 4A A6 155Ø          LDA P3L
Ø2FØ- Ø9 3Ø    156Ø          ORA #$3Ø      Always print the last
Ø2F2- 2Ø 47 8A 157Ø          JSR OUTCHR     digit, even if it's zero
Ø2F5- 6Ø       158Ø          RTS
               159Ø
Ø2F6- ØØ ØA    16ØØ TABL     .BY $ØØ $ØA     ;Decimal 1Ø
Ø2F8- ØØ 64    161Ø          .BY $ØØ $64     ;Decimal 1ØØ
Ø2FA- Ø3 E8    162Ø          .BY $Ø3 $E8     ;Decimal 1ØØØ
Ø2FC- 27 1Ø    163Ø          .BY $27 $1Ø     ;Decimal 1ØØØØ
               164Ø
Ø2FE- ØØ       165Ø ZFLAG    .BY $ØØ         ;Leading zeros flag
               166Ø
               167Ø          .EN
```

FIGURE BELOW IS MORE FULLY DESCRIBED ON PAGE 15-21

Sample DIRECTORY listing from VIC=20/CBM-64 1541 Disk Drive System which, it is hoped, can be adapted to the SYM-1. "MAE" is the CBM-64 equivalent of RAE-1 on the SYM-1.

```
0 [illegible]
9    "MAE/DOSMD.EXE64"   PRG
39   "MAE.EXE64"         PRG
17   "MICROMON.EXE64"    PRG
22   "MAE.NOT"           PRG
12   "WORDP.EXE64@8500"  PRG
13   "WORDP.REL64"       SEQ
16   "WORDP.INS"         PRG
8    "REL.EXE64"         PRG
8    "REL.REL64"         SEQ
4    "UART.CTL"          PRG
9    "UART.M01"          PRG
4    "UART.M02"          PRG
3    "UART.M03"          PRG
11   "PET.LIB CBM64"     PRG
29   "IEEE.LIB"          PRG
14   "MLMACROS.MLIB"     PRG
8    "SWEET16.MLIB"      PRG
2    "SECTOR.CTL"        PRG
10   "SECTOR.PGM"        PRG
12   "PET SOURCES.BAS"   PRG
3    "STARWRITER.BAS"    PRG
5    "ASCII.BAS"         PRG
7    "GL.SORT.CTL"       PRG
5    "GL.SORT.M00"       PRG
28   "GL.SORT.M01"       PRG
12   "SM-SORT.ASM"       PRG
1    "SORT.TEST.BAS"     PRG
2    "1525 PRINTER.BAS"  PRG
26   "TAPE64.ASM"        PRG
3    "TAPE64.REL"        SEQ
3    "TAPE.EXE64@8000"   PRG
5    "SCROLL.EXE64"      PRG
6    "SCROLL.REL64"      SEQ
306 BLOCKS FREE.
```

FDC-1 TECHNICAL NOTES - ISSUE 2

Number 2.1

The following letter, from David W. Lewis, 1424 N. Chigwell Lane, Webster, NY 14580, contains some very helpful information on the FDC-1:

Lux:

Enclosed you will find my edited listing of EDB. Normally I would provide you with full source code. However, my system is not a standard SYM and my EDB source is greatly modified for my parallel port keyboard, memory mapped video, parallel port printer, and expanded I/O.

In the EDB listing enclosed, you will find the code for the real time clock is changed. This change prevents the clock from generating an IRQ until it is enabled and the time is set with the .STIME command. If this change to EDB is made, it is not necessary to fix the IRQ bug in FDC-1 code unless the clock function is need. This also lets EDB run slightly faster. [Editor's Note: The file described here, and listed below, is a direct replacement for EDB File 50, for those of you who have copies of Jack Brown's EDB.]

EDB will patch in the disks when ever a cold or warm start is made (.G 200, .G 203, or .G after a break to MON). I have tested all functions and found no problems. However, there are probably bugs. If you find any, please let me know.

One area of concern I have not yet investigated is the variable file loading when the BASIC source is enlarged. Also, I believe that there is a problem if HIMEM is lowered (i.e., lowered to $4000 from $8000) to allow room for an assembly language program. The variable file may load over the protected code.

Enclosed with the marked EDB source you will find the EDB FDC-1 patch listing, the IRQ and DC command listing. Also, on tape you will find the following:

1) A copy of this letter, file F1
2) A copy of EDB FDC-1 disk handler EDB.10, file F2
3) A copy of IRQ and DC patches, file F3

Regarding the problem of CRC disk errors, I am enclosing a copy of an article on this type of disk controller. On my system with 40 track drives with double density storage it is not unusual to get CRC errors on the inner 5 tracks. I found that the 1791 was slowly degrading in performance until the only way it would work was to cool it with freeze spray. I found it impossible to get a Synertek 1791, so I replaced it with a Western Digital chip. To do this the +5 vdc land to pin 40 was cut. Then +12 vdc from my bus was provided to pin 40 of the 1791 through an unused pin on the PWB edge connector.

Another unusual error I originally had was lost data. Due to the delay through my bus buffer card, the disk controller DRQ was not detected. The S.O. (set overflow) input of a 6502 must be synchronized with the falling edge of the phase 1 clock. This was done on my bus interface card with a D-flip flop 74LS74.

FDC-1 IRQ Interupt BUG

There is a bug in the FDC-1 IRQ software IRQRTN at $9C5D that prevents any IRQ from being executed from the user UIRQVC location in system ram. Any IRQ will be executed thru IRQVEC at $A67E which points to the disk

IRQRTN routine. This causes the system to hang up on user IRQ's or a software BRK.

The reason for this is simple. Whenever the 1791 Disk Controller chip is executing a command, the busy status bit is set and data transfers are controlled by the DRQ (data request line) and the 6502 S.O. (set overflow) input. After the command is complete, the busy bit goes low and then the IRQ goes high. Therefore, the disk IRQ can never occur when the busy bit is set.

Examination of the IRQRTN code shows that the branch to the disk routine is taken whenever the busy bit is low. This is true for all IRQ's.

The fix for this is to test a flag, not the busy bit in the status register. Since the only entry to to disk handling routine is DISKIO at $9800, the 5 calls to this routine can be pointed to a routine to set and clear the disk IRQ flag. The address selected is $9780 (easy to remember). If the modified IRQRTN routine is also moved here, an added bonus can be gained. The upper 2k of the disk handler can be simply paged in memory with an I/O line, providing an extra 2k of memory space.

The only problem I see is finding a byte of RAM for the flag. On my system I have 2 blocks of 512 bytes of RAM for disk use in the I/O space (total 1k of RAM for disk use). So finding the the extra byte for DISK.FLAG was no problem. I have included the software listing for this bug fix.

FILE SAVE BUG

There is a bug in the file save routine. If the last byte of a file is the only byte in the last sector, the byte will not be saved.

Example: Sector size 256, save 200 - 300.
Only 200 - 2FF will be saved.

The directory will show the full file range of 200 - 300. I have not looked into this, but I believe that the file size is computed by END.ADDRESS - START.ADDRESS, which is 1 byte short. This has a 1 in 256 or 1 in 128 (etc.) chance of missing the last byte on random length files.

Number 2.2

Here are several FDC-1 patches by Dave Lewis:

```
;*************************************************************
;
; FDC-1 PATCHES FOR :
; 1) IRQ BUG
; 2) PAGING OF UPPER 2K BYTES OF FDC-1 EPROM
; 3) DC COMMAND FOR RAE DISK AND TAPE
;
; USAGE -
;
;DISK.FLAG -
; THIS FLAG IS USED TO TAKE CARE OF THE FDC-1 IRQ BUG.
; BIT 0 OF THE FLAG IS SET TO INDICATE A DISK OPERATION
; IS IN PROGRESS. THE IRQRTN CHECKS THIS BIT, NOT THE
; 1791 DISK CONTROLLER CHIP, TO DETERMINE THE SOURCE OF
; AN IRQ. IF BIT 0 = 1, THEN A DISK IRQ HAS OCCURRED AND
; IRQRTN JUMPS TO IOCOMP AT $9C7D. IF BIT 0 = 0, A USER
; IRQ HAS OCCURRED AND IRQRTN JUMPS TO IRQBRK AT $800F.
;
;IRQRTN -
```

```
; THE VERSION OF IRQRTN HERE IS COPIED FROM $9C5D WITH SLIGHT
; CHANGES TO ALLOW I/O PAGING OF THE UPPER EPROM WITH THE
; VIDEO PWB. IF PAGING IS NOT DESIRED, CHANGE THE LABEL
; STAREG TO DISK.FLAG IN LINE 4070 OF THE ORIGINAL IRQRTN
; ROUTINE. I HAVE NOT DETERMINED THE BEST RAM LOCATION
; FOR DISK.FLAG ON A STANDARD SYM-1 WITH FDC-1. IF PAGING IS
; USED, IRQRTN MUST BE MOVED DOWN INTO THE FIRST 2K AND THE
; THE IRQVEC INITIALIZATION IN DINIT AT $9880 MUST POINT
; TO THE NEW ADDRESS.
;
;GO.DISK -
; ALL CALLS TO DISKIO AT $9800 (5 CALLS) MUST NOW POINT TO
; GO.DISK. GO.DISK WILL THEN SET DISK.FLAG, PAGE EPROM IF
; DESIRED, CALL DISKIO AT $9800, THEN CLEAR DISK.FLAG BEFORE
; RETURNING. THE STARTING ADDRESS OF $9780 WAS CHOSEN TO BE
; EASILY REMEMBERED SINCE THIS IS THE NEW DISK HANDLER ENTRY
; POINT (NO LONGER $9800 DISKIO).
;
;RAE.DC -
; THIS CODE USES THE DC (DISK COMMAND) FUNCTION OF RAE TO
; SWITCH BETWEEN TAPE AND DISK. IF TAPE LOADS AND STORES
; (NOT .CT TAPE ASSEMBLY) IS DESIRED, ENTER >DC T AND THE
; DISK FUNCTION IS DISABLED. TO SWITCH BACK TO DISK, ENTER
; >DC D.
;      DC T    : DISABLE DISK, ENABLE TAPE
;      DC D    : ENABLE DISK
; TO USE THIS FUNCTION, ADD THE FOLLOWING LINE INTO THE
; RAELINK CODE AT $971C.
;      LINE 5506 JSR SET.DCVEC
;
;***************************************************************
;
            .BA $9780
            .MC $7780
;
GO.DISK     PHA
            LDA #05       ;disable video card, enable disk eprom
            STA $A113     ;video/FDC-1 eprom paging I/O address
            STA DISK.FLAG         ;SET FLAG FOR DISK IRQ
            PLA
            JSR DISKIO    ;run disk
            PHA
            LDA #04       ;enable video, disable disk eprom
            STA $A113     ;video/FDC-1 eprom paging I/O address
            STA DISK.FLAG         ;CLEAR FLAG FOR USER IRQ
            PLA
            RTS
;
;
;  IRQ HANDLER
;
IRQBRK      .DE $800F     ;monitor IRQ handler
IOCOMP      .DE $9C7D
PAGE.1      .DE $0100
BSYBIT      .DE $01
STAREG      .DE $F000
;
IRQRTN      PHP
            PHA
            TXA
            PHA
            TSX
            LDA PAGE.1+4,X
            AND #$10      ;MASK FOR B FLAG
            BNE IRQRET    ;IF A BREAK INSTRUCTION
```



```
            LDA #BSYBIT
            BIT DISK.FLAG          ;CHECK FOR ACTIVE DISK
            BEQ IRQRET    ;IF DISK NOT ACTIVE, BRANCH & LET SYM HANDLE IRQ
            LDA STAREG    ;CLEAR DISK IRQ
            JMP IOCOMP    ;DISK BUSY, JUMP TO DISK IRQ HANDLER
IRQRET      PLA  ;BRK OR NON-DISK IRQ
            TAX
            PLA
            PLP
            JMP IRQBRK    ;LET SYM HANDLE IT
;
;-------------------------------------------------------------
;
;INITIALIZE DC VECTOR FOR RAE DISK COMMAND VECTOR
;
SET.DCVEC   LDA #H,RAE.DC
            STA *$ED
            LDA #L,RAE.DC
            STA *$EC
            RTS
;
;RAE DISK COMMAND DC HANDLER
;
RAE.DC      LDY #0        ;point to start of RAE input buffer
            JSR MVNEXT    ;move past DC to next field
            CPY #80       ;past end of buffer?
            BEQ NOT.GOOD          ;branch if at buffer end
            LDA $135,Y    ;get 1st char of 2nd field in buffer
            CMP #'D       ;is char a D for enable disk
            BNE TAPE?     ;branch if not D
DISK.DC     LDA #1        ;yes, a D
STORE.DC    STA $EE       ;alter DC vector flag
            RTS  ;finished
TAPE?       CMP #'T       ;is 1st char in 2nd field T
            BNE NOT.GOOD          ;branch if not T
            LDA #0        ;disable disk,allows proper tape load
            BEQ STORE.DC          ;forced branch
NOT.GOOD    JMP ERROROUT          ;char not D or T, input error
;
            .EN
```

Number 2.3

Here is Dave Lewis' FDC-1/EDB-1 Link for users of Jack Brown's Extended Disk Basic (EDB-1):

```
; EDB.10 9:30 PM MON FEB 21 1983
;---------------------------------------------
;
            IFE DISK-1
;
;*****************************************************
; USAGE
;
; .IN=2 set disk drive 0 as input device
; .IN=3 set disk drive 1 as input device
; .OUT=2 set disk drive 0 as output device
; .OUT=3 set disk drive 1 as output device
;
; Automatically reverts back to application drive 1
; after any access on system drive 0. If this is not
; desired, remove the four lines in this file which
; forces this function. Set default read/write device
; numbers as desired in page 2 locations RDEV and WDEV.
;
```

