SYM-PHYSIS

THE SYM-1 USERS' GROUP NEWSLETTER

VOLUME II, NUMBER 1 (ISSUE NO. 7) - SPRING 1981 (JAN/FEB/MAR)

SYM-PHYSIS is a quarterly publication of the SYM-1 Users' Group, P. O. Box 315, 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
Associate Editors: Thomas Gettys, Jack Brown
Jack Gieryic

SUBSCRIPTION RATES (1981):

USA/Canada - $10.00 for a volume of four issues. Elsewhere - $13.50. Make checks payable in US dollars to "SYM-1 Users' Group", P. O. Box 315, Chico, CA 95927, Telephone (916) 895-8751.

Issue #0, the Introductory Issue (1979), and Issues 1 through 6 (1980), are available, as a package, for $12.00, US/Canada, and $16.00, First Class/Airmail, elsewhere.

EDITORIAL POLICY

SYM-PHYSIS is not intended to be your typical periodical, to which new subscriptions begin with the current issue. Instead, new subscribers start out with Issue 0 and Issues 1 through 6, and, hopefully, do not remain beginners long. Thus, there will be no "Beginner's Corner" in each issue. Rather, we hope to increase the level of sophistication of our material, as we and you grow in experience with the SYM-1, and continue to make ever increasing demands on its performance.

We will include in each issue several program listings for both BASIC and RAE users. We will also attempt to keep readers current on what is available for the SYM-1, in the way of both hardware and software, from every source of which we know, and publish any tips or hints for improving the SYM's performance which we or our readers discover. We also hope to present concepts, ideas, thoughts, software and hardware design principles, philosophical whimsies, etc., at least some of which should be useful, to at least some of our readers, at least some of the time.

We will try for four mailings per year, with an average of 36 single spaced manuscript pages per mailing. Outside pressures may possibly force an occasional "double-issue" (as happened with 5/6 last year!).

HELP US TO HELP YOU

We are now sufficiently organized to keep up with the unexpectedly large number of letters and phone calls which arrive seven days a week, and can even get caught up with the mail backlog after a week's absence. We are even making a slight dent on the enormous backlog which accumulated before we figured out how to handle it. Please bear in mind, too, that we do have a full-time teaching position, and that publishing SYM-PHYSIS is only a leisure-time (!) activity.

There is, however, no way in which we can get back to last December's mail, so if we have not answered an earlier letter of yours, please accept our apologies, and try again. The following suggestions will help us to help you more efficiently:

Please use separate sheets of paper (each with your name and address on it!) for each of the following:

1. Requests for HELP. These will get first priority.
2. Requests for general information, no emergency.
3. Purchase requests. These too, get priority.
4. Letters of praise, condemnation, articles, ideas, etc. These we will read at leisure, for pleasure.

It will help greatly if each item is so clearly obvious as to its proper category that even a "typical" clerk, who could not care less about learning to do a job well, could sort it out into the proper file. This is one clerical task for which we still need human help. Our SYMs won't help us here!

While there is no charge for the "research" involved in getting answers to your questions, because we enjoy the learning, we do pay someone to make Xerox copies, and stuff and address the envelopes, and postage costs are rising. You can help us cover these costs by slipping a dollar or so into your envelope occasionally. Overseas currency is OK, too, since Jean loves to travel, and will find a way to spend it.

OUR SUPPORT POLICY

We will fully support all of our software products, notifying all purchasers of known buss and their fixes. When upgraded versions are available, purchasers of earlier versions will be given discounts on the new versions.

Most software presently available for distribution supports only cassette I/O, and is available on cassette. Owners of FODS systems may order disk versions. The disk versions are Load and Go, called by RUN %name. Disk drive turnoff and any pages 0 and 1 initialization are built into the programs. When we have the Disk I/O patches ready, notices will be sent to all owners of record. The patches, including source code in RAE/FODS format, will be made available for a nominal sum to cover the media, shipping, and labor costs involved.

We're even more anxious than you are to get these patches ready for use! We have modified, or are in the process of modifying, all existing packages to support FODS Disk I/O. For example RAE-1, SWP-1, and BAS-1 are fully integrated with FODS. BAS-1 now supports .CHAIN, .APPEND, and .ED (Enter Data) and .LD (Load Data) commands. These patches were written by Tom Gettys. We are currently working on the FORTH and tiny-c Disk I/O patches.

It is a real pleasure to watch RAE-1 assemble and list a 48+K source code file with .CT modified to mean Continue on Disk, or to watch SWP-1 print out a 90 page report from disk files!

MORE ON SOFTWARE 'THEFT'

We received the following post card recently, and reprint it, in its entirety, omitting only the signature:

> Dear Lux:
>
> As a result of your statements on page 4-27, I am not renewing my membership in SYM-PHYSIS. Theft is theft, regardless of whether the thief deems it "fair" or it "occurs spontaneously," and I cannot condone it with continued membership. Nor will I pay prices for software inflated by the anticipation of "sharing." Would you be "encouraged" to teach if only 1 out of 5 students paid the tuition that pays your salary?

We are truly sorry to lose the writer as a member of the SYM-1 Users' Group, because we sense from the tone of his message that he has given the matter much thought, and there is much that he could contribute to the rest of us in the way of ideas, and software, etc. We will send him a copy of this issue, so that he will know our ideas on the matter. (We have since telephoned the writer, and neither of us convinced the other, but we are still friendly!)

We have always considered ourselves to be more of a "scientist" than an "engineer" and jokingly described the former as anxious to accumulate a string of published papers (for the glory!), and the latter to acquire a string of patents (for the fiscal return). (We hope we haven't made any more enemies with this last remark!). As a scientist, we have frequently exchanged manuscripts, rough drafts, notes, etc., with others, and often sent Xeroxed articles to others marked up with our comments and questions, asking them for their ideas and suggestions on something we have seen in the literature.

We have viewed this in the spirit of "research" and information exchange, and considered it "fair use" of published research materials. I have sent copies of my published articles to colleagues, and have allowed them to include copies as appendices to reports they have submitted to their clients. The clients would never have seen the original articles, nor would they ever consider subscribing to the journals which published them; the publishers lost no income as a result of these "gift copies".

I did object once, and very strongly, too, when a "colleague" had one of my published (and copyrighted by the publisher) articles retyped, substituted his name and consulting firm's name for mine and charged one of his clients for the report he "prepared" for them!

We consider the unauthorized marketing of someone else's product as one's own as the real violation of the spirit and letter of the copyright and patent laws, not the sharing by close associates. For example, a small group of chess players might pool their funds in order to acquire all available chess programs for their mutual use. On the other hand, when a large group acts as a purchasing "collective" for the purpose of, in effect, "manufacturing" and "distributing" reproductions of a product, be it software or hardware (circuit boards are also reproducible), it becomes a commercial activity, and should be considered as such, even if it is a not-for-profit organization.

What we think we meant in the referenced statement on page 4-27, was to pick some arbitrary number, in this case five, as being a "fair" upper limit for a resource-pooling commune! Perhaps the number was too high? Too low? Or what?

A BELL FOR THE KTM-2 AND/OR KTM-2/80

You may have noticed in the KTM reference manual that pin 22 on both the Main and Aux ports of the KTM-2 (and -2/80) is labeled 'BELL'. When we first got our KTM-2, we tried hanging a small 8 ohm speaker between pin 22 and ground, with unsatisfactory results. The speaker made noise even when it was not enabled, so we gave up that idea immediately. One of our associates, Lew Davis, suggested we try instead a piezo-electric beeper. Lew also suggested that we we would get a better tone if we disabled the on-board oscillator. We then connected a Radio Shack Piezo Buzzer (273-060) between pin 22 (positive or red lead) and one of the ground pins. The combination tone of the beeper (4.8 kHz) and the lower frequency on-board oscillator was not pleasant, so we cut a trace and added a jumper to disconnect the oscillator.

RAE now signals us audibly on error messages, and also lets us know when we are near the end of an input line (72 characters). We can now PRINT CHR$(7) from BASIC, instead of LET X = USR(&"8972",0) when we want a beep. Our only complaint is that the Bell Enable signal is too long, nearly two seconds. One fix for this is to install a "flasher LED" in series with the beeper; this will produce several short beeps instead of the one long beep. The beeper volume may be reduced in intensity with either a series resistor or a piece of masking tape over the opening. To disable the oscillator cut the long trace just above R 23, and jumper the left end of this trace to the right end of the long trace just below U 38. This cut and jumper will disconnect pin 5 of U 38 from the on-board oscillator and permanently ground it.

Radio Shack also has available a much more compact, less expensive, Piezo Element, without the built-in oscillator (273-064). We tried this device also, tieing its red and blue leads together, depending on the built-in oscillator to generate the tone. The result was a very pleasant, but very quiet, low pitched buzz. We felt the volume was a little too low to alert us from across the room, but just right in a noise-free environment (i.e., no hi-fi, TV, or conversation going).

Another alternative would be to use the same type of beeper as is on the SYM-1 itself, but this is not as readily available as the Radio Shack device, and is very likely less cost effective and more troublesome to mount on the KTM because of the exposed metal contacts.

While we are looking at the KTM, let us remind you about pin 23, labeled 'DC'. This is enabled (low) by CHR$(19) and disabled by CHR$(20); these are Control-S (DC3), and Control-T (DC4), respectively. 'DC' can be used, with a suitable relay, to control the AC power to your printer, for example (NOTE: 'DC' means 'Device Control', not Direct Current!).

CONTROLLING I/O FROM BASIC

Here, slightly modified, are Andre Hoolandts' subroutines for switching between a 110 baud TTY on the 20 mA loop and a 4800 baud CRT on the RS 232 interface:

```
1000 X=USR(-29818,0):POKE 42580,208:POKE 42577,  1:RETURN
2000 X=USR(-29818,0):POKE 42580,224:POKE 42577,213:RETURN
```

The USR function is a JSR ACCESS, and 42580 and 42577 are the locations of TOUTFL ($A654) and SDBYT ($A651), respectively. The numbers poked into TOUTFL, 208 and 224, are the decimal equivalents of $D0 and $E0, respectively. It might seem that these should be $60 and $90, but not necessarily so. Note that with the values $D0 and $E0, CRT IN is enabled with TTY IN/OUT, and TTY IN is enabled with CRT IN/OUT. This permits INSTAT and TSTAT to check for the BREAK key down on either (only possible with MON 1.1, MON 1.0 only checked only the device on PB7 of the 6532, normally the CRT). Any keys on the unselected device during

an INCHR or INTCHR will, of course, produce gibberish because of the incorrect baud rate.

The $D0 for the CRT choice is much better than the SYM-1 default value of $B0, since this latter activates TTY OUT, not TTY IN, and the TTY will chatter on unintelligibly at the wrong baud rate unless it is turned off. Of course a disconnected TTY "sends" a permanent break signal, so you may prefer the $90 in many cases.

The 1 and the 213 are the decimal equivalents of $01 and $D5, for rates of 4800 and 110 baud, respectively. These may be changed to fit your own peripheral rates.

ANOTHER CASSETTE PROBLEM AND FIX

Most of the cassette read problems on the SYM-1 seem to be "fixed" by replacing C16 with a smaller capacitor (0.01 uF), aligning the heads, cleaning the pinch roller, etc. We recently met a SYM-1 whose output cassettes were unreadable on any other SYM/Recorder combination. We tried three or four SYMs and recorders to no avail. We finally decided that the output level was marginally low, and changed R88 from 470 ohms to 1.0 kohm, to approximately double the recording signal level. This worked out fine.

THE RAE USer FUNCTION

The RAE-1 USer function allows the passing of one parameter through the A Register if you so desire. Enter the following program at $0003 to check it out:

```
0003-   20 A0 8A      JSR TOUT
0006-   4C AC B0      JMP RAE.WARM
```

Call this with USer (any character), and that character will be printed. The value of the parameter passed (remember it is the ASCII equivalent of the character) can be used to select one of a number of optional subroutines.

WIGGLE YOUR CHIPS AND FLEX YOUR BOARDS

While it's a good idea from many stand-points to socket all of your chips, rather than solder them in, poor socket contacts could give rise to the same sorts of problems as cold solder joints. Several readers (as well as we, ourselves) have traced their memory and other problems back to these two sources. We suggest that you wiggle your chips in their sockets to increase contact likelihood, and flex the circuit board slightly to help locate bad solder joints. The flexing may not show you where they are, but it might "fix" them semi-permanently.

MORE ON DISK SYSTEMS AND COMMUNICATIONS

We have been studying Apple II DOS (3.2 and 3.3) to see how it works. Apple DOS is very elegant; we like best its ability to use long file names. The HDE FODS is also elegant; we prefer its command structure and syntax. DOS and FODS are both very versatile. We can not rate either as clearly superior to the other (to say nothing about CP/M for the 80-type machines and FLEX for the 6800 systems). To do so would be like comparing apples and _____ !

One feature that Apple DOS has is the ability to OPEN, WRITE, READ, and CLOSE text files to the disk, from various languages, e.g., from either Integer or Microsoft BASIC. For the uninitiated, this means that when LIST is called from BASIC, or PRint from RAE, the output may be buffered to a "named" file on the disk, instead of, or in addition to, appearing on the terminal or printer. The file is, of course, in ASCII, with each <cr> followed with a <lf>. Next, naturally, when

BASIC, or RAE, is awaiting input of a file, it must be made possible to "OPEN" the proper named file and "READ" that file instead of expecting input from the keyboard). Thus RAE can be used to prepare and edit programs for BASIC, for example.

We started to follow that path over a year ago, before we had a disk system, with the MERGE program, which dumped the ASCII listing from BASIC into high memory for later recall (from BASIC). Our next step was to allow RAE to access and redump (edited). Why did we not follow through on that? Because we felt that each higher level language should be "stand-alone"; why should a second language be required to make up for the inadequacies of a first. Jack Brown, pointed out to us about that time, that SYM-BASIC could be its own editor (more on that elsewhere).

There is however, a very valid reason for being able to dump ASCII to disk or memory, and we hope to get this going on our system during the next quarter. ASCII is the American Standard Code for Information Interchange, between computers, terminals and data banks, etc. Buffering data, in ASCII, in memory, if there is enough, otherwise on a disk, will allow you to interface your SYM to data services, time share, etc., through a modem. A number of readers are working on this. Our suggestion to those who asked was essentially as follows: Duplicate the RS232 hardware and software in SYM (at a fixed baud rate to save memory space) and let the external system interrupt to get action. Your added software will have to provide the necessary communication protocol (use full duplex) to "handshake", using some of those control codes you may have wondered about. So far no one has told us they have completed the general task, although a number of readers have had success in talking between SYMs or SYMs and Apples.

Incidentally, it is a rather simple task to interface two SYMs along the cassette interface for hex interchange, in an ad hoc way, putting one SYM in the .L2 mode and the other in the .S2 mode, or LOAD and SAVE in BASIC. We have not gone all of the way with this, automating the procedure so that a signal on the AUDIO IN line triggers an interrupt. To do this would necessitate tieing PB6 of VIA #1 to either CB1 or CA1, and programming the selected pin to generate an IRQ on the "first" transition.

