-Punch along this edge for insertion into a binder-

THE SYM USERS' GROUP OFFICE WILL BE STAFFED ONLY FROM 10:00 AM to

4:00 PM on M-W-F FROM APRIL 7 TO MAY 15, 1982

SYM-PHYSIS

THE SYM-1 USERS' GROUP NEWSLETTER

VOLUME III, NUMBER 1 (ISSUE NO. 11) - SPRING 1982 (JAN/FEB/MAR)

SYM-PHYSIS is a quarterly publication of the SYM-1 Users' Group, P. O. Box 319, Chico, CA 95927. SYM-PHYSIS and the SYM-1 Users' Group (SUG) are in no way associated with Synertek Systems Corporation (SSC), and SSC has no responsibility for the contents of SYM-PHYSIS. SYM is a registered trademark of SSC. SYM-PHYSIS, from the Greek, means the state of growing together, to make grow, to bring forth.

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

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

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

BACK ISSUES ARE STILL AVAILABLE AS FOLLOWS:

Issue 0, the Introductory Issue (1979), and Issues 1 through 6 (Volume I, 1980), are available, as a package, for $12.00, US/Canada, and $1600, 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.

ON LATE NEWSLETTERS

We received today, in mid-March 1982, our copy of "THE TARGET - an AIM65 newsletter" for July/December 1981. We get lots of SYM applicable ideas from Donald Clem's (R.R. #2, Conant Road, Spencerville, OH 45887) newsletter, just finishing its third year. It is a bi-monthly, so this was actually a triple issue, covering July/August, September/October, and November/December of 1981. So you see, SYM-PHYSIS is not really "later than you think". We are merely conforming with computer newsletter tradition! Speaking seriously, though, now that we have gotten "organized" to the point of using reviewers to help evaluate, debug, and polish submitted software, and volunteers to answer requests for help (see below), we should be able to meet the quarterly deadlines.

"HELP"

We apologize once again for not being able to answer all of your letters for help, and ask you to write again if your problems have not yet resolved themselves. We think that we will be able to provide faster response time in the future, even while we ourselves are traveling, or otherwise not available, through the following procedure:

If, and really only if, your requests for help are on separate sheets of paper from any other type of correspondence, clearly marked "HELP", and are accompanied by a self addressed, stamped (US only) return envelope, whoever opens the mail will be able to "batch" them and send the entire package to one of a number of SYMmers who have offered to provide such help. It would be unfair to ask these volunteers to also pay for your postage. Overseas reply postage costs can best be handled by enclosing low denomination local currency.

SYM DISK OPERATING SYSTEMS

The SYM-1, as it comes out of its box, is a 1K RAM, 4K ROM, single cassette based system, powering-up, and/or resetting, to SUPERMON. When fully loaded (no external expansion, but with the Blalock RAM Board or 2114 piggy-backing considered "internal"), it becomes an 8K RAM, 20K ROM, dual cassette based system, still resetting to SUPERMON, but with RAE and BASIC capabilities.

At this point all SYMs are essentially equal, and all software is fully transportable via physical cassette transfers. Most of the SYMmers with whom we have communicated have brought their cassette systems up to nearly 100% reliability (some at double and triple times the standard speed). We have managed to read every cassette we have received (even double speed ones) because good cassette practice includes making at least double dumps (we use triple dumps on distribution cassettes) to provide data redundancy in the event of any glitches caused by tape dropouts.

We are very much satisfied with the SYM cassette interface as the primary means for inter-SYM data and software interchange. With a 4K, or even an 8K system, the cassette interface provides an acceptable mass storage system. With expansion to 24K or 32K, and the concomitant longer files, cassettes become impractical, except for backup purposes (when we had only one SYM/FODS system we backed up our mailing lists on a second disk and triple cassette dumps!). Have you ever seen a 48K Apple II system without at least a single Disk II beside it?

Note that we said Disk II; we emphasize this, because all Apple IIs use the same (or wholly compatible) controllers, drives and DOSes, thus ensuring full software transportability between Apples. It is this "universality" of software exchange that provides a large market for software entrepreneurs, thereby encouraging the development of good (and bad!) software for the Apple. Furthermore this software is distributed mainly on diskettes.

We are now too far downstream with the SYM for a universal DOS (Disk Operating System) to evolve, and perhaps this is for the best, after all. We are free to chose any combination of hardware and software that matches our needs, subject, of course, to our financial abilities. In the following paragraphs, we will briefly describe some of the disk systems now available for SYM, but first presenting some preliminary background information on drives and disks in general.

GENERAL

For "personal" use, especially for the type of research and report preparation we do, we prefer the 5 1/4 inch drive systems because they are quieter, more compact, and cheaper than 8 inch drive systems. Where the noise, size, and cost are not important factors, the greater access speed and on-line storage capacity of the 8 inch drives are really nice to have, and in some applications, even these might be inadequate. That's why hard disks are becoming so popular!

The choice of drive size is yours alone to make, as is the choice of make and model. The major differences between the various brands appear to be in the speeds at which the heads are loaded against the disk and the rates at which the heads are stepped from track to track. The controller software has built-in delays to accomodate the slowest available drives. If you use one of the faster drives it is well worth your time to customize the software to it. Disk load times can be speeded up by as much as a factor of five times. While most suppliers of DOSes guard their source codes as if they were were divine mysteries, a disassembly and study of that part of the object code containing the

(continued to page 11-36)

A 40K SYM-1 MEMORY EXPANSION BY GEORGE WELLS

Here is a memory expansion scheme for the SYM-1 that has the following features:

1. 40K of RAM continuous from $0000 to $9FFF.
2. Top three 1K groups of RAM ($9400-$9FFF) are write protectable.
3. EPROM or ROM can overlay RAM from $1000 to $8FFF with automatic switching between them.

The components used in this arrangement would typically be:

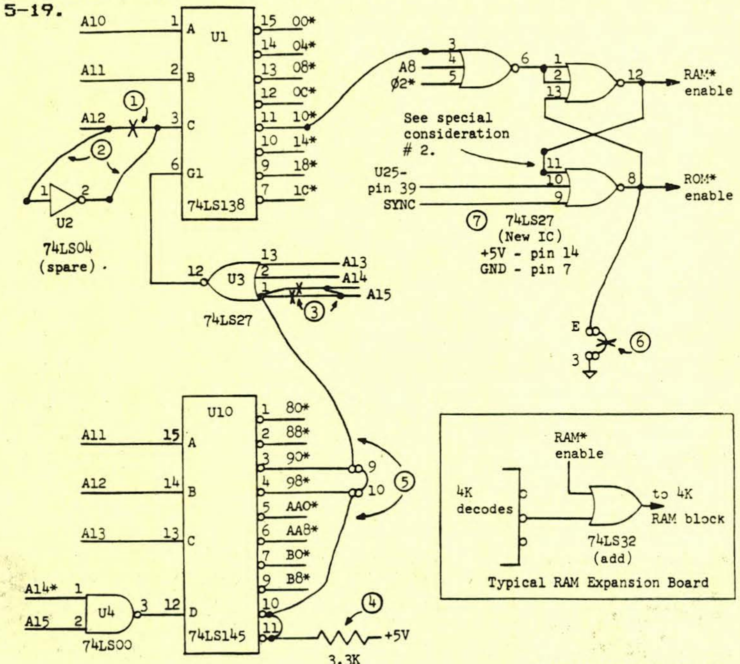
1. 4K on-board static RAM with modified decoding to appear at $9000-$9FFF.
2. Blalock's 4K static RAM expansion with modified decoding to appear at $0000-$0FFF.
3. 32K dynamic RAM at $1000 to $8FFF (available from several sources).
4. Monitor ROM at $8000 to $8FFF (normal).
5. Additional EPROMs, as desired, between $1000 and $7FFF.
6. One IC wired to automatically switch between the RAM and ROM banks.

Anyone attempting to implement this idea should thoroughly understand it before beginning. The procedure given assumes that you can find the various signals on your PC boards and that you have some knowledge of logic design.

Before starting, the address space between $0000 and $9FFF should be clear of all memory and I/O except for the original 4K RAM, the Blalock 4K RAM, and the System Monitor ROM. Also, verify correct operation of the Write Protect feature as described in the SYM-1 Reference Manual, pages 4-26 and 5-19.

For the following steps in modifying the SYM-1 board, refer to the schematic. The first two steps move the write protectable memory to the top of the 8K RAM space.

(If you have only 8K of RAM, you can perform just these first two steps now, and the remainder when you get 32K more.)



(1) Cut the A12 trace leading to pin 3 of U1 on the bottom side of the board.

(2) Insert the spare inverter by adding two wires to the bottom of the board as shown. Make sure U2-pin 2 goes to U1-pin 3.

At this point, you should again verify correct operation of the Write Protect function, but this time the three 1K groups are $1400-$17FF, $1800-$1BFF, and $1C00-$1FFF. To write protect the last 1K of RAM, it is only necessary to enter W 1 (instead of W 001).

The next three steps move the 4K block of RAM currently at $1000-$1FFF to $9000-$9FFF. This includes the write protectable RAM. The Blalock RAM stays at $0000-$0FFF.

(3) On the bottom of the board, cut the two traces leading to U3-pin 1 and join them with wire, leaving pin 1 out of the connection.

(4) On the top of the board, install a 3.3K pull-up resistor from pins 10 and 11 of U10 to any convenient +5V source.

At this point, the previously unused outputs of decoder U10 will go low any time an address block beginning with $0, $4, or $C is accessed.

(5) On the bottom of the board, continue wiring the pull-up to jumper pads 9 and 10 and then to U3-pin 1. Make sure all other jumpers to pads 9 and 10 are removed.

Now, pin 1 of U3 will go low for block $0, $4, $9, or $C; however, when block $4 or $C is accessed, pin 2 (A14) will be high. Therefore, pin 12 of U3 will go high (enabling the RAM decoder U1) only for blocks $0 and $9. Test to see that you do indeed have RAM at these two blocks and that you can write protect the three 1K groups at $9400-$97FF, $9800-$9BFF, and $9C00-$9FFF.

If everything works correctly, you are ready to add the RAM/ROM bank selector.

(6) Remove the ground jumper between pads E and 3 on the SYM-1.

(7) Wire up the IC as shown in the schematic. Don't forget the power and ground connections.

[Note - Asterisks, "*", indicate barred signals. For example, pin 5 of the new IC must go to the clock signal that is brought out on pin Y of the expansion connector--not pin U.]

One method of adding this IC to the SYM-1 is to cement it to the board with its pins facing up and solder wires directly to the pins. Depending on how you use the bank selection to switch between the 32K RAM expansion and EPROM, you will need to bring one or both of the RAM* and ROM* Enable signals off the SYM-1 board to the expansion board(s). Both of these are active low signals.

Some memory boards have bank switching capabilities built into them which makes the interface simple, but almost all boards provide jumper decoding in 4K blocks which allow the addition of an OR-gate to provide the required gating for each 4K block of RAM (see schematic). It is not necessary to switch the entire 32K of RAM--only those blocks which share the same address space with ROM or EPROM. Until you add bank switched EPROM, the only RAM block you will need to switch is block $8 ($8000 to $8FFF). Most EPROM expansion boards provide several jumpers to select the EPROM type and address decoding. For a 2716, pins 18 and 20 are usually tied together to the address decode. In such a case, bank selection is easily performed by disconnecting pin 20 from pin 18 and tying it instead to the ROM* Enable signal.

Theory of Operation

The key to understanding how the bank selector works is in realizing that the only way a 6502 instruction in ROM can have access to the entire address space is through one of the indirect addressing modes which reads page zero just before accessing the desired memory location. (BASIC also accesses memory through an absolute mode instruction which fortunately was copied to page zero RAM.) Remember that BASIC does not actually execute the "program" in RAM--that is really "data" for the interpreter program in ROM. The same is true for RAE and the Monitor: they all access the RAM indirectly, treating it as "data". The Monitor also needs to access itself, but it always does this through absolute mode instructions (again--fortunately!).

Thus, all that is needed to switch between the RAM and ROM banks is a flip-flop which is set one way (to enable ROM) whenever an op-code fetch occurs (SYNC goes high) and set the other way (to enable RAM) whenever any access to page zero is made. The circuit detects page zero whenever the pins 3, 4, and 5 of the new IC are all low. Pin 3 goes to U1-pin 11 which is labeled "10*" but is really "00*" because of the changes made in steps 1 and 2. Actually, to make the hardware a little simpler, the circuit also detects page two, which is of little consequence as long as page one is excluded. (This is important so that JSRs in the Monitor or EPROM will work.) There is one other way that the ROM bank can get enabled and that is with the software-controlled Power On Reset signal coming from the CA2 output on U25 (pin 39). Without this signal it would be possible for the RAM bank to be enabled while the 6502 RESET vector was being fetched causing the CPU to go to an unknown location and possibly modifying RAM.

Special Considerations

1. If you cold start to BASIC and let it figure out how much RAM you have, it will start writing $55 and then $AA to every memory address starting at $0200 and continuing until an address is reached which will not accept the $55 or $AA. If you have not write protected any RAM, then the memory test will continue up to address $A000 which is port B of VIA #1. If you have installed a second cassette control on bit 7 of this port (as per RAE requirements), then that cassette will become activated when you enter BASIC, just like it does when you enter RAE. If you have any other devices on this port, make sure they will not be damaged by BASIC's initialization.

2. Sometimes you may want to examine the contents of the Monitor ROM, but if you try you'll discover that what you are reading is the RAM--not the ROM. A simple way to disable the bank selector is to install a switch or jumper between pins 11 and 12 of the flip-flop. Opening this connection will force all indirect memory accesses to go to the ROM bank. If you need software control of this feature, you can instead tie pin 10 of the flip-flop to a different port bit output which your software can drive high to read the ROM. However, if you do this you will have to drive it low after every reset in order to enable the bank selector.

3. If you put any machine code on page zero or two of memory, remember that all fetches or stores to bank memory will access the RAM bank.

4. If you install EPROMs between addresses $1000 and $7FFF, they can only contain executable machine code and data that is accessed with absolute mode instructions. You cannot put the BASIC trig function expansion in this region since it contains data that is fetched indirectly by the BASIC interpreter. The best place to put an EPROM containing the BASIC trig functions is at $F000-$F7FF.

5. Do not try to bank switch the BASIC ROMs. They contain several data tables that are fetched indirectly.

6. You can bank select the two RAE ROMs, putting either RAM or EPROM in the "RAM" banks. However, as with all the other RAM banks, this memory can contain only data that is accessed indirectly. This would be an ideal place to put a disk buffer or video memory (for RAM), or character generator tables, or sound generation constants (for EPROM). You could also put some I/O in these regions as long as the programs that access them use indirect addressing. If you decide to implement any of these exotic expansions, you're on your own! Just make sure you know what you're doing.

Conclusion

Now you don't have to feel jealous of those other guys with their super bank switcher computers--you can have one too at a fraction of the cost. Furthermore, you can understand exactly how yours works. And it sure is nice to be able to sign on to BASIC and see it print, "40447 BYTES FREE"!

RECOMMENDED SYM-1 MODS

We "routinely" modify all of our personal SYMs, and those going into OEM systems, as follows (listed in priority order):

1. To improve READ performance - replace C16 with a 0.01 ufd disc cap.
2. To improve WRITE performance - replace R88 with a 1 K resistor.
3. To improve "From TTY Keyboard" performance (on 20 mA current loop) - install a 1 K resistor from the base of Q28 to ground.
4. To recover the use of PB6 of VIA No. 1 (for 8 bit D/A and A/D applications) - install a 1 Meg resistor from pin 3 of U26 to ground.

The word "improve" is used in the sense of increasing reliability when interfacing with external equipments, i. e., cassette recorders and current loop terminals.

TAPE TIP

There are often times when it is desirable to determine the ending address (EAD) of a file being read in with .L2. EAD is required when using the .L2 FF,SAD,EAD option; also, in cases where most, but not all of a file is readable, it may be helpful to know how much of the file has actually been read in up to the abort point. Find out this way:

After either a successful or an aborted .L2, enter the .M <cr> command and the address of the memory location whose contents are being displayed is EAD+1 or the address of the first non-read byte. Armed with this information, partial BAS-1 and RAE-1 files can be read in with .L2, "terminated" to match BAS-1 or RAE-1 protocol, and the proper pointers then set to permit at least partial recovery of otherwise "lost" material.

"DOUBLE-DECKING" THE SYM

The suggestions of "piggy-backing" 2114s to get 8K of on-board RAM, and the two RAE-1/2 chips (also the two BAS-1 chips, when BAS-1 came only in the two chip version) to get two ROMs into one socket, were reported on in early issues of SYM-PHYSIS. We also reported on the existence of three unused inverters, and gave their locations (in U2, U9, and U38).

