

SYM-PHYSIS

THE SYM-1 USERS' GROUP NEWSLETTER

VOLUME II, NUMBER 2 (ISSUE NO. 8) - SUMMER 1981 (APR/MAY/JUN)

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.

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

Editor/Publisher: H. R. 'Lux' Luxenburgs
Business/Circulation: Jean Luxenburgs
Associate Editors: Dennis Hall, Jack Brown
Jack Gieryc

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.

COMMENTS & REMARKS

We started the Users' Group, and its newsletter, because no one else had. We feel that the main reason we were first is because we were among the fortunate few to receive a preproduction release of RAE-1 (on cassette), and were early able to recognize SYM/RAE's true potential. We immediately used RAE's editor for report writing. That's why we justified getting a personal computer in the first place (the educational value was pure bonus). From report writing to newsletter publishing was a natural step.

RAE's Editor has been enhanced as a word processor by SWP-1, which we have upgraded to perhaps a SWP-1.4. There may be a SWP-2 released one day soon, but, since the source code is 'unprotected', most users have done their own customization. The fine Label Sort program by Cyr is a nice enhancement to RAE's Assembler section.

In this issue we publish what we feel is one of the most useful utilities ever, a symbolic DISAssembler into RAE. The DISARAE combination makes an extremely powerful development tool.

We also publish a 'beginners' level article on using the Hex Keypad as an ASCII input device. There is far more power available in SUPERMON

```

0010 ; "HEXTOASCII"
0020 ;OR, 'HOW TO USE THE HEXPAD FOR ASCII INPUT'
0030
0040 ;SUBMITTED BY MATT WILSON, 137 ROSE STREET
0050 ; YAGOONA NSW 2199, AUSTRALIA
0060
0070 ;MODIFIED AND ANNOTATED BY LUX
0080
0090 ;This program patches INVEC to permit ASCII
0100 ;input from the Hex Keypad on the SYM-1.
0110
0120 ;The SCOPE driver program published in the
0130 ;SYM-1 Reference Manual can be patched through
0140 ;OUTVEC. If you extend its published 5x7
0150 ;matrix character set, and treat the control
0160 ;characters as non-printing, you will have
0170 ;a one-line, thirty-two character, horizon-
0180 ;tally scrolling display. The combination of
0190 ;HEXTOASCII with (the enhanced) SCOPE will pro-
0200 ;vide a simple alphanumeric terminal, suitable
0210 ;for experimenting with, say, TINYBASIC, or 2KSA,
0220 ;both of which work well, even with only 4K.
0230
0240 HEXTOASCII .BA $0100          GOOD FOR UTILITIES
0250          .MC $0F00          RAE USES PAGE 01,
0260          .OS                SO USE RAE'S BUFFER
0270
0280 ; SUPERMON SUBROUTINES
0290
0300 SAVER .DE $8188
0310 GETKEY .DE $88AF
0320 NOLABEL .DE GETKEY+$0A
0330 INCHR .DE $8A1B
0340 OUTCHR .DE $8A47
0350 ACCESS .DE $8B86
0360
0370 ; SYSTEM RAM
0380
0390 INVEC .DE $A660
0400
0410 PATCH.INVC JSR ACCESS      TO UNPROTECT SYSRAM
0420
0430          LDA #L,HEX.ASCII   TO RESET INVEC JUMP
0440          LDY #H,HEX.ASCII
0450          STA INVEC+1
0460          STY INVEC+2
0470          RTS
0480
0490 ;Note in the program below that GET.ASCII must be
0500 ;called as a subroutine, even though used only once.
0510 ;It cannot be placed in the program main-line because
0520 ;of the way JSR SAVER and JMP RESXAF (which is built-
0530 ;in to NOLABEL) are designed to work together. The
0540 ;programming technique used here is worth studying.
0550
0560 HEX.ASCII JSR GET.ASCII
0570          JSR OUTCHR          FULL DUPLEX NEEDS ECHO
0580          RTS
0590
0600 GET.ASCII JSR SAVER          NEEDED MAINLY FOR RETURN
0610          JMP NOLABEL        RETURN IS VIA JMP RESXAF
0620
0630          .EN

```

COMMENTS & REMARKS (continued from 8:1)

than is documented in the manual, perhaps much more than even its creator(s) realized!

When we first started there was very little software for the SYM. Now there is far more coming in than we can possibly publish. One basis for selection for publication is the User Community's interests and hardware capabilities. We publish no FORTH, tiny-C, FOCAL, XPL0 programs, for example, since language groups constitute, in general, separate communities. We publish no Visible Memory or ColorMate graphics programs, because these would benefit only a small percentage of the readers.

Even so, however, a large number of excellent general interest software cannot be published because of space (and time) limitations. Most readers asked for fully commented source listings over disassemblies or object code listings, fully realizing the reduction in number of published programs this would mean. And now we feel guilty about withholding all the fine software being made available. So we ask your agreement on the following proposal:

As before, copyrights remain with the author; we ask only first publication rights, if published in SYM-PHYSIS, or one-time rights, if published as part of a general collection of six or more items, to be sold on a non-periodical basis. Authors would receive a complementary copy of the volume(s) containing their contribution(s).

SYM-PHYSIS carries no advertising, hence revenue is from subscription only. Publication will remain 40 single-spaced pages (70% reduction). We will mention sources of interesting "goodies" for the SYM - but, in general, no prices. We will recommend only those products we have personally evaluated and liked. If we like them enough and can pass on a price savings to our readers, we will market them through the Users' Group.

We have been asked by several sources to publish full descriptions, specifications, and price lists for their entire product lines as a service to our readers! We will be pleased to insert such materials with future issues as paid advertising. Contact us for rates. Each issue reaches over 1400 subscribers, many of whom share their copies.

A DEDUCTIVE STORY (PART II)

There is no need to continue this series because of a very welcome development. The May '81 issue of MICRO contains an excellent tutorial by Greg Paris, "How Microsoft BASIC Works", which is so comprehensive that we refer you to it instead.

An endnote states that the author "has been doing postdoctoral research in neurobiology, and hopes to program microcomputer-based instrumentation for a living." How many others are migrating from the biological to the computer sciences for economic reasons? We see this student (and faculty) trend on our own campus as well.

One correction: MICRO's editor notes that "Integer variables are not supported by OSI and SYM BASIC". He is only half-right (or is it half-wrong?). While OSI's BASIC does not support integer variables, giving an error for LET AX = 2.5, BAS-1 fully supports the '%' suffix to variable names. For LET TEST% = 2.5, PRINT TEST% will return the correct value "2".

(NOTE: We could only check out OSI's lower-precision version of Microsoft for failure to support "%", but we believe this may be true for the 9-digit version as well.)

SYM-PHYSIS 8:3

RAM VS ROM: HOW MUCH OF EACH?