SYMPHYSIS 1979-80 INDEX AVAILABLE

Jack Gieryic has prepared an index covering the contents of Issues 0 through 6. You can order printed copies from him for $2.00, US/Canada, $3.00 overseas. The index was prepared in RAE-1 format; he sent us a copy on cassette, which we now have on disk. We will compare the relative utility of softcopy (CRT) and hard copy information retrieval, using the index as a test vehicle. Jack's address is 2041 - 138th Ave., N.W., Andover, MN 55303.

SPEAKING SOFTLY (ON SOFTWARE)

Some very useful software is coming in faster than we can check it out. We will describe some of the choicer items here. They are either too long or too numerous to publish in a regular issue. Also many are in preliminary form, and will need additional commenting and instructional documentation before release is advisable.

>>>tiny-c and TECO-TYPE WORD PROCESSOR<<<

The ones we do plan, for sure, to market, include tiny-c (from tiny-c associates), relocated downwards to $0400 from the original SYM-1 version by Jim Goodnow II, with a modified cassette patch, and, later this year, one of two (or perhaps both) TECO-type word processors. TECO is a Text Editor available on DEC systems, and many SYM-1 owners would prefer TECO to RAE, because they learned TECO first.

Actually, there are more word processors available for SYM-1 now than we can use, but we have checked out those we know of, and feel that the choice is a matter of personal preference. If you use FORTH with its own built-in Resident Assembler Editor (that's RAE !), you may not need RAE-1 and SWP-1, so you could opt for TECO-type (we can't call it TECO).

>>>SUPERMON EXTENSIONS<<<

One of these days, we'll set up a package of all of the goodies designed by Tom Gettys as SUPERMON extensions, to fill the 4K gap at $9000, or wherever, since you'll get fully commented source code on cassette, and can add, subtract, and modify to your mind's content before burning it into EPROM. He has added "named" .S2 and .L2 (very nice) and five or six kinds of .V (for Verify). These include Verify for Checksum only, very fast, and a really fancy one which we publish elsewhere in this issue as a free-standing (not linked to MON as a built-in command) version, with several examples of its use.

>>>A SYMBOLIC DISASSEMBLER<<<

During the next quarter we hope to distribute the RAE Sybolic Disassembler, being developed by John Hissink. It will be a two-pass disassembler, the best kind, as RAE is a two-pass assembler. Perhaps the combination should be called RADE? John has completed PASS 2, the most sophisticated part, and is adding what he calls "all sorts of bells and whistles". What the RSD does is create a pseudo source code, which may be edited and reassembled as desired by RAE-1. The user is, of course, responsible for flagging all addresses outside the program as externals, adding the .BY for table entries and the #L, and the #H, for all vectors within the program. The human algorithm for doing this is still not well enough defined to program. The user adds a .EN at the end, and a .BA $xxxx at the beginning, makes any desired customizations, and reassembles to suit. See how easy this would make it to move all of the RIOT (RAM,I/O,Timer) block and MON up and down? We include a sample output elswhere for your information. It is a very powerful aid. Can't wait to try it on Microsoft BASIC!

Since even uncommented source code occupies much more memory space than its corresponding object code, the source code could be generated in segments and stored with the .CT directive at the end of all sections except the last. You could also work on desired sections for study purposes by using .CE (Continue on Error) to permit assembly with undefined (external to the section) labels. After you have analyzed the sections you can replace the arbitrary labels with meaningful labels, add comments, etc., and have your own "unofficial" source code. Does anyone out there really know the law on whether such a synthesized source code, bearing so much of the doer's sweat, blood, curses, and tears, can at least be put into the public domain? I'm sure Microsoft has a firm position on this, but has there been a test case and definitive decision?

A DEDUCTIVE STORY (PART I)
- --------- ----- --------

We had planned to publish in Issue No. 1 the first part of what we would whimsically call "A Detective Story", showing how to explore the mysteries of BAS-1, in particular, and of Microsoft BASIC, in general, but our plans went awry. We felt that this would be of particular interest to beginners, since, not only was the Reference Manual supplied with BAS-1 quite sketchy, it was downright shot full of errors in the sections dealing with precision and method of data storage. So, here is the long delayed part one. We describe only the procedure. We leave it to you to find out, not "Who did it?" (that was Microsoft), but "How was it done?"

First, fill part of the memory with AA's, because they are easy to spot. Use the MON commands '.F AA,0,F0', and '.F AA,200,3FF'. Next enter BASIC with '.J 0', and answer the MEMORY SIZE? prompt with 1024, and the WIDTH? prompt with whatever value you wish. Later on try to find the maximum and minimum values BAS-1 will accept. For now, however, enter a simple program, such as the followins (you might do better, perhaps, to start with only portions of the program, and gradually build up to test other features, e. g., string arrays and integer arrays!):

```
100 A=3
110 B=4
120 C=SQR(A*A+B*B)
130 A$="HELLO"
140 B$="GOODBYE"
150 C$=A$+" AND "+B$
160 FOR I=1 TO 5
170 A(I)=I*I
180 NEXT
190 PRINT A$,B$,C$
200 PRINT A,B,C
210 FOR J=1 TO 5
220 PRINT J,A(J)
230 NEXT
```

'RUN' the program, then enter MON with a RST. Now, print out the contents of the first 1K of memory with '.V 0,3FF'. Note the values of the seven pointers (14 bytes) from $007B through $0088, and examine the memory contents of the six sections of memory between successive pairs of these pointers. The six sections are for:

1) The program itself
2) Simple variables and string pointers
3) Array variables
4) Free space
5) "Computed" strings
6) "Garbage"

Now, with the followins hints, and the erroneous, but at least suggestive, material in the reference manual, see if you can "figure it out". We know, from our correspondence, that many readers are in very isolated places, where a task like this one would actually be the most exciting thing they could find to do on most any given evening.

1) Line numbers are converted to two hex bytes, inverted order.

2) Keywords, e. g., PRINT, FOR, USR, SQR, etc., are converted to one byte "tokens", with the high order bits equal to 1.

3) In the variable storage areas and in the string pointer area, the variable names are encoded in modified ASCII form (two bytes), with the two high order bits indicating whether the variable is integer, floating, or string depending on how they are set.

4) Integer variables are stored, rather wastefully, in easy to recognize hex form, with unnecessary 00 bytes. No storage space is saved by declaring a variable as an integer. For integer arrays, however, it does pay to use the '%' to save memory, as experimentation will show.

5) Floating point variables, i. e., those variables whose "names" do not include a '%' or '$', are stored with exponent and mantissa, with each part carrying its own sign bit. Since the standard storage form for a floating point number is "normalized" so that the mantissa is greater than one-half and less than one, the first bit of the mantissa is always considered to be a one, so it need not be "written". This convention

(Continued on page 7:17)

A KANSAS CITY STANDARD TAPE DUMP

We received a very interesting letter and software from John Newman, 1/14 Marine Pde, Victoria 3182, Australia. We reproduce parts of both below. Please note that he provides the source code only for the Kansas City Dump, not for Load. For this reason we have not been able to test it completely. Perhaps one of our readers will be able to supply the Load program.

We print only those portions of Mr Newman's source code pertinent to cassette operations, since much of the rest has been previously published in SYM-PHYSIS. Note that John operates his SYM-1 format cassette saves and loads at 2800 Baud (twice normal speed)! You will note that he does not call his saves and loads in the "usual" way, but makes the saves through LIST in BASIC, and PRint in RAE, by changing OUTVEC. The data is printed in blocks on the terminal, and then dumped on cassette, in ASCII format. His loads are made by changing INVEC to point to his load routine. We plan to use this approach on our disk system, too. Not only can ASCII data files now be exchanged directly between many types of computers, using the KC Standard, but they can be passed indirectly between RAE and BAS via cassette, for editing purposes.

If you have enough memory, you can, of course, ASCII dump and load to and from memory, speeding up the editing process. Remember we suggested this in an early issue, but never got around to working out all of the details? We'd like to hear from any of you who do complete this task. As they say in the textbooks, this is left as an exercise for the reader!

We suggest two modifications to John's programs: First, whenever writing to system RAM, include a JSR ACCESS. Second, when changing OUTVEC and INVEC, as is done here, save the old entries in RAM on entry, and restore them on exit. This will eliminate the necessity for the two different exits, one for BASIC and another for RAE, as used in this program (we killed the BASIC exit in this listing, since the location of GETCHR vhich BASIC uses for INVEC is not available). We have not tested all aspects of the program. Those we have tested do work properly.

Dear Lux:

I am now running 32K of RAM on my SYM. The boards I use for RAM are boards designed by a colleague to fit a 6800 bus as used by Telecom Australia, and built by myself. There is also 4 K of RAM at $9000 to $9FFF for utility routines, etc.

I talk to the SYM with a KTM-2/80 and use an old Olivetti terminal as a printer. Unfortunately, the printer is currently out of action, and, for hard copy, I am forced to make Kansas City Standard tapes and use them on one of the terminals at work.

To generate K.C. tapes, I have written a routine which is included in the routines in the assembly language programs on this tape in File 2. Other programs in the file (apart from those copied from SYM-PHYSIS) are some routines for input and output of data files in BASIC. These routines are called from BASIC by "USR" statements. They are also usable from RAE-1. Sorry about the lack of comments, but when I developed these, I didn't have very much memory and I haven't yet got 'round to writing them. There is a BASIC program "C" on the tape after F02 which has an example of their use.

I have been working on a "Super-SYM" along the lines mentioned in issue 5/6 of SYM-PHYSIS, but with BASIC & RAE-1 in EPROM attached to a couple of peripheral ports, instead of on disk. This approach will probably develop into a complete RAM simulated disk system, in time.

I spoke to Carl Moser on the telephone about obtaining a relocated version of RAE-1, but he didn't seem very enthusiastic about the idea. As I had already done a lot of work on disassembling RAE, I persevered and now have a completely relocatable source version which works, in RAE format, on tape.

Prices of disk systems in this country are still fairly high, e.g., approx A$400 for a single minifloppy drive without controller, whereas memory chips are very cheap, 2114's at A$2.95 & 4116's at A$4.90.

In case you have trouble reading this tape, I have enclosed hard copy of the letter, and on the "B" side of the tape is a long "SYN" track and 64 blocks of test pattern as described in "SYNERTEK TECHNICAL NOTES". There are also three copies of each file on the tape.