We have added additional logic chips to several of our SYMs by glueing them, pins-up, to the board, and wiring them in as required. Joe Hobart's suggestion, in Issue No. 10, of piggy-backing a 74LS04 (hex inverter) over the 7408 (and gate) at U24, to pick up the +5 V and GND, started us to thinking about the following: What other double-decking possibilities might be useful?

Perhaps logic to provide full address decoding for the VIAs and the 6532 to allow more effective use of $A000-$AFFF and $F800-$FFF9? Actually, only the top 6 bytes of SYSRAM at $A67A-$A67F need be "echoed" at $FFFA-$FFFF. Full address decoding would permit installation of a 2716 EPROM (less six bytes) at $F800.

Your first impulse might be to suppress the SYSRAM echo altogether and put your choice of NMI and IRQ into a 2716 EPROM at $F800, or a 2732 at $F000. This is not too good an idea, however, since one rather "widely-used", "very well-known", (how's that for "one-upsmanship"!) programming technique for subroutine calling involves "calling" subroutines with BRK (not JSR!), and returning with RTI (not RTS!). Both FODS and CODOS use this technique very effectively, and SUPERMON returns from subroutine USRENT with an RTI, just as it would from a real BRK. Since this approach requires changing IRQVEC, IRQVEC should itself be in RAM or at least point to RAM, where changes can be made.

Other possibliities for chip piggy-backing include buffer/drivers, multiplexers, flipflops, or almost any TTL chip, for that matter. If any of your I/O subsystems require such chips, such as, for example, a 20 mA current loop to inverted TTL conversion, perhaps you could mount them directly on the SYM?????

NEW BOOK REVIEW

We have long recommended Lance A. Leventhal's "6502 Assembly Language Programming" as one of the two books which every serious SYMmer should have on hand (the other is Marvin De Jong's "Programming & Interfacing the 6502, With Experiments"). For 6809 SYMmers we have recommended Leventhal's "6809 Assembly Language Programming".

We now recommend a third book for the 6502ers: Lance A. Leventhal and Winthrop Saville's "6502 Assembly Language Subroutines", OSBORNE/McGraw-Hill, Berkeley, California, 1982. This nearly 550 page book is a veritable encyclopedia of both general programming concepts for the 6502 (with lots of carry-over to other systems) and specific subroutines, very thoroughly documented, for array manipulation, string processing, code conversion, bit manipulation, I/O, interrupt processing, etc.

The highest praise that I can give this book is to say that even after more than four years of using the 6502 on a nearly daily basis, I will now check with Leventhal and Saville first, before starting any major programming effort, to find the "right way" to do it. The information on common programming errors, and how to avoid them, will save enough in "wasted" development time to pay for the book many times over; the subroutines are given in a form that is immediately usable.

A BASIC VARIABLE CROSS REFERENCE LISTER

June 10, 1981

Dear Lux:

Enclosed is a copy of a program which searches a BASIC text file and picks out all the new variable names. It has three distinct parts; the text search, a sorter, and a printing segment.

The first part creates a file (starting at $3400) of 5-byte elements, one for each new variable it encounters in the text. The first byte is for the type of variable, the next two are the variable name, and the last two are the line number where the variable occurs in the text. I have chosen the type values so that the sorting routine will put simple real variables first, string variables second, and on through to subscripted integer variables last. All characters in a variable name after the first two are ignored; if it is a one-character variable a space is substituted for the second character. The line number bytes are copied directly from the text file.

New variable names can be introduced in BASIC only in certain ways; they may be the first word of a statement, or they can occur only after the reserved words DIM, FOR, INPUT, LET, READ, and DEF FN (and in some versions after GET). Therefore, in this program all other occurrences of variable names are ignored.

The sorting segment is a (more or less) standard bubble sort which sorts the list in place.

The printing segment has two counters which I have set for my system, but they should probably be changed for others. These are: (1) the maximum numbers of line numbers for a given variable printed for each line of output, and (2) the number of lines of output per page on the terminal screen. I have set these numbers at 8 and 15 respectively. The latter feature was included to allow time to study the list of variables before it disappears off the screen. Hitting any key causes the printout to continue.

The program ends with a simple RTS which works fine if it is run with a .G 3000 out of SYM MON 1.1. Care must be taken if the program is called in a way such that the return address is not stacked.

The program could be modified for other 6502 systems by making the appropriate changes for the reserved-word tokens and adding steps to recognize other reserved words. Also the addresses of the BASIC routines and the MON 1.1 subroutines must be made correct for other systems.

Best Regards,

/s/ Jim Pengra

21 February 1982

Dear Jim:

Finally getting around to going through the backlog of tapes and cassettes, after all these months. Tried the program, found one bug, and have several suggstions.

The bug is that, while it works fine when called from MON, and does what it should when called from BASIC with X=USR(ORIGIN,0) it returns to BASIC with a ?TM ERROR message. Didn't have time to track down the source of the bug, which is most likely traceable to "playing" with the pointers during the program and not restoring them prior to return. The ?TM ERROR message can be suppressed, however, by the ad hoc trick of calling the program with a string variable name, e. g., X$=USR(ORIGIN, 0). While the return to BASIC with an RTS is okay, I have gotten used to returning from USR calls with JMP $D14C, so I made that change in the program.

The suggestions (some posed as questions) are as follows:

(1) Rather than use space above the machine language program for temporary storage, why not use the space between the end of program space and top of memory?

(2) Instead of the format you use, how about one where you do not use the headings to indicate and separate variable types, but instead indicate the variable types by following their names with %, $, (), %(), $()? (For arrays, the number of subscripts, and perhaps even the dimensions could be indicated?????)

(3) The printing of eight line numbers per line is too many if four decimal digit line numbers are used, so I cut the maximum down to six.

(4) Since one of the reasons for a program like this is to give the user information on possible variable name conflicts, thus permitting renaming if necessary, shouldn't the listing include ALL occurrences of the name?

Am now using the version of your program listed below where the output format comes closer to complying with suggestion (2) above. Studying how your program works will give readers a good insight into how BASIC itself works!

