SYM-PHYSIS

THE SYM USERS' GROUP NEWSLETTER

VOLUME IV, NUMBER 3 (ISSUE NO. 17) - WINTER 1983 (SEP/OCT/NOV/DEC)

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

SUBSCRIPTION RATES:

This is the FAREWELL ISSUE of SYM-PHYSIS, and subscriptions for future issues are no longer being accepted. Only Back Issues, in complete volumes, are available, as noted below. Make checks payable in US dollars to "SYM Users' Group", 20 Sunland Drive, Chico, CA 95926, Telephone (916) 895-8751.

BACK ISSUES:

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.

Issues 15 through 17 (Volume IV, 1983), are available for $10.50, US/Canada, and $14.00, First Class/Airmail, elsewhere.

L'ENVOI

Our original KIM-1, mounted on a large masonite panel, with an added 4 K of RAM, and a few assorted add-ons, is languishing in a dark corner of the garage. A second KIM-1, which was inoperative when we traded a new SYM-1 to a student for it, has been repaired, and now sits neatly boxed on a shelf. Also on the shelf is an odd lot of miscellaneous single board computers: an AIM-65, a SYM-69, a Sinclair ZX81, some RCA COSMAC VIPs, etc. None of these occupied the places in our minds and hearts that the SYM-1 did.

While the KIM-1 got us started, with the help of the 6502/KIM-1 Users' Group, it was the SYM-1, with the help of so many members of the SYM Users' Group, that really taught us how computers actually do work, deep down inside the operating system.

The KIM-1, so named after its 2 K ROM operating system, the Keyboard Interface Monitor (or Module) made a good entry level system, but the SYM-1, nee VIM-1 after its 4 K ROM operating system, the Versatile Interface Monitor (or Module), taught us all at a much more sophisticated level, thanks to all of the capabilities packed into MON 1.1, and RAE-1, which we still consider to be the very best full-featured 6502 Conditional/Macro Resident Assembler Editor around for the 6502, despite of, or perhaps even because of, its use of non-

(continued to page 17-6)

A 3-D GRAPHICS PACKAGE

Here is the BASIC portion of a 3-Dimensional Manipulation Package developed over a year ago as a class project by a former student, Tim Calhoun. We saw it demonstrated, and it worked very well, indeed, but we cannot now use it as it is, or provide sample graphic printouts, because our MTU Visible Memory SYM-1 system has been much too reconfigured to permit its easy use (BAS is now in RAM, VM is relocated, no cassette interface, etc.) with this program.

Even though very few readers may have Visible Memories on their SYM-1 systems (our main reason for not publishing it earlier), we are publishing it at this late date for three reasons:

One is the valuable collection of 4 x 4 matrix manipulation subroutines; second is its adaptability to 40 x 24 (80 x 48) graphics on the KTM-2 or 80 x 24 (160 x 48) graphics on the KTM-2/80. Third is the adaptability to the COM-64 in either the low resolution or the high resolution graphics modes. We'll provide copies of this listing to several of our friends (we no longer have students, since our retirement; they are now friends!) with COM-64s, to see what they can do with it.

In this connection, we should point out that the Visible Memory uses a direct linear mapping of its 320 x 192 pixels to RAM, while the COM-64 uses a mapping compatible with an 8 x 8 Character Generator Matrix. The Apple II/IIe uses a more "indirect" mapping (with 280 x 192 grid), while dot matrix printers in their graphics modes require an additional "remapping" to accomodate the vertical stacking of the printing "pins".

All of these remappings could easily be accomplished, if desired, by adding subroutines to the published program. We needed an Apple to Visible Memory remappimg when we uploaded the public domain Apple SLIDE SHOW to our SYM-1, and will be needing a Visible Memory to COM-64 remapping when we download SLIDE SHOW to our COM-64.

```
 1 REM*****************************************************************
 2 REM
 3 REM      PROGRAM: SYM-1 3-DIMENSIONAL MANIPULATION PACKAGE
 4 REM      PROGRAMMER: TIM CALHOON
 5 REM      DATE: 12-6-82
 6 REM
 7 REM      FUNCTION:TO ALLOW THE USER TO CREATE, MANIPULATE, DISPLAY,
 8 REM               AND SAVE THREE-DIMENSIONAL OBJECTS.
 9 REM
10 REM      HARDWARE NEEDED: SYM-1 WITH 8K MEMORY, MTU VISIBLE MEMORY,
11 REM                       AND KTM TERMINAL.
12 REM      EXTERNAL SOFTWARE: HUGH E. CRISWELL'S BASIC SAVE AND LOAD
13 REM                         SUBROUTINES AND SYNERTEK'S TRIG-PATCH
14 REM
15 REM*****************************************************************
50 REM
51 REM
52 REM
53 REM*****************************************************************
54 REM
55 REM      MAIN LINE ROUTINE:
56 REM       FIRST IT INITIALIZES SOME ESSENTIAL VALUES AND THEN IT
57 REM       SETS TEMP AND HOLD TO IDENTITY MATRICES.
58 REM       SECOND IT DISPLAYS THE MENU AND ASKS THE USER FOR
59 REM       A CHOICE.
60 REM       LASTLY IT BRANCHES TO THE APPROPRIATE SUBROUTINE, THEN
70 REM       LOOPS AROUND AND RE-DISPLAYS THE MENU.
71 REM*****************************************************************
```

```
100 DIM PROD(4,4):PI=3.1415
105 DIMHOLD(4,4),TEMP(4,4),D(7,30),MM(4),C(2,4)
107 LGTH=0
110 GOSUB600
120 FORA=1TO4:FORB=1TO4:HOLD(A,B)=TEMP(A,B):NEXT:NEXT
150 PRINTCHR$(27)+"E";:FORP=1TO9:PRINT:NEXT
170 PRINT"1. DISPLAY","2. ADD FILE","3. LOAD FILE"
180 PRINT"4. SAVE FILE","5. ROTATE","6. SHIFT"
190 PRINT"7. SCALE","8. ORIGINAL","9. CLEAR"
200 PRINT"10. DELETE","11. LIST FILE","12. EXIT"
205 PRINT:INPUT"INPUT NUMBER NEXT TO CHOICE ";A
210 IFA<1ORA>12THEN205
220 ONAGOSUB300,700,1800,1900,1000,1490,1520,110,1700,800,900,2000
230 GOTO150
250 REM************************************************************
251 REM
252 REM      PERSPECTIVE SUBROUTINE:
253 REM       GOES THROUGH FILE CREATING PERSPECTIVE X AND
254 REM       Y VALUES FROM 3-D COORDINATE FILE USING A DIVISION OF
255 REM       SIMILAR TRIANGLES METHOD WITH VIEWPOINT ON THE Z-AXIS
256 REM       AND SENDING THOSE X,Y VALUES TO A DDA ROUTINE.
257 REM
258 REM************************************************************
300 INPUT"INPUT DISTANCE FROM ORIGIN ";DST:MM(4)=1
310 FORR=1TOLGTH:Q=1:FORS=1TO4STEP3
330 MM(1)=D(S,R):MM(2)=D(S+1,R):MM(3)=D(S+2,R)
335 IFS=4THENQ=2
340 FORJ=1TO4:C(Q,J)=0:FORK=1TO4
350 C(Q,J)=C(Q,J)+MM(K)*HOLD(K,J):NEXT:NEXT:NEXT
390 Y1=(C(1,2)*DST)/(DST+C(1,3)):Y2=(C(2,2)*DST)/(DST+C(2,3))
400 X1=(C(1,1)*DST)/(DST+C(1,3)):X2=(C(2,1)*DST)/(DST+C(2,3))
401 REM************************************************************
402 REM
403 REM      DDA LINE DRAWING SUBROUTINE:
404 REM
405 REM      THIS ROUTINE USES THE SIMPLE DDA ALOGRITHM FOR DRAWING
406 REM      A LINE BETWEEN TWO GIVEN POINTS DEFINED BY X1,X2 AND
407 REM      Y1,Y2.
408 REM
409 REM************************************************************
410 LNTH=ABS(X2-X1):IFABS(Y2-Y1)>LNTHTHENLNTH=ABS(Y2-Y1):X=X1:Y=Y1
415 X=INT(X1+159):Y=INT(Y1+99)
420 GOSUB 500
440 IFLNTH=0THENNEXT
450 DX=(X2-X1)/LNTH:DY=(Y2-Y1)/LNTH
455 XA=X1+.5:YA=Y1+.5
460 FORB=1TOLNTH
465 XA=XA+DX:YA=YA+DY:X=INT(XA+159):Y=INT(YA+99)
480 GOSUB500
490 NEXT:NEXT:RETURN
500 VM=8192+((199-Y)*40+INT(X/8)):BIT=((X/8)-INT(X/8))*8:DOT=2^BIT
505 IFVM<8192ORVM>16383THENRETURN
510 MASK=PEEK(VM):DOT=128/DOT:DOT=MASKORDOT:POKEVM,DOT:RETURN
550 REM************************************************************
551 REM
552 REM      TEMP = IDENTITY SUBROUTINE
553 REM
554 REM      THIS ROUTINE SETS THE TEMP MATRIX TO THE IDENTITY MATRIX.
558 REM
559 REM************************************************************
600 FORI=1TO4:FORJ=1TO4:TEMP(I,J)=0:NEXT:NEXT
610 FORI=1TO4:TEMP(I,I)=1:NEXT:RETURN
650 REM************************************************************
651 REM
```



```
652 REM      ADD FILE SUBROUTINE:
653 REM      THIS ROUTINE ACCEPTS COORDINATE VALUES FROM THE USER
654 REM      AND ADDS THEM INTO THE FILE AT ITS END.  IF A -999
655 REM      IS FOUND IN THE X1 POSITION THE RETURN IS EXECUTED.
656 REM
657 REM************************************************************
700 LGTH=LGTH+1:PRINT"INPUT -999 IN X1 POSITION AND 0 IN REST TO RETURN
710 PRINT"VECTOR ";LGTH
720 PRINT"INPUT X1,Y1,Z1,X2,Y2,Z2"
730 INPUTD(1,LGTH),D(2,LGTH),D(3,LGTH),D(4,LGTH),D(5,LGTH),D(6,LGTH)
735 IFD(1,LGTH)=-999ORLGTH>30THEN750
740 D(7,LGTH)=LGTH:GOTO700
750 D(1,LGTH)=0:LGTH=LGTH-1:RETURN
760 REM************************************************************
761 REM
762 REM      DELETE SUBROUTINE:
763 REM
764 REM      THIS ROUTINE DELETES THE LINES BETWEEN THE STARTING AND
765 REM      ENDING LINES VALUES. IT THEN MOVES THOSE LINES ABOVE THE
766 REM      DELETED AREA DOWN TO FILL THE SPACES LEFT AFTER THE
767 REM      DELETE.
768 REM
769 REM************************************************************
800 INPUT"INPUT START AND FINISH OF DELETE ";S,F
810 IFF>30THENRETURN:IFS>FTHENRETURN
815 IFF>LGTHTHENF=LGTH
820 IF F=LGTHTHEN860
830 FORI=(F+1)TOLGTH:FORJ=1TO7:D(J,S)=D(J,I):NEXT
840 D(7,S)=S:S=S+1:NEXT:LGTH=S-1:RETURN
860 LGTH=LGTH-((F+1)-S):RETURN
870 REM************************************************************
871 REM
872 REM      LIST FILE SUBROUTINE:
873 REM
874 REM      THIS ROUTINE LISTS THE RECORDS FROM A GIVEN
875 REM      STARTING RECORD NUMBER TO THE ENDING RECORD NUMBER.
876 REM
877 REM
878 REM************************************************************
900 INPUT"INPUT START AND FINISH OF LISTING ";S,F
910 FORI=STOF
915 IFI>LGTHTHEN940
920 PRINTI,D(1,I),D(2,I),D(3,I)
930 PRINT,D(4,I),D(5,I),D(6,I)
935 PRINT:NEXT
940 INPUT"INPUT 1 TO GET MENU ";A
950 RETURN
960 REM************************************************************
961 REM
962 REM      ROTATION SUBROUTINE:
963 REM
964 REM      THIS ROUTINE FIRST FINDS A CENTER FOR THE OBJECT,
965 REM      THEN TRANSLATES THE OBJECT TO THE ORIGIN, THEN
966 REM      BRANCHES TO THE APPOPRIATE SUBROUTINE TO EXECUTE
967 REM      A ROTATION ABOUT THE X,Y OR Z AXIS, AND FINALLY
968 REM      TRANSLATES THE OBJECT BACK AGAIN.
969 REM
970 REM************************************************************
1000 XGT=D(1,1):XLT=XGT:YGT=D(2,1):YLT=YGT:ZGT=D(3,1):ZLT=ZGT
1010 FORI=1TOLGTH:FORJ=1TO4STEP3:
1020 IFXGT<D(J,I)THENXGT=D(J,I):IFXLT>D(J,I)THENXLT=D(J,I)
1030 IFYGT<D(J+1,I)THENYGT=D(J+1,I):IFYLT>D(J+1,I)THENYLT=D(J+1,I)
1040 IFZGT<D(J+2,I)THENZGT=D(J+2,I):IFZLT>D(J+2,I)THENZLT=D(J+2,I)
1050 NEXT:NEXT
```

```
1060 DX=-(((XGT-XLT)/2)+XLT):DY=-(((YGT-YLT)/2)+YLT)
1061 DZ=-(((ZGT-ZLT)/2)+ZLT)
1065 GOSUB600
1070 GOSUB1497
1080 INPUT"INPUT 1,2,3 FOR ROTATION ABOUT X,Y,ORZ AXIS ";B
1085 IFB<1ORB>3THEN1080
1086 INPUT"INPUT ANGLE OF ROTATION ";ANG
1087 ANG=ANG*PI/180
1090 ONBGOSUB1200,1300,1400
1100 DX=-DX:DY=-DY:DZ=-DZ:GOSUB600
1105 GOSUB1497
1110 RETURN
1150 REM*********************************************************
1151 REM
1152 REM      X,Y, AND Z ROTATION SUBROUTINES:
1153 REM
1154 REM      THESE ROUTINES SET VALUES IN 4 X 4 MATRICES AND
1155 REM      CONCATENATE THEM, THROUGH MULTIPLICATION INTO
1156 REM      A RESULTANT MATRIX TO BE USED TO SET TRANSFORMED
1157 REM      X,Y,Z VALUES IN THE PERSPECTIVE ROUTINE.
1158 REM
1159 REM*********************************************************
1160 REM
1161 REM
1162 REM*****************************************************
1163 REM      X-ROTATE
1164 REM*****************************************************
1200 GOSUB600
1210 TEMP(2,2)=COS(ANG):TEMP(2,3)=SIN(ANG):TEMP(3,2)=-SIN(ANG)
1220 TEMP(3,3)=COS(ANG):GOSUB1600
1230 RETURN
1250 REM*****************************************************
1251 REM      Y-ROTATE
1252 REM*****************************************************
1300 GOSUB600
1310 TEMP(1,1)=COS(ANG):TEMP(1,3)=-SIN(ANG):TEMP(3,1)=SIN(ANG)
1320 GOTO1220
1350 REM*****************************************************
1351 REM      Z-ROTATE
1352 REM*****************************************************
1400 GOSUB600
1410 TEMP(1,1)=COS(ANG):TEMP(2,1)=-SIN(ANG):TEMP(1,2)=SIN(ANG)
1420 TEMP(2,2)=COS(ANG):GOSUB1600
1430 RETURN
1490 GOSUB600
1491 REM*****************************************************
1492 REM      SHIFT SUBROUTINE:
1493 REM      SETS UP 4 X 4 MATRIX FOR A GIVEN SHIFT ALONG X,Y, OR Z
1494 REM*****************************************************
1495 INPUT"INPUT SHIFT IN X,Y,Z ";DX,DY,DZ
1497 TEMP(4,1)=DX:TEMP(4,2)=DY:TEMP(4,3)=DZ
1500 GOSUB1600
1510 RETURN
1520 GOSUB600
1521 REM*********************************************************
1522 REM
1523 REM      SCALE SUBROUTINE:
1524 REM      SETS UP A 4 X 4 MATRIX FOR SCALE ON GIVEN X,Y, AND Z
1525 REM      AXISES.
1526 REM*********************************************************
1530 INPUT"AMOUNT OF SCALE IN X,Y,Z ";TEMP(1,1),TEMP(2,2),TEMP(3,3)
1540 GOSUB1600
1550 RETURN
```



```
1551 REM*********************************************************
1552 REM
1553 REM      4 X 4 MATRIX MULTIPLICATION TO CONCATENATE A FUNCTION
1554 REM      INTO THE HOLD MATRIX.
1555 REM*********************************************************
1600 FORA=1TO4:FORB=1TO4:PROD(A,B)=0:FORK=1TO4
1610 PROD(A,B)=PROD(A,B)+HOLD(A,K)*TEMP(K,B):NEXT:NEXT:NEXT
1620 FORA=1TO4:FORB=1TO4:HOLD(A,B)=PROD(A,B):NEXT:NEXT:RETURN
1700 FORI=8192TO16383:POKEI,0:NEXT:RETURN
1750 REM*********************************************************
1751 REM      CALL TO CRISWELLS DATA LOAD ROUTINE
1752 REM*********************************************************
1800 INPUT"SET UP FOR CASSETTE LOAD AND INPUT FILE NUMBER ";ID
1810 X=USR(7807,256*ID):RETURN
1900 PRINT"FILE WILL BE SAVED 3 TIMES"
1901 REM*********************************************************
1902 REM      CALL TO CRISWELLS DATA SAVE ROUTINE
1903 REM*********************************************************
1910 INPUT"SET UP FOR CASSETTE SAVE AND INPUT FILE NUMBER ";ID
1920 FORI=1TO3:X=USR(7718,256*ID):NEXT:RETURN
2000 END
```

L'ENVOI (continued from page 17-1)

standard MOS Technology pseudo op-codes (this may be the longest non-stop one sentence paragraph we've ever written!).

We'll never forget all we learned from the SYM, or from the many fine SYMmers we met by mail, telephone, or personal contacts on past and future visits to us, or travels to them. Thanks to everyone from both of us.

RAM-BLINGS

In the B. C. (Before Computer) era, our home had a number of leisure- and/or pleasure-type rooms, including, among others, a guest room, a family room with a magnificent fireplace and the main TV ("telly"), and a combination den/study where our SYM-1 setup lived on a corner of our desk. Jean suggested that the SYM system be set up on a card table in the family room so that there could be some "togetherness", with her watching the TV while we watched the KTM-2 monitor.

Today the guest room is now Jean's cluttered office, the study/den is now a cluttered warehouse/storeroom, and the family room is now a cluttered computer laboratory/workshop, with desks, workbenches, book shelves, and filing cabinets along all four walls (effectively blocking the fireplace), and a custom made 4' x 8' computer table in the center of the room. And the elegant darkroom in one corner of the garage is now unuseable, having become a catch-all storeroom.

Thus the SYM has very strongly affected our way-of-life, as well as our standard of living. By chance, or whatever, we and the SYM are retiring simultaneously, but it will probably take several years for our home to return to "normal", if ever. We won't even begin to think about it till our European trip is over!