Yours sincerely,

John Newman

ASSEMBLE LIST

```
0001 ;   ABBREVIATED VERSION OF JOHN NEWMAN'S UTILITY
0002 ;   PROGRAM, MOVED TO $3000 BY LUX
0003 ;   LINE NUMBERS AND SEQUENCE NOT MODIFIED
0010            .ES
0020            .LS
0030            .OS
0040 ;TERMINAL CONTROL PATCH AND BASIC TAPE I/O ROUTINES
0050 ;RENUMBER
0060 ;SYM BASIC RENUMBER PROGRAM
0070            .BA $3000
0080 !!!SAI     .MD
0090            STA (INDEXA),Y
0100            .ME
0110 !!!LAI     .MD
0120            LDA (INDEXA),Y
0130            .ME
0140 !!!LBI     .MD
0150            LDA (INDEXB),Y
0160            .ME
0170 !!!SBI     .MD
0180            STA (INDEXB),Y
0190            .ME
0200 !!!STB     .MD (BYT ADR)
0210            LDA #BYT
0220            STA ADR
0230            .ME
0240 !!!OCI     .MD (CH)
0250            LDA #CH
0260            OC
0270            .ME
0280 !!!OC      .MD
0290            JSR $8AA0
0300            .ME
0310 !!!DS      .MD (DATA ADDRS) ;PUT DATA IN ADDRS
0320            LDA #L,DATA
0330            STA ADDRS
0340            LDA #H,DATA
0350            STA ADDRS+1
0360            .ME
0370 !!!DSZ     .MD (DATA ADDRS) ;PUT DATA IN ADDRS
0380            LDA *DATA
0390            STA *ADDRS
0400            LDA *DATA+1
0410            STA *ADDRS+1
0420            .ME
```

```
                        0430 ;TAPE SPEED SETTING MACROS
                        0440 !!!T1S     .MD
                        0450            JSR T1S.SET
                        0460            .ME
                        0470 !!!T2S     .MD
                        0480            JSR T2S.SET
                        0490            .ME
                        0500 ;THIS MACRO ACTUALLY SETS TAPE SPEED
                        0510 !!!TSS     .MD (HB T1 T2)
                        0520            LDA #HB
                        0530            STA $A632
                        0540            LDA #T1
                        0550            STA $A635
                        0560            LDA #T2
                        0570            STA $A63C
                        0580            RTS
                        0590            .ME
                        0600 ;JUMP TABLE
3000- 4C CC 30          0610            JMP TP.INIT
3003- 4C 51 31          0620            JMP TPEND
3006- 4C F8 30          0630            JMP TR.INIT
                        0640 ; JMP BTREND
                        0641 ; THE ABOVE IS TO RESTORE GET.CHR TO INVEC
3009- 4C 78 31          0650            JMP ETREND
                        0690 ;SYSTEM EQUATES
                        0700 T1LL       .DE $A806
                        0710 T1CH       .DE $A805
                        0720 T1LH       .DE $A807
                        0730 ACR        .DE $A80B
                        0740 IFR        .DE $A80D
                        0750 IER        .DE $A80E
                        0760 TRIG.PATCH .DE $00C4
                        0770 BUFFER .   .DE $0135
                        0780 BASIC.COLD .DE $C000
                        0790 BASIC.WARM .DE $C27E
                        0800 OUTVEC     .DE $A663
                        0810 INVEC      .DE $A660
                        0820 RESXAF     .DE $81B8
                        0840 TAPOUT     .DE $A402
                        0850 CPOSP      .DE $A600
                        0860 CPOSR      .DE $A601
                        0870 SAVER      .DE $8188
                        0880 ID         .DE $A643
                        0890 MODE       .DE $FD
                        0900 CONFIG     .DE $89A5
                        0910 ZERCK      .DE $832E
                        0920 P2SCR      .DE $829C
                        0930 LOADT      .DE $8C78
                        0940 RESALL     .DE $81B8
                        0950 DUMPT      .DE $8E87
                        0960 TSTART     .DE $A64C
                        0970 TEND       .DE $A64A
                        0980 TSTAT      .DE $8B3C
                        0990 OUTCTX     .DE $8F13
                        1000 TAPDEL     .DE $A630
                        1010 TXTTAB     .DE $7B
                        1020 NWSTRT     .DE $58
                        1030 BEGIN      .DE $5A
                        1040 STEP       .DE $5C
                        1050 SLINE      .DE $5E
                        1060 HLINE      .DE $60
                        1070 VARTAB     .DE $7D
                        1080 FACTO      .DE $B2
                        1090 LINNUM     .DE $1C
                        1100 TXTPTR     .DE $D3
                        1110 INDEXA     .DE $77
```



```
                        1120 INDEXB     .DE $79
                        1130 CHRGET     .DE $CC
                        1140 CHRGOT     .DE $D2
                        1150 FIXLNK     .DE $C323
                        1160 LINGET     .DE $C7F5
                        1170 FLOATC     .DE $D9FF
                        1180 FOUT       .DE $DB9A
                        1190 MESSUB     .DE $C954
                        1200 BREAKIN    .DE $DD
                        1210 BACK       .DE $E0
                        1220 ERMESS     .DE $C25A
                        1230 GVARAD     .DE $CE63
                        1240 OUT        .DE $8A47
                        1250 OUTBYT     .DE $82FA
                        1260 CRLF       .DE $834D
                        1270 ACCESS     .DE $8B86
                        1280 INTCHR     .DE $8A58
                        1290 SPACE      .DE $8342
                        1300 ;
                        1310 ;ROUTINE FOR OUTPUTTING TO TAPE IN KANSAS CITY STANDARD
                        1320 ;
                        1330 PINIT      DS (OUTCHT OUTVEC+1)

300C- A9 2E
300E- 8D 64 A6
3011- A9 30
3013- 8D 65 A6

3016- A9 C0             1340 TONE       LDA #%11000000
3018- 8D 0B A8          1350            STA ACR
301B- A9 00             1360            LDA #0
301D- 8D 0E A8          1370            STA IER
3020- A9 D0             1380            LDA #$D0
3022- 8D 06 A8          1390            STA T1LL
3025- A9 00             1400            LDA #0
3027- 8D 07 A8          1410            STA T1LH
302A- 8D 05 A8          1420            STA T1CH
302D- 60                1430            RTS
302E- 20 67 30          1440 OUTCHT     JSR PARITY
                        1450            OC

3031- 20 A0 8A

3034- 8E 38 A6          1460            STX $A638
3037- 8C 39 A6          1470            STY $A639
303A- 85 FC             1480            STA *$FC
303C- 20 B0 30          1490            JSR LOW
303F- A9 FF             1500            LDA #$FF
3041- 48                1510 K.C.BIT    PHA
3042- 46 FC             1520            LSR *$FC
3044- 90 06             1530            BCC LF
3046- 20 91 30          1540            JSR HIGH
3049- 4C 4F 30          1550            JMP ROT
304C- 20 B0 30          1560 LF         JSR LOW
304F- 68                1570 ROT        PLA
3050- 0A                1580            ASL A
3051- B0 EE             1590            BCS K.C.BIT
3053- 20 B0 30          1600            JSR LOW
3056- 20 B0 30          1610            JSR LOW
3059- 20 91 30          1620            JSR HIGH
305C- AE 38 A6          1630            LDX $A638
305F- AC 39 A6          1640            LDY $A639
3062- AD 00 A6          1650            LDA $A600
3065- 98                1660            TYA
3066- 60                1670            RTS
```

```
                    1680 ;
                    1690 ;EVEN PARITY GENERATING ROUTINE
                    1700 ;
3067- 20 88 81      1710 PARITY     JSR SAVER
306A- 8D 00 A6      1720            STA $A600
306D- 8D 01 A6      1730            STA $A601
3070- A2 00         1740            LDX #0
3072- A0 08         1750            LDY #8
3074- 18            1760 PARLOOP    CLC
3075- 2E 00 A6      1770            ROL $A600
3078- 90 01         1780            BCC ZERO
307A- E8            1790            INX
307B- 88            1800 ZERO       DEY
307C- D0 F6         1810            BNE PARLOOP
307E- 8A            1820            TXA
307F- 29 01         1830            AND #1
3081- D0 06         1840            BNE ODD
3083- AD 01 A6      1850            LDA $A601
3086- 4C B8 81      1860            JMP RESXAF
3089- AD 01 A6      1870 ODD        LDA $A601
308C- 09 80         1880            ORA #$80
308E- 4C B8 81      1890            JMP RESXAF
                    1900 ;
                    1910 ;OUTPUT ONE BIT OF 2400 HZ (300 BAUD)
                    1920 ;
3091- A9 D0         1930 HIGH       LDA #$D0
3093- 8D 06 A8      1940            STA T1LL
3096- A9 00         1950            LDA #0
3098- 8D 07 A8      1960            STA T1LH
309B- 8D 05 A8      1970            STA T1CH
309E- A2 10         1980 TIMH       LDX #16
30A0- AD 0D A8      1990 LOOPH      LDA IFR
30A3- 29 40         2000            AND #%01000000
30A5- F0 F9         2010            BEQ LOOPH
30A7- 09 40         2020            ORA #%01000000
30A9- 8D 0D A8      2030            STA IFR
30AC- CA            2040            DEX
30AD- D0 F1         2050            BNE LOOPH
30AF- 60            2060            RTS
                    2070 ;
                    2080 ;OUTPUT ONE BIT 1200 HZ
                    2090 ;
30B0- A9 A1         2100 LOW        LDA #$A1
30B2- 8D 06 A8      2110            STA T1LL
30B5- A9 01         2120            LDA #1
30B7- 8D 07 A8      2130            STA T1LH
30BA- A2 08         2140 TIML       LDX #8
30BC- AD 0D A8      2150 LOOPL      LDA IFR
30BF- 29 40         2160            AND #%01000000
30C1- F0 F9         2170            BEQ LOOPL
30C3- 09 40         2180            ORA #%01000000
30C5- 8D 0D A8      2190            STA IFR
30C8- CA            2200            DEX
30C9- D0 F1         2210            BNE LOOPL
30CB- 60            2220            RTS
                    3670 ;
                    3680 ;2800 BAUD TAPE PRINT INITIALISATION
                    3690 ;
30CC- A9 18         3700 TP.INIT    LDA #$18
30CE- 8D 30 A6      3710            STA TAPDEL
                    3720            T2S

30D1- 20 D5 31
30D4- A2 FF         3730            LDX #$FF
```



```
30D6- A9 00         3740            LDA #$00
30D8- E8            3750 LOOP9      INX
30D9- 9D E5 31      3760            STA WORKBF,X
30DC- E0 FF         3770            CPX #$FF
30DE- D0 F8         3780            BNE LOOP9
30E0- 20 9A 31      3790            JSR TSAVE
                    3800            DS (TPRINT OUTVEC+1)

30E3- A9 18
30E5- 8D 64 A6
30E8- A9 31
30EA- 8D 65 A6

30ED- A2 00         3810            LDX #$00
30EF- 8E 00 A6      3820            STX CPOSP
30F2- A9 02         3830            LDA #$02
30F4- 8D 30 A6      3840            STA TAPDEL
30F7- 60            3850            RTS
                    3860 ;
                    3870 ;2800 BAUD TAPE READ INITIALISATION
                    3880 ;
30F8- 20 88 81      3890 TR.INIT    JSR SAVER
                    3900            T2S

30FB- 20 D5 31

                    3910            DS (TREAD INVEC+1)

30FE- A9 34
3100- 8D 61 A6
3103- A9 31
3105- 8D 62 A6

3108- 20 86 31      3920            JSR TLOAD
310B- A2 00         3930            LDX #$00
310D- EC E6 31      3940            CPX WORKBF+1
3110- D0 E6         3950            BNE TR.INIT
3112- 8E 00 A6      3960            STX CPOSP
3115- 4C B8 81      3970            JMP RESALL
                    3980 ;
                    3990 ;PRINT DATA TO TAPE BUFFER IGNORE LINE FEEDS
                    4000 ;WHEN BUFFER IS FULL WRITE DATA TO TAPE
                    4010 ;
3118- AE 00 A6      4020 TPRINT     LDX CPOSP
311B- 20 67 30      4030            JSR PARITY
311E- 9D E5 31      4040            STA WORKBF,X
                    4050            OC

3121- 20 A0 8A

3124- C9 0A         4060            CMP #$A
3126- F0 08         4070            BEQ LINEFEED
3128- E0 FF         4080            CPX #$FF
312A- D0 03         4090            BNE NOTENDP
312C- 20 9A 31      4100            JSR TSAVE
312F- E8            4110 NOTENDP    INX
3130- 8E 00 A6      4120 LINEFEED   STX CPOSP
3133- 60            4130            RTS
                    4140 ;
                    4150 ;READ DATA FROM TAPE BUFFER
                    4160 ;WHEN BUFFER IS EMPTY GET MORE DATA FROM TAPE
                    4170 ;
3134- 20 88 81      4180 TREAD      JSR SAVER
3137- AE 01 A6      4190            LDX CPOSR
```

```
313A- BD E5 31  4200            LDA WORKBF,X
313D- E0 FF     4210            CPX #$FF
313F- D0 09     4220            BNE NOTENDR
3141- 8D 08 A6  4230            STA $A608
3144- 20 86 31  4240            JSR TLOAD
3147- AD 08 A6  4250            LDA $A608
314A- E8        4260 NOTENDR    INX
314B- 8E 01 A6  4270            STX CPOSR
314E- 4C B8 81  4280            JMP RESALL
                4290 ;
                4300 ;WHEN TAPE WRITE IS FINISHED FILL UNUSED BUFFER
                4310 ;WITH NULLS AND WRITE TO TAPE. RESTORE OUTPUT VECTOR
                4320 ;
3151- AE 00 A6  4330 TPEND      LDX CPOSP
3154- E0 FF     4340            CPX #$FF
3156- F0 0B     4350            BEQ BUFF.FULL
3158- A9 00     4360            LDA #$00
315A- 9D E5 31  4370            STA WORKBF,X
315D- EE 00 A6  4380            INC CPOSP
3160- 4C 51 31  4390            JMP TPEND
3163- 20 9A 31  4400 BUFF.FULL  JSR TSAVE
                4410            DS ($8AA0 OUTVEC+1)

3166- A9 A0
3168- 8D 64 A6
316B- A9 8A
316D- 8D 65 A6

                4420            T1S

3170- 20 C5 31

3173- 60        4430            RTS
                4440 ;
                4450 ;WHEN TAPE INPUT FINISHED RESTORE INPUT VECTOR
                4460 ;TO BASIC CONTROL PATCH
                4470 ;
                4480 ;BTREND DS (GETCHR INVEC+1)
                4490            T1S

3174- 20 C5 31

3177- 60        4500            RTS
                4510 ;
                4520 ;RESTORE EDITOR INPUT VECTOR
                4530 ;
                4540 ETREND     DS ($8A58 INVEC+1)

3178- A9 58
317A- 8D 61 A6
317D- A9 8A
317F- 8D 62 A6

                4550            T1S

3182- 20 C5 31

3185- 60        4560            RTS
                4570 ;
                4580 ;LOAD DATA FROM TAPE INTO BUFFER
                4590 ;
3186- 20 AE 31  4600 TLOAD      JSR TINIT
3189- A9 FF     4610            LDA #$FF
```



```
318B- 8D 43 A6  4620            STA ID
318E- 20 88 81  4630            JSR SAVER
3191- 20 78 8C  4640            JSR LOADT
3194- D8        4650            CLD
3195- A9 00     4660            LDA #$00
3197- 4C B8 81  4670 SKP        JMP RESALL
                4680 ;
                4690 ;SAVE DATA IN BUFFER ON TAPE
                4700 ;
319A- 20 AE 31  4710 TSAVE      JSR TINIT
319D- A9 00     4720            LDA #$00
319F- 8D 43 A6  4730            STA ID
31A2- 20 88 81  4740            JSR SAVER
31A5- 20 87 8E  4750            JSR DUMPT
31A8- D8        4760            CLD
31A9- A9 00     4770            LDA #$00
31AB- 4C B8 81  4780 SKIP1      JMP RESALL
                4790 ;
                4800 ;INITIALISE TAPE BUFFER ADDRESSES
                4810 ;
                4820 TINIT      DS (WORKBF TSTART)

31AE- A9 E5
31B0- 8D 4C A6
31B3- A9 31
31B5- 8D 4D A6

                4830            DS (WORKBF+256 TEND)

31B8- A9 E5
31BA- 8D 4A A6
31BD- A9 32
31BF- 8D 4B A6

31C2- A0 80     4840            LDY #$80
31C4- 60        4850            RTS
                4860 ;
                4870 ;TAPE SPEED SETTING SUBROUTINES
                4880 ;
                4890 T1S.SET    TSS ($46 $33 $5A)

31C5- A9 46
31C7- 8D 32 A6
31CA- A9 33
31CC- 8D 35 A6
31CF- A9 5A
31D1- 8D 3C A6
31D4- 60

                4900 T2S.SET    TSS ($23 $19 $2D)

31D5- A9 23
31D7- 8D 32 A6
31DA- A9 19
31DC- 8D 35 A6
31DF- A9 2D
31E1- 8D 3C A6
31E4- 60

                4910 ;
31E5-           7820 WORKBF     .DS $256
                7830            .EN
```

(Continued from page 7:8)

frees the position so that it may be used as the sign bit position. If no one told you this, you could waste much time wondering why the floating point number didn't 'compute'!

6) We purposely did not include a DIM statement so that you could observe the default value.

If we told you any more, the fun would all be gone!

A SIMPLE DATA SAVE AND DATA LOAD PROGRAM FOR BASIC

For those of you who wish to be able to save and (re)load variable files from BASIC, we publish the following machine language program submitted by Hugh Criswell, and modified (very slightly) by us. Note that the program is fully relocatable; all you need modify, if you relocate, are the first parameters in the USR calls. It is extremely important that the program which recalls the data be no longer than the program which stored the data. Also, if you modify the calling program after recalling the data it (the data) will be cleared. And, one final warning, don't start your program with a RUN (since RUN includes an implicit CLEAR); rather, use a GOTO the appropriate line number. If you wish to pass data files between several BASIC programs with this type of program, "pad" your programs with nonessential REMs, so that any data files recalled will not overlay meaningful parts of any of the programs.

Of what use is this type of program? For one example, you might set up a data file for an inventory of a record collection, involving A$(I,J) and A(I,J). "I" would be an index number for each item, and "J" would be an index number for each "fact" to be stored for that item. A$(I,1) might be the Artist's Name, A$(I,2) the Composer, A$(I,3) the Title. A(I,J) would be used for numerical facts, like cost, current value, etc., so that you can do arithmetic on these numbers. So now you know how to set up Data Bases from BASIC!