```
; Real time clock IRQ must be disabled during disk calls.
; If it is desired that the clock always run after cold
; start, remove 2 lines of the CLK.FLAG check in this
; file. CLK.FLAG is set to $C0 during the .STIME routine
; when the clock IRQ hardware is enabled. If the original
; EDB clock enble function is used, these lines must be
; removed and CLK.FLAG is not needed.
;
;*****************************************************
;
; Extended Disk Basic. Parameters for FDC-1.
;
NAME.PTR   .DE $FC
;
PARNR      .DE $A649
P3         .DE $A64A
P3L        .DE $A64A
P3H        .DE $A64B
P2         .DE $A64C
P2L        .DE $A64C
P2H        .DE $A64D
P1         .DE $A64E
P1L        .DE $A64E
P1H        .DE $A64F
;
MONENTRY   .DE $9006    ;initialize FDC-1 vectors, to set IRQ only
POINTNAM   .DE $9064    ;point *FC to NAME.BUF
AR2        .DE $908D    ;FDC-1 file load entry
NMBLANK    .DE $9199    ;put spaces in NAM.BUF
S3CHECK    .DE $92CA    ;FDC-1 file save entry
;
;--------------------------------
;
DISK.SAVE  JSR POINT.NAME       ;set ptr, clear buffer, move name
           JSR ADJ.WRITE        ;get drive # with verify
           STX P1L      ;set drive # for save with verify
           JSR S3CHECK+16       ;do save
           LDA #3       ;force drive 1 after write access of drive 0
           STA WDEV     ;set write device
           JMP DISK.DONE        ;check for clock before return
;
LOAD.NOREL LDA #1       ;1 parm load file with no relocation
           BNE LOAD.FILE        ;forced branch
DISK.LOAD  LDA #2       ;2 parm load file with relocation
LOAD.FILE  STA PARNR    ;set up for 1 or 2 parm load
           JSR POINT.NAME       ;set up NAME.BUF
           JSR ADJ.READ         ;get drive # with verify
           STX P2L      ;set drive # for load
           JSR AR2      ;do load
           LDA #3       ;force drive 1 after read access of drive 0
           STA RDEV     ;set read device
DISK.DONE  LDA #$C0     ;check clock flag
           CMP CLK.FLAG         ;flag = $C0 if clock on, else 0
           BNE DISK.RET         ;branch if no clock
           STA VIAIER   ;enable clock IRQ hardware in 6522 chip
DISK.RET   RTS  ;return from disk command
;
POINT.NAME JSR POINTNAM         ;point *FC to NAME.BUF
NAME.BLK   JSR NMBLANK  ;put spaces in NAME.BUF
MOVE.NAME  LDY #0       ; move NAME to NAME.BUF
           LDX *LABLOC
NAME.LOOP  LDA PGONE,X  ;move file name, 10 char max
           BEQ NAME.END         ;done if 0
           STA (NAME.PTR),Y     ;store char in NAM.BUF
           INY  ;get next char
```



```
           INX
           BNE NAME.LOOP        ;forced branch
NAME.END   CLC  ;no error
           LDA #$40     ;disable clock IRQ at 6522 chip
           STA VIAIER   ;IRQ must be off during disk access
           RTS  ;now do disk command, load or save file
;
ADJ.READ   LDX RDEV     ;get read device, 2 or 3
           BNE ADJ.WRITE+3      ;forced branch
ADJ.WRITE  LDX WDEV     ;get write device, 2 or 3
           INX  ;adjust drive number, force verify
           INX  ;drive 0 = 4, drive 1 = 5
           RTS
;
           ***
;
END.PGM    .EN
```

Number 2.4

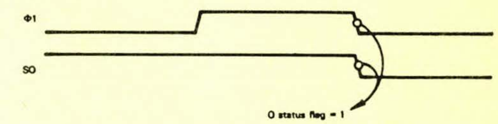
Here are some comments by Dave Lewis on the proper use of the 6502 SO line. He has annotated material on page 9-16 of Leventhal's (OSBORNE/McGraw-Hill) "6800 Assembly Language Programming."

[Note: That's right, the 6800 book! Couldn't find anything on the SO in Leventhal's 6502 book. Does anyone else have inputs on the need to clock S0 with Phase 1 ???]

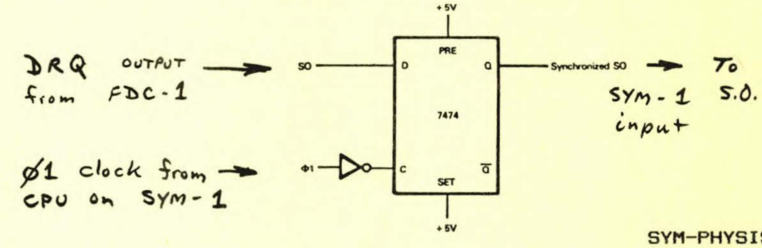
I found that this circuit was required when I put my FDC-1 on my extended bus. I placed the hardware on my bus interface card. If the S.O. input is not in sync, Lost DATA Errors occur. Any use of the S.O. input requires the Ø1 sync.

Dave Lewis

The Set Overflow flag (SO) signal can be used to set to 1 the Overflow bit of the Status register. The SO input must make a high-to-low transition on the trailing edge of the Φ1 pulse in order for the Overflow bit of the Status register to be set to 1. This may be illustrated as follows:



You cannot use the SO input signal in order to reset the Overflow bit of the Status register to 0. Note that external logic must use the Φ1 clock signal in order to synchronize the SO high-to-low transition. A simple 7474 flip-flop can be used for this purpose:



THE SYM-1, THE CBM-64, AND THE VIC=20

For many years the SYM-1 stood alone as the most cost-effective 6502-based single-board computer available. We felt that it was the ideal beginner's computer for those with a reasonable amount of hardware background and some skill with hand tools, or who at least knew which end of a soldering iron was the handle end.

We still believe this, especially since Lance Leventhal's "Microcomputer Experimentation with the Synertek SYM-1" is now available to go along with it. Several factors which others might consider as shortcomings, we consider to be advantages. As one example, we feel that the initial absence of BASIC and a QWERTY keyboard is a strong plus for the SYM-1, since the user is "forced" to learn machine language from the outset. There is also no need to "tie up" a TV set in order to use the computer.

The required power supply and cassette recorder add less than $50 to the initial system cost, and the necessity for interfacing these items to the SYM-1 is an integral part of the learning process. From this point on the SYM-1 is fully expandable in any direction(s) desired by the user, and, in this sense, is the most "personal" computer available.

The absence of games, and a seeming "unfriendliness" to non-technically oriented users, makes the SYM-1 relatively non-accessable for any youngsters around the household. For this reason, as well as the desire for color graphics and wealth of software availability, many SYM-1 owners have Apple II's around as second (or perhaps even first) computers.

While we think highly of the Apple, and very soon will even have one installed in our campus office, courtesy of a special arrangement to provide all full-time computer science faculty with Apple IIE's, we never considered getting one for home use. On the other hand, we now have one each VIC=20 and CBM-64, to supplement several of our SYM-1 systems. Here's why:

The CBM-64 has probably the very best color graphics and music synthesis capabilities available at anywhere near its low cost. We installed Carl Moser's "MAE" (Macro Assembler Editor, first cousin to RAE-1), as the first order of business. As you can see from the printed "Directory Listing" of the MAE disk, among the many utilities, is one called "WORDP.EXE64@8600". This we SYM-PHYSIS readers know under the name of SWP! Hence, much of what we learned on the SYM-1 is directly applicable to the CBM-64. [NOTE: Directory Listing is reproduced on page 15-14.]

MAE and RAE are also first cousins to ASSM/TED, long available on other CBM systems, including the PET, and a tremendous amount of public domain software is available, at $10 per diskette, from the ATUG (ASSM/TED Users' Group), including an excellent disassembler into MAE, similar to Dessaintes' Disassembler into RAE.

Thus, the CBM-64 is far more compatible with SYM-1 than is the Apple II, thanks to the MAE/RAE relationship, at much lower cost (at this writing in the neighborhopd of $350 in the US), and we intend to make it even more compatible, as we shall describe below. We will be using our CBM-64 primarily to develop teaching software for the VIC=20, again as described below.

We have long felt that our computer science students were being trained by 16th century methods, for the job market as it existed three years ago (dropping the editorial "we" for a few paragraphs, this is my personal opinion, not necessarily shared by others on the faculty!). This semester I am teaching an experimental course, "Small Computer System Design", for juniors and seniors with absolutely no hardware

background. The objectives of the course include learning to read schematics, understanding the use of TTLs, VIAs, RAMs, ROMs, etc., and how to use a disassembler (which most of the students had never even heard of!) to probe the inner workings of a system.

In the hope that the students would be encouraged to buy their own personal computer, I chose as the "Model System" the lowest cost system available, the VIC=20. While the Timex/Sinclair had an apparently lower initial cost, it was not considered as effective, since the VIC=20 has more RAM (5K vs 2K), as well as built-in serial, parallel, and RS-232-C interfaces, including both the hardware (two 6522s) and software drivers (20K ROM vs 8K), all of which are extra cost options on the T/S.

Additionally, the VIC=20 has a better keyboard, color graphics, the more universal Microsoft BASIC, the easier to learn 6502 (vs the Z80), and easier to learn logical design (specs on the inner workings of the multipurpose main chip of the T/S are hard to come by and the knowledge gained from its study would not be applicable to other systems.

As the price of the VIC=20 dropped from $200 to $80 during the semester more students purchased their own computers, and most say that they wish they had started earlier. Next semester I will require that students form small study groups, with each student having at least a one-third share of a VIC=20, since this will actually cost them less than a text. In addition I will place a collection of books on the VIC=20 on Library Reserve for them to use.

Now to get back to the SYM-1! We removed the ROMs from our VIC=20, and inserted them in place of the BAS-1 chip on one of our SYM-1s. (We don't yet have the disassembler into MAE running on the CBM-64, and the disassemblers available in the Machine Language Monitors for the CBM-64 are only simple one-pass versions.) We dissambled their contents and edited the results, and provided copies of the listings to students for their study and annotation. The I/O management portion of the VIC=20 OS is excellent, and the method of handling the disk via a simplified IEEE interface (serial vs parallel) is well worth adopting to the SYM-1.

The 1541 single disk drive, for BOTH the VIC=20 and the CBM-64, is the least expensive one we've ever seen (around $350 discounted). The units are self contained, and only two pages of RAM (for data buffers) are required. Only the software driver is required to interface them to the SYM-1! How's that for hardware compatiblity?

Some readers will remember that we added color graphics to our SYM-1s, first with Turpin's ColorMate, then with one of the RCA VP3301 Data Terminals. Both are directly compatible with VCRs. The former has pixel mapping (requiring 4K of the SYM-1's RAM), the latter permits a user specified graphics character set, and can be used on the SYM-1's 20 mA loop.

The VIC=20 with its built-in RS-232-C interface (actually inverted TTL) would make an inexpensive color terminal for the SYM-1. The VIC=20 has an interlace mode permitting its output to be superimposed onto a video image during editing of VCR recordings. Additionally, the KTM-2/80, when interfaced via RS-232-C with either the VIC=20 or the CBM-64, would add the 80 column display so nice for word processing.

The SYM-1 and/or the KTM-2/80 and either or both the VIC=20 and CBM-64 are natural go-togethers. No additional hardware elements (unlike the Apples) other than connectors and cables are required for interfacing them. Do you see why we are so excited by these two new low priced systems?

All that is required is the time to do the software job!!!! We'll be glad to work with any of our readers with VIC/SYM systems by providing copies of our VIC disassemblies in RAE-1 readable format. The 1541 Disk Drive software is almost directly usable in the SYM-1, providing the timing loops are modified to the ratio of the 1.022727 MHz to 1.000000 MHz clocks (a 2% error), and that the appropriate IEEE protocol is followed. Linkage to RAE through the DC command would be relatively simple. Linkage to BAS-1 could then be handled by a .DC command using essentially the same subroutines, except for possible relocation of the buffers.

A MORSE CODE KEYER

Here's a program by our Number One Son, Jim Luxenberg, 949 Hensley, San Bruno, CA 94066. He has been a SYMmer for about a year, and got his Ham Ticket just a few months ago. His wife has been a Systems Analyst with IBM for many years, so they now have an IBM Personal Computer in addition to his SYM-1.