Being a SYMmer almost implied being a "loner", somewhat akin to the "loneliness of the long distance runner", since very few of us were in a position to be able to work closely with other SYMmers, and to swap hardware, software, and, most importantly, ideas, either in a one-on-one or on a "live" group basis. There have never been more than two or three SYMmers in or near Chico, at any given time, with whom we could work closely. That, in fact, was why the Users' Group was started! At no time, however, were there more than 2000 members.

There were many times when we almost envied our Apple owning friends, and we did think occasionally of switching. It was not the cost that stopped us, it was the realization of how little we actually would be getting for our money, compared with what the SYM was giving us. But, even more importantly, we had made so many new friends, by mail, telephone, and personal contacts, and thus were receiving so much more personal satisfaction from being a SYMmer then we could ever have gotten from the Apple.

Our five "main" SYMs are now mostly idle. One is used by Jean to handle the cassette and FODS diskette software duplication, the mailing list, and the accounts payable records. That one still gets the most use, but its active days are nearly over. Another one is used only as a test bed for KTMs. It comes up in the 2 K Synertek FORTH ROM, with a .J Ø, and a simple one line FORTH definition "checks" out the KTMs before they are shipped. This gets very little useage.

A third system supports both FODS and FDC-1 (modified, and in RAM) DOSes, and is used only for making distribution copies of Wharrie's FDC-1 FORTH (which ALL FDC-1 owners SHOULD own, it's great!). The CODOS/Visible Memory SYM is used only to demonstrate the high resolution graphics, and Jack Brown's CODOS FORTH to the occasional visitor. It is turned on so seldom that the NiCad backup batteries on Jeff Lavin's hardware real time clock never really get a full charge. Its major use in the near future will be to download its graphics images to the COM-64.

The CODOS/VM SYM is truly impressive in both demonstrations, because the disk drives can transfer 8 K in either direction in a matter of a second or so, making animated graphics sequences easily possible. Also, when using the standard virtual memory management built into FORTH, only the clicking sounds from the 8-inch drives give away the secret that the screens being requested were not already resident.

Our own "personal" SYM is used for an occasional demonstration of the MTU Advanced Music Software Package (truly astounding), and is being used to prepare this final issue of SYM-PHYSIS only because we have not yet become truly proficient in the uses of any one of the half-dozen or so word processors we now have on the COM-64 (we also have two, including Quick Brown Fox, for the VIC=20).

There are modest amounts of test gear, hand tools, miscellaneous spare parts and components, all useful on other systems. But there are literally hundreds of "used" cassettes and diskettes neatly filed away, some of which we are beginning to "recycle" for use with the 1541 drives, since we see very little future use for the data they now hold.

What's to become of all of this stuff? That we'll leave to the future to decide. We had thought of giving the stuff to students, but it has been our sad experience that such "gifts" were not always useful to the recipients. Unless they themselves had put up some of their own hard earned cash to get started, there was too little motivation to continue. In the old days, several students started with the SYM-1, and as they showed that they were finding the time to learn how to use what they already had, they got "good deals" on RAEs, BASes, KTMs, etc. Today they are much smarter to start out with the VIC=20, the COM-64, the Timex-Sinclair, etc.

And that brings us up to our known plans for the foreseeable near future: to learn as much as we can about the inner workings of the VIC=20 and the COM-64, so that we can help students and others to get the most out of their systems. To that effect we've been putting as much time on these systems (or even more, since they are so much more versatile, and there is so much more to learn) as we once did on the

SYMs! We did build up a valuable skill during the process. We can now switch from one system or DOS to another, and our mind automatically shifts to the proper memory bank which contains the "smarts" for that system or DOS; we no longer become schizophrenic at each shift!

OPEN LETTER TO THE SYM COMMUNITY FROM JEFF LAVIN

AND

ANNOUNCEMENT OF SYMDOS2 BY KIN-PING KWOK

26 January 1984

ALTERNATIVE ENERGY PRODUCTS
P.O. Box 329
6000 Running Springs Road
Ukiah, California 95482
707:462-9244

Members of the SYM community:

In this, the last issue of SYM-PHYSIS, we would like to thank all the people who have made the SYM USER'S GROUP the spawning ground for such a preponderance of ideas and energy. We especially want to thank those of you who have purchased products from us. And we proudly announce our plans for the future.

First of all, don't worry, we will continue to offer our products for as long as current supplies hold out. We will also continue to repair SYMs, FDCs, etc. (NO KTMs) Please direct all future orders directly to AEP at the above address.

Secondly, we are very happy to announce a new DOS written for the FDC by Kin-ping Kwok. Those of you familiar with Kwok's previous work need no further assurance regarding the quality of his programs. I would like to say, though, that he has outdone himself. We have excerpted the introduction and list of commands from the SYMDOS2 manual (they are reproduced following this article). In addition to the features mentioned here, the DOS defaults to a 32K system. Also, there is no longer the IRQ bug to worry about when doing interrupt driven programming, and a very extensive directory search routine is included. Those of you familiar with FLEX (trademark of TSC) will recognize the three character filename extensions; the protocol is to name your file according to type e.g. .RAE .COM .OBJ .TXT etc.

A very useful and important feature of SYMDOS2 is the creation of a "cold-start sector". Upon cold start, the DOS will read the first sector of track zero into memory and test for an "IDMARK". If the mark exists, the DOS will transfer control to a user program. Can anyone think of a use for this?!!!

SYMDOS2 will be available by the time you read this and will consist of: a 5.25" floppy disc, a 2732 EPROM to replace the one on the FDC and a documentation manual. Although SYMDOS2 will support 8" drives, we will only be supporting 5.25" drives for two reasons: One, 5.25" drives are the de facto standard for personel computers; two, only a few people using the FDC are known to use 8" drives AND we aren't one of them! The price is $100 U.S. (shipping included). Please specify if you desire the disc I/O located anywhere besides $F000 and $F100; that is where they will be if you do not specify elsewhere.

Those of you who do not yet own FDCs, and desire to, may still have an opportunity to purchase one. As of this writing we have 17 boards left. These may be purchased as assembled and tested units, as kits, or as bare boards. Call or write for information.

The preceeding information about SYMDOS2 leads us into our last bit of news. We are going to start a special interest group/newsletter for the FDC/SYMDOS2. We will be publishing a newsletter, probably on a quarterly basis, devoted to programs written for the SYM/FDC combination. There will in all likelihood also be included things of general interest from time to time, but the starting intention is as stated. We hope to include hardware and software improvements and tutorials, shopping guide (maybe to include advertisements) and, of course, programs. We will also distribute SYMDOS2 compatible programs on a royalty basis for any interested parties. The newsletter will cost $15 in the United States and Canada, $18.50 elsewhere per 4 issue volume. Make checks payable in U.S. funds to Alternative Energy Products, P.O. Box 329, Ukiah, CA 95482. To those of you who implored us to continue the SYM USER'S GROUP, we are sorry, but this is the best we can do. We do not have the time or energy to devote that Lux did. Most of our articles will be written by readers, and programs will not be edited as much - only checked to insure they at least run and seem to do what the author says they will do. We will have a smaller, and it is hoped closer, user's group than SUG; hence the higher per member cost. We are doing this in the hopes that the SYM community will continue.

After the SYM-PHYSIS has gone to press, we will be mailing a letter to all FDC owners. To the rest of you, this is probably our last contact UNLESS YOU WRITE (or phone)! I have enjoyed these past 2 1/2 years immensely.

Peace,

/s/ Jeff Lavin

INTRODUCTION to SYMDOS2

The FDC-1 is a disk controller for the SYM with a 4K DOS on board. However, there are bugs in the DOS. The undeletable file format is very inconvenient. The user has to reformat a diskette very often. The design of the DOS also limits its expansion. For the above reasons, I have written a new DOS for the FDC-1 to replace the original one.

SYMDOS2 directly replaces the original DOS. It can operate with either a 5" or 8" dual drive system. See the FDC-1 manual for the configuration required. It can also operate up to four single sided drives with a little hardware modification. When using the on-board keyboard, you have to change the vectors yourself as in the origional DOS.

SYMDOS2 can operate in as small as a 1K system. The default is for a 32K system. The diskette format defaults to 5" single density 128 bytes/sector or 5" double density 256 bytes/sector. See ALTERATION for change of defaults. Diskettes formatted by the original DOS may be used by SYMDOS2. However, the directory format is not compatible.

SYMDOS2 has been carefully checked to eliminate bugs. If you find any bugs, please drop me a note.

Finally I hope you like SYMDOS2.

Kin-ping Kwok

SUMMARY of COMMANDS

SUPERMON	BAS-1	RAE-1
FUNCTION: Link to SYMDOS2		
.G 9006	X=USR(&"9000",0)	>RU $9003
FUNCTION: Save to disc		
.S3 filename,u,sa,ea	SAVE u:"filename"	>EN filename u
FUNCTION: Load from disc		
.L3 filename,u	LOAD u:"filename"	>LO filename u
FUNCTION: Load and relocate or append		
.L3 filename,u,sa	LOAD u:A,"filename"	>LO filename u A
FUNCTION: Delete files		
.S4 filename,u		>DC KILL filename u
FUNCTION: Rename file		
.S0 newname,oldname,u		
FUNCTION: List directory		
.L7 u		>DC DIR u
or .L7 filename,u	or	>DC DIR filename u
FUNCTION: Continue to disc		>nnnn .CT filename u

This is an example of the two types of directory listing:

```
FIO3    RAE :FIO3    XRF :DDI2    RAE :DDI2    XRF :FORMAT  RAE :FORMAT  XRF
XRF11A  RAE :XRF11A  DOC :
```

or you can have it this way:

```
FIO3    RAE 0200-4F17 :FIO3    XRF 0200-3AC6 :DDI2    RAE 0200-1E4C :
DDI2    XRF 0200-1A0F :FORMAT  RAE 0200-12F0 :FORMAT  XRF 0200-104E :
XRF11A  RAE 0200-34B9 :XRF11A  DOC 0200-48D5 :
```

MORE ON FDC-1

Reprinted below is a letter from Alan Foster, whom we very much enjoyed meeting during our visit to Australia. He is offering some enhancements for FDC-1 users. In a handwritten postcript he asked two questions, which we will answer here:

First, he asked our opinion regarding a fair price for the package; we feel that $25.00 U.S., postage prepaid anywhere, should cover his handling and shipping costs, and give him a little extra to pay for more equipments. So, write him directly, if interested.

Second, he asked if there would be any copyright problems, in excerpting so heavily from the object code in the FDC-1 EPROM. The answer is not at all, for the following reason: The SYM Users' Group, was given, in writing, all software, firmware, and hardware rights to the FDC-1. We hereby, officially, surrender all software and firmware rights to the FDC-1 into the public domain.

28 GAVIN PLACE,
KINGS LANGLEY, NSW,
AUSTRALIA, 2147

12 NOVEMBER 1983

Dear Lux,

Received your letter re FDC-1 and am enclosing details of my version. Having disassembled and reassembled the FDC-1 firmware I believe I now have a completely bug free version as well as a number of enhancements. Some of the changes are listed below.

Fixes for all firmware bugs mentioned in SYM-PHYSIS have been incorporated. This includes a fix for the "File Save Bug" mentioned in Issue #15 (no solution available at that time).

I have also included a fix for the fact that Supermon's execute command will not work in conjunction with FDC commands. This also allows XRAY's execute command to work with FDC. The execute commands can now be used for copying disks.

This version also includes a power on reset routine which, among other things, initialises the DOS.

On initialising FDC, an expandable table of vectors to FDC routines is moved into RAM. This means that no matter what changes I make to the DOS, or even if I am running an experimental version in RAM, it will always look the same to RAE, FORTH, BASIC etc. The alternative would require that these systems be changed every time the DOS is changed.

DOS memory usage is now as follows:

9000-97FF RAM for DOS variables and buffers
9800-98FF SYMDOS DISK DRIVE INTERFACE
F000-F1FF CONTROL PORT & 1791
F800-FFFF SYMDOS INTERFACE, AND POR ROUTINE

SYSRAM is no longer used by the DOS

If a file is saved with a name which already exists on the disk, the user is prompted with the following options:

Choose a new name (to avoid smudging the existing file).
Smudge the old file.
Overwrite the old file (new file must fit in the space available).

As a separate package I have written a RAE interface which corrects several bugs and deficiencies in the original firmware.

I have also written several disk copy and associated routines as one package. These provide the following:

1. List the directory of the disk to be copied and prompt for selections from this list (up to 20). Selected files will then be copied to a second disk.

2. As above, but files are copied to tape in "named format" with a header file which contains the file name.

3. Copy selected files from tape to disk.

4. Read one tape in named format into memory.

5. Re-initialise a previously initialised disk. (Much faster as it only writes a zero to the first byte in the directory.)

The source code for all of the above plus details of hardware changes for the POR routine are available on 5 1/4" diskette at nominal cost, and may be obtained by writing to me at the above address.

Regards,

A.L.Foster.

MISCELLANEA

BORIS GOLDOWSKY, 23 Culver Hill, Southampton, NY 11968, sent us the object code for an "Etch-a-Sketch" (tm) type program using the MTU Visible Memory and a joystick (analog-type, we believe). We have not yet been able to test the program, since our VM is not at the same location as his. Those of you with Visible Memories may wish to contact him directly for a copy.

We are definitely a creature of habit, and find it hard to adapt to new ways of doing things. Although we have three different Assemblers and five different Word Processors for the Commodore 64, we still use RAE-1 and SWP 2.5 on the "good ol' SYM" for most of our serious work.

We have been comparing all of the assemblers available for the COM-64, and MAE (for Macro Assembler Editor, a variant of RAE, for Resident Assembler Editor) is the only one we have found in which the Editor portion and the Assembler portion are co-resident. In all others the Editor prepares the text file which must then be dumped to mass storage. The Assembler must then be loaded, and it must recall the text file from mass storage for assembly, etc. This back-and-forth switching is inconvenient, to say the least.

MAE also comes with an improved SWP, and an extended Machine Language Monitor (with even more useful commands than SYM's SUPERMON). The ATUG (ASM/TED Users' Group - MAE also goes by the name of ASseMbler/TExt eDitor) provides a (public domain) disassembler into MAE format of the same high level as Dessainte's Disassembler into RAE. It even has the additional convenience feature that it can be advised NOT to attempt to disassemble certain ranges which the simple disassembler built into the ML monitor has "advised" you contain text, tables and/or vectors; it treats these ranges as being composed of easily edited ".BY" pseudo-opcodes.

This means that when we really get going seriously on the COM-64 we'll not have to break too many old habits! And, too, we hope to be able to swap RAE and MAE files between the SYM and the COM-64 when we get our 1541 Drive interfaced to the SYM.

AN FDC-1 BASIC PATCH

BILL CRAMER, 5609 N. Colony Blvd., The Colony, TX 75056, sent along an FDC-1 diskette with the RAE source code for a BASIC DATA SAVE/LOAD routine which permits data files to be passed between BASIC programs, and permits BASIC programs to access multiple files.

The ability to access multiple files is a particularly valuable feature, since otherwise the data files would be size-limited by RAM availability.

It is based on the cassette versions published in previous issues, but is fully linked to the FDC-1 system. We suggest that you contact Bill directly for a copy.

An alternative approach to saving data files which we have been investigating lately is the concept of "sequential" files (and the related concept of "relative" files), as implemented in the various Commodore systems.

In this approach, the data is (are?) dumped to disk as "text" files, and read back in the inverse fashion. The disk system is, in effect, treated as an alternate ASCII terminal interfaced through an IEEE (or serial equivalent) bus. The Commodore disk drives are "intelligent", handling their own buffering and file management. The SYM-1 would have to handle these two tasks, either for FDC-1 or FODS, but this capability is already built into CODOS.

CHEAP RAM FOR SYM AND HAIR-LINE CRACKS

RALPH TEICHEL, P.O. Box 426, Elsternwick 3185, Melbourne, Victoria, Australia, had problems with RAE not storing data past $2000. We advised him that the fault was probably not with RAE, but with his RAM (others have reported similar problems with both RAE and BAS, "fixed" by getting their RAM to work correctly). Sure enough, he found hair-line cracks on his RAM board. One seldom suspects such faults, but we have run into them ourselves.

How to find them? First isolate the problem to the particular board, based on the "behaviour" of the error, and then good luck! This is one of the reasons we recommend "flexing" the boards. This will either close the crack, or break it wide open. In the first case, great!; in the second case you have replaced an intermittent failure with a steady-state one, which is much easier to find.

The main reason for publishing this brief note, is Ralph's suggestion that VIC=20 RAM expansion boards work well with the SYM also. He got his free from a friend(?) who had used them briefly, and then gave them away (Ralph now understands why!).

Ralph calls his SYM "F.R.E.D.", for reasons "not too polite". Since he was too polite to tell us, can anyone else explain the acronym?

AN INTERESTING OFFER AND AN OBJECT CODE TRANSFER PROGRAM

As you know, object code (and BASIC programs as stored in tokenized form) cannot be transmitted directly, byte for byte, since ASCII format handles only seven of the eight bits per byte. Instead, each byte must be broken into two nibbles, and the value of each nibble, corresponding to hex digits 0 through F, is sent as the corresponding ASCII character. Some protocol is required concerning message length, load address, check sums, etc. One such protocol appears in Appendix D of the SYM-1 Reference Manual, as the Paper Tape Format.

An alternate protocol is the Intel Hex Format, mentioned in the letter reproduced below. The letter is published for general interest, and for the interesting offer. A program for the SYM-1 to receive Intel Hex Format follows. We have not yet had the time to try it, although we have a Modem Program for the COM-64 which both sends and receives this format. Our lag in testing this program is that we have never installed a modem "permanently" on any of our SYMs; besides we will be transfering object code from SYM-1 to COM-64 (but not the other way) via 1541 diskettes.

10-29-83

Dear Lux:

I enjoyed our conversation on the telephone regarding hex file transfer from an assembler using the paper tape read function on the SYM-1. I never did get that mode to work, but wound up writing a ML program for the sym to accomplish this. Since I was writing the program myself, I chose to use the more standard Intel Hex Format instead of the KIM format as we discussed. I enclose the source code for the program as an attachment to this letter. I will be glad to supply the program in SYM tape form to anyone sending a tape and a SASE large enough to return the tape and with sufficient postage. The program is written in RAE-1 format. Use of this program allows one to assemble 6502 source code using the AVOCET 6502 assembler on either an IBM-PC or on a CPM based computer and then to transfer the resulting HEX file to the SYM-1. This makes a very nice development system.

The main reason that I'm writing, however, is to tell you about some exciting products I've been developing for MWM Electronics. MWM Electronics is and Atlanta based firm specializing in electronic circuit development for third party manufacturing and sales (usually onder brand names). They started out several years ago manufacturing accessories for satellite television receivers and I have acted as their chief engineer since 1981.

About two years ago we decided to develop a general purpose controller for a satellite receiving station, allowing IR remote control of the entire system (antenna, polarization, channel, etc) from one's easy chair in any room in the house. To accomplish this, we used a Motorola 6802 microprocessor, primarily because we had a large inventory of these chips at the time. Since that time we have designed and built an number of types of controllers for microwave receiving systems and to control test equipment, etc. Each time, we have had to design a new printed circuit board from scratch, requiring several iterations for debugging and modifications.