A selected few SYMMers need no help in expanding their systems, particularly those who are disk-based. They are disassembling, re-"source"ing, enhancing, and relocating their RAEs, BASes, MONs, and DOSes, and moving their I/Os clear up to the top, to permit the maximum possible amount of contiguous RAM for their applications programs and data, as well as for their operating systems (which need not necessarily be contiguous).

RAE and BAS are reassembled to reside in the same BK block (since they are never co-resident), stored on disk, and recalled to RAM as needed. MON, too, lives on the disk, and could even be considered as merely a support to the DOS, since initial POR (Power-On Reset) is to the DOS BOOT. The original ROMs are removed and stored, and their address spaces are filled with RAM. The only ROM needed is for the DOS BOOT (the BOOTs for FODS and CODOS are well under 256 bytes). All of the VIAs and the RIOT (the 6532 with its RAM, I/O, and TIMER) go into the top one-half K of memory space.

Since they have the know-how to do all this, they need not worry about software compatibility, having the skills to adapt any programs they wish, not to mention the ability to write their own, with no help.

On the other hand, those of us with cassettes want a larger proportion of ROM, and are burning EPROMs to fill all available memory gaps! Cassettes have a real value for mass storage (we used them to back up our disk-based mailing lists until we provided a back-up disk system), but having RAE and BAS in ROM does save lots of time. We know, because our first RAE was provided us on cassette, and it does take time to read in two 4K blocks! Of course, that was part of the game, to help debug it before casting it permanently in silicon. And, for many months, till we got our EPROMmer going, we had to boot FODS on power-on or system crash (while designing the RAE/FODS and BAS/FODS linkages) from the cassette.

To summarize, a cassette based system needs lots of ROM support; a disk based system very little; on the other hand, a disk based system may not need as much RAM as might be expected, because an experienced programmer can "chain" and/or overlay programs and data, or invoke "virtual" memory.

Speaking of "virtual" memory, we are testing Jack Brown's SYM/FODS-FORTH designed for a 32K disk based system. Eight "screens" are resident, 104 additional are virtual, fetched as requested. Only the whirring and clicking of the disk drives lets you know of the fetch. Incidentally a screen is a FORTH text file of 16 64-character lines.

Only where time is truly of the essence, as in high resolution animation graphics, or in real-time high frequency signal processing, can one never have enough RAM. Actually, when working with the computer in an interactive mode, we don't like being kept waiting for long periods, but the second or two necessary for a virtual memory access actually provides a "breathing spell" and can relax some of the tension!

Anyway, we have never felt constrained personally by the 32K "ceiling" imposed on the RAM contiguity by the location of SUPERMON at \$8000, and feel that the "transportability" of software between SYMs is a far more important factor than RAM contiguity. We therefore leave MON in ROM, put our DOS(es) at \$6000-\$7FFF, and have the 24K contiguous RAM from \$0000-\$5FFF for applications. In one system we have installed the ColorMate at \$9000-\$9FFF, and in another we have filled this gap with utilities in RAM.

SYM-PHYSIS 8:4

SYSTEM EXPANSION SUGGESTIONS

The RAE-1 SET defaults are appropriate for a 4K SYM, and BAS-1 can be useful even within only 4K. What we are saying is that the SYM-1 can be useful with NO off-board expansion. RAE-1 is greatly enhanced by the addition of SWP-1, and BAS-1 is made much more responsive by the addition of BBE-1. Both of these software packages require a minimum of 8K, i. e., at least a 4K expansion.

A 4K expansion is very inexpensively provided by merely "pissy-backing" a second set of 2114's onto the existing set (they will cling together, or you may solder them, if you prefer), bending "free" their chip select pins (#8) and wiring the already decoded 10, 14, 18, 1C (all active low) lines directly to these pins in pairs.

Alternatively, you may use the Blalock 4K RAM Board to hold the added chips, but this will hide the Synertek logo! Also, it will cost you for 10 18-pin sockets, a pair of capacitors, a pair of short 16-wire flat jumper cables with 16-pin DIP plugs on both ends, and a pair of 16-pin sockets.

A 4K SYM is nice, an 8K SYM is nicer, and a 16K, or 32K, or 64K, or 128K (yes, it can be done with memory banking) would be even nicer. Here are some of the approaches we have tried for memory expansion, and our (definitely subjective) opinions.

MOTHERBOARDS: These were necessary in the days of expensive 4K boards, of which you needed many. Now with 32K on a single board, a simple (short - less than two inches in length) cable or special socket is all you need for interconnection. A good example is the 32K Beta DRAM Board (Yes, you can intermix static and dynamic RAMS in a single system).

A second argument for the motherboard approach, was that you could then add RS-232C, 20 mA loop, cassette interface, printer interface, assorted parallel and serial I/O interfaces, timers, etc., via plug-in boards. But, ALL OF THESE ARE ALREADY ON THE SYM! By tradition, motherboards are buffered, but tradition has not insisted that the bus be properly terminated, and this can lead to troubles. Also, on most motherboards some fancy rewiring is necessary to permit part of a 4K block to reside on the SYM and another part to be on a plug-in board. This makes it very difficult to make efficient use of the 3K block assigned to only three VIAs on the SYM.

We see no need for the KIM-4 bus conceived by MOS Technology for the KIM-1, and orphaned almost immediately by Commodore Business Machines, in order to push the PET over KIM! If someone gave us a free boxful of S-100 boards we might consider setting a KIM to S-100 motherboard converter, not just to be able to use them, but to explore the idea of adding Z-80 CP/M power to the system. On second thought, we'd give up the idea, and donate the boards to a worthy cause. We'd rather explore the 6809/FLEX possibilities; we think the 6809 is perhaps the ultimate in 8 bit micros.

SINGLE BOARD EXPANSION: We like the idea of a single 4"x6" 32K DRAM card plugging directly onto the SYM at the expansion connector, and hiding modestly underneath it. The Beta board is an example of this approach. The edge connector on the Beta board is designed to fit the "S-44" bus. This bus has a unique "anti-symmetric" pin assignment which, when used with 4K or 8K RAM boards, is indifferent to which way the board is inserted. Very safe, and no socket "key" needed. Fortunately the Beta designers also provided 44 holes on the card into which a right angled socket can be fitted from EITHER side. Depending on which side is selected, the board will extend out from, or tuck under, the SYM. (Note: We use the terms card and board interchangeably here)

SYM-PHYSIS 8:5

We have dubbed this arrangement, in which the board can carry a socket which plugs directly onto the SYM Expansion Connector, the Reverse-KIM bus. Another example of such a card is the ColorMate. It has edge fingers for the Reverse-KIM bus. To use such a card (alone), either solder two 44-pin connectors together, back-to-back, mount the pair on the SYM fingers and insert the ColorMate in the free socket, or solder the lugs of a 44-pin connector onto the ColorMate, and mount the socket onto the SYM's edge fingers.