- - - - Lux

```
0010 ;                    VARIABLE NAME FINDER
0020
0030 ;                            by
0040
0050 ;                     JAMES G. PENGRA
0060
0070 ;                   Physics  Department
0080 ;                     Whitman College
0090 ;                 Walla Walla, Washington
0100 ;                          99362
0110
0120
0130 ;THIS PROGRAM SEARCHES A BASIC TEXT FILE FOR
0140 ;VARIABLE NAMES, SORTS THE LIST BY TYPE (STRING,
0150 ;INTEGER, ETC.) AND ALPHABETICALLY, AND THEN PRINTS
0160 ;THE NAMES AND THEIR LINE NUMBERS. IT ALSO FINDS
0170 ;SOME ERRORS.
0180
0190
0200 ;  VARIOUS SYM MON 1.1 SUBROUTINES
0210
0220 CRLF       .DE $834D
0230 SPC2       .DE $833F
0240 OUTCHR     .DE $8A47
0250 INTCHR     .DE $8A58
0260 TECHO      .DE $A653
0270 ACCESS     .DE $8B86
0280
0290 ;  SYM BASIC ROUTINES AND ADDRESSES
0300
0310 FACTO      .DE $B2
0320 FLOATC     .DE $D9FF
0330 FOUT       .DE $DB9A
0340 CHRGET     .DE $CC
0350 CHRGOT     .DE $D2
0360 SEARCH     .DE $CC5D
0370 TEXT.PTR   .DE $C49F
0380 ALPHA      .DE $CEE9
0390 USRRET     .DE $D14C
0400 TXTPTR     .DE $D3
0410 DISSTK     .DE $66
0420 ADPTR1     .DE $72
0430
0440 ;  OTHER COUNTERS AND VECTOR LOCATIONS
0450
0460 COUNT      .DE $E8
0470 P0         .DE $EA
0480 P1         .DE $EE
0490 BASE       .DE $EC
0500 BUFF       .DE $61
0510
0520 ORIGIN     .DE $3000
0530 HOLD       .DE ORIGIN+$0400
0540
```



```
                0550            .BA ORIGIN
                0560            .MC $9000
                0570            .OS
                0580
3000- 20 9F C4  0590 START      JSR TEXT.PTR   ;SET POINTER FOR CHRGET/GOT
3003- 20 0A 31  0600            JSR SET.PTRS   ;SET OTHER PTRS
3006- A0 02     0610 LINLINK    LDY #02        ;CHECK LINE LINK HI,
3008- B1 D3     0620            LDA (TXTPTR),Y        ;IF ZERO, THEN
300A- D0 03     0630            BNE CONT
300C- 4C 24 31  0640            JMP DONE       ;END OF SEARCH
                0650 ;
300F- C8        0660 CONT       INY
3010- B1 D3     0670            LDA (TXTPTR),Y        ;STORE LINE NUMB
3012- 85 64     0680            STA *BUFF+3    ;IN BUFFER
3014- C8        0690            INY
3015- B1 D3     0700            LDA (TXTPTR),Y
3017- 85 65     0710            STA *BUFF+4
3019- E6 D3     0720 INCR       INC *TXTPTR    ;MOVE CHRGET/GOT
301B- D0 02     0730            BNE DINC       ;POINTER THRU
301D- E6 D4     0740            INC *TXTPTR+1         ;LINKAGE AND LINE
301F- 88        0750 DINC       DEY            ;NUMBER BYTES
3020- D0 F7     0760            BNE INCR
3022- 20 CC 00  0770 ONE        JSR CHRGET     ;GET NEXT CHAR
3025- C9 80     0780            CMP #$80       ;IF < $80 IT'S A NAME
3027- 90 7D     0790            BCC NAME       ;OTHERWISE, CHECK FOR TOKENS WHICH
3029- C9 81     0800            CMP #$81       ;'FOR'   PRECEDE NEW VARIABLES
302B- F0 F5     0810            BEQ ONE
302D- C9 84     0820            CMP #$84       ;'INPUT'
302F- F0 4D     0830            BEQ BEE
3031- C9 85     0840            CMP #$85       ;'DIM'
3033- F0 ED     0850            BEQ ONE
3035- C9 86     0860            CMP #$86       ;'READ'
3037- F0 E9     0870            BEQ ONE
3039- C9 A7     0880            CMP #$A7       ;'LET'
303B- F0 E5     0890            BEQ ONE
303D- C9 95     0900            CMP #$95       ;'DEF'
303F- D0 09     0910            BNE DEE
3041- 20 CC 00  0920            JSR CHRGET     ;CHECK FOR REST OF 'DEF FN'
3044- C9 9F     0930            CMP #$9F       ;'FN'
3046- F0 DA     0940            BEQ ONE
3048- D0 4B     0950            BNE ERR
                0960 ;
304A- 20 CC 00  0970 DEE        JSR CHRGET     ;CONTINUE UNTIL FIND
304D- C9 3A     0980 CMP3A      CMP #':        ;END OF STATEMENT
304F- F0 D1     0990            BEQ ONE
3051- C9 00     1000            CMP #00        ;OR END OF LINE
3053- F0 B1     1010            BEQ LINLINK
3055- C9 22     1020            CMP #'"        ;QUOTES MAY CONTAIN COLONS
3057- D0 F1     1030            BNE DEE
3059- 20 8A 30  1040            JSR QUOTES
305C- D0 EF     1050            BNE CMP3A      ;ALWAYS
                1060 ;
305E- 20 D2 00  1070 AAY        JSR CHRGOT     ;FIND OUT HOW VAR NAME ENDED
3061- F0 EA     1080 THREE      BEQ CMP3A      ;WAS IT END OF LINE OR STATEMENT?
3063- C9 AC     1090            CMP #$AC       ;WAS IT AN '='?
3065- F0 E3     1100            BEQ DEE
3067- C9 2C     1110            CMP #',        ;MORE VARS IN THIS STATEMENT?
3069- F0 B7     1120            BEQ ONE
306B- C9 28     1130            CMP #'(        ;IF SUBSCR'P'D THEN FIND END OF IT
306D- F0 02     1140            BEQ FIN
306F- D0 24     1150            BNE ERR        ;IF NONE OF THESE, THEN ERROR
                1160 ;
3071- 20 CC 00  1170 FIN        JSR CHRGET     ;FIND END OF SUBSCR'P'D VAR
3074- C9 29     1180            CMP #')
3076- D0 F9     1190            BNE FIN
```

```
3078- 20 CC 00            JSR CHRGET    ;AND SEE WHAT'S NEXT
307B- 4C 61 30            JMP THREE
                  ;
307E- 20 CC 00   BEE      JSR CHRGET    ;ANY MESSAGE IN 'INPUT' STATEMENT?
3081- C9 22               CMP #'"
3083- D0 21               BNE NAME      ;IF NOT, GET VAR NAME
3085- 20 8A 30            JSR QUOTES    ;IF SO FIND END
3088- D0 F4               BNE BEE       ;ALWAYS, TO SKIP ';' AFTER STRING
                  ;
308A- 20 5D CC   QUOTES   JSR SEARCH    ;MOVE THRU STRING AND GET
308D- 20 D2 00            JSR CHRGOT    ;CHAR AFTER CLOSING QUOTE
3090- A2 69               LDX #$69      ;RESET DESCRIPTOR
3092- 86 66               STX *DISSTK   ;STACK PTR
3094- 60                  RTS
                  ;
3095- A0 38      ERR      LDY #M5-M0    ;SEND ERROR MESSAGE
3097- 20 7F 32            JSR MESSAGE
309A- A9 63               LDA #$63      ;SET BASE PTR TO LINE NUMB
309C- 85 EC               STA *BASE     ;POSITION IN BUFFER - 1
309E- 84 ED               STY *BASE+1   ;Y=0
30A0- 20 52 32            JSR PR.LINUM
30A3- 4C 4C D1            JMP USRRET    ;TERM PROGRAM
                  ;
30A6- A2 02      NAME     LDX #02       ;ASSUME REAL, NONSUBSCRIPTED VAR
30A8- 86 61               STX *BUFF     ;1ST POS IN BUFF IS FOR 'TYPE'
30AA- A2 00               LDX #00
30AC- 86 63               STX *BUFF+2   ;CLEAR  2ND CHAR IN NAME BUFF
30AE- 20 E9 CE            JSR ALPHA     ;IS FIRST CHAR A LETTER?
30B1- 90 E2               BCC ERR       ;ERROR IF NOT
30B3- 95 62      STBUFF   STA *BUFF+1,X         ;IF SO STORE IT
30B5- E8                  INX
30B6- 20 CC 00   GET      JSR CHRGET    ;GET NEXT CHAR
30B9- F0 34               BEQ STORE     ;Z=1 IF CHAR IS 0 OR ':'
30BB- C9 AC               CMP #$AC      ;'=' TOKEN MEANS END OF VAR
30BD- F0 30               BEQ STORE
30BF- C9 30               CMP #'0       ;IF CHAR >= '0' THEN
30C1- B0 06               BCS CHKX      ;SEE IF NAME IS > 2 CHARS LONG
30C3- C9 2C               CMP #',       ;A COMMA?
30C5- F0 28               BEQ STORE     ;THEN STORE
30C7- D0 06               BNE AG        ;IF ANYTHING ELSE, CHECK VAR TYPE
30C9- E0 02      CHKX     CPX #02
30CB- 90 E6               BCC STBUFF    ;IF X<2, STORE CHAR IN BUFF
30CD- B0 E7               BCS GET       ;ELSE CONT
30CF- C9 24      AG       CMP #'$       ;A STRING VAR?
30D1- F0 10               BEQ STRING
30D3- C9 25               CMP #'%       ;INTEGER VAR?
30D5- F0 14               BEQ INTEGER
30D7- C9 28               CMP #'(       ;SUBSCRIPTED?
30D9- D0 BA               BNE ERR       ;IF NOT, THEN THERE'S AN ERROR
30DB- A9 01               LDA #01       ;SET 0-BIT FOR
30DD- 05 61               ORA *BUFF     ;SUBSCRIPTED VAR
30DF- 85 61               STA *BUFF
30E1- D0 0C               BNE STORE     ;ALWAYS
                  ;
30E3- A9 10      STRING   LDA #$10      ;SET 4-BIT FOR
30E5- 05 61      OR       ORA *BUFF     ;STRING VARS
30E7- 85 61               STA *BUFF
30E9- D0 CB               BNE GET       ;ALWAYS
                  ;
30EB- A9 80      INTEGER  LDA #$80      ;SET 7-BIT FOR INTEGER VAR
30ED- D0 F6               BNE OR        ;ALWAYS
                  ;
30EF- A0 04      STORE    LDY #04       ;STORE 5 BYTES FROM BUFF
30F1- B1 EE      LOADS    LDA (P1),Y
30F3- 91 EC               STA (BASE),Y  ;IN VAR ARRAY
```



```
30F5- 88                  DEY
30F6- 10 F9               BPL LOADS
30F8- 20 FE 30            JSR INBAS5
30FB- 4C 5E 30            JMP AAY
                  ;
30FE- 18         INBAS5   CLC               ;INCR BASE PTR BY 5
30FF- A5 EC               LDA *BASE
3101- 69 05               ADC #05
3103- 85 EC               STA *BASE
3105- 90 02               BCC OUTS
3107- E6 ED               INC *BASE+1
3109- 60         OUTS     RTS
                  ;
310A- A9 34      SET.PTRS LDA #H,HOLD       ;SET POINTERS - ALL NOT USED
310C- 85 EB               STA *P0+1         ;IN EVERY SECTION OF PROGRAM
310E- 85 73               STA *ADPTR1+1
3110- 85 ED               STA *BASE+1
3112- A9 61               LDA #L,BUFF       ;USE P1 TO
3114- 85 EE               STA *P1           ;POINT TO SORT BUFFER
3116- A9 05               LDA #05
3118- 85 72               STA *ADPTR1
311A- A9 00               LDA #00
311C- A8                  TAY
311D- 85 EA               STA *P0
311F- 85 EF               STA *P1+1
3121- 85 EC               STA *BASE
3123- 60                  RTS
                  ;
                  ;       * * * BUBBLE SORT * * *
3124- A8         DONE     TAY               ;ZERO Y
3125- 91 EC               STA (BASE),Y      ;END LIST WITH 0 TO STOP SORT
3127- 20 0A 31            JSR SET.PTRS      ;RESET POINTERS
312A- A9 0A               LDA #10
312C- 85 EC               STA *BASE         ;BASE NOW USED AS UPPER SORT PTR
312E- B1 EA               LDA (P0),Y        ;CHECK 1ST BYTE OF VAR ARRAY
3130- D0 03               BNE NEXT2
3132- 4C 4C D1            JMP USRRET        ;NO VARS FOUND, END PROG
                  ;
3135- A0 00      NEXT2    LDY #00
3137- B1 72               LDA (ADPTR1),Y
3139- D0 03               BNE SAVEBUF       ;IF NO MORE VARS THEN
313B- 4C A4 31            JMP PRINT         ;GO TO PRINTOUT
                  ;
313E- A0 04      SAVEBUF  LDY #04
3140- B1 72      LOADB    LDA (ADPTR1),Y          ;PUT HIGHER VAR IN
3142- 91 EE               STA (P1),Y        ;BUFFER
3144- 88                  DEY
3145- 10 F9               BPL LOADB
3147- A0 00      ZEROY    LDY #00
3149- B1 EE      CPCHAR   LDA (P1),Y        ;THEN COMPARE IT TO LOWER ONE
314B- D1 EA               CMP (P0),Y
314D- 90 25               BCC EXCH          ;EXCHANGE IF (P0) > (ADPTR1)
314F- D0 05               BNE STORBUFF      ;IF CHARS EQUAL,
3151- C8                  INY
3152- C0 03               CPY #03
3154- 90 F3               BCC CPCHAR        ;KEEP CHECKING, OTHERWISE
3156- A0 04      STORBUFF LDY #04           ;PUT
3158- B1 EE      LOADT    LDA (P1),Y        ;CONTENTS OF BUFFER
315A- 91 72               STA (ADPTR1),Y          ;IN ADPTR1 SPACE
315C- 88                  DEY
315D- 10 F9               BPL LOADT
315F- A5 ED      PTRS     LDA *BASE+1       ;ADVANCE PTRS BY 5
3161- 85 EB               STA *P0+1
```

```
3163- 85 73                    STA *ADPTR1+1
3165- A5 EC                    LDA *BASE
3167- 85 72                    STA *ADPTR1
3169- 85 EA                    STA *PØ
316B- 20 8B 31                 JSR DEPØ5
316E- 20 FE 30                 JSR INBASS5
3171- 4C 35 31                 JMP NEXT2        ;AND SHUFFLE AGAIN
                   ;
3174- A0 04        EXCH        LDY #Ø4          ;MOVE PØ VAR UP ONE SPACE
3176- B1 EA        LOADE       LDA (PØ),Y
3178- 91 72                    STA (ADPTR1),Y
317A- 88                       DEY
317B- 10 F9                    BPL LOADE
317D- A5 EA        PØ>ADPTR1   LDA *PØ          ;AND THEN DECR PTRS
317F- 85 72                    STA *ADPTR1
3181- A5 EB                    LDA *PØ+1
3183- 85 73                    STA *ADPTR1+1
3185- 20 8B 31                 JSR DEPØ5
3188- 4C 47 31                 JMP ZEROY
                   ;
318B- 38           DEPØ5       SEC              ;DECR PØ PTR BUT,
318C- A5 EA                    LDA *PØ          ;NOT BELOW
318E- E9 05                    SBC #Ø5          ;HOLD
3190- 85 EA                    STA *PØ
3192- B0 0F                    BCS OUTD
3194- A5 EB                    LDA *PØ+1
3196- E9 00                    SBC #ØØ
3198- C9 34                    CMP #H,HOLD
319A- B0 05                    BCS STORIT
319C- 68                       PLA              ;PULL RETURN OFF STACK,
319D- 68                       PLA              ;THERE'S NO MORE,
319E- 4C 56 31                 JMP STORBUFF     ;SO STORE BUFFER
                   ;
31A1- 85 EB        STORIT      STA *PØ+1
31A3- 60           OUTD        RTS
                   ;

                   ;    * * * PRINTOUT ROUTINE * * *

31A4- 20 0A 31     PRINT       JSR SET.PTRS     ;RESET POINTERS
31A7- A0 02                    LDY #Ø2          ;CLEAR BUFFER, USED FOR COMPARISON
31A9- 91 EE        ST          STA (P1),Y       ;OF VAR NAMES
31AB- 88                       DEY
31AC- 10 FB                    BPL ST
31AE- A9 0F                    LDA #15
31B0- 85 EA                    STA *PØ          ;PØ NOW USED AS I/O LINE COUNTER
31B2- A2 0E                    LDX #14          ;SEND 28 SPACES
31B4- 20 75 32                 JSR CRLF.SPS
31B7- C8                       INY              ;WAS $FF, WANT Ø
31B8- 20 7F 32                 JSR MESSAGE      ;PRINT HEADINGS
31BB- A0 00        NEXTVAR     LDY #Ø
31BD- B1 EC                    LDA (BASE),Y     ;GET VAR TYPE
31BF- F0 E2                    BEQ OUTD         ;Ø MEANS END OF LIST & PROGRAM
31C1- C5 61                    CMP *BUFF        ;SAME AS LAST TIME? - NO NEED
31C3- F0 30                    BEQ VAR          ;TO INDEX HERE
31C5- 85 61                    STA *BUFF        ;IF NOT, ESTABLISH NEW TYPE
31C7- 20 4D 83                 JSR CRLF
31CA- 20 4D 83                 JSR CRLF
31CD- A9 00                    LDA #ØØ
31CF- 85 62                    STA *BUFF+1      ;A=Ø, IF NEW TYPE THEN NEW NAME
31D1- A5 61                    LDA *BUFF        ;GET TYPE AGAIN
31D3- 10 11                    BPL ANDIT        ;INTEGER TYPES HAVE NEG TYPE #
31D5- A0 30                    LDY #M3-MØ       ;SEND '%'
31D7- 20 7F 32     JMESS       JSR MESSAGE
31DA- A5 61        SUBSC       LDA *BUFF        ;GET TYPE AGAIN
```



```
31DC- 4A                       LSR A            ;SUBSCRIPTED TYPES HAVE Ø-BIT SET
31DD- 90 16                    BCC VAR
31DF- A0 34                    LDY #M4-MØ       ;SEND '( )'
31E1- 20 7F 32                 JSR MESSAGE
31E4- F0 0F                    BEQ VAR          ;ALWAYS
                   ;
31E6- 29 10        ANDIT       AND #$1Ø         ;CHECK FOR STRING TYPE
31E8- D0 04                    BNE STR
31EA- A0 29                    LDY #M1-MØ       ;SEND ' '
31EC- D0 E9                    BNE JMESS        ;ALWAYS
                   ;
31EE- A0 2C        STR         LDY #M2-MØ       ;SEND '$'
31F0- D0 E5                    BNE JMESS        ;ALWAYS
                   ;
31F2- 20 4D 83     TWO         JSR CRLF
31F5- C8           VAR         INY
31F6- B1 EC                    LDA (BASE),Y     ;COMPARE VAR NAME TO LAST ONE
31F8- D1 EE                    CMP (P1),Y       ;IF IT'S NOT THE SAME
31FA- D0 0A                    BNE PUT          ;START NEW LINE
31FC- 48                       PHA    ;SAVE 1ST CHAR TEMP
31FD- C8                       INY
31FE- B1 EC                    LDA (BASE),Y     ;COMPARE 2ND CHAR
3200- D1 EE                    CMP (P1),Y
3202- F0 2E                    BEQ NUMB         ;IF EQUAL, JUST PRINT LINE NUMB
3204- 68                       PLA    ;GET 1ST CHAR BACK
3205- 88                       DEY
3206- 91 EE        PUT         STA (P1),Y       ;UPDATE BUFF 1ST POS
3208- 20 4D 83                 JSR CRLF         ;NEW LINE
320B- C6 EA                    DEC *PØ          ;PØ IS LINE COUNTER
320D- 10 12                    BPL JOCHR
320F- 48                       PHA              ;SAVE ACC TEMP
3210- A9 0F                    LDA #15          ;15 LINES/PAGE
3212- 85 EA                    STA *PØ
3214- 20 86 8B                 JSR ACCESS
3217- CE 53 A6                 DEC TECHO        ;NO ECHO
321A- 20 58 8A                 JSR INTCHR       ;WAIT FOR KEY
321D- EE 53 A6                 INC TECHO        ;REENABLE ECHO
3220- 68                       PLA    ;GET 1ST CHAR BACK
3221- 20 47 8A     JOCHR       JSR OUTCHR       ;PRINT 1ST CHAR
3224- C8                       INY
3225- B1 EC                    LDA (BASE),Y     ;GET 2ND CHAR AGAIN
3227- 91 EE                    STA (P1),Y       ;UPDATE BUFF 2ND POS
3229- D0 02                    BNE JOUT
322B- A9 20                    LDA #'           ;SUBST A SPACE IF NAME 1 CHAR LONG
322D- 20 47 8A     JOUT        JSR OUTCHR
3230- D0 0A                    BNE RES.CNT      ;ALWAYS
                   ;
3232- 68           NUMB        PLA    ;1ST CHAR OFF STACK
3233- C6 E8                    DEC *COUNT
3235- 10 09                    BPL NORM
3237- A2 00                    LDX #ØØ          ;NEW LINE - PUT IN SPACES
3239- 20 75 32                 JSR CRLF.SPS
323C- A2 05        RES.CNT     LDX #Ø5          ;MAX SIX LINE NUMS PER LINE
323E- 86 E8                    STX *COUNT       ;RESET COUNT OF LINE NUMB'S
3240- 20 3F 83     NORM        JSR SPC2
3243- 20 3F 83                 JSR SPC2
3246- 20 3F 83                 JSR SPC2
3249- 20 52 32                 JSR PR.LINUM     ;PRINT LINE NUMB
324C- 20 FE 30                 JSR INBASS5      ;INCR PTR FOR NEXT VAR
324F- 4C BB 31                 JMP NEXTVAR
                   ;
3252- C8           PR.LINUM    INY    ;SUBR TO CALC & PRINT LINE NUMB
3253- B1 EC                    LDA (BASE),Y     ;PUT LINE NUM
3255- 85 B1                    STA *FACTO-1     ;IN FACTO REGISTER
3257- 85 B3                    STA *FACTO+1
```

```
3259- C8        3800            INY
325A- B1 EC     3810            LDA (BASE),Y
325C- 85 B2     3820            STA *FACTO
325E- A2 90     3830            LDX #$90
3260- 38        3840            SEC
3261- 20 FF D9  3850            JSR FLOATC     ;FLOAT IT
3264- 20 9A DB  3860            JSR FOUT       ;AND CONVERT TO ASCII IN $100
3267- A2 00     3870            LDX #00
3269- BD 01 01  3880 LOADC      LDA $101,X     ;GET LINE NUMB
326C- F0 06     3890            BEQ DPR        ;0 MEANS END OF LINE NUMB
326E- 20 47 8A  3900            JSR OUTCHR
3271- E8        3910            INX
3272- D0 F5     3920            BNE LOADC      ;ALWAYS
                3930 ;
3274- 60        3940 DPR        RTS
                3950 ;
3275- 20 4D 83  3960 CRLF.SPS   JSR CRLF       ;SEND CRLF & 2*X SPACES
3278- 20 3F 83  3970 SPACES     JSR SPC2
327B- CA        3980            DEX
327C- 10 FA     3990            BPL SPACES
327E- 60        4000            RTS
                4010 ;
327F- B9 8C 32  4020 MESSAGE    LDA M0,Y       ;GET MESSAGE CHAR
3282- D0 02     4030            BNE JPRINT     ;MESSAGES END WITH 0
3284- A8        4040            TAY
3285- 60        4050            RTS            ;RETURNS WITH Z=1 & Y=0
                4060 ;
3286- 20 47 8A  4070 JPRINT     JSR OUTCHR
3289- C8        4080            INY
328A- D0 F3     4090            BNE MESSAGE    ;ALWAYS
                4100 ;
                4110 ;
328C- 0A 0A 56  4120 M0         .BY 10 10 'VARIABLES USED' 13 10 10
328F- 41 52 49
3292- 41 42 4C
3295- 45 53 20
3298- 55 53 45
329B- 44 0D 0A
329E- 0A
329F- 4E 41 4D  4130            .BY 'NAME' '    LINE NUMBER(S)' 00
32A2- 45 20 20
32A5- 20 4C 49
32A8- 4E 45 20
32AB- 4E 55 4D
32AE- 42 45 52
32B1- 28 53 29
32B4- 00
32B5- 20 20 00  4140 M1         .BY $20 $20 $00
32B8- 20        4150 M2         .BY $20
32B9- 20 24 00  4160            .BY ' $' $00
32BC- 20        4170 M3         .BY $20
32BD- 20 25 00  4180            .BY ' %' $00
32C0- 28 20 29  4190 M4         .BY '( )' $00
32C3- 00
32C4- 0D 0A 45  4200 M5         .BY $0D $0A 'ERROR IN LINE ' $00
32C7- 52 52 4F
32CA- 52 20 49
32CD- 4E 20 4C
32D0- 49 4E 45
32D3- 20 00
                4210 ;
                4220            .EN
```


A CORRECTION ON BAS-1 AND INTEGER VARIABLES

In Issue No. 10 we stated that the BASIC (BAS-1) Reference Manual made no mention of integer variables. We stand corrected. The top paragraph on page 9 contains the sentence "If the (variable) name ends in "%", then the variable is an integer variable, and may contain only integer values."

Oh, well, our error is not quite as bad as that made by a writer in one of the popular 6502 journals who stated that SYM's BAS-1 did not support integer variables at all. Apparently he never tried to use the "%" to see what would happen.

RFI (TVI) FROM THE KTM?

One of our callers asked us what measures we had taken to reduce TV interference from our KTM-2. He stated that his KTM-2 created interference on every TV in his apartment building. We told him that we had no experience with TVI, and after he hung up, we realized we had forgotten to ask whether he was using an RF modulator on his KTM-2.

We are in an area served only by one off-the-air TV station, on Channel 12, whose antenna is within 10 miles. No UHF within 90 miles, and only marginal reception from two distant VHF stations on Channels 7 and 9. Our "entertainment" video comes in by 14 channel cable (two channeld are "scrambled"), and we have two VCRs (Beta and VHS) for taping computer system demonstrations and lectures, plus at least four TVs and monitors within 25 feet of three nearly-always working SYM/KTM systems, and have seen absolutely no signs of TVI.

Our understanding is that TVI may be most troublesome on Channel 2, which we receive via shielded cable, with no interference. Are there any SYMmers out there bothered with TVI, and if so, what can be done to minimize the problems?

COMPUTER VIDEO

We used to carry a compact SYM-1 system along with us for demonstration purposes, and often still do, not so much now to demonstrate it, but for working purposes. We decided to give up the idea of trying to demonstrate the tremendous versatility of the SYM-1 (nee the VIM - Versatile Interface Module) by actual "live" demonstrations. This is because we have a number of SYM-1 systems dedicated to special applications, including speech synthesis, music generation, high resolution black & white graphics, color graphics, semi-automated production of distribution cassettes and disks, word processing, program development, hardware development and checkout, 6809 experimentation, etc., and it is really not practical to transport all of these for show-and-tell sessions.