Another problem we had was that of assembling M6800 source code. For one thing, I was much more familiar with the 6502, having been a long time SYM-1 and Commodore Pet owner (8032, 4032, 4040, 4022, etc). For another thing, I had no computer which employed a M6800 microprocessor as its CPU. At first we hand assembled (ugh!). Later, we wrote a crude assembler in basic (it took 12 hours to assemble 4 K of code). All things considered, I became very frusterated with the M6800 and wanted to change processors, but the momentum was directed against it.

One day, I decided to develop a general purpose mother board, with an eight connector bus which was oriented toward control applications, and with almost nothing else but the processor, bus drivers, a boot ROM and zero page RAM on the mother board, thereby allowing maximum flexibility of configuration. In laying out the board, one of my engineers mentioned that by offsetting

another 40 pin socket slightly, he could arrange the PC board to accomidate either a M6802 or a 6502.....WOW!!! Thus was born the SYSTEM-6000.

By way of brief description, the MWM Electronics SYSTEM-6000 mother board consists of the following:

Microprocessor (6502 or 6802)
Crystal
One EPROM (2716 or 2532) at top of memory
One (optional) 6116 2K RAM chip located at $0000
One 6522 VIA located at $A000
A power supply with +5 Volts at 2 Amps
+15 Volts at 0.5 Amps
-15 Volts at 0.5 Amps
+5 Volt back up battery
A 16 key keypad (optional) connected to PA0-PA7 of the VIA
An eight digit HEX display driven by PB0-PB7 of the VIA
An eight slot BUS with using 0.1 in spaced dual 22 pin connectors with the following pinputs:

1	BA0	A	Ground
2	BA1	B	NOT D400-D7FF
3	BA2	C	NOT D000-D3FF
4	BA3	D	NOT D800-DBFF
5	BA4	E	NOT C400-C7FF
6	BA5	F	NOT C800-CBFF
7	BA6	H	NOT CC00-CFFF
8	BA7	J	NOT RESET
9	BA8	K	PHASE 2 CLOCK
10	BA9	L	NOT IRQ
11	BA10	M	BD7
12	BA11	N	BD6
13	BA12	P	BD5
14	BA13	R	BD4
15	BA14	S	BD3
16	BA15	T	BD2
17	NOT NMI	U	BD1
18	READY	V	BD0
19	R/W	W	-15 VOLTS
20	+8 VOLTS UNREG	X	+15 VOLTS
21	+5 VOLTS STBY	Y	+5 VOLTS REG
22	GROUND	Z	GROUND

In addition to the SYSTEM-6000 Mother Board, we have developed a number of plug-in cards for the bus. The cards which are either finished or in various stages of development are as follows:

MEMORY BOARD (holds four either 2716's or 6116's)
6522 BOARD
6532 BOARD
8 BIT D/A CONVERTER BOARD
TMS-9918A SPRITE VIDEO BOARD
RESOLVER/SYNCHRO INTERFACE (ANGULAR POSITION)
IR SENSOR INTERFACE
RELAY BOARD (8 DIP RELAYS OR OPEN COLLECTORS)
STEPPER MOTOR CONTROLLER BOARD
RS232 INTERFACE
IEEE 488 INTERFACE
WIRE WRAP PLUG-IN
LED DISPLAY BOARD

Also, there will be available a Mother Board with a wire wrap section replacing the processor and its associated electronics, but including the power supply and the eight slot bus. This would greatly simplify those wanting to build a unique computer. I have used one of these (Called the BB-6000) to provide power and a bus for my SYM-1, giving me a real RS-232 port, more

memory, etc available by using the plug-in's. I am in the process of interfacing one to my Commodore 8032 for the same reasons. I hope to develop interface cards and cables for the SYM-1, Commodore 8032/4032, the Commodore 64, and perhaps the VIC-20.

We have written a small monitor for the SYSTEM-6000, called MONDEC19. It incorporates many control oriented functions, however it is not as powerful as the SYM Monitor. We hope to enhance it in the future (we will be looking for help on that), but we also plan on writing a patch to allow the sym monitor to be used, providing we are successful in our endeavors to acquire rights to do that (we are in the process of negotiating for the purchase of SYNNERTEK SYSTEMS). In any case, I think an individual could plug his/her copy of the sym monitor into the SYSTEM-6000 and modity it to have a super sym system.

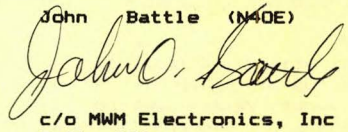
One last comment. We are interested in finding some competant SYM-1 owners who would like to be BETA test sites for the SYSTEM-6000. They would receive a SYSTEM-6000 in consideration for certain reporting responsibilities to MWM Electronics. Interested parties should write to:

SYSTEM-6000
MWM ELECTRONICS, INC
2555 CUMBERLAND PARKWAY
SUITE 280
ATLANTA, GEORGIA 30339

Even if you aren't interested in becomming a BETA sight, write us and we'll put you on our mailing list for future products and/or a possible newsletter.

I'm sorry to hear of SYMPHYSIS ending, but I guess all GOOD things must come to an end sometime, and SYMPHYSIS certainly has been a good thing. Goodby and good luck with your future endeavors. 73.

John Battle (N4OE)



c/o MWM Electronics, Inc
Suite 280
2555 Cumberland Pkwy
Atlanta, GA 30339

ASSEMBLE LIST

```
                        0010          .OS
0200- 0D 0A 46  0020 M1       .BY $0D $0A 'FORMAT ERROR' $0D $0A
0203- 4F 52 4D
0206- 41 54 20
0209- 45 52 52
020C- 4F 52 0D
020F- 0A
0210- 0D 0A 43  0030 M2       .BY $0D $0A 'CHECKSUM ERROR' $0D $0A
0213- 48 45 43
0216- 4B 53 55
0219- 4D 20 45
021C- 52 52 4F
021F- 52 0D 0A
                        0040 OUTCHR   .DE $8A47
                        0050 TECHO    .DE $A653
                        0060 CSUM     .DE $00F2
                        0070 PGMADH   .DE $00F3
                        0080 PGMADL   .DE $00F4
```

```
                0090 ADRL       .DE $00F0
                0100 ADRH       .DE $00F1
                0110 INCHR      .DE $8A1B
                0120 INBYTE     .DE $81D9
                0130 BEEP       .DE $8972
0222- 20 1B 8A  0140 DLOAD      JSR INCHR
0225- C9 3A     0150            CMP #$3A      ; CHECK FOR COLON
0227- D0 F9     0160            BNE DLOAD
0229- A9 00     0170            LDA #$00
022B- 8D F2 00  0180            STA CSUM
022E- D0 F2     0190            BNE DLOAD
0230- 20 D9 81  0200            JSR INBYTE    ; INPUT NUMBER OF BYTES
0233- AA        0210            TAX
0234- F0 55     0220            BEQ FINISH
0236- 18        0230            CLC
0237- 6D F2 00  0240            ADC CSUM
023A- 8D F2 00  0250            STA CSUM
023D- 20 D9 81  0260            JSR INBYTE    ; LOAD HIGH ADDRESS BYTE
0240- 8D F1 00  0270            STA ADRH
0243- 18        0280            CLC
0244- 6D F2 00  0290            ADC CSUM
0247- 8D F2 00  0300            STA CSUM
024A- 20 D9 81  0310            JSR INBYTE    ; LOAD LOW ADDRESS BYTE
024D- 8D F0 00  0320            STA ADRL
0250- 18        0330            CLC
0251- 6D F2 00  0340            ADC CSUM
0254- 8D F2 00  0350            STA CSUM
0257- 20 D9 81  0360            JSR INBYTE    ; LOAD UNUSED BYTE
025A- D0 3C     0370            BNE MSG1      ; CHECK FOR $00
025C- A0 00     0380 START      LDY #$00
025E- 20 D9 81  0390 SAVE       JSR INBYTE
0261- 91 F0     0400            STA (ADRL),Y
0263- 08        0410            PHP
0264- 18        0420            CLC
0265- 6D F2 00  0430            ADC CSUM
0268- 8D F2 00  0440            STA CSUM
026B- 28        0450            PLP
026C- C8        0460            INY
026D- CA        0470            DEX
026E- D0 EE     0480            BNE SAVE
0270- AD F2 00  0490            LDA CSUM      ; TWO'T COMPLEMENT
0273- 49 FF     0500            EOR #$FF
0275- 18        0510            CLC
0276- 69 01     0520            ADC #$01
0278- 8D F2 00  0530            STA CSUM
027B- 20 D9 81  0540 CHECK      JSR INBYTE    ; LOAD CHECKSUM (CSUM)
027E- CD F2 00  0550            CMP CSUM
0281- D0 23     0560            BNE MSG2
0283- A9 0A     0570            LDA #$0A
0285- 20 47 8A  0580            JSR OUTCHR
0288- 4C 22 02  0590            JMP DLOAD
028B- 20 D9 81  0600 FINISH     JSR INBYTE    ; GET HIGH BYTE OF
028E- 8D F3 00  0610                            PGM START ADR
0291- 20 D9 81  0620            JSR INBYTE    ; GET LOW BYTE OF
0294- 8D F4 00  0630                            PGM START ADR
0297- 00        0640            BRK
0298- A0 00     0650 MSG1       LDY #$00
029A- B9 00 02  0660 L1         LDA M1,Y
029D- 20 47 8A  0670            JSR OUTCHR
02A0- C8        0680            INY
02A1- C0 10     0690            CPY #$10
02A3- D0 F5     0700            BNE L1
02A5- 00        0710            BRK
02A6- A0 00     0720 MSG2       LDY #$00
02A8- B9 10 02  0730 L2         LDA M2,Y
```



```
02AB- 20 47 8A  0740            JSR OUTCHR
02AE- C8        0750            INY
02AF- C0 12     0760            CPY #$12
02B1- D0 F5     0770            BNE L2
02B3- 00        0780            BRK
                0790            .EN
```

USE OF NULL STRING IN RAE

Often we have wished to enter a ".BY" ASCII string with leading spaces. This cannot be entered as .BY ' STRING' (in this example we assume that five leading spaces are desired), since RAE will accept at most one leading space in any ASCII string immediately following the .BY pseudo-op, as in line 20 below, because of the way in which FOrmat is implemented.

We have been getting around this by entering the first space as $20 or ' ', and entering one less leading space than desired in the actual string, as in lines 30 and 40 below.

By rereading (for at least the tenth time!) the manual which came with our MAE for the PET systems (including the COM-64), we discovered that we could use a null string, instead, as in line 50 below.

Incidentally, when Carl Moser lent us a copy of the RAE-1 source code to study, he sent us an annotated copy which he had marked up in red (we are giving up here a golden opportunity to use the word "rubric" in a sentence!) to indicate where he would "upgrade" RAE-1 into a RAE-2 version if Synertek Systems ever so requested. The COM-64 version of MAE incorporate these modifications.

Again, incidentally, for SYMmers going to a COM-64 as their "second" system, MAE actually supports dual 1541 drives, since these are addressable as D8 and D9, instead of by the 0: and 1: used in so much other PET derived software. This is because MAE was actually designed to support TWO dual drive systems on the PETs!

```
>FORMAT SET

>ASSEMBLE LIST
                0010            .LS
0200- 20 53 54  0020            .BY ' STRING'    ;FIVE SPACES ENTERED HERE
0203- 52 49 4E
0206- 47
0207- 20 20 20  0030            .BY $20 '    STRING' ;FOUR SPACES ENTERED HERE
020A- 20 20 53
020D- 54 52 49
0210- 4E 47
0212- 20 20 20  0040            .BY ' ' '    STRING' ;FOUR SPACES ENTERED HERE
0215- 20 20 53
0218- 54 52 49
021B- 4E 47
021D- 20 20 20  0050            .BY '' '     STRING' ;FIVE SPACES ENTERED HERE
0220- 20 20 53
0223- 54 52 49
0226- 4E 47
//0000,0228,0228
```

WARNING NOTICE

If you have entered more than one leading space in the string, your assembly will be different with FO C than with FO S. Here, for your information, is an assembly listing of the very same program as above, but with FO C as the format option. Note that the leading spaces in the string have been assembled into the object code.

```
>FORMAT CLEAR

>ASSEMBLE LIST
                    0010  .LS
0200- 20 20 20      0020  .BY '      STRING'     ;FIVE SPACES ENTERED HERE
0203- 20 20 53
0206- 54 52 49
0209- 4E 47
020B- 20 20 20      0030  .BY $20 '     STRING' ;FOUR SPACES ENTERED HERE
020E- 20 20 53
0211- 54 52 49
0214- 4E 47
0216- 20 20 20      0040  .BY ' ' '     STRING' ;FOUR SPACES ENTERED HERE
0219- 20 20 53
021C- 54 52 49
021F- 4E 47
0221- 20 20 20      0050  .BY '' '      STRING' ;FIVE SPACES ENTERED HERE
0224- 20 20 53
0227- 54 52 49
022A- 4E 47
//0000,022C,022C
```

MORE ON THE 65C02

We finally got our first 65C02 today. It is an NCR65C02A; we specifically indicate the source, NCR, because none of the MOS Technology spec sheets (marked preliminary) we have on hand seem to indicate that their version has the added 27 new instructions and the additional addressing modes of the NCR version. Does anyone have this information on the MOS version?

The 65C02 may be directly substituted for the 6502 for lower power consumption, but to take full advantage of its extended instruction set an upgraded RAE-1 will be required. Here are some thoughts on the subject from Phil Kohl:

```
        0010  ;  EXTRACTS BY LUX FROM A PROGRAM BY PHIL KOHL
        0020  ;  File name: RAE2
        0030
        0040  ;  Creation date:  August 14, 1983
        0050  ;  External references deleted:  September 3, 1983
        0060  ;  65SC02 Modifications:  October 15, 1983
        0070
        0080
        0090  BB6CA       LDA SI.OP,X  ;Single byte op codes
        0100              BNE CHECK.CHAR
        0110              RTS
        0120
        0130  BB6D0       LDA RAE.COM,X :RAE commands
        0140              BNE CHECK.CHAR
        0150              RTS
        0160
        0170
        0180              .BY ') ' ;Two byte indirect addressing
        0190              .SI IND2
        0200              .BY $00
        0210              .BY 'A ' ;Accumulator addressing
        0220              .SI IMP
        0230              .BY $00
        0240              .BY ') ' ;Three byte indirect addressing
        0250              .SI IND3
        0260              .BY $00
        0270
        0280  SI.OP       .BY 'TAX'
        0290              .BY $AA
```

(listing continued to page 17-21, text continued to page 17-23)

A 16-BIT 6502!
(AT LONG LAST!)

The extract below, from the Electronic Engineering Times, was sent to us by William Luitje. The clipping on the right, from Sol Libes' BITS & BYTES column in the March 1984 issue of Computers & Electronics, was called to our attention by Dave Wagner.

16-Bit Version of 6502 Announced

Sol Libes in B & P

▶ When Commodore scrapped the 16-bit microprocessor it had been developing for several years in favor of the Zilog Z8000, it left the market wide open for an upward-compatible 16-bit version of the 6502. Sure enough a company has seized the opportunity. Western Design Center Inc., Mesa, AZ, has announced a 16-bit microprocessor that runs 6502 software in an emulation mode without revision. The CMOS chip can address 16M bytes of memory compared to the 6502's 64K. It has an 8-bit external bus and internal 16-bit bus. The most amazing feature is that it is pin-compatible with the 6502. You just remove the 6502 from its socket and replace it with the W65SC816. Then set the E-bit in the status register and it performs exactly like the 6502. If the bit is off, the device becomes a 16-bit device.

WDC To Market 16-Bit Version Of 6502 μP

By Stan Baker

MESA, Ariz. — The biggest-selling 8-bit microprocessor, the 6502, will soon have CMOS 16-bit family members with 8-bit and 16-bit databus versions.

These parts and several major CMOS peripherals will come from the Western Design Center Inc. (WDC) which developed and owns rights to the CMOS version of the 6502 and its CMOS peripherals.

However, WDC will not license the new 16-bit units for a one-time fee as it has the 8-bit parts. Rather, the design firm has ambitious plans to supply chips for the first time and to provide high-level custom chip design equipment, software and services.

The first chips of the 16-bit version of the 6502 microprocessor are due for testing at WDC by the end of January. The first of these CMOS chips, the 65C816, will most likely come from Santa Clara, Calif., where they are being fabricated by American Microdevices Inc. But GTE Microcircuits is also processing the new design in Tempe, Ariz., having bought a license to market the chip. GTE wants to be its first volume producer.

WDC is an IC design operation that has specialized in developing 6502 CMOS parts and licensing them for one-time fees. The n-channel original version has the highest production volume of any 8-bit processor. This is because it is used in many highly successful products, such as Apple computers, Atari games and computers, and Commodore computers. The CMOS 6502 and its CMOS peripherals are now licensed by WDC to Synertek, NCR, Rockwell, Plessey, and Marconi.

[EDITOR'S NOTE: Observe that Commodore is NOT a licensee for the "enhanced" 65C02!]

The n-channel 6502 was designed in 1975 at MOS Technology, Valley Forge, Pa., by a group led by William Mensch Jr. Commodore Business Machines bought that firm in 1976 and Mensch left in 1977 for his former home of Phoenix, Ariz., where he had been on the Motorola team that designed the 6800 microprocessor. After spending a short time at Integrated Circuit Engineering Corp., he founded WDC in mid-1978, and is now its president.

Commodore still owns the rights to the n-channel 6502, which is made for the merchant market by Synertek and Rockwell. Commodore now makes it only for in-house use.

Mensch explained that his 20 employees at WDC have designed about 20 chips in the past 18 months and are now embarking on a program to provide 16-bit processors and major new peripherals and to put its design systems and services at customer premises.

In a major change of strategy, the firm will become a chip supplier, rather than a design house, and will not license the designs of its new 16-bit chips. "Western Design Center is going to ship the product, with GTE as second source," Mensch said.

Larger Market Share Sought

GTE Microcircuits is the only licensee of the 65C816 and its 8-bit databus version, the 65C802. Mensch explained, "We are interested in other licensees, but for increasing market share, not just to license people."

He is hoping and expecting to license large systems companies, such as GE, Philips, and IBM, to be their own in-house alternate sources. Large systems companies could supply themselves and be licensed to use the WDC designs as "super cells" in large chips of their own design, Mensch said.

New Processors

The new 16-bit processors will be in 40-pin packages, with pinouts similar to the 6502. The 8-bit version, 65C802, will be a direct pin-for-pin replacement for the 65C02. Also, the 65C802 will use the same chip as the 65C816. Only the pinouts will be different.

Mensch noted that the new 8-bit unit will emulate the current 6502 without a noticeable change in performance. But it can implement many new capabilities in the old socket, if the customer wishes.

The new processors will have 95 instructions, compared to 56 for the 6502, and 24 addressing modes, while the 6502 has 13. The 24 address bits of the new units will be multiplexed.

In one phase, all 24 address bits will be presented, with eight of them on data lines. In the next phase, the data lines will pass data only.