ASSEMBLE LIST(DSAV2)

```
0010 ;  SAVE AND LOAD BASIC DATA SUBROUTINES
0020
0030 ;  SUBMITTED BY HUGH E. CRISWELL
0040 ;  MINOR MODS AND REMS BY LUX
0050
0060 ;  CALL WITH X=USR(SAVE.DATA, 256*ID) TO SAVE DATA
0070 ;  CALL WITH X=USR(LOAD.DATA, 256*ID) TO LOAD DATA
0080
0090 ;  DEFINE THE VARIABLE X IN THE PROGRAM, TO
0100 ;  ALLOCATE SPACE FOR IT, I.E., INCLUDE A
0110 ;  STATEMENT SUCH AS X=1:I=1 IN THE PROGRAM
0120
0130 ;  FULLY RELOCATABLE AND EPROMABLE
0140
0150 ;  DON'T FORGET TO RESERVE 12 BYTES AT THE TOP OF
0160 ;  MEMORY FOR THE POINTERS!!!!!!
0170
0180 ;  FOR EXAMPLE, WITH SAVE DATA AT $0C0C, THE MEMORY
0190 ;  SIZE PROMPT SHOULD BE ANSWERED WITH 3072 (=$0C00)
0200
0210 ;  IT WOULD BE CONVENIENT TO INCLUDE STATEMENTS
0220 ;  SIMILAR TO THE FOLLOWING AT YOUR PROGRAM'S END:
0230
0240 ;       5000 END
0250 ;       5010 FOR I=1 TO 3:SAVE A:NEXT:END
0260 ;       5020 FOR I=1 TO 3:X=USR(3084,256*65):NEXT:END
0270 ;       5030 X=USR(3173,256*65):END:REM-RELOAD DATA
0280 ;       5040 REM- GOTO THE PROPER LINE ABOVE TO PERFORM
0290 ;       5050 REM- YOUR DESIRED SAVES AND LOADS
0300 ;       5060 REM- GOTO 5020 ONLY AFTER A SUCCESSFUL RUN
0310
0320
0330 ; BAS-1 AND MON-2 DEFINITIONS:
0340
0350 BAS.USRENT .DE $D14C
0360 BLOCKMOVE .DE $8740
0370 WRITE .DE $8E87
0380 READ .DE $85F3
0390 ACCESS .DE $8B86
0400
0410  .BA $0C0C
0420  .OS
0430
0C0C- 48          0440 SAVE.DATA PHA
0450
0C0D- A0 0B       0460 PTRS.UP LDY $12-1
0470
0C0F- B9 7D 00    0480 MOVE.UP  LDA $7D,Y
0C12- 91 87       0490  STA ($87),Y
0C14- 88          0500  DEY
0C15- 10 F8       0510  BPL MOVE.UP
0520
0C17- 20 86 8B    0530 DATA.DOWN JSR ACCESS
0540
0C1A- A5 81       0550  LDA *$81
0C1C- 8D 4E A6    0560  STA $A64E
0C1F- A5 82       0570  LDA *$82
0C21- 8D 4F A6    0580  STA $A64F
0590
0C24- A5 83       0600  LDA *$83
0C26- 8D 4C A6    0610  STA $A64C
0C29- A5 84       0620  LDA *$84
0C2B- 8D 4D A6    0630  STA $A64D
0640
0C2E- 18          0650  CLC
0660
0C2F- A5 87       0670  LDA *$87
0C31- 69 0B       0680  ADC #12-1
0C33- 8D 4A A6    0690  STA $A64A
0700
0C36- A5 88       0710  LDA *$88
0C38- 69 00       0720  ADC $0
0C3A- 8D 4B A6    0730  STA $A64B
0740
0C3D- 20 40 87    0750  JSR BLOCKMOVE
0760
0C40- A0 80       0770 TAPE.OUT LDY #$80
0780
0C42- 68          0790  PLA
0C43- 8D 4E A6    0800  STA $A64E
0C46- A9 00       0810  LDA #0
0C48- 8D 4F A6    0820  STA $A64F
0830
0C4B- A5 7D       0840  LDA *$7D
0C4D- 8D 4C A6    0850  STA $A64C
0C50- A5 7E       0860  LDA *$7E
0C52- 8D 4D A6    0870  STA $A64D
0880
0C55- A5 FD       0890  LDA *$FD
0C57- 8D 4B A6    0900  STA $A64B
0C5A- A5 FC       0910  LDA *$FC
0C5C- 8D 4A A6    0920  STA $A64A
0930
```

```
0C0C 48 A0 0B B9 7D 00 91 87,41
0C14 88 10 F8 20 86 8B A5 81,28
0C1C 8D 4E A6 A5 82 8D 4F A6,52
0C24 A5 83 8D 4C A6 A5 84 8D,AF
0C2C 4D A6 18 A5 87 69 0B 8D,E7
0C34 4A A6 A5 88 69 00 8D 4B,45
0C3C A6 20 40 87 A0 80 68 8D,E7
0C44 4E A6 A9 00 8D 4F A6 A5,AB
0C4C 7D 8D 4C A6 A5 7E 8D 4D,A4
0C54 A6 A5 FD 8D 4B A6 A5 FC,0B
0C5C 8D 4A A6 20 87 8E 38 B0,A5
0C64 0D 48 20 86 8B 8D 4A A6,A8
0C6C 20 F3 85 68 B0 F3 38 A5,28
0C74 FE E9 0C 85 FE A5 FF E9,2B
0C7C 00 85 FF A0 0B B1 FE 99,A2
0C84 7D 00 88 10 F8 A5 83 8D,64
0C8C 4E A6 A5 84 8D 4F A6 A5,A8
0C94 81 8D 4C A6 A5 82 8D 4D,A9
0C9C A6 A5 FE 8D 4A A6 A5 FF,13
0CA4 8D 4B A6 20 40 87 4C 4C,10
0CAC D1,E1
 51E1
```

```
0C5F- 20 87 8E  0940  JSR WRITE
                0950
0C62- 38        0960  SEC
0C63- B0 0D     0970  BCS PTRS.DOWN
                0980
                0990
0C65- 48        1000 LOAD.DATA PHA
0C66- 20 86 8B  1010  JSR ACCESS
0C69- 8D 4A A6  1020  STA $A64A
                1030
0C6C- 20 F3 85  1040  JSR READ
                1050
0C6F- 68        1060  PLA
0C70- B0 F3     1070  BCS LOAD.DATA
                1080
0C72- 38        1090 PTRS.DOWN SEC
                1100
                1110 ;  NOTE THAT, AFTER A TAPE READ OR WRITE,
                1120 ;  FE/FF CONTAINS EOT (TOP OF MEMORY) + 1
                1130
0C73- A5 FE     1140  LDA *$FE
0C75- E9 0C     1150  SBC #12
0C77- 85 FE     1160  STA *$FE
                1170
0C79- A5 FF     1180  LDA *$FF
0C7B- E9 00     1190  SBC #0
0C7D- 85 FF     1200  STA *$FF
                1210
0C7F- A0 0B     1220  LDY #12-1
                1230
0C81- B1 FE     1240 MOVE.DOWN LDA ($FE),Y
0C83- 99 7D 00  1250  STA $7D,Y
0C86- 88        1260  DEY
0C87- 10 F8     1270  BPL MOVE.DOWN
                1280
0C89- A5 83     1290 DATA.UP  LDA *$83
0C8B- 8D 4E A6  1300  STA $A64E
0C8E- A5 84     1310   LDA *$84
0C90- 8D 4F A6  1320  STA $A64F
                1330
0C93- A5 81     1340  LDA *$81
0C95- 8D 4C A6  1350  STA $A64C
0C98- A5 82     1360  LDA *$82
0C9A- 8D 4D A6  1370  STA $A64D
                1380
0C9D- A5 FE     1390  LDA *$FE
0C9F- 8D 4A A6  1400  STA $A64A
0CA2- A5 FF     1410  LDA *$FF
0CA4- 8D 4B A6  1420  STA $A64B
                1430
0CA7- 20 40 87  1440  JSR BLOCKMOVE
                1450
                1460 ; I (LUX, THAT IS) PREFER THE FOLLOWING METHOD
                1470 ; OF RETURNING FROM USR OVER A SIMPLE RTS, BECAUSE
                1480 ; FOR THE LATTER, THE RETURNED VALUE IS GIBBERISH,
                1490 ; AND, I THINK, HAS CAUSED ME PROBLEMS IN THIS
                1500 ; PARTICULAR TYPE OF PROGRAM.  HAVE ANY OF YOU
                1510 ; HAD PROBLEMS RELATED TO THIS "PHENOMENON" ?
                1520
0CAA- 4C 4C D1  1530  JMP BAS.USRENT
                1540
                1550  .EN
```


THE HDE FODS

Many of our readers have enquired about FODS, HDE's File Oriented Disk System, so we'll tell you a little about it, here. First of all, FODS is a complete DOS, including its own text editor, TED, assembler, ASM, text output system, TOPS, a great version of Microsoft BASIC, and a very advanced interactive disassembler, AID, plus a number of supporting utilities.

It was designed primarily for near "bare-bone" systems like TIM (Terminal Input Monitor), and KIM (Keyboard Input Monitor), whose only factory supplied firmware is in the 1K and 2K ROMS named after their contents. We have a nearly complete version (all but the BASIC), and consider FODS to be one of the most useful of the DOS's we have studied. Naturally, Dick Grabowsky, its designer, agrees with us! We particularly appreciate the fact that the FODS TED/ASM/AID package (unlike RAE-1) is fully compatible with the original MOS Technology syntax, making for better transportability between 6502 systems.

Furthermore, files can be passed between BASIC and TED, for editing purposes, since the method of storing BASIC files is in ASCII, with the carriage returns treated the same as in TED. You might wish to check how BAS-1 and RAE-1 handle their carriage returns differently, and, of course, how BAS-1 keywords are actually stored on cassette in token form. We never have gotten around to adding modified SAVE, LOAD, PUT, and GET, to BAS and RAE to provide SYM with the same capability.

Special versions of FODS are available for AIM-65 and the SYM-1, which make use of the available factory supplied firmware, and we have committed ourselves to fully supporting SYM/FODS. Dick argues that greater software transportability could be achieved with a standard FODS; we argue that SYM (and AIM) owners have invested some money in purchasing the firmware, and much time in developing software, before they are ready to invest in a disk system, and their investments should be protected. Several SYM/FODS users read their BAS-1 and RAE-1 (and even MON 1.1) to disk (disassembled, relocated as desired, and reassembled) and replace the ROM with RAM, maintaining that, with a disk system, only a simple Power-on Reset Disk Boot EPROM is all the firmware required.

We haven't gone that far, but we do have both the SYM/FODS and the standard FODS available, hence the best of all possible worlds. To give you some idea of how FODS files are organized, we print below a copy of the Standard FODS Master Disk, then a copy of our own SYM/FODS Master Disk, and a copy of the first few pages of our personal System Manual:

```
>DC DIR 2       FODS MASTER DISK IS ON DRIVE 2

01 %V3.2X   7300 7FFF 01 01     02 %SYM     6D00 6D2F 02 11
03 %HDASX   6000 6FFC 02 12     04 %HDASY   6000 6120 04 12
05 %SSR     6D00 6DD9 04 15     06 %AID     6000 6C55 05 01
07 %DDT     6000 6B4C 06 10     08 %CMT     6000 65B9 08 01
09 %TOP     6300 6FED 08 13     10 %DIR     6D00 6DC0 10 07
11 %FRE     6D00 6E0E 10 09     12 %CPY     6D00 6DE9 10 12
13 %DEL     6D00 6D32 10 14     14 %SOR     6D00 6E74 10 15
15 %*FM     6D00 6FEC 11 02     16 %REA     6D00 6DA3 11 08
17 %PAK     6D00 6EAD 11 10     18 %LDN     6D00 6D82 11 14
19 %NAM     6D00 6D7F 11 16     20 %BLM     6D00 6F07 12 01
21 %BAS     6000 696F 12 06     22 %PON     6C60 6CA3 13 09
23 %POF     6C70 6CA3 13 10     24 %TOM     6000 6499 13 11
25 %VER     6D00 6E10 14 05     26 %TED     6000 6BFE 14 08
27 %ASM     6000 60EF 15 16     28 %ONN     6A00 6A49 16 02
29 %OFF     6A10 6A49 16 03     30 %NNN     6D00 6D49 16 04
31 %FFF     6D10 6D49 16 05
```

NEXT: T16 S06

```
>DC DIR 1   SYM/FODS MASTER DISK IS ON DRIVE 1

01 %V3.2    7300 7FFF 01 01     02 %DIR     6D00 6DC0 02 11
03 %FRE     6D00 6E0E 02 13     04 %CPY     6D00 6DE9 02 16
05 %DEL     6D00 6D32 03 02     06 %SOR     6D00 6E74 03 03
07 %*FM     6D00 6FEC 03 06     08 %REA     6D00 6DA3 03 12
09 %PAK     6D00 6EAD 03 14     10 %LDN     6D00 6D82 04 02
11 %NAM     6D00 6D7F 04 04     12 %BLM     6D00 6F07 04 05
13 %LAB     6D00 6D8F 04 10     14 %FOD     6D00 6D1E 04 12
15 %NUM     6D00 6F1C 04 13     16 %TED     6000 6BBA 05 02
17 %BAX     6000 696F 06 10     18 %SWP     6000 6791 07 13
19 %PUB     6000 6799 08 13     20 %TOM     6000 6499 09 13
21 %RAE     6B00 6C80 10 07     22 %RAY     6B00 6CF3 10 11
23 %PON     6A00 6A43 10 15     24 %POF     6A10 6A43 10 16
25 %TIC     0200 31A3 11 01     26 %TIB     0200 0B91 17 01
27 %FOC     0200 242F 18 05     28 %FOR     0200 26AE 22 10
29 %ESB     0200 16F3 27 04     30 %HUE     0200 0B21 29 14
31 %PAC     6D00 6DDC 31 01     32 %BAS     6000 65F6 31 03
33 %VER     6D00 6E10 31 15

NEXT: T32 S02

>DC FRE 1   HOW MANY BYTES ARE FREE ON DRIVE 1?

8064 ($1F80) BYTES FREE
```

SYM/FODS Operating Manual
------- --------- ------

The followins files are now on the master disk:

```
01 %V3.1     FODS
02 %DIR      Directory
03 %FRE      Free memory available?
04 %CPY      Copy disk to disk
05 %DEL      Delete specified file
06 %SOR      Sort specified file
07 %*FM      Format new disk
08 %REA      Reassign address space to file
09 %PAK      Pack disk to consolidate files
10 %LDN      Load file by number (if name deleted!)
11 %NAM      Change name of specified file
12 %BLM      Block move data
13 %LAB      Sort RAE label file
14 %FOD      Return to FODS from other systems
15 %NUM      Renumber BASIC programs
16 %TED      HDE's Text Editor
17 %BAX      HDE's BAS/FODS Link
18 %SWP      SYM Word Processor
19 %PUB      Enhanced SWP
20 %TOM      Tom Gettys' SUPERMON enhancements
21 %RAE      RAE/FODS Link
22 %RAY      Enhanced RAE, permits .CT to disk
23 %PON      Printer patch in
24 %POF      Printer patch out
25 %TIC      tiny-c
26 %TIB      Tiny BASIC
27 %FOC      Enhanced FOCAL
28 %FOR      Enhanced FORTH
29 %ESB      Brown's Extended SYM/BAS
30 %HUE      HUEY - Reverse Polish Calculator
32 %BAS      Gettys' BAS/FODS link
33 %VER      "Wide-Screen" Verify with ASCII added
```


The use of each of these programs is described briefly in the following pages (one program per page). The storage location of the accompanying manuals and any additional documentation, user aids, software, listings, etc., is also provided.

In addition to the Master Disk, a collection of Applications Disks is available. Each Disk is named and numbered, and a correspondingly named and numbered Application Binder contains the support documentation.

Where extensive listings or printouts are available for reference, these are stored in named/numbered folios, whose storage location is specified in the Binder.

(NOTE: The above material is extracted from a manual being prepared for students using the University SYM-1 System. As usual, the documentation is far behind the hardware!)

A WIDE-SCREEN HEX/ASCII MEMORY DUMP

Here is a very useful, easily relocatable, memory dump utility. In addition to providing a "header" to help you to read the addresses more readily, and listing sixteen bytes per line instead of eight, if the hex byte is a printable ASCII code (the parity bit is not considered), the ASCII character is also printed in a separate table. This sort of dump can prove very helpful in locating text files buried in a program. Some examples of its use follow the listing. It should be pointed out that not all ASCII characters printed are "meaningful". For example, the BASIC tokens will be printed in a misleading manner.