We now feel that it is far simpler to bring along videotaped demonstrations, instead. Most schools and governmental agencies, and many industrial facilities have, or have access to, 3/4 inch U-Matic Format VCRs. Our university video crew prepared a 28 minute tape on the SYM-1, and a Honeywell video crew videotaped a 28 minute lecture entitled "How to Select a Personal Computer" for us. These tapes are so helpful that we decided it would be nice to be able to do our own, but on the 1/2 inch Beta and VHS VCR formats (much less expensive!). We can then dub to a borrowed U-Matic (the computer science department has one) recorder for school use. Here are some thoughts on the subject:

The RCA VP3301 Video Terminal (which we reviewed several issues ago, and for which we are dealers) turns out to be just the item for the generation of both still and animated titles, and areas of uniform color for headers and trailers, and to separate segments. The VP3301 has both RS-232C (EIA) (with inverted TTL compatibility) and 20 mA current loop (CL) interfaces.

We use our VP on the 20 mA interface and call it with a version of our decwriter II printer patch, modified for two reasons: first the VP does not accept 600 baud (which was added to our decwriter), and second, the VP requires "handshaking" at its higher baud rates. On the CL interface the rates are 110, 300, and 1200 baud. On the EIA interface the rates are 110, 300, 1200, 4800, 9600, and 19,200 baud! Of course the SYM-1 software stops at 4800, but as we get going, and convert one SYM's CL interface into a second EIA interface, we'll also look into modifying the software to the 19,200 rate. What beautiful animation that could permit! As of now we're working at the 110 rate, so that the titles are generated character by character, as if being hand typed.

We lent Jack Gieryic, who has a VHS VCR system, a VP3301 to "play" with, in exchange for the use of any software he developed for it. The VP also has excellent music capabilities. The video output is available at an RCA phono jack, and the audio output at a built-in speaker. Two additional VP3XXX series terminals are available. The VP3303 provides both video and audio modulation on an RF carrier (channel 3 or 4 selectable), and the VP3501 also has a built-in direct-connect modem, so that any facility with a telephone and a "telly" can become a computer terminal location. Even if you are not into Video Recording, you might want to consider the VP3XXX series terminals purely as a color graphics output device for the SYM-1, or as a "spare" terminal.

We tried feeding the KTM-2/80 video directly into a VCR, rather than using a video camera to copy the screen, and ran into two problems:

(1) The same reason that makes the use of an RF converter impractical also operates here. The 80 column format requires around 7.2 MHz bandwidth. We will try again with a KTM-2 (the 40 column terminal), since this requires only around 3.6 MHz. Meanwhile, we'll zoom in with a camera to portions of the KTM-2/80 monitor screen to make the characters easily readable on a color TV.

(2) Monochromatic signals at frequencies near 3.59 MHz confuse color TV sets into thinking color signals are being received (this is how the Apple II generates its color graphics); spurious colors are then displayed (no problem on B/W sets!).

Jack Gieryic showed us some video recordings of his MTU Visible Memory graphics, with pretty, but difficult to predict, color fringing. The colors do enhance the graphics, but the fringes make text rather unreadable. The best solution here seems to be to avoid videotaping high resolution black and white graphics for later presentation on a color monitor (unless you have an Apple II!). Of course, if your computer outputs NTSC color signals there is no problem. When we reinstall our ColorMate Board we'll try videotaping its output. It has been temporarily removed from service because it and the FDC-1 Floppy Disk Controller cannot co-reside at the $9000 block. The ColorMate (and the FDC-1) will both be moved to an 8K system and the ColorMate will be restrapped to $7000 for testing.

ANOTHER APPLE READER/WRITER?

Dave Kemp sent us many months ago a very compact Apple II Cassette Read/Write program. We didn't publish it until now because we hadn't tested it as yet. This was because he sent us only a source code listing (not in RAE format, because he does all of his software development on a larger system, then downloads the object code to the SYM).

Our typing is slow, so we now type in only the object code, turn Dessaintes' Disassembler loose on that, and then replace the "meaningless" labels with mnemonics from the printed listing. This gives us time to study the code as we replace the labels, as well as saving us typing time. We then insert comments only for the really obscure points.

To save on space, the version printed here is uncommented, and the label size was reduced to five characters with FO S 5 (the default is 10) so that we could get two columns to the page. Incidentally, if you have only a 40 column terminal, you will get nicer screen listings if you do the same, and also put all comments on separate lines. For your information the more pertinent commented lines are given below in a separate listing.

We publish this now mainly in the hope that owners of PETs, OSIs, Ataris, etc., will be inspired to write equivalent programs for their own systems so that "pure" ASCII text files, such as, for example, those generated by TECO, can be freely transportable.

Some of you might also want to compare the relative merits and speeds of the two cassette subsystems, and might perhaps even prefer to use the Apple format!

Notice also Dave's use of a D/A converter on the A Port to display the measured times during readback (the lines involving D2A and D2AD may be omitted if you do not need them). Since we have D/A converters on both the A and B Ports (for our stereo music system), we enjoyed watching both the signal itself and the measured time display on a dual trace scope.

COMMENT LINES (EXTRACTED FROM A MORE DETAILED VERSION WITH THE USE OF RAE'S ">FI " COMMAND) FOLLOW. THE UNCOMMENTED SOURCE CODE LISTING (CUT AND PASTED TO FIT A SINGLE PAGE!) APPEARS ON PAGE 19.

```
0310  APPLE  CPX #$02 !TWO PARAMS     0890         EOR #$FF !COMPLEMENT
0340         CMP #$14 !L3 HASH CODE   0940         LDY #$FF !RESTART
0360         CMP #$1F !S3 HASH CODE   0960         CMP #$CE !???
0390         LDA #$07 !CONFIGURE      1010         AND #$40 !BIT SIX
0410         LDA #$20 !FIVE SEC HDR   1080  HEADR  LDY #$AF !650 USEC
0430  WR1    LDX #$00 !ZERO INDEX     1110         ADC #$FE !???
0510  WRBYT  LDX #$10 !WRITE BYTE     1130  WRBIT  LDY #$E1 !(F0) 250 USEC
0590  READ1  LDX #$FF !INITIALIZE     1150         LDY #$C2 !(E1) 500 USEC
0810  RBYT   LDX #$08 !READ BYTE      1180         LDA #$FF !---
0840         CMP #$A8 !(DC)           1230         EOR #$08 !---
```

SYM-PLE COPY

SYM-ple Copy is a very useful utility program for making duplicates of SYM Format tapes. It provides the user with multiple options for duplicating anything from single programs to entire cassettes, with minimal effort. SYM-ple Copy is completely relocatable, so that it can be entered far away from any programs read in.

It is necessary to have two cassette recorders, one for reading programs from the master tape, the other for recording to the copy tape. Each must have a remote jack, or the equivalent, so that both can be under computer control. In order to control them automatically, an inexpensive external circuit must be implemented, as shown in Figure 1. This circuit is essentially the same as the on-board cassette control circuitry, with the exception of resistors R1 and R2.

SYM-ple Copy is actually two programs in one. After the user starts the program a question mark appears on the display. This asks the user if he wishes to use Option A or Option B.

Option A allows the user to start the program and leave while all programs on the master tape are duplicated to the copy tape. In answer to the "?" the user enters "F" for the "fast and easy" way of copying tapes. Immediately the tape player begins reading the first program from the master tape; it then stops and the recorder writes that program to the copy tape. These steps are repeated until all programs have been copied.

(continued to page 11-20)

```
ØØ1Ø ;     APPLE CASSETTE READ/WRITE
ØØ2Ø ;     MONITOR EXTENSIONS FOR SYM
ØØ3Ø ;   ADAPTED FROM APPLE MONITOR ROM
ØØ4Ø ;        BY D. KEMP    SEPT 79
ØØ5Ø
ØØ6Ø ;TO ENABLE:   .SD STRT,A66D
ØØ7Ø ;TO WRITE :   .S3 STAD,ENAD
ØØ8Ø ;TO READ  :   .L3 STAD,ENAD
ØØ9Ø
Ø1ØØ OLD      .DE $F9
Ø11Ø CKSUM    .DE $FA
Ø12Ø BUFAD    .DE $FE
Ø13Ø
Ø14Ø INCP3    .DE $8293
Ø15Ø INCMP    .DE $82B2
Ø16Ø START    .DE $8DA9
Ø17Ø
Ø18Ø TAPIN    .DE $AØØØ
Ø19Ø D2A      .DE $AØØ1
Ø2ØØ D2AD     .DE $AØØ3
Ø21Ø
Ø22Ø TAPOU    .DE $A4Ø2
Ø23Ø TIMER    .DE $A4Ø6
Ø24Ø TIM8     .DE $A415
Ø25Ø
Ø26Ø STRT     .DE $Ø2ØØ
Ø27Ø
Ø28Ø          .BA STRT
Ø29Ø          .OS
Ø3ØØ

Ø2ØØ- EØ Ø2     Ø31Ø APPLE   CPX #$Ø2
Ø2Ø2- DØ 61     Ø32Ø         BNE ERR
Ø2Ø4- 2Ø 93 82  Ø33Ø         JSR INCP3
Ø2Ø7- C9 14     Ø34Ø         CMP #$14
Ø2Ø9- FØ 2B     Ø35Ø         BEQ READ
Ø2ØB- C9 1F     Ø36Ø         CMP #$1F
Ø2ØD- DØ 56     Ø37Ø         BNE ERR
Ø2ØF- 2Ø A9 8D  Ø38Ø WRITE   JSR START
Ø212- A9 Ø7     Ø39Ø         LDA #$Ø7
Ø214- 8D Ø2 A4  Ø4ØØ         STA TAPOU
Ø217- A9 2Ø     Ø41Ø         LDA #$2Ø
Ø219- 2Ø 97 Ø2  Ø42Ø         JSR HEADR
Ø21C- A2 ØØ     Ø43Ø WR1     LDX #$ØØ
Ø21E- 41 FE     Ø44Ø         EOR (BUFAD,X)
Ø22Ø- 48        Ø45Ø         PHA
Ø221- A1 FE     Ø46Ø         LDA (BUFAD,X)
Ø223- 2Ø 2C Ø2  Ø47Ø         JSR WRBYT
Ø226- 2Ø B2 82  Ø48Ø         JSR INCMP
Ø229- 68        Ø49Ø         PLA
Ø22A- 9Ø FØ     Ø5ØØ         BCC WR1
Ø22C- A2 1Ø     Ø51Ø WRBYT   LDX #$1Ø
Ø22E- ØA        Ø52Ø WRBY1   ASL A
Ø22F- 2Ø A2 Ø2  Ø53Ø         JSR WRBIT
Ø232- DØ FA     Ø54Ø         BNE WRBY1
Ø234- 18        Ø55Ø WRBY2   CLC
Ø235- 6Ø        Ø56Ø         RTS
                Ø57Ø
Ø236- 2Ø A9 8D  Ø58Ø READ    JSR START
Ø239- A2 FF     Ø59Ø READ1   LDX #$FF
Ø23B- 86 FA     Ø6ØØ         STX *CKSUM
Ø23D- 8E Ø3 AØ  Ø61Ø         STX D2AD
Ø24Ø- 2Ø 7A Ø2  Ø62Ø READ2   JSR RBIT1
Ø243- BØ F4     Ø63Ø         BCS READ1
Ø245- CA        Ø64Ø         DEX
Ø246- DØ F8     Ø65Ø         BNE READ2
Ø248- 2Ø 7A Ø2  Ø66Ø READ3   JSR RBIT1
Ø24B- 9Ø FB     Ø67Ø         BCC READ3
Ø24D- 2Ø 7A Ø2  Ø68Ø         JSR RBIT1
Ø25Ø- 2Ø 67 Ø2  Ø69Ø READ4   JSR RBYT
Ø253- 81 FE     Ø7ØØ         STA (BUFAD,X)
Ø255- 45 FA     Ø71Ø         EOR *CKSUM
Ø257- 85 FA     Ø72Ø         STA *CKSUM
Ø259- 2Ø B2 82  Ø73Ø         JSR INCMP
Ø25C- 9Ø F2     Ø74Ø         BCC READ4
Ø25E- 2Ø 67 Ø2  Ø75Ø         JSR RBYT
Ø261- 45 FA     Ø76Ø         EOR *CKSUM
Ø263- FØ CF     Ø77Ø         BEQ WRBY2
Ø265- 38        Ø78Ø ERR     SEC
Ø266- 6Ø        Ø79Ø         RTS
                Ø8ØØ
Ø267- A2 Ø8     Ø81Ø RBYT    LDX #$Ø8
Ø269- 48        Ø82Ø RBYT1   PHA
Ø26A- 2Ø 77 Ø2  Ø83Ø         JSR RBIT
Ø26D- C9 A8     Ø84Ø         CMP #$A8
Ø26F- 68        Ø85Ø         PLA
Ø27Ø- 2A        Ø86Ø         ROL A
Ø271- CA        Ø87Ø         DEX
Ø272- DØ F5     Ø88Ø         BNE RBYT1
Ø274- 49 FF     Ø89Ø         EOR #$FF
Ø276- 6Ø        Ø9ØØ         RTS
                Ø91Ø
Ø277- 2Ø 88 Ø2  Ø92Ø RBIT    JSR GETTR
Ø27A- 2Ø 88 Ø2  Ø93Ø RBIT1   JSR GETTR
Ø27D- AØ FF     Ø94Ø         LDY #$FF
Ø27F- 8C 15 A4  Ø95Ø         STY TIM8
Ø282- C9 CE     Ø96Ø         CMP #$CE
Ø284- 8D Ø1 AØ  Ø97Ø         STA D2A
Ø287- 6Ø        Ø98Ø         RTS
                Ø99Ø
Ø288- AD ØØ AØ  1ØØØ GETTR   LDA TAPIN
Ø28B- 29 4Ø     1Ø1Ø         AND #$4Ø
Ø28D- C5 F9     1Ø2Ø         CMP *OLD
Ø28F- FØ F7     1Ø3Ø         BEQ GETTR
Ø291- 85 F9     1Ø4Ø         STA *OLD
Ø293- AD Ø6 A4  1Ø5Ø         LDA TIMER
Ø296- 6Ø        1Ø6Ø         RTS
                1Ø7Ø
Ø297- AØ AF     1Ø8Ø HEADR   LDY #$AF
Ø299- 2Ø AB Ø2  1Ø9Ø         JSR WRBI2
Ø29C- DØ F9     11ØØ         BNE HEADR
Ø29E- 69 FE     111Ø         ADC #$FE
Ø2AØ- BØ F5     112Ø         BCS HEADR
Ø2A2- AØ E1     113Ø WRBIT   LDY #$E1
Ø2A4- 9Ø Ø2     114Ø         BCC WRBI1
Ø2A6- AØ C2     115Ø         LDY #$C2
Ø2A8- 2Ø AB Ø2  116Ø WRBI1   JSR WRBI2
Ø2AB- 48        117Ø WRBI2   PHA
Ø2AC- A9 FF     118Ø         LDA #$FF
Ø2AE- CC Ø6 A4  119Ø WRBI3   CPY TIMER
Ø2B1- 9Ø FB     12ØØ         BCC WRBI3
Ø2B3- 8D 15 A4  121Ø         STA TIM8
Ø2B6- AD Ø2 A4  122Ø         LDA TAPOU
Ø2B9- 49 Ø8     123Ø         EOR #$Ø8
Ø2BB- 8D Ø2 A4  124Ø         STA TAPOU
Ø2BE- 68        125Ø         PLA
Ø2BF- CA        126Ø         DEX
Ø2CØ- 6Ø        127Ø         RTS
                128Ø
                129Ø         .EN
```

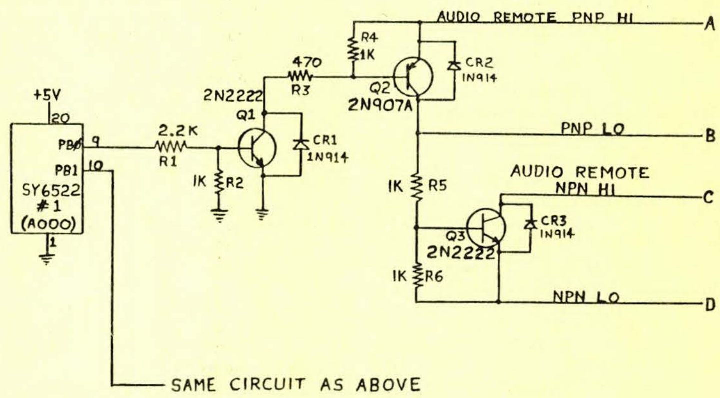

SYM-PLE COPY (continued from page 11-18)

Option B allows the user full control over which programs are to be copied. In answer to the initial "?" the user may hit any key except "F", for example, "D", for do-it-yourself! When the key is hit the player reads in the first program and stops, and the ID and the starting address of the program are displayed. The user may enter "Ø" to bypass copying, if he so desires. Otherwise, if he wishes a copy to be made, he may enter either a "1" followed by two hex digits which will become the new ID, or any other character, in which case the ID will remain the same as on the source cassette. The process is repeated for each file on the master tape.