```
10 REM MORSE PROGRAM BY JIM LUXENBERG KA6WRZ 9 APRIL 1983
20 REM THIS PROGRAM ACCEPTS 3 LINES OF TEXT AND OUTPUTS MORSE CODE
25 REM THROUGH PORT PB0. THIS PORT WILL DRIVE A RELAY WHICH CAN BE
30 REM USED TO KEY A TRANSMITTER OR CODE PRACTICE OSCILLATOR.
40 REM NOTE- PROGRAM WILL NOT ACCEPT A COMMA (,) AS INPUT. SOME OTHER
45 REM NOT COMMONLY USED PUNCTUATION MARKS HAVE BEEN LEFT OUT OF THE
50 REM PROGRAM BUT THEY CAN EASILY BE INCLUDED IF DESIRED.
100 CLEAR
110 PRINTCHR$(27)+"E":FORN=1TO9:NEXT
120 PT=44032
125 DIMC$(50)
130 C$(1)="--..--":C$(2)="-...-":C$(3)=".-.-.-":C$(4)="-..-."
140 C$(5)="-----":C$(6)=".----":C$(7)="..---":C$(8)="...--"
150 C$(9)="....-":C$(10)=".....":C$(11)="-....":C$(12)="--..."
160 C$(13)="---..":C$(14)="----.":C$(20)="..--..":C$(22)=".-"
170 C$(23)="-...":C$(24)="-.-.":C$(25)="-..":C$(26)=".":C$(27)="..-."
180 C$(28)="--.":C$(29)="....":C$(30)="..":C$(31)=".---":C$(32)="-.-"
190 C$(33)=".-..":C$(34)="--":C$(35)="-.":C$(36)="---":C$(37)=".--."
200 C$(38)="--.-":C$(39)=".-.":C$(40)="...":C$(41)="-":C$(42)="..-"
210 C$(43)="...-":C$(44)=".--":C$(45)="-..-":C$(46)="-.--"
220 C$(47)="--.."
250 POKEPT+2,255
260 PRINT"COMPUTER GENERATED MORSE CODE PROGRAM"
270 PRINT:PRINT:PRINT
280 INPUT"ENTER DESIRED CODE SPEED IN WPM  ";S
290 S=INT(514/S)
300 PRINT"ENTER THE TEXT TO BE CONVERTED TO CODE"
310 FORB=1TO3
320 INPUTA$(B)
330 A$=A$+A$(B)
340 NEXT B
350 FORE=1TOLEN(A$)
360 IFMID$(A$,E,1)=" "THENGOSUB540:NEXTE
370 X$=MID$(A$,E,1)
380 X=ASC(X$)
390 C$=C$(X-43)
400 FORI=1TOLEN(C$)
410 IFMID$(C$,I,1)="-"THENGOSUB520
420 IFMID$(C$,I,1)="."THENGOSUB510
430 NEXTI
440 FORD=1TO(3*S):NEXTD
450 NEXTE
500 END
510 POKEPT,255:FORN=1TOS:NEXTN:POKEPT,0:FORN=1TOS:NEXTN:RETURN
520 POKEPT,255:FORN=1TO(3*S):NEXTN:POKEPT,0:FORN=1TOS:NEXTN:RETURN
540 FORN=1TO(4*S):NEXT:RETURN
```


LANCE LEVENTHAL'S LATEST BOOK:

"MICROCOMPUTER EXPERIMENTATION WITH THE SYNERTEK SYM-1"

We have a whole bookcase (actually several bookcases!) full of computer books. There are books on computers in general, microcomputers in general, microprocessors in general, particular computers, particular microcomputers, particular microprocessors, languages in general, particular languages, etc., etc. [Among the perks of teaching, of course, are the review copies sent us for possible class adoption. But we actually buy and pay for, out of our own pocket, more than half the books we own.]

Most of the books we have skimmed, and placed on the shelves, never to be looked at again. Many of these books would be useful for beginners, but not truly useful for reference. We have a new city/county (Chico/Butte) library, and we are in the process of clearing out our bookshelves so that we can donate literally scores of these books to help fill the shelves of this new building.

To amuse ourselves during this process, we made a mental list of the top twenty books, the ones we would never part with, at least not during our lifetime. Five of our "Top 20" books are by Lance Leventhal; this should give you some idea of our respect for Dr. Leventhal's writing abilities. We regret that we have not yet met him in person, but hope that one day we will, now that we have found, through a brief exchange of letters, that we have a mutual friend here at the university.

We have reviewed and highly recommended three of his books in earlier issues. We now review and recommend his most recent book, "MICROCOMPUTER EXPERIMENTATION with the SYM-1", Prentice-Hall, Inc., ©1983. To do this 500 page book full justice and to illustrate its tremendous breadth and depth of coverage of the SYM-1 would require far more time than we have available and the few pages we can devote here. So, we'll just let the book speak for itself by reprinting its Table of Contents on pages 15-25,26. You can then judge the value of the book to you for yourselves. Surely there must be at least a few topics in that listing that are "new" for each of us.

The book is organized into 16 "Laboratories", rather than chapters, since the approach is meant to be "hands-on", not just casual reading. The Laboratories are numbered 0, 1, 2, . . . , D, E, F (a nice touch, that!). The material could easily be covered in a one day a week 15 week semester course, or squeezed into a two day a week quarter course. The book is also definitely suitable for self-study.

The book is remarkably free of errors; we didn't find any during our quick examination (of course, our proof-reading eye still needs some minor repair work done to sharpen it up). We do have one very serious complaint about the book, however! Why wasn't it available two years ago!!!!!! But then we would have had to wait for Leventhal's 6809 and Leventhal and Saville's 6502 Subroutine books!

The "SYM-1" book is similar in format to Leventhal's 1981 "MICROCOMPUTER EXPERIMENTATION WITH THE MOTOROLA MEK 6800D2", which we examined to see if we could build a course around the ten MEK 6800D2 kits which were laying around, essentially unused, in one of the storerooms. While the book was great, we didn't feel that the -D2 kits were worth "rehabilitating" for laboratory use. Of course we do admit to having a strong bias towards 6502-based systems, and when the MEK 6800D2 kits were ordered (by another instructor, of course) we fought a losing battle to convince "management" that the SYM-1s would do more for less money.

THIS IS ONE BOOK WHICH EVERY SYM OWNER SHOULD HAVE!

Note that much of the material is directly applicable to 6502 systems in general.

Table of Contents from Leventhal's "MICROCOMPUTER EXPERIMENTATION WITH THE SYNERTEK SYM-1"

ON RECURSION TECHNIQUES - BY TOM GETTYS

Recursion is an extremely powerful programming technique, as those who are versed in languages such as PASCAL and C know.

However, most do not realize that recursion can be used to benefit in BASIC also! While it is up to the user to define and maintain the parameter stack explicitly, the advantages of recursion can often still be realized.

The first example is a routine which computes the factorial of an integer. Notice how close the BASIC implementation matches the standard recursive definition of N factorial (note that no parameter stack is needed here, due to the global nature of all BASIC variables).

The second example is a recursive solution to the ubiquitous Tower of Hanoi probem. Here three arrays are used as a parameter stack. Each time the routine is to call itself the current parameter values are "pushed" on the stack.

I have used this technique to implement several algorithms which lend themselves naturally to a recursive solution, e.g., tree traversal, the QUICKSORT algorithm, etc.

Below you will find two algorithms which utilize recursion. You may enjoy trying your hand at writing these as recursive BASIC programs.

The first searches the array A (of size N) for the first occurrence of the value x. If A(i)=x then i is returned, otherwise 0 is. The second determines the greatest common divisor of the integers a and b, where a>b.

```
  PROCEDURE SEARCH(i)
   BEGIN
     CASE
       IF i>N THEN SEARCH=0
       IF A(i)=x THEN SEARCH=i
                 ELSE SEARCH=SEARCH(i+1)
   END

  PROCEDURE GCD(a,b)
   BEGIN
     IF b=0
        THEN GCD=a
        ELSE GCD=GCD(b,a MOD b)
   END
```

```
100 INPUT "Find the factorial of "; N
110 :
120 GOSUB 310
130 PRINT N "factorial is" F
140 :
150 END
160 :
170 :
180 :
190 REM     The following routine computes the value of
200 REM     of N factorial by the use of recursion.
210 :
220 REM     A pseudo-code version of this routine is as follows:
230 :
240 REM     PROCEDURE FACT(N)
250 REM     BEGIN
260 REM       IF N=1 THEN FACT=1
270 REM               ELSE FACT=N*FACT(N-1)
280 REM     END
```



```
290 :
300 :
310 IF N=1 THEN F=1 : RETURN
320 :
330 N=N-1 : GOSUB 310
340 :
350 N=N+1 : F=N*F : RETURN
```

```
100 INPUT "Number of disks: "; N
110 :
120 DIM S$(N), I$(N), D$(N)
130 :
140 S$(N)="left  "
150 I$(N)="center"
160 D$(N)="right "
170 :
180 GOSUB 530
190 END
200 :
210 :
220 :
230 REM     The following is a recursive routine which
240 REM     solves the TOWER OF HANOI problem.
250 :
260 REM     The underlying idea is this:
270 REM     To move N disks from pole 1 to pole 3
280 REM     1) move N-1 disks from pole 1 to pole 2
290 REM     2) move the bottom disk from pole 1 to pole 3
300 REM     3) move the N-1 disks from pole 2 to pole 3!
310 :
320 REM     The routine to move N disks simply calls upon itself
330 REM     to solve the problem of doing steps 2 and 3, that of
340 REM     moving N-1 disks!
350 :
360 :
370 REM     An equivalent PASCALese version would look something like:
380 :
390 REM     PROCEDURE move(count,source,destination)
400 REM     BEGIN
410 REM       IF count=1
420 REM          THEN WRITE(source,destination)
430 REM          ELSE BEGIN
440 REM                 intermediate=NOT(source OR destination)
450 REM                 move(count-1,source,intermediate)
460 REM                 WRITE(source,destination)
470 REM                 move(count-1,intermediate,destination)
480 REM               END
490 REM     END
500 :
510 :
520 :
530 IF N=1 THEN PRINT S$(N) " ==> " D$(N) : RETURN
540 :
550 S$(N-1)=S$(N)
560 I$(N-1)=D$(N)
570 D$(N-1)=I$(N)
580 N=N-1
590 GOSUB 530
600 :
610 PRINT S$(N+1) " ==> " D$(N+1)
620 :
630 S$(N)=I$(N+1)
640 I$(N)=S$(N+1)
650 D$(N)=D$(N+1)
660 GOSUB 530
670 :
680 N=N+1 : RETURN
```

A 96ØØ BAUD TERMINAL PATCH
- ---- ---- -------- -----

We received the following letter and program from Dr. A. J. Hissink several years ago (!), promptly tested it, and then "lost" the program somewhere in our almost unmanageable collection of cassettes and diskettes! Tom Gettys supplied us with his copy, and we publish it now because of the many requests we have received for it:

Dear Lux,

At last I'm getting around to putting a few thoughts on tape and sending in a few of my utility programs. Most of them were developed from programs in SYM-PHYSIS and adapted to my particular requirements. They may be of interest to some of the SYMaddicts.

One utility will be of general interest to KTM-2 owners. I noted that the KTM-2 terminal was capable of 96ØØ baud but the upper limit of the MON 1.1 I/O routines was 48ØØ baud. I analysed the timing of "TOUT" and "TIN" and found that 96ØØ baud was possible but that these routines would have to be rewritten. This was desirable from another viewpoint too - the inclusion of parallel printer control.

My first attempt at the I/O routine timing was a linear extrapolation of the lower baud rate timings. However, I found that the loop delays were more critical then they should have been so I calculated the times from scratch and found that the 48ØØ baud was not optimum but a compromise to get the wide range of baud rates. I believe the timings in these routines are optimum. They certainly aren't critical and should work first time in all terminals.

My routines are now built into a new reset program. However, this program will work as is by "G" to the object code starting address. The terminal will go dead. Switch the baud rate selector on the KTM-2 to 96ØØ, do a CONTROL SPACE to reread the option switches and you will be up and running. Note you don't have to send a character to get things going any more (another source of annoyance!).

Each call to the object code at label "PRINTER" will initialize the port for a 7 data bit parallel printer with "BUSY" on bit 7 (ie the 8th bit) and toggle the printer I/O on and off. Note that it uses bit Ø of TOUTFL to determine the printer output status.

```
ØØ1Ø ;***
ØØ2Ø ;***   SYM-1 TERMINAL I/O - 96ØØ BAUD
ØØ3Ø ;***
ØØ4Ø                .BA $7FØØ    ;(OR WHEREVER!)
ØØ5Ø ; .OS
ØØ6Ø                .ES
ØØ7Ø ;
ØØ8Ø ;  ADDRESS DECLARATIONS
ØØ9Ø ;
Ø1ØØ SAVER          .DE $8188
Ø11Ø PBDA           .DE $A4Ø2    ;TERM INPUT
Ø12Ø TOUTFL         .DE $A654
Ø13Ø TECHO          .DE $A653
Ø14Ø INVEC          .DE $A661
Ø15Ø OUTVEC         .DE $A664
Ø16Ø ORB            .DE $A8ØØ    ;PARALLEL PRINTER PORT
Ø17Ø DDRB           .DE $A8Ø2
Ø18Ø PCR            .DE $A8ØC
Ø19Ø USRENT         .DE $8Ø35
Ø2ØØ RESXAF         .DE $81B8
Ø21Ø RESALL         .DE $81C4
Ø22Ø ACCESS         .DE $8B86
Ø23Ø ;
Ø24Ø ;
```



```
                 Ø25Ø ; MACRO DEFN
                 Ø26Ø ;
                 Ø27Ø !!!SL      .MD (ROUTINE LINK)   ;SET LINK
                 Ø28Ø            LDA #L,ROUTINE
                 Ø29Ø            STA LINK
                 Ø3ØØ            LDA #H,ROUTINE
                 Ø31Ø            STA LINK+1
                 Ø32Ø            .ME
                 Ø33Ø ;
                 Ø34Ø ;***   VECTOR PATCH
                 Ø35Ø ;
7FØØ- 2Ø 86 8B   Ø36Ø            JSR ACCESS
                 Ø37Ø            SL (TOUT OUTVEC)
7FØ3- A9 6Ø
7FØ5- 8D 64 A6
7FØ8- A9 7F
7FØA- 8D 65 A6
                 Ø38Ø            SL (INTCHR INVEC)
7FØD- A9 18
7FØF- 8D 61 A6
7F12- A9 7F
7F14- 8D 62 A6
7F17- 6Ø         Ø39Ø            RTS
                 Ø4ØØ ;***
                 Ø41Ø ;***   SYM-1 TERMINAL I/O - 96ØØ BAUD
                 Ø42Ø ;***
7F18- 2Ø 88 81   Ø43Ø INTCHR     JSR SAVER    ;IN TERMINAL CHAR
7F1B- A9 ØØ      Ø44Ø            LDA #Ø
7F1D- 85 F9      Ø45Ø            STA *$F9
7F1F- AD Ø2 A4   Ø46Ø LOOK       LDA PBDA     ;FIND LDG EDGE
7F22- 2D 54 A6   Ø47Ø            AND TOUTFL
7F25- 38         Ø48Ø            SEC
7F26- E9 4Ø      Ø49Ø            SBC #$4Ø
7F28- 9Ø F5      Ø5ØØ            BCC LOOK
7F2A- AØ Ø6      Ø51Ø TIN        LDY #6       ;31 uS DELAY
7F2C- 88         Ø52Ø TLP2       DEY
7F2D- DØ FD      Ø53Ø            BNE TLP2
7F2F- AD Ø2 A4   Ø54Ø            LDA PBDA     ;TERMINAL BIT
7F32- 2D 54 A6   Ø55Ø            AND TOUTFL
7F35- 38         Ø56Ø            SEC
7F36- E9 4Ø      Ø57Ø            SBC #$4Ø     ;OR BITS 6,7 (TTY,CRT)
7F38- 2C 53 A6   Ø58Ø            BIT TECHO    ;ECHO BIT?
7F3B- 1Ø Ø6      Ø59Ø            BPL DMY1
7F3D- 2Ø 9Ø 7F   Ø6ØØ            JSR OUT
7F4Ø- 4C 49 7F   Ø61Ø            JMP SAVE
                 Ø62Ø ;
7F43- AØ Ø7      Ø63Ø DMY1       LDY #7
7F45- 88         Ø64Ø TLP1       DEY
7F46- DØ FD      Ø65Ø            BNE TLP1
7F48- EA         Ø66Ø            NOP
7F49- 66 F9      Ø67Ø SAVE       ROR *$F9
7F4B- EA         Ø68Ø            NOP  ;TIMING - 8 uS DELAY
7F4C- 48         Ø69Ø            PHA
7F4D- 68         Ø7ØØ            PLA
7F4E- 9Ø DA      Ø71Ø            BCC TIN
7F5Ø- AØ Ø8      Ø72Ø            LDY #8       ;TIMING - 41 uS DELAY
7F52- 88         Ø73Ø TLP3       DEY
7F53- DØ FD      Ø74Ø            BNE TLP3
7F55- 18         Ø75Ø            CLC
7F56- 2Ø 9Ø 7F   Ø76Ø            JSR OUT
7F59- A5 F9      Ø77Ø            LDA *$F9
7F5B- 49 FF      Ø78Ø            EOR #$FF
7F5D- 4C B8 81   Ø79Ø            JMP RESXAF
                 Ø8ØØ ;
7F6Ø- 85 F9      Ø81Ø TOUT       STA *$F9     ;TERM CHR OUT
7F62- 2Ø 88 81   Ø82Ø            JSR SAVER
7F65- A9 Ø1      Ø83Ø            LDA #$Ø1     ;CHECK FOR HARD COPY
```