Shrinking CMOS Process

A critical part of this program is shrinking from the 3-micron CMOS process currently used for the new processors, to a 1.5-micron process now in development at GTE Microcircuits. The current process will yield 8-MHz to 10-MHz clock-rate parts, with a minimum of 4 MHz, Mensch said. The objective for the 1.5-micron process is not only to reduce the chip size, but to produce a full family of 20-MHz parts.

Mensch is now studying the feasibility of WDC having its own fab facility, based on the 1.5-micron CMOS process. However, he has not made a commitment to a fab facility yet.

Electronic Engineering Times -

Monday, January 9, 1984

LISTING (continued from page 17-19)

```
0300            .BY 'TAY'
0310            .BY $A8
0320            .BY 'TSX'
0330            .BY $BA
0340            .BY 'TXA'
0350            .BY $8A
0360            .BY 'TXS'
0370            .BY $9A
0380            .BY 'TYA'
0390            .BY $98
0400            .BY 'CLC'
0410            .BY $18
0420            .BY 'CLD'
0430            .BY $D8
0440            .BY 'CLI'
0450            .BY $58
0460            .BY 'CLV'
0470            .BY $B8
0480            .BY 'SEC'
0490            .BY $38
0500            .BY 'SED'
0510            .BY $F8
0520            .BY 'SEI'
0530            .BY $78
0540            .BY 'NOP'
0550            .BY $EA
0560            .BY 'RTI'
0570            .BY $40
0580            .BY 'RTS'
0590            .BY $60
0600            .BY 'DEX'
0610            .BY $CA
0620            .BY 'DEY'
0630            .BY $88
0640            .BY 'INX'
0650            .BY $E8
0660            .BY 'INY'
0670            .BY $C8
0680            .BY 'PHA'
0690            .BY $48
0700            .BY 'PHP'
0710            .BY $08
0720            .BY 'PLA'
0730            .BY $68
0740            .BY 'PLP'
0750            .BY $28
0760            .BY 'PHY'
0770            .BY $5A
0780            .BY 'PLY'
0790            .BY $7A
0800            .BY 'PHX'
0810            .BY $DA
0820            .BY 'PLX'
0830            .BY $FA
0840            .BY 'BRK'
0850            .BY $00
0860            .BY $00
0870            .BY 'BCC'
0880            .BY $90
0890            .BY 'BCS'
0900            .BY $B0
0910            .BY 'BEQ'
0920            .BY $F0
0930            .BY 'BMI'
0940            .BY $30
0950            .BY 'BNE'
0960            .BY $D0
0970            .BY 'BPL'
0980            .BY $10
0990            .BY 'BVC'
1000            .BY $50
1010            .BY 'BVS'
1020            .BY $70
1030            .BY 'BRA'
1040            .BY $80
1050            .BY $00
1060
1070  MU.OP     .BY 'ADC'
1080            .BY $F9 $DA
1090            .BY $6D $7D $79 $72 $65 $75 $71 $61 $69
1100
1110            .BY 'AND'
1120            .BY $F9 $DA
1130            .BY $2D $3D $39 $32 $25 $35 $31 $21 $29
1140
1150            .BY 'ASL'
1160            .BY $C5 $C1
1170            .BY $0E $1E $06 $16 $0A
1180
1190            .BY 'BIT'
1200            .BY $C5 $C2
1210            .BY $2C $3C $24 $34 $89
1220
1230            .BY 'CMP'
1240            .BY $F9 $DA
1250            .BY $CD $DD $D9 $D2 $C5 $D5 $D1 $C1 $C9
1260
1270            .BY 'CPX'
1280            .BY $83 $82
1290            .BY $EC $E4 $E0
1300
1310            .BY 'CPY'
1320            .BY $83 $82
1330            .BY $CC $C4 $C0
1340
1350            .BY 'DEC'
1360            .BY $C5 $C1
1370            .BY $CE $DE $C6 $D6 $3A
1380
1390            .BY 'EOR'
1400            .BY $F9 $DA
1410            .BY $4D $5D $59 $52 $45 $55 $51 $41 $49
1420
1430            .BY 'INC'
1440            .BY $C5 $C1
1450            .BY $EE $FE $E6 $F6 $1A
1460
1470            .BY 'JMP'
1480            .BY $93 $08
1490            .BY $4C $6C $7C
1500
1510            .BY $00
1520
1530  MU.OP2    .BY 'LDA'
1540            .BY $F9 $DA
1550            .BY $AD $BD $B9 $B2 $A5 $B5 $B1 $A1 $A9
1560
```



```
1570            .BY 'LDX'
1580            .BY $A5 $A2
1590            .BY $AE $BE $A6 $B6 $A2
1600
1610            .BY 'LDY'
1620            .BY $C5 $C2
1630            .BY $AC $BC $A4 $B4 $A0
1640
1650            .BY 'LSR'
1660            .BY $C4 $C1
1670            .BY $4E $5E $46 $56 $4A
1680
1690            .BY 'SR'      ;J from previous byte
1700            .BY $81 $00
1710            .BY $20
1720
1730            .BY 'ORA'
1740            .BY $F9 $DA
1750            .BY $0D $1D $19 $12 $05 $15 $11 $01 $09
1760
1770            .BY 'ROL'
1780            .BY $C5 $C1
1790            .BY $2E $3E $26 $36 $2A
1800
1810            .BY 'SBC'
1820            .BY $F9 $DA
1830            .BY $ED $FD $F9 $F2 $E5 $F5 $F1 $E1 $E9
1840
1850            .BY 'ROR'
1860            .BY $C5 $C1
1870            .BY $6E $7E $66 $76 $6A
1880
1890            .BY 'STA'
1900            .BY $F8 $D8
1910            .BY $8D $9D $99 $92 $85 $95 $91 $81
1920
1930            .BY 'STX'
1940            .BY $83 $A0
1950            .BY $8E $86 $96
1960
1970            .BY 'STY'
1980            .BY $83 $C0
1990            .BY $8C $84 $94
2000
2010            .BY 'STZ'
2020            .BY $C4 $C0
2030            .BY $9C $9E $64 $74
2040
2050            .BY 'TRB'
2060            .BY $82 $80
2070            .BY $1C $14
2080
2090            .BY 'TSB'
2100            .BY $82 $80
2110            .BY $0C $04
2120
2130            .BY $00
2140
2150            .CT RAE3
```

Note that opcodes DEA, $3A, and INA, $1A, are not considered in the above tables.

MORE ON THE 65C02 (continued from page 17-19)

What is the above listing all about? Well, it is a portion of the "source code" for an "unoffical RAE-2", sent us by Phil Kohl. As you can see by the "non-mnemonic" labels in lines 0090 and 00130, Phil has disassembled Carl Moser's RAE-1, analyzed and commented it, and then extended it to include the new op-codes for the 65SC02 (apparently an alternate name for the non-CBM 65C02). For copyright reasons we cannot make available his source code in its entirety, but we publish the portion above to give those among you who wish to do the same a head start, by showing how and where additional op-codes may be added to RAE-1.

In addition to the 65C02/65SC02, a 65C802 with 95 instructions and 24 addressing modes can be used in place of the 6502 (see the clippings on page 17-20), and some of you will, no doubt want to extend RAE to accomodate that and the 16 bit version, the 65C816, as well. We firmly approve of the type of reverse engineering and modification (for personal use and for limited distribution for non profit research purposes) which is exemplified by Phil's work. Note that, wherever practical, SYM-PHYSIS has always published, and the SYM Users' Group has always distributed, fully commented source code to save you the time and trouble of disassembly, and to encourage "customization", and to enhance your system understanding.

For those who wish to know more about the new "16-bitter", we reprint the following paragraphs from the February 1984 Issue of the IEEE Philadelphia Section Newsletter, "Update", which we receive thanks to the courtesy of George Bodenstein:

W65SC802 and W65SC816 Microprocessors. These 16-bit microprocessors are CMOS devices designed to replace the 6500 8-bit microcomputer family. They operated in two modes, 6502 emulation and native. They start-up in the 6502 mode so that they can be used to replace the 6502 in any system without having to change software. The 65802 will fit in the same socket as the 6502 and requires NO HARDWARE changes. The 65816 requires moderate hardware changes to fit in a 6502 system. The 65802 and 65816 operating the native mode will execute programs up to 3.5 times faster than the 6502. In the native mode the processors execute all the orginal 56 NMOS and 10 new CMOS 6502 instructions on 8 and 16 bit data. All registers can either be 8 or 16 bit wide. The processors also execute 30 new instructions which include block moves, coprocessor and system control instructions. They have 11 new addressing modes including long branches, program and stack relative. The 65816 can address up to 16 Megabytes of memory in either a linear or segmented mode. These two products, a floating point coprocessor, and operating system will be the subjects of the March Update meeting.

W65SC02 CMOS Microprocessor. Hardware and software compatible with the NMOS 6502. They have 10 new instructions and two new addressing modes. Low power operation.

W65SC21 Peripheral Interface Adapter. Direct replacement for the NMOS 6521 or 6821 PIAs. Low power operation.

W65SC22 Versatile Interface Adapter. Direct low powered replacement for the 6522 VIA.

For additional information, contact: The Western Design Center, Inc., 2166 East Brown Road, Mesa, AZ 85203, (602) 962-4545

MORE RAE ENHANCEMENTS

John Mattox sent us a three page listing of a program which "allows RAE to produce and capture files via modem". Since the listing was not easily reproducible, we asked John to send us the program on cassette. He sent us, instead, a cassette containing the much longer program listed below. While such a program would normally be too long for SYM-PHYSIS, we make an exception in this case. As you know, the power of RAE is what attracted us to SYM in the first palce, and any enhancements to RAE are well worth disseminating.

We have not been able to fully test the program because it breaks our disk and printer links, but we have tested several sections, including the "menu" portion, and we feel that it is definitely worth studying and extracting ideas from. Note too, his links to SWP!

The listing is published exactly as submitted, with no editing on our part to compress the output lines to the 80 character printer limit. We omitted this customary editing step because the listing would have "spilled over" onto a second line in so many places anyway, because of the way John entered his line feeds directly into the .by statements before the final "'", rather than externally, as a 10 (or $0A) following the final "'". This may "louse up" the listing, but it does save keystrokes!