Only the first six locations in page zero are used. These are for EAL, EAH, SAL, SAH, ID, and a scratch byte, respectively. A "safe" origin for SYM-ple Copy in smaller systems would be at $ØØØ6, since it is good programming practice for 6502 systems to origin most programs at $Ø2ØØ and above, and for SYM programs to do their own initialization of page zero (and page one), if required. This is because SYM will "hang-up" when reading data from cassettes which cross the page zero/page one boundary, and reading cassette data across the page one/page two boundary will clobber the stack and any returns you may have saved there!

SYM-PLE COPY - BY: P. GLENN NORMAN, 8Ø6 WAVECREST, HOUSTON, TX 77Ø62

[EDITOR'S NOTE: This program will work with BAS-1 files (which all start at $Ø2Ø1, incidentally) but Option B will not be usable with RAE-1 files. This is because RAE-1 files are dumped with an initial header in which the File Number is imbedded, and all headers and files are dumped to tape with ID = ØØ. RAE-1 is designed to work with a dual cassette system such as Mr. Norman suggests, and so is most of Jack Brown's software, but with the write casseette remote driven by CB2 and the read cassette remote driven by PB7 of VIA #1, rather than by PB1 and PBØ as Figure 1 shows.]



NOTE: PBO controls the tape player
, PB1 controls the tape recorder
Refer to the SYM-1 Reference Manual for a particular recorder's hookup.
If outputs C & D are used, A must be tied to +5V.

Figure 1

AUDIO CASSETTE REMOTE CONTROL

SYM-PLE COPY

```
0300  20 86 8B          JSR ACCESS     Unwrite System RAM
0303  A9 09      HERE:  LDA #$09       Set up ability
0305  20 A5 89          JSR CONFIG          to use display
0308  A9 53             LDA #$53       Load "?" in accumulator
030A  8D 00 A4          STA DIG        Put "?" on display
030D  20 23 89          JSR KEYQ       Is key down on keypad?
0310  F0 F1             BEQ HERE       If no, keep "?" displayed
0312  20 AF 88          JSR GETKEY     If yes, get that key
0315  85 05             STA $0005           and store it in $0005
0317  A9 03             LDA #$03       Configure PB0 & PB1
0319  8D 02 A0          STA $A002           as outputs
031C  A9 01     BEGIN:  LDA #$01       Tape player, on
031E  8D 00 A0          STA $A000      Tape recorder, off
0321  A0 80             LDY #$80       Set up mode for SYM format
0323  20 A9 8D          JSR START      Initialize
0326  20 52 8D  SEARCH: JSR SYNC       Get in sync
0329  20 E1 8D    DATA: JSR RDCHTX     Read first character
032C  C9 2A             CMP "*"        Start of data?
032E  F0 06             BEQ LOAD       If so, get data
0330  C9 16             CMP "sync"     If no, sync character?
0332  D0 F2             BNE SEARCH     If not, start sync search
0334  F0 F3             BEQ DATA       If yes, keep looking for "*"
0336  A9 80       LOAD: LDA #$80       Reset to SYM mode
0338  85 FD             STA MODE            for tape format
033A  20 E5 8D          JSR RDBYTH     Read ID off tape
033D  8D 00 A4          STA DIG        Display on LED (not decoded)
0340  85 04             STA $04        Store ID in user buffer
0342  20 74 8E          JSR RDCHK      Get SAL from tape
0345  85 FE             STA BUFADL     Put in monitor buffer
0347  85 02             STA $02        Also store in user buffer
0349  20 74 8E          JSR RDCHK      Get SAH from tape
034C  85 FF             STA BUFADH     Put in monitor buffer
034E  85 03             STA $03        Also store in user buffer
0350  20 74 8E          JSR RDCHK      Get EAL from tape
0353  8D 4A A6          STA EAL        Save in monitor buffer
0356  85 00             STA $00        Also store in user buffer
0358  20 74 8E          JSR RDCHK      Get EAH from tape
035B  8D 4B A6          STA EAH        Save in monitor buffer
035E  85 01             STA $01        Also store in user buffer
0360  20 DE 8C          JSR LT7H       Ok, read data off tape
0363  90 02             BCC OK         Data read in ok?
0365  B0 BA       BACK: BCS BEGIN      No - start over
0367  A5 03         OK: LDA $03        Pick up SAH from buffer
0369  20 FA 82          JSR OUTBYT     Display on LEDs
036C  A5 02             LDA $02        Pick up SAL from buffer
036E  20 FA 82          JSR OUTBYT     Display on LEDs
0371  A5 04             LDA $04        Pick up ID from buffer
0373  20 FA 82          JSR OUTBYT     Display on LEDs
0376  A5 05             LDA $05        Pick up operation type
0378  C9 46             CMP "F"        See if user had hit "F"
037B  F0 15             BEQ KEEP       Yes - no need to wait
037D  A9 00             LDA #$00       Tape recorder, off
037F  8D 00 A0          STA $A000      Tape player, off
0381  20 AF 88          JSR GETKEY     Wait for user to hit key
0384  C9 30             CMP "0"        Is choice = "0"?
0386  F0 94             BEQ BEGIN      Yes - then start all over
0388  C9 31             CMP "1"        Is choice = "1"?
038A  D0 05             BNE KEEP       No - keep record as is
038C  20 D9 81          JSR INBYTE     If "1", get ID from user
038F  85 04             STA $04             and store in buffer
0391  A2 FB       KEEP: LDX #$FB       Load pointer with minus 5
0393  B5 05      TABLE: LDA $05,x      Pick up all data from buffer
0395  9D 4F A6          STA $A64F,x    Store data in monitor area
```



```
0398  E8                INX            Increment pointer
0399  D0 F8             BNE TABLE      Keep loading if not finished
039B  A9 02             LDA #$02       Tape player, off
039D  8D 00 A0          STA $A000      Tape recorder, on
03A0  A0 80             LDY #$80       Load Y with SYM format
03A2  20 87 8E          JSR DUMPT      Save data on tape
03A5  38                SEC            Ok, start
03A6  B0 BD             BCS BACK                 all over again
```

"PRETTY-PRINTING" IN RAE

Here is a small section of a program sent to us by John Blalock. He has solved the problem of making the comments following one byte instructions line up with the comments following multi-byte instructions in a very "pretty" way.

```
                 0001 PTR        .DE 0
                 0002 COUNT      .DE 0
                 0003
0200- 20 11 02   2540 GETCH      JSR SAVER    ;save all registers on stack
0203- 08         2550            PHP :        ;save flag register separately
0204- A0 00      2560            LDY #0       ;clear Y register
0206- A5 00      2570            LDA *COUNT   ;was COUNT = zero ?
0208- D0 07      2580            BNE ONE?     ;no, then branch
020A- C8         2590            INY :        ;it was zero, add 1 to Y
020B- B1 00      2600            LDA (PTR),Y  ;get MSD of line number
020D- E6 00      2650            INC *COUNT   ;now COUNT = one
020F- D0 00      2660            BNE NDONE    ;branch always
                 9001 ONE?
                 9002 SAVER
                 9003 NDONE
```

A VERY USEFUL "SIGNAL GENERATOR"

Occasionally SYMmers may have troubles with their SYMs failing to operate when new RAM or ROM is added. This is often due to some added memory being "stuck" in the selected position so that its data is always dumping to the data lines, due to faulty decoders, or whatever. If you have a scope, or even just an inexpensive logic probe, all you need is a simple device to get your SYM to cycle through all 64K addresses, so that you can observe the responses throughout the system.

Volume 1, Issue 3, Page 1, of Commodore's TECHTOPICS describes just such a device, which they call a NO OP TESTER. In brief, just take a "spare" 6502, and bend up pins 26 through 33 so that they will not fit into the socket. These pins are data lines DA7-DA0, respectively. Next, wire pins 26, 27, 28, 30, and 32 to pin 8 (+5V), and pins 29, 31, and 33 to pin 21 (GND). Install this "doctored" 6502 into your system and then do your signal tracing.

What you have done is forced the 6502 data lines to "read" $EA, which is the NOP code. Since NOP is a two cycle operation, the 6502 will count through all 64K addresses in 128K useconds, or at a 7.63 Hz rate. You should then "probe" address lines, decoder outputs, chip selects, etc.

Just for the fun of it, we wired up a NOOP TESTER, and checked out the address decoding chips on a working SYM with both a scope and an under $20.00 logic probe from Radio Shack. We may make up a 6507 version for use in KTM-2 trouble-shooting. (Thanks to Chuck Harrison, of Groton, CT, for sending us the copy of TECHTOPICS and other valuable material.)

SEAT THOSE CHIPS

FORETHOUGHT PRODUCTS, 87070 Dukhobar Road, Eugene, OR 97402, makes a number of accessory products for the AIM 65 which will also work with the SYM-1. We quote from their newsletter, "The AIM-Mate Monitor", Vol. 1, No. 3, some advice which is also pertinent to the SYM:

"A number of AIM 65 system problems have been traced to faulty IC sockets on older AIM 65 boards. These sockets, which make contact with only the outside shoulder of the IC pin, can develop a high resistance between the socket and the IC pin over time. If trouble occurs with older AIM 65 boards, try re-seating all the ICs (pressing firmly on both ends will usually do the trick) before you ship it off for repair. Keeping the ICs firmly seated in their sockets (especially the 40-pin ICs) will often head off system trouble with these not-so-perfect sockets. Note: Newer AIM 65 boards use a socket type which is not subject to these problems."

We pointed out this problem as existing with some SYMs and KTMs in an earlier issue. The types of sockets used by Synertek Systems Corporation do vary from one production run to the next, and often differ from 18-pin to 24-pin to 40-pin types, so you'll just have to look at your own systems to see which you have. Meanwhile, "flexing your boards and wiggling your chips" often helps, at least temporarily, in "fixing" some of your intermittent problems. Our FODS system gives an error message when what has been down-loaded to RAM does not check when compared immediately to the contents of the disk (really a great feature). Whenever this happens, we wiggle the connectors on our external 16K RAM Board (one made by Synertek to fit the Motorola EXORCISOR bus) and the problem clears up.

MORE ON THE CASSETTE INTERFACE
---- -- --- -------- ---------

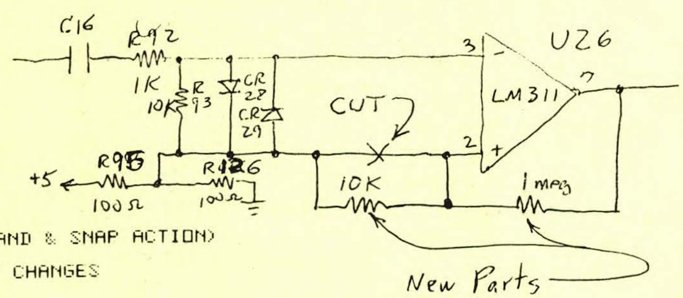
TRANSCONTINENTAL MEMO

FROM: Jerry Avins
TO: Lux Luxenberg
SUBJECT: SYM Tape Interface

139 Kendall Rd.
Kendall Park
N.J. 08824

Dear Lux,

The other day, I had some trouble with a tape that had worked often in the past, provided I used a "good" recorder at the "right" volume. A quick look with the oscilloscope raised my eyebrows, turned my stomach, and caused my heart to sink. The risetime of the comparitor output was very long, and the transition very irregular. After reformatting my anatomy, I took a hard look at the soft errors.



HYSTERESIS (DEAD BAND & SNAP ACTION)

CIRCUIT CHANGES

The schematic showed that no hysteresis is provided, and before changing the diode connections as J. Sinnett (SYM-PHYSIS 3-4) suggests - a good idea that I haven't gotten around to yet - I chose to install the hysteresis. In the diagram , the 10K resistor makes the hysteresis possible and incidentally temperature-compensates the comparitor. The 1 meg resistor could be varied to change the amount of hysteresis, but the need should not arise. This mod doesn't interfere with biasing the comparator to make PB6 on U25 available. The new risetime is better than my audio scope can follow, so there is no longer any need to turn volume up to where rectification becomes a problem.

Now all of my cheap recorders are "good", and much lower volumes are "right". I just turn up the knob until the 'AUDIO' LED glows dimly and things work fine. Incidently, C16 at .22 or .05 microfarads works about as well: maybe the best value is in the middle!

P.S. In a later telecon Jerry says that, in his opinion, after many experiments, a 0.22 ufd capacitor works best at C16. We feel that the choice may depend on the recorder being used. We like the smaller values since low frequency response is not required, and the smaller values reduce the effects of any 60 Hz hum present, and permit faster "settling" if there is a DC component at the input to C16. Slow settling could (and does) prevent RAE GEts from locking onto the short synch bursts if the read recorder is started under remote control.

Many recorders do not have a DC-blocking capacitor at the earphone jack. For example, our Radio Shack Realistic CTR 80(A) puts out -0.47 V at the earphone jack when in STOP; this jumps to over 5.2 V and settles down to 2.77 V when in PLAY. When we parallel the jack with an 8 ohm earphone and then remove the earphone the polarity reverses (this DC shift is one reason we recommend leaving an earphone in; the other is for proper loading).

MORE ON COPYRIGHTS
---- -- ----------

ITEM 1

We received, through Jack Gieryic, a copy of the Honeywell Computer Club Newsletter, dated February/March 1981 (sic, it should have said 1982!), with an article by Dan Buchler, entitled "Copyright Software Considerations for Microprocessor Users", from which we quote the following sentence:

". . . If you have two persons using a program on the same or different systems, you may not copy a copyright program simply for the convenience of the second person who wants to use the system. . . ."

We feel that the home environment is so distinctively different from the traditional academic and industrial environments in which computers once exclusively resided, that whatever family members do with their "family" computers in the privacy of their own homes must be treated differently under the law.

We treat purchased software much as we do a reference book (with the exception that we immediately make a backup copy). We buy only one copy, and each family member uses it as required. Since we have elected to satisfy our family requirements by providing individual systems for each user, rather than with a single time-shared multi-user system, each of the systems has a copy available on its own mass storage device. This is for convenience mainly, and it is very unlikely that two copies are being used simultaneously.

ITEM 2

We have just received a copy of Saturn Software's SK-FORTH 79 (Release 2.0), and a beautiful package it is, indeed. The two manuals which come with the package are extremely thorough, and, of course, are coprighted!

We worked with Jack Brown on the production of the manuals for Release 1.0, and we know how much time and effort went into those, and what the printing costs for a small production run can be, so we have a pretty good idea of how many copies of Release 2.0 must be sold in order for Jack Brown to break even on his production costs, let alone grow exorbitantly rich on the profits.

We certainly hope that Jack finds it worth his time and effort to continue publishing such quality software, with full source code available. He can only do this if enough copies are sold. This means that we, as users, should ask ourselves how we would feel if Jack were giving away, or swapping, copies of similiar quality software we ourselves were trying to market, whenever we are tempted to exchange a copy of FORTH for a copy of Pascal, or whatever!

ITEM 3

Source code is protected under the copyright law in much the same manner as any other "literary" work. But what about object code, in ROMs, for example?

Richard H. Stern, in the February 1982 issue of Computer Design, Vol. 21, No. 2 (pp. 131-144), in an article entitled "Copying ROMs: Right or Wrong" cites the two following court decisions, which should answer the question(?).

(1) 1979: Chicago Federal Trial Court, in Data Cash Systems, Inc v JS&AA Group, Inc - - ROMs ARE NOT protected under copyright law.

(2) 1981: San Francisco Federal Trial Court, in Tandy Corp v Personal Micro Computers, Inc - - ROMs ARE protected under copyright law.

Take your choice!

ON ROMS, SRAMS, AMD EPROMS

We will be rebuilding our EPROM burner(s) to handle 2532s and/or 2732s, and, while we are at it, we will add some "convenience" features to be able to download 2K ROMs as well, in particular those with chip-select polarities different from "standard", i.e., those which we can't just plug into a SYM for reading. Examples of such ROMs are those in the Apple II, and, closer to home, the "main" ROM in the KTM.

We have D(ynamic)RAM at $9000 ("surplus" from the 32K Beta Board) on our main development system in which we test and debug programs to be burned into 2716 EPROMs at the same address for OEM systems. A variety of 2K x 8 S(tatic)RAMs, (some CMOS) - Hitachi 6116, TMM 2016, NEC 446 and 447, Toshiba 5516, etc. - is now available which are pin compatible with the 2716s, and may be used for similiar program development. We prefer RAM to ROM, anyway, in disk systems (except for a BOOT ROM), and so will be putting a 6116 in at $F000 in our CODOS system.

