EDITED AND REPRINTED MAY 1981

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

THE SYM-1 USERS' GROUP NEWSLETTER

ISSUE NUMBER 1 - JANUARY/FEBRUARY 1980

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

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

Editor/Publisher: H. R. "Lux" Luxenberg
Business/Circulation: Jean Luxenberg

SUBSCRIPTION RATES:

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

Copies of the Introductory Issue, while they last, are available for $1.50 postpaid anywhere in the world.

EDITOR'S NOTES:

We were quite unprepared for the deluge of subscriptions, articles for publication, and purchase orders which began arriving within a few days of the mailing of the Introductory Issue. We were especially surprised at the large number of overseas subscribers, since we had not realized so many SYM-1s had been sold abroad.

We wish to thank Synertek Systems Corporation for their help and cooperation in getting the SYM-1 Users' Group and SYM-PHYSIS going, by the printing and mailing of the Introductory Issue, and we hope that the SYM-biotic relationship between SSC and SUG continues.

We did expect many questions about features of the SYM-1 hardware and firmware and many questions beginning with "How do I . . . .", so we were not too surprised by these. Where the answers were easy, we wrote immediately. Where the questions were not so easy to answer, we are still "researching," and we will answer as soon as possible. Many of your questions about the features of SUPERMON are answered in the programs and articles in this issue.

This issue and the Introductory Issue both emphasize Utility Programs. We feel that having programs such as RELOCATE, DISASSEMBLE, FIND, RENUMBER, MERGE, etc., on call, make programming a less frustrating task.

Now that the most important utility programs are available, future issues will concentrate on applications.

One of our colleagues, Tom Gettys, Lecturer in Mathematics at California State University, Chico, and I are just beginning to interface dual mini-disks to our systems. RAE has built in linkages for disk (these are not described in the manual), BASIC links for SAVE and LOAD are in page zero, and can be trapped to transfer to disks, and MON can have .S3 and .L3 added for disk transfers. We are also "PEEKing" around in BASIC so that we can use its subroutines for our own purposes. We will keep you posted on our progress in these areas.

We have been asked by many subscribers for our recommendations on how to implement the expansion capabilities of the SYM-1, i.e., what sort of motherboard should they buy? Believe it or not, we have no recommendations to make. Our system is laid out flat on a 2ft x 2ft board, which sits on a work table. The various add-on boards lie flat on the supporting board. We have a 16K memory board designed for the Motorola EXORCISOR Buss cabled to the expansion port, and will be adding a disk controller designed for the KIM-4 Buss in parallel to the expansion port. Various D/A boards and prototyping sockets are cabled to the applications connectors. The system does get dusty but it is working. If we had any recommendation at all to make, it would be for you to decide for yourself which you would like the best. Note that while Commodore is "supporting" the KIM-4 Buss, the other two suppliers of the 65XX family, Synertek and Rockwell International, are supporting the EXORCISOR Buss; this will mean increased commonality between 65XX and 68XX systems.

We are tabulating the data on the subscription blanks you sent in to determine what the typical SYM system looks like, and there is no such thing. There is no such person as a typical SYM user either. We do feel, however, that perhaps half of the material in each issue will be of value to each user or system, no matter what his (there are no hers!) background or system capabilities. We certainly hope so.

There were more articles submitted then we had room for, and this will continue to be the case, since the majority of subscribers voted for commented source code, rather than the space-saving object code dumps. We had some excellent tutorial articles on the keyboard/display interface, power-on reset, cassette problems, etc. There were some great programs which it hurt us to leave out. If the authors give permission, we will publish the titles and descriptions of these items, and offer them for sale by the half-ounce (2 pages plus envelope per half-ounce). Not being facetious here, overseas airmail is $0.31 per half-ounce!

COMING ATTRACTIONS:

In Issue No 2 we will (really) get around to graphics and music, with both hardware and software considerations (e.g., D/A converters), and lots of references. We will include oscilloscope and terminal graphics and include a program by Carl Moser (part BASIC/part 6502) which shows how to map the CRT terminal memory into a portion of the SYM memory to provide for more interactive capability. Also a number of very short, but sweet, utilities will be published.

SPECIAL NOTICE RE RAE-1/2

The two chip version of RAE-1/2 is now available through the SYM Users' Group. If you have been worried about not having enough ROM sockets on the SYM to hold both BAS-1 and the two chip version of RAE-1/2, see the article by George Wells, on page 18 of this issue, for how to make the sockets do double duty. The addition of RAE to BAS and SUPERMON makes SYM-1 the most powerful and versatile single board computer available today. We would not have even considered publishing SYM-PHYSIS without the use of RAE as a word processor. Ordering information on back page.

ASSEMBLY LANGUAGE PROGRAMMING

The following paragraphs, written by Tom Gettys, extracted from the RAE-1 Reference Manual, express beautifully the advantages of switching over from "hand-assembly" to the use of a symbolic assembler as early as is financially practical, that is, as soon as you can get a terminal:

> An assembler is a program which allows the user to compose and enter programs at the machine language level in a form that is much more convenient than actual machine code. The assembler accepts mnemonic names for individual instructions, allows symbolic names to be assigned to memory locations and data, provides for address arithmetic in terms of symbolic names, and certain other features, depending on the sophistication of the assembler in question.
>
> It is commonly thought that the primary feature offered by an assembler is that of writing machine instructions in a more convenient form. However, this is only one aspect of the advantage of an assembler, and perhaps not even the most significant. The use of symbolic names to represent numbers makes variables of what most likely would have been considered constants. The very presence of symbols bestows a generality and flexibility to a program which otherwise might have seemed quite rigid. This encourages the programmer to abstract the immediate problem and perhaps develop a more adaptable program. Also, since the actual calculation or assignment of a value to a symbol can be deferred, the development of logically separate modules can proceed freely. Programs so organized become much more readable and manageable, both in their maintenance and amenability to revision.

The least expensive, smallest, 6502 symbolic assembler/editor I know of is Robert Denison's 2K Symbolic Assembler (2KSA); one of the best ones I know of is Synertek's RAE-1, in ROM. If you have only the on-board 4K of RAM, consider the 2KSA. If you have also an extra 8K at 2000-3FFF, and don't mind cassette loading, consider Carl Moser's Macro-Assembler/Text Editor (ASSM/TED) for the SYM-1. The 2KSA is ideal for minimal systems, ASSM/TED and RAE-1 can also be used in word processing systems.

As you probably have discovered by this time, there is no "standard" 6502 assembly language syntax. Bob Tripp, in an editorial in MICRO, No. 2 (Dec 77-Jan 78), said, (and I quote!) " . . MOS Technology syntax is so horrible." Hal Chamberlin, in MICRO, No. 4 (April-May 1978) took a very strongly opposing viewpoint. In any event, the controversy centers on whether the addressing mode should be indicated in the opcode field or in the operand field. I have used both, and have no strong preference. For the benefit of newcomers to assembly language programming a conversion table is provided below, with notes describing minor variations in syntax within the two major schools of thought.

Also shown below is an example of the use of 2KSA for a simple program. The full power of a symbolic assembler is apparent, however, only when it becomes necessary to modify the program to incorporate new features, such as, in the example given, to relocate and to include JSR BEEP.

EXAMPLE OF USE OF 2KSA

SSC Technical Note #49, December 1978, (pages 12 and 13 of "Technical Notes") presents a Rotating Display program in MOS Technology assembly form. Presented here is the same program in 2KSA with the following modifications:

1) Since 2KSA occupies 0200-09FF the program is assembled at the 2KSA default location 0C80, rather than at 0200. The program is named "DSPLY".
2) Since 2KSA does not permit absolute addressing within a module, the final JMP is replaced with CLV/BVC (other methods for forcing an unconditional branch may be substituted).
3) FILE has been moved to 0D00 and COUNT has been moved to 0060 to avoid conflict with 2KSA and its page zero variables (0000-005B).
4) FILE (0D00-0D06) contains the segment codes for "HELLO ".

Following -ASSEM to assemble, -PRINT 00T031 to print the object/source code summary, and -SYM.. to print the origin location and symbol table, SUPERMON was called with $. (For the purpose of this example, the ESC code 1B at 0776 was replaced with the $ code 24 to provide a printable character.) The program at 0100 (ending with BRK) is a patch to transfer selected pointers from page zero to unused memory in page 0B. The source code, symbol table, and pointers were then dumped to cassete.