Contact John directly for cassette copies (his address appears in lines 0020 and 0740). Have fun with this one!

```
>PASS2

                0010 ;A program to extend RAE, written by John Mattox
                0020 ;331 Nova Lane, Menlo Park, CA 94025.
                0030 ;The program provides supporting commands accessed by esc.
                0040 ;Lines sensitive to relocaton are indicated with *>! .
                0050 ;My machine uses tape for storage. To free the zero page di
sc
                0060 ;vectors, one can use one zero page vector for the indirect
                0070 ;addressing done in this program, storing the vectors
                0080 ;in non zero page ram.
                0090 swp        .de $3736 Beginning of SWP. *>!
                0100 techo      .de $a653
                0110 param      .de $a64a
                0120 inbyte     .de $81d9
                0130 outchr     .de $8a47
                0140 intchr     .de $8a58
                0150 LFflg      .de $f4       ;Inhibit LF if not zero
                0160 SPflg      .de $f5       ;Add space before CR if not zero
                0170 chrindex   .de $ec index for text output
                0180 outputdex  .de $f0
                0190 inputdex   .DE $00CA WORKING INDEX
                0200 putdex     .de $f2
                0210 prsdex     .de $f6
                0220 symb       .de $e8
                0230 nuprs      .de $cc
                0240 min        .de $280 Beginning of buffer for input. *>!
                0250 max        .de $2fff Maximum value of buffer for input *>!
                0260 sdbyt      .de $a651
                0270 begin      .de $2c00 Address for beginning of prom. *>!
                0280 ram        .de $2b00 Beginning of phrase storage ram block.
 *>!
                0290            .ba begin
                0300 ;.os
2C00- 20 86 8B  0310 initialize jsr $8b86 access
2C03- A9 30     0320            lda #h,outchr@
2C05- 8D 65 A6  0330            sta $a665
2C08- A9 D6     0340            lda #l,outchr@
```

```
2C0A- 8D 64 A6  0350            sta $a664
2C0D- A9 2E     0360            lda #h,intchr@
2C0F- 8D 62 A6  0370            sta $a662
2C12- A9 0D     0380            lda #l,intchr@
2C14- 8D 61 A6  0390            sta $a661
2C17- A9 00     0400            lda #00
2C19- 85 F4     0410            sta *LFflg
2C1B- 85 F5     0420            sta *SPflg
2C1D- 85 CC     0430            sta *nuprs
2C1F- A9 2D     0440            lda #chr?
2C21- 85 EC     0450            sta *chrindex
2C23- A9 2C     0460            lda #h,chr?
2C25- 85 ED     0470            sta *chrindex+1
2C27- 20 F8 30  0480            jsr outputchr
2C2A- A9 0D     0490            lda #13
2C2C- 60        0500            rts
2C2D- 57 65 6C  0510 chr?       .by 'Welcome to the world of the RAE escape func
tion.
' 13
2C30- 63 6F 6D
2C33- 65 20 74
2C36- 6F 20 74
2C39- 68 65 20
2C3C- 77 6F 72
2C3F- 6C 64 20
2C42- 6F 66 20
2C45- 74 68 65
2C48- 20 52 41
2C4B- 45 20 65
2C4E- 73 63 61
2C51- 70 65 20
2C54- 66 75 6E
2C57- 63 74 69
2C5A- 6F 6E 2E
2C5D- 0A 0D
2C5F- 65 73 63  0520            .by 'esc ?  -  Print menu
' 13 $ff
2C62- 20 3F 20
2C65- 20 2D 20
2C68- 20 50 72
2C6B- 69 6E 74
2C6E- 20 6D 65
2C71- 6E 75 0A
2C74- 0D FF
2C76- A9 84     0530 menu       lda #chr^
2C78- 85 EC     0540            sta *chrindex
2C7A- A9 2C     0550            lda #h,chr^
2C7C- 85 ED     0560            sta *chrindex+1
2C7E- 20 F8 30  0570            jsr outputchr
2C81- A9 0D     0580            lda #13
2C83- 60        0590            rts
2C84- 65 73 63  0600 chr^       .by 'esc c  -  RAE cold start
' 13
2C87- 20 63 20
2C8A- 20 2D 20
2C8D- 20 52 41
2C90- 45 20 63
2C93- 6F 6C 64
2C96- 20 73 74
2C99- 61 72 74
2C9C- 0A 0D
2C9E- 65 73 63  0610            .by 'esc a  -  Adjust output
' 13
2CA1- 20 61 20
2CA4- 20 2D 20
```



```
2CA7- 20 41 64
2CAA- 6A 75 73
2CAD- 74 20 6F
2CB0- 75 74 70
2CB3- 75 74 0A
2CB6- 0D
2CB7- 65 73 63  0620            .by 'esc s  -  Output text with SWP
' 13
2CBA- 20 73 20
2CBD- 20 2D 20
2CC0- 20 4F 75
2CC3- 74 70 75
2CC6- 74 20 74
2CC9- 65 78 74
2CCC- 20 77 69
2CCF- 74 68 20
2CD2- 53 57 50
2CD5- 0A 0D
2CD7- 65 73 63  0630            .by 'esc b  -  Reset baud rate
' 13
2CDA- 20 62 20
2CDD- 20 2D 20
2CE0- 20 52 65
2CE3- 73 65 74
2CE6- 20 62 61
2CE9- 75 64 20
2CEC- 72 61 74
2CEF- 65 0A 0D
2CF2- 65 73 63  0640            .by 'esc i  -  Input into buffer
' 13
2CF5- 20 69 20
2CF8- 20 2D 20
2CFB- 20 49 6E
2CFE- 70 75 74
2D01- 20 69 6E
2D04- 74 6F 20
2D07- 62 75 66
2D0A- 66 65 72
2D0D- 0A 0D
2D0F- 65 73 63  0650            .by 'esc d - Define symbol
' 13
2D12- 20 64 20
2D15- 2D 20 44
2D18- 65 66 69
2D1B- 6E 65 20
2D1E- 73 79 6D
2D21- 62 6F 6C
2D24- 0A 0D
                0660 ;User enter symbol and phrase into ram block.
2D26- 65 73 63  0670            .by 'esc esc # - Produce text for symbol #
' 13
2D29- 20 65 73
2D2C- 63 20 23
2D2F- 20 2D 20
2D32- 50 72 6F
2D35- 64 75 63
2D38- 65 20 74
2D3B- 65 78 74
2D3E- 20 66 6F
2D41- 72 20 73
2D44- 79 6D 62
2D47- 6F 6C 20
2D4A- 23 0A 0D
                0680 ;First searches prom(assembled) block and then ram block.
2D4D- 65 73 63  0690            .by 'esc q - delete a symbol
' 13
```

```
2D50- 20 71 20
2D53- 2D 20 64
2D56- 65 6C 65
2D59- 74 65 20
2D5C- 61 20 73
2D5F- 79 6D 62
2D62- 6F 6C 0A
2D65- 0D
2D66- 65 73 63  0700            .by 'esc p - print symbols and text
' 13
2D69- 20 70 20
2D6C- 2D 20 70
2D6F- 72 69 6E
2D72- 74 20 73
2D75- 79 6D 62
2D78- 6F 6C 73
2D7B- 20 61 6E
2D7E- 64 20 74
2D81- 65 78 74
2D84- 0A 0D
2D86- 65 73 63  0710            .by 'esc e  -  Echo input buffer to RAE
' 13 $ff
2D89- 20 65 20
2D8C- 20 2D 20
2D8F- 20 45 63
2D92- 68 6F 20
2D95- 69 6E 70
2D98- 75 74 20
2D9B- 62 75 66
2D9E- 66 65 72
2DA1- 20 74 6F
2DA4- 20 52 41
2DA7- 45 0A 0D
2DAA- FF
                0720 ;Storage table for prom symbols and text
                0730 ;format is symbol, then phrase, then $ff
2DAB- 71 4A 6F  0740 prom       .by 'qJohn Mattox' $ff 'w331 Nova Lane, Menlo Pa
rk, CA 94025' $ff
2DAE- 68 6E 20
2DB1- 4D 61 74
2DB4- 74 6F 78
2DB7- FF 77 33
2DBA- 33 31 20
2DBD- 4E 6F 76
2DC0- 61 20 4C
2DC3- 61 6E 65
2DC6- 2C 20 4D
2DC9- 65 6E 6C
2DCC- 6F 20 50
2DCF- 61 72 6B
2DD2- 2C 20 43
2DD5- 41 20 39
2DD8- 34 30 32
2DDB- 35 FF
2DDD- 75 2E 6D  0750            .by 'u.m 0 73' $ff 'i.m 5 63' $ff 'o.m 0 73 63 3
' $ff
2DE0- 20 30 20
2DE3- 37 33 FF
2DE6- 69 2E 6D
2DE9- 20 35 20
2DEC- 36 33 FF
2DEF- 6F 2E 6D
2DF2- 20 30 20
2DF5- 37 33 20
2DF8- 36 33 20
```



```
2DFB- 33 FF
                0760 ;formatting statements for SWP
2DFD- 70 2E 6D  0770            .by 'p.m 0 73 1000 3' $ff
2E00- 20 30 20
2E03- 37 33 20
2E06- 31 30 30
2E09- 30 20 33
2E0C- FF
2E0D- 20 58 8A  0780 intchr@    jsr intchr
2E10- 29 7F     0790            and #$7f
2E12- C9 1B     0800            cmp #$1b ?escape
2E14- F0 01     0810            beq escape
2E16- 60        0820            rts
2E17- 20 58 8A  0830 escape     jsr intchr
2E1A- 29 7F     0840            and #$7f
2E1C- C9 63     0850            cmp #'c
2E1E- F0 36     0860            beq alpha
2E20- C9 61     0870            cmp #'a
2E22- F0 35     0880            beq gamma
2E24- C9 73     0890            cmp #'s
2E26- F0 2B     0900            beq sp
2E28- C9 62     0910            cmp #'b
2E2A- F0 36     0920            beq baud
2E2C- C9 69     0930            cmp #'i
2E2E- F0 2C     0940            beq epsilon
2E30- C9 65     0950            cmp #'e
2E32- F0 13     0960            beq lamda
2E34- C9 71     0970            cmp #'q
2E36- F0 12     0980            beq quit
2E38- C9 64     0990            cmp #'d
2E3A- F0 11     1000            beq define
2E3C- C9 1B     1010            cmp #$1b ?escape
2E3E- F0 10     1020            beq dbl.esc
2E40- C9 70     1030            cmp #'p
2E42- F0 1B     1040            beq print
2E44- 4C 76 2C  1050            jmp menu
2E47- 4C 9E 30  1060 lamda      jmp echo
2E4A- 4C A0 31  1070 quit       jmp Quit
2E4D- 4C 36 31  1080 define     jmp Define
2E50- 4C 0D 31  1090 dbl.esc    jmp Dbl.esc
2E53- 4C 36 37  1100 sp         jmp swp
2E56- 4C 9E 2F  1110 alpha      jmp cold
2E59- 4C C9 2E  1120 gamma      jmp alter
2E5C- 4C 63 30  1130 epsilon    jmp input
2E5F- 4C 14 32  1140 print      jmp Print
2E62- A9 8E     1150 baud       lda #chrJ
2E64- 85 EC     1160            sta *chrindex
2E66- A9 2E     1170            lda #h,chrJ
2E68- 85 ED     1180            sta *chrindex+1
2E6A- 20 F8 30  1190            jsr outputchr
2E6D- 20 58 8A  1200            jsr intchr
2E70- C9 B1     1210            cmp #$b1
2E72- D0 05     1220            bne two
2E74- A9 4C     1230            lda #$4c
2E76- 4C 88 2E  1240            jmp setbaud
2E79- C9 B2     1250 two        cmp #$b2
2E7B- D0 05     1260            bne three
2E7D- A9 10     1270            lda #$10
2E7F- 4C 88 2E  1280            jmp setbaud
2E82- C9 B3     1290 three      cmp #$b3
2E84- D0 DC     1300            bne baud
2E86- A9 01     1310            lda #$01
2E88- 8D 51 A6  1320 setbaud    sta sdbyt
2E8B- A9 0D     1330            lda #13
2E8D- 60        1340            rts
```

```
2E8E- 61 6C 74  1350 chr1       .by 'alter baud rate: enter 1 for 300, 2 for 120
0,'
2E91- 65 72 20
2E94- 62 61 75
2E97- 64 20 72
2E9A- 61 74 65
2E9D- 3A 20 65
2EA0- 6E 74 65
2EA3- 72 20 31
2EA6- 20 66 6F
2EA9- 72 20 33
2EAC- 30 30 2C
2EAF- 20 32 20
2EB2- 66 6F 72
2EB5- 20 31 32
2EB8- 30 30 2C
2EBB- 20 33 20  1360            .by ' 3 for 4800
' 13 $ff
2EBE- 66 6F 72
2EC1- 20 34 38
2EC4- 30 30 0A
2EC7- 0D FF
2EC9- A9 DC     1370 alter      lda #chr3
2ECB- 85 EC     1380            sta *chrindex
2ECD- A9 2E     1390            lda #h,chr3
2ECF- 85 ED     1400            sta *chrindex+1
2ED1- 20 F8 30  1410            jsr outputchr
2ED4- A9 80     1420            lda #$80
2ED6- 8D 53 A6  1430            sta techo
2ED9- 4C 78 2F  1440            jmp brn0
2EDC- 0A 54 59  1450 chr3       .by '
TYPE 00 or 01   for inserion of space before CR
' 13
2EDF- 50 45 20
2EE2- 30 30 20
2EE5- 6F 72 20
2EE8- 30 31 20
2EEB- 20 20 66
2EEE- 6F 72 20
2EF1- 69 6E 73
2EF4- 65 72 69
2EF7- 6F 6E 20
2EFA- 6F 66 20
2EFD- 73 70 61
2F00- 63 65 20
2F03- 62 65 66
2F06- 6F 72 65
2F09- 20 43 52
2F0C- 0A 0D
2F0E- 54 59 50  1460            .by 'TYPE 00 or 01  for inhibit line-feeds
' 13
2F11- 45 20 30
2F14- 30 20 6F
2F17- 72 20 30
2F1A- 31 20 20
2F1D- 66 6F 72
2F20- 20 69 6E
2F23- 68 69 62
2F26- 69 74 20
2F29- 6C 69 6E
2F2C- 65 2D 66
2F2F- 65 65 64
2F32- 73 0A 0D
2F35- 45 6E 74  1470            .by 'Enter PADBIT value  (A0 for modem, 01 for s
creen,'
```



```
2F38- 65 72 20
2F3B- 50 41 44
2F3E- 42 49 54
2F41- 20 76 61
2F44- 6C 75 65
2F47- 20 20 28
2F4A- 41 30 20
2F4D- 66 6F 72
2F50- 20 6D 6F
2F53- 64 65 6D
2F56- 2C 20 30
2F59- 31 20 66
2F5C- 6F 72 20
2F5F- 73 63 72
2F62- 65 65 6E
2F65- 2C
2F66- 32 30 20  1480            .by '20 for printer)
' 13 $ff
2F69- 66 6F 72
2F6C- 20 70 72
2F6F- 69 6E 74
2F72- 65 72 29
2F75- 0A 0D FF
2F78- 20 D9 81  1490 brn0       jsr inbyte
2F7B- 8D F5 00  1500            sta SPflg
2F7E- AD 0A 00  1510            lda $a
2F81- 20 47 8A  1520            jsr outchr
2F84- 20 D9 81  1530            jsr inbyte
2F87- 8D F4 00  1540            sta LFflg
2F8A- AD 0A 00  1550            lda $a
2F8D- 20 47 8A  1560            jsr outchr
2F90- 20 D9 81  1570            jsr inbyte
2F93- 8D 50 A6  1580            sta $a650 padbit #
2F96- A9 00     1590            lda #0
2F98- 8D 53 A6  1600            sta techo
2F9B- A9 0D     1610            lda #13
2F9D- 60        1620            rts
2F9E- A0 00     1630 cold       ldy #00
2FA0- A9 D7     1640            lda #chr>
2FA2- 85 EC     1650            sta *chrindex
2FA4- A9 2F     1660            lda #h,chr>
2FA6- 85 ED     1670            sta *chrindex+1
2FA8- 20 F8 30  1680            jsr outputchr
2FAB- 20 58 8A  1690            jsr intchr
2FAE- 29 7F     1700            and #$7f
2FB0- C9 79     1710            cmp #'y
2FB2- F0 03     1720            beq rptcold
2FB4- 4C 76 2C  1730            jmp menu
2FB7- A9 FD     1740 rptcold    lda #chr2
2FB9- 85 EC     1750            sta *chrindex
2FBB- A9 2F     1760            lda #h,chr2
2FBD- 8D ED 00  1770            sta chrindex+1
2FC0- B1 EC     1780            lda (chrindex),y
2FC2- AA        1790            tax
2FC3- A9 80     1800            lda #min
2FC5- 85 EC     1810            sta *chrindex
2FC7- A9 02     1820            lda #h,min
2FC9- 8D ED 00  1830            sta chrindex+1
2FCC- 8A        1840            txa
2FCD- C8        1850            iny
2FCE- C9 FF     1860            cmp #$ff
2FD0- F0 7D     1870            beq brn7
2FD2- 91 EC     1880            sta (chrindex),y
2FD4- 4C B7 2F  1890            jmp rptcold
2FD7- 54 79 70  1900 chr>       .by 'Type y to execute a cold start to RAE' $ff
```

```
2FDA- 65 20 79
2FDD- 20 74 6F
2FE0- 20 65 78
2FE3- 65 63 75
2FE6- 74 65 20
2FE9- 61 20 63
2FEC- 6F 6C 64
2FEF- 20 73 74
2FF2- 61 72 74
2FF5- 20 74 6F
2FF8- 20 52 41
2FFB- 45 FF
2FFD- 03 6D 33  1910 chr2       .by 03 'm3' 13 '20' '002c' '4c03b0' 13
3000- 0D 32 30
3003- 30 30 32
3006- 63 34 63
3009- 30 33 62
300C- 30 0D
                1920 ;Change 0030 to be consistent with the beginning of this pr
ogram. *>!
300E- 67 62 30  1930            .by 'gb003' 13 'set $200 $2eff'  13 'fo c' 13 'a
u 10' 13
3011- 30 33 0D
3014- 73 65 74
3017- 20 24 32
301A- 30 30 20
301D- 24 32 65
3020- 66 66 0D
3023- 66 6F 20
3026- 63 0D 61
3029- 75 20 31
302C- 30 0D
                1940 ;Change 2eff to be ram-1 *>!
302E- 31 30 2E  1950            .by '10.rr' 13 '.m 0 73 1003 3' 13 '.p 1 5' 13 '
.t' 13
3031- 72 72 0D
3034- 2E 6D 00
3037- 20 30 20
303A- 37 33 20
303D- 31 30 30
3040- 33 20 33
3043- 0D 2E 70
3046- 20 31 20
3049- 35 0D 2E
304C- 74 0D
304E- FF        1960            .by $ff
304F- 18        1970 brn7       clc
3050- 98        1980            tya
3051- 69 80     1990            adc #min
3053- 8D CA 00  2000            sta inputdex
3056- A9 00     2010            LDA #00
3058- 69 02     2020            ADC #H,min
305A- 8D CB 00  2030            STA inputdex+1
305D- 20 9E 30  2040            JSR echo
3060- 4C 00 B0  2050            jmp $b000 Cold Start
3063- A0 00     2060 input      ldy #00
3065- A9 47     2070            lda #71
3067- 20 47 8A  2080            jsr outchr
306A- A9 4F     2090            lda #79
306C- 20 47 8A  2100            jsr outchr
306F- A9 0A     2110            lda #10
3071- 20 47 8A  2120            jsr outchr
3074- A9 80     2130            lda #$80
3076- 8D 53 A6  2140            sta techo
3079- A9 80     2150            LDA #min INITIALIZE WORKING INDEX
```



```
307B- 85 CA     2160            sta *inputdex
307D- A9 02     2170            LDA #H,min
307F- 85 CB     2180            STA *inputdex+1
3081- 20 58 8A  2190 rpt1       jsr intchr
3084- 91 CA     2200            sta (inputdex),y
3086- E6 CA     2210            inc *inputdex
3088- D0 02     2220            bne goon
308A- E6 CB     2230            inc *inputdex+1
308C- A5 CA     2240 goon       lda *inputdex
308E- C9 FF     2250            cmp #L,max
3090- D0 EF     2260            bne rpt1
3092- A5 CB     2270            lda *inputdex+1
3094- C9 2F     2280            cmp #H,max
3096- D0 E9     2290            bne rpt1
3098- 20 72 89  2300            jsr $8972 BEEP-TO MUCH
309B- A9 0D     2310 stop       lda #13
309D- 60        2320            rts
309E- A9 80     2330 echo       LDA #min
30A0- 85 F0     2340            STA *outputdex
30A2- A9 02     2350            LDA #H,min
30A4- 85 F1     2360            sta *outputdex+1
30A6- A9 B0     2370            LDA #vec Change invec to get input from buffer.
30A8- 8D 61 A6  2380            STA $A661 invec
30AB- A9 30     2390            LDA #H,vec
30AD- 8D 62 A6  2400            STA $A662
30B0- A0 00     2410 vec        LDY #0
30B2- B1 F0     2420            LDA (outputdex),Y
30B4- A8        2430            TAY
30B5- E6 F0     2440            inc *outputdex
30B7- D0 02     2450            bne cont3
30B9- E6 F1     2460            inc *outputdex+1
30BB- A5 F0     2470 cont3      lda *outputdex
30BD- CD CA 00  2480            cmp inputdex
30C0- D0 12     2490            bne QUIT
30C2- AD F1 00  2500            lda outputdex+1
30C5- CD CB 00  2510            cmp inputdex+1
30C8- D0 0A     2520            BNE QUIT
30CA- A9 0D     2530            LDA #intchr@
30CC- 8D 61 A6  2540            sta $a661 invec
30CF- A9 2E     2550            LDA #h,intchr@
30D1- 8D 62 A6  2560            STA $A662
30D4- 98        2570 QUIT       TYA
30D5- 60        2580            RTS
30D6- 20 88 81  2590 outchr@    jsr $8188 saver
30D9- AE F5 00  2600            ldx SPflg If SPflq set then output space before
c/r. Some systems
                2610 ;stop input upon receiving c/r at the beginning of a line.
30DC- F0 0B     2620            beq cont4
30DE- C9 0D     2630            cmp #13 ?CR
30E0- D0 07     2640            bne cont4
30E2- A9 20     2650            lda #32 spc
30E4- 20 A0 8A  2660            jsr $8aa0 tout
30E7- A9 0D     2670            lda #13
30E9- AE F4 00  2680 cont4      ldx LFflg If LFflg set then inhibit the output o
f linefeeds,
                2690 ;as needed for SYM-1 transfer to other systems.
30EC- F0 04     2700            beq cont1
30EE- C9 0A     2710            cmp #10 ?LF
30F0- F0 03     2720            beq cont
30F2- 20 A0 8A  2730 cont1      jsr $8aa0 tout
30F5- 4C B8 81  2740 cont       jmp $81b8 resxaf
30F8- A2 00     2750 outputchr  ldx #00
30FA- A1 EC     2760 rpt        lda (chrindex,x)
30FC- C9 FF     2770            cmp #$ff
30FE- D0 01     2780            bne brn1
3100- 60        2790            rts
```

```
3101- 20 47 8A  2800 brn1       jsr outchr
3104- E6 EC     2810            inc *chrindex
3106- D0 F2     2820            bne rpt
3108- E6 ED     2830            inc *chrindex+1
310A- 4C FA 30  2840            jmp rpt
310D- 20 58 8A  2850 Dbl.esc    jsr intchr
3110- 29 7F     2860            and #$7f
3112- 85 E8     2870            sta *symb
3114- A2 06     2880            ldx #6        ;number of prom phrases *>!
3116- A9 AB     2890            lda #prom
3118- 85 F6     2900            sta *prsdex
311A- A9 2D     2910            lda #h,prom
311C- 85 F7     2920            sta *prsdex+1
311E- 20 4A 32  2930            jsr phrase
3121- AE CC 00  2940            ldx nuprs
3124- F0 0D     2950            beq Beep
3126- A9 00     2960            lda #ram
3128- 85 F6     2970            sta *prsdex
312A- A9 2B     2980            lda #h,ram
312C- 85 F7     2990            sta *prsdex+1
312E- 20 4A 32  3000            jsr phrase
3131- A9 20     3010            lda #$20
3133- 4C 72 89  3020 Beep       jmp $8972 Beep
3136- A0 00     3030 Define     ldy #0
3138- E6 CC     3040            inc *nuprs
313A- A6 CC     3050            ldx *nuprs
313C- A9 00     3060            lda #ram
313E- 85 F6     3070            sta *prsdex
3140- A9 2B     3080            lda #h,ram
3142- 85 F7     3090            sta *prsdex+1
3144- CA        3100 Alpha      dex
3145- F0 0B     3110            beq charac
3147- A9 FF     3120            lda #$ff
3149- 20 80 32  3130            jsr ff?
314C- 20 88 32  3140            jsr inc.prsdex
314F- 4C 44 31  3150            jmp Alpha
3152- A9 89     3160 charac     lda #\chr
3154- 85 EC     3170            sta *chrindex
3156- A9 31     3180            lda #h,\chr
3158- 85 ED     3190            sta *chrindex+1
315A- 20 F8 30  3200            jsr outputchr
315D- 20 8F 32  3210            jsr limit?
3160- 20 58 8A  3220            jsr intchr
3163- 20 47 8A  3230            jsr outchr
3166- 29 7F     3240            and #$7f
3168- 91 F6     3250            sta (prsdex),y
316A- A9 2D     3260            lda #'-
316C- 20 47 8A  3270            jsr outchr
316F- 20 88 32  3280 beta       jsr inc.prsdex
3172- 20 8F 32  3290            jsr limit?
3175- 20 58 8A  3300            jsr intchr
3178- 20 47 8A  3310            jsr outchr
317B- C9 9B     3320            cmp #$9b ?escape
317D- F0 05     3330            beq Stop
317F- 91 F6     3340            sta (prsdex),y
3181- 4C 6F 31  3350            jmp beta
3184- A9 FF     3360 Stop       lda #$ff
3186- 91 F6     3370            sta (prsdex),y
3188- 60        3380            rts
3189- 54 79 70  3390 \chr       .by 'Type symbol,text,esc
' 13 $ff
318C- 65 20 73
318F- 79 6D 62
3192- 6F 6C 2C
3195- 74 65 78
```



```
3198- 74 2C 65
319B- 73 63 0A
319E- 0D FF
31A0- A9 ED     3400 Quit       lda #chr@
31A2- 85 EC     3410            sta *chrindex
31A4- A9 31     3420            lda #h,chr@
31A6- 85 ED     3430            sta *chrindex+1
31A8- 20 F8 30  3440            jsr outputchr
31AB- 20 58 8A  3450            jsr intchr
31AE- 29 7F     3460            and #$7f
31B0- C9 79     3470            cmp #'y
31B2- F0 03     3480            beq CoNt
31B4- 4C 76 2C  3490            jmp menu
31B7- C6 CC     3500 CoNt       dec *nuprs
31B9- A9 FF     3510            lda #begin-1
31BB- 8D 4A A6  3520            sta param
31BE- A9 2B     3530            lda #h,begin-1
31C0- 8D 4B A6  3540            sta param+1
31C3- A5 CA     3550            lda *inputdex
31C5- 85 F6     3560            sta *prsdex
31C7- A5 CB     3570            lda *inputdex+1
31C9- 85 F7     3580            sta *prsdex+1
31CB- 20 88 32  3590            jsr inc.prsdex
31CE- A5 F6     3600            lda *prsdex
31D0- 8D 4C A6  3610            sta param+2
31D3- A5 F7     3620            lda *prsdex+1
31D5- 8D 4D A6  3630            sta param+3
31D8- A5 F2     3640            lda *putdex
31DA- D0 02     3650            bne hope
31DC- C6 F3     3660            dec *putdex+1
31DE- C6 F2     3670 hope       dec *putdex
31E0- A5 F2     3680            lda *putdex
31E2- 8D 4E A6  3690            sta param+4
31E5- A5 F3     3700            lda *putdex+1
31E7- 8D 4F A6  3710            sta param+5
31EA- 4C 40 87  3720            jmp $8740 block move
31ED- 54 79 70  3730 chr@       .by 'Type y to delete last phrase retreived' $ff
31F0- 65 20 79
31F3- 20 74 6F
31F6- 20 64 65
31F9- 6C 65 74
31FC- 65 20 6C
31FF- 61 73 74
3202- 20 70 68
3205- 72 61 73
3208- 65 20 72
320B- 65 74 72
320E- 65 69 76
3211- 65 64 FF
3214- A0 00     3740 Print      ldy #0
3216- A6 CC     3750            ldx *nuprs
3218- E8        3760            inx
3219- A9 00     3770            lda #ram
321B- 85 F6     3780            sta *prsdex
321D- A9 2B     3790            lda #h,ram
321F- 85 F7     3800            sta *prsdex+1
3221- CA        3810 gone       dex
3222- F0 23     3820            beq StOp
3224- B1 F6     3830            lda (prsdex),y
3226- 20 47 8A  3840            jsr outchr
3229- A9 2D     3850            lda #'-
322B- 20 47 8A  3860 going      jsr outchr
322E- 20 88 32  3870            jsr inc.prsdex
3231- B1 F6     3880            lda (prsdex),y
3233- C9 FF     3890            cmp #$ff
3235- D0 F4     3900            bne going
```

```
3237- 20 88 32  3910            jsr inc.prsdex
323A- A9 0D     3920            lda #13
323C- 20 47 8A  3930            jsr outchr
323F- A9 0A     3940            lda #10
3241- 20 47 8A  3950            jsr outchr
3244- 4C 21 32  3960            jmp gone
3247- A9 0D     3970 StOp       lda #13
3249- 60        3980            rts
324A- A0 00     3990 phrase     ldy #00
324C- B1 F6     4000            lda (prsdex),y
324E- C5 E8     4010            cmp *symb
3250- F0 0C     4020            beq Output
3252- A9 FF     4030            lda #$ff
3254- 20 80 32  4040            jsr ff?
3257- 20 88 32  4050            jsr inc.prsdex
325A- CA        4060            dex
325B- D0 ED     4070            bne phrase
325D- 60        4080            rts
325E- 20 88 32  4090 Output     jsr inc.prsdex
3261- A5 F6     4100            lda *prsdex
3263- 85 F2     4110            sta *putdex
3265- 85 F0     4120            sta *outputdex
3267- A5 F7     4130            lda *prsdex+1
3269- 85 F3     4140            sta *putdex+1
326B- 85 F1     4150            sta *outputdex+1
326D- A9 FF     4160            lda #$ff
326F- 20 80 32  4170            jsr ff?
3272- 68        4180            pla
3273- 68        4190            pla
3274- A5 F6     4200            lda *prsdex
3276- 85 CA     4210            sta *inputdex
3278- A5 F7     4220            lda *prsdex+1
327A- 8D CB 00  4230            sta inputdex+1
327D- 4C A6 30  4240            jmp echo+8
3280- 20 88 32  4250 ff?        jsr inc.prsdex
3283- D1 F6     4260            cmp (prsdex),y
3285- D0 F9     4270            bne ff?
3287- 60        4280            rts
3288- E6 F6     4290 inc.prsdex inc *prsdex
328A- D0 02     4300            bne ipdend
328C- E6 F7     4310            inc *prsdex+1
328E- 60        4320 ipdend     rts
                4330 limit?
328F- A5 F6     4340            lda *prsdex
3291- C9 00     4350            cmp #begin
3293- D0 15     4360            bne QuIt
3295- AD F7 00  4370            lda prsdex+1
3298- C9 2C     4380            cmp #h,begin
329A- D0 0E     4390            bne QuIt
329C- A9 AB     4400            lda #chr<
329E- 85 EC     4410            sta *chrindex
32A0- A9 32     4420            lda #h,chr<
32A2- 85 ED     4430            sta *chrindex+1
32A4- 20 F8 30  4440            jsr outputchr
32A7- 68        4450            pla
32A8- 68        4460            pla
32A9- 60        4470            rts
32AA- 60        4480 QuIt       rts
32AB- 45 78 63  4490 chr<       .by 'Exceeding RAM allocation, entry not retaine
d' $ff
32AE- 65 65 64        32BD- 63 61 74        32CC- 6F 74 20
32B1- 69 6E 67        32C0- 69 6F 6E        32CF- 72 65 74
32B4- 20 52 41        32C3- 2C 20 65        32D2- 61 69 6E
32B7- 4D 20 61        32C6- 6E 74 72        32D5- 65 64 FF     4500            .en
32BA- 6C 6C 6F        32C9- 79 20 6E
```