If you plan to use only 2K EPROMs or SRAMs in the SYM sockets, you might wish to abandon the wiring convention suggested for 2716s used on the SYM, as described in Table 4-3a of the SYM-1 Reference Manual. This convention was adopted so that pin 20 could be used for chip selection for all ROMs and EPROMs used on the SYM. The standard convention for 2716s is to use pin 18 as the Chip Enable (low) and to tie pin 20 - Output Enable (low) to ground. This choice has the added advantage that the 2716s are placed in a low power "standby" mode when not selected. Pin 21 should remain connected to +5 V. The only change then needed when installing a 2K x 8 SRAM in place of a 2716 is to rewire pin 21 to RAM R/W (Read-High/Write-Low), which is available at pin Z of the Expansion Connector (E).

Mitchell G. VanOchten, of Livonia, MI, recommends the following method of installing SRAMs in U21, U22, and/or U23:

> Tie K, L, and/or M (pins 20) to ground at jumper position 2 - 3.
> Tie F, G, and/or H (pins 21) to RAM R/W at E - Z.
> Tie B, C, and/or D (pins 18) to the desired 2K address block at the appropriate jumper point, with external 2.2 K pull-up resistors to +5 V.

Mr. VanOchten reports that he had spurious addressing problems when using pins 20 for chip selection, and that the method presented above eliminated the troubles. We have not yet installed our "sample" SRAM, so we don't know if the "fix" he suggests is necessary. We would prefer to use the pull-up resistors already on board at K, L, and M. Our own suggestion would be to tie K, L, and M to the 2K address block jumpers, as recommended in the reference manual, but then tie B, C, and D to H, L, and M, respectively, so that CE (pins 18) and OE (pins 20) are tied together. The power-saving standby mode is thus still enabled.

SYNERTEK ROMS

Here is a list of the Synertek proprietary ROMs used in the SYM-1. We do not have any information on the ROMs used in the SYM-2, MDT-1000, KTM-3, etc.

02-0012A	2332 (4K)	MON 1	(SY 1.0)
02-0012B	2332 (4K)	*MON 2	(SY 1.1)
02-0053A	2364 (8K)	RAE-1	(Requires inversion of A12)
02-0053B	2364 (8K)	*RAE-1	(Current one chip version)
02-0023A,24A	2x2332 (4K,4K)	*RAE-1/2	(Two chip version)
02-0058A	2364 (8K)	*BAS-1	(One chip version)
02-0019A,20A	2x2332 (4K,4K)	BAS-1	(Two chip version)

NOTES: The suffixes "A" and "B" in some production runs are replaced by "-01" and "-02", respectively. "*" indicates current production.

KTM-2/80 CHIPS

The 2K ROM (2316E) currently being supplied with the KTM-2/80s bears the house number 02-0050-02 (the -02 indicates a "B" version). Our original "Old Faithful" KTM-2/80 has a ROM marked 02-0050-A. Since we have not disassembled either we do not know how the two ROMs differ. It is possible that the differences may be significant when when you replace the ROM with an EPROM of your own programming.

The KTM-2s use the 02-0016B as the main ROM. The very, very, old KTM-2s used the 02-0016-01 (we had one of these), and the display was much too wide for the typical overscanning type of monitor, or a TV set (with RF input) to handle. Switching from the -01 (or A) ROM to the -02 (or B) ROM required also a change of the crystal from 12.598 MHz to the current 14.31818 MHz value.

All KTMs use 02-0017B ROMs for character generation; unlike all other Synertek 2316Es these are directly replaceable by 2716 EPROMs, since the three CS lines have the same polarity specifications.

Bob Myers asked us to point out that some KTMs use 20-pin 8304s at U34 and U35, while others use 18-pin 8T245s. The board will accept either type, although all the manuals we have seen specify the 8304. The 8304 is described in the manual as a port, bi-directional. The 8T245 is similiar to the 74245, but has a different pinout.

MODIFYING KTM-2/80 ESCAPE SEQUENCES

The following extracts from a recent letter from Dr. Strube provide additional information on KTM-2/80 customization. Note that he, too, is an experimental psychologist. The FORMATTER he describes is even more versatile than SWP-1! We have not tried it ourselves, as yet, because we must first "recustomize" it for our own system configurations(s), which differ from his in memory mapping, etc. His instruction manual is superb. We'll be contacting him to see if he wishes to offer it for sale.

D r. G E R H A R D S T R U B E

Sckellstraße 5 D-8000 München 80 Tel. 089-4801417

Re: Customizing the KTM, and text FORMATTER

thank you for including my notes on the KTM in the latest issue of SYMPHYSIS. Since you now got an EPSON printer, with a real wealth of ESC codes (not all compatible with the KTM), I should like to point out how to 'customize' KTM escape sequences.

The 2k KTM program (let us assume addresses from $000 to $7FF) tests for ESC at locations $58C and $5DB by CPX §$D8 (backwards as usual, so $D8 actually means $1B). Ensuing CPX commands test for all the ESC sequences:

LOCATION	CPX §$	MEANING	
$5C9	BC	=	abs. cursor addr.
$5D7	D4	+	rel. cursor addr.
$5E3	A2	E	form feed
$5EA	12	H	home
$5F4	52	J	clear EOS
$5FB	D2	K	clear EOL
$602	E2	G	enter graphics mode
$60F	4A	R	enter reverse mode
$617	E6	g	leave graphics mode
$624	4E	r	leave reverse mode
$62C	36	l	aux. port off
$639	32	L	aux. port on

I am glad to credit my colleague, Dr. Werner Schubö, for the discovery of these addresses. On my system, I have changed the ESC l to an ESC CR in order to avoid having printed an 'l' when I switch off my printer (which is connected to the KTM aux port).

My printer, by the way, is an Olivetti typewriter which uses daisy-wheels and prints up to 30 chars. per second. As you may have guessed, it is the most expensive part of my system (about $ 1600), but, since I use it for all my manuscripts, it is indispensable. The rest of the system consists of a

SYM, together with a Computerist DRAM, a Philips Mini-DCR (digital cassette recorder, 6kBaud, ECMA norm), and an Anderson-Jacobson modem (connecting my SYM to a Cyber 175). Through the addition of two 6551 ACIAs, the modem and the KTM are both handled by interrupt. - I use a second, 'naked' SYM, with some analog and digital interfaces, for control of experiments, then read the data into my SYM at home, send them to the Cyber for statistical analysis, and get the results back to where I write the reports (I'm an experimental psychologist, by the way).

Since you took interest in the text formatting program I use, I have prepared some information which, along with the source included, should enable you to adapt FORMATTER to your system with little effort. I'm eager to get your

remarks, since I do not know other formatters for RAE, e.g. the Moser TWP.

Sincerely yours,

Ihr

CONVERTING TTY OUT TO A SECOND CRT OUT

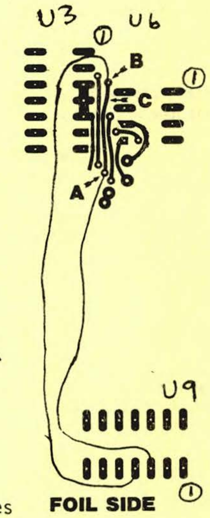
Here's a very simple way to convert SYM's 20 mA Current Loop (CL) output to Inverted TTL output. Remember that Inverted TTL is accepted by most modern modems (DCE - Data Communication Equipments) and/or terminals and printers (DTE - Data Terminal Equipments) which conform to RS-232C (EIA) specifications. All such equipments designed around the 1488/1489 transmitter/receiver chip pair will accept Inverted TTL. All Epson serial interface adapters accept Inverted TTL, so that if you want to free your parallel interface for more interesting uses, such as Voice I/O, EPROM burning, or whatever, here is how to do so.

Incidentally, the SYM-1 is configured as a DCE, the main port on the KTM as a DTE, and the aux port on the KTM as a DCE. When interconnecting DCEs to DTEs, wire 2 to 2 and 3 to 3. When interconnecting DCEs to DCEs or DTEs to DTEs, wire 2 to 3 and 3 to 2. In all cases 7 is the signal ground on both. Be careful in using pin 1 as a signal ground, as on some equipments it is connected to the third wire of the power cord, and accidents can happen in this area.

TELEX COMMUNICATIONS, INC.
Divisions: HY-GAIN • MAGNECORD • TURNER®
9600 Aldrich Avenue South • Minneapolis, Minnesota 55420 U.S.A. • Telephone: (612) 884-4051 • telex: telexcomdi mps 29-7053

Dear Lux,

To modify the current loop output to the same configuration as the CRT output for inverted TTL output to a serial printer, an inverter and a resistor must be added. This was discussed in an earlier newsletter.



An easy way to do this is to cut the foil at point "C" located on the foil side of the board between U3 and U6. Solder a wire from point "A" to pin 3 of U9 (input to inverter), and solder a wire from point "B" to pin 4 of U9 (output of inverter). Push the wires right into the feedthrough holes at points "A" and"B". You have now connected the unused inverter in U9 between PB5, pin 18 of U27, and the base resistor of Q30. (See fig. 4.2 in the Sym Reference Manual.)

Next connect a 200 or 220 ohm resistor from the emmiter of Q30 to ground.

This is done by first locating resistor R110. On the newer SYM-1's R110 is located just to the right of pins 13 and 14 of the "K" connector. On the bottom of the board, solder the new resistor from the inside end (the end farthest from the "K" connector) of R110, to pin 12 of the "K" connector (ground).

However, on older boards, that have discrete resistors instead of RN1, R110 is located about ½ inch down from the "K" connector and your new resistor should be soldered to the outside end (nearest board edge) of R110 and Pin 12 of the "K" connector. This completes the modification.

I am using this port to drive an Epson MX-80 printer, which is equiped with the 2K buffered serial interface. I have connected the "Busy" signal, pin 20 of the printer, to pin "K" of the "A" connector as prescribed by Browns Extended-Sym Basic. This allows me to operate the KTM-2/80 at 4800 baud and the printer at 2400 baud. It works GREAT!

Before I connected the "Busy" signal I was operating the printed at 1200 baud. This worked OK for short listings but for longer ones, above 2.5K or so it was too fast and I had to reduce it to 600 baud.

I like the serial printer interface because it frees the ports for other purposes.

Harold Hansen

Harold Hansen

A BASIC DISASSEMBLER

Here's a handy little utility program for BASIC users who have a once-in-a-while need to use a disassembler to help them debug a USR function, or perhaps make a few minor changes. Dean Garth, its author, would like to organize a users' group in his area. His address is 28619 Golden Meadow Drive, Rancho Palos Verdes, CA 90274.

The workings of the disassembler are easy to figure out. It is a so-called "table-driven" disassembler, and by changing the table entries it can be made to work with any micro, such as, for example, the 8049 as used in the Epsons. Following the LISTing is a disassembly of a portion of SUPERMON, obtained by answering the START ADDRESS ? and END ADDRESS ? questions by &"8000", and &"80FF", respectively.

```
100 REM """"""""""""""""""""""""""""
110 REM "  A BASIC DISASSEMBLER  "
120 REM "        DEAN GARTH        "
130 REM """"""""""""""""""""""""""""
140 DIM H$(15),A(15),B(15)
150 FOR K=0 TO 15: READ H$(K),A(K),B(K):NEXT K
160 DIM C$(15,15)
170 FOR K=0 TO 14
180 FOR J=0 TO 15
190 READ C$(J,K)
200 NEXT J
210 NEXT K
220 INPUT "START ADDRESS ? ";X
230 INPUT "END ADDRESS ? ";Y
240 PRINT
250 PRINT
260 PRINT "ADRESS";TAB(9);"NMEMONIC";TAB(21);"OPCODE"
270 PRINT "----------------------------"
280 IF X>0 THEN 300
290 X=65536+X
300 IF Y>0 THEN 320
310 Y=65536+Y
320 C=PEEK(X)
330 A1=INT(X/4096): B1=X-A1*4096
340 C1=INT(B1/256): D1=B1-C1*256
350 E1=INT(D1/16): F1=D1-E1*16
360 PRINT H$(A1);H$(C1);H$(E1);H$(F1);TAB(9);
370 E2=INT(C/16): D2=C-E2*16
380 IF D2>0 THEN 400
390 F=A(E2)-1: GOTO 490
400 IF D2>6 THEN 420
410 F=1: GOTO 490
420 IF D2>8 THEN440
430 F=0:GOTO 490
440 IF D2>9 THEN 460
450 F=B(E2)-1: GOTO 490
460 IF D2>10 THEN 480
470 F=0: GOTO 490
480 F=2: GOTO 490
490 PRINT C$(E2,D2);TAB(21);H$(E2);H$(D2);: IF F=0 THEN 550
500 FOR K=1 TO F
510 C=PEEK(X+K)
520 D3=INT(C/16): E3=C-D3*16
530 PRINT SPC(1);H$(D3);H$(E3);
540 NEXT K
550 PRINT
560 X=X+F+1
570 IF X>Y THEN 950
580 GOTO 320
590 DATA 0,1,2,1,2,3,2,3,2,3,2,3
600 DATA 4,1,2,5,2,3,6,1,2,7,2,3
610 DATA 8,0,0,9,2,3,A,2,2,B,2,3
620 DATA C,2,2,D,2,3,E,2,2,F,2,3
630 DATA BRK,BPL,JSR,BMI,RTI,BVC,RTS,BVS, ,BCC,LDY IMM
640 DATA BCS,CPY IMM,BNE,CPX IMM,BEQ,ORA IND X,ORA IND Y
650 DATA AND IND X,AND IND Y,EOR IND X,EOR IND Y
660 DATA ADC IND X,ADC IND Y,STA IND X,STA IND Y
670 DATA LDA INDX,LDA IND Y,CMP IND X,CMP IND Y
680 DATA SBC IND X,SBC IND Y, , , , , , , , , ," "
690 DATA LDX IMM, , , , , , , , , , , , , , ," "
700 DATA " ", , , , , ,BIT Z PAGE, , , , ," "
710 DATA STY Z PAGE, STY Z PAGE X,LDY Z PAGE,LDY Z PAGE X
720 DATA CPY Z PAGE, ,CPX Z PAGE, ,ORA Z PAGE
730 DATA ORA Z PAGE X,AND Z PAGE,AND Z PAGE X,EOR Z PAGE
740 DATA EOR Z PAGEX,ADC Z PAGE,ADC Z PAGE,STA Z PAGE
750 DATA STA Z PAGE X,LDA Z PAGE, LDA Z PAGE X
760 DATA CMP Z PAGE, CMP Z PAGE X,SBC Z PAGE,SBC Z PAGE X
770 DATA ASL Z PAGE,ASL Z PAGE X,ROL Z PAGE,ROL Z PAGE X
780 DATA LSR Z PAGE,LSR Z PAGE X,ROR Z PAGE,ROR Z PAGEX
790 DATA STX Z PAGE,STX Z PAGE Y,LDX Z PAGE,LDX Z PAGE Y
800 DATA DEC Z PAGE,DEC Z PAGE X,INC Z PAGE,INC Z PAGEX
810 DATA " ", , , , , , , , , , , , , , ," "
820 DATA PHP,CLC,PLP,SEC,PHA,CLI,PLA,SEI,DEY,TYA,TAY,CLV
830 DATA INY,CLD,INX,SED,ORA IMM,ORA ABS Y,AND IMM,AND IMM Y
840 DATA EOR IMM,EOR ABS Y,ADC IMM,ADC ABS Y, ,STA ABS Y
850 DATA LDA IMM,LDA IMM Y,CMP IMM,CMP ABS Y,SBC IMM,SBC ABS Y
860 DATA ASL A, ,ROL A, ,LSR A, ,ROR A, ,TXA,TXS,TAX,TSX
870 DATA DEX, ,NOP, , , , , , , , , , , , , ," "
880 DATA " ", , ,BIT ABS, ,JMP ABS, ,JMP IND, ,STY ABS, ,LDY ABS
890 DATA LDY ABS X,CPY ABS, ,CPX ABS, ,ORA ABS,ORA ABS X
900 DATA AND ABS,AND ABSX,EOR ABS,EOR ABS X,ADC ABS,ADC ABS X
910 DATA STA ABS,STA ABS X,LDA ABS,LDA ABS X,CMP ABS,CMP ABS X
920 DATA SBC ABS,SBC ABS X,ASL ABS,ASL ABS X,ROL ABS,ROL ABS X
930 DATA LSR ABS,LSR ABS X,ROR ABS,ROR ABS X,STX ABS, ,LDX ABS
940 DATA LDX ABS Y,DEC ABS,DEC ABS X,INC ABS,INC ABS X
950 END
```

```
ADRESS   NMEMONIC    OPCODE
----------------------------
8000     JMP ABS     4C 7C 8B
8003     JSR         20 FF 80
8006     JSR         20 4A 81
8009     JSR         20 71 81
800C     JMP ABS     4C 03 80
800F     PHP         08
8010     PHA         48
8011     TXA         8A
8012     PHA         48
8013     TSX         BA
8014     LDA ABS X   BD 04 01
8017     AND IMM     29 10
8019     BEQ         F0 07
801B     PLA         68
801C     TAX         AA
801D     PLA         68
801E     PLP         28
801F     JMP IND     6C F6 FF
8022     PLA         68
8023     TAX         AA
8024     PLA         68
8025     PLP         28
8026     JMP IND     6C F8 FF
8029     JSR         20 86 8B
802C     SEC         38
802D     JSR         20 64 80
8030     LDA IMM     A9 31
8032     JMP ABS     4C 53 80
8035     PHP         08
8036     JSR         20 86 8B
```

TOM GETTY'S "3-D TIC-TAC-TOE"

Tom Gettys is an extremely talented and versatile Computer Scientist. Without his very close collaboration SYM-PHYSIS would never have been born. To give you just one example of how prolific he is we reproduce below the directory of a disk (HDE/FODS) which he gave us more than a year ago. The "." which appears before each five character file name (and the starting addresses Ø201) indicates that these files are written in BAS-1 (with disk interface added, of course). This disk is but one of many in his disk library.