```
7F67- 2C 54 A6            BIT TOUTFL
7F6A- FØ Ø8               BEQ TERM
7F6C- 8D ØØ A8            STA ORB       ;SEND TO PRINTER
7F6F- 2C ØØ A8   WAIT     BIT ORB       ;IS PRINTER STILL BUSY?
7F72- 3Ø FB               BMI WAIT
7F74- A9 3Ø      TERM     LDA #$3Ø      ;SET FOR OUTPUT
7F76- 8D Ø3 A4            STA PBDA+1    ;DATA DIRECTION
7F79- A5 F9               LDA *$F9      ;RECOVER CHR DATA
7F7B- A2 ØB               LDX #$ØB      START BIT, 8 DATA, 3 STOP BITS
7F7D- 49 FF               EOR #$FF      ;INVERT DATA
7F7F- 38                  SEC
7F80- 20 90 7F   OUTC     JSR OUT       ;OUTPUT BIT FROM CARRY
7F83- AØ ØC               LDY #$ØC
7F85- 88         PHAKE    DEY
7F86- DØ FD               BNE PHAKE
7F88- EA                  NOP
7F89- 4A                  LSR A
7F8A- CA                  DEX
7F8B- DØ F3               BNE OUTC
7F8D- 4C C4 81            JMP RESALL
7F90- 48         OUT      PHA           ;TERMINAL BIT OUT
7F91- AD Ø2 A4            LDA PBDA
7F94- 29 ØF               AND #$ØF
7F96- 90 Ø2               BCC OUTONE
7F98- Ø9 3Ø               ORA #$3Ø
7F9A- 2D 54 A6   OUTONE   AND TOUTFL    ;MASK OUTPUT
7F9D- 8D Ø2 A4            STA PBDA
7FAØ- 68                  PLA
7FA1- 6Ø                  RTS
                 ;
                 ;***   PRINTER CONTROL - ON/OFF TOGGLE
                 ;
7FA2- 48         PRINTER  PHA
7FA3- 2Ø 86 8B            JSR ACCESS
7FA6- AD 54 A6            LDA TOUTFL
7FA9- 49 Ø1               EOR #%ØØØØØØØ1        ;BIT Ø IS PRINTER
7FAB- 8D 54 A6   PRIOUT   STA TOUTFL
7FAE- A9 AØ               LDA #%1Ø1ØØØØØ        ;SET FOR ONE SHOT MODE
7FBØ- 8D ØC A8            STA PCR
7FB3- A9 7F               LDA #%Ø1111111        ;BIT 7 IS "BUSY"
7FB5- 8D Ø2 A8            STA DDRB
7FB8- 68                  PLA
7FB9- 4C 35 8Ø            JMP USRENT
                 ;
                 ;        PRINTER ON
                 ;
7FBC- 48         HARDON   PHA
7FBD- 2Ø 86 8B            JSR ACCESS
7FCØ- A9 Ø1               LDA #%ØØØØØØØ1        ;TURN ON BIT Ø
7FC2- ØD 54 A6            ORA TOUTFL
7FC5- 4C AB 7F            JMP PRIOUT
                 ;
                 ;        PRINTER OFF
                 ;
7FC8- 48         HARDOFF  PHA
7FC9- 2Ø 86 8B            JSR ACCESS
7FCC- A9 FE               LDA #%1111111Ø
7FCE- 2D 54 A6            AND TOUTFL    ;TURN OFF BIT Ø
7FD1- 8D 54 A6            STA TOUTFL
7FD4- A9 ØØ               LDA #Ø
7FD6- 8D ØC A8            STA PCR
7FD9- 8D Ø2 A8            STA DDRB
7FDC- 68                  PLA
7FDD- 4C 35 8Ø            JMP USRENT
                 ;
                          .EN
```


(MORE ON 65CXX, CIA AND SID - continued from page 15-34)

tain, and Release) control capabilities. "Hard Synch", "Ring Modulation", and programmable filters are built-in, and two A/D converters (for reading potentiometers) are thrown-in, for good measure! The SIDs accept externally generated audio signals for processing, and may be daisy-chained, or combined in various ways, for stereo, etc.

Our previous experience with sound effects chips has been with the TI SN 76477, which we built into a stand-alone system with manually operated switches and potentiometers, and with the GI AY-3-891Ø chip, which we interfaced to the SYM-1 through a VIA. Not only is the 651Ø SID far nore versatile than either of these previous chips, it is ever so much simpler to interface, and, because of the CBM-64 "connection", there will be lots of published software, both 65Ø2 ML and Microsoft BASIC, adaptable for it (only the PEEKs, POKEs, USRs, and SYSes need be changed).

Our CBM-64 has been lent to a colleague, so that we could concentrate on the VIC=2Ø. We expect him, in exchange for the loan, to show us how to set the alarm in the CIA, and how to get the most out of the SID.

INFORMATION RETRIEVAL PROBLEMS

As part of the pre-preparation effort for this issue, we took several days out to examine but a small fraction of the magnetic storage media on hand. Here are the results of the review, and some of our conclusions:

While none of our own original materials are on cassettes, we do have a collection of over three hundred cassettes sent in by readers. Most are "neatly" organized in two attache-style cases, each holding 48 cassettes, and ten plastic cassette storage boxes, each holding 15 cassettes. The most recent arrivals, some 5Ø or so, have not yet been "archived", but will be, as soon as we get more storage containers.

The only indication as to the information contained on each cassette is a small label on the visible edge of the cassette case with the name of the sender. For the more prolific contributors the label also bears a date and only a brief hint as to the contents.

Our conclusion? The inadequate indexing method makes information retrieval nearly impossible. Why didn't we do better, and what is the solution? Our excuse is that all cassettes were immediately transcribed to (FODS) diskettes, and that the cassettes were needed only for backup. We have never ever referred to the cassettes a second time. We should have "recycled" the cassettes and skipped buying the fancy storage containers.

We now have over 2ØØ sequentially numbered FODS 5 1/4" diskettes which were in-house generated, plus some 3Ø or so sent in by contributors. We have some 2Ø CODOS 8" disks, both in-house and contributed, and a dozen or so FDC-1 5 1/4" diskettes. In the early days, we actually backed up each diskette with another. We stopped doing this long ago, and plan to reuse some 8Ø backup disks for new materials.

With disks and diskettes retrieval problems still exist however. File names are length limited, and the abbreviations are often much too cryptic. After a few weeks the names no longer serve well as file identifiers. Below, for example, are directory listings from each of our three systems. It should be obvious, on examining these listings, that many of the files are essentially "lost", and would take considerable effort to recover. Only when strongly motivated to find a particular file have we made the necessary effort!

(continued to page 15-35)

Synertek.

SY65C00

CMOS 8-Bit Microprocessor Family

PRELIMINARY

Features

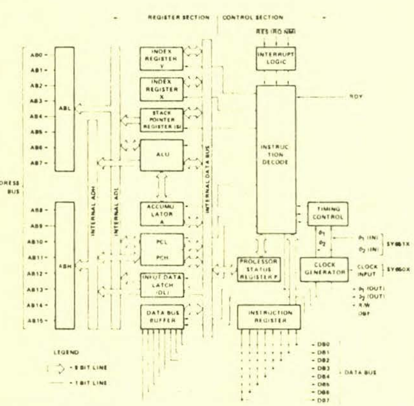
- High Performance 0 Hz to 4 MHz Operation
- Low Power, 8 mA at 4 MHz, 10 Micro Amp Standby at 5 Volts
- Memory Lock (ML) Output During Read-Modify-Write
- Single 3 to 6 Volt Power Supply
- On-Chip Oscillator
- 40 Pin or 28 Pin Versions
- Bus Enable (BE) Allows DMA Operations
- RDY Input to Extend Data Access Times for Use with Slow Memories
- Sync Output Indicating Opcode Fetch
- Improved Bus Timing
- Earlier Valid Address Allows Use of Slower Memories
- 27 New Instructions
- Plug Compatible with NMOS 6502

Description

The CMOS 65C00 microprocessor is compatible with the NMOS 6500 family of microprocessors. This 8-bit microprocessor unit designed in Synertek's proprietary high performance N-well silicon gate technology offers higher performance than the original NMOS 6502. The design allows for operating frequencies up to 4 MHz, and below 1 MHz further reducing its already low power consumption.

Not only is the 65C00 a low power version of the popular 6500 microprocessor, it also has these new features. Ability to tri-state the R/W line, address and data bus for DMA applications. Improved T_{ACC} specs allowing use with slower memory devices. A new optional output enhancing multiprocessing capabilities. Two new addressing modes, and a larger instruction set providing the user with more compact programming capabilities.

Block Diagram



27 New Instructions

Hex	Mnemonic	Description
80	BRA	Branch Relative Always
3A	DEA	Decrement Accumulator
1A	INA	Increment Accumulator
DA	PHX	Push X on Stack
5A	PHY	Push Y on Stack
FA	PLX	Pull X from Stack
7A	PLY	Pull Y from Stack
9C	STZ	Store Zero (Absolute)
9E	STZ	Store Zero (Absolute,X)
64	STZ	Store Zero (Zero Page)
74	STZ	Store Zero (Zero Page,X)
1C	TRB	Test and Reset Memory Bits with Accumulator (Absolute)
14	TRB	Test and Reset Memory Bits with Accumulator (Zero Page)
0C	TSB	Test and Set Memory Bits with Accumulator (Absolute)
04	TSB	Test and Set Memory Bits with Accumulator (Zero Page)
89	Bit	Test Immediate with Accumulator
3C	Bit	Test Memory Bits with Accumulator (Absolute,X)
34	Bit	Test Memory Bits with Accumulator (Zero Page,X)

New Addressing Modes

7C	JMP	Jump (Indirect Absolute,X)
72	ADC	Add Memory to Accumulator with Carry (Indirect)
32	AND	"AND" Memory with Accumulator (Indirect)
D2	CMP	Compare Memory and Accumulator (Indirect)
52	EOR	"Exclusive OR" Memory with Accumulator (Indirect)
B2	LDA	Load Accumulator with Memory (Indirect)
12	ORA	"OR" Memory with Accumulator (Indirect)
F2	SBC	Subtract Memory from Accumulator with Borrow (Indirect)
92	STA	Store Accumulator in Memory (Indirect)

Indexed Absolute Indirect (JUMP)

The contents of the second and third instruction bytes are added to the X register. The result is a 16-bit memory address that contains the low-order eight bits of the effective address. The next memory location contains the high order eight bits of the effective address.

Indirect

In indirect addressing the second byte of the instruction points to a memory location on page zero whose contents is the low order byte of the effective address. The next location on page zero contains the high order byte of the effective address.

Miscellaneous Instruction Changes

Indexed Addressing across the page boundaries will retain the last byte of instruction address rather than an invalid page address.

Processor Hangup on certain invalid opcodes has been eliminated.

Jump Indirect across page boundaries will now increment the page address instead of wrapping around on itself. If a page boundary is crossed the instruction cycle time will increase by one.

Decimal operations involving addition and subtraction will take an additional cycle time. The NMOS Z,N and V flags were invalid, the CMOS flags will be valid.

Read-Modify-Write cycles will be flagged by the $\overline{ML}$ output.

RDY transitioning low will cause the CPU to halt even during write operations. The NMOS version allowed transitions only during read cycles.

DMA Operations on the CMOS 6502 are possible by pulling BE low, thus tri-stating the address and data bus and R/W line.

Decimal Mode Flag condition defaults to the binary mode upon a reset. The NMOS version the flag was random.

New Signals

Memory Lock ($\overline{ML}$) an output, active low, indicates the need to defer the rearbitration of the next bus cycle to insure integrity of read-modify-write cycles in a multiprocessor environment.

Bus Enable (BE) an input, when true allowing normal operation of the microprocessor, when low tri-states R/W, address and data lines, allowing true DMA operations. An improvement over the NMOS version, in that DBE when pulled low would only tri-state the data lines.

Applications Areas

The CMOS version of the 6502 is ideally suited for any low power application or application where noise immunity and potential swings on V_{CC} might occur. It is well suited for automotive, industrial, business, harsh environment (high temp) and communications markets. Not only does it fill the typical CMOS niche, it also is an upgraded version of the NMOS part, providing the new inputs and outputs, better bus timing and 27 new instructions.

Device Pinouts

The CMOS 65C00 family offers the same full line of 10 microprocessor pin configurations as the NMOS family. In addition to those, the CMOS family offers user selectable metal mask options for selection of clock circuitry and bus control input options. Below are the various pin configurations and additional mask options available for all devices.

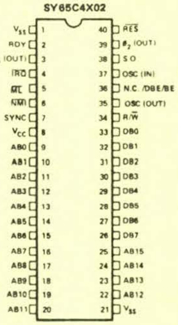
Optional Pull-Up for:

RDY, IRQ, NMI, S.O., RES and DBE/BE inputs, each individually selectable by user.

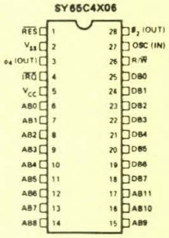
Pin Configurations

4X CLK/OSC

SY65C4X02

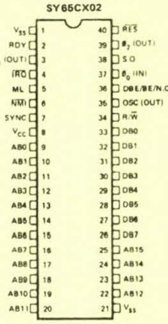


SY65C4X06

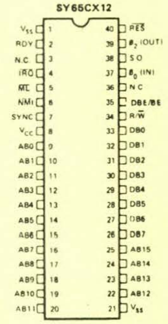


1X CLK/OSC

SY65CX02



SY65CX12



MORE ON THE 65CXX MICROPROCESSOR CHIPS

ALSO, THE CIA AND THE SID

Synertek is an alternate source for the 65CXX family. We reprint above portions of three pages of descriptive material on the microprocessor members of the family. [We regret that the only material available to us for reproduction was a 70% reduction from the original 8 1/2" x 11" sheets, and that the additional 70% reduction factor in our publication process will produce final copy at half-size of the original, so that you may need a magnifying glass to read it!]

The material is from the 1983 issue of the Synertek "Data Manual", which is obtainable through Synertek Distributors, Sales Representatives, and International Sales Offices. The "Data Manual" is fascinating reading, and well worth getting.

Leaving the CMOS technology for the moment, let us remind you that the NMOS 6526 CIA (Complex Interface Adaptor) and the 6581 SID (Sound Interface Chip), available from Commodore's MOS Technology Division, but not yet from Synertek, are very easily interfaceable to the SYM-1. Both are used in the Commodore 64.

The 6526 CIA in an enhancement of the 6522 VIA; the most important new feature is the 24 hour (AM/PM) time-of-day clock with programmable alarm. Thus, you no longer need worry about interrupt driven real time clocks which may lose time during cassette or tape or even RS-232-C I/O operations.

The 6581 SID is a full-fledged, three voice, synthesizer with each voice having its own Tone Oscillator/Waveform Generator, Envelope Generator, and Amplitude Modulator, with a broad range of ADSR (Attack, Decay, Sus-

(continued to page 15-32)

(INFORMATION RETRIEVAL - continued from page 15-32)