USING MAE/STP ON THE COM-64

Since we'll soon be doing much of our work in the future on the COM-64, we thought we'd give MAE/STP a try. STP, for Simplified Text Processor, is MAE's equivalent of RAE's SWP. Our first thoughts are these: We find the 40 column screen with its larger characters easier to read, but it will take a while to get used to the narrower display, and the screen editing features available on the -64 sure do beat the CTRL-F ED function used in RAE.

STP has a macro ".RU $xxxx", where $xxxx is the address of any machine language subroutine. The macro is used to call subroutines for intelligent printer control. This was not needed in SWP, where escape sequences and control codes could be entered directly from the terminal; this cannot be done on most CBM machines. These subroutines will be an early order of business.

Please see that this is the only part of any issue of SYM-PHYSIS which we did NOT do on a SYM. We feel slightly guilty about this so it's back to SYM for the rest of this last issue!

COM-64/KTM-2 COMPATIBILITY

Judging by our incoming mail, many of you out there are getting Commodore 64s as second computers. This is part of the reason so much of this issue is devoted to the compatibility between SYM and -64. For those of you who are into RAE and SWP and like the 80 column display of the KTM-2/80 for your program development (with long ";" comments in your source code) and for word processing), here is some really good news, reproduced from the MAE manual:

MAE Macro Assembler/Text Editor for Commodore Computers

19. CONNECTION OF A SERIAL DEVICE

A serial device may be connected to your PET and controlled by MAE software. MAE generates data in TTY (or RS232) data format on the USER port (bit 7 pin L = output, bit 6 pin K = input). The data format consists of one start, seven data, and two stop bits. Since these signals on the user port are TTL levels, circuitry may be required to provide a proper electrical interface. We have found, though, that RS232 terminals such as the Synertek KTM-80 can be connected directly to the user port. If you do provide interface circuitry, you should not invert the signals as they are in positive true state.

The commands]TI and]TO are provided to direct MAE to input or output on this serial port.

18. CONTROL CODES (for Serial Device)

The following applies to the optional serial device connected to the PET. Ascii characters whose hex values are between hex 00 and 20 are normally non-printing characters. With a few exceptions, these characters will be output in the following manner: ↑c where ↑c is the associated printable character if hex 40 was added to its value. For example, ascii 03 will be output as ↑C, and 18 as ↑X, etc.

[NOTE: STP does not support the printing of the "^", so these were added "by hand". Printing of the "^" can be provided by substitution of some other seldom used character, such as "\", for the "symbolic space". We have a special version of SWP for just this purpose!]

In addition, some of these control codes have special functions in MAE.

Control codes which have special functions are:

CODE		DESCRIPTION
@		Null (hex 00)
B		Restore zero page and go to Basic
C		Restore zero page and go to Monitor
G	*	Bell
H	*	Backspace (delete previously entered char.)
I	*	Horizontal tab to next 8-th char. position
J	*	Line feed
M	*	Carriage return
O		Continue processing but no output (same as DEL)
Q	*	Continue after stop via break key
X		Delete entire line altered
Y		Restore zero page and jump to location $0000. (you may reenter at $5003)
Z		Terminate processing and go to "]" level
[	*	Escape character

* = Non-printing control character.

[NOTE: Compare this with page 8-1 of the RAE-1 Reference Manual.]

The manual was written for the PETs, with 6502 and VIAs, but the MAE 64 version has been modified to work with 6510 and CIAs. The major changes are these:

1) 6502 Microsoft BASIC has its USR JMP at $0000. Since the 6510 uses addresses $00 and $01 for its internal control port, the 6510 version of Microsoft BASIC puts the USR JMP at $0310, instead. This implies a modification in the CTRL Y jump location.

2) The USER port on the COM-64 differs from that on the PET, using PA2 pin M = output, FLAG* pin B = input. Only a three wire cable is needed; all the software is there, already.

Once this newsletter is out we'll add a KTM-2/80 to our COM-64, and then, as far as RAE and SWP (MAE/STP) are concerned, it'll be as if we were still on a SYM, giving us the best of all possible worlds. [Incidentally, MAE is the only 6502 assembler we have found where the Editor and Assembler are co-resident, and which is so beautifully integrated and co-resident with both BASIC and a very powerful, much extended ML monitor.]

WHAT TO DO WITH YOUR OLD SYM(S)

We have received so many letters of thanks and commendation that it almost makes us blush to think of them, and we are much too modest to even consider printing any of them. We make an exception in this one case, because of the extremely good suggestion of what one might do with his "retired" SYM.

Incidentally, in our four years with SYM-PHYSIS, we have received only two really harsh criticisms, and these were answered by advising the writer and the telephoner to re-read the documentation (of course, they had not yet even given it a first reading), to find out where they had failed.

Dear Lux, Jean, and company:

I'm sorry to hear that the SYM line in coming to a halt, but I guess that's progress. I am especially sorry to hear the next issue will be the last, but I know (and you especially know) you've done a super job and it's probably best to quit while you're ahead. JUST WANTED YOU TO KNOW YOUR FOUR YEARS OF WORK HAVEN'T GONE UNNOTICED!!!! I still have all of my back issues, and someday when I give my grandkid my antique SYM-1, it will have a complete set of documentation and especially the entire set of user notes. Thanks for all of the good work. I look forward to the "farewell" issue. You have truly been involved in a pioneering effort and have set a standard for "interactive" user groups. So long.

SYM-cerely,

Jon D. Jackson

Jon D. Jackson
205 Tapoco Drive
Eglin AFB, FL 32542

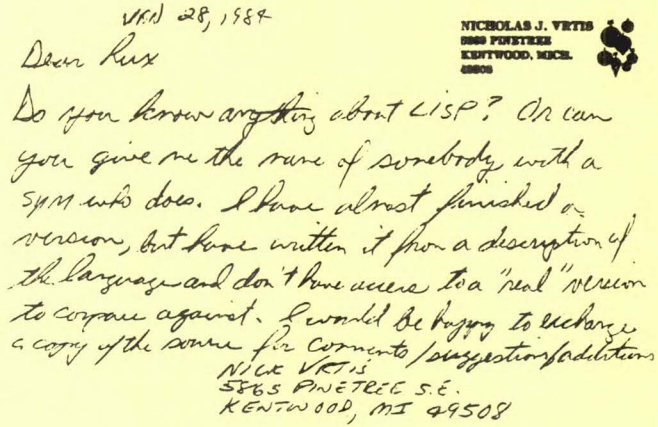
A NOTE TO JON'S GRANDKID(S)

Your grandfather wrote that note above. He really thought about you often, even way back then! And best wishes from us, too. /s/ Lux & Jean

LISP, ANYONE?

NICK VRTIS sent the request for help with his SYM version of LISP on the postal card reproduced here. We enjoyed his TINY PILOT for SYM, and are looking forward to trying his finished LISP. We suggest that LISPers get in touch with Nick and take him up on his offer of a preliminary version.

Nick has been quite prolific with his SYM. He also sent along a copy of his SYM/CBM-1541 disk drive software.



Feb 28, 1984

Dear Lux

Do you know anything about LISP? Or can you give me the name of somebody with a SYM who does. I have almost finished a version, but have written it from a description of the language and don't have access to a "real" version to compare against. I would be happy to exchange a copy of the source for comments/suggestions/additions

NICK VRTIS
5865 PINETREE S.E.
KENTWOOD, MI 49508

NICHOLAS J. VRTIS
5865 PINETREE
KENTWOOD, MICH.
49508

WHY THE SWITCH FROM SYM?

While there are "loners" in the computer field, as in any other hobby/profession, even loners feel the occasional need of an "audience" to whom they can proudly display what they have done, are doing, and are going to do. Thus, it can be much more ego-satisfying being associated with a computer with a large user base; such popularity tends to breed greater sales, and hence even more popularity, etc.

We have often felt quite frustrated when friends and/or acquaintances asked us about our personal computer, of which they had never before heard, and certainly had never seen advertised on television. Have any of you ever felt the same? It is, of course, much different with the Commodore! And how much great software is now becoming available!

We could recommend the SYM-1 only to our technically oriented friends, but the Commodore 64 is the ideal beginner's system, at a really low budget cost, for those who have not been deluded by the much over-rated sales talk of IBM or CP/M or S-100 compatibility, or the need for 16-bits, or more than 64 K of RAM. And that is why we switched our loyalty. Our only regret is that Synertek really missed out on a truly golden opportunity here.

We suggested to Synertek Systems, over three years ago, that they had a real "sleeper" in the SYM-1/KTM-2 combination, and that if they combined these two items on a single board, and threw in BAS-1 and RAE-1 FREE, at a price of, say, $795, they might be able to compete quite seriously with Apple, etc. Such a system would have been far more useful than Rockwell's AIM-65, and could very well have become the "VW", or "Model T", or "Jeep", "work-horse" of the computer field. Instead Commodore filled the void, first with the VIC=20, then the Commodore 64.

Synertek Systems management decided to go for the OEM market instead, and leave the educational/hobbyist market to others. Commodore's success in keeping costs low has been attributed, in part, at least, to their "vertical integration", since they were in a position to make the majority of their own LSI chips. Synertek is certainly in the same position, and the parent corporation, Honeywell, does have some expertise in mass marketing.

Although we offered our services as a "Beta" test site to Synertek, we were never advised of any of their new products in advance. The first we ever heard of any of their new products, such as the SYM-2, Mod-69, KTM-3, etc., was by receiving a brochure, often long after their availability.

One product we did not even learn about until just this week, was the SYM-2/69, which uses the 6809 instead of the 6502. This would have been an ideal way to get started in the 6809 field, far less expensive than retrofitting the SYM-1 with the Mod-69 "upgrade" kit. This we think might have been a viable product, if anyone outside Synertek (and very few inside even knew) had known about it. We discovered its existence only when a former student told us that a friend had purchased one at a very low price, somewhere. He told us that it differed from the SYM-2 mainly in the fact that some of the sockets were in alternate positions on the board. Sure enough we examined a SYM-2, and discovered that alternate socket positions were provided for several chips. We then reexamined a SYM-2 schematic, and, sure enough, the schematic indicated that either the 6502 or 6809 could be installed!

We have the greatest admiration for the engineering which has gone into Synertek Systems products, especially into the versatility aspect. Never before have we seen boards with so many jumper and trace cutting and alternate socket options as in the Synertek Systems product line.

Now that they are switching to a new, non computer, OEM product line, which they are not yet ready to announce publicly, we wish them great success.

Meanwhile, we are very much enjoying helping many of our new friends, especially the teenagers, and the junior/senior high school teachers, get started "right" in the computer field. This is how we'll be spending our retirement.

AN "ALL-BASIC" BASIC DATA SAVER

We have begun to use BASIC much more frequently then in the past,

because so many of the COM-64 and VIC=20 programs we have seen lately are in their "natural" language, BASIC. We are much impressed with the skills which so many BASIC programmers have acquired in the use of PEEKs, POKEs, READs, and DATA statements to get the systems to do their bidding.

This is illustrated in the letter and sample program below, which shows how to modify the starting values of variables so that a SAVEd version of the program will now hold the "updated" starting values. This program is well worth studying to see how it can be done.

The SYM cassette interface is so much faster than that in the various Commodore systems that it might be well worth adapting the Commodore's idea of treating the cassette as an alternate I/O "device", and PRINT#ing and GET#ing variables from an automatically managed "cassette buffer" as SEQuential files. While we have learned a lot from a study of SYM's SUPERMON, there is a lot more to be learned, especially about I/O and data file management, from a study of the Commodore's "KERNAL".

In passing, it should be pointed out that the SYM's implementation of USR is far more powerful than that in any other version of Microsoft BASIC, in that an essentially unlimited number of parameters may be passed. It is even more powerful than the SYSs and CALLs included in other Microsoft BASICs.

Dear Lux,

My sons and I have put several games into our SYM using BASIC. Some of the games are rather involved and utilize changing variables to make the game 'go'. Occasionally, usually due to supper or bedtime, a game will have to be interrupted. This results in shutting down the system and reloading and re-starting the game at another time. What we needed was a way to save the game with its current parameters. This requirement led to much experimentation and finally a method for accomplishing the goal. The short program which follows isn't very productive but it does illustrate a method for saving a program with its current parameters by poking the current values into a DATA statement in the beginning of the program.

Hopefully, some SYM user, somewhere, may be able to make use of it. It has proved helpful to us.

Steve Schady

7 Enid Road

East Lyme, Ct. 06333

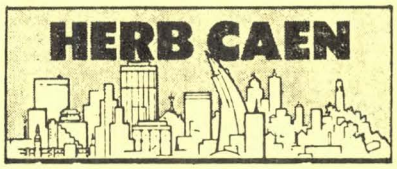
```
 10 DATA001,001,001,001,0: REM ENTER THIS LINE WITH NO SPACES !
 20 READ A, B, C, D, Y: V=10: W=V↑2: U=48
 30 E=A+B+C+D: PRINT A, B, C, D, E: Y=Y+1
 40 A=A+1: B=B+2: C=C+3: D=D+4
 50 IF Y=10 THEN Y=0
 60 FOR F=1 TO 1000: NEXT F: PRINT CHR$(26)
 70 INPUT "(S)AVE OR (P)LAY OR (Q)UIT ?    ";B$
 80 A$=LEFT$(B$,1)
 90 IF A$="Q" THEN END
100 IF A$="P" THEN 30
110 G=INT(A/V):H=INT(A/W):J=A-V*G:K=518:L=519:M=520:GOSUB200
120 G=INT(B/V):H=INT(B/W):J=B-V*G:K=522:L=523:M=524:GOSUB200
130 G=INT(C/V):H=INT(C/W):J=C-V*G:K=526:L=527:M=528:GOSUB200
140 G=INT(D/V):H=INT(D/W):J=D-V*G:K=530:L=531:M=532:GOSUB200
150 POKE 534, Y+U: PRINT: PRINT
160 PRINT "PUT IN A TAPE AND PUT TAPE RECORDER IN RECORD MODE."
170 PRINT: INPUT "READY ?    ";C$: PRINT
180 SAVE A: PRINT " PROGRAM SAVED AS "; CHR$(34);"A"; CHR$(34);
190 PRINT "WITH CURRENT PARAMETERS. ": END
200 I=G-V*H: POKE K, H+U: POKE L, I+U: POKE M, J+U: RETURN
```

A FINAL APOLOGY FOR THE UNANSWERED LETTERS

Every letter received here is opened and read by Jean on the same day it arrives. Jean handles the "business" aspects of the letters, usually on the very same day, if at all possible. The letters requesting help go into my "incoming" basket, and it is often several months before they can be read by me. [The truly "emergency" ones are handed to me directly, and Jean gets me to answer those at once.]