The second printout shows the method used to reenter, relocate, modify, and store the object code for a previously saved source code. The patch at 0100 (same as the one mentioned above) and the method for relocation of the origin can be written in a more elegant form. The form given here is the KIM version appearing in the first update to the 2KSA. After the object code is stored, exit is made with $ (or ESC) to the monitor, the data for FILE is entered at 0D00, and the program is run (successfully) at 0E00. There was no requirement to use either -PRINT or -SYM.., but these were entered to show the new origin, an added entry to the symbol tables, and the corrected object code. Source code can be saved on tape only prior to -STORE, since -STORE erases all local labels.

The addresses, -0C00, etc., provided as prompts by 2KSA are for the source code storage area. In the object/source code printout, the first six columns are the object code, and the last two columns are the object code addresses relative to the origin. The remaining columns are the source code. The 01 and 06 following DISBUF are additive offsets to DISBUF; i.e., DISBUF+1 and DISBUF+6 are being specified.

In the printout below entries made by the user are enclosed within "frames"; items outside the frames are 2KSA-generated prompts and outputs.

One of the interesting features of 2KSA is that, due to the method of encoding the source code for storage and processing, the source code and the object code are exactly the same length, and correspond byte-by-byte.

This example does not show error messages (since no errors were made!) or the full editing capability provided by -INSRT to insert, delete, correct, replace, or append lines, depending on the line number syntax entered.

INITIAL ENTRY OF PROGRAM FROM KEYBOARD

```
.G 05B8

?       ?ASSGN  FILE    0D00
?ASSGN          DISBUF  A640
?ASSGN          SCANDS  8906
?ASSGN          COUNT   0060
?ASSGN          ACCESS  8B86
?ASSGN
?       ?BEGIN  DSPLY
- 0C00                  JSR     ACCESS
- 0C03                  LDY#    06
- 0C05          ONE     LDAY    FILE
- 0C08                  STAY    DISBUF
- 0C0B                  DEY
- 0C0C                  BPL     ONE
- 0C0E          CYCLE   LDA#    FF
- 0C10                  STAZ    COUNT
- 0C12          TWO     JSR     SCANDS
- 0C15                  DECZ    COUNT
- 0C17                  BNE     TWO
- 0C19                  LDA     DISBUF
- 0C1C                  PHA
- 0C1D                  LDY#    00
- 0C1F          THREE   LDAY    DISBUF  01
- 0C22                  STAY    DISBUF
- 0C25                  INY
- 0C26                  CPY#    06
- 0C28                  BNE     THREE
- 0C2A                  PLA
- 0C2B                  STA     DISBUF  06
- 0C2E                  CLV
- 0C2F                  BVC     CYCLE
- 0C31-ASSEM
- 0C31-PRINT   00TO31
20868B DSPLY    JSR     ACCESS      00
A006           LDY#    06          03
B9000D ONE      LDAY    FILE        05
9940A6          STAY    DISBUF      08
88              DEY                 0B
10F7            BPL     ONE         0C
A9FF    CYCLE   LDA#    FF          0E
8560            STAZ    COUNT       10
200689 TWO      JSR     SCANDS      12
C660            DECZ    COUNT       15
D0F9            BNE     TWO         17
AD40A6          LDA     DISBUF      19
48              PHA                 1C
A000            LDY#    00          1D
B941A6 THREE    LDAY    DISBUF  01  1F
9940A6          STAY    DISBUF      22
C8              INY                 25
C006            CPY#    06          26
D0F5            BNE     THREE       28
68              PLA                 2A
8D46A6          STA     DISBUF  06  2B
B8              CLV                 2E
50DD            BVC     CYCLE       2F
- 0C31-SYM..
0C80..
THREE   0C1F
TWO     0C12
CYCLE   0C0E
ONE     0C05
DSPLY   0C00
ACCESS  8B86
COUNT   0060
SCANDS  8906
DISBUF  A640
FILE    0D00
-INSRT  0965
-PRINT  090D
-STORE  07A6
-SYM..  071F
-ASSEM  06EB
-LOCAL  0665
?REDEF  0672
?BEGIN  0610
?ASSGN  062E
- 0C31$
077A,0
.G 0100

0110,0
.S2 01,0A00,0C7F
.S2 01,0A00,0C7F
.S2 01,0A00,0C7F
.
```

RE-ENTRY OF SOURCE CODE FROM TAPE

```
.G 05B8

?       $
077A,0
.L2 01
.G 0100

0110,0
.M 0040
0040,80,00
0041,0C,0E
0042,A5,
.G 05D6

? 0C31-LOCAL   BEEP    8972
-LOCAL
- 0C31-INSRT   12
- 0C12                  JSR     BEEP
- 0C15-ASSEM
- 0C34-PRINT   00TO34
20868B DSPLY    JSR     ACCESS      00
A006           LDY#    06          03
B9000D ONE      LDAY    FILE        05
9940A6          STAY    DISBUF      08
88              DEY                 0B
10F7            BPL     ONE         0C
A9FF    CYCLE   LDA#    FF          0E
8560            STAZ    COUNT       10
207289          JSR     BEEP        12
```



```
200689 TWO      JSR     SCANDS      15
C660            DECZ    COUNT       18
D0F9            BNE     TWO         1A
AD40A6          LDA     DISBUF      1C
48              PHA                 1F
A000            LDY#    00          20
B941A6 THREE    LDAY    DISBUF  01  22
9940A6          STAY    DISBUF      25
C8              INY                 28
C006            CPY#    06          29
D0F5            BNE     THREE       2B
68              PLA                 2D
8D46A6          STA     DISBUF  06  2E
B8              CLV                 31
50DA            BVC     CYCLE       32
- 0C34-SYM..
0E00..
BEEP    8972
THREE   0C22
TWO     0C15
CYCLE   0C0E
ONE     0C05
DSPLY   0C00
ACCESS  8B86
COUNT   0060
SCANDS  8906
DISBUF  A640
FILE    0D00
-INSRT  0965
-PRINT  090D
-STORE  07A6
-SYM..  071F
-ASSEM  06EB
-LOCAL  0665
?REDEF  0672
?BEGIN  0610
?ASSGN  062E
- 0C34-STORE
?       $
077A,0
.D 0D00
0D00 76 79 38 38 3F 00 00
.G 0E00
```

ADDRESSING MODE SYNTAX COMPARISIONS

	MODE	2KSA	RAE-1
1.	Absolute		__
2.	Implied		
3.	Relative		__
4.	Immediate	#	#__
5.	Zero Page	Z	*__
6.	Accumulator	A	A
7.	Absolute Indexed X	X	__,X
8.	Absolute Indexed Y	Y	__,Y
9.	Zero Page Indexed X	ZX	*__,X
10.	Zero Page Indexed Y	ZY	*__,Y
11.	Indirect	I	(__)
12.	Indexed X Indirect	IX	(__,X)
13.	Indirect Indexed Y	IY	(__),Y

Note 1: In 2KSA the addressing mode information is in the opcode field. In RAE-1 it is in the operand field.

Note 2: In MOS Technology/System 65 syntax the"*" to denote zero page addressing is not used. Otherwise identical with RAE-1.

NOTE 3: In MICRO magazine syntax IM is used for immediate mode instead of '#'. Otherwise identical with 2KSA.

Miscellaneous Notes on 2KSA: 2KSA is self-formatting, i.e., entry of a single space tabs to the next field. The first line printed after the -SYM.. command is the module's origin address followed by "..". The cold start is at 05B8, the warm start is at 05D6. A more elegant way to relocate the origin is to replace the "?" (3F) at locations 0681 and 09D0 with "-" (2D), then use -REDEF before -STORE. The patch at 0100 can then be modified to jump to the warm start (4C D6 05).

RELOCATE FOR THE SYM-1

If you are limited to hand assembly, or even if you are not, the program RELOCATE, adapted from Jim Butterfield's program in the First Book of KIM, can be very useful. Since the published program is well commented, comments have been omitted from this version. Be careful with RELOCATE; a tape backup of the original program is advisable. Here are the rules for RELOCATE:

1. Programs to be relocated should have an FF inserted after the last executable instruction and before any tables.
2. RELOCATE will modify only instructions between the start address and the FF, but assumes any addresses referring to tables and programs beyond the FF and up to the page limit are to be modified.
3. RELOCATE will take care of all relative addresses but will give no warning if you have "spread" the program too far.
4. RELOCATE will take care of all absolute addresses if they are not below the start of the program or above the page limit (set by default to 80).
5. RELOCATE will NOT handle vectors or addresses in tables. These must be handled "manually."
6. After using RELOCATE use .B with three parameters to do the actual movings.
7. Sometimes, but not always, you can use .B before .R, for example, when down-moving but not when up-moving an entire program. This is because when you block move an entire program up before using RELOCATE all absolute addresses within the program are now below the start of the program, and will not be modified.