ASSEMBLE LIST

```
0010 ;A "WIDE-SCREEN" COMBINED ALPHANUMERIC/HEX MEMORY DUMP
0020 ;(DISPLAY IS TOO WIDE FOR THE 40 COLUMN KTM-2)
0030
0040 ;     AN IMPROVED VERIFY FOR SYM-1
0050 ;     ADAPTED FROM TOM GETTY'S' SYM-
0060 ;     FODS VERSION.  THIS IS NOT
0070 ;     LINKED TO SUPERMON, BUT IS
0080 ;     CALLED FROM MON BY A .G TO
0090 ;     ITS STARTING ADDRESS.
0100
0110 ;     WHEN PROMPTED, ENTER THE
0120 ;     BEGINNING AND ENDING ADDRESSES
0130 ;     OF THE MEMORY SECTION WHOSE
0140 ;     CONTENTS YOU WISH TO EXAMINE.
0150
0160 ;     HALT LISTING WITH THE "BREAK" KEY
0170 ;     RESUME PROGRAM WITH ."G" <RET>
0180
0190            .OS
0200            .BA $1000
0210
0220 S.ADR      .DE $F0
0230 LINPTR     .DE $F1
0240 LN.CNT     .DE $FD
0250
0260
0270 PARM       .DE $8220
0280 P2SCR      .DE $829C
```

```
                        0290 INCCMP     .DE $82B2
                        0300 DECCMP     .DE $82BE
                        0310 CHKSAD     .DE $82DD
                        0320 OUTBYT     .DE $82FA
                        0330 CRLFSZ     .DE $8316
                        0340 OCMCK      .DE $8325
                        0350 ZERCK      .DE $832E
                        0360 SPC2       .DE $833F
                        0370 SPACE      .DE $8342
                        0380 CRLF       .DE $834D
                        0390 INSTAT     .DE $8386
                        0400 CHKSUM     .DE $867B
                        0410 OUTCHR     .DE $8A47
                        0420 ACCESS     .DE $8B86
                        0430 UBRKVC     .DE $A676
                        0440 UBRK       .DE $804A
                        0450 USRENT     .DE $8035
                        0460
                        0470
1000- 20 86 8B          0480 VERIFY     JSR ACCESS
1003- A2 00             0490            LDX #0
1005- BD EA 10          0500 MORE       LDA MESSAGE,X
1008- F0 06             0510            BEQ GO.ON
100A- 20 47 8A          0520            JSR OUTCHR
100D- E8                0530            INX
100E- 10 F5             0540            BPL MORE
                        0550
1010- 20 20 82          0560 GO.ON      JSR PARM
1013- 20 B2 10          0570            JSR HEADER
1016- 20 2E 83          0580            JSR ZERCK
1019- 20 9C 82          0590            JSR P2SCR
                        0600
101C- A5 FF             0610            LDA *$FF
101E- 85 F2             0620            STA *LINPTR+1
1020- A5 FE             0630            LDA *$FE
1022- 85 F1             0640            STA *LINPTR
                        0650
1024- 48                0660            PHA
1025- 29 F0             0670            AND #$F0
1027- 85 FE             0680            STA *$FE
1029- 20 16 83          0690            JSR CRLFSZ
102C- 68                0700            PLA
102D- 85 FE             0710            STA *$FE
102F- 29 0F             0720            AND #$F
1031- 85 F0             0730            STA *S.ADR
1033- 0A                0740            ASL A
1034- 18                0750            CLC
1035- 65 F0             0760            ADC *S.ADR
1037- AA                0770            TAX
1038- 20 C7 10          0780            JSR BLANK
103B- A9 10             0790            LDA #$10
103D- 85 FD             0800            STA *LN.CNT
103F- 38                0810            SEC
1040- E5 F0             0820            SBC *S.ADR
1042- AA                0830            TAX
                        0840
1043- A0 00             0850 VER.LIN    LDY #0
1045- B1 FE             0860            LDA (*FE),Y
1047- 20 DD 82          0870            JSR CHKSAD
104A- 20 42 83          0880            JSR SPACE
104D- 20 FA 82          0890            JSR OUTBYT
1050- 20 B2 82          0900            JSR INCCMP
1053- 70 31             0910            BVS VER.END
1055- F0 02             0920            BEQ VER.NXT
```



```
1000 20 86 8B A2 00 BD EA 10,8A
1008 F0 06 20 47 8A E8 10 F5,5E
1010 20 20 82 20 B2 10 20 2E,50
1018 83 20 9C 82 A5 FF 85 F2,2C
1020 A5 FE 85 F1 48 29 F0 85,2B
1028 FE 20 16 83 68 85 FE 29,F6
1030 0F 85 F0 0A 18 65 F0 AA,9B
1038 20 C7 10 A9 10 85 FD 38,05
1040 E5 F0 AA A0 00 B1 FE 20,F3
1048 DD 82 20 42 83 20 FA B2,D3
1050 20 B2 82 70 31 F0 02 B0,6A
1058 2D CA D0 E7 20 25 83 20,00
1060 D2 10 A5 FE 85 F1 A5 FF,9F
1068 85 F2 20 86 83 B0 3C C6,F1
1070 FD F0 07 20 16 83 A2 10,50
1078 D0 C9 20 4D 83 20 B2 10,BB
1080 A9 10 85 FD D0 ED 86 FD,36
1088 20 BE 82 E0 10 F0 03 E8,61
1090 10 F6 20 25 83 48 A6 FD,1A
1098 CA F0 09 20 42 83 20 3F,21
10A0 83 4C 98 10 20 D2 10 68,02
10A8 20 7B 86 18 20 35 80 4C,5C
10B0 00 10 20 4D 83 A2 05 20,23
10B8 C7 10 8A 20 FA 82 20 42,82
10C0 83 E8 E0 10 D0 F4 60 E0,E1
10C8 00 F0 06 20 42 83 CA D0,56
10D0 FA 60 20 3F 83 A0 00 B1,E3
10D8 F1 29 7F C9 20 B0 02 A9,C0
10E0 20 20 47 8A C8 C0 10 D0,39
10E8 EE 60 45 6E 74 65 72 20,A5
10F0 72 61 6E 67 65 20 6C 69,A7
10F8 6D 69 74 73 3A 20 00,BE
 75BE
```

```
1057- B0 2D             0930            BCS VER.END
1059- CA                0940 VER.NXT    DEX
105A- D0 E7             0950            BNE VER.LIN
                        0960
105C- 20 25 83          0970            JSR OCMCK
105F- 20 D2 10          0980            JSR ASCII.OUT
                        0990
1062- A5 FE             1000            LDA *$FE
1064- 85 F1             1010            STA *LINPTR
1066- A5 FF             1020            LDA *$FF
1068- 85 F2             1030            STA *LINPTR+1
                        1040
106A- 20 86 83          1050 BRK.CHK    JSR INSTAT
106D- B0 3C             1060            BCS VER.RTS
106F- C6 FD             1070            DEC *LN.CNT
1071- F0 07             1080            BEQ NXT.PAG
                        1090
1073- 20 16 83          1100 NXT.LIN    JSR CRLFSZ
1076- A2 10             1110            LDX #$10
1078- D0 C9             1120            BNE VER.LIN
                        1130
107A- 20 4D 83          1140 NXT.PAG    JSR CRLF
107D- 20 B2 10          1150            JSR HEADER
1080- A9 10             1160            LDA #$10
1082- 85 FD             1170            STA *LN.CNT
1084- D0 ED             1180            BNE NXT.LIN
                        1190
1086- 86 FD             1200 VER.END    STX *LN.CNT
1088- 20 BE 82          1210 VER.END1   JSR DECCMP
108B- E0 10             1220            CPX #$10
108D- F0 03             1230            BEQ CHECKOUT
108F- E8                1240            INX
1090- 10 F6             1250            BPL VER.END1
                        1260
1092- 20 25 83          1270 CHECKOUT   JSR OCMCK
1095- 48                1280            PHA
                        1290
1096- A6 FD             1300            LDX *LN.CNT
1098- CA                1310 OUT.1      DEX
1099- F0 09             1320            BEQ OUT.2
109B- 20 42 83          1330            JSR SPACE
109E- 20 3F 83          1340            JSR SPC2
10A1- 4C 98 10          1350            JMP OUT.1
                        1360
10A4- 20 D2 10          1370 OUT.2      JSR ASCII.OUT
10A7- 68                1380            PLA
10A8- 20 7B 86          1390 VER.OUT    JSR CHKSUM
                        1400
10AB- 18                1410 VER.RTS    CLC
                        1420 ;If you wish to call this as
                        1430 ;a subroutine from RAE, substitute
                        1440 ;an RTS for the following two
                        1450 ;instructions.
10AC- 20 35 80          1460            JSR USRENT
10AF- 4C 00 10          1470            JMP VERIFY
                        1480
10B2- 20 4D 83          1490 HEADER     JSR CRLF
10B5- A2 05             1500            LDX #5
10B7- 20 C7 10          1510            JSR BLANK
10BA- 8A                1520 NXT.NUM    TXA
10BB- 20 FA 82          1530            JSR OUTBYT
10BE- 20 42 83          1540            JSR SPACE
10C1- E8                1550            INX
10C2- E0 10             1560            CPX #$10
```

```
10C4- D0 F4      1570            BNE NXT.NUM
10C6- 60         1580            RTS
                 1590
10C7- E0 00      1600 BLANK      CPX #0
10C9- F0 06      1610            BEQ BLK.RTS
10CB- 20 42 83   1620 BLANK1     JSR SPACE
10CE- CA         1630            DEX
10CF- D0 FA      1640            BNE BLANK1
10D1- 60         1650 BLK.RTS    RTS
                 1660
10D2- 20 3F 83   1670 ASCII.OUT  JSR SPC2
10D5- A0 00      1680            LDY #0
10D7- B1 F1      1690 NXT.CHR    LDA (LINPTR),Y
10D9- 29 7F      1700            AND #$7F
10DB- C9 20      1710            CMP #$20
10DD- B0 02      1720            BCS IS.ASCII
10DF- A9 20      1730            LDA #$20
10E1- 20 47 8A   1740 IS.ASCII   JSR OUTCHR
10E4- C8         1750            INY
10E5- C0 10      1760            CPY #$10
10E7- D0 EE      1770            BNE NXT.CHR
10E9- 60         1780            RTS
                 1790
10EA- 45 6E 74   1800 MESSAGE    .BY 'Enter range limits: ' 0
10ED- 65 72 20
10F0- 72 61 6E
10F3- 67 65 20
10F6- 6C 69 6D
10F9- 69 74 73
10FC- 3A 20 00
                 1810            .EN
```

Here is an example of how ASCII VER can help in analyzing how BAS-1 stores its programs and variables.

FIG. 1 - The LISTing →

```
10 AA=5
20 AA%=5
30 AA$="DOGS"
40 BB$="CATS"
50 CC$=AA$+" AND "+BB$
OK
```

FIG. 2 - The VERify ↓

```
Enter range limits:
     00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
0200 00 0A 02 0A 00 41 41 AC 35 00 14 02 14 00 41 41,25       AA,5     AA
0210 25 AC 35 00 23 02 1E 00 41 41 24 AC 22 44 4F 47,BC  %,5 #    AA$,"DOG
0220 53 22 00 32 02 28 00 42 42 24 AC 22 43 41 54 53,2E  S" 2 ( BB$,"CATS
0230 22 00 4A 02 32 00 43 43 24 AC 41 41 24 A4 22 20,B0  " J 2 CC$,AA$$"
0240 41 4E 44 20 22 A4 42 42 24 00 00 00 41 41 83 20,36  AND "$BB$    AA
0250 00 00 00 C1 C1 00 05 00 00 00 41 C1 04 1D 02 00,E2     AA      AA
0260 00 42 C2 04 2C 02 00 00 43 C3 0D EA 02 00 00 AA,C1   BB ,    CC J    *
0270 AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA,61  ****************
0280 AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA,01  ****************
0290 AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA,A1  ****************
02A0 AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA,41  ****************
02B0 AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA,E1  ****************
02C0 AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA,81  ****************
02D0 AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA,21  ****************
02E0 AA AA AA AA AA AA AA AA AA AA 44 4F 47 53 20 41,53  **********DOGS A
02F0 4E 44 20 43 41 54 53 44 4F 47 53 20 41 4E 44 20,70  ND CATSDOGS AND
 6D70
```


ZENER DIODE PROTECTION

The power connector on the SYM-1 is notched just the reverse of that on the older VIM-1 (Versatile Interface Module or Monitor; we really prefer that old name!), so, naturally, we applied power to our first SYM-1 with reverse polarity, during initial checkout! After the initial panic, when we thought that both the power supply and the SYM were dead, we realized that the SYM was safe, but that what had saved the SYM had killed the power supply. The 6.2 V Zener diode at CR34 protects from both overvoltage and reverse polarity. Its failure mode is not to open a circuit like a fuse, but to short it like the old copper pennies we were told never to substitute for burned out fuses. My new power supplies all have OVP (Over Voltage Protection), current limiting, thermal cut-off, etc., etc., but we still think it is good insurance to install inexpensive 6.2 V Zeners on all of our KTM-2's. A good location is near C9, by the power connector. A series fuse to protect the power supply if the Zener does its job right is worth adding, also.

A PERSONAL INFORMATION MANAGEMENT SYSTEM (PIMS)

We would like to recommend for your consideration the book from SCELBI Publications, P. O. Box 3133, Milford, CT 06460, bearing the above title (contact SCELBI, or your own book or computer dealer, NOT us!). It contains the complete listing of a BASIC program to provide what the title implies. It was written in TRS-80 Microsoft, but you will have to modify only two instructions (easy to do). You will, however, need to add a SAVE VARIABLES/LOAD VARIABLES patch. One such patch was published in an earlier issue, another is included in BBE-1 and BBE-2, and still another is published in this issue.

To load in machine languages patches easily from BASIC, dump them from MON's .S2 with an ID for which you know the ASCII equivalent, e.g., a MON tape dump with ID =41 can be read from BAS with a LOAD A, etc. Be sure to leave memory!

The program, while not the ultimate, can easily be customized, and extended to fit your needs; we added a disk patch to our version, and made a number of convenience and "cosmetic" changes. Speaking of "theft", as we were elsewhere in this issue, one of our correspondents sent us a copy of PDMS (the D is for Data), actually PIMS by an alias, which he picked up on an international Ham Radio software exchange net!

MONITOR/CASSETTE INTERFERENCE PROBLEMS?

Jack Brown, who has long used a dual cassette system, one for read, the other for write, both SYM controlled, reported an interesting problem the other day. He reversed the roles of the two recorders, and had read/write problems for the first time. One of the recorders is much closer to the video monitor, and he conjectures that the magnetic fields produced by the transformer and/or yokes in the monitor could have been the source of the problem. After this issue has gone to press, we will test out a SYM system, far from our library of disks and tapes, by waving a bulk tape eraser near a recorder, first while recording a long synch signal, then while reading same, meanwhile watching the signal on a scope. Will report results next issue.

HOW WE PREPARE FOR PUBLISHING

The camera-ready copy for SYM-PHYSIS is prepared on a 24K SYM/FODS system, and printed on a decwriter II. The entire copy for a 40 page issue fits on a little more than a single 5 1/4 inch diskette. For your information, we print the disk directory for that part of the current issue which is ready as of this date.