```
>dc dir 2
Ø1 :SCOLE  1ØØØ 16DC Ø1 Ø1     Ø2 :FILE2  1ØØØ 3FE4 Ø1 15
Ø3 :FILE1  1ØØØ 37DF Ø7 15     Ø4 :RAEDI  1ØØØ 2AC6 12 15
Ø5 %BASLU  6ØØØ 699A 16 Ø5     Ø6 .BANK   Ø2Ø1 1D97 17 Ø9
Ø7 :96ØØB  Ø2ØØ 12DØ 21 Ø1     Ø8 :MTEST  Ø2ØØ Ø9E7 23 Ø3
Ø9 .HANOI  Ø2Ø1 Ø7AF 24 Ø3     1Ø .FACT   Ø2Ø1 Ø3F5 24 15
11 .GET$   Ø2Ø1 Ø315 25 Ø3     12 :RECUR  Ø2ØØ Ø88E 25 Ø6
13 :HILUX  Ø2ØØ Ø9E5 26 Ø4     14 .BREAK  Ø2Ø1 Ø3ØE 27 Ø4
15 :HISS1  Ø2ØØ ØABD 27 Ø7     16 :HISS2  Ø2ØØ ØAFA 28 Ø9
17 :RECR2  Ø2ØØ Ø92A 29 11

NEXT: T3Ø S1Ø
```

FIGURE 1: Directory of FODS Diskette from Tom Gettys

```
]DIR *.?:1

CODOS.Z         :1 L  21 MAR 83  $0018B3
SYSERRMSG.Z     :1 L  21 MAR 83  $0007A5
SVCPROC.Z       :1 L  21 MAR 83  $00021C
CODX.A          :1 L  21 MAR 83  $008E13
DIR.C           :1 L  21 MAR 83  $000209
STARTUP.J       :1 L  21 MAR 83  $00008F
CODX.C          :1 L  21 MAR 83  $000FFD
RAE.X           :1 L  21 MAR 83  $002014
CODXSIGNON.T    :1 L  21 MAR 83  $00035E
WORDX.A         :1 L  21 MAR 83  $008FCB
WORDX.C         :1 L  21 MAR 83  $001000
WORDXSIGNON.T   :1 L  21 MAR 83  $00038D
WORDXSWP.A      :1 L  21 MAR 83  $005107
WORDXSWP.C      :1 L  21 MAR 83  $000850
LUXLETTER.T     :1 -  *UNDATED*  $00111B
```

FIGURE 2: Directory of CODOS Disk from A. M. Mackay

```
.17 1
%WANDDEMO -0200-0A5B-0201
%ELIZA    -0200-5BCA-0212
%CLKDRVR2 -0200-1AD5-0812
%ACIADRVR -0200-075A-0A08
%EPROGRAMR-0200-3CEA-0A13
%KTM/80ROM-0200-3CF3-0E11
.17 1
%FDC/F1   -1000-4A84-0201
%FDC/F2   -1000-55E7-051D
%BTCP/1.6 -1000-5658-0A13
%BTCP/1.6G-1000-54B5-0F0A
%RTCP/1.4 -1000-22DA-131C
%RTCP/1.4G-1000-2C19-1506
```

FIGURE 3: Directories of FDC-1 Diskettes from Jeff Lavin

What are we doing to solve the problem? For FODS (the majority of our diskettes) we keep a notebook in which each page contains a directory printout. Each printout is fully hand-annotated with sufficient information to fully identify each file, and where the supporting hard-copy documentation (if any) may be found. This we have only begun to do recently; for the older diskettes we make annotations only as we have occasion to refer back to them. (Several readers have inquired about materials which would require many hours of search time to locate on old diskettes. We regret that we cannot find enough time to satisfy their requests.)

Since we have fewer of these, for CODOS and FDC-1 disks the directory listings are tucked into the storage envelopes. It is coincidental that an 8 1/2"x11" sheet of paper folded to quarter-size fits just right into the 5 1/4" envelopes.

We already have a half-dozen or so diskettes, each, for the VIC=2Ø and the CBM-64, and expect another batch to be provided with or for the soon-to-be-installed office Apple IIE, and pledge never to let these get out of control! Since we have so many types of systems running at once, each of our 5 1/4" diskettes now bears a bright color coded big dot to help prevent us from installing them in the wrong system.

COMPUTER SPEECH FOR THE SYM

We have been comparing the costs and capabilities of two approaches to speech synthesis for the SYM-1. One is the VOTRAX SC-Ø1-A chip, the other is the Speak & Spell (S&S) interface mentioned in earlier issues (VOTRAX we tested on the VIC=2Ø, S&S on the SYM-1).

The costs are roughly equal. The VOTRAX chip does permit a more compact unit, but the S&S interface provides for greater versatility, and besides, the S&S is fun to play with, all by itself!

The inputs to the VOTRAX system are sequences of phoneme code numbers ($ØØ-$3F) to access predetermined phonemes. On the other hand, the inputs to the S&S system are coded sequences for the necessary energy, pitch, and filter parameters to produce as many allophones as desired (allophones are phoneme variants which differ in pitch, inflection, accent, duration, etc).

Studying this approach will provide a deeper insight into what is actually going on during the synthesis process. Also, working at this "lower-level" permits for introducing subtle nuances into the spoken output, including real "singing".

For those who wish to try the S&S approach with their SYM-1s, a complete documentation package is available through the Users' Group. All items described below are by John P. Cater of S.pee.k µP Software.

MANUALS:
"LPC Hardware Manual" - This manual fully describes the theory of operation of the Speak & Spell (including a full schematic!), and provides schematic and construction details for a very simple (three chips - 74165, 74LS175, and NE555 - plus one transistor, one diode, two resistors, and two capacitors) interface between the S&S and only one port of a 6522 VIA. Primitive driver software is included.

"65Ø2 Phonetic Generator Software" - This manual provides more advanced software and a hex dump listing of a phoneme table for the hardware system above. [NOTE: This manual and the manual described below were originally written to accompany Dave Kemp's S&S interface to the SYM-1, which is no longer available, to the best of our knowledge, but the software is easily convertable to Cater's S&S interface.]

"65Ø2 Experimenter Package" - This manual provides still more advanced software and tables of frame data for phonemes, the alphabet, and selected words.

CASSETTE:
"Demonstration Tape" - This is an AUDIO tape which illustrates the capabilities of the system.

BOOK:
"Electronically Speaking: Computer Speech Generation" - An excellent introduction to the theory and practice of voice output. Howard W. Sams & Co., Inc. Paperback.

We have been using the Kemp S&S Interface for several years now. This was a two-way interface, and permitted getting frame data from the S&S ROMs into the SYM's RAM for analysis. As such it was more versatile than the Cater Interface, but more complicated, in hardware, software, and interfacing. Once the analysis is available, and published, as in the manuals above, the two-way feature is no longer a vital necessity. In the future we will be building several of the Cater Interfaces, since only a different connector plug and a VIA address change are necessary in switching the speech synthesis system between 65Ø2 computers!

NEW PRODUCTS

The following new hardware and software products are now available through the SYM Users's Group:

HARDWARE

PRG-1/S EPROM PROGRAMMER - - ALTERNATIVE ENERGY PRODUCTS

COM-1/S DUAL ACIA BOARD - ALTERNATIVE ENERGY PRODUCTS

We'll describe both of the above together, not because they are in any way interdependent, but because of their "common" method of interfacing to the SYM-1.

As you know, part of the power of the SYM-1 is in its built-in I/O capabilities, with two 6522 VIAs and one 6532 RIOT. While this is far more than is available on any other system, we have found that we need much, much, more (we find that we need added I/O far more than added RAM for the kinds of things we do). In any event, the problem was solved for us with the I/OX-122 I/O Expansion Board, which adds up to four additional VIAs in the 1K address space assigned to VIA #2 (Device U28-User Supplied), $A800-$ABFF, and provides additional decoding for other devices, such as the ACIAs on the Dual ACIA Board.

One of our SYMs has an I/OX-122 installed. We run the Epson off VIA #2, a CLK-1/S clock off one of the added VIAs, the PRG-1 from two of the added VIAs, and a cute little "toy", given us as a get-well gift, by Jeff Lavin, a so called "Magic Wand", from the fourth added VIA. The COM-1 mounts on edge fingers on the I/OX-122, and gets its chip selects from the "extra" decoding lines.

What Lavin has done is to provide an integrated approach to adding I/O capability to the SYM-1 which is inexpensive and elegantly simple. [Jeff lent us a beautiful little accessory board for the expansion port of the VIC=20 which contained 11K of RAM (6-2214s + 4-4016s) and two more 6522s. This was a beautiful hand wired package, not a finished, "for sale" product.]

Now that you know how the devices are most simply interfaced to the SYM-1, although other methods may be used, let's describe the devices, themselves:

THE "PROMMER"

The PRG-1/S comes complete with ALL software and ALL hardware, and ALL "personality" modules needed to "burn" the widest variety of (+5V only) EPROMS you might consider using (only the 3-9V alkaline batteries are not included). The software is beautifully "human-engineered", and the programming overhead time is almost trivial, at most a second or two for any size EPROM. No time is spent on "$FF" bytes, either!

So far we have burned only 2716s, a dozen or so, but we expect to try some 2732s next. The best way to illustrate the versatility, and ease of use of the system is to reproduce some of the terminal "dialog". Whenever a prompt is displayed, entry of a "?" for "help" will give you your "menu". Impossible entries are rejected, especially on addressing ranges, where an "explanation" is given. Seldom have we seen a better designed hardware/software package at any price, and NEVER at such a low price as this one.

PRINTED RECORD OF EPROM BURNING SESSION

```
M.T.U. CODOS 1.2
ENTER DATE (EXAMPLE:04-JUL-76)?= 05-MAY-83

]PROMMER
```



```
EPROM PROGRAMER V1.0 COPYRIGHT 1983

ALTERNATIVE ENERGY PRODUCTS

TYPE "?" FOR HELP

EPROM TYPE?        Here we entered:  ?<cr>

1 = 2508
2 = 2516
3 = 2532
4 = 2564
5 = 2758A
6 = 2758B
7 = 2716
8 = 2732
9 = 2732A
A = 2764
B = 27128
C = 27C16
D = 27C32
E = 27C64
F = 68764

 = FOR CURRENT TYPE
↑C = GO TO MON


EPROM TYPE?        Here we entered:  2716<cr>, by mistake!
NOT DEFINED

EPROM TYPE?        Here we entered:  7<cr>

INSERT MODULE # 2716

ADDRESSES?         Here we entered:  ?<cr>

TYPE IN YOUR ADDRESSES IN THIS FORMAT:

PPPP,SSSS,EEEE<CR>

PPPP IS EPROM STARTING ADDRESS
SSSS IS BUFFER STARTING ADDRESS
EEEE IS BUFFER ENDING ADDRESS
<CR> IS A CARRIAGE RETURN

 = FOR CURRENT ADDRESSES
T = GO TO TYPE INPUT
↑C = GO TO MON


ADDRESSES?         Here we entered: 0,2000,3FFF<cr>, to annoy the system!
$2000 BYTES, EPROM END=$1FFF IS TOO HIGH

ADDRESSES?         Here we entered: 0,2000,2FFF<cr>, to annoy the system!
$1000 BYTES, EPROM END=$0FFF IS TOO HIGH

ADDRESSES?         Here we entered: 0,2000,27FF<cr>
$0800 BYTES, EPROM END=$07FF

READY FOR NEXT EPROM

COMMAND?           Here we entered:  ?<cr>
```

```
V = VERIFY EPROM AREA ERASED
N = VERIFY ENTIRE EPROM ERASED
R = READ EPROM INTO BUFFER
C = COMPARE EPROM TO BUFFER
W = WRITE BUFFER INTO EPROM
E = TOGGLE ERROR PRINTOUT
A = GO TO ADDRESS INPUT
T = GO TO TYPE INPUT
↑C = GO TO MON


COMMAND?            Here we entered:  N<cr>

EPROM IS ERASED  $0800 BYTES

READY FOR NEXT EPROM

COMMAND?            Here we entered:  R<cr>

READY FOR NEXT EPROM

COMMAND?            Here we entered:  W<cr>

                    Note that asterisks are printed at regular intervals
PROGRAMING          to inform you that something, at least, is happening.
*****************************************************
COMPARING           A "compare" is routinely made, and an error report is
                    automatically given.