To give you an idea of the quality of his programming we also are publishing his program ".3D", which is a three dimensional Tic-Tac-Toe written for the SYM with KTM-2/8Ø. We wish we could reproduce the opening graphics on the Epson but the KTM's cursor control ESC sequences "don't compute" on the Epson (incidentally, Tom showed us recently some high resolution Epson graphics he had produced with his Graftrax 8Ø driver routine).

Following the opening graphics, four four-by-four grids are drawn on the screen and the computer asks: "Who gets to move first?" If your answer begins with "Y" (for "YOU", meaning, in this case, the computer), it plays first. The computer plays a strong game; you may have to study the implemented algorithm if you want to increase your chances of winning!

It's only a very minor point, of course, but notice the "pretty-printing" format Tom uses, especially the nested FOR . . . NEXT loops in lines 7Ø8Ø through 711Ø. Tom has also provided us with a very useful utility, "PAC", which removes all "null" lines, "surplus" spaces, and REMs (make certain first that you never GOTO a line beginning with a REM!) from a BASIC program, to allow more "RUNning" room.

We have often asked Tom to compile his best SYM software into book form, but he replies that no one would really be interested. We'll keep working on him!

```
>DC DIR 2   Disk No. 5Ø - Miscellaneous Gettys' BASIC Programs
Ø1 .MULT    Ø2Ø1 Ø9D6 Ø1 Ø1      Ø2 .BIO     Ø2Ø1 Ø7F5 Ø2 Ø1
Ø3 .EVEN    Ø2Ø1 Ø4DB Ø2 13      Ø4 .RESEQ   Ø2Ø1 Ø516 Ø3 Ø3
Ø5 .FIND    Ø2Ø1 Ø3D4 Ø3 1Ø      Ø6 .FFT     Ø2Ø1 ØB46 Ø3 14
Ø7 .HANOI   Ø2Ø1 ØAA5 Ø5 Ø1      Ø8 .WARI    Ø2Ø1 Ø4FF Ø6 Ø3
Ø9 .PLOT    Ø2Ø1 Ø76C Ø6 Ø9      1Ø .PLOT1   Ø2Ø1 Ø7E8 Ø7 Ø4
11 .PLOT2   Ø2Ø1 Ø8AA Ø7 16      12 .PLOT3   Ø2Ø1 Ø8A1 Ø8 14
13 .DEPTH   Ø2Ø1 ØCD4 Ø9 12      14 .OTHEL   Ø2Ø1 ØC76 11 Ø2
15 .THINK   Ø2Ø1 ØC52 12 Ø7      16 .DEMOS   Ø2Ø1 ØE8D 13 12
17 .PLOTS   Ø2Ø1 ØA2B 15 Ø6      18 .REVRS   Ø2Ø1 Ø33F 16 Ø7
19 .TREE    Ø2Ø1 Ø482 16 1Ø      2Ø :AUX     Ø2ØØ ØC7D 16 16
21 .PRIM1   Ø2Ø1 Ø29E 18 Ø5      22 .PRIM'   Ø2Ø1 Ø47A 18 Ø7
23 .PRIM2   Ø2Ø1 Ø3F3 18 12      24 .BINOM   Ø2Ø1 Ø657 18 16
25 .SAMPL   Ø2Ø1 Ø385 19 Ø9      26 .QKSRT   Ø2Ø1 Ø863 19 13
27 .PRIME   Ø2Ø1 Ø56Ø 2Ø 1Ø      28 .LIFEØ   Ø2Ø1 Ø6DF 21 Ø1
29 .LIFE    Ø2Ø1 1192 21 11      3Ø .LIFER   Ø2Ø1 Ø9Ø8 23 11
31 .3D      Ø2Ø1 19D4 24 1Ø      32 .PIMS    Ø2Ø1 179B 27 1Ø
33 .STATS   Ø2Ø1 Ø41C 3Ø Ø6      34 :LIFE    Ø2ØØ 16D1 3Ø 11

NEXT: T33 SØ5
```

NOTE: File 34 is RAE source code for a ML version of "The Game of Life", very much faster than the BASIC version.

```
1ØØØ DIM PA%(63,6), VA%(75), SQ%(63), WN(3)
1Ø1Ø :
1Ø2Ø REM          Define the KTM 2 display control constants
1Ø3Ø :
1Ø4Ø ES$=CHR$(27)
1Ø5Ø ER$=ES$+"R" : DR$=ES$+"r"
```



```
1Ø6Ø EG$=ES$+"G" : DG$=ES$+"g"
1Ø7Ø CR$=ES$+"+" : CA$=ES$+"="
1Ø8Ø CS$=ES$+"E" : HM$=ES$+"H"
1Ø9Ø EL$=ES$+"K" : ES$=ES$+"J"
11ØØ :
111Ø :
112Ø REM          MAIN CONTROL STRUCTURE
113Ø :
114Ø PRINT CS$
115Ø GOSUB 9ØØØ       REM      display the game name
116Ø GOSUB 7ØØØ       REM      generate the data base constants
117Ø PRINT CS$
118Ø GOSUB 6ØØØ       REM      display the playing board
119Ø :
12ØØ PRINT
121Ø INPUT "Who gets to move first? "; A$
122Ø IF LEFT$(A$,1)="Y" THEN PA=-1 : GOTO 146Ø
123Ø :
124Ø REM          Get and check player's move for errors
125Ø :
126Ø PRINT
127Ø INPUT "What is your move? "; BD, SQ
128Ø MV=16*(BD-1)+SQ-1
129Ø IF BD<1 OR BD>4 THEN PRINT "Illegal board number" : GOTO 126Ø
131Ø IF SQ<1 OR SQ>16 THEN PRINT "Illegal square number" : GOTO 126Ø
132Ø IF SQ%(MV)<>Ø THEN PRINT "That's already occupied" : GOTO 126Ø
133Ø :
134Ø REM          Display move and update data base
135Ø :
136Ø MK$="X" : GOSUB 226Ø
137Ø PRINT ES$
138Ø SQ%(MV)=1 : PA=-1
139Ø FOR I=Ø TO 6
14ØØ :    SP=PA%(MV,I)
141Ø :    IF SP=-1 THEN 146Ø
142Ø :    VA%(SP)=VA%(SP)+1
143Ø :    IF VA%(SP)=4 THEN 2Ø4Ø
144Ø :    IF VA%(SP)=3 THEN PA=SP
1445 NEXT
145Ø :
146Ø REM          Check for a win by the computer
147Ø :
149Ø FOR I=Ø TO 75
152Ø :    IF VA%(I)=15 THEN WN=-1 : PA=I : GOTO 155Ø
154Ø NEXT
155Ø IF PA<>-1 THEN GOSUB 21ØØ : MV=EM : GOTO 184Ø
156Ø :
157Ø REM          Determine computer's move
158Ø :
159Ø MX=Ø : MC=-1 : MV=-1
16ØØ FOR I=Ø TO 63
161Ø :    IF SQ%(I)<>Ø THEN 178Ø
162Ø :    SV=Ø
163Ø :    FOR J=Ø TO 6
164Ø :        PA=PA%(I,J)
165Ø :        IF PA=-1 THEN 173Ø
166Ø :        VA=VA%(PA)
167Ø :        IF VA=Ø THEN 171Ø
168Ø :        IF VA<5 THEN SV=SV+VA*VA : GOTO 171Ø
169Ø :        QU=INT(VA/5)
17ØØ :        IF VA=5*QU THEN SV=SV+QU
171Ø :    NEXT J
172Ø :
173Ø :    IF SV<MX THEN 178Ø
174Ø :    PC=SGN(PA%(I,6))
175Ø :    IF SV>MX OR PC>MC THEN 177Ø
```

```
1760 :   IF PC<MC OR RND(1)<Ø.5 THEN 178Ø
1770 :   MX=SV : MV=I : MC=PC
178Ø NEXT I
179Ø :
18ØØ IF MV=-1 THEN PRINT "It looks like a cat's game!" : END
181Ø :
182Ø REM          Update data base
183Ø :
184Ø SQ%(MV)=5
185Ø FOR I=Ø TO 6
186Ø :   PA=PA%(MV,I)
187Ø :   IF PA=-1 THEN 191Ø
188Ø :   VA%(PA)=VA%(PA)+5
189Ø NEXT
19ØØ :
191Ø REM          Show player our move
192Ø :
193Ø MK$="O" : GOSUB 226Ø
194Ø PRINT "I moved to board"; BD; " square"; SQ
195Ø IF WN=Ø THEN 124Ø
1955 PRINT : PRINT "...and winning!!!"
196Ø MK$=EG$+CHR$(96)+DG$
198Ø FOR I=Ø TO 3
199Ø :   MV=WN(I)
2ØØØ :   GOSUB 226Ø
2Ø1Ø NEXT
2Ø2Ø PRINT : PRINT : END
2Ø3Ø :
2Ø4Ø PRINT : PRINT "You won!!!"
2Ø45 PA=SP
2Ø5Ø GOSUB 21ØØ : GOTO 196Ø
21ØØ :
211Ø :
212Ø REM          This subroutine returns the number of the remaining
213Ø REM          empty square (I) in the path MV%.
214Ø :
215Ø K=Ø
216Ø FOR I=Ø TO 63
217Ø :   FOR J=Ø TO 6
218Ø :      SP=PA%(I,J)
219Ø :      IF SP=-1 THEN 224Ø
22ØØ :      IF SP<>PA THEN 223Ø
221Ø :      IF SQ%(I)=Ø THEN EM=I
222Ø :      WN(K)=I : K=K+1 : IF K=4 THEN RETURN
223Ø :   NEXT J
224Ø NEXT I
225Ø :
226Ø :
227Ø REM          This subroutine enters a mark (MK$) on the board
228Ø REM          in square SQ%.
229Ø :
23ØØ BD=INT(MV/16)+1
231Ø SQ=MV-16*(BD-1)+1
232Ø QU=INT((SQ-1)/4)
233Ø MO=SQ-4*QU
234Ø Y=2*QU+34
235Ø X=18*BD+4*MO+12
236Ø PRINT CA$+CHR$(Y)+CHR$(X)+MK$;
237Ø PRINT CA$+CHR$(43)+CHR$(33)
238Ø RETURN
6ØØØ :
6Ø1Ø :
6Ø2Ø REM          Print a new board
6Ø3Ø :
6Ø4Ø FOR I=1 TO 4 : PRINT TAB(18*I-13); "Board"; I; : NEXT
6Ø5Ø :
```



```
6Ø6Ø PRINT ER$; EG$
6Ø7Ø PRINT BD$(Ø)
6Ø8Ø FOR I=Ø TO 2 : PRINT BD$(1) : PRINT BD$(2) : NEXT
6Ø9Ø PRINT BD$(1) : PRINT BD$(3)
61ØØ :
611Ø PRINT DR$; DG$
612Ø RETURN
7ØØØ REM          For each square, compute or read in the
7Ø1Ø REM          number of any path involving that square.
7Ø2Ø :
7Ø3Ø FOR I=Ø TO 3
7Ø4Ø :   READ BD$
7Ø5Ø :   BD$(I)=BD$+" "+BD$+" "+BD$+" "+BD$
7Ø6Ø NEXT
7Ø7Ø :
7Ø8Ø FOR I=4 TO 6
7Ø9Ø :   FOR J=Ø TO 63
71ØØ :      PA%(J,I)=-1
711Ø :   NEXT
712Ø NEXT
713Ø :
714Ø FOR I=Ø TO 15
715Ø :   QU=INT(I/4)
716Ø :   PA%(I,Ø)=QU
717Ø :   PA%(I,1)=I-4*QU+4
718Ø :   PA%(I,2)=I+4Ø
719Ø :   READ PA%(I,3)
72ØØ NEXT
721Ø :
722Ø FOR I=16 TO 63
723Ø :   QU=INT(I/16)
724Ø :   MO=I-16*QU
725Ø :   PA%(I,Ø)=PA%(MO,Ø)+1Ø*QU
726Ø :   PA%(I,1)=PA%(MO,1)+1Ø*QU
727Ø :   PA%(I,2)=PA%(MO,2)
728Ø :   READ PA%(I,3)
729Ø NEXT
73ØØ :
731Ø FOR I=Ø TO 15
732Ø :   READ S
733Ø :   FOR J=4 TO 6
734Ø :      READ PA%(S,J)
735Ø :   NEXT
736Ø NEXT
737Ø :
738Ø RETURN
739Ø :
8ØØØ DATA "AOOOyOOOyOOOyOOOB", "V   V   V   V   V"
8Ø1Ø DATA "hOOOpOOOpOOOpOOOi", "COOOxOOOxOOOxOOOD"
8Ø2Ø :
8Ø3Ø DATA  8,6Ø,62, 9,57, 8, 9,65,58, 9, 8,66, 9,61,63, 8
8Ø4Ø DATA 18,69,7Ø,19,68,18,19,71,72,19,18,75,19,73,74,18
8Ø5Ø DATA 28,7Ø,69,29,72,28,29,75,68,29,28,71,29,74,73,28
8Ø6Ø DATA 38,61,63,39,65,38,39,57,66,39,38,58,39,6Ø,62,38
8Ø7Ø :
8Ø8Ø DATA Ø,56,68,69, 3,64,7Ø,71, 12,59,72,73, 15,67,74,75
8Ø9Ø DATA 21,56,57,6Ø, 22,62,64,65, 25,58,59,61, 26,63,66,67
81ØØ DATA 37,61,65,67, 38,57,59,63, 41,6Ø,64,66, 42,56,58,62
811Ø DATA 48,67,7Ø,72, 51,59,69,75, 6Ø,64,68,74, 63,56,71,73
812Ø :
9ØØØ :
9Ø1Ø :REM          Output the game banner
9Ø2Ø :
9Ø3Ø PRINT ER$; EG$
9Ø4Ø T%=22
9Ø5Ø PRINT TAB(T%);
```

```
 Y!!!!oZ              \''''Z
o     Zo              !     \
1Ywwwk !              ! jwwwZ\
    Y! !    Y!!Z      ! a   !g
    1q !    1qqm      ! a   !g
Yoqqqm !              ! a  Y\g
ao'''' Y              ! !!!\Yo
\qqqqqo               Zqqqqqo

tttJIt tttJ    tttJIttt tttJ    tttJIttJIttt
Ld L d d  L rr Ld LdttL d  L rr Ld Ld  dds K
 d  IK Kttd     d  d  d Kttd     d  dttddttL
```

NOTE: On the KTM-2/8Ø the above gibberish reads, in large, shaded, three dimensional appearing, block characters, on two lines:

3 - D

TIC-TAC-TOE

```
9060 PRINT " Y!!!!"+DR$+"o"+ER$+"Z             "+DR$+"\````"+ER$+"Z"
9070 PRINT TAB(T%);
9080 PRINT "o      "+DR$+"Zo"+ER$+"             !      \"
9090 PRINT TAB(T%);
9100 PRINT "l"+DR$+"Ywww"+ER$+"k !             ! j"+DR$+"www"+ER$+"Z\"
9110 PRINT TAB(T%);
9120 PRINT "     Y! !    Y!!Z    ! a    !"+DR$+"g"+ER$
9130 PRINT TAB(T%);
9140 PRINT "     lq !    lqqm    ! a    !"+DR$+"g"+ER$
9150 PRINT TAB(T%);
9160 PRINT "Y"+DR$+"o"+ER$+"qqqm !             ! a  Y"+DR$+"\g"+ER$
9170 PRINT TAB(T%);
9180 PRINT "a"+DR$+"o```` Y"+ER$+"             ! !!!"+DR$+"\Y"+ER$+"o"
9190 PRINT TAB(T%);
9200 PRINT "\qqqqqo              "+DR$+"Z"+ER$+"qqqqqo"
9210 :
9220 T%=13
9230 PRINT : PRINT
9240 PRINT TAB(T%);
9250 PRINT "tttJIt tttJ    tttJIttt tttJ    tttJIttJIttt"
9260 PRINT TAB(T%);
9270 PRINT "L"+DR$+"d "+ER$+"L d d  L rr L"+DR$+"d "+ER$+"L";
9280 PRINT DR$+"d"+ER$+"tt"+DR$+"L "+ER$+"d  L rr L"+DR$+"d ";
9290 PRINT ER$+"L"+DR$+"d  "+ER$+"d"+DR$+"d"+ER$+"s K"
9300 PRINT TAB(T%);
9310 PRINT DR$+" d  "+ER$+"I"+DR$+"K K"+ER$+"ttd     "+DR$+"d  d  d K";
9320 PRINT ER$+"ttd      "+DR$+"d  d"+ER$+"ttd"+DR$+"d"+ER$+"tt"+DR$+"L"
9330 :
9340 PRINT DR$; DG$
9350 PRINT CA$+CHR$(32+23)+CHR$(32+0);
9360 T%=23
9370 PRINT TAB(T%); "Copyright 1980, thomas gettys";
9380 PRINT CA$+CHR$(32+1)+CHR$(32+24);
9390 RETURN
```

JACK BROWN'S VISIT

Jack Brown (Mr. Saturn Software) and family visited with us overnight on their way to the 7th West Coast Computer Faire (we first met Jack in person two years ago, when he visited with us overnight on his way to the 5th West Coast Computer Faire!). History was repeating itself, but on this visit he brought even more wonderful goodies than last time. We describe one of them below, in connection with Pascal.