The parameters have the following meanings:

P1 is the signed "adjustment," e.g.,

0004	up	04 bytes
FFFC	down	04 bytes
0200	up	02 pages
FE00	down	02 pages

P2 is the program starting address.
P3 is the start address of the block to be moved.
It will equal P2 if the entire program is to be moved.
It will be greater than P2 if a "gap" is to be opened or closed.

Two examples of the use of RELOCATE are given following the listing.

MORE SUPERMON EXTENSIONS

Elsewhere in this issue are examples of how to add new commands of "recognized" syntax to the monitor. Several of you have asked about the new unrecognized syntax vector in MON 1.1. Here are some hints on its use. Suppose you wish to include "named" tape saves and loads to the monitor, e.g., S2 BLACKJACK,0200,0735, or L2 BLACKJACK. This would be quite elegant, no? This is easily done by pointing URSVEC to subroutines to do the job. Perhaps we'll publish the solution next issue. Note that URSVEC is called whenever either a delimiter other than a comma is used, or more than three parameters are entered, or a non-hex character is entered. It might be simpler to use a special symbol before the file name, like % or #, since the B in BLACKJACK is a hex character, and would have been "lost" before the unrecognized syntax was "recognized." While we used .D and .R with non-recognized numbers of parameters for our DISASSEMBLE and RELOCATE programs, if there had been a conflict in numbers of parameters, we could have used other letters, or the unimplemented user functions, U0 through U7, which are more easily called from the hex keypad.

MERGE AND DELETE FOR SYM BASIC

The purpose of this routine is to provide a machine language means for merging two BASIC programs on the SYM-1. As a side benefit, selective deletion (actually selective retention) is also possible.

The merge is accomplished in 2 steps. First, the current program is saved in ASCII format at the top of the memory allocated to BASIC. This is accomplished by changing the output vector so that the ASCII output stream is trapped and sent to high memory instead of to the terminal by the LIST command, using any of the available line number options. A new BASIC program may be entered by using the NEW command and entering the program from the terminal, or a previously SAVEd program may be LOADed. When you are ready to MERGE in the previously saved program, the input vector is altered so that the ASCII input stream is obtained from high memory instead of the keyboard.

The instructions for use are simple: Call MERGE via the USR command. The prompt symbol ":" will be printed. There are only three valid commands; LIST, MERGE and E.

LIST — causes the current program to be saved in high memory. If a range is specified (e.g., LIST 100-10000) only those lines within that range will be transferred.

MERGE — (or any word beginning with M) causes the code saved in high memory to be retrieved. If MERGE is terminated with a slash (/), instead of a carriage return, terminal echo will be suppressed during the transfer.

E — exits to BASIC.

To delete a block of lines, use the LIST command and specify the lines to be kept. Now type NEW to clear away the current program, and then enter MERGE again and use the MERGE command.

This program may be interfaced with the BASIC Terminal Control Patch given in the last issue of SYM-PHYSIS by simply changing the address INTCHR in the macro at line 1310 to the address of GET.CHR in the Terminal Control Patch. If you are using the Terminal Control Patch you may also wish to modify it to allow you to call MERGE with some special character, as for example, CTRL M, instead of with USR.

THE USR FUNCTION IN BASIC

The first time you call the MERGE function, described elsewhere in this issue, you must call it as X = USR(4096,0) or X = USR(&"1000",0). All future calls may be of the simpler form X = USR(0), since any time you call USR with just one parameter, the location of the previous call is assumed. This makes it much easier to use. Note that you must pass at least one parameter, even when not needed, so that the A and Y registers are not preserved. If you wish USR to return a value to the calling variable your subroutine must load A with the high byte and Y with the low byte, and return, not with RTS, but with JMP to $D14C. USR in SYM BASIC is very versatile in allowing the passing of multiple parameters; not too many similar BASICs allow this feature.

NEW PUBLICATION: Add to your list of periodicals "COMPUTE. The Journal for Progressive Computing." This is the successor to the PET Gazette, and now covers all 6502 systems. Six issue subscription $9.00 US and Canada, $22.50 Europe Air Mail, elsewhere inquire for rates. Contact Bob Lock, Small Systems Services, Inc., P.O. Box 5119, Greensboro, NC 27403.

ALL PRICES OBSOLETE

```
>ASSEMBLE LIST

                0010 ;       MONITOR PATCH FOR THE SYM-1
                0020 ;       FOR "RELOCATE" AND "DISASSEMBLE"
                0030
                0040 ;       "RELOCATE" FOR THE SYM-1
                0050
                0060 ;       P1 IS THE "ADJUSTMENT"
                0070 ;       P2 IS THE START OF PROGRAM
                0080 ;       P3 IS THE LOWER BOUND FOR ADJUSTMENT
                0090 ;       $8000 IS THE DEFAULT UPPER BOUND
                0100
                0110            .BA $2200
                0120            .OS
                0130
                0140 URCVEC     .DE $A66C
                0150 URSVEC     .DE $A669
                0160 WARM       .DE $8003
                0170 ACCESS     .DE $8B86
                0180 P3L        .DE $A64A
                0190 P2L        .DE $A64C
                0200 P1L        .DE $A64E
                0210 P2SCR      .DE $829C
                0220 PAGLIM     .DE $80
                0230 POINT      .DE $FE
                0240 ALOC       .DE $F3
                0250 LIMIT      .DE $F4
                0260 DISASSEM2  .DE $2000
                0270
                0280 ;       PATCH NEW COMMANDS HERE
                0290 ;       This would be the place to include any
                0300 ;       changes you wish to make to the other
                0310 ;       default parameters, for example the USR
                0320 ;       JUMPS, interrupt vectors, etc.
                0330
2200- 20 86 8B  0340 PATCH      JSR ACCESS
2203- A9 16     0350            LDA #L,RELOCATE
2205- 8D 6D A6  0360            STA URCVEC+1
2208- 8D 6A A6  0370            STA URSVEC+1
220B- A9 22     0380            LDA #H,RELOCATE
220D- 8D 6E A6  0390            STA URCVEC+2
2210- 8D 6B A6  0400            STA URSVEC+2
2213- 4C 03 80  0410            JMP WARM
                0420
                0430 ;       DEFINE RELOCATE AS .R XXXX,YYYY,ZZZZ
                0440 ;       DEFINE DISASSEMBLE AS .D XXXX,YY
                0450
                0460
2216- C9 52     0470 RELOCATE   CMP #'R
2218- D0 04     0480            BNE DISASSEM
221A- E0 03     0490            CPX #03
221C- F0 0A     0500            BEQ RELOCATE1
221E- C9 44     0510 DISASSEM   CMP #'D
2220- D0 04     0520            BNE ERROR
2222- E0 02     0530            CPX #02
2224- F0 05     0540            BEQ DISASSEM1
                0550 ;       PLACE ANY OTHER NEW COMMANDS HERE
2226- 38        0560 ERROR      SEC
2227- 60        0570            RTS
                0580 ;       PLACE JUMPS TO COMMAND SUBS HERE
2228- 4C 61 22  0590 RELOCATE1  JMP RELOCATE2
222B- AD 4A A6  0600 DISASSEM1  LDA P3L
```



```
222E- 85 F2     0610            STA *$F2
2230- AD 4C A6  0620            LDA P2L
2233- 85 F0     0630            STA *$F0
2235- AD 4D A6  0640            LDA P2L+1
2238- 85 F1     0650            STA *$F1
223A- 4C 00 20  0660            JMP DISASSEM2
                0670 ;(Leave room for future expansion here)
                0680            .BA $2261
                0690 ;"RELOCATE" ITSELF BEGINS HERE
                0700 ;BUT MAY BE MOVED ELSEWHERE
                0710
2261- 20 9C 82  0720 RELOCATE2  JSR P2SCR MOVE P2 TO FE,FF
                0730
                0740 ;(From here on the program is almost
                0750 ;identical with the published version
                0760 ;except for addresses and exit point,
                0770 ;and the correction of a wrong label)
                0780
2264- D8        0790 START      CLD
2265- A0 00     0800            LDY #00
2267- B1 FE     0810            LDA (POINT),Y
2269- A8        0820            TAY
226A- A2 07     0830            LDX #07
226C- 98        0840 LOOP       TYA
226D- 3D E9 22  0850            AND TAB1-1,X
2270- 5D F0 22  0860            EOR TAB2-1,X
2273- F0 03     0870            BEQ FOUND
2275- CA        0880            DEX
2276- D0 F4     0890            BNE LOOP
2278- BC F8 22  0900 FOUND      LDY TAB3,X
227B- 30 0D     0910            BMI TRIP
227D- F0 24     0920            BEQ BRAN
227F- E6 FE     0930 SKIP       INC *POINT
2281- D0 02     0940            BNE INEX
2283- E6 FF     0950            INC *POINT+1
2285- 88        0960 INEX       DEY
2286- D0 F7     0970            BNE SKIP
2288- F0 DA     0980            BEQ START
228A- C8        0990 TRIP       INY
228B- 10 02     1000            BPL CONTINUE
228D- 18        1010            CLC
228E- 60        1020            RTS
228F- C8        1030 CONTINUE   INY
2290- B1 FE     1040            LDA (POINT),Y
2292- AA        1050            TAX
2293- C8        1060            INY
2294- B1 FE     1070            LDA (POINT),Y
2296- 20 CF 22  1080            JSR ADJUST
2299- 91 FE     1090            STA (POINT),Y
229B- 88        1100            DEY
229C- 8A        1110            TXA
229D- 91 FE     1120            STA (POINT),Y
229F- A0 03     1130            LDY #03
22A1- 10 DC     1140            BPL SKIP
22A3- C8        1150 BRAN       INY
22A4- A6 FE     1160            LDX *POINT
22A6- A5 FF     1170            LDA *POINT+1
22A8- 20 CF 22  1180            JSR ADJUST
22AB- 86 F3     1190            STX *ALOC
22AD- A2 FF     1200            LDX #$FF
22AF- B1 FE     1210            LDA (POINT),Y
22B1- 18        1220            CLC
```

```
22B2- 69 02     1230            ADC #02
22B4- 30 01     1240            BMI OVER
22B6- E8        1250            INX
22B7- 86 F4     1260 OVER       STX *LIMIT
22B9- 18        1270            CLC
22BA- 65 FE     1280            ADC *POINT
22BC- AA        1290            TAX
22BD- A5 F4     1300            LDA *LIMIT
22BF- 65 FF     1310            ADC *POINT+1
22C1- 20 CF 22  1320            JSR ADJUST
22C4- CA        1330            DEX
22C5- CA        1340            DEX
22C6- 8A        1350            TXA
22C7- 38        1360            SEC
22C8- E5 F3     1370            SBC *ALOC
22CA- 91 FE     1380            STA (POINT),Y
22CC- C8        1390            INY
22CD- 10 B0     1400            BPL SKIP
22CF- C9 80     1410 ADJUST     CMP #PAGLIM
22D1- B0 15     1420            BCS OUT
22D3- CD 4B A6  1430            CMP P3L+1
22D6- D0 03     1440            BNE TES2
22D8- EC 4A A6  1450            CPX P3L
22DB- 90 0B     1460 TES2       BCC OUT
22DD- 48        1470            PHA
22DE- 8A        1480            TXA
22DF- 18        1490            CLC
22E0- 6D 4E A6  1500            ADC P1L
22E3- AA        1510            TAX
22E4- 68        1520            PLA
22E5- 6D 4F A6  1530            ADC P1L+1
22E8- 60        1540 OUT        RTS
22E9- FF        1550            .BY $FF       ;END OF INSTRUCTIONS
22EA- 0C 1F 0D  1560 TAB1       .BY $0C $1F $0D $87 $1F $FF $03
22ED- 87 1F FF
22F0- 03
22F1- 0C 19 08  1570 TAB2       .BY $0C $19 $08 $00 $10 $20 $03
22F4- 00 10 20
22F7- 03
22F8- 02 FF FF  1580 TAB3       .BY $02 $FF $FF $01 $01 $00 $FF $FE
22FB- 01 01 00
22FE- FF FE
                1590            .EN
```