The letters average some 15-20 per week, and too many of them would take hours of "research" time to work out an answer. Thus, much too many of them have, of necessity, had to go unanswered. A certain percentage of the requests for help, fortunately, have been "self-answering", in the sense that the answers could be found by re-reading the documentation a little more carefully this time.

We have truly enjoyed working with SYM-PHYSIS; the only painful part was in being unable to keep up with the correspondence, and having to apologize for it in each issue. Again, please accept our apologies. The only time we feel less guilty about our negligence is when we realize that we are not alone (guiltiness loves company, we guess). Here's what Herb Caen, the San Francisco Chronicle columnist, has to say in his (i.e., our!) defense.



San Franciscaena

I DON'T MIND telling you that I get a lot of mail. The reason I don't mind is that I can now apologize publicly for not answering it all. Lordy knows we try, but 1000 or so letters a week pile up fast. Still, every letter deserves a reply. This is not the golden age of letter- writing, and I am still impressed that some friendly stranger in faraway Olema or Paradise will sit down, set pen to paper and unburden his or her thoughts on this subject or that to the friendly stranger — 'tis I! — at this end of the line. In the age of instant communication — and rather iffy mail service — this is an amazing thing. My guilt increases exponentially with the height of the pile of unanswered mail teetering on the edge of my desk. Now and then, a few letters flutter off and into the wastebasket. It pains me to confess that sometimes I am too bushed to dredge them out.

★ ★ ★

ONCE, YEARS AGO, on the eve of a vacation, I picked up the entire batch and threw it in the wastebasket. It marked the first time I had seen the top of my desk in at least a decade. The action should have brought me release and relief, but it merely spoiled my vacation. I realize that a lot of letter-writers don't expect replies. It's the world we live in. Many is the time I get a delighted "Hey, you answered my letter! Don't you know nobody answers letters these days?", which gives me another letter to answer.

★ ★ ★

ULTRA HIGH PERFORMANCE SCOPE LINE DRIVER

As the years have gone by, the programs submitted for publication have grown better and better, but, unfortunately, have also grown much too long for publication in these few pages.

This is particularly true of a program submitted last fall by Leland Goertz. Rather than just letting the program get "lost", we'll print parts of Leland's material, so that those interested can contact him directly for a copy, as suggested in his letter. In "laboratory-type" applications an oscilloscope is frequently more easily available than a full terminal, and we can easily see the utility of his program in such an environment.

The last paragraph of his letter mentions some interesting things he has done with his SYM; some of you may wish to correspond with him on these also! And we really do agree with him on his comparison of SYM-1 vs. VIC=20, in spite of any contrary impression created by our comments elsewhere in this newsletter. We remember when the Commodore PET first appeared how unhappy so many initial purchasers were because there were no ML capabilities built-in, and how Commodore had to provide an ML Monitor (Tiny Mon) on cassette to satisfy them. The VIC=20 does really need a version of MAE in ROM to bring it up to SYM-1 power, but this is highly unlikely to appear. Anyway, we have a strong affection for both.

Dear Lux and SYM-1 users,

I am submitting the enclosed program, ULTRA HIGH PERFORMANCE SCOPE LINE DRIVER, for your consideration of publication in SYM-PHYSIS. If it is worthy of publication, you have my o.k..

The software is roughly based on the scope line driver supplied by SYNERTEK in the SYM REFERENCE MANUAL. All I have done is alter the way SYNERTEK'S driver accesses character data. For a futher inquiry into the differences between the two, I suggest studying the source code of each driver.

For those who have scopes with Z-axis modulation capabilities, there is an added plus (both with this driver and SYNERTEK'S). Connect the Z-axis probe to the base of Q10. This will rid the display of the bright base line and makes a much easier to read display.

Because of my systems current configuation, the program has been assembled at $C800. This can be altered to $0200 by changing the following locations: 0246=02 , 0261=02 , 0281=02 , 0286=02 , 0288=02 , 02AF=02 , 028C=02.

The driver will fit in a bare bones SYM with 2K of memory and still leave room for most programs that will run in 1K of memory.

I am sorry that I can not supply you with RAE source code on tape and hope that the enclosed photocopy is clear enough for publication. I have, however, enclosed a tape with the object code assembled at $0200. A .G $028C will start the driver. I will also supply any users' with the same tape for $6.00. Please write or call the above address for orders.

The software is romable and contains the boot routines. The boot routines may be omitted if they are not going to be used. LOGTAB and CHRTAB can then be moved up to fill the hole that the omitted boot routines left.

I have been following Jeff Lavin's SUPER SYM article with much intrest. However, my SYM at this point in time is a $50.00 per hour money maker. Thus, I, or my SYM, do not have the time to sit down and study the procedures on how to implement the modifications. I'm sure many other SYMmers are in a simular situation. So, I would like to suggest a step-by-step instruction sheet on how to implement the SUPER SYM. Of course, a fee would be charged, but the advantages of a SUPER SYM are worth such a fee!!!

I am sorry to hear the end of the users' group and to the end of the manufacturing of the SYM. Progress I guess. I do not, however, feel that the VIC 20 can replace the SYM. I have a VIC 20 and don't like it at all. I have used my SYM for a variety of things including a 12 projector multi-media system complete with program editor, a business security system, an automatic telephone dialer, and now in the medical field as a specialized floppy reformatter. I would hate to try to implement these things on a VIC 20. I have been using the SYM since my college days and received a degree in computer science mainly using the SYM. Call me loyal to the royal. Nothing will ever replace my SYM.

Leland Goertz
40573 Road 84
Dinuba, CA 93618
(209) 255-1765

Sincerely,
Leland Goertz

[SOFTWARE DESCRIPTION IS ON PAGE 17-44]

A CALL TO COMPUTER ARTISTS

WALTER "WALLY" GLAB, a long-time SYMmer with a strong background in art, is part of a group of four artists working in the area of Computer Art. He would like to form a Special Interest Group for SYMmers, and, presumably others, working in this field, for the interchange of software and ideas. His address and telephone number are: 2538 N. Wayne, Chicago, IL 60614, (312) 525-7617.

We know that a number of readers have gotten together by mail or phone to exchange ideas and software in their own areas of interest, and hope that this will continue in the future. As you reread your back issues of SYM-PHYSIS, you may wish to contact some of the contributors and others whose addresses and phone numbers were published, for "updates", enhancements, etcetera.

ULTRA HIGH PERFORMANCE SCOPE LINE DRIVER

GENERAL

This software package enables SYM-1 users' to divert there output from the LED displays to a 32 character oscilloscope display. Display format is the standard 5 x 7 dot matrix. The driver supports the full 96 character ASCII set of printable characters. Characters may be added (up to 13,011) by writing the starting address of the new character data to PTRLO, and PTRHI.

OPERATION

The "vidio" signal is from the collector of Q10, and is 3 volts peak-to-peak with a cycle time of about 60ms. The sync pulse which begins the line should synchronize all triggered sweep scopes and most recurrent sweep scopes. The sync pulse may be brought out on a separate pin by replacing the code from SYNC to CHAR with a routine that would output a pulse on some other output line.

CONNECTION

Connect the oscilloscope vertical input to pin R on connector AA and connect the scopes ground to pin 1 of AA. For scopes with Z-axis modulation, connect the Z-axis probe to the base of Q10. This will rid the display of the bright base line. If the sync pulse was output on a seperate pin, connect the scopes trigger to this pin.

BOOT

The software may be started 1 of 3 ways. If the software is in a prom, the boot jumpers may be altered so that the system always boots to the scope driver. Using this method, the RST key will start the driver. A .G C88C (028C) will also start the driver. This is the starting address of the scope drivers boot routine. The third way to start the driver is to execute the following: SD C800 (0200)-A670. This alters SCNVEC to point to the scope driver.

USAGE

Since the scope buffer resides in system ram, a JSR ACCESS must be performed before any writing to the scope buffer can occur. One scan of the scope buffer may be performed by calling LINE ($C800 [$020]). Line can be used in the same way that SUPERMON's SCAND is used.

COMMENTS

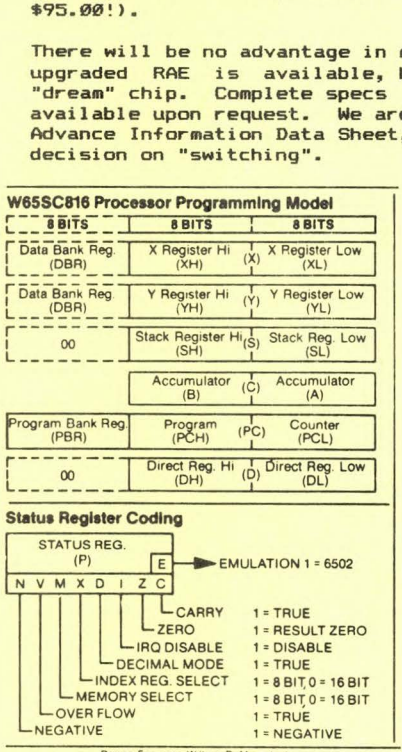
This software is roughly based on the scope line driver supplied it the SYM REFERENCE MANUAL. For further information see chapter 7.

THE W65SC802

We have not yet replaced any of our 6502 chips with The Western Design Center's W65SC02 (list price $9.50). We are doing a "paper study" of the advantages to be gained by switching, instead, to the W65SC802. There was a time when we were (very briefly) thinking of going to the 6809, but gave that idea up because of the complete lack of either hardware or software compatibility with the 6502. On the other hand the W65SC802 is totally compatible; just plug it in and go (list price is $95.00!).

There will be no advantage in making the switch UNTIL and UNLESS an upgraded RAE is available, but it is still worth looking into the new "dream" chip. Complete specs for the entire WDC product line are available upon request. We are reprinting here some extracts from their Advance Information Data Sheet, so that you can begin to make your own decision on "switching".

W65SC816 Processor Programming Model

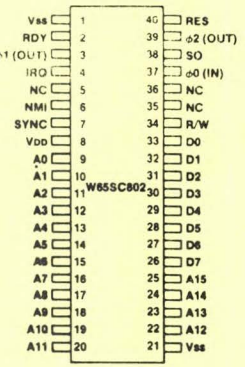


Status Register Coding

Design Engineer: William D. Mensch, Jr.

Address Mode

1. Immediate
2. Absolute
3. Absolute Long
4. Direct
5. Accumulator
6. Implied
7. Direct Indirect Indexed (IND), Y
8. Direct Indirect Indexed Long (IND), Y Long
9. Direct Indexed Indirect (IND, X)
10. Direct, X
11. Direct, Y
12. Absolute, X
13. Absolute Long, X
14. Absolute, Y
15. Relative
16. Relative Long
17. Absolute Indirect (Jump)
18. Direct Indirect
19. Direct Indirect Long
20. Absolute Indexed Indirect (Jump)
21. Stack
22. Stack Relative
23. Stack Relative Indirect Indexed
24. Block Move X, Y, C (Source, Destination, Block Length)



We are wondering what the future holds for WDC's 16-bitters. While the W65SC802 can be "retrofitted" into Apples, Ataris, SYMs, VICs (but not COM-64s), etc., by hobbyists and experimenters, such as ourselves, we do not see this as a big market, nor do we forsee any new systems being built around this chip.

The 68000, Z-8000, and the 8086/8088 have become so firmly entrenched that the W65SC816 will have a tough time getting established, especially since there are no semi-standard DOSes, such as FLEX, CP/M, MS-DOS, PC-DOS, built around the 6502. The strongest advantage it will have is that its instruction set is a superset of the 6502's, so that pre-existing software will only need to be reconfigured for the I/O and DOS peculiar to each system, and this can still be written with 6502 assemblers.

This is similar to what was done for the COM-64 CP/M card. Although a Z-80 chip is installed, the CP/M system software, including the BIOS and BDOS portions, is written in 8080 syntax, and the supplied assembler and debugger/disassembler are also only for the 8080.

W65SC816 Instructions (256 OP Codes)

A. The Original 6502 Instruction Set (151 Op Codes)

1. ADC Add Memory to Accumulator with Carry
2. AND "AND" Memory with Accumulator
3. ASL Shift Left One Bit (Memory or Accumulator)
4. BCC Branch on Carry Clear
5. BCS Branch on Carry Set
6. BEQ Branch on Result Zero
7. BIT Test Bits in Memory with Accumulator
8. BMI Branch on Result Minus
9. BNE Branch on Result Not Zero
10. BPL Branch on Result Plus
11. BRK Force Break
12. BVC Branch on Overflow Clear
13. BVS Branch on Overflow Set
14. CLC Clear Carry Flag
15. CLD Clear Decimal Mode
16. CLI Clear Interrupt Disable Bit
17. CLV Clear Overflow Flag
18. CMP Compare Memory and Accumulator
19. CPX Compare Memory and Index X
20. CPY Compare Memory and Index Y
21. DEC Decrement Memory by One
22. DEX Decrement Index X by One
23. DEY Decrement Index Y by One
24. EOR "Exclusive-or" Memory with Accumulator
25. INC Increment Memory by One
26. INX Increment Index X by One
27. INY Increment Index Y by One
28. JMP Jump to New Location
29. JSR Jump to New Location Saving Return Address
30. LDA Load Accumulator with Memory
31. LDX Load Index X with Memory
32. LDY Load Index Y with Memory
33. LSR Shift One Bit Right (Memory or Accumulator)
34. NOP No Operation
35. ORA "OR" Memory with Accumulator
36. PHA Push Accumulator on Stack
37. PHP Push Processor Status on Stack
38. PLA Pull Accumulator from Stack
39. PLP Pull Processor Status from Stack
40. ROL Rotate One Bit Left (Memory or Accumulator)
41. ROR Rotate One Bit Right (Memory of Accumulator)
42. RTI Return from Interrupt
43. RTS Return from Subroutine
44. SBC Subtract Memory from Accumulator with Borrow
45. SEC Set Carry Flag
46. SED Set Decimal Mode
47. SEI Set Interrupt Disable Status
48. STA Store Accumulator in Memory
49. STX Store Index X in Memory
50. STY Store Index Y in Memory
51. TAX Transfer Accumulator to Index X
52. TAY Transfer Accumulator to Index Y
53. TSX Transfer Stack Pointer to Index X
54. TXA Transfer Index X to Accumulator
55. TXS Transfer Index X to Stack Register
56. TYA Transfer Index Y to Accumulator

G. New Register Transfer Instructions (8 Op Codes)

1. TCD Transfer C Accumulator to Direct Register D
2. TDC Transfer Direct Register D to C Accumulator
3. TCS Transfer C Accumulator to Stack Register
4. TSC Transfer Stack Register to Accumulator C
5. TXY Transfer X to Y
6. TYX Transfer Y to X
7. XBA Exchange B and A
8. SCE Exchange Carry Bit C with Emulation Bit E.

H. New Branch, Jump and Return Instructions (6 Op Codes)

1. BRL Branch Relative Long Always (16 Bit Relative—32768 to + 32767) (Addressing Mode)
2. JML Jump Indirect Long
3. JMP Jump Absolute Long
4. JSL Jump to Subroutine Long (Uses RTL for Return)
5. JSR Jump to Subroutine (Indexed Indirect)
6. RTL Return from Subroutine Long

I. New Block Move Instructions (2 Op Codes)

1. MVN Move Block from Source (X Addressed) to Destination (Y Addressed). Block Length Defined by C. X, Y are Incremented.

B. New W65SCXXX Instructions (13 Op Codes)

1. BRA Branch Relative always
2. PLX Pull X from Stack
3. PLY Pull Y from Stack
4. PHX Push X on Stack
5. PHY Push Y on Stack
6. STZ Store Zero in Memory (Direct, Direct, X; Abs, Abs, X)
7. TRB Test and Reset Memory Bits Determined by Accumulator A (Direct and Absolute).
8. TSB Test and Set Memory Bits Determined by Accumulator A (Direct and Absolute).

C. New W65SCXXX Addressing Modes (14 Op Codes)

2. BIT Test Bits in Memory with Accumulator (Direct, X; Absolute, X; Immediate).
2. DEC Decrement (Accumulator)
3. Group I Instructions (Direct Indirect (8 Op Codes))
4. INC Increment (Accumulator)
5. JMP Jump to New Location (Absolute Indexed Indirect)

D. Group I Instructions with New Addressing Modes (48 Op Codes)

- Direct Indirect Long Indexed with Y (8 Op Codes)
- Direct Indirect Long (8 Op Codes)
- Absolute Long and Absolute Long Indexed with X (16 Op Codes)
- Stack Relative (8 Op Codes)
- Stack Relative Indirect Indexed Y. (8 Op Codes)

1. ADC Add Memory to Accumulator with Carry
2. AND "AND" Memory with Accumulator
3. CMP Compare Memory and Accumulator
4. EOR "Exclusive-or" Memory with Accumulator
5. LDA Load Accumulator with Memory
6. ORA "Or" Memory with Accumulator
7. SBC Subtract Memory from Accumulator with Borrow
8. STA Store Accumulator in Memory

E. New Push and Pull Instructions (7 Op Codes)

1. PEA Push Effective Absolute Address or Immediate Data Word on Stack
2. PEI Push Effective Indirect Address or Direct Data Word on Stack
3. PER Push Effective Program Counter Relative Indirect Address or Program Counter Relative Data Word on Stack
4. PLB Pull Data Bank Register from Stack
5. PLD Pull Direct Register from Stack
6. PHB Push Data Bank Register on Stack
7. PHD Push Direct Register on Stack
8. PHK Push Program Bank Register on stack

F. Status Register Instructions (2 Op Codes)

1. REP Reset Status Bits Defined by Immediate Byte 1 = Reset 0 = Do not change
2. SEP Set Status Bits Defined by Immediate Byte 1 = Set 0 = Do not change
2. MVP Move Block from Source (X Addressed) to Destination (Y Addressed). Block Length Defined by C. X, Y are Decremented.

J. New Co-Processor Operations (1 Op Code)

1. COP Co-Processor Instruction with Associated COP Vector and ABORT Input Supports Co-Processing Function i.e., Floating Point Processors, etc.

K. New System Control Instructions (3 Op Codes)

1. STP Stop-the-clock Instruction Stops the Oscillator Input (or 02 Input) During 02 = 1. This Mode Is Released When RES Goes to a Zero. System Initialization May Be Desired; However, if After RESET One Performed an RTI, Program Execution Begins With the Instruction Following the STP Op Code in Program Sequence.
2. WAI Wait for Interrupt Pulls RDY Low and Is Cleared by IRQ or NMI Active Input.
3. WDM There is One Reserved Op Code Defined as WDM Which Will Be Used For Future Systems. The W65SC816 Performs a No-Operation.

TAXAN MONITOR PROBLEM AND FIX

ART WILLIAMSON sent along the following note for users of Taxan monitors:

> Sorry it took me so long to get this note to you about the Taxan display problem we had.
>
> First symptoms of our failure were display blooming and loss of focus. This was followed by a loss of display.
>
> We found a capacitor, C212 on our schematic to be defective. This component is messy to replace because it is under the shield in the horizontal section of the circuit board.
>
> Two of our three displays had this failure.
>
> Hope this is helpful to other owners of the Taxan unit. Still think it is a fine display.