NO  ERRORS $0800 BYTES

READY FOR NEXT EPROM

COMMAND?            Here we entered:  ↑C<cr>
0470,3
.
```

THE "COMMUNICATIONS" CARD -- COM-1

We have not yet had the time to do more than read over the spec sheets on this device and check over the physical unit, but we have been kept well posted as to the progress of the product development. As usual with AEP products, we know we will soon wonder how we ever got along without it. We plan to have it "up-and-running" within a day or two after this issue goes into the mail.

While the SYM-1 has both a 20 mA current loop and an (inverted TTL equivalent) RS-232-C interface, we have often felt the need for a second RS-232-C channel for modem use. Now, we even feel the need for a third, to interface with our VIC=20 and CBM-64. We're therefore especially pleased to get two-in-one with this new card, and at just the right time, too. [We prefer to leave the 20 mA current loop intact, because our decwriter II (with 20 mA card) can then be switched from system to system for hard copy without requiring a special printer patch, by simply using a ".J 1" to switch to TTY Input/Output, at 110 baud.]

We print below a few extracts from the seven pages of documentation (including a source code listing of the required software driver) supplied with each unit to give you some ideas on both its use and the thoroughness of the documentation:

> The COM-1 is a serial communication board designed to perform, in hardware, the I/O functions previously executed in software. It is especially important to relegate this task to hardware when using data links (a modem for example). The COM-1 supports all asynchronous serial communication (RS232), uses a crystal controlled clock to generate all standard baud rates from 50 to 19,200 and may be used at non-standard baud rates with an external clock. The actual parallel/serial conversion is done by two 6551 ACIAs, providing two full duplex I/O channels. This configuration eliminates much overhead for the computer and allows I/O to proceed much faster than when done in software. As received from the factory, the COM-1 comes with Line Receivers, and is set up to transmit TTL level signals, but has the capacity to support RS232C with the addition of Line Drivers (plug compatible) and an external source of +/- 12V. This board is specifically designed to interface to our I/O Expansion Board, but may be adapted to other installations.
>
> * * * * * * * * * * * *
>
> As previously mentioned, the COM-1 comes from the factory equipped with Quad Line Recievers. This is done so that, if it is inadvertantly connected to equipment operating at RS232C voltage levels, the COM-1 would not be damaged. The transmit section employs 74LS00 ICs, which are plug compatible with Quad Line Drivers, but transmit TTL level signals (+5V and ground). There are few modern data communication devices employing RS232C specification that will not work with TTL level signals. However, some older pieces of equipment may need the different voltage levels to function properly. If RS232C operation is desired, three wires from the power supply must be brought to the three pads located between the two I/O connectors.
>
> * * * * * * * * * * * *
>
> The simplest method of serial communication is the 3-wire interface (see Fig. 3a). A 3-wire interface provides transmit data, recieve data, and a signal ground. It does not provide for handshaking. The effect of this is that both ends transmit blindly - with no indication that the reciever is recieving or, in fact, is there at all. The ACIA handily overcomes this problem by providing for handshaking signals
>
> * * * * * * * * * * * *
>
> It is not possible in this small user's manual to fully describe the RS232(C) specifications; our intent is to give you enough information to be able to intelligently connect and use the COM-1 serial communication board. If you are unfamiliar with the terms used in this discussion, turn to Appendix B for a glossary.
>
> The COM-1 has two complete and separate full duplex communication channels that are compatible with the RS232 specification. Each channel can transmit and receive at a user definable baud rate and format simultaneously. In the programming section, we will describe how to select these formats. The ACIAs handle parallel/serial and serial/parallel conversion, communications control (handshaking), and detection of overrun, framing and parity errors. The ACIAs can also be used for interrupt driven I/O. The outputs from the ACIAs are buffered and inverted by TTL (or Line Drivers - user installed option) and the inputs to the ACIAs are buffered and inverted by Line Receivers. The RS232 standard defines two types of communications equipment: Data Set and Data Terminal. These designations determine the connections to the standard DB-25 connector BY POSITION. For example, pin #2 is defined as signal BA and described as "data from terminal". This means that if the equipment were a Data Terminal, this line would be an output; if the equipment were a Data Set, the line would be an input. The COM-1 is configured as a Data Terminal.

SOFTWARE

ELIZA -- JEFF LAVIN

ELIZA is the, by now, "classical", public domained, AI (Artificial Intelligence) demonstration program originally written in LISP (LISt Processor), by Joseph Weitzenbaum of MIT to emulate a "human" psychoanalyst. (We understand that Professor Weitzenbaum now regrets having published ELIZA beacuse of its "misuse" by those who allege that the program "proves" that machines can be programmed to "think".)

According to Turing, a "system" demonstrates AI if a user cannot be sure whether he is dealing with a "man" or a "machine". Based on Turing's criterion, ELIZA is "intelligent", since whenever we deal with "her" we find ourself getting as emotionally involved and as frustrated with her probing questions and occasional evasive "behaviour" as we probably might feel when dealing with a real "shrink".

Jeff Lavin has prepared a truly delightful SYM-1 version of ELIZA, written wholly in 65Ø2 ML code. You will need at least 12K to hold the object code. Lots of RAM is required to store the large vocabulary at ELIZA's command. Only object code will be provided initially, on either cassette or FDC-1 diskette. RAE-1 source code will be available (requires 32K) in the near future, again, in both media.

FORTH FOR THE FDC-1 -- BILL WHARRIE

This is a full implementation of fig-FORTH, completely integrated with the FDC-1 system. It will be supplied either on 5 1/4" FDC-1 diskettes, 1Ø24 byte per sector, double density, format, or, for those with 8" systems, on cassette (perhaps by the time you are ready for FORTH, we will have completed our arrangements to have Joe Hobart generate 8" disk copies). A variety of FORTH utility "SCREENS" will also be provided.

Below is a copy of its "VLIST" for your evaluation. This is followed by (partial) "VLIST"s for the EDITOR and ASSEMBLER VOCABULARIES. Note the "conditionals" built into ASSEMBLER, to permit "structured" programming. If you like FORTH, you'll LOVE Bill's FDC-1 implementation! We're going to install an FDC-1 controller on our SUPER-SYM with this FORTH as our main language.

```
.G 9006

.L3
FORTH2,1
.G 200

FIG-FORTH  1.0
VLIST
CODE   ASSEMBLER   2SWAP   2DUP   2DROP   WHERE   EDITOR   LINE
TEXT   W-START   C-START   ECHO-OFF   U<   8000-   COLD   (R-V)
REPLACED.BY   WORD.IN   U*   R/W   ERRCNT   DISKIO   CASSETTE   DISK
FLAGS   BUFAD   SEC#   TRK#   UNIT#   D/C   CSAVE   CLOAD   CLMSG
MON   VLIST   TRIAD   INDEX   LIST   ?   .   .R   D.   D.R   #S
#   SIGN   #>   <#   SPACES   WHILE   ELSE   IF   REPEAT   AGAIN
END   UNTIL   +LOOP   LOOP   DO   THEN   ENDIF   BEGIN   BACK   FORGET
'   LOADC   R/W   -BCD   SAVE   -->   LOAD   MESSAGE   .LINE   (LINE)
BLOCK   BUFFER   DR1   DRO   EMPTY-BUFFERS   FLUSH   UPDATE   +BUF
PREV   USE   M/MOD   */   */MOD   MOD   /   /MOD   *   M/   M*
MAX   MIN   DABS   ABS   D+-   +-   S->D   COLD   ABORT   QUIT
(   DEFINITIONS   FORTH   VOCABULARY   IMMEDIATE   INTERPRET   ?STACK
DLITERAL   LITERAL   [COMPILE]   CREATE   ID.   ERROR   (ABORT)
-FIND   NUMBER   (NUMBER)   UPPER   WORD   PAD   HOLD   BLANKS
ERASE   FILL   QUERY   EXPECT   ."   (.")   -TRAILING   TYPE
COUNT   DOES>   <BUILDS   ;CODE   (;CODE)   DECIMAL   HEX   SMUDGE
]   [   COMPILE   ?LOADING   ?CSP   ?PAIRS   ?EXEC   ?COMP   ?ERROR
!CSP   PFA   NFA   CFA   LFA   LATEST   TRAVERSE   -DUP   SPACE
ROT   >   <   U<   =   -   C,   ,   ALLOT   HERE   2+   1+   HLD
R#   CSP   FLD   DPL   BASE   STATE   CURRENT   CONTEXT   OFFSET
SCR   OUT   IN   BLK   VOC-LINK   DP   FENCE   WARNING   WIDTH
TIB   +ORIGIN   B/SCR   B/BUF   LIMIT   FIRST   C/L   BL   3   2
1   0   USER   VARIABLE   CONSTANT   ;   :   C!   !   C@   @   TOGGLE
+!   DUP   SWAP   DROP   OVER   DMINUS   MINUS   D+   +   0<   0=
R   R>   >R   LEAVE   ;S   RP!   SP!   SP@   XOR   OR   AND   U/
U*   CMOVE   CR   ?TERMINAL   KEY   EMIT   ENCLOSE   (FIND)   DIGIT
I   (DO)   (+LOOP)   (LOOP)   0BRANCH   BRANCH   EXECUTE   CLIT
LIT   OK
```



```
EDITOR OK
VLIST
UNDER   NEW   .BS   NULL?   ENTER   ENTER?   TILL   X   B   F   N
C   DELETE   FIND   1LINE   MATCH   -TEXT   COPY   CLEAR   TOP
I   P   R   L   T   M   D   S   E   H   -MOVE   #LAG   #LEAD   #LOCATE