LABEL FILE: [/ = EXTERNAL]

```
/URCVEC=A66C            /URSVEC=A669            /WARM=8003
/ACCESS=8B86            /P3L=A64A               /P2L=A64C
/P1L=A64E               /P2SCR=829C             /PAGLIM=0080
/POINT=00FE             /ALOC=00F3              /LIMIT=00F4
/DISASSEM2=2000         PATCH=2200              RELOCATE=2216
DISASSEM=221E           ERROR=2226              RELOCATE1=2228
DISASSEM1=222B          RELOCATE2=2261          START=2264
LOOP=226C               FOUND=2278              SKIP=227F
INEX=2285               TRIP=228A               CONTINUE=228F
BRAN=22A3               OVER=22B7               ADJUST=22CF
TES2=22DB               OUT=22E8                TAB1=22EA
TAB2=22F1               TAB3=22F8
//0000,2300,2300
```



```
.F AA,223D,2260
.
.
.V 2200,22FF
2200 20 86 8B A9 16 8D 6D A6,90
2208 8D 6A A6 A9 22 8D 6E A6,99
2210 8D 6B A6 4C 03 80 C9 52,21
2218 D0 04 E0 03 F0 0A C9 44,DF
2220 D0 04 E0 02 F0 05 38 60,22
2228 4C 61 22 AD 4A A6 85 F2,05
2230 AD 4C A6 85 F0 AD 4D A6,B9
2238 85 F1 4C 00 20 AA AA AA,99
2240 AA AA AA AA AA AA AA AA,E9
2248 AA AA AA AA AA AA AA AA,39
2250 AA AA AA AA AA AA AA AA,89
2258 AA AA AA AA AA AA AA AA,D9
2260 AA 20 9C 82 D8 A0 00 B1,EA
2268 FE A8 A2 07 98 3D E9 22,19
2270 5D F0 22 F0 03 CA D0 F4,09
2278 BC F8 22 30 0D F0 24 E6,16
2280 FE D0 02 E6 FF 88 D0 F7,1A
2288 F0 DA C8 10 02 18 60 C8,FE
2290 B1 FE AA C8 B1 FE 20 CF,BD
2298 22 91 FE 88 8A 91 FE A0,AF
22A0 03 10 DC C8 A6 FE A5 FF,AE
22A8 20 CF 22 86 F3 A2 FF B1,8A
22B0 FE 18 69 02 30 01 E8 86,AA
22B8 F4 18 65 FE AA A5 F4 65,C1
22C0 FF 20 CF 22 CA CA 8A 38,27
22C8 E5 F3 91 FE C8 10 B0 C9,DF
22D0 80 B0 15 CD 4B A6 D0 03,B5
22D8 EC 4A A6 90 0B 48 8A 18,16
22E0 6D 4E A6 AA 68 6D 4F A6,EB
22E8 60 FF 0C 1F 0D 87 1F FF,27
22F0 03 0C 19 08 00 10 20 03,8A
22F8 02 FF FF 01 01 00 FF FE,89
 8889
.
```

EXAMPLE 1: Eliminate the NOPs (EA) in the "trivial" program below.

```
.D 2800
2800 4C 05 28 EA EA D0 F9 FF
2808 AA AA AA AA AA AA AA AA
2810
.
.G 2200

.R FFFE,2800,2805
.V 2800
2800 4C 03 28 EA EA D0 FB FF,15
 0515
.B 2803,2805,280F
.V 2800
2800 4C 03 28 D0 FB FF AA AA,95
 0495
.
```

EXAMPLE 2: Interchange the locations of RELOCATE and DISASSEMBLE.

RELOCATE is at 2200-22FF with possible garbage at 223D-2260. DISASSEMBLE is at 2000-21FF. Do the following:

1. .G 2200 to patch programs
2. Insert FF at 211A following last instruction
3. .F AA,223D,2260 to kill garbage
4. .R 0100,2000,2000 to relocate
5. .B 2300,2000,21FF to save temporarily.
6. .B 2000,2200,22FF to copy RELOCATE
7. .R FE00,2000,2000 to relocate
8. .B 2100,2300,24FF to position correctly
9. Correct addresses as follows:
   200C from 22 to 20
   203C from 1E to 21
10. .G 2000 to repatch correctly
11. Save 2000-22FF on cassette

```
                        0010            .BA $1000
                        0020            .OS
                        0030            .MC $2000
                        0040            .ES
                        0050
                        0060 ;=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
                        0070 ;[                                                       ]
                        0080 ;[       MERGE/DELETE PROGRAM FOR SYM BASIC              ]
                        0090 ;[                                                       ]
                        0100 ;[         COPYRIGHT 1979 BY THOMAS GETTYS               ]
                        0110 ;[              ALL RIGHTS RESERVED                      ]
                        0120 ;[                                                       ]
                        0130 ;-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
                        0140 ;
                        0150 ;
                        0160 ;       DOUBLE STORE MACRO DEFINITION
                        0170 ;
                        0180 !!!DS      .MD (DATA ADDRS)
                        0190            LDA #L,DATA
                        0200            STA ADDRS
                        0210            LDA #H,DATA
                        0220            STA ADDRS+1
                        0230            .ME
                        0240 ;
                        0250 ;       ADDRESS DECLARATIONS
                        0260 ;
                        0270 COUNT      .DE $6C       COMMAND CHARACTER COUNT
                        0280 MEM.END    .DE $83       BASIC END OF FREE MEMORY
                        0290 TXT.PTR    .DE $93       ASCII TEXT POINTER
                        0300 RAM.PTR    .DE $FA       POINTER FOR RIN SUPERMON ROUTINE
                        0310 BUF        .DE $135      COMMAND BUFFER
                        0320 SUPRESS    .DE $809A     USED TO SUPRESS OUTPUT
                        0330 CRLF       .DE $834D     PRINT CR AND LF
                        0340 RIN        .DE $887E     GET COMMANDS FROM RAM
                        0350 RESXAF     .DE $8A3E     RESTORE ALL BUT A AND F
                        0360 INTCHR     .DE $8A58     INPUT CHARACTER
                        0370 TOUT       .DE $8AA0     TERMINAL OUTPUT
                        0380 ACCESS     .DE $8B86     UNWRITE PROTEC SYS RAM
                        0390 BASIC.WARM .DE $C27E     WARM START TO BASIC
                        0400
                        0410 SCRA       .DE $A63A     SYS RAM USED BY RIN
                        0420 INVEC      .DE $A660     INPUT TRANSFER VECTOR
                        0430 OUTVEC     .DE $A663     OUTPUT TRANSFER VECTOR
                        0440
                        0450 ;
                        0460 ;  "MERGE" ENTRY POINT  -  BEGIN MAINLINE
                        0470 ;
                        0480 ;       -=(PROCESS COMMAND)=-
                        0490 ;
1000- 20 4D 83          0500 PROMPT     JSR CRLF
1003- A9 3A             0510            LDA #':
1005- 20 A0 8A          0520            JSR TOUT PRINT PROMPT SYMBOL
1008- 20 58 8A          0530            JSR INTCHR
100B- 29 7F             0540            AND #$7F
100D- C9 4C             0550            CMP #'L       LIST?
100F- F0 3A             0560            BEQ LIST
1011- C9 4D             0570            CMP #'M       MERGE?
1013- F0 05             0580            BEQ MERGE
1015- C9 45             0590            CMP #'E       EXIT?
1017- D0 E7             0600            BNE PROMPT IF INVALID COMMAND, IGNORE IT
1019- 60                0610            RTS           RETURN TO BASIC
                        0620 ;
                        0630 ;       -=(MERGE ROUTINE)=-
```



```
                        0640 ;
101A- 20 58 8A          0650 MERGE      JSR INTCHR
101D- 29 7F             0660            AND #$7F
101F- C9 0D             0670            CMP #$D       CR?
1021- F0 0E             0680            BEQ MERGE.1
1023- C9 2F             0690            CMP #'/       SUPRESS OUTPUT?
1025- D0 F3             0700            BNE MERGE
                        0710            DS (SUPRESS OUTVEC+1)

1027- A9 9A
1029- 8D 64 A6
102C- A9 80
102E- 8D 65 A6

                        0720
1031- 38                0730 MERGE.1    SEC           POINT TO START OF TEXT
1032- AD 83 00          0740            LDA MEM.END
1035- ED 6C 00          0750            SBC COUNT
1038- 85 93             0760            STA *TXT.PTR
103A- A5 84             0770            LDA *MEM.END+1
103C- E9 00             0780            SBC #0
103E- 85 94             0790            STA *TXT.PTR+1
                        0800            DS (GET.CHR INVEC+1)

1040- A9 D9
1042- 8D 61 A6
1045- A9 10
1047- 8D 62 A6

104A- 60                0810            RTS
                        0820 ;
                        0830 ;       -=(LIST ROUTINE)=-
                        0840 ;
104B- A0 05             0850 LIST       LDY #5
104D- 99 30 01          0860            STA BUF-5,Y   SAVE THE 'L'
1050- C8                0870 LIST.1     INY           INCREMENT BUFFER POINTER
1051- 20 58 8A          0880            JSR INTCHR    GET NEXT CHARACTER
1054- 29 7F             0890            AND #$7F
1056- 99 30 01          0900            STA BUF-5,Y   AND SAVE IT
1059- C9 18             0910            CMP #$18      CTRL X?
105B- F0 A3             0920            BEQ PROMPT IF SO ABORT COMMAND
105D- C9 0D             0930            CMP #$D       CR?
105F- D0 EF             0940            BNE LIST.1
1061- A9 00             0950            LDA #0
1063- C8                0960            INY
1064- 99 30 01          0970            STA BUF-5,Y   MARK END OF COMMAND
1067- 8C 6C 00          0980            STY COUNT     SAVE CHARACTER COUNT
106A- A5 83             0990            LDA *MEM.END SET UP TEXT POINTER
106C- 85 93             1000            STA *TXT.PTR
106E- A5 84             1010            LDA *MEM.END+1
1070- 85 94             1020            STA *TXT.PTR+1
1072- 20 EE 10          1030            JSR DEC.PTR
                        1040 ;
                        1050 ;       FIX-UP VECTORS
                        1060 ;
1075- 20 86 8B          1070            JSR ACCESS
                        1080            DS (BUF RAM.PTR)

1078- A9 35
107A- 8D FA 00
107D- A9 01
107F- 8D FB 00
```

```
                1090            DS (DONE SCRA)

1082- A9 A1
1084- 8D 3A A6
1087- A9 10
1089- 8D 3B A6

                1100            DS (RIN INVEC+1)

108C- A9 7E
108E- 8D 61 A6
1091- A9 88
1093- 8D 62 A6

                1110            DS (OUTPUT OUTVEC+1)

1096- A9 CD
1098- 8D 64 A6
109B- A9 10
109D- 8D 65 A6

10A0- 60        1120            RTS
                1130 ;
                1140 ;      -=(AFTER LISTING, COME HERE)=-
                1150 ;
10A1- 18        1160 DONE       CLC             ADJUST FOR THE "OK"
10A2- A5 93     1170            LDA *TXT.PTR
10A4- 69 03     1180            ADC #3
10A6- 85 93     1190            STA *TXT.PTR
10A8- A5 94     1200            LDA *TXT.PTR+1
10AA- 69 00     1210            ADC #0
10AC- 85 94     1220            STA *TXT.PTR+1
                1230 ;
10AE- A9 00     1240            LDA #0          FLAG END OF TEXT
10B0- A8        1250            TAY
10B1- 91 93     1260            STA (TXT.PTR),Y
                1270 ;
                1280 ;      RESTORE TRANSFER VECTORS
                1290 ;
                1300 DONE.1     DS (TOUT OUTVEC+1)

10B3- A9 A0
10B5- 8D 64 A6
10B8- A9 8A
10BA- 8D 65 A6

                1310            DS (INTCHR INVEC+1)

10BD- A9 58
10BF- 8D 61 A6
10C2- A9 8A
10C4- 8D 62 A6

10C7- 20 4D 83  1320            JSR CRLF
10CA- 4C 7E C2  1330            JMP BASIC.WARM
                1340 ;
                1350 ;      -=(SAVE ASCII TEXT IN MEMORY)=-
                1360 ;
10CD- C9 0A     1370 OUTPUT     CMP #$A         IF CHARACTER A LF?
10CF- F0 07     1380            BEQ OUT.RTN     IF SO, IGNORE IT
10D1- A0 00     1390            LDY #0
10D3- 91 93     1400            STA (TXT.PTR),Y
```



```
10D5- 20 EE 10  1410            JSR DEC.PTR     ADJUST TEXT POINTER
10D8- 60        1420 OUT.RTN    RTS             RETURN FOR NEXT CHARACTER
                1430 ;
                1440 ;      -=(GET CHARACTER FROM MEMORY)=-
                1450 ;
10D9- A0 00     1460 GET.CHR    LDY #0
10DB- B1 93     1470            LDA (TXT.PTR),Y
10DD- F0 D4     1480            BEQ DONE.1      END OF TEXT?
10DF- 20 63 A6  1490            JSR OUTVEC      PRINT CHARACTER
10E2- AA        1500            TAX
10E3- 20 EE 10  1510            JSR DEC.PTR     ADJUST TEXT POINTER
10E6- 68        1520            PLA             ALLOW LOWER CASE CHARACTERS
10E7- 68        1530            PLA
10E8- 8A        1540            TXA
10E9- C9 0D     1550            CMP #$D
10EB- 4C 3E 8A  1560            JMP RESXAF
                1570 ;
                1580 ;      -=(SUPPORT ROUTINES)=-
                1590 ;
10EE- C6 93     1600 DEC.PTR    DEC *TXT.PTR DECREMENT TEXT POINTER
10F0- A5 93     1610            LDA *TXT.PTR
10F2- C9 FF     1620            CMP #$FF
10F4- D0 02     1630            BNE DEC.RTS
10F6- C6 94     1640            DEC *TXT.PTR+1
10F8- 60        1650 DEC.RTS    RTS
                1660
                1670
                1680            .EN
```