The ":" indicates a RAE file, the "." a BASIC file, the "&" a hex file, etc. Each system (tiny-c, FORTH, FOCAL, etc.) generates its own file

identifying character, or what is more commonly referred to as its extension. FODS permits only single symbol extensions and five character file names. The latter limitation is FODS' only major weakness for short-memoried people such as ourselves. Weeks later, or even only hours later, we find that we have forgotten what the cryptic mnemonic file names mean!

```
DC DIR 2     Contents of SYM-PHYSIS Issue No. 7

01 :LOGO   0200 0486 01 01    02 :MAST   0200 0857 01 07
03 :EDIT   0200 0740 02 04    04 :HELP   0200 09B8 02 15
05 :SUPP   0200 07C4 03 15    06 :THEFT  0200 0FC3 04 11
07 :BELL   0200 0D57 06 07    08 :BASIO  0200 0844 07 14
09 :FIX    0200 044F 08 11    10 :RAEUS  0200 0418 08 16
11 :FLEX   0200 048B 09 05    12 :DISK   0200 0F6E 09 11
13 :INDEX  0200 0410 11 06    14 :SOFT   0200 1143 11 11
15 :KCS    0200 0AC0 13 10    16 :NEW1   0200 0BFE 14 12
17 :NEW2   0200 1594 15 16    18 :S&L    0200 082A 18 08
19 :DSAVE  0200 0BE9 19 05    20 :DETEC  0200 1166 20 09
21 :FOD1   0200 0C70 22 08    22 :FOD2   0200 0A3B 23 13
23 :VER2   0200 04C6 24 14    24 :VER1   0200 0BF9 25 04
25 &VER    1000 10FE 26 08    26 :VER3   0200 030C 26 10
27 .VER    0201 024C 26 13    28 :ZENER  0200 065D 26 14
29 :CASS   0200 0552 27 07    30 :PIMS   0200 0769 27 14
31 :NUMS   0200 0362 28 09    32 :WISNI  0200 0EDA 28 12
33 :TECO   0200 03C4 30 06    34 .WISNI  0201 0D3A 30 10
35 :SYM69  0200 0BE0 32 01    36 :PUBLI  0200 058D 33 05
37 :PUB    0200 0645 33 13

NEXT: T34 S06
```

AN ALTERNATE BASIC INPUT TECHNIQUE

One of BASIC's basic (!) failings is its often frustrating behaviour when the wrong kind, and/or number, of data inputs is entered. If the <cr> is the first key entered, BAS-1 is not very forgiving; no opportunity for correcting this "boo-boo" is provided. This is especially unfortunate when young children are being taught to use SYM.

Jeff Wisnia, of Burlington, MA 01803, sent in a partial solution, which did prevent the program "abort" in the event of an initial carriage return, but did not permit character or line correction. We publish essentially his program below, but with our comments and error correction (but only before the <cr>!) features added. We included the CONT X, CONT H, and DEL as well as the the "@" and the "←" used by BAS-1, since these are frequently used in other BASICs. If you do wish to abort, use the BREAK key. You might also wish to add a CONT C exit, since this is used to cause an abort in many BASIC systems.

Since some users might try to use the ESC key to abort, this should also be allowed for. Don't forget to send out at least one "null" to the KTM-2/80, because the next character sent after an ESC is not printed. This caused us lots of worry in trying to get TECO up on our system. TECO uses ESC for control purposes, and is programmed to echo a "$" in its place. Not having the "$" show up was disconcerting, and it was a tight squeeze to get in the five bytes necessary to echo both a null and the "$". TECO was originally designed for I/O devices which ignored ESC. While you are polishing up the following program, you might wish to have CONT H echo a SPACE ($20) and another CONT H ($08) to clean up the screen as you correct your errors. Also, the "DEL" or "DELETE" key ($7F) should echo a "\" as is customary in many systems. A more elaborate echo scheme is the following: With the first "DEL" echo the "\" and the deleted character. With following "DEL"s echo only the deleted character. With the first replacement character, echo the "\" first, and then the new character.

This form of input is particularly useful in certain applications, where the data stream may include commas and quotes, since strings containing commas need not be delimited by quotes, and quotes do confuse BASIC. For example, BAS-1 will accept H. R. "Lux" Luxenberg as a string but not "Lux" Luxenberg. In other words, leading quotes are not accepted, but embedded quotes are.

Now that you know how to enter commas easily as part of strings, when you get around to implement "PIMS" (see elsewhere in this issue), you can enter a CITY, STATE ZIP item as a single string, with comma, and use the comma as a delimiter to indicate the start of the STATE ZIP part of the item for SORT purposes, while still permitting the item to be printed on a single line.

A good programmer, whose programs are intended to be used by novices, should make every effort to "idiot-proof" his programs, i.e., he should anticipate errors in input protocol, and guard against them. Jack Gieryic's newest programs are beautifully human-factored in this respect, and are well worth studying for this feature alone.

```
100 REM     WISNIA/LUX ALMOST "IDIOT-PROOF" BASIC INPUT SUBROUTINE
110 :
120 :
130 REM     THE ONLY WAY TO GET OUT OF THIS PROGRAM SHORT OF
140 REM     RESET, OR POWER DOWN, OR OTHER DRASTIC MEASURES
150 REM     IS WITH THE BREAK KEY.
160 :
170 :
180 PRINT "Enter any string of letters, numbers, or symbols..... ";
190 :
200 REM     Now go where the action is!!!!!
210 :
220 GOSUB 580
230 :
240 REM     Your string may contain lower case, commas, quotes,
250 REM     line feeds, etc., in fact any character except the
260 REM     special ones tested for below.
270 :
280 REM     ACTUALLY, THERE IS ANOTHER WAY TO GET OUT OF THIS
290 REM     WITHOUT USING THE BREAK KEY;  THE PROGRAM WILL
300 REM     HALT ITSELF WITH A BRK AFTER EITHER THE MAXIMUM
310 REM     STRING SIZE LIMIT (255 BYTES) IS EXCEEDED OR THE
320 REM     "GARBAGE" GENERATED BY THE STRING CONCATENATIONS
330 REM     CAUSES YOU TO RUN OUT OF MEMORY.
340 :
350 REM     IT MIGHT BE WORTHWHILE TO ADD BOTH A CONTROL C
360 REM     AND AN ESCAPE EXIT.
370 :
380 :
390 PRINT:PRINT:PRINT "The string you entered was: " A$
400 :
410 REM     Your string may be a pure numeric and you
420 REM     can check the number against range limits
430 REM     before accepting it, if desired.
440 :
450 A=VAL(A$)
460 PRINT "The numeric value of your string was:" A
470 :
480 REM     Continue till you are convinced it works, or
490 REM     till you get tired.  Exit (to BASIC) with BREAK.
500 :
510 PRINT:GOTO 180
520 :
530 REM     Here is the main show. Above is only a
540 REM     simple test program.
```

```
550 :
560 REM     Start out with an empty string.
570 :
580 A$=""
590 :
600 REM     Then go set the characters to fill your string!
610 REM     -30120 is $8A58, INTCHR, which accepts lower case, and
620 REM     bypasses INVEC, so that this program will work even
630 REM     with a Terminal Control Patch to BASIC.  Don't use
640 REM     INCHR here!
650 :
660 REM     The ones' complement of the "gotten" character is "saved"
670 REM     by INTCHR to $00F9 (examine the MON listing).  Because of
680 REM     a BAS-1 bug, the value returned in X is gibberish.
690 REM     Anyway, do as MON does, AND #$7F to clear the parity bit.
700 :
710 X=USR(-30120,0):Y=127 AND NOT PEEK(249)
720 :
730 REM     First, of course, make sure each character is not
740 REM     one requiring "special" handling.......
750 :
760 REM     Check for <cr>, but NOT as the first element.
770 :
780 IF Y=13 AND LEN(A$)<>0 THEN RETURN
790 :
800 REM     Don't accept an initial <cr>.
810 :
820 IF Y=13 THEN GOTO 1020
830 :
840 REM     Check for '^H', '<-', or 'DELETE'
850 :
860 IF Y=8 OR Y=95 OR Y=127 THEN GOTO 980
870 :
880 REM     Check for '^X' or 'at symbol' (cannot print it here!)
890 :
900 IF Y=24 OR Y=64 THEN GOTO 1020
910 :
920 REM     The character should be accepted
930 :
940 A$=A$+CHR$(Y):GOTO 710
950 :
960 REM     Delete the last character accepted
970 :
980 IF LEN(A$)<>0 THEN A$=LEFT$(A$,LEN(A$)-1):GOTO 710
990 :
1000 REM     In case of an empty string, come here.....
1010 :
1020 PRINT:PRINT "         Try again!":GOTO180
```

THE SYM-1/68 AND SYM-1/69

Synertek Systems Corporation recently announced the arrival of two new members of the SYM-1 family. One, the SYM-1/68, is 6800 based; the second, the SYM-1/69 is 6809 based. Also available, are conversion kits for our existing SYM-1s. The kits include the microprocessor chip, an adaptor socket, and a new monitor chip. We understand the versatility of SUPERMON was retained in the new monitors.

While the 6800 leaves us cold, because we prefer the 6502's Y-Register to the 6800's B-Register, the 6809 is another story. Forget the added speed of its 16 bit multiplication; its real power is in the PAIR of 16 bit index registers, and the PAIR of 16 bit stack pointers (one for the system, the other for the user). There is also an 8 bit Direct Page Register, so that instead of being limited to the special Zero Page addressing modes, ANY page may be selected for the special addressing

modes. If it is not obvious (and it really shouldn't be!), let us point out that these added features of the 6809 permit the writing of position independent code. If you have forgotten a few lines of code, or want to rearrange subroutines, just use the Block Move (.B); no need to reassemble, or use a relocating loader or program. What a lot of power, there! What a "well-stacked" system!

One of the best reviews we have seen on the 6809 is in the March, 1981 issue of BYTE (both this and the February issue are worthwhile reading for SYMmers), in the article "What's in Radio Shack's Color Computer?" (Yes, that's correct!), by Arens, Browne, and Scales. The article also covers the capabilities of the MC6847 (Color) Video Display Generator, as used in the SYM ColorMate, by Turpin.

A really strong argument for the 6809 (which supports 6800 code) is that the 6800 family is compatible with the FLEX DOS, and there is a lot of great software available out there. FLEX is to the 6800 world what CP/M is to the 80xx/Z80 universe, and what does not exist for the 6502 community.

The SYM-1/69 will, of course, be needing the equivalent of a RAE-1/69 (and, less importantly, a BAS-1/69) to complete the system, if we are to use it to its fullest capabilities. We will be evaluating the SYM-1/69 during the next quarter, and report in Issue No. 8.

If, after you evaluate the 6809 features, you are interested, contact SSC for additional technical information, and the SUG for prices and delivery information.

THREE STATISTICAL BASIC PROGRAMS

Prof. Hugh E. Criswell, Psychology Department, East Tennessee State University, Johnson City, TN 37614, whose BASIC Data Save and Load program appears elsewhere in this issue, sent us the following note and programs. We publish them as received. Incidentally, several former Psychology Instructors are now teaching courses in Computer Science at California State, Chico, as a result of having gotten "turned-on" by micros.