ASSEMBLER OK
VLIST
END-CODE   >=   0<   0=   CS   NOT   ELSE,   THEN,   IF,   UNTIL,
BEGIN,   BIT,   JMP,   JSR,   STY,   LDY,   LDX,   CPY,   CPX,
STX,   ROR,   ROL,   LSR,   INC,   DEC,   ASL,   STA,   SBC,   ORA,
LDA,   EOR,   CMP,   AND,   ADC,   M/CPU   TYA,   TXS,   TXA,   TSX,
TAY,   TAX,   SEI,   SED,   SEC,   RTS,   RTI,   PLP,   PLA,   PHP,
PHA,   NOP,   INY,   INX,   DEY,   DEX,   CLV,   CLI,   CLD,   CLC,
BRK,   CPU   UPMODE   RP)   SEC   BOT   )   )Y   X)   ,Y   ,X   MEM
#   .A   MODE   INDEX   SETUP   NEXT   PUSHOA   PUSH   PUT   POPTWO
POP   N   IP   UP   W   XSAVE
```

P.S. For those of you with at least 24K of RAM and no FDC-1 as yet, note that this FORTH can ALSO be used on a CASSETTE based system. Full instructions for modifying the object code are provided. You can get started on the cassette version and add the FDC-1 later. Actually, both cassette and FDC-1 can be used interchangeably. Note that the FORTH words DISK and CASSETTE appear in the FORTH VOCABULARY. These are used to select the desired I/O medium. Just be sure to specify that you need the cassette format.

HELICOPTER -- DANIEL WUETHRICH

This is another interactive video graphics game by the author of SYMMAN. Like SYMMAN, it requires a Visible Memory and an "Atari" compatible joystick. Supplied as RAE source code on cassette. Requires 32K for assembly.

We found this to be even more fun than SYMMAN. Here are the rules, as extracted from the game "manual":

Move the helicopter with the joystick. Pressing the ACTION button makes the helicopter fire. Down on the ground gas tanks and enemy bases are generated by random control, slowly at the beginning and then faster and faster. Hitting one of the bases counts the following points:

- small base : 2Ø points
- medium base: 1Ø points
- large base : 5 points

The bases fire at you as you fly overhead, attempting to dodge (U, D, L or R) their fire, while firing at them in return.

Your helicopter uses 2 units of gas per second. You start the game with an initial 1ØØ units. Getting more gas is done by touching a gas tank on the ground with your helicopter. This gives 1 to 2Ø units of gas, according to how full the gas tank is and how fast the game is already. Because the gas tanks have holes, the gas flows out in about 2Ø seconds. Hitting a gas tank counts points from Ø (full tank) to 5 (empty tank). An empty gas tank is removed automatically after 4 seconds.

You start the game with 5 lives. One life is lost when the helicopter is hit or when you run out of gas. Each time you lose a life, you get an additional 2Ø units of gas. If high-score is reached "??????????????" is displayed. Now enter your name and fill with spaces (no CR or LF).

If you wish to save the high-score and the name after the game, then simply save the whole program back to disk or cassette.

SWP 2.5 -- A. M. "SANDY" MACKAY

SWP-1 has been the most popular word processor for the SYM-1. It is essentially a text FORMATTER for text files edited under RAE-1. At the time it was initially released there were a number of known "weaknesses". The demand for a word processor was so urgent that it was released "as-is", without a real user manual, with only a sample text file and the fully commented source code to guide the purchaser in its use.

Because all users had RAE-1 installed, and hence had a reasonable knowledge of 65Ø2 assembly language, they were able to "figure-out" the workings from a study of the source code. This knowledge led many of them to customize SWP-1 to fully meet their own personal requirements. We sent a copy of our own "upgrading" to Sandy Mackay as "SWP-2", and he returned it to us, with further embellishments, as "SWP-2.5". The weaknesses of SWP-1 have been removed, and a number of new features added.

It is so much stronger than SWP-1 that we are making it available as an added cost option to all past and future purchasers of SWP-1.

SOME EXPANDING IDEAS - JAMES E. TRUESDALE

April 1, 1983

Dear Lux:

I just expanded my Sym-1 to 32K of RAM for less money than anything else that I have seen for the Sym or it's relatives. I thought that you and other Symmers might be interested in hearing about it.

I bought John Bell Engineering's 81-330 RAM/EPROM Memory Board and built it myself. Here are a few of the board's features. The board is PIN FOR PIN compatable with the Sym's expansion connector, all I had to do was wire up the connectors. It uses 6116 Rams (2K X 8) and/or 2716 EPROMS in any combination. 6116 Rams are getting pretty cheap now, I've seen them for $4.28 each. All lines are buffered (I've had NO problems), and the board only draws 500ma at 5v. The board is a standard size of 4.5" X 6.5" and has a gold card edge connector. It also seems to fit ok in my father's MTU card cage for his Kim.

I built the board in a few hours and it worked the first time that I tried it (after I hooked it up to the expansion connector instead of the applications connector of the Sym. Boy was THAT a debugging problem! What one will do when one is in a hurry!).

The cost breakdown looks like this -

1	74LS244	1.50	1.50
1	74LS245	1.50	1.50
1	74LS10	.35	.35
1	74LS365	.50	.50
2	74LS138	1.00	2.00
3	16 Pin IC Sockets	.75	2.25
1	14 Pin IC Socket	.20	.20
2	20 Pin IC Sockets	N/C	N/C
16	24 Pin IC Sockets	.40	6.40

10	Monolithic .1 mfd. Caps.	.12	1.20
16	6116 150ns Memory Chips	4.38	70.08
1	32K Memory Board	52.45	52.45
		Total	$138.43

I bought the 24 Pin sockets at a local electronics junk house and the rest of the extra chips, caps, and sockets from my father or from my junk box. I used monolithic caps because they take up less space than standard disc caps.

I bought the 6116 chips from Microprocessors Unlimited in Beggs, Oklahoma. They are FAST and reliable and sell only top quality chips. We ordered these chips over the phone on a Sunday and had them the following Friday. Since we had ordered from them before, they just billed us. Our first order was by credit card, and was equally fast. They advertise in The Computer Shopper, but call for the latest prices since they change so fast.

Enclosed is a copy of some literature for a connector that I bought for the memory board that I am going to use to build a "card cage" (The MTU card cage is WAY to expensive for me) for the memory board and the Sym (I will use standard connectors for Sym). I intend to mount them both vertically and put them either inside of my surplus CRT terminal that I use as a monitor for my KTM-2/80, or else mount them free standing behind the terminal.

Sorry that this letter isn't in RAE format, but I composed this letter on my father's Radio Shack Color Computer using the Telewriter-64 Text Editor. It is just sooooo neat! I printed it on my surplus GE Terminet 300 Terminal. The Co-Co is really an impressive machine even with this funky keyboard.

Well, I just wanted to tell you and other Symmers about this (in my opinion) great way to expand a Sym for less.

Sincerely,

James E. Truesdale

James E. Truesdale
1400 Hudson Road
Ferguson, MO 63135

[EDITOR'S NOTE]

We have discovered a way to create camera-ready copy from materials typed with old, tired ribbons. We copy them on our office copier with the control set to darken the copy. We go through several generations until the contrast is sufficiently enhanced. Image quality is not degraded, since the electrostatic copying process inherently provides "edge enhancement". Unfortunately, the process does not incorporate spelling or grammar correcting features, so a RAE readable tape is still preferable.

RAE .CT PROBLEMS

A number of readers have had problems with the .CT pseudo-op "bug" in RAE-1. The first printing of the RAE-1 Reference Manual provided the correct fix (a patch in page zero) but all later printings put the patch in page $ØE. This is OK for a 4K SYM-1, but the patch conflicts with text or label files which extend beyond the original 4K of RAM. You may wish to correct page 1Ø-2 of your RAE-1 Reference Manual to read as follows:

The patch shown below is placed ~~at the end of the default label file.~~ in page zero.

LOCATION	CONTENT	COMMENT
EE	01	Enter flag
F6	~~F5~~ A0	Enter vector to
F7	~~0C~~ 00	. . .patch
~~EF5~~ A0	~~AD~~ A9	Patch is 3
~~EF6~~ A1	~~11~~ 00	. . .instructions
~~EF7~~	~~01~~	
~~EF8~~	~~D0~~	
~~EF9~~	~~03~~	
~~EFA~~ A2	8D	Store 0 into
~~EFB~~ A3	10	location $110
~~EFC~~ A4	01	
~~EFD~~ A5	4C	Jump back into
~~EFE~~ A6	68	RAE-1
~~EFF~~ A7	EF	

To install the patch, perform the following:

1. Enter RAE-1 — Type: G B000
2. Exit RAE — Type: BR
3. Use M command three times to modify EE, F6-F7, and ~~EF8-EFF~~ A0-A7

LOTS OF IDEAS FROM HARRY FORR

Here are some extracts from a recent letter which describe several useful modifications to SUPERMON, implemented by replacing the original 2332 ROM with your own 2532/2732 EPROM. The major modification is to a 9600 baud CRT data rate.

Harry also describes a simple current loop to RS-232-C "converter". We haven't studied his mods enough to figure out why they produce the loss of RAE-1's CTRL C and BRK exits to SUPERMON. His reference to the RU $9003 "fixing" the problem, is based on this being the RAE-1 "patch" to FDC-1, and this patch does modify a goodly number of vectors.

Dear Lux:

In the last issue of SYM-PHYSIS there was a little gem tucked away on 13/14-0 and 331. *** Modified Supermon *** by Paul L. Beaupre .

This was all the help I needed to finish my "System patch" converting my SYM 1 to run communcations at 9600 baud. I had been altering Supermon to allow lower case in basic by NOPing out the AND #$DF now it is just a #$7F and the old AND command. But now, MY dream of a 9600 baud system has taken shape.

Modifications include:

1. TTY port becomes a printer port with DTR. (DTR line is not checked if printer flag "TOUTFL" is not set.)

2. 1 stop bit instead of 2 [note: SYM 1 documentation error, page 26 of the SUPERMON PROGRAM states "start bit, 8 Data, 3 Stops" but the Zero loop is not executed, therefore ... 2 stops.]

3. Default value changed to start up I/O CRT only.

4. Lower case enable to BASIC.

5. Control O toggles on/off output to printer.

Now that the sales pitch is over, there is a bug. (isn't there always?) When first entering RAE with a .G B000 cr., the control C (ctrl c) to exit to the monitor will not work. Nor will the BRK command function. This problem went unnoticed for awhile since the cure for the bug is RU $9003 cr. I have had no problems in BASIC. Poking a 144 (CRT only) or 160 (printer only) into 42580 ($A654 TOUTFL) turns the printer on and off, leaving the break key enabled on the CRT.

Like Mr. Beaupre I have been burning an EPROM (2532) and then just replacing the monitor chip. References are made in the program for moving the object code to the buffer I use to program the EPROM.

I have included (separate page) a copy of the hardware modification used to bring the TTY port around to a CRT way of thinking. For the inverter, I used a 4049 CMOS inverter which allows up to 18 volt inputs with 5 volt (vdd level) output. I mount this inverter external to the SYM 1 in the break out box built to house the CRT port connector, the PRINTER port connector (or second CRT port), and tape I/O.

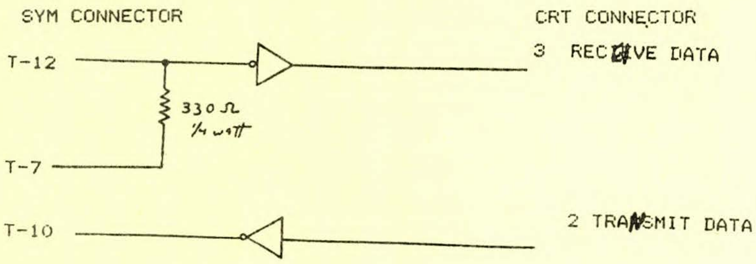
SYMcerely,

H.J.Forr Jr.
CW Industries

```
>ASSEMBLED LISTING:
              ;     9600 BAUD FIX WITH PRINTER DATA TRANSFER REQUEST [DTR]
              ;
              ;                 H.J.FORR JR.
              ;                 12/14/82
              ;
              ;
              ;
                    .OS
              ;
              ;
              ;     PROGRAM DEFINITIONS :
              ;
              DTR      .DE $40
              PBDA     .DE $A402
              PRINTER  .DE $20
              SAVER    .DE $8188
              TOUTFL   .DE $A654
              ;
              ;
              ;
              ;-------------------------------------------
              ;
              ;     DON'T CONVERT TO UPPER CASE.
              ;
                    .BA $8A2C
                    .MC $1A2C
              ;
8A2C- 7F            .BY $7F
              ;
              ;-------------------------------------------
              ;
              ;     TURN PRINTER ON WITH CONTROL-O
              ;
                    .BA $8A31
                    .MC $1A31
              ;
8A31- AD 54 A6      LDA TOUTFL
8A34- 49 20         EOR #PRINTER        ;TOGGLE PRINTER !
8A36- 8D 54 A6      STA TOUTFL
              ;
              ;-------------------------------------------
              ;
              ;     INTERRUPT "TOUT" TO WAIT FOR DTR.
              ;
                    .BA $8AA2
                    .MC $1AA2       ;EPROM BURNING BUFFER.
              ;
8AA2- 4C EB 8A      JMP WAIT?
              ;
              ;
              ;-------------------------------------------
              ;
              ;     CHANGE NUMBER OF STOP BITS TO 1.
              ;
                    .BA $8AB0
                    .MC $1AB0
              ;
8AB0- 0A            .BY $0A       ; 1 START, 8 DATA, 1 STOP BIT(S).
              ;
              ;-------------------------------------------
              ;
              ;     NEW DELAY HALF [DLYH]  FOR 9600 BAUD RATE.
              ;
                    .BA $8AE9
                    .MC $1AE9       ;EPROM BURNING BUFFER.
              ;
8AE9- EA            NOP
8AEA- 60            RTS
              ;
              ;
              ;-------------------------------------------
              ;
              ;     PRINTER DELAY DATA TRANSFER REQUEST  [DTR]
              ;
8AEB- AD 54 A6 WAIT?    LDA TOUTFL
8AEE- 29 20             AND #PRINTER        ;PRINTER ON ?
8AF0- F0 07             BEQ DONEWAIT        ;NO, SO GO.
8AF2- AD 02 A4 WAIT     LDA PBDA
8AF5- 29 40             AND #DTR
8AF7- F0 F9             BEQ WAIT
8AF9- 20 88 81          JSR SAVER
8AFC- 4C A5 8A DONEWAIT JMP $8AA5       ;WE RETURN YOU TO "TOUT" ALREADY IN PROGRESS.
              ;
              ;
              ;-------------------------------------------
              ;
              ;     CHANGE DEFAULT I/O TO PRINTER OFF.
              ;
                    .BA $8FD4   ;TOUTFL DEFAULT
                    .MC $1FD4
              ;
8FD4- 90            .BY $90     ;"CRT" ONLY.
              ;
              ;
              ;-------------------------------------------
              ;
              ;     END  9600 BAUD NEW CODE.
              ;     COPY MONITOR TO EPROM BUFFER [ BMOVE $1000-$8000-$8FFF ]
              ;     THEN ASSEMBLE NEW CODE.
              ;
                    .EN
```

HARDWARE MODIFICATIONS

THESE HARDWARE MODIFICATIONS WILL GIVE THE SYM-1 A SECOND "CRT" PORT.