LABEL FILE: [/ = EXTERNAL]

/COUNT=006C	/MEM.END=0083	/TXT.PTR=0093
/RAM.PTR=00FA	/BUF=0135	/SUPRESS=809A
/CRLF=834D	/RIN=887E	/RESXAF=8A3E
/INTCHR=8A58	/TOUT=8AA0	/ACCESS=8B86
/BASIC.WARM=C27E	/SCRA=A63A	/INVEC=A660
/OUTVEC=A663	PROMPT=1000	MERGE=101A
DATA=8A58	ADDRS=A661	MERGE.1=1031
LIST=104B	LIST.1=1050	DONE=10A1
DONE.1=10B3	OUTPUT=10CD	OUT.RTN=10D8
GET.CHR=10D9	DEC.PTR=10EE	DEC.RTS=10F8

```
.B 1000,2000,20FF
.V 1000,10FF
1000 20 4D 83 A9 3A 20 A0 8A,1D
1008 20 58 8A 29 7F C9 4C F0,CC
1010 3A C9 4D F0 05 C9 45 D0,EF
1018 E7 60 20 58 8A 29 7F C9,A9
1020 0D F0 0E C9 2F D0 F3 A9,18
1028 9A 8D 64 A6 A9 80 8D 65,64
1030 A6 38 AD 83 00 ED 6C 00,CB
1038 85 93 A5 84 E9 00 85 94,0E
1040 A9 D9 8D 61 A6 A9 10 8D,6A
1048 62 A6 60 A0 05 99 30 01,41
1050 C8 20 58 8A 29 7F 99 30,7C
1058 01 C9 18 F0 A3 C9 0D D0,97
1060 EF A9 00 C8 99 30 01 8C,4D
1068 6C 00 A5 83 85 93 A5 84,22
1070 85 94 20 EE 10 20 86 8B,8A
1078 A9 35 8D FA 00 A9 01 8D,26
1080 FB 00 A9 A1 8D 3A A6 A9,81
1088 10 8D 3B A6 A9 7E 8D 61,14
1090 A6 A9 88 8D 62 A6 A9 CD,F6
1098 8D 64 A6 A9 10 8D 65 A6,DE
10A0 60 18 A5 93 69 03 85 93,12
10A8 A5 94 69 00 85 94 A9 00,76
10B0 A8 91 93 A9 A0 8D 64 A6,22
10B8 A9 8A 8D 65 A6 A9 58 8D,7B
10C0 61 A6 A9 8A 8D 62 A6 20,6A
10C8 4D 83 4C 7E C2 C9 0A F0,89
10D0 07 A0 00 91 93 20 EE 10,72
10D8 60 A0 00 B1 93 F0 D4 20,9A
10E0 63 A6 AA 20 EE 10 68 68,3B
10E8 8A C9 0D 4C 3E 8A C6 93,08
10F0 A5 93 C9 FF D0 02 C6 94,34
10F8 60 AA AA AA AA AA AA AA,3A
 7D3A
```

To relocate a whole number of pages, change the underlined page numbers to the new page number.

TERMINAL CONTROL PATCH ADDENDUM

In the TCP source listing published in the Introductory Issue we forgot to include the RAE-1 pseudo-op .ES (Expansion Set). The default is .EC (Expansion Clear) so the macro expansions were not listed. The TRIG.START address should be changed to $0F68 for this version. Here are the missing macro expansions:

```
0DE8- A9 F5    0DF5- A9 68    0E12- A9 21    0E87- A9 58    0E9C- A9 21
0DEA- 8D 61 A6 0DF7- 8D C4 00 0E14- 8D 61 A6 0E89- 8D 61 A6 0E9E- 8D 61 A6
0DED- A9 0D    0DFA- A9 0F    0E17- A9 0E    0E8C- A9 8A    0EA1- A9 0E
0DEF- 8D 62 A6 0DFC- 8D C5 00 0E19- 8D 62 A6 0E8E- 8D 62 A6 0EA3- 8D 62 A6
```

INPUT LOWER-CASE WITH (OR IN SPITE OF) SUPERMON

By Gary K. Humphrey
PSC #2, Box 12203
APO San Francisco, CA 96367

The SYM Technical Notes package contains a short routine which allows input of lower case characters. I have been using a different method which is even shorter (see program listing). Just place these instructions anywhere in memory (I have them at the entry to my video driver) and change INVEC to point to them. The return address stored on the stack will be changed so that your input routine will return to the monitor at INRT1 (8A2D), thereby avoiding the SUPERMON routine which converts lower-case characters to upper-case.

```
1E0A- 68        PLA         Pull return address (low) from
                            stack and discard,
1E0B- A9 2C     LDA #2C     Load new return address (low)...
1E0D- 48        PHA         ...and push it onto the stack.
1E0E- 4C 58 8A  JMP INTCHR  Then set character input.
```

NOTE: the jump at 1E0E can be changed to point to your own input routine, if different.

SUPERMON EXTENSIONS

Elsewhere in this issue is the program RELOCATE, and in the introductory issue appeared the program DISASSEMBLE. MON 1.1 currently has the zero parameter command .R for REGISTER and the zero or one parameter command .D for DEPOSIT. The listing for RELOCATE shows how to add a two parameter .D for DISASSEMBLE and a three parameter .R for RELOCATE.
If you have entered DISASSEMBLE at 2000 and RELOCATE (including the monitor patch) at 2200, you can patch the new commands to MON 1.1 by using the command .G 2200 after every RESET.

It should be obvious from the listing how other commands can be added and how the default values entered into system RAM by RESET can be replaced at the same time. The CLC prior to the RTS at line 228E is to ensure that no error message will be printed at the return from RELOCATE, and may be required in your other extensions.

To call DISASSEMBLE enter, for example, .D 2200,16. The 2200 is the start of the program to be disassembled, the 16 will give you 22 lines on a CRT terminal.

A PROGRAM TO DISPLAY SYM-1 LED SEGMENT CODES

By Maurie Du Feu
P.O. Box 257
Lindfield, NSW 2070
Australia

(Orginally written Nov '78; modified and documented by the editor Nov '79)
This short but elegant program should prove very helpful to those SYMMERS who wrote in asking for more information on how SUPERMON subroutines can be used in their own programs. It illustrates how DISBUF and RDIG at A640-A645 are used, but even more importantly, it illustrates the use of TV at A656 and DELAY at 835A. The value of TV (=09) at 0203 gives a reasonable counting rate; try varying it. DELAY permits over-riding the chosen value of TV, since every time any key on the Hex Keypad (or on the terminal, if you started the program with the terminal) is hit, an additional count is added. If TV=0 only the keys will produce a count. The two left hex digits will count from 00-FF and the right-most digit will display the symbol for which these digits are the segment code.