> "If you want to get some psychologists interested in microcomputers, show them these three Analysis of Variance programs. They are used a lot in the behavioural sciences; probably not in physics. Sorry they aren't commented, but someone experienced in statistics should have no trouble using them."

```
1 REM THIS IS A ONE WAY ANALYSIS OF VARIANCE ALSO CALLED A
2 REM SIMPLE RANDOMIZED ANOVA.  IT RUNS IN 4K.
5 L=0:M=0
10 DIM B(6),C(12),D(6),E(12)
20 T=0:TS=0
30 INPUT"# OF TESTS?";NT:PRINT NT
40 INPUT"# OF SS";NS:PRINTNS
45 PRINT"ENTER DATA BY SUBJECTS"
50 FOR I=1TONS
60 FOR J=1TONT
70 PRINT I;J;
80 INPUT L:PRINT L
90 B(J)=B(J)+L
100 C(I)=C(I)+L
110 D(J)=D(J)+(L*L)
120 E(I)=E(I)+(L*L)
130 T=T+L
140 TS=TS+(L*L)
150 NEXT J
160 NEXT I
170 FOR I=1TONT
180 PRINT"MEAN";I;"=";B(I)/NS
190 NEXT I
200 FOR I=1TONT
210 U=(B(I)*B(I))/NS
211 V=(D(I)-U)/(NS-1)
212 V=SQR(V)
213 V=V/SQR(NS)
215 PRINT"SEM";I;"=";V
220 NEXT I
240 T=(T*T)/(NS*NT)
```

```
250 TS=TS-T
255 L=0
260 FOR I=1TONT
270 L=L+(B(I)*B(I))
280 NEXT I
290 L=(L/NS)-T
300 FOR I=1TONS
310 M=M+(C(I)*C(I))
320 NEXT I
330 M=(M/NT)-T
340 N=TS-L
350 PRINT"SS(TOT)=";TS;"SS(TR)=";L;"SS(SS)=";M;"SS(ER)=";N
360 O=NS*NT-1
370 P=NT-1
380 O=O-P
390 PRINT"DF(TR)=";P,"DF(ER)=";O
400 L=L/P
410 N=N/O
420 PRINT"MS(TR)=";L,"MS(ER)=";N
430 PRINT"F(";P;",";O;")=";L/N
440 R=SQR((2*N)/NS)
450 PRINT "CRITICAL DIFFERENCE=";R;"TIMES T(";O;")"
```

```
1 REM THIS IS A CORRELATED ONE WAY ANOVA OR TREATMENT BY
2 REM SUBJECTS DESIGN.  IT ALSO SHOULD RUN IN 4K.
5 L=0:M=0
10 DIM B(5),C(12),D(5),E(12)
20 T=0:TS=0
30 INPUT"# OF TESTS?";NT:PRINT NT
40 INPUT"# OF SS";NS:PRINTNS
50 FOR I=1TONS
60 FOR J=1TONT
70 PRINT I;J;
80 INPUT L:PRINT L
90 B(J)=B(J)+L
100 C(I)=C(I)+L
110 D(J)=D(J)+(L*L)
120 E(I)=E(I)+(L*L)
130 T=T+L
140 TS=TS+(L*L)
150 NEXT J
160 NEXT I
170 FOR I=1TONT
180 PRINT"MEAN";I;"=";B(I)/NS
190 NEXT I
200 FOR I=1TONT
210 U=(B(I)*B(I))/NS
211 V=(D(I)-U)/(NS-1)
212 V=SQR(V)
213 V=V/SQR(NS)
215 PRINT"SEM";I;"=";V
220 NEXT I
240 T=(T*T)/(NS*NT)
250 TS=TS-T
255 L=0
260 FOR I=1TONT
270 L=L+(B(I)*B(I))
280 NEXT I
290 L=(L/NS)-T
300 FOR I=1TONS
310 M=M+(C(I)*C(I))
320 NEXT I
330 M=(M/NT)-T
340 N=TS-L-M
```

```
350 PRINT"SS(TOT)=";TS;"SS(TR)=";L;"SS(SS)=";M;"SS(ER)=";N
360 O=NS*NT-1
370 P=NT-1
380 O=O-P-(NS-1)
390 PRINT"DF(TR)=";P,"DF(ER)=";O
400 L=L/P
410 N=N/O
420 PRINT"MS(TR)=";L,"MS(ER)=";N
430 PRINT"F(";P;",";O;")=";L/N
440 R=SQR((2*N)/NS)
450 PRINT "CRITICAL DIFFERENCE=";R;"TIMES T(";O;")"
```



```
1 REM THIS IS A TWO WAY OR A X B ANOVA. IT USED TO RUN IN 4K BUT
2 REM I ADDED SOME PRINT STATEMENTS TO MAKE IT EASIER TO USE SO
3 REM IT MIGHT TAKE MORE SPACE.
10 DIM X(5,8),XS(5,8)
20 T=0:TS=0
30 PRINT "# OF COLUMNS"
40 INPUT NR
50 PRINT"# OF ROWS"
60 INPUT NC
70 PRINT"# OF SS/CELL"
80 INPUT NS
85 PRINT"ENTER DATA BY ROWS---)"
90 FOR J=1TONR
100 FOR K=1TONC
110 FOR I=1TONS
115 PRINT"ENTER S(";I;") ROW (";K;") COLL.(";J;")"
120 INPUT N
130 X(J,K)=X(J,K)+N
140 XS(J,K)=XS(J,K)+N*N
150 T=T+N
160 TS=TS+N*N
170 PRINT"X(";I;",";J;",";K;"=";N
180 NEXT I
190 NEXT K
200 NEXT J
201 PRINT"IF YOU WANT HARD COPY TYPE Y OTHERWISE TYPE N"
202 INPUT TW$
203 IF TW$<>"Y"GOTO210
204 TW=USR(&"1CFD",&"0000")
210 FOR J=1TONR
220 FOR K=1TONC
230 M=X(J,K)/NS
240 V=((XS(J,K)-((X(J,K)*X(J,K))/NS))/(NS-1))
250 SE=SQR(V)
260 ME=SE/(SQR(NS))
270 PRINT"MEAN(";J;",";K;")=";M;"SEM=";ME
280 NEXT K
290 NEXT J
300 C=(T*T)/(NR*NC*NS)
310 ST=TS-C
320 T=0:TS=0
330 FOR K=1TONC
340 FOR J=1TONR
350 T=T+X(J,K)
360 NEXT J
370 TS=TS+(T*T)
380 T=0
390 NEXT K
400 SC=(TS/(NS*NR))-C
410 T=0:TS=0
420 FOR J=1TONR
430 FOR K=1TO NC
440 T=T+X(J,K)
450 NEXT K
460 TS=TS+(T*T)
470 T=0
480 NEXT J
490 SR=(TS/(NS*NC))-C
500 TS=0
510 FOR J=1TONR
520 FOR K=1TONC
530 TS=TS+(X(J,K)*X(J,K))
540 NEXT K
550 NEXT J
560 SI=(TS/NS)-C-SC-SR
561 SE=ST-(SI+SR+SC)
565 PRINT"SS(TOT)=";ST
570 PRINT"SS(ROWS=";SR
580 PRINT"SS(COLL)=";SC
590 PRINT"SS(RXC)=";SI
600 PRINT"SS(ERROR)=";SE
610 DE=(NS*NC*NR)-1
620 DR=NR-1
630 DC=NC-1
640 DI=DR*DC
650 DE=DE-(DR+DC+DI)
670 SR=SR/DR
680 SC=SC/DC
690 SI=SI/DI
700 SE=SE/DE
701 PRINT
710 PRINT"MS(R)=";SR,"DF=";DR
720 PRINT"MS(C)=";SC,"DF=";DC
725 PRINT"MS(RXC)=";SI,"DF=";DI
730 PRINT"MS(ERR)=";SE,"DF=";DE
731 PRINT
740 PRINT"F(ROW)=";SR/SE,"DF=";DR",";DE
750 PRINT"F(COL)=";SC/SE,"DF=";DC;",";DE
760 PRINT "F(RXC)=";SI/SE,"DF=";DI;",";DE
770 TW=USR(&"1EC7",&"0000")
```

MORE FROM JACK GIERYIC

As has become his custom, Jack sent in almost enough material to fill a complete issue, and we had to pick and chose the one article below as being of most general interest. First, let us comment on one major change we have observed in Jack's programming style, and then describe the programs we didn't have room for.

In some of Jack's earlier programs the beauty of his graphics and the continuity of his games could be destroyed if the user entered the wrong number and/or type of inputs, a numerical entry which exceeded the allowable range, or a too hasty carriage return. All of us have had troubles with this, we're sure, and an article elsewhere in this issue shows one way to solve this problem.

Jack's newest programs are now nearly uncrashable; the "nearly" merely means we did our very best to crash them, and failed. It might require what the French call an "idiot-savant" to find a way. So now, the only way we can make an error is by actually enterins incorrect values, based on our own wrong decisions. The only way around that is to let the computer do all of our thinkins for us. But then, there would be no games for us to play!

>>>KTM-2 CHARACTER GENERATOR PROGRAMMER<<<

The character generator ROM(s) in the KTM-2 and KTM-2/80 (all are identical) are directly replaceable with 2716s. Jack has written a companion program to go with his earlier EPROM Burner Program which permits customizins the character set to your needs. We wish we could show you, in print, the appearance of the display screen during the process. The user can display any existins character, upper or lower case, alphanumeric or graphic, or, direct or reverse. The character appears on the screen in a large format, and a cursor can be moved around, only within the bounds of the display, with the U, D, L, and R keys. Pixels can be turned on with N and off with F.

When you are through with your desisn the data may be immediately EPROMmed, or taped for replay at a more convenient time. We have never seen a better "human-factored" program, and this is the very first program we have ever seen which we didn't feel we could improve!

Our use of this program to date has been minimal, since we have not yet had the time to desisn the "ideal" graphics set. We did, however modify the cursed blinking cursor from that annoying 8x8 rectansle to a modest sinsle dot in the lowest row of the character matrix. This creates less of a disturbins appearance in a graphics display. We did this for two of our terminals. Jean didn't like the rectansle because it was too big, and didn't like the dot, either, because it was too small! She now has her own terminal with an "underline" for the cursor. Looks sreat!

One of our students has replaced the sraphcs symbols with the Farsi (Persian) alphabet. Like Arabic, Hebrew, and other mid-east languases, Farsi is written from right to left, and it is very intrisuins to watch his programs ask for (numeric) inputs in Farsi. Arabic numerals, even in the mid-east are still written left to right, so no problem, there. He has not yet written any programs askins for Farsi input strings!

>>>HIGH RES LASER GUN<<<

This 1K machine language program presents the user with 8 targets traveling across the display created by MTU's 8K Visible Memory, and a moveable "laser" sun. Gun positionins and fire control is via the hex keypad on the SYM, or a supplementary key pad, whose desisn Jack describes. Sound effects are provided by General Instruments' Programmable Sound Generator (see below). This is a fascinating program, instructive in that it shows how the "arcade" type games may be programmed, but it does require the MTU board for its use.


```
**********************************
*                                *
* AY-3-8910/8912 DEMONSTRATOR    *
*                                *
**********************************
```

This package provides a simple means of exercising the functions on General Instrument's AY-3-8910/8912 Programmable Sound Generator. This will provide the user with a better understanding of the PSG's functions and capabilities.

Hardware requirements: 6K Memory
BASIC
Keyboard terminal with at least 40 characters per line and at least 24 lines
AY-3-8912 wired per Table 1

This package interfaces the PSG by means of the two ports on the Application (A) connector although the PSG could be memory mapped. It is also assumed the user has a copy of the PSG's data manual and is thoroughly familiar with its contents. This package only provides easy hands-on experience with the PSG. It is assumed the user understands the PSG's register organization.

The PSG requires a minimal amount of hardware. It is designed to operate under computer software control, thereby providing a high degree of versatility without the need to reconfisure any hardware connected to the PSG. This means a sinsle PSG can provide a wide range of sound effects and tones for any number of programs as each program has its own software to drive the PSG.

PROGRAM OPERATION: Log on to BASIC with at least 6144 bytes free. Enter the command RUN. The program asks the user if this is beins run on a KTM-2/80 keyboard. A Y or N response is sufficient. Do not hit the RETURN key. If Y is entered then the program automatically uses cursor positionins and other KTM-2/80 features to provide a display similar to Fisure 1. If N is entered, the program will pause for a second or two and then provide a display per Fisure 2.

The NEXT OPERATION NUMBER can be any of the following:

1, 2 or 3 - This number represents the selected channel. This will permit chanse to one of the functions (Frequency, Tone, Noise or Amplitude) of the selected channel.
4 - Permits chanse to the NOISE PERIOD
5 - Permits chanse to the ENVELOPE SHAPE/CYCLE
6 - Permits chanse to the ENVELOPE PERIOD
T - Program termination

Do not hit the RETURN key.

If 1, 2 or 3 is selected then the program prompts the user for an ITEM SELECTION. The four permitted responses are:

F - Permits a frequency chanse on the selected channel. The program prompts the user for a coarse value (0 to 15) and then a fine value (0 to 255). Hit the RETURN key after each value.
N - Permits noise to be added to the selected channel's tone. The program prompts the user for a Y or N. Do not hit the RETURN key.
T - Permits the selected channel's tone to be turned on(Y) or off(N). Do not hit the RETURN key.
A - Permits a chanse of the selected channel's amplitude. A value of 0 to 16 is permitted. Hit the RETURN key after the value is entered. Note a value of 16 will turn the amplitude control over to the selected envelope shape/cycle. This value(16) sets the M bit in the selected channel's amplitude resister.

```
                CH. 1   CH. 2   CH. 3     X  R 0   0
                                          X  R 1   0
FREQ (COARSE)                             X  R 2   0
      (FINE)                              X  R 3   0
                                          X  R 4   0
NOISE                                     X  R 5   0
TONE                                      X  R 6   0
AMPLITUDE                                 X  R 7   63
                                          X  R 8   0
                                          X  R 9   0
4 NOISE PERIOD                            X  R 10  0
5 ENVELOPE SHAPE/CYCLE                    X  R 11  0
6 ENVELOPE PERIOD (COARSE)                X  R 12  0
                  (FINE)                  X  R 13  0
                                          X  R 14  0
                                          X  R 15  0
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
NEXT OPERATION NUMBER
```

Figure 1 - KTM-2/80 DISPLAY

```
                  CH. 1    CH. 2    CH. 3

FREQ (COARSE)       0        0        0
      (FINE)        0        0        0

NOISE               NO       NO       NO
TONE                NO       NO       NO
AMPLITUDE           0        0        0

4 NOISE PERIOD                0
5 ENVELOPE SHAPE/CYCLE        0
6 ENVELOPE PERIOD (COARSE)    0
                  (FINE)      0

 R0 0  R1 0  R2 0  R3 0
 R4 0  R5 0  R6 0  R7 63
 R8 0  R9 0  R10 0  R11  0
 R12 0  R13 0  R14 0  R15 0

NEXT OPERATION NUMBER
```

Figure 2 - GENERAL DISPLAY

If 4 is selected the program prompts the user for a new NOISE PERIOD. Values from 0 thru 31 are permitted. Hit the RETURN key after the selected value is entered.

If 5 is selected the program prompts the user for a new ENVELOPE SHAPE/CYCLE. Values from 0 thru 15 are permitted. Hit the RETURN key after the selected value is entered. Refer to the PSG Data Manual for the various envelope shapes.

If 6 is selected the program prompts the user for a new ENVELOPE PERIOD. The program prompts the user for a coarse value (0 thru 255) and then a fine value (0 thru 255). Hit the RETURN key after entering each value.

After each operation is completed the CRT will display the new PSG status per Figure 1 or 2 and again prompt the user for the NEXT OPERATION NUMBER.

ITEM	8912 PIN	8910 PIN	APPLICATION (A) PIN
GROUND	6	1	1 (AA- 1)
+5	3	40	A (AA- A)
DA0	28	37	14 (AA- D)
DA1	27	36	4 (AA- 3)
DA2	26	35	3 (AA- C)
DA3	25	34	2 (AA-12)
DA4	24	33	5 (AA- N)
DA5	23	32	6 (AA-11)
DA6	22	31	7 (AA- M)
DA7	21	30	8 (AA-10)
BC1	20	29	9 (AA- L)
BC2	19	28	1 (AA- 1)
BDIR	18	27	10 (AA- 9)
A8	17	25	11 (AA- K)
NOT A9		24	1 (AA- 1)
CLOCK	15	22	FIG. 15 DATA MANUAL
CH. 1	5	4	FIG. 16 DATA MANUAL
CH. 2	4	3	FIG. 16 DATA MANUAL
CH. 3	1	38	FIG. 16 DATA MANUAL

TABLE 1 - WIRE LIST (ALL "AA- " ANNOTATIONS ADDED BY LUX)

FOR YOUR CONVENIENCE, SHOULD YOU WISH TO USE VIA #2 ON THE AA CONNECTOR, INSTEAD OF VIA #1 ON THE A CONNECTOR, THE PROPER PIN NUMBERS HAVE BEEN ADDED ABOVE IN PARENTHESES.