PRODUCT LINE RECOMMENDATION

SERGE MATOVIC, of INCON Electronics Inc., 762 Damien Way, Mississauga, Ontario, Canada L5C 3H2, (416) 273-4499, has been keeping us posted on his company's products. These include a PROGRAMMER/EMULATOR, a SIMULATOR, and a PROGRAMMABLE CONTROLLER. The EMULATOR can be used to emulate the SYM.

While we have not actually tested the products (Serge offered a loan, but we had to decline because of time pressures), we have studied the detailed spec sheets and the photographs he sent, and these equipments are the kinds of "tools" we would want to have in a research or industrial environment.

We suggest that anyone interested in this class of products phone or write Serge for additional information, including price and delivery schedules.

FUTURE PLANS

As of this date there is no word as to who will be "carrying-on" the SYM-1 product line, although Synertek is still negotiating with several possible individuals and groups. It appears likely that Synertek will continue manufacturing SYMs as long as the demand remains at its present level, but not the KTM-2 series.

We will be traveling in Europe (and will try to contact or visit as many SYMmers as possible, especially those who have visited us here in Chico) from 9 April through 16 May, 1984. We have been invited to present a talk/seminar for the Department of Electrical Engineering Science of the University of Essex, by Nigel Helsby and Ian Dillworth, and we are looking forward to this, and to finally meeting them in person. We'll then spend two weeks visiting with some of our (U.S.) East Coast SYMmers.

During the months of April and May of 1984 no telephone or mail orders will be accepted, nor will inquiries or requests for help be handled. After 1 June 1984, the Users' Group will again be available for help, and as a source of software and documentation, and we will continue to be a source for Synertek's proprietary chips and spares.

MSD \ LSD	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	MSD
0	BRK s 2 8	ORA(d,x) 2 6	COP s 2*8	ORA sr 2*4	TSB d 2*5	ORA d 2 3	ASL d 2 5	ORA(dl) 2*6	PHP s 1 3	ORA imm 2 2	ASL acc 1 2	PHD s 1*4	TSB a 3*6	ORA a 3 4	ASL a 3 6	ORA al 4*5	0
1	BPL r 2 2	ORA(d),y 2 5	ORA (d) 2•5	ORA(sr),y 2*7	TRB d 2•5	ORA d,x 2 4	ASL d,x 2 6	ORA(dl),y 2*6	CLC imp 1 2	ORA a,y 3 4	INC acc 1•2	TCS imp 1*2	TRB a 3•6	ORA a,x 3 4	ASL a,x 3 7	ORA al,x 4*5	1
2	JSR a 3 6	AND(d,x) 2 6	JSL al 4*8	AND sr 2*4	BIT d 2 3	AND d 2 3	ROL d 2 5	AND(dl) 2*6	PLP s 1 4	AND imm 2 2	ROL acc 1 2	PLD s 1*5	BIT a 3 4	AND a 3 4	ROL a 3 6	AND al 4*5	2
3	BMI r 2 2	AND(d),y 2 5	AND(d) 2•5	AND(sr),y 2*7	BIT d,x 2•4	AND d,x 2 4	ROL d,x 2 6	AND(dl),y 2*6	SEC imp 1 2	AND a,y 3 4	DEC acc 1•2	TSC imp 1*2	BIT a,x 3•4	AND a,x 3 4	ROL a,x 3 7	AND al,x 4*5	3
4	RTI s 1 7	EOR(d,x) 2 6	WDM RESERVED*	EOR sr 2*4	MVP xyc 3*7	EOR d 2 3	LSR d 2 5	EOR(dl) 2*6	PHA s 1 3	EOR imm 2 2	LSR acc 1 2	PHK s 1*3	JMP a 3 3	EOR a 3 4	LSR a 3 6	EOR al 4*5	4
5	BVC r 2 2	EOR(d) y 2 5	EOR (d) 2•5	EOR(sr),y 2*7	MVN xyc 3*7	EOR d,x 2 4	LSR d,x 2 6	EOR(dl),y 2*6	CLI imp 1 2	EOR a,y 3 4	PHY s 1•3	TCD imp 1*2	JMP al 3•4	EOR a,x 3 4	LSR a,x 3 7	EOR al,x 4*5	5
6	RTS s 1 6	ADC(d,x) 2 6	PER s 3*6	ADC sr 2*4	STZ d 2•3	ADC d 2 3	ROR d 2 5	ADC(dl) 2*6	PLA s 1 4	ADC imm 2 2	ROR acc 1 2	RTL s 1*6	JMP (a) 3 5	ADC a 3 4	ROR a 3 6	ADC al 4*5	6
7	BVS r 2 2	ADC(d),y 2 5	ADC(d) 2•5	ADC(sr),y 2*7	STZ d,x 2•4	ADC d,x 2 4	ROR d,x 2 6	ADC(dl),y 2*6	SEI imp 1 2	ADC a,y 3 4	PLY s 1•4	TDC imp 1*2	JMP(a,x) 3•6	ADC a,x 3 4	ROR a,x 3 7	APC al,x 4*5	7
8	BRA r 2•2	STA(d,x) 2 6	BRL rl 3*3	STA sr 2*4	STY d 2 3	STA d 2 3	STX d 2 3	STA(dl) 2*6	DEY imp 1 2	BIT imm 2•2	TXA imp 1 2	PHB s 1*3	STY a 3 4	STA a 3 4	STX a 3 6	STA al 4*5	8
9	BCC r 2 2	STA(d),y 2 6	STA (d) 2•5	STA(sr),y 2*7	STY d,x 2 4	STA d,x 2 4	STX d,y 2 4	STA(dl),y 2*6	TYA imp 1 2	STA a,y 3 5	TXS imp 1 2	TXY imp 1*2	STZ a 3•4	STA a,x 3 5	STZ a,x 3•5	STA al,x 4*5	9
A	LDY imm 2 2	LDA(d,x) 2 6	LDX imm 2 2	LDA sr 2*4	LDY d 2 3	LDA d 2 3	LDX d 2 3	LDA(dl) 2*6	TAY imp 1 2	LDA imm 2 2	TAX imp 1 2	PLB s 1*4	LDY a 3 4	LDA a 3 4	LDX a 3 4	LDA al 4*5	A
B	BCS r 2 2	LDA(d),y 2 5	LDA(d) 2•5	LDA(sr),y 2*7	LDY d,x 2 4	LDA d,x 2 4	LDX d,y 2 4	LDA(dl)y 2*6	CLV imp 1 2	LDA a,y 3 4	TSX imp 1 2	TYX imp 1*2	LDY a,x 3 4	LDA a,x 3 4	LDX a,y 3 4	LDA al,x 4*5	B
C	CPY imm 2 2	CMP(d,x) 2 6	REP imm 2*3	CMP sr 2*4	CPY d 2 3	CMP d 2 3	DEC d 2 5	CMP(dl) 2*6	INY imp 1 2	CMP imm 2 2	DEX imm 1 2	WAI imp 1*3	CPY a 3 4	CMP a 3 4	DEC a 3 6	CMP al 4*5	C
D	BNE r 2 2	CMP(d),y 2 5	CMP (d) 2•5	CMP(sr),y 2*7	PEI s 2•6	CMP d,x 2 4	DEC d,x 2 6	CMP(dl),y 2*6	CLD imp 1 2	CMP a,y 3 4	PHX s 1•3	STP imp 1*3	JML (a) 3•6	CMP a,x 3 4	DEC a,x 3 7	CMP al,x 4*5	D
E	CPX imm 2 2	SBC(d,x) 2 6	SEP imm 2*3	SBC sr 2*4	LPX d 2 3	SBC d 2 3	INC d 2 5	SBC(dl) 2 6	INX imp 1 2	SBC imm 2 2	NOP imp 1 2	XBA imp 1*3	CPX a 3 4	SBC a 3 4	INC a 3 6	SBC al 4*5	E
F	BEQ r 2 2	SBC(d),y 2 5	SBC (d) 2•5	SBC(sr),y 2*7	PEA s 3*5	SBC d,x 2 4	INC d,x 2 6	SBC(dl),y 2*6	SED imp 1 2	SBC a,y 3 4	PLX s 1•4	XCE imp 1*2	JSR(a,x) 3•6	SBC a,x 3 4	INC a,x 3 7	SBC al,x 4*5	F

*New W65SC816 Op Codes

• W65SC02 Op Codes

Op Code Matrix Legend

INSTRUCTION MNEMONIC	(COMMENT)	ADDRESSING MODE
BASE NO. BYTES		BASE NO. CYCLES

Note A complete assembler syntax description is available upon request. The final data sheet will contain the assembler syntax mnemonics.

THE SYM-1/VIC-1541 CONNECTION

Part of the reason for the lateness of this final issue is the fun we were having with our Commodore Systems. We justified our lateness, to ourselves, at least, by telling ourselves that we must delay publication until the SYM-1/VIC-1541 Connection was available. Fortunately, Ron Jordan advised us that we should be getting the prototype for test and announcement this week. We thus hurriedly finished the first 48 pages today, so that they could go to the printer immmediately, and will publish Ron's material as a four page "quick-printed" supplement as soon as it arrives!

THE SYM-1/VIC 1541 CONNECTION

Our long awaited SYM-1/VIC 1541 Connection Package arrived today. We haven't yet tried it because the object code as supplied (at $7000) conflicts with both our CODOS and FODS systems. Ron is using FDC-1 at $9000, so he does not have the conflict. Since Ron supplied us the RAE source, we'll relocate the code to $9000, and install it first on our CODOS system to download all our MTU graphics. Then we'll move it to our FODS/FDC-1 system at both $7000 and $9000, so that it can co-operate with either of these systems. Imagine, a triple-DOS SYM! The unit merely plugs on the A-connector, with an additional lead to the RST line, and the 1541 Drive cord just plugs directly into the unit. How simple, how elegant, how easily transportable, how inexpensive (the 1541 drives are now selling for less than $250 in the US)! Ron has been working closely with Don Lewis, who has developed an AIM-65 version of the system; you can contact Jordan and Associates for either the SYM or AIM versions.

SYM-1 DISK OPERATING SYSTEM FOR THE COMMODORE

1541 DISK DRIVE

MONITOR LINKS

RAE LINKS

BAS LINKS

Distributed by

JORDAN & ASSOCIATES

2611 Madrono Drive

Ann Arbor, MI 48103

INTRODUCTION

The SYM-1 DOS for the Commodore 1541 disk drive greatly expands the capability and compatibility of the SYM-1. Although several disk systems are available for the SYM-1, all are relatively expensive. In addition, each offers its own unique disk formating, which prevents disk interchangeability and greatly limits access to commercial and public software. The SYM-1 1541 DOS helps to fill this gap by using the Commodore 1541 disk drive to create Commodore compatible disks. Since the Commodore 1541 has the DOS built into it, the SYM-1 DOS can take advantage of the Commodore DOS features and reside in RAM or EPROM very compactly (approx. 2K). With the installation of SYM-1 1541 DOS the SYM-1 can become a much more powerful little computer that is easier and more enjoyable to use.

Functionally, the SYM-1 1541 DOS consists of four modules: the primitive routines, the MONITOR link, the RAE link, and the BAS link. The primitives include all of the low level routines needed to communicate with the Commodore 1541 disk drive over the serial bus. The SYM-1 has several different VIA ports that could be connected to the serial bus. However, the primitive interface routines are dependent on the selected bus configuration on the VIA. The standard VIA port configuration uses VIA #1 (Port A) on the A-connector. Other configurations are available upon request at a nominal fee. The MONITOR link interfaces with

SUPERMON. All commands are vectored through the unrecognized syntax vector (URSVEC) and may be easily enhanced or altered as desired. The commands include load and save memory to disk with the option for a relocated load. Other commands allow easy display of the disk directory, reading the error channel, changing the device number to another drive, and sending Commodore 1541 DOS commands. The assembler editor (RAE) link includes the monitor disk commands which are implemented through the DC command. The load and save commands use special forms of the PUT and GET commands. The load command will load RAE source files with the option for an append and the save will save the source files. Files may also be assembled from disk. To enter RAE, a simple monitor jump command is used which then completely configures the file parameters for a 28K (or whatever desired) system. The monitor may be reentered with a control C and all of the monitor commands are still available. To start BASIC a simple monitor jump command is also used, which configures BASIC for a 28K system with 80 columns and then patches in the new command processor using INVEC and OUTVEC. The disk commands are implemented through OUTVEC so that future commands may be added easily and used under program control. Examples might be OPEN and CLOSE commands which could enable writing data to disk. Numerous enhancements and utilities for all links will be available (see Utilities and Enhancements). Currently, BASIC load and save to disk commands are supported. The other disk commands are also available in BASIC. BASIC may be exited with a control C and then may be warm started with .G without the loss of the BASIC text. Normal cassette I/O is functional in BASIC, RAE, and the monitor. With some precautions, the SYM-1 1541 DOS can function concurrently with the FDC-1 disk system.

The SYM-1 1541 DOS system includes the following:

1. Hardware interface module for the serial bus connection to the SYM-1. VIA #1, Port A. (optional configurations available)

2. Complete source listing for the primitives, MONITOR, RAE, and BAS links with Cross Referenced Label Listing

3. Cassette tape with object code. (normal start address $7000, but others available at no charge)

4. SYM-1 1541 DOS manual.

5. EPROM with object code for primitives and MONITOR links. (optional)

6. Source files on disk or cassette. (optional)

COMMAND SUMMARY

MONITOR LINK:

1. S2 xxxx,yyyy/FILENAME
   save memory to disk with the name
2. L2 /FILENAME load memory
   L2 xxxx/FILENAME relocated memory load
3. SC #x change device number
   SC ! read error channel
   SC ? list directory
   SC .DISKCOMMAND send disk command
4. J0 cold start BASIC
   J5 cold start RAE

RAE LINK:

1. PUT/FILENAME save source file
2. GET/FILENAME load source file
   GET/FILENAME A append to source file
3. DC #x change device number
   DC ! read error channel
   DC ? list directory
   DC .DISKCOMMAND send disk command
4. CT FILENAME continue on disk

BAS LINK:

1. CONTROL C exit to monitor
2. #SP "FILENAME" save program to disk
3. #LP "FILENAME" load program from disk
4. #DC "#x" (same as RAE LINK)
   #DC "!"
   #DC "?"
   #DC ".DISKCOMMAND"

UTILITIES AND ENHANCEMENTS

The basic SYM-1 1541 DOS provides the foundation on which future commands may be added. Several commands have already been written, such as append BASIC programs, a RUN command for BASIC to load and run a program, and OPEN and CLOSE commands to write data to disk. Some utilities are also available such as a disk copy program and a disk sector read/write program, but many more are planned. It is hoped many new enhancements and utilities may be provided as they become available at a very reasonable cost.

All prices include shipping and handling unless otherwise stated. Please allow 4-6 weeks for delivery. Overseas orders add $10.00.

(1) SYM-1541 DOS $95.00
(2) DOS - Special I/O config. (add $25.00)
(3) EPROM option (add $15.00)
(4) Source files on disk or cassette (add $25.00)

Address mail orders to the address above. For additional information, telephone on weekdays, 6:00 PM-9:00 PM EST, or weekends, 9:00 AM-6:00 PM EST, at (313) 663-6374.

A FINAL MESSAGE FROM DICK ALBERS

March 5,1984

Dear Lux,

I hope this reaches you before publication of the last issue of SYM-PHYSIS (SYM-PHYZZLE ?). I have been meaning to write for a long time now, but must plead preoccupation; I have been learning a new system - Radio Shack's Color Computer with the FLEX and OS-9 DOSes. It is an easy way to use up much more time than I really should. I recommend the CoCo to anyone who wants experience with a powerful operating system without expensive or complex hardware.

Thank you for publishing my programs. The latest ones in issue 15 caused Phil Kohl to suspect an ommission (see #16-39 bottom of page). I had the pleasure of visiting his home late last year (1983) and we discussed that (and many,many other things). Phil has modified MON on his SYM; we assume that is the reason he had trouble. It does point out a possible pitfall: if you modify a system, be prepared for incompatibilities.

The programs have an unusual history. Way back when, before I got RAE, I needed an editor and attempted to write my own. Although it was never finished, I learned a lot about writing programs. One of the things I learned was that hand assembly is hard and frustrating work. Efficient code is a must; the fewer bytes to key in by hand, the better.

That's where the CMP #$0D went; it's included in INCHR so I didn't put it in my code. Another lesson was that an assembler is a necessity for

serious programming. After I had RAE I didn't need my own editor, so work on it ceased. However, I couldn't have all that time and effort wasted so I salvaged these two subroutines and converted them to stand alone. They are very handy when I need their functions.

I have a tip for other programmers. Some time ago you published an example of RAE's conditional assembly capabilities. I can't find it in my library so I must rely on my memory and I can't refer to the issue, but I think it was written by our guru, Lux. This tip could be considered "self-defensive programming".

This should work with assemblers other than RAE, so it may be useful on other systems too. The source for a program may be assembled into two or more versions depending on the value of specific flags. The problem is remembering to properly set those flags before waiting a considerable length of time for an assembly to complete, only to find that Murphy has struck again.

Assume a program that can be assembled for either tape or disk but not both, and if neither flag is set the result is incomplete. In your permanent source leave both flags clear and include the first two of the conditionals shown below:

```
0100          TAPE    .DE 0      ;Tape flag
0110          DISK    .DE 0      ;Disk flag
0130
0140          IFE     TAPE+DISK ;Test for both clear
0150ERROR NEED TAPE OR DISK FLAG SET
0160          ***
0170          IFN     TAPE+DISK-1 ;Test for both set
0180ERROR TAPE AND DISK FLAGS BOTH SET
0190          ***
0200          IFN     TAPE
0210;Tape version unique source
0220          ***
0230          IFN     DISK
0240;Disk version unique source
0250          ***
0260;Source common to both versions
0270          .EN
```

A flag-setting error will cause RAE to attempt to assemble the meaningless "code" within one of the conditionals, which will cause an error message (even if listing is turned off with .LC) and assembly will stop early. This example used only two flags so only two Boolean tests are required; more flags and more complex relationships between them will require more complex tests.

Did you know that a "^Z" (control-Z) will work in place of "//" to exit RAE's auto-number mode? It will work anywhere on the input line, not just at the beginning, and any text on that line is ignored.

Finally I want to thank you and Jean for forming and continuing S.U.G. and all those who contributed their efforts. Without SYM-PHYSIS I would not have been able to learn nearly as much about programing and certainly would not have had as much fun doing so. Thank you all.

Very sincerely,
Richard Albers

AND ONE FROM JEAN

Even though this is our last published contact with the SYM users community, I hope it will not be our last contact. From the very first day that Lux and I began this (ad)venture, we have made many hundreds of not just customers, but friends. Thanks to all of you, and especially to so many who very quickly turned an ordinary business call into a friendly personal one. It has been lots of fun -- let's keep in touch.

As ever, SYM-cerely yours,

Jean