After you have this program entered and working you can replace the redundant JSR ACCESS at 0223 and the JSR SCAND at 0244 with NOPs (since SCAND is included in DELAY) and practice using RELOCATE and BLOCKMOVE to save the 6 bytes. The FF at 024F marks the end of the instructions (for use of RELOCATE). From 0250 to 025F are the segment codes for the hex digits 0 to F.

```
.V 200,25F
0200 20 86 8B A9 09 8D 56 A6,6C
0208 A9 00 85 F3 A9 50 85 F6,01
0210 A9 02 85 F7 D8 A5 F3 4A,E2
0218 4A 4A 4A 85 F5 A5 F3 29,FB
0220 0F 85 F4 20 86 8B A4 F5,4D
0228 B1 F6 8D 40 A6 A4 F4 B1,B0
0230 F6 8D 41 A6 A9 00 8D 42,92
0238 A6 8D 43 A6 8D 44 A6 A5,CA
0240 F3 8D 45 A6 20 06 89 20,04
0248 5A 83 E6 F3 4C 15 02 FF,1C
0250 3F 06 5B 4F 66 6D 7D 07,62
0258 7F 67 77 7C 39 5E 79 71,BC
 2FBC
.
```

SUGGESTED HARDWARE MODIFICATION

By George Wells, 1620 Victoria Place, La Verne, CA 91750

I have a very simple solution to the problem of not having enough PROM sockets on board the SYM-1. All you do is stack the ROM's one on top of the other on one socket and hardwire individual chip-selects with pull-up resistors to each of them except the bottom one. I have done this with my two BASIC ROM's in socket U21 and it works fine. I plan to put the two RAE-1/2 ROM's into socket U22 and leave U23 available for a 2716 EPROM. This is what I did:

1. Remove Jumper L-13.
2. Add 3.3K resistor from pad 13 to feed-throush pad near pin 16 of I.C. U11 (+5V).
3. Remove ROM 02-0020-01 from socket U22 and spring its leads slightly inward except for pin 20 (chip select) which should be bent slightly outward.
4. Place ROM 02-0020-01 on top of ROM 02-0019-01 making sure that all corresponding pins are making contact with each other except of course pin 20. Secure with tape if desired.
5. Hand wrap a piece of 30-gauge wire around pin 20 of the top ROM and bring the other end to the jumper between pads 13 and 14.

Everybody ought to get Blalock's 4K RAM expansion board. Imagine that--30K of ROM/RAM on board the SYM. I'm going to have to get a new power supply!

FAST FOURIER TRANSFORM

Here is a Fast Fourier Transform program for those who requested it. The graph drawing portion has been omitted because it was terminal dependent; you will have to add your own. This program is based very closely on the program given by William D. Stanley and Steven J. Peterson in "Fast Fourier Transforms on Your Home Computer," BYTE Magazine, December, 1978. First a sample run, then the listing:

```
RUN
Number of samples? 16

The input function is of the form
sin (2 pi f t) + 0.5 cos (4 pi f t),
where f is the frequency.

Frequency? 3
List input function ? Y
 N    Real            Imag            Mag             Phase
 -    ----            ----            ---             -----
 0     .500000001     0                .500000001      0
 1     .570326142     0                .570326142      0
 2     .707106782     0                .707106782      0
 3    -.0291300428    0                .0291300428     180
 4    -1.5            0                1.5             180
 5    -.0291300383    0                .0291300383     180
 6     .707106781     0                .707106781      0
 7     .570326143     0                .570326143      0
 8     .499999997     0                .499999999      0
 9    -1.27743292     0                1.27743292      180
 10   -.707106773     0                .707106779      180
 11    .736236824     0                .736236825      0
 12    .5             0                .5              0
 13    .736236823     0                .736236823      0
 14   -.707106791     0                .707106792      180
 15   -1.27743292     0                1.27743292      180
 16    .500000001     0                .500000001      0
Scaling
Computing the FFT
Resequencing
List transform? Y
 N    Real            Imag            Mag             Phase
 -    ----            ----            ---             -----
 0    0               0               0
 1    0               0               0
 2    0               0               0
 3    0               -.5             .5              270
 4    0               0               0
 5    0               0               0
 6    .25             0               .25             0
 7    0               0               0
 8    0               0               0
 9    0               0               0
 10   .25             0               .25             0
 11   0               0               0
 12   0               0               0
 13   0               .5              .5              90
 14   0               0               0
 15   0               0               0
 16   0               0               0
 Finished
```



```
1000 INPUT"Number of samples? ";N:M=LOG(N)/LOG(2)
1020 DIMXR(N),XI(N):PI=3.1415926535
1040 PRINT:REM DEFINE THE INPUT FUNCTION
1060 PRINT"The input function is of the form"
1080 PRINT"sin (2 pi f t) + 0.5 cos (4 pi f t),"
1100 PRINT"where f is the frequency.":PRINT
1120 INPUT"Frequency? ";F:F=2*F*PI/N
1140 FORI=0TON
1160 F1=F*I:XR(I)=SIN(F1)
1180 XR(I)=XR(I)+.5*COS(2*F1)
1200 XI(I)=0:NEXT
1220 XR(0)=XR(0)/2+XR(N)/2:XR(N)=XR(0)
1240 REM END OF FUNCTION DEFINITION
1260 INPUT"List input function ? ";A$
1280 IFLEFT$(A$,1)="Y"THENGOSUB1820
1300 PRINT"Scaling"
1320 FORI=0TON:XR(I)=XR(I)/N:XI(I)=XI(I)/N:NEXT
1340 W=-1:GOTO1380
1360 W=+1:REM ENTER HERE FOR INVERSE
1380 PRINT"Computing the FFT"
1400 I1=N/2:I2=1:V=2*PI/N
1420 FORJ=1TOM:I3=0:I4=I1
1440 FORK=1TOI2:X=INT(I3/I1):GOSUB2200
1460 I5=Y:ZR=COS(V*I5):ZI=W*SIN(V*I5)
1480 FORL=I3TOI4-1:AR=XR(L):AI=XI(L)
1500 BR=ZR*XR(L+I1)-ZI*XI(L+I1)
1520 BI=ZI*XR(L+I1)+ZR*XI(L+I1)
1540 XR(L)=AR+BR:XI(L)=AI+BI
1560 XR(L+I1)=AR-BR:XI(L+I1)=AI-BI
1580 NEXTL:I3=I3+2*I1:I4=I4+2*I1
1600 NEXTK:I1=I1/2:I2=I2*2:NEXTJ
1620 XR(N)=XR(0):XI(N)=XI(0)
1640 PRINT"Resequencing"
1660 FORK=0TON-1:X=K:GOSUB2200
1680 IFK<=YTHEN1740
1700 TR=XR(K):TI=XI(K):XR(K)=XR(Y):XI(K)=XI(Y)
1720 XR(Y)=TR:XI(Y)=TI
1740 NEXT
1760 INPUT"List transform? ";A$
1780 IFLEFT$(A$,1)="Y"THENGOSUB1820
1800 PRINT" Finished":END
1820 PRINT" N";TAB(6);"Real";TAB(22);"Imag";TAB(38);"Mag";TAB(54);"Phase
1840 PRINT" -";TAB(6);"----";TAB(22);"----";TAB(38);"---";TAB(54);"-----
1860 FORI=0TON:XR=XR(I)
1880 XI=XI(I)
1900 IFABS(XR)<1E-4THENXR=0
1920 IFABS(XI)<1E-4THENXI=0
1940 XM=SQR(XR^2+XI^2)
1960 IFXM=0THENXP=0:GOTO2040
1980 IFXM=0THENXP=0:GOTO2100
2000 IFXR=0THENXP=SGN(XI)*90:GOTO2040
2020 XP=180/PI*ATN(XI/ABS(XR))
2040 IFXR<0THENXP=180-XP
2060 IFXP<0THENXP=360+XP
2080 XP=1E-2*INT(1E2*XP+.5)
2100 PRINTI;TAB(5);XR;TAB(21);XI;
2120 PRINTTAB(37);XM;
2140 IFXM=0THENPRINT:GOTO2180
2160 PRINTTAB(53);XP
2180 NEXT:RETURN
2200 Y=0:N1=N:FORI=1TOM:N1=N1/2:IFX<N1THEN2240
2220 Y=Y+2^(I-1):X=X-N1
2240 NEXTI:RETURN
```

MISCELLANEOUS GOOD THINGS

HARDWARE: If you are into control applications, or otherwise need to do lots of prototyping, the "First Mate" provides a very generous working area immediately above the SYM, effectively on-board. From one to three "Second Mates" bring signals from the three edge connectors on the SYM-1 (without interfering with motherboard connections) to sockets on First Mate. Write to Richard Turpin, MicroMate, P.O. Box 50111, Indianapolis, IN 46256 for information and prices.

SOFTWARE: One of the best ways to improve your programming abilities is to study outstanding examples. One such example is the GRAPHICS DRAWING COMPILER for SYM, by Hall and Moser. While the GDC has value in and of itself, its real value to me was the know-how it provided on the design of compilers, which translate programs writen in high-level languages into machine language code for fast execution. As an example, after you have defined Macros CAR and CLRCAR to draw and erase the image of an automobile on the CRT screen, the program to move the automobile 10 units to the right is written in only 7 instructions, with a DO loop:

```
        DEFINE (J 10)
        DO (EXIT J)
        CLRCAR
        CAR
        POSREL (0 1)
        END
EXIT
```

RAE-1 and the GDC then generate the machine language code to do the job. For additional information and prices write Carl Moser, Eastern House Software, 3239 Linda Drive, Winston Salem, NC 27106.