DISK EXPANSION: If your memory expansion is motherboardless, you can still easily add a disk controller. We like the 4"x6" HDE controller, which happens to use the KIM-4 bus. We mounted the controller card to the bottom of SYM with strips of Velcro (tm) material, right beside the Beta board. We put 44-pin connectors on both boards and wired a "conversion" cable joining the S-44 bus to the KIM-4 bus. The SYM with the two underslung add-on boards is now mounted atop a pair of 5 1/4" disk drives, and is 'his' system, while the original VIM system is 'hers'. Jean and I can now work together, side by side, each at our very own computer, without having to resolve priorities, or resort to interrupts.

"CARD CAGE" EXPANSION: While the card cage seems to resemble an enclosed motherboard approach, it differs in one very significant aspect. The motherboard contains no buffers, in fact no active or passive elements of any kind; it is a very simple two sided printed circuit board, serving merely to wire five 44-pin edge connectors in parallel. The distance from the SYM socket to the furthest socket is 3 1/4". We will discuss this approach in a separate article in this issue.

COLOR FOR THE SYM

Perhaps the strongest selling point for the Apple over the TRS-80 and the PET was its color graphics. We would like to remind you of Dick Turpin's ColorMate (see previous issues for reviews of this and other "MicroMate" products, and write for his new brochure). We were mildly disappointed that the highest resolution (6C and 6R) modes of the Motorola MC6847 Video Display Generator were not implemented in the ColorMate, since the resolution was then only half that of the Apple.

The reason for this limitation is best understood by examining the RAM requirements for the 3C and 3R modes. The '3' implies that 3K of RAM are required to support these modes. In addition, a "Control Register" is required. The 3K of RAM and the Control Register are assigned a full 4K address block. To implement the 6C and 6R graphics modes in this same direct manner would require an inconvenient 7K address block.

MicroMate now supplies the ColorMate Plus as a 3K RAM "pissy-back" add-on for the ColorMate. The extra 3K are "bank-switched" to permit 6K of RAM and the Control Register to support 6C and 6R graphics in only 4K of SYM space. Also two "Pages" for 3C and 3R! We use \$9000-\$9FFF.

MicroMate also has a ROM Bank SwitchMate for the SYM which permits six ROMS and/or EPROMS to reside in sockets P1, P2, and P3. This could permit (we think, since we have not yet received ours) patches to RAE-1 (e.g., SWP-1) to co-reside in BAS-1 space, and BAS-1 enhancements to co-exist in RAE-1 space, with bank switching under program control.

We have seen copies of the ColorMate Users' Group Newsletter, published by MicroMate to provide customer support. Dick is to be commended for providing so much application software for his hardware products. And, speaking of newsletters and passing out commendations, let us mention Rockwell International's excellent support of the AIM-65 with their INTERACT, and Saturn Software's (Jack Brown) SOFTNEWS for SYM-FORTH and Extended SYM-BASIC.

SYM-PHYSIS 8:6

MORE COLOR FOR THE SYM

We are now dealers for RCA products, having established an excellent Dunn & Bradstreet rating during our first year! We have had an order, for nearly five weeks now (six weeks was promised), the VP-3301. This unit offers an interesting alternative to the KTM-2 as an RS-232 terminal for the SYM-1, for those wishing color graphics and rather good single voice music capabilities. We are very much impressed with the published specifications, and will review its performance in Issue No. 9.

It permits either 24 rows of 40 characters (including lower case with descenders - better than the Apple!) or 12 rows of 20 characters (useful for those with impaired vision), and, with a set of 128 user defined characters (ahead of the KTM-2), provides interesting graphics, if the illustrations in their brochure are for-real. We will have to try it on the SYM ourselves before we can comment further.

The VP-3303 is the deluxe model, with built-in RF Modulator (with sound subcarrier). If you are interested in either unit, write or phone RCA for literature, and contact us for delivery (slow) and prices (mildly discounted).

We plan to use ours mostly as a display, using the KTM-2/80 for "serious" terminal I/O because of its more "classical" keyboard.

ANOTHER WAY TO GO

When we bought our KIM-1, so many years ago, we bought a Micro Technology Unlimited (MTU) power supply to drive it. After reading Hal Chamberlin's now-classic article on computer music, we purchased the MTU DAC (Digital to Analog Converter) Board and the Advanced Music Software Package to drive it. We have since gone Stereo-DAC, and are thrilled by the new Musical Instrument Synthesis Software Package (more on this elsewhere).

When we wanted high resolution graphics we ordered the MTU Visible Memory 8K RAM and its associated software. When we needed additional memory to bring the 8K on-board and 8K Visible Memory up to 32K, we purchased the now-discontinued 16K DRAM board, together with the MTU "Card File" for the three boards, or cards, one of which, the SYM, fits into the top slot. The system was totally reliable, even with no buffering on the little motherboard.

MTU decided not to set on the KIM-4 bandwagon, since KIM-4, by definition, includes buffering, and to make the edge connector pinout essentially identical with that on the KIM (also now SYM and AIM). Be advised that KIM-4 boards do not, in general, fit the Card File. However the HDE Disk Controller (KIM-4), and all other 4"x6" cards of any pin-out, do fit into unwired 44-pin connectors on the application connector side of the File, and are easily wired to the motherboard side or to the application connector socket.

We have described above our first 32K MTU/SYM expansion, but pointed out that the old 16K DRAM board has been discontinued. Since we are installing our old 16K board in the SYM at the school's Microprocessor Lab for student use, we needed some type of replacement. Our decision was to replace the old board with MTU's dual purpose 15 3/4 K RAM/Disk Controller Card. The "missing" 1/4 K is occupied by a 256 byte BOOTSTRAP ROM (which itself surrenders 8 bytes to serve as control registers for the disk controller. For awhile we'll be using it purely as a RAM board. The only inconvenience is that the upper half of the memory is write-protected on reset, and must be unprotected by writing to one of the control registers. We will just add this little task to our standard log-on utility patch.

SYM-PHYSIS 8:7

AND MORE ON DISK SYSTEMS

With the MTU Disk Controller Card described above comes complete documentation and software (on an 8" disk) for the MTU CODOS (Channel Oriented Disk Operating System). All you need add is one or more single or double density, single or double sided, 8" disk drives, power supplies, and interconnecting cables and you will have lots of on-line mass storage. Each SIDE of a double density, double sided disk holds 500 Kbytes. We will be adding one or more such drives this summer.

We have been studying CODOS, and find it to be very powerful, and fast, because of the DMA approach. We think we will like it. CODOS is already patched to RAE-1, and we will be generating the links to BAS-1. We will in the future support both CODOS and FODS. In fact, we have made arrangements with MTU to adapt and market SYM versions of all MTU software, since the KIM and AIM versions are not ready-to-go for the SYM.