X-RAY

Saturn Software's latest products, SYM-Pascal (Release 2.0), and X-RAY (Release 2.0) are by Ralph Dean, and by Ralph Dean & Jack Brown, respectively. Pascal needs no introduction, and we did mention earlier that SYM-Pascal is integer only.

We do not find this a real limitation, since most of our interests do not require floating point arithmetic. Hal Chamberlin's MTU Advanced Music Synthesis package (in machine language) and Jack Brown's "Turtle-Graphics" (in FORTH) both use simple algorithms and simple table lookups to do fairly sophisticated tasks such as Fourier Synthesis, and the like, with the 16-bit precision that is more than adequate for the jobs, and, much more important, fast enough for the jobs! The lack of floating point is a challenge, not a handicap, to a skilled programmer.

We also mentioned earlier that Pascal operates "under" RAE. This means, in effect, that Pascal source files are actually RAE files, and that all RAE commands, including disk linkages and other add-ons, are built-in to Pascal. Now comes the exciting part!

X-RAY does for RAE even more than Brown's Enhancements do for BASIC. Among other things, it adds the following commands to RAE (and hence also to Pascal):

ADdress line# (gives RAM location of start of line), APpend line# (see below), EXecute addrs (see below), FL(for FiLe) addrs (lets you keep several files going), REstore (recover after clear or cold start), SAve ID addrs1 addrs2 (for object code!), TApe 1, TApe 2 (to change tape speed to single or double!), and an improved EDit line#.

There is not enough space to describe all of these new commands in the detail they deserve, but here are very brief explanations of two of them:

EXecute is similiar in function to .E in SUPERMON, in that a sequence of RAE commands can be prestored in RAM, and then EXecuted by giving the address as a parameter. Since all DOS (Disk Operating System) commands are callable from RAE, any desired DOS sequence may also be EXecuted.

APpend lets you "scroll" through a text file by holding down the RETURN key. The vertical scrolling stops with the cursor at the end of a line when the RETURN key is released. The direction of "vertical" scrolling is reversed with CTRL U (up), and CTRL D (down). The cursor may be scrolled "horizontally" in wrap-around fashion through a line with CTRL H (BS or left), and CTRL I (TAB or right), and characters may be inserted, deleted, replaced, etc., exactly as if you were using a memory mapped display. Nearly any terminal which responds to CTRL H by backspacing can be used. The command is called APpend because its most "natural" function (since the cursor waits at the end of a line) is to append comments to source code.

X-RAY makes text editing and source code commenting almost a pleasure. We originally purchased our SYM mainly because RAE was "promised" as an accessory, and we knew that RAE would be great. We are continually being surprised by RAE becoming even greater and greater with such add-ons as X-RAY. We never dreamed it could become this good!

SYM DISK OPERATING SYSTEMS (continued from page 11-2)

disk "primitives" may let you speed up your system significantly. MTU's CODOS system comes with a built-in utility to customize your system, bless them! The program lists the default value of each built-in delay, and waits for either a <cr> or the entry of a better value, which you obtain from the spec sheet on your disk drive.

DRIVES

The 5 1/4 inch drives come in 35 track (Shugart SA-400) and 40 track versions (Shugart SA-400L). We mention the Shugart brand name here, not as a specific recommendation, but because their models are, in a way, de facto "standards". These are single-sided drives. The Shugart SA-450 types are 40 track double-sided drives. All of these drives may be operated either in single- or double-density mode; this choice depends primarily on the capability of the disk controller and which mode(s) the software supports. Some 5 1/4 inch drives reduce the intertrack spacing from the Shugart "standard" to provide twice as many tracks in the same space.

The Shugart SA-800/801 is the single-sided 8 inch "prototype", and the Shugart SA-850/851 is the double-sided version. The SA-800 may be used only with soft-sectored disks, the SA-801 with hard-sector as well. The SA-850 and SA-851 differ from each other in this same regard. The 801 (and 851) may be used with soft-sectored disks and a 34-wire cable instead of the standard 50-wire cable (e.g., FODS 8 inch systems).

All Shugart compatible disk drives accept the same types of power and controller cable connectors. All have an almost bewildering selection of "options", selectable by cutting traces and/or adding jumpers, and must be configured to meet any special requirements imposed by the controller and software. The factory installed jumpers are for single drive systems, and must be modified for multiple drive systens. All 5 1/4 drives require +12 V and +5 V regulated, and all 8 inch drives require 110 V AC (US), +24 V, and +5 V. Some 8 inch drives also require -5 V.

DISKS

The most "universal" type of disk is the soft-sectored disk, which has only a single hole to mark the "origin" for each revolution. Hard-sectored disks may have 10, 16, or 32 additional holes to mark sector boundaries. We will consider only soft-sectored disks here. The least expensive disks are the single-sided, single-density ones. This does not necessarily mean that they cannot be used at double density, or that the second side (which is the "top" side, by the way) is not usable. By cutting out a second "write-protect" notch (on 5 1/4 inch only), and punching a second "track-hole" in the protective case, most single-sided disks may be flipped over to use either side. It is claimed by some that this is not a good idea, but Apple II owners do this all the time, since the Drive II does not even require the punching of a second track-hole.

Double-sided disks have the track hole in a slightly different location and will not work in a single-sided drive. Double-sided drives will accept either type of disk and can be jumpered to work either as a double-sided drive when double-sided disks are used, or as, in effect, two distinct single-sided drives, callable separately if a single-sided disk is used.

HUDSON DIGITAL ELECTRONICS' - FILE ORIENTED DISK SYSTEM (FODS)

We now have two, and will soon have three, dual 5 1/4 inch SYM-FODS systems operational, and in almost constant use, with no preventive maintenance. One system is 35 track, the other two are 40 track. Disks with 35 or fewer tracks "in-use" are freely interchangeable. FODS was our very first DOS and we learned a lot from it. FODS is strictly a single-density, single-sided system, as it stands. We have also had on hand, for nearly a year, a FODS 8 inch controller, which operates at twice the clock rate (the only difference), and we think that the 5 1/4 software could be modified to support double-density operation with the 8 inch controller.

While we have 7 mini-floppy (5 1/4 inch) drives around, until last week we had only one 8 inch drive, and that was "permanently" on our CODOS system. We will soon be packaging a self-contained dual 8 inch drive unit on a wheeled table which can be rolled around to service three computer systems. Each will have an extension cable dangling from it so that we can plug the drive system into it. One computer system will have both 8 and 5 1/4 inch FODS controllers installed, one at $A888, the other at $A880. Both may be co-resident and either booted up separately. This will permit us to support both FODS formats.

The FODS controllers are about 4 x 6 inches in size, and, being built for the KIM-4 bus, require unregulated +8 V, -8 V, and +16 V. The on-board regulators may be shorted across, and regulated +5 V, -5 V, and +12 V supplied instead. Interface to the SYM is from the Expansion Connector to a VIA on the Controller Card, which shares space with the SYM's "extra" VIA at $A800. Since FODS needs nearly 4K of RAM at $7000 for its own use, and perhaps 2K of RAM in the $6000 block, external memory is required. FODS goes very well with the 32K Beta DRAM board, which needs only +5 V, but generates its own +12 V and -5 V on-board, enough to support the FODS controller, as well as itself.

FODS stores its files sequentially on the disk, never over-writing existing files, and thus must be periodically "packed" to compact the active files. This may not be the most "popular" technique, but it is good insurance. Even when you "clear" a disk by deleting the directory (DEL INDEX!), the data is still easily recoverable by indirect methods.

UK SYM-DOS

Even though FODS source code is not published, a group of SYMmers in the United Kingdom has disassembled and studied the inner workings of FODS. As a result they have generated a new DOS (called by them UK SYM-DOS), wholly compatible with the HDE Controller and existing FODS disks. In UK SYM-DOS, they have compacted the code, speeded up the PAK operation, and worked out a way to squeeze all of the utilities except #FM (ForMat) into the same 4K as the main portion of the DOS, thus speeding up all utilities by avoiding the access to the disk to download and overlay the utilities.

They have added a number of new instructions which permit accessing individual sectors, overwriting files if desired (avoiding the need to pack), creating EXECute files (all SYM-DOS command names are four characters long, instead of three, as in FODS), etc. Best of all, it is available in well commented source code form, and studying the source is a good way to learn how disk systems really work.

While UK SYM-DOS is descended from FODS it is a wholly new creation. The authors deserve commendation for making it 100% compatible with existing FODS files. We feel it is an enhancement to FODS, and cannot hurt FODS sales in any way. In fact, UK SYM-DOS is available only on a FODS compatible disk, which requires that the purchaser have a FODS system to begin with.

MTU'S CHANNEL ORIENTED DISK OPERATING SYSTEM (CODOS)

We have been using Micro Technology Unlimited's CODOS for over a year on our high resolution graphic system, and CODOS forms the basis for Jack Gieryic's graphics as well. Jack Brown also has both CODOS and FODS. Both systems are excellent; they represent two completely different programming philosophies and approaches to system design. The "channel orientated" concept (no time to define it here) is a very elegant technique for I/O management, which takes a while to get used to, but makes CODOS a really versatile instrument.

Having only a single-drive system available kept us from providing the SYM/CODOS community with full support. There was just no way to automate the copying of disks (for distribution) on a single-drive system. This situation will be resolved by this summer, and we will provide full SYM/CODOS support.

We should point out that CODOS supports only 8 inch drives, up to four of them, in ddouble-density mode. Double-sided drives are supported (if double-sided disks are used, but then not all users can read your disks), and, with four double-sided drives you will have over 4 Megabytes of on-line storage. MTU recommends the Qume DataTrak 8 to go with CODOS, and we have followed their recommendation.

Because we had only a single Qume, and were not yet certain what our second drive would be, we never bothered to optimize our CODOS system to the Qume. When Jack Brown heard our disk drives chugging along, he asked if he could optimize one of our CODOS disks for us; he did and we could not only see the speed improvement, the system now purred, rather than chugged. Things did not have to hurry-up and then stop to wait for the software delays to time out. Except for the CODOS.Z program itself, all disk programs are independent of the optimization parameters. All MTU boards use the KIM-1 pinout (same as the SYM E-connector) and use small amounts of unregulated +8 V and +16 V.

Who should consider CODOS? If you want high resolution B/W graphics you will need the 8K RAM Visible Memory. Both it and the SYM fit neatly into an MTU Card Rack. Then the CODOS controller is actually part of a 16K Dynamic RAM board that fits into the same card rack. You can buy it just for the 16K memory expansion, and add the disk drive(s) at a later date. You will then have one of the fastest, highest capacity, floppy disk systems available for any microcomputer system, bar none. Consider CODOS too, if your data base is likely to be on the largish side, or extremely rapid access is needed.

We will be supporting the SYM/Visible Memory software and the RAE/CODOS interface. We recommend that the BASIC/CODOS interface be handled through Jack Brown; that way all of his BASIC enhancements will be an integral part of the interface.

THE SYM USERS' GROUP FLOPPY DISK CONTROLLER (FDC-1)

When we accepted Synertek Systems Corporation's offer to be allowed to "adopt" the FDC-1 as "our baby" if we promised to support it properly, we did not fully realize what the "child-support" involved, or how much of an initial outlay of time and money would be required.

If we are to fully support FDC-1 with a strong software base, it would be stupid to distribute the software on cassette. This means that we will have to set up two 32K dual disk drive systems, one 8 inch, and one 5 1/4 inch, to do the job. That works out to lots of dollars. A dollar saving alternative is a single system on which we change controllers to switch from one size drive to the other. We'll start out this way.

So much for the hardware costs. Now for the software costs!

Only a preliminary partial manual in rough draft form now exists. This must be completed and edited, and we'll have to hire a typist and train him/her to "SWP" out the final version on one of our SYMs, and hire someone to do the drawings. Then there will be the printing costs, which are very high for small runs. Should we print 500, 1000, 2000, 5000 manuals? At perhaps $3.00 per copy a run of 2000 will mean a $6000 outlay immediately.

The EPROM source code was not available to us in RAE format so we had to disassemble the object code, and use RAE to replace the meaningless labels with meaningful mnemonics. Next we will append the comments. We are very thankful for Dessaintes' Disassembler and X-RAY; without them the job would be even more tedious than it has to be. Should we supply source code on disk, with the higher medium and copy-time costs, or should we go to a printed listing, with the higher initial costs?

We tell you our problems, not to elicit your SYMpathy, but to answer, in advance, those who are wondering why it takes so long, and especially those who want to know why we (SUG) won't "honor" the price for a fully assembled and tested FDC-1 at the price originally announced by SSC.

The Synertek logo will not appear on the FDC-1, so we'll design our own. Since the FDC-1 is primarily "for SYM-1", and since fruit mames are state-of-the-art these days, should the logo read "perSYMone"?

And now, here's the bottom line:

We will be placing initial orders for circuit boards, components, and drive cables in early April, and FDC-1 kits, complete with double-drive cables will be ready for shipment by mid-June, 1982, on a first ordered, first shipped basis, at the following prices (enclose check with order):

US	$195 UPS, Street address only, not P.O. Box
Canada	$200 Airmail
Europe	$205 Airmail
Elsewhere	$209 Airmail

>>>>>Please state whether 8 inch or 5 1/4 inch cables are needed<<<<<

California residents please add 6% sales tax.
Foreign residents please advise classification for lowest duty rate.

MISCELLANIA

This is a "special" issue for those readers who do not like special issues devoted to single topics. For those who like games, there is a "3 - D TIC-TAC-TOE", and for those who feel we are biased against BASIC in favor of assembly language, there is a BASIC DISASSEMBLER, which might even arouse a latent interest in assembly language programming.

As usual, and with regret, we must apologize for not being able to answer all of the letters requesting help. Getting out two issues in the same quarter left little "free" time. We will apologize in advance for the next quarter as follows: While all letters arriving durng the period mid-April through mid-May (1982) will be opened and read by the office staff while we are visiting Australia and New Zealand, few technical questions will be answered, so we request you hold your questions till later in May.

When we return we'll start our policy of batching questions and sending them on to those who volunteer to help (see page 1). We would gladly like to hear from some of the more experienced SYMmers who would enjoy helping beginneers, and even non-beginners needing help.

For those readers who object to our use of the editorial "we" as sounding too pretentious, or pontifical, I can only reply that I was taught I should avoid the use of I as much as I could, and I don't know how I can, in all due modesty, use an "I" for an "I" when I was educated to use 'we' instead. I'm sorry about that.

JEFF LAVIN was working very hard on preparing a questionnaire to be mailed with this issue. The answers would have gone into a computerized data base which would be useful in forming "Special Interest Groups". He sent us a preliminary draft which looked great, but we were too overloaded to get it back to him in time for this issue. It will have highest priority for Issue No. 12.

There were a number of notes to go in the space below about books written by SYMmers, and products available from SYMmers. Since there is not enough space to include them all, and we don't want to upset those whose notes were not selected, we'll just leave the space blank, and get all of the notes into Issue No. 12

We will be teaching a SYM based Microprocessor Course at the University of California at Davis, Davis, CA 95616. If you are interested in more information, contact Garrett Jones, c/o University Extension, phone (916) 752-2177.

This issue should reach Australia and New Zealand about the same time we do. We are both looking forward to an Easter in autumn, and to meeting many of our readers down-under. Look for Issue No. 12 around mid- July.