SPEECH SYNTHESIS: The SP-1 Speech Synthesizer Interface ($49 Kit, $69 assembled, includins commented source listings) is available to interconnect the SYM-1 to the Texas Instruments "Speak & Spell (TM)" toy (around $49). The interface PC board fits into the S & S in place of the battery and is driven through the 6522 VIA. One of the programs supplied is a hex digit speaker, to read back programs or data you have entered. Imagine enterins data with .D or .M and gettins SYM to sive you verbal feedback! Much better than the beeper. For more information contact Dave Kemp, East Coast Micro Products, 1307 Beltran Court, Odenton, MD 21113.

MISCELLANEOUS NOTES

If you have the first printing of the MON 1.1 Enhancement Description be aware that all addresses in the listing from 84F4 through 8C70 should be increased by 5. The missing instructions are:

```
        84F4 F0 03     BEQ DEPN
        84F6 20 20 83 JSR OUTQM  ;TYPE "?" IF NG
```

Cross reference table has correct addresses. (Info from Nick Vrtis.)

The first printing of the RAE-1 Reference Manual has a typographical error in reference to memory locations. RAE-1 actually resides in memory locations B000-BFFF and E000-EFFF, and there is no conflict with BAS, which co-exists at C000-CFFF and D000-DFFF. If you were able to set the correct jumpers for BAS you should be able to figure out how to do so for RAE. Since you will be in the neighborhood anyway, you might want to rearrange the jumpers to permit two BAS chips to live as cheaply as one in the same socket, as described elsewhere in this issue.

The new address for the 6502 USER NOTES is: Eric Rehnke, 540 S. Ranch View Circle, Apt. 61, Anaheim, CA 92807.

THE PARITY BIT "PROBLEM"

Gary Humphrey's program "Input Lower Case . . ." has a very subtle bug which can only be exterminated by using the Synertek published version. We elected to publish it "in spite of" the bug because it does illustrate a novel use of the stack and because the bug itself is of some interest. The program will work for his terminal (Video-Plus) and many others, but can cause hangups on terminals which generate parity bits, such as for example, the KSR-35 TTY. Sgt. Humphrey is using TINY BASIC, and probably (as do I) uses INTCHR to permit lower case input. Those with TINY will notice that soon after the JSR to the input link at 06B7, the accumulator is ANDed with #7F to clear parity at 06BE before all the checks for NUL, RUBOUT, CR, etc. are made.

When I first used 2KSA, over a year ago, I used INTCHR for its input, and it took me many hours of experimentins spread out over several weeks to find out why 2KSA worked at school with an ASR-33 but not at home with a KSR-35. Even went to the trouble of bringing the -33 home and the -35 to school. Only when I found that the trouble was with the terminal, and not its location, did I write a simple program to print out the ASCII codes for the input characters to find out that the -35 used a parity bit. The 2KSA does not erase the parity bit (but it does not use lower case either) so INCHR is its required input point. Can't blame Denison for this omission, since it was not needed for KIM.

SYM-PATHY

One of our overseas subscribers points out that a KTM-2 costs over $700 in his country, and that it will be a long time before he can afford one. He does have an oscilloscope, however, and asks for help in its use. My recommendation to him, and to others, in the same position is as follows:

First, implement the Oscilloscope Output Feature as described in Chapter 7 of the SYM Reference Manual, and study the listing until you understand every line of the program, including the remark following line 0088. If you note that the hex keypad has 25 keys whose values are assigned through software, you will see that you can use it as a typewriter keyboard by giving each key two meanings. Dedicate one key to shifting or unshifting the following keystrokes, and you can then enter 48 alphanumerics, punctuation marks, and some control characters with the hex keypad. Use a paper overlay over the pad, with the dual names for each key written on it. Replace the ASCII table in ROM and the character set table in page 04 by tables of your own design and write a program that will enable you to use the hex keypad and the scope as a 32 character, one line, terminal. If you add a third table to generate the correct ASCII values for your keystrokes you can change INVEC and OUTVEC to point to these new I/O routines you have written. If you can interface an inexpensive QWERTY keyboard to one of the PIAs your input will be easier, but the output will be the same. The DISASSEMBLE program can easily be modified to use the scope as its output. Hardware is the big expense for hobbyists, software costs only time and patience! Before I got my terminal, I modified a clock program for the SYM (A Digital Clock Program for the SYM-1, Chris Sullivan, 9 Galsworthy Place, Bucklands Beach, Auckland, New Zealand, in MICRO No. 7, Oct-Nov 1978, pp 45-46) to use the scope display in place of the LED Display, so I have a feeling for how to do the job. Will be happy to correspond with anyone who needs help, and will publish the first successful Scope Terminal Program submitted. I am giving Mr. Sullivan's address as published in MICRO for the benefit of all of our many Australia/New Zealand subscribers.

ALL PRICES GIVEN BELOW ARE NOW OBSOLETE. PLEASE USE PRICES GIVEN ON THE MOST RECENTLY ISSUED 'SHOPPING LIST'.

SHOPPING LIST OF ITEMS AVAILABLE FROM SYM-1 USERS' GROUP

2K SYMBOLIC ASSEMBLER MANUAL (BY ROBERT DENISON)
$10.25 IN US FUNDS-FIRST CLASS POSTPAID FOR NO. AMERICAN COUNTRIES.
$11.00 IN US FUNDS-AIR MAIL POSTPAID FOR EUROPEAN COUNTRIES.
$12.00 IN US FUNDS-AIR MAIL POSTPAID FOR ASIA/PACIFIC COUNTRIES.

2K SYMBOLIC ASSEMBLER ON CASSETTE TAPE
$5.35 IN US FUNDS-FIRST CLASS POSTPAID FOR NO. AMERICAN COUNTRIES.
$5.75 IN US FUNDS-AIR MAIL POSTPAID FOR EUROPEAN COUNTRIES.
$6.75 IN US FUNDS-AIR MAIL POSTPAID FOR ASIA/PACIFIC COUNTRIES.

SYNERTEK TECHNICAL NOTES
$4.10 IN US FUNDS-FIRST CLASS POSTPAID FOR NO. AMERICAN COUNTRIES.
$4.60 IN US FUNDS-AIR MAIL POSTPAID FOR EUROPEAN COUNTRIES.
$5.60 IN US FUNDS-AIR MAIL POSTPAID FOR ASIA/PACIFIC COUNTRIES.

SUPERMON VERSION 2(MON 1.1)
$16.00 IN US FUNDS-FIRST CLASS POSTPAID FOR NO. AMERICAN COUNTRIES.
$17.00 IN US FUNDS-AIR MAIL POSTPAID FOR EUROPEAN COUNTRIES.
$18.00 IN US FUNDS-AIR MAIL POSTPAID FOR ASIA/PACIFIC COUNTRIES.

RAE-1/2 (THIS IS THE TWO CHIP VERSION BUT BOTH CHIPS MAY BE MOUNTED IN ONE SOCKET. FULL INSTRUCTIONS SUPPLIED ALONG WITH EXCLUSIVE UPDATING SERVICE DESCRIBING ADDITIONAL FEATURES NOT DESCRIBED IN THE SYNERTEK MANUAL).
$99 IN US FUNDS-AIR MAILED POSTPAID-INSURED ANYWHERE IN THE WORLD.

SCHEMATIC DIAGRAM OF SYM-1
$1.50 IN US FUNDS-AIR MAIL POSTPAID ANYWHERE IN THE WORLD.

We are truly sorry to have to raise our shipping and handling charges but we found we underestimated the cost of the packing materials and postage after we were swamped with orders for the above items. The prices quoted in the Introductory Issue did not take care of the costs, and we simply mailed the item and paid for it out of pocket. Some of our Symmers, themselves, realized our error and remitted a little more than was quoted in SYM-PHYSIS. We thank them for being so thoughtful. We know our overseas buyers can save money on the postage by letting us send items by slow boat, but our policy is to send everything Air Mail unless otherwise requested.

A good Source for Cassette Tapes is BOB MYERS, 109 FIRE LANE, NORTH CAPE MAY, NJ 08204. His prices are $6.50 US for 10-50 foot Cassettes in soft plastic boxes-for hard boxes please add 50 cents. 4th. class postpaid.

We will be teaching a week-end course called 'Microprocessor Fundamentals,' for the University of California at Davis, January 25-27, 1980. The course fee is $450.00, and each student will receive a SYM-1 plus some software goodies. If interested, please contact, University Extension, University of California, Davis, Davis,CA 95616.

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

TIME VALUE PRINTED MATTER

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

Address Correction Requested