```
T-1----------------------------------------------1 GROUND
T-6----------------------------------------------6 DATA SET READY*
T-7----------------------------------------------7 LOGIC GROUND
T-7---------------- GATE GND PIN #8
T-5---------------- GATE VDD PIN #1
```

*Also; #8 Carrier Detect

RAE.DOS AND RELATED TOPICS

Many months ago Jack Brown (Saturn Software) sent us a collection of five diskettes with a note saying "Here is some software to play with!" Two manuals, entitled "RAE.DOS" and "MEAN14" came along with the package. We really did have fun following his suggestion.

"MEAN14" we have described earlier, but "RAE.DOS" is really something else! It is a truly elegant DOS designed to supercede FODS, but does require the HDE disk controller and the FODS bootstrap loader to get it operational. Jack provides a special BOOT disk running under FODS and the FODS boot to load-in and execute RAE.DOS. The BOOT disk is then removed and from that point on only RAE.DOS generated disks are used.

We booted up as per instructions, and came up in what, at first glance, appeared to be RAE, and can, in effect, be treated as RAE. An examination of the accompanying manual showed however, that this was now RAE with a powerful new line editor and a truly elegant new DOS, with a very versatile and "user-friendly" command structure.

We then removed the BOOT diskette from the System Drive and replace it with the RAE.DOS UTILITY disk, which contained all sorts of "goodies", in both .OBJ (machine language run-time code) and .TXT (RAE source code form). The other three diskettes contained source and object code for RAE.DOS itself, MEAN14, etc, etc, etc.

The entire package was a real pleasure to use and examine. RAE.DOS is one of the best software development packages we have ever seen. We commend it to all FODS users. It was with regret that we put it away, never to look at it again until today. The reason we set it aside? . . . . Because it is difficult to "shift" mental "gears" between DOSes, and we are already having enough problems remaining proficient in CODOS, FODS, and FDC-1 simultaneously.

Why are we looking at it again? . . . . Because we received a RAE.DOS diskette today from one of our long-time readers. We reprint portions of his letter below for general interest, and also a few samples of his printer outputs, so that you can see its versatility (he forgot to set >FO C before printing!).

P.O.Box 257,
Lindfield,
N.S.W. 2070.
Australia.
15.April, 1983.

Dear Jean and Lux,

For some time I've been looking around for a second computer but finding it very difficult to make up my mind as to which one it should be. I'm very attracted to the BBC but am a bit disappointed that some of the add-ons are so slow in appearing. Also that, with the exception of games programs, there is not much to run on it that can be bought off the shelf. And that is mainly why I'm thinking of another machine - to have access to ready made programs, particularly of the VisiCalc kind. I can have all the programming (and hardware) fun I can find time for with the SYM.

My main purpose in getting in touch with you at this time is to send you this diskette. It is probably of little use for "SYMPHYSIS", partly because it contains a number of routines from Jeff Holtzman's "MONEX/SYM-BUG", (although whether or not he would mind I don't know), partly because the MX80FT III printer routines are probably not compatible with the EPSON sold in your country with the same model number, (I know they differ but I don't know how) and also because of its hardware requirements. However I thought you, personally, might find some of it interesting.

You may also be interested in a few details of the extensions I've added to the SYM lately and which are used by the programs on this diskette. An additional 6532 has ben added at $A500 with its RAM at $A700-$A77F. This is mounted on a separate board with room for several more I/O chips. The processor, a 6502A, has been removed from the its usual position and relocated on another board where its data and address lines are buffered, and which also has decoding and bank switching logic for four banks of (hardware) switch selectable RAM (6116s) or ROM (2716s) at $9000 - $97FF. There are also 6116s at $9800 - $9FFF and $F000 - $F7FF. Later I hope to replace the 2716s with 2532s and to have both SWP and XRF in the one chip. XRF will be called in somewhat the same way as SWP is at present.

I've started on a board to put RAM at $B000 - $EFFF and hope to finish it before too long. However I'm continually distracted by playing with FORTH. I wonder whether you've tried Leo Brodie's 'Quick Text Formatter' described fairly recently in 'FORTH DIMENSIONS' ? Its really magic to be able to add words to meet special requirements just as one needs them.

Just in case you don't have RAE.DOS readily available I'll print this letter and enclose it with the diskette.

With very best wishes to you both,

...Maurice.....
M.A.Du Feu

I've started on a board to put RAM at $B000 - $EFFF and
hope to finish it before too long. However I'm continually
distracted by playing with FORTH. I wonder whether you've
tried Leo Brodie's 'Quick Text Formatter' described
fairly recently in 'FORTH DIMENSIONS' ? Its really magic
to be able to add words to meet special requirements
just as one needs them.

Just in case you don't have RAE.DOS readily available
I'll print this letter and enclose it with the diskette.

are you both? Well I trust, and enjoyi
of you recent eye operations Lux.

are going along fine here and lately I
time to spend with the SYM.

For some time I've been looking around
but finding it very difficult to make
one it should be. I'm very attracted to
disappointed that some of the add-ons

Also that, with the exception of games
much to run on it that can be bought o

they differ but I don't know how)
and also because of its hardware
requirements. However I thought you,
of it interesting.

...continued
26 73 63 3 ~left margin
3 70 63 3 ~left margin
0 50

ROBOTICS

Quite a few of our readers are heavily into robotics. Several have sent us photographs and reprints of technical articles which they have had published elsewhere. We list below their names and addresses and the names of their robots, so that your robots may correspond directly with theirs!

LCDR BART EVERETT, Assistant for Robotics, (SEA-90M3), Naval Sea Systems Command, Washington, DC 20362, sent an 8x10 (non-autographed!) glossy of "ROBART", whose specs, particulary in the sensor area, are very impressive. ROBART could easily serve as a night watchman, on the lookout for intruders, fire, smoke, floods, etc.

GENE OLDFIELD, Robot Repair, 816 1/2 21st Street, Sacramento, CA 95814, sent us similar information on "ENTROPY". Since ENTROPY "lives" only some 90 miles from us, we hope to visit him (her?) early this summer.

RICK KIRSCHBROWN, 595 Hunter Lake Drive, Reno, NV 89509, sent us a color photo of "HOMER" (HOME Robot). Rick was a student at CSUC several years ago.

JIM GRAHAM, a current student, and our Lab Assistant, at CSUC, is working on an as yet un-named robot based on the Milton Bradley toy "Big Track" as the "vehicle" and the Polaroid Camera Ultrasonic Rangefinder as the principal sensor. The idea is to use the little "beastie" to map out strange rooms. We will keep you posted on the progress of this one.

RAM-BLINGS

First a few personal notes for those who were kind enough to write and ask: The eye problems are finally resolved. Didn't get a wide-angle lens implant in one eye and a telephoto in the other (medical technology is not at that point, yet) but one eye is set for near vision, the other for far, so that I can drive or read without glasses by mental selection of the "dominant" eye. With bifocals both eyes are 20/20, and I can actually see well enough to solder again. The muscles which change the shape of the natural lens for focusing are now "in training" to move the plastic implants to-and-fro for focusing.

Now that my vision is back to the days of my youth, I am tempted to have a rather distorted right wrist, badly shattered in a fall from a bar stool (no, I was sober, and standing on it to reach a high shelf) some years ago, rebuilt, to restore its "youthful" dexterity. No it is not a hang-up on youth, but it would be nice to regain the hand-eye coordination necessary to be a high scorer in SYMMAN, HELICOPTER, and the arcade type games on the VIC=20 and CBM-64. It is very frustrating to have nearly everyone I know able to beat my "lifetime high scores" after only a few minutes of practice.

As usual, we have fallen behind in answering the mail, and getting the newsletter out on time; for this we apologize again. At last, though, we do see a solution, beginning next year. We will retire from our teaching position, to become the Computer Science Department's first Professor Emeritus, effective 1 June, 1983. We will continue to teach one semester (fall) each academic year, but will then have eight months free each year for travel and personal research. We hope to be able to visit many of our European readers next spring.

We have no scheduled lectures or teaching assignments this summer, so that we will have a full "uninterrupted" three months to get caught up on unanswered mail and unfinished projects. We plan that the 1984 volume of SYM-PHYSIS will include most of the software and articles that have backlogged on us, and will start "organizing" for that this summer.

PRODUCT REVIEWS

A 64K MEMORY BOARD WITH BANK SWITCHING

Bob Peck sent us, for evaluation and review, a sample of the 64K DRAM (Dynamic RAM) Board he is marketing for the SYM-1, SYM-2, and the AIM 65. It is a very well designed, compact package, using 8 OKI M3764-20RS DRAMs, a Motorola MC6883 as the "main" chip, sockets for a pair of 2114s for the lowest 1K of RAM (since thia area may NOT be bank switched), buffers, a handful of TTLs and a "customizing" PROM (for either AIM or SYM memory maps). It also includes a 16.000 MHz crystal (which in effect replaces the SYM's 1.000 crystal.

The board is installed extending out from the Expansion Connector (or it may be tucked under). The SYM's 6502 and all on-board RAM are removed. The 6502 is reinstalled in a special header socket cabled from the DRAM board to get the new clock signal.

The "new" memory map is as shown below, with bank switching accomplished with the machine language sequences indicated. To initialize the system, log on, then .G 7000. SYM will then respond "64K ONLINE!" with a blinking cursor, waiting for a second log-on. It's quite a thrill to see this, almost unbelievable!

```
FFFF   +-----------------+
       !                 !
       !   ROM + I/O     !
       !                 !
       !     SPACE       !
8000   !-----------------!        +---------------+
7FFF   !    31K RAM      !        !    31K RAM    !
       !                 !        !               !
       !     BANK        !        !     BANK      !
       !                 ! (===)  !               !
       !       0         !        !       1       !
0400   +-----------------+        +---------------+
03FF   ! 1K STATIC (2114)!
0000   +-----------------+

LDA $FFD5                       LDA $FFD4
STA $FFD5  ;switch to bank 1    STA $FFD4  ;switch to bank 0
```

A retrofit kit is being planned for this card to provide Motorola 6847 Color Graphics. This will require installing a 14.3818 MHz Crystal in place of the 16.000 MHz one, but a "replacement" SUPERMON EPROM will also be supplied correcting all time dependent parameters to conform to the 12% slower clock rate.

The board comes with a well written Installation and User Manual, and is one of a new line of products Bob's company, BYTE Microsystems Corporation, of Sunnyvale, CA, is introducing for the SYM/KIM/AIM family.

PROGRAMS BY TOM GETTYS

Tom Gettys wrote us recently that he has been looking over his collection of programs for the SYM-1. He sent us quite a few, two of which appear in this issue, and is "polishing" them up for distribution. Write to him directly at the address below for a listing of programs available, and prices for either cassette or FODS diskette versions.

His programs include utilities, such as COMPACT, which removes spaces and REMs, from BASIC programs, games such as "GAME OF LIFE", and a wide variety of applications programs which he developed for his own use, and for use in teaching. His "catalog" includes programs in both BASIC and 65Ø2 Source Code. He prepared for Jean's use an ACCOUNTS PAYABLE program (running under Saturn Software's Extended Disk BASIC) for handling some of her book-keeping chores.

Tom Gettys, 4539 Beachcomber Court, Boulder, CO 8Ø3Ø1

COLORMATE II BY MICROMATE

Dick Turpin, of MicroMate, has been at the Unversity of California, Davis (UCD), on sabbatical from his home campus, for the past year. During a recent visit he showed us the spec sheets for a new product which should be available early this fall.

It will incorporate an INTEL 8Ø31 single-chip microprocessor for serial interface to the host computer, with custom firmware in EPROM. It will also include two GI AY-3-891Ø Programmable Sound Generators, a National Semiconductor ADCØ8Ø9-based fast (1ØØ usec) 8-bit, 8-channel multiplexed A/D conversion subsystem, twenty I/O lines, and last but not least, extra-ordinary color graphics, as follows:

> A Texas Instruments 9918A Video Display Processor supports four modes of color video ranging from twenty-four 40-character rows of text to 256 X 192 resolution graphics, with 15 unique colors plus transparent, 35 display planes, and 32 sprites. 16K bytes dynamic RAM are dedicated to the video display. The output is composite video.

Contact MicroMate at P.O. Box 5Ø111, Indianapolis, IN 4625Ø, for further information on the ColorMate II.

PROGRAM CORRECTION
------- ----------

Bob Peck informs us that the "FORCED CASSETTE TAPE READ ROUTINE" on page 13/14-57,58 is missing the following line:

```
Ø265    BNE    INCDUN
```

Fortunately, the error and fix are sufficiently obvious that most readers spotted it at once, so little damage done!

A HARDWARE NOTE
- -------- ----

MILES E. ANDERSON, KB5UW, passes along the following suggestion to make ROM/EPROM interchanging less painful:

> A HARDWARE NOTE. If all the ROM addressing jumpers 1-18, 46-47, A-M are removed from the SYM board and the holes cleaned with a solder sucker, the board will then accept two 16-pin DIP sockets. Headers in these sockets will permit endless jumper changes without danger of damage to the board. I made up separate sets for the two-chip versions of BAS and RAE and can now switch from one to the other in less than half a minute. An 8-pin socket to the left of the crystal will provide similar flexibility in write-protect changes. This socket scheme is not original. My son, David, (also a Symmer) suggested it to me.

A CALL FOR HELP
- ---- --- ----

We reprint in the next column portions of a letter that we did not have time to answer in the detail it deserved. Can any of our European (PAL/SECAM areas) readers help provide the answers to their questions? Thanks, if you will.

Moorsele, 6 March 1983

Dear Lux:

We are three computer amateurs (or should it be amateur computerists?) and we would very much welcome it if you could answer a few questions:

Our main problem is the following: We are designing and building our own 65Ø2 based computer system and we have been looking for a suitable video controller. So far we haven't had any success. (One thing we found was the AMI S68Ø47 VDG, video display generator, from the magazine Microcomputing, February 198Ø, out this chip doesn't seem to be available in Europe. Would it be possible to purchase it through the SYM-1 Users' Group?)

We would like to know if you could tell us about any system that has at least 8 color capability and preferably a 256 by 256 dot display (of course 65Ø2 compatible). We have been looking for information all over Belgium, but we didn't find anything. You're our only hope. So please send us information about a Color Video Display system we could build, or a CRT controller chip we might be able to use. We would be very grateful. We hope it isn't much trouble and we very much hope you could help us.

Yours sincerely,

/s/ Kris Coolsaet, Jacques Buyse, Henri Deleplanque (member of SUG)

M. Buyse's address is: M. De Tayelann 33, B854Ø Moorsele, Belgium.
M. Deleplanque's address is: Stokerijstraaat 24, B855Ø Zwevegem, Belgium.

MISCELLANEA

DR. JOHN E. ALDRICH, Director, Medical Physics Department, Radiation Oncology, Victoria General Hospital, Halifax, Nova Scotia, B3H 2Y9, would very much like to get in touch with other SYM users who have developed applications programs in areas related to medical physics.

Several readers have been kind enough to send in Indexes to SYMPHYSIS. These include CHUCK HARRISON of Groton, CT, who submitted a RAE cassette version which permits using RAE's FInd to locate the proper issue number and page number. It is arranged serially by issue and page, and he has used lots of "KEYWORDS" for each article. It is best used for machine retrieval, and after we bring it up to date we'll release it on cassette. We publish as an addendum to this issue an alphabetic index contributed by BORIS GOLDOWSKY; we thank him for the many hours he put in on this difficult task.

Our regular printer will only handle the newsletter in multiples of eight pages, so we sent him the first 48 pages to do, and are sending these last four pages to a "jiffy" printer. We point this out, just in case you are wondering why the extra "loose" sheet. Besides, it gave us an extra week to finish up this issue.

The hardest part is the last part, where we worry about not being able to include everything we wanted to. There is as much good material still in our backlog pile as was put in. Our summer vacation starts next weekend. We plan to spend a month getting caught up on unfinished projects, then the next month getting started on Issue 16. We'll spend some time in relaxing, too, with a few trips within California.

If all goes as scheduled, you should receive Issue 16 early in September. A happy summer (winter to our down-under friends) from Jean, Joyce, Denny, and . . . . .

Lux