We still like the 5 1/4" FODS system for personal use. It is much more compact, and 5 1/4" drives rotate only when accessed. The larger drives run continuously, and we find the noise distracting. So why the need for an 8" system? Because our subscription list of 1400 names spreads over three 5 1/4" diskettes, and to do a search or a sort requires shuffling and dealing diskettes like a deck of playing cards. That's why! Based on both personal and business experience, we feel that an 8" system is "overkill" for personal use, and that a 5 1/4" system requires too much manual disk handling for even a small business. We will, of course support both, because we ourselves need both.

AN ABSOLUTELY ZERO-COST ASCII KEYBOARD

Way up in front of this issue, on page 8:2, to be specific, we give each of our readers an ABSOLUTELY FREE ASCII KEYBOARD. Actually, you already received it as a hidden bonus with your SYM. We're just telling you how to use it. Certainly we all knew that you could enter ASCII characters, at the cost of four keystrokes per character, by using the SHIFT and ASCII keys, followed by the two hex digit code. But you really need only two keystrokes per character.

Enter the published patch at \$0100 to change INVEC; it will fit the very minimum SYM. Then G 100 <cr>. You can now input all MON commands by their ASCII hexcodes. For example, to input G 200 <cr>, enter the sequence 47 32 30 30 0D (don't enter the spaces). Try using the M command to modify the memory. Use 4D instead of M, 20 (space) instead of the right arrow, to advance, and 0D instead of CR to terminate a command. You may use either 2D (-) or 2C (,) as the delimiter between parameters. Instead of LD 2 <cr> (actually L2 <cr>), use 4C 32 0D, etc.

We had fun using this, relocated out of harm's way, up in the unused scope buffer, with BASIC and RAE, using the video monitor on the KTM-2 as our display. With RAE you must NOP the echo in line 570, since RAE handles TECHO in its own way. Of course, we prefer the full terminal, but if you implement the approach followed by Len Green on page 2-21, scrolling alphanumeric slowly along the six seven-segment displays, you will have full alphanumeric capability at zero cost. And if you expand the character generator for the scope output program to include the full set of alphanumeric and control codes, you will have a 32-character horizontal scrolling display, analogous to the ticker-tape hard copy in use before roll paper was added-on to the old TTY, back in the good old days (remember the old newsreels of the ticker-tape parades?). We can also consider the scope display as "free", since you buy a scope as test gear, not as an I/O unit! This could be lots of fun with Tiny BASIC.

Now let's give credit where it is due; the idea is not ours. It was
(continued on page 8:20)

SYM-PHYSIS 8:8

```

0010 ; HISSINK'S SYMBOLIC DISASSEMBLER
0020
0030      .BA $3000
0040      .OS
0050      .ES
0060
0070 ;      DATA LOCATION DECLARATIONS
0080
0090 STEP      .DE $5
0100 PGM.PTR- .DE $D3
0110 LBL.PTR-  .DE $D5
0120 PARTIAL  .DE $E5
0130 PTR.STRT .DE $E6
0140 RUN.NU   .DE $E8
0150 END      .DE $E9
0160 FORMAT  .DE $EB
0170 LENGTH  .DE $EC
0180 LMNEM   .DE $ED
0190 RMNEM   .DE $EE
0200 YSAVE    .DE $EF
0210 PCL     .DE $F0
0220 PCH     .DE $F1
0230 LBL.IDX .DE $F2
0240 TEMP    .DE $F4
0250 INVALID .DE $F5
0260 LINE.NU .DE $F6
0270 TXT.LOW .DE $100
0280 LBL.LOW .DE $104
0290
0300 ;      SYM ROUTINES USED
0310
0320 INBYTE   .DE $B1D9
0330 NIBASC   .DE $B309
0340 INCHR    .DE $BA1B
0350 ACCESS  .DE $BB86
0360 TECHO    .DE $A653
0370 OUTVEC  .DE $A663
0380 RAE.WARM .DE $B003
0390
0400 ;      MACRO DEF'NS
0410
0420 !!!SZ    .MD (ADR.1 ADR.2) ;SET ADR - Z PAGE
0430          LDA *ADR.1
0440          STA *ADR.2
0450          LDA *ADR.1+1
0460          STA *ADR.2+1
0470          .ME
0480
0490 !!!SA    .MD (ADR.1 ADR.2) ;SET ADR - MEM TO Z PAGE
0500          LDA ADR.1
0510          STA *ADR.2
0520          LDA ADR.1+1
0530          STA *ADR.2+1
0540          .ME
0550
0560 ;      GET START AND END ADDRESSES
0570
3000- 20 44 34 0580      JSR TOGGLE ;TURN ON ECHO
3003- A9 00    0590      LDA #0
3005- 85 E8    0600      STA *RUN.NU ;ZERO BIT 7
3007- 85 F5    0610      STA *INVALID
3009- A2 12    0620      LDX ##12
300B- B5 E5    0630 MOVE1  LDA **ES,X
300D- 9D 50 34 0640      STA Z.STORE+1,X ;SAVE ZERO PAGE
3010- CA      0650      DEX

```

SYM-PHYSIS 8:9

```

3011- 10 F8    0660      BPL MOVE1
                0670
3013- E8      0680      INX
3014- 86 E5    0690      STX *PARTIAL
3016- B0 01 34 0700 MSG3  LDA MESS3,X
3019- F0 06    0710      BEQ FIN.3
301B- 20 63 A6 0720      JSR OUTVEC
301E- E8      0730      INX
301F- D0 F5    0740      BNE MSG3
3021- 20 1B BA 0750 FIN.3  JSR INCHR
3024- C9 44    0760      CMP #'D
3026- F0 1E    0770      BEQ CONT
3028- C9 4C    0780      CMP #'L
302A- D0 06    0790      BNE GO.RAE
302C- A0 80    0800      LDY ##80
302E- 84 E5    0810      STY *PARTIAL
3030- D0 14    0820      BNE CONT
                0830
3032- 4C DE 30 0840 GO.RAE  JMP RAE.RTN
                0850
3035- A2 00    0860 PASS2  LDX #0
3037- B0 3B 34 0870 MSG.4  LDA MESS4,X
303A- F0 06    0880      BEQ FIN.4
303C- 20 63 A6 0890      JSR OUTVEC
303F- E8      0900      INX
3040- D0 F5    0910      BNE MSG.4
3042- A0 40    0920 FIN.4  LDY ##40
3044- 84 E5    0930      STY *PARTIAL
                0940
3046- A2 00    0950 CONT   LDX #0
3048- B0 DD 33 0960 MSG1  LDA MESS1,X
304B- F0 06    0970      BEQ FIN.1
304D- 20 63 A6 0980      JSR OUTVEC
3050- E8      0990      INX
3051- D0 F5    1000      BNE MSG1
3053- 20 D9 B1 1010 FIN.1  JSR INBYTE
3056- 85 E7    1020      STA *PTR.STRT+1
3058- 20 D9 B1 1030      JSR INBYTE
305B- 85 E6    1040      STA *PTR.STRT
305D- A2 00    1050      LDX #0
305F- B0 F0 33 1060 MSG2  LDA MESS2,X
3062- F0 06    1070      BEQ FIN.2
3064- 20 63 A6 1080      JSR OUTVEC
3067- E8      1090      INX
3068- D0 F5    1100      BNE MSG2
306A- 20 D9 B1 1110 FIN.2  JSR INBYTE
306D- 85 EA    1120      STA *END+1
306F- 20 D9 B1 1130      JSR INBYTE
3072- 85 E9    1140      STA *END
                1150
3074- 24 E5    1160      BIT *PARTIAL
3076- 50 07    1170      BVC FA.1
3078- A9 00    1180      LDA #0
307A- 85 E5    1190      STA *PARTIAL
307C- 4C C3 30 1200      JMP FA.2
                1210
                1220 ;      DIS PROGRAM
                1230
                1240 PA.1      SA (LBL.LOW PGM.PTR)

```

SYM-PHYSIS 8:10

BNE DIS.AS ;ALWAYS - FIRST RUN TO SET UP LBL5

317E- F0 0B	2400	BEQ MNNDX3	31F6- E6 F0	3060	INC *PCL
3180- 4A	2410 MNNDX1	LSR A	31F8- D0 02	3070	BNE =+3
3181- 90 0B	2420	BCC MNNDX3 ;FORM INDEX INTO MNEMONIC TABLE	31FA- E6 F1	3080	INC *FCH
3183- 4A	2430	LSR A	31FC- 20 C8 32	3090	JSR LABEL
3184- 4A	2440 MNNDX2	LSR A ;1XXX1010 -> 00101XXX	31FF- C6 F0	3100	DEC *PCL
3185- 09 20	2450	DRA ##20 ;XXXXYY01 -> 00111XXX	3201- A5 EF	3110	LDA *YSAVE
3187- 8B	2460	DEY ;XXXXYY10 -> 00110XXX	3203- 85 F1	3120	STA *PCH
3188- D0 FA	2470	BNE MNNDX2 ;XXXXYY100 -> 00100XXX	3205- B0 0E	3130	BCS END.NMEM
318A- C8	2480	INY ;XXXXX000 -> 000XXXXX	3207- 38	3140	SEC
318B- 8B	2490 MNNDX3	DEY	3208- A5 D3	3150	LDA *PGM.PTR
318C- D0 F2	2500	BNE MNNDX1	320A- E9 04	3160	SBC #4
318E- A2 03	2510	LDX ##03 ;CHAR COUNT FOR MNEMONIC PRINT	320C- 85 D3	3170	STA *PGM.PTR
3190- A8	2520	TAY	320E- B0 02	3180	BCS =+3
3191- B9 5D 33	2530	LDA MNEML,Y	3210- C6 D4	3190	DEC *PGM.PTR+1
3194- 85 ED	2540	STA *LMNEM ;FETCH 3-CHAR MNEMONIC	3212- 4C 67 31	3200	JMP ERR
3196- B9 9D 33	2550	LDA MNEMR,Y ;(PACKED IN TWO BYTES)		3210	
3199- 85 EE	2560	STA *RMNEM	3215- A2 06	3220	END.NMEM LDX ##06 ;COUNT FOR 6 PRINT FORMAT BITS
319B- 24 EB	2570	BIT *RUN.NU	3217- E0 03	3230	PRADR1 CPX ##03
319D- 10 76	2580	BPL END.NMEM	3219- D0 21	3240	BNE PRADR3 ;IF X=3 THEN PRINT ADDRESS VAL
319F- A9 00	2590 PRMN1	LDA ##00	321B- A4 EC	3250	LDY *LENGTH
31A1- A0 05	2600	LDY ##05	321D- F0 1D	3260	BEQ PRADR3 ;NO PRINT IF LENGTH = 0
31A3- 06 EE	2610 PRMN2	ASL *RMNEM	321F- A5 EB	3270	LDA *FORMAT
31A5- 26 ED	2620	ROL *LMNEM ;SHIFT 5 BITS OF CHAR INTO 'A'	3221- C9 E8	3280	CMP ##E8 ;HANDLE REL ADDRESSING MODE
31A7- 2A	2630	ROL A ;(CLEAR CARRY)	3223- B0 47	3290	BCS RELADR ;PRINT TARGET ADDRESS
31A8- 8B	2640	DEY	3225- A9 4C	3300	LDA #'L
31A9- D0 F8	2650	BNE PRMN2	3227- C0 02	3310	CPY #2
31AB- 69 3F	2660	ADC ##3F ;ADD '?' OFFSET	3229- F0 06	3320	BEQ PRADR2
31AD- C9 3F	2670	CMF ##3F	322B- 24 EB	3330	BIT *RUN.NU
31AF- D0 04	2680	BNE =+5	322D- 10 37	3340	BPL CK.REL
31B1- A9 2E	2690	LDA #'	322F- A9 24	3350	LDA #'\$
31B3- 85 F5	2700	STA *INVALID	3231- 20 B9 32	3360	PRADR2 JSR A.STORE
31B5- 20 B9 32	2710	JSR A.STORE ;OUTPUT A CHAR OF MNEMONIC	3234- B1 F0	3370	BYTE2 LDA (PCL),Y
31B8- CA	2720	DEX	3236- 20 A0 32	3380	JSR ASC.STORE
31B9- D0 E4	2730	BNE PRMN1	3239- 8B	3390	DEY ;MORE SIGNIFICANT BYTE FIRST
	2740		323A- D0 F8	3400	BNE BYTE2
31BB- A1 F0	2750	LDA (PCL,X)	323C- 24 EB	3410	PRADR3 BIT *RUN.NU
31BD- 85 F4	2760	STA *TEMP	323E- 10 1D	3420	BPL PRADR4
	2770		3240- A5 EB	3430	LDA *FORMAT
31BF- A9 20	2780	LDA ##20	3242- C9 01	3440	CMF ##01 SEE CORRECTION ON PAGE 8:34
31C1- 20 B9 32	2790	JSR A.STORE	3244- B0 05	3450	BNE NZERO
	2800		3246- A9 2A	3460	LDA #'*
31C4- A5 F5	2810	LDA *INVALID	3248- 20 B9 32	3470	JSR A.STORE
31C6- F0 24	2820	BEQ CK.BIT	324B- 06 EB	3480	NZERO ASL *FORMAT ;TEST NEXT PRINT FORMAT BIT
31C8- A9 24	2830	LDA #'\$;YES, OUTPUT AS DATA BYTE	324D- 90 0E	3490	BCC PRADR4 ;IF 0, DONT PRINT
31CA- 20 B9 32	2840	JSR A.STORE	324F- B0 50 33	3500	LDA CHAR1-1,X ; CORRESPONDING CHAR
31CD- A1 F0	2850	LDA (PCL,X)	3252- 20 B9 32	3510	JSR A.STORE ;OUTPUT 1 OR 2 CHARS
31CF- 48	2860	PHA	3255- B0 56 33	3520	LDA CHAR2-1,X ;(IF CHAR FROM CHAR2 IS 0,
31D0- 20 A0 32	2870	JSR ASC.STORE	3258- F0 03	3530	BEQ PRADR4 ;DON'T PRINT IT)
31D3- 68	2880	PLA	325A- 20 B9 32	3540	JSR A.STORE
31D4- 29 7F	2890	AND ##7F	325D- CA	3550	PRADR4 DEX
31D6- C9 20	2900	CMF ##20	325E- D0 B7	3560	BNE PRADR1
31D8- 90 0F	2910	BCC X.END	3260- 24 EB	3570	BIT *RUN.NU
31DA- 48	2920	PHA	3262- 10 35	3580	BPL RTS1
31DB- A9 20	2930	LDA ##20	3264- 30 34	3590	BMI END.LINE
31DD- 20 B9 32	2940	JSR A.STORE	3266- A5 EB	3600	CK.REL LDA *FORMAT
31E0- A9 3B	2950	LDA #'	3268- C9 1D	3610	CMF ##1D
31E2- 20 B9 32	2960	JSR A.STORE	326A- D0 2D	3620	BNE RTS1
31E5- 68	2970	PLA	326C- B1 F0	3630	RELADR LDA (PCL),Y
31E6- 20 B9 32	2980	JSR A.STORE	326E- 20 BE 32	3640	JSR PCADJ2 ;PCL,H + DISPL + 1 TO 'A','Y'
31E9- 4C 9A 32	2990 X.END	JMP END.LINE	3271- AA	3650	TAX
	3000 ;		3272- E8	3660	INX
31EC- A1 F0	3010 CK.BIT	LDA (PCL,X)	3273- D0 01	3670	BNE PRNTYX ; +1 TO 'X','Y'
31EE- C9 2C	3020	CMF ##2C ;BIT?	3275- C8	3680	INY
31F0- D0 23	3030	BNE END.NMEM	3276- 98	3690	PRNTYX TYA
31F2- A5 F1	3040	LDA *PCH	3277- 48	3700	PRNTAX FHA
31F4- 85 EF	3050	STA *YSAVE	3278- A9 4C	3710	LDA #'L

327A- 20 B9 32	3720	JSR A.STORE	32E0- 88	4320	NO.INC	DEY
327D- 68	3730	PLA	32E1- D0 F7	4330		BNE NXT.CHR
327E- 20 A0 32	3740	JSR ASC.STORE ;PRINT TRGT ADDS OF BRANCH	32E3- A5 F1	4340		LDA *FCH
3281- 8A	3750	TXA ;AND RETURN	32E5- D1 F2	4350		CMF (LBL.IDX),Y
3282- 20 A0 32	3760	JSR ASC.STORE	32E7- D0 09	4360		BNE CK.END
3285- 24 E8	3770	BIT *RUN.NU	32E9- C8	4370		INY
3287- 10 10	3780	BPL RTS1	32EA- A5 F0	4380		LDA *PCL
3289- 30 0F	3790	BMI END.LINE	32EC- D1 F2	4390		CMF (LBL.IDX),Y
328B- A5 EC	3800	PCADJ LDA *LENGTH ;0=1-BYTE, 1=2-BYTE, 2=3-BYTE	32EE- D0 02	4400		BNE CK.END
328D- 38	3810	SEC	32F0- 18	4410		CLC ;LABEL MATCH FOUND
328E- A4 F1	3820	PCADJ2 LDY *FCH	32F1- 60	4420		RTS
3290- AA	3830	TAX ;TEST DISPL SIGN (FOR REL	32F2- A5 F2	4430	CK.END	LDA *LBL.IDX
3291- 10 01	3840	BPL PCADJ3 ;BRANCH). EXTEND NEG	32F4- C5 D5	4440		CMF *LBL.PTR
3293- 88	3850	DEY ;BY DECREMENTING FCH	32F6- 90 E0	4450		BCC NXT.LBL ;ACC < MEM FOR BR
3294- 65 F0	3860	PCADJ3 ADC *PCL	32F8- A5 F3	4460		LDA *LBL.IDX+1
3296- 90 01	3870	BCC RTS1 ;PCL+LENGTH (OR DISPL) +1 TO 'A'	32FA- C5 D6	4470		CMF *LBL.PTR+1
3298- C8	3880	INY ;CARRY INTO 'Y' (PCH)	32FC- 90 DA	4480		BCC NXT.LBL
3299- 60	3890	RTS1 RTS	32FE- 60	4490		RTS ;CARRY SET FOR RETURN WITHOUT MATCH
	3900			4500		
329A- A9 A0	3910	END.LINE LDA **A0 ;MARK EOL		4510		THE TABLES FOLLOW---
329C- 20 B9 32	3920	JSR A.STORE		4520		
329F- 60	3930	RTS	32FF- 40 02 45	4530	MODE	.BY \$40 \$02 \$45 \$03 \$D0 \$08 \$40 \$09 \$30
	3940		3302- 03 D0 08			
32A0- 08	3950	ASC.STORE PHP	3305- 40 09 30			
32A1- 24 E8	3960	BIT *RUN.NU		4540		XXXXXXXXZ0 INSTRUCTIONS
32A3- 10 15	3970	BPL A.STORE+1	3308- 22 45 33	4550		.BY \$22 \$45 \$33 \$D0 \$08 \$40 \$09 \$40 \$02 \$45
32A5- 4B	3980	PHA	330B- D0 08 40			
32A6- 4A	3990	LSR A	330E- 09 40 02			
32A7- 4A	4000	LSR A	3311- 45			
32A8- 4A	4010	LSR A		4560		Z=0, LEFT HALF BYTE
32A9- 4A	4020	LSR A		4570		Z=1, RIGHT HALF BYTE
32AA- 20 09 83	4030	JSR NIBASC	3312- 33 D0 08	4580		.BY \$33 \$D0 \$08 \$40 \$09 \$40 \$02 \$45 \$B3 \$D0
32AD- 20 B9 32	4040	JSR A.STORE	3315- 40 09 40			
32B0- 68	4050	PLA	3318- 02 45 B3			
32B1- 20 09 83	4060	JSR NIBASC	331B- D0			
32B4- 20 B9 32	4070	JSR A.STORE	331C- 08 40 09	4590		.BY \$08 \$40 \$09 \$00 \$22 \$44 \$33 \$D0 \$8C \$44
32B7- 28	4080	PLP	331F- 00 22 44			
32B8- 60	4090	RTS	3322- 33 D0 8C			
	4100		3325- 44			
32B9- 08	4110	A.STORE PHP	3326- 00 11 22	4600		.BY \$00 \$11 \$22 \$44 \$33 \$D0 \$8C \$44 \$9A \$10
32BA- 84 EF	4120	STY *YSAVE	3329- 44 33 D0			
32BC- A0 00	4130	LDY #0	332C- 8C 44 9A			
32BE- 91 D3	4140	STA (PGM.PTR),Y	332F- 10			
32C0- 18	4150	CLC	3330- 22 44 33	4610		.BY \$22 \$44 \$33 \$D0 \$08 \$40 \$09 \$10 \$22 \$44
32C1- E6 D3	4160	INC *PGM.PTR	3333- D0 08 40			
32C3- D0 02	4170	BNE PG.RTN	3336- 09 10 22			
32C5- E6 D4	4180	INC *PGM.PTR+1	3339- 44			
32C7- 28	4190	PG.RTN PLP	333A- 33 D0 08	4620		.BY \$33 \$D0 \$08 \$40 \$09 \$62
32C8- A4 EF	4200	LDY *YSAVE	333D- 40 09 62			
32CA- 60	4210	RTS		4630		YYXXXXZ01 INSTRUCTIONS
	4220		3340- 13 78 A9	4640		.BY \$13 \$78 \$A9
	4230	;	3343- 00	4650	MODE2	.BY \$00 ;ERR
	4240	SUBROUTINE LABEL, RETURNS CARRY CLEAR IF MATCH	3344- 21	4660		.BY \$21 ;IMM
	4250	LABEL SA (LBL.LOW LBL.IDX)	3345- 01	4670		.BY \$01 ;Z-PAG
			3346- 02	4680		.BY \$02 ;ABS
32CB- AD 04 01			3347- 00	4690		.BY \$00 ;IMPL
32CE- 85 F2			3348- 80	4700		.BY \$80 ;ACC
32D0- AD 05 01			3349- 59	4710		.BY \$59 ;(Z-PAG,X)
32D3- 85 F3			334A- 4D	4720		.BY \$4D ;(Z-PAG),Y
			334B- 11	4730		.BY \$11 ;Z-PAG,X
32D5- A0 01	4260	LDY #1 ;SKIP 'L'	334C- 12	4740		.BY \$12 ;ABS,X
32D7- 2C	4270	.BY #2C	334D- 06	4750		.BY \$06 ;ABS,Y
32D8- A0 03	4280	NXT.LBL LDY #3 ;SKIP 2ND PART OF ADDR & 'L'	334E- 4A	4760		.BY \$4A ;(ABS)
32DA- E6 F2	4290	NXT.CHR INC *LBL.IDX	334F- 05	4770		.BY \$05 ;Z-PAG,Y
32DC- D0 02	4300	BNE NO.INC	3350- 1D	4780		.BY \$1D ;REL
32DE- E6 F3	4310	INC *LBL.IDX+1	3351- 2C 29 2C	4790	CHAR1	.BY \$2C \$29 \$2C \$23 \$28 \$41
						SYM-PHYSIS 8:16


```

3354- 23 28 41
3357- 59 00 58 4800 CHAR2 .BY $59 $00 $58 $00 $00 $00
335A- 00 00 00
4810 ;
335D- 1C 8A 1C 4820 MNEML .BY $1C $8A $1C $23 $5D $8B $1B $A1 $9D
3360- 23 5B 8B
3363- 1B A1 9D
3366- 8A 1D 23 4830 .BY $8A $1D $23 $9D $8B $1D $A1 $00 $29 $19
3369- 9D 8B 1D
336C- A1 00 29
336F- 19
3370- AE 69 AB 4840 .BY $AE $69 $AB $19 $23 $24 $53 $1B $23
3373- 19 23 24
3376- 53 1B 23
3379- 24 53 19 4850 .BY $24 $53 $19 $A1 $00
337C- A1 00
4860 ;
337E- 1A 5B 5B 4870 .BY $1A $5B $5B $A5 $69 $24 $24
3381- A5 69 24
3384- 24
4880 ;
3385- AE AE AB 4890 .BY $AE $AE $AB $AD $29 $00 $7C $00
3388- AD 29 00
338B- 7C 00
4900 ;
338D- 15 9C 6D 4910 .BY $15 $9C $6D $9C $A5 $69 $29 $53
3390- 9C A5 69
3393- 29 53
4920 ;
3395- 84 13 34 4930 .BY $84 $13 $34 $11 $A5 $69 $23 $A0
3398- 11 A5 69
339B- 23 A0
4940 ;
339D- DB 62 5A 4950 MNEMR .BY $DB $62 $5A $48 $26 $62 $94 $8B
33A0- 48 26 62
33A3- 94 88
33A5- 54 44 CB 4960 .BY $54 $44 $CB $54 $68 $44 $EB $94 $F4 $B4
33A8- 54 68 44
33AB- E8 94 F4
33AE- B4
33AF- 08 84 74 4970 .BY $08 $84 $74 $B4 $28 $6E $74 $F4 $CC $4A
33B2- B4 28 6E
33B5- 74 F4 CC
33B8- 4A
33B9- 72 F2 A4 4980 .BY $72 $F2 $A4 $8A
33BC- 8A
4990 ;
33BD- F4 AA A2 5000 .BY $F4 $AA $A2 $A2 $74 $74 $74 $72
33C0- A2 74 74
33C3- 74 72
5010 ;
33C5- 44 68 B2 5020 .BY $44 $68 $B2 $32 $B2 $F4 $22 $F4
33C8- 32 B2 F4
33CB- 22 F4
5030 ;
33CD- 1A 1A 26 5040 .BY $1A $1A $26 $26 $72 $72 $88 $CB
33D0- 26 72 72
33D3- 88 CB
5050 ;
33D5- C4 CA 26 5060 .BY $C4 $CA $26 $48 $44 $44 $A2 $CB
33D8- 48 44 44
33DB- A2 CB
5070
33DD- 0D 0A 53 5080 MESS1 .BY $0D $0A 'START ADDRESS =#' $00
33E0- 54 41 52
33E3- 54 20 41

```

```

33E6- 44 44 52
33E9- 45 53 53
33EC- 20 3D 24
33EF- 00
33F0- 0D 0A 45 5090 MESS2 .BY $0D $0A 'END ADDRESS =#' $00
33F3- 4E 44 20
33F6- 41 44 44
33F9- 52 45 53
33FC- 53 20 3D
33FF- 24 00
3401- 0D 0A 44 5100 MESS3 .BY $0D $0A 'DISASSEMBLE, LABELS FIRST'
3404- 49 53 41
3407- 53 53 45
340A- 4D 42 4C
340D- 45 2C 20
3410- 4C 41 42
3413- 45 4C 53
3416- 20 46 49
3419- 52 53 54
341C- 0D 0A 4F 5110 .BY $0D $0A 'OR RETURN TO RAE (D,L OR R)?' $00
341F- 52 20 52
3422- 45 54 55
3425- 52 4E 20
3428- 54 4F 20
342B- 52 41 45
342E- 20 28 44
3431- 2C 4C 20
3434- 4F 52 20
3437- 52 29 3F
343A- 00
343B- 0D 0A 50 5120 MESS4 .BY $0D $0A 'PASS 2' $00
343E- 41 53 53
3441- 20 32 00
5130
3444- 20 86 8B 5140 TOGGLE JSR ACCESS ;ECHO CONTROL
3447- AD 53 A6 5150 LDA TECHD
344A- 09 80 5160 ORA $80
344C- 8D 53 A6 5170 STA TECHD
344F- 60 5180 Z.STORE RTS
5190
5200 .EN

```

USE OF THE DISASSEMBLER (DIS)

With RAE, assemble DIS at any appropriate address, say \$XXXX, and store the object code on mass storage for future use.

When you wish to use DIS, enter its object code from mass storage. Enter RAE at its cold start (\$B000), and modify the default SET parameters to meet your needs (more on this below). The object code to be disassembled should also be in RAM, or in ROM, if you want to work on MON, BAS, or RAE!

Next enter RUN \$XXXX, and when DIS gives you your choice of entering D, L, or R, enter L to set the Label File set up. When prompted to enter the starting and ending address of the file to be disassembled, ENTER ONLY FOUR HEX DIGITS AND DO NOT HIT RETURN. If you enter any digits in error there is no way to erase. You must abort and begin again. For this first pass, enter the address limits of the object code, and DIS will build up the complete Label File.

For PASS2 you are once again asked for starting and ending addresses. You may enter partial ranges for the object code, depending on how large a block is available for the 'source code' to be generated by DIS. You will soon set the knack of how much source code area should be allocated for each 1K or 4K block of object code.

The following example, in which DIS is asked to disassemble itself, will illustrate the disassembler's use (a 24K system is assumed):

The object code is located at \$3000-\$344F, so when RAE is entered, SET \$0200 \$2FFD \$3500 \$3FFD, to allow plenty of room for the source code without "clobbering" the object code. Then enter RUN \$3000 to call DIS. For this example enter 3000 and 344F (for both passes) when the limits are requested. REMEMBER - NO COMMAS, CARRIAGE RETURNS, OR ERRORS PERMITTED.

When RAE's ">" prompt reappears, type PRINT <cr> to see the source code listings. You would need add only a .BA \$xxxx at the beginning, and a .EN at the end, to be able to Assemble your program, if there were no external references (including pages 0 and 1). Since there are, you need only replace the 'L' for these labels with a '\$', or use .DE pseudo-ops. The latter is preferable, since "meaningful" labels, or mnemonics may then be substituted more easily.

Now you are prepared to study, comment, enhance, relocate, modify, EPROM or whatever you wish to do with the program.

A partial listing of DIS's output of DIS follows, so that you may see how it handles potential "problem" areas. Note the ';' line following RTS and JMP instructions; these are to flag possible tables. Note how the .BY's are used to indicate non-opcodes, and how the ASCII values are given as "comments" to aid in locating embedded text. Note that there are still a few areas where DIS shows that your brainpower is still needed occasionally by SYM. But what a perfect SYM-biosis!

PARTIAL DISASSEMBLY OF DIS BY DIS

```

0100          JSR L3444          2155 L330D      RTI
0105          LDA #$00          2160          .ORA #$40
0110          STA *$E8          2165          .BY $02
0115          STA *$F5          2166 #Lines 2170 3285 deleted
0120          LDX *$12          3290 L3401      .ORA L440A
0125 L300B     LDA *$E5,X        3295          .EOR *$53
0130          STA L3450,X       3300          .EOR ($53,X)
0136 #Lines 135 to 205 deleted  3305          .BY $53      ;S
0210          STY *$E5          3310          .BY $45      ;E
0215          BNE L3046         3315          .BY $4D      ;M
0220 L3032     JMP L30DE         3320          .BY $42      ;B
0225 ;          3325          .BY $4C      ;L
0230 L3035     LDX *$00          3330          .BY $45      ;E
0235 L3037     LDA L343B,X       3335          .BY $2C      ;
0240          BEQ L3042         3336 #Lines 3340 3520 deleted
0241 #Lines 245 to 2065 deleted  3525          .BY $4F      ;D
0270          BCC L32DB         3530          .BY $52      ;R
0275          RTS              3535          .BY $20      ;
0280 ;          3540          .BY $52      ;R
0285 L32FF     RTI              3545          .BY $29      ;)
0290          .BY $02          3550          .BY $3F      ;?
0295          .BY $45          ;E 3555          .BY $00
0300          .BY $03          3560 L343B     .ORA L500A
0305          .BY $D0          ;P 3565          .EOR ($53,X)
0310          .BY $08          3570          .BY $53      ;S
0315          .BY $40          ;@ 3575          .BY $20      ;
0320          .BY $09          3580          .BY $32      ;2
0325          .BY $30          ;0 3585          .BY $00
0330          .BY $22          ;" 3590 L3444     JSR L8B86
0335          .BY $45          ;E 3595          LDA LA653
0340          .BY $33          ;3 3600          .ORA *$80
0345          .BY $D0          ;P 3605          STA LA653
0350          .BY $08          3610          RTS

```

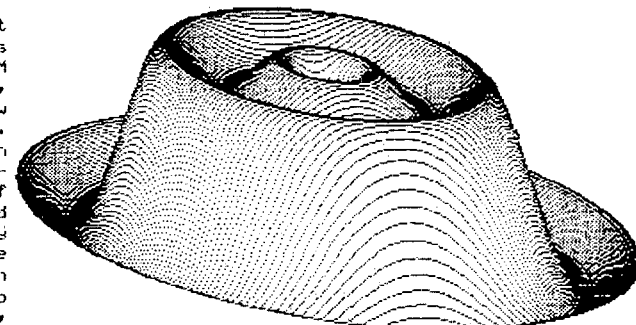
SYM-PHYSIS 8:19

(continued from Page 8:8)

submitted by Matt Wilson, whose address is in the Program listing. He also sent along a copy of an overlay he places on the keypad to assist in determining the proper numeric codes to use with each letter or symbol. Note that this program supports lower case, also.

SYM PRINTER GRAPHICS

Jack Gierysc has sent us a number of graphics programs for the SYM Visible Memory System, including one to draw the figure shown here. Jack is one-up on us in having a graphic printer. The original of the figure was copied from the Visible Memory bit image with the Paper Tizer 445 with graphics option. To quote Jack directly, "This printer (or one



like it) is an excellent addition to the 'Perfect' system, which includes MTU's Visible Memory. I'll also be trying MPI's 886 printer so I'll be able to do a comparison."

WILL THERE BE A VOLUME 3?

We know of no computer newsletters which have survived past their third year as originally conceived. Eric Rehnke's KIM-1/6502 User Notes (later renamed 6502 User Notes), a bimonthly, made it only through No. 17, and that issue was many, many months late! All subscribers received copies of the first issue of the new magazine COMPUTE, in lieu of the never issued No. 18. COMPUTE, itself, is actually the successor to the "late" Pet Gazette. VIPER, for the RCA VIP, completed