VIA #1 ADDRESSES BEGIN AT 40960(-24576)
VIA #2 ADDRESSES BEGIN AT 43008(-22528)

ADD 2048 TO ALL VIA ADDRESSES IN THE PROGRAM LISTING SHOULD YOU MAKE THIS CHANGE

THE DATA LINES ARE CONNECTED TO THE A-PORT
THE CONTROL LINES ARE CONNECTED TO THE B-PORT
BC1 IS DRIVEN BY PB0
BC2 IS HELD LOW
BDIR IS DRIVEN BY PB1
A8 IS DRIVEN BY PB2
NOT A9 IS HELD LOW

```
1 E=27:S=116:T1=1:T2=2:T3=4:N1=8:N2=16:N3=32:GOTO100
2 PRINTCHR$(E)+"=";:RETURN
3 PRINTCHR$(E)+"R";:RETURN
4 PRINTCHR$(E)+"G";:RETURN
5 PRINTCHR$(E)+CHR$(114);:RETURN
6 PRINTCHR$(E)+CHR$(103);:RETURN
7 POKE42579,0:GOSUB8:POKE42579,128:RETURN
8 Q=USR(-30120,-11957,0):CH=128-(Q/(-256)):RETURN
20 IFK$="Y"THENGOSUB2:PRINTCHR$(RE+32)+"Y";DT;"  "
22 POKEA1,RE:POKEA0,7:POKEA0,0:POKEA1,DT:POKEA0,6:POKEA0,0
24 R(RE)=DT:RETURN
60 DT=0
```

```
61 GOSUB7:IFCH=13THENRETURN
62 IFCH<48THEN61
63 IFCH>57THEN61
64 PRINTCHR$(CH);:DT=DT*10+CH-48:IFDT>WTHENRETURN
65 GOTO61
70 RE=7:DT=T1+T2+T3+N1+N2+N3:GOSUB20:RETURN
90 PRINTCHR$(E)+"K";:RETURN
100 GOSUB600
101 PRINT"IS THIS A KTM-2/80? ";:GOSUB8:K$="N":IFCH=89THENK$="Y"
102 A0=40960:A1=40961:POKE40962,7:POKE40963,255
103 IFK$="Y"THENGOSUB7000
104 DIMR(16):FORA=1TO16:R(A-1)=0:RE=A-1:DT=0:GOSUB20:NEXT
105 RE=7:DT=63:GOSUB20
110 IFK$<>"Y"THENGOSUB6000
112 IFK$="Y"THENGOSUB2:PRINT"1 ";:PRINTCHR$(E)+"J";:FORA=1TO9:NEXT
120 PRINT"NEXT OPERATION NUMBER ";:IFK$="Y"THENGOSUB90
122 GOSUB7:IFCH<49THEN122
123 IFCH=84THENEND
124 IFCH>54THEN122
125 PRINTCHR$(CH);
126 A=CH-48:POKE25,0:ONAGOSUB200,200,200,300,400,500
130 GOTO110
200 CL=A:PRINT"  CHANNEL";A
202 IFK$<>"Y"THEN206
204 GOSUB2:PRINT"2 ";
206 PRINT"ITEM SELECTION - ";
208 GOSUB7:IFCH=84THENGOSUB1000:RETURN
211 IFCH=70THENGOSUB3000:RETURN
212 IFCH=65THENGOSUB2000:RETURN
213 IFCH=78THENGOSUB4000:RETURN
220 IFK$<>"Y"THEN208
230 GOSUB2:PRINT"21";:GOTO208
300 IFK$<>"Y"THENPRINT""
301 PRINT" NOISE PERIOD - NEW VALUE = ";
305 W=31:GOSUB60:IFDT<32THEN340
320 IFK$<>"Y"THEN300
325 GOSUB2:PRINT"1S";:GOSUB90:POKE25,0:GOTO305
340 RE=6:GOSUB20:IFK$="Y"THENGOSUB2:PRINT"+;";DT;" "
350 RETURN
400 IFK$<>"Y"THENPRINT""
401 PRINT" ENVELOPE SHAPE/CYCLE - NEW VALUE = ";
405 W=15:GOSUB60:IFDT<16THEN440
420 IFK$<>"Y"THEN400
425 GOSUB2:PRINT"1C";:GOSUB90:POKE25,0:GOTO405
440 RE=13:GOSUB20:IFK$="Y"THENGOSUB2:PRINT",;";DT;" "
450 RETURN
500 IFK$<>"Y"THENPRINT""
501 PRINT" ENVELOPE PERIOD (COARSE) ";:IFK$<>"Y"THENPRINT""
502 PRINT" NEW VALUE = ";:POKE25,0
505 W=255:GOSUB60:IFDT<256THEN540
520 IFK$<>"Y"THEN500
525 GOSUB2:PRINT"1"+CHR$(94);:GOSUB90:POKE25,0:GOTO505
540 T=DT:IFK$="Y"THENGOSUB2:PRINT"2I";
542 IFK$<>"Y"THENPRINT""
545 PRINT"(FINE)   NEW VALUE = ";
550 W=255:GOSUB60:IFDT<256THEN590
560 IFK$<>"Y"THEN542
570 GOSUB2:PRINT"2"+CHR$(94);:GOSUB90:POKE25,0:GOTO550
590 RE=11:GOSUB20:RE=12:DT=T:GOSUB20:IFK$<>"Y"THENRETURN
594 GOSUB2:PRINT"-;";R(12);"   ":GOSUB2:PRINT",;";R(11);"   ":RETURN
600 PRINT"":PRINT"     JACK  BUILT  PROGRAMS"
610 PRINT"  AY-3-8910/8912 DEMONSTRATOR":PRINT"":RETURN
1000 PRINT"TONE (Y OR N) ";
1010 GOSUB7:IFCH=89THENPRINT"YES":A=1:GOTO1030
```



```
1015 IFCH=78THENPRINT"NO":A=0:GOTO1030
1020 GOTO1010
1030 IFCL=1THENT1=1:IFA=1THENT1=0
1040 IFCL=2THENT2=2:IFA=1THENT2=0
1050 IFCL=3THENT3=4:IFA=1THENT3=0
1060 GOSUB70:IFK$<>"Y"THENRETURN
1070 GOSUB2:PRINT"/"+CHR$(43+(10*CL));
1080 IFA=1THENPRINT"YES":RETURN
1090 PRINT"NO ":RETURN
2000 IFK$<>"Y"THENPRINT""
2010 PRINT" AMPLITUDE - NEW VALUE = ";
2020 W=16:GOSUB60:IFDT<17THEN2040
2025 IFK$<>"Y"THEN2000
2030 GOSUB2:PRINT"2J";:GOSUB90:POKE25,0:GOTO2020
2040 RE=7+CL:GOSUB20:IFK$="Y"THENGOSUB2:PRINT"("+CHR$(43+(10*CL));DT;"
"
2050 RETURN
3000 IFK$<>"Y"THENPRINT""
3001 PRINT" FREQUENCY (COARSE) ";:IFK$<>"Y"THENPRINT""
3002 PRINT" NEW VALUE = ";:POKE25,0
3005 W=15:GOSUB60:IFDT<16THEN3040
3020 IFK$<>"Y"THEN3000
3025 GOSUB2:PRINT"2R";:GOSUB90:POKE25,0:GOTO3005
3040 T=DT:IFK$="Y"THENGOSUB2:PRINT"3"+CHR$(61);
3042 IFK$<>"Y"THENPRINT""
3045 PRINT"(FINE)   NEW VALUE = ";
3050 W=255:GOSUB60:U=DT:IFU<256THEN3090
3060 IFK$<>"Y"THEN3042
3070 GOSUB2:PRINT"3R";:GOSUB90:POKE25,0:GOTO3050
3090 RE=(CL-1)*2:GOSUB20:RE=(CL-1)*2+1:DT=T:GOSUB20:IFK$<>"Y"THENRETURN
3094 GOSUB2:PRINT"#"+CHR$(43+(10*CL));T;"  ":GOSUB2:PRINT"$"+CHR$(43+(1
0*CL));U;
3095 PRINT"  ":RETURN
4000 PRINT"NOISE (Y OR N) ";
4010 GOSUB7:IFCH=89THENPRINT"YES":A=1:GOTO4030
4015 IFCH=78THENPRINT"NO":A=0:GOTO4030
4020 GOTO4010
4030 IFCL=1THENN1=8:IFA=1THENN1=0
4040 IFCL=2THENN2=16:IFA=1THENN2=0
4050 IFCL=3THENN3=32:IFA=1THENN3=0
4060 GOSUB70:IFK$<>"Y"THENRETURN
4070 GOSUB2:PRINT"&"+CHR$(43+(10*CL));
4080 IFA=1THENPRINT"YES":RETURN
4090 PRINT"NO ":RETURN
6000 PRINT"":PRINT"":PRINT"                CH. 1   CH. 2   CH. 3"
6010 PRINT"":PRINT"FREQ (COARSE)   ";R(1);"   ";R(3);"   ";R(5)
6020 PRINT"      (FINE)     ";R(0);"   ";R(2);"   ";R(4)
6030 PRINT"":PRINT"NOISE           ";
6032 A=R(7)AND8:IFA=8THENPRINT" NO";:GOTO6034
6033 PRINT"YES";
6034 PRINT"      ";:A=R(7)AND16:IFA=16THENPRINT" NO";:GOTO6036
6035 PRINT"YES";
6036 PRINT"      ";:A=R(7)AND32:IFA=32THENPRINT" NO":GOTO6040
6037 PRINT"YES"
6040 PRINT"TONE            ";:A=R(7)AND1:IFA=1THENPRINT" NO";:GOTO6042
6041 PRINT"YES";
6042 PRINT"      ";:A=R(7)AND2:IFA=2THENPRINT" NO";:GOTO6044
6043 PRINT"YES";
6044 PRINT"      ";:A=R(7)AND4:IFA=4THENPRINT" NO":GOTO6050
6045 PRINT"YES"
6050 PRINT"AMPLITUDE       ";R(8);"   ";R(9);"   ";R(10):PRINT""
6060 PRINT"4 NOISE PERIOD            ";R(6)
6070 PRINT"5 ENVELOPE SHAPE/CYCLE    ";R(13)
6080 PRINT"6 ENVELOPE PERIOD (COARSE) ";R(12)
```

```
 6090 PRINT"                    (FINE)  ";R(11)
 6100 PRINT"":FORB=1TO2:FORA=1TO4:PRINT" R"+CHR$(47+((B-1)*4)+A);
 6110 PRINTR(A-1+((B-1)*4));:NEXTA:PRINT"":NEXTB
 6120 FORA=1TO2:PRINT" R"+CHR$(55+A);R(7+A);:NEXT
 6130 FORA=1TO2:PRINT" R1"+CHR$(47+A);R(9+A);:NEXT:PRINT""
 6140 FORA=1TO4:PRINT" R1"+CHR$(49+A);R(11+A);:NEXT:PRINT""
 6199 PRINT"":RETURN
 7000 GOSUB4:PRINTCHR$(E)+"H"+CHR$(E)+"J";:FORA=1TO10:NEXT
 7010 FORA=1TO78:GOSUB2:PRINT"0"+CHR$(A+31)+CHR$(S);:NEXT
 7020 GOSUB3:S=124:FORA=1TO16:GOSUB2:PRINTCHR$(A+31)+"Q"+CHR$(S):NEXT
 7030 GOSUB5:GOSUB6:FORA=1TO16:GOSUB2:PRINTCHR$(A+31)+"TR";A-1:NEXT
 7040 GOSUB2:PRINT" 4CH. 1":GOSUB2:PRINT" >CH. 2":GOSUB2:PRINT" HCH. 3"
 7050 GOSUB2:PRINT"# FREQ (COARSE)":GOSUB2:PRINT"$%(FINE)":GOSUB2:PRINT"
& NOISE"
 7060 GOSUB2:PRINT"' TONE":GOSUB2:PRINT"( AMPLITUDE":GOSUB2:PRINT"+ 4 NO
ISE PERIOD"
 7080 GOSUB2:PRINT", 5 ENVELOPE SHAPE/CYCLE":GOSUB2
 7090 PRINT"- 6 ENVELOPE PERIOD (COARSE)":GOSUB2:PRINT".3(FINE)":RETURN
```

MISCELLANIA

*** ANDREE HOOLANDTS (ON4HU), Leusenstraat 3A, 9560 Herzele, Belgium (see the article on page 7:4), sent us a magnificent package of material, including a copy of CQ QSO, June 1979, the Bulletin of the Belgian Radio Amateurs Union. This bilingual publication included both Flemish and French versions of his BASIC program 'QTH-locator'. He also sent an English version, but our command of written (not spoken!) French let us read the original article, with much pleasure. Imagine your SYM giving you prompts and error messages in French, or whatever language you please! He included maps and charts for the European area, which supported the program beautifully. We suggest interested hams contact him for further info.

*** JACK BROWN has enhanced his BASIC enhancements. He has taken nearly all of the goodies from every other Microsoft BASIC and made them available to SYM BASIC. These include a real time clock, LISTing with pagination (including program NAME and page number on each sheet), APPEND, VERIFY (for cassette dump reliability assurance), CHAIN, EXEC (to allow your procedure to accept commands from within itself, rather than having to wait for keyboard inputs), etc. Hex arithmetic using the '$' rather than the beastly '&"XXXX"' structure is fully supported.

Here, extracted from the manual, is a list of the new commands:

```
$                       Prefix for hex numbers
@HH                     Returns current clock hours
@MM                     Returns current clock minutes
@SS                     Returns current clock seconds
.APPEND id              Append new program to current program
.AUTO ln1,step,ln2      Enable auto line number promptins
.CA(horz,vert,char)     Absolute cursor addressing
.CALL addr,p1,p2,. . .  Machine language call
.CR(horz,vert,char)     Relative cursor addressing
.CHAIN id,ln1,ln2       Chain command
.DEL ln1-ln2            Range delete command
.DR x,y                 Cassette motor control command
.EDIT ln1               Edit a program line
.EXEC string            Execute command
.GET variable           Get one key without echo
.GOTO expression        Computed GOTO command
.IN=value               Set input cassette
.LIST ln1-ln2           List using page parameters
.LOADP id               Load program from cassette
```


At this point we're both running out of space, and getting tired of typing, so we'll just bunch the rest of the new commands together, while reminding you that "id" can be a string, such as "1980 TAX RECORDS"!

.LOADV id; .LOADB id,addr; .NUM ln1,step,ln2,ln3; .OUT=val; .PAGE ln1,sep,flg; .PRINTOFF; .PRINTON; .PRINTUSING mask,expr; .SAVEP id,ln1,ln2; .SAVEV id; .SAVEB id,addr1,addr2; .STIME hr,min,sec; .TRACE f1,f2,f3; VERIFY id.

While the addition of these commands makes SYM-BASIC non-transportable to other machines, the added power is worth it!

*** NICK VRTIS has given us the go-ahead to distribute his version of Tiny PILOT for the SYM. He will be rewriting the source code in RAE-1 format, and we will be working closely with him to make the input routines more nearly "fool-proof" (see the BASIC article on page 7:27 for what this concept implies). SYM Tiny PILOT should be available by Issue No. 8. RAE source code will be available on cassette (or disk!) to permit easy expansion or "customization".

*** RAE NOTES No. 3 should be in the mail by the end of March. Notes No. 3 will include a copy of the first few pages of Carl Moser's original source code, written in ASSM/TED (the PET version of RAE, nearly identical). These will give all of the pages zero and one usage. We will also list the entry points for user available subroutines. In addition, a very fast LABELSORT program, by J. CYR, and some useful enhancements to SWP-1, by Tom Gettys, will be source-listed. Please note, of course, that the set of RAE Notes is available only to those who purchased their RAE-1 or RAE-1/2 directly from us, or who purchased the RAE Notes separately. If you don't receive your copy of RAE Notes No. 3 by 15 April 1981, please let us know.

*** DICK TURPIN sent us a copy of the first issue of the ColorMate Newsletter he is publishing for users of the ColorMate Color Graphics Board for the SYM. We were very much impressed and truly pleased to see this level of support for a product. Would that other vendors could do likewise!

We were very slightly disappointed when we first installed our board to find that the color resolution was not quite up to that of the Apple II by a factor of two. After reading the specs on the Motorola 6847 VDG chip, we resigned ourselves to this, reasoning that the highest resolution mode required 6K of dedicated RAM, and we only had 4K available. Dick now tells us that in a few months he will have an adaptor board available to mount on the ColorMate, which will fool it into thinking that the available 4K is really 6K, so that the full resolution of 256x192 becomes available. He has several other new products almost ready to announce. We will report on these as soon as we have had a chance to evaluate them.

*** BOB MYERS asks us to remind you that the cost for the upgrade kit to convert the KTM-2 to a KTM-2/80 costs $65 for the two main ROM chips, or $85 for all chips and all necessary sockets, postage paid anywhere, full instructions included. See Issue 1, back page for his address.

*** WELL, THAT is all we have space for! We already have some very exciting material for the next issue, but no room to tell you about it here. Thanks to all whose material we could not get into this issue; we will, however, as is our custom, make individual copies available to people who ask for information on those topics which your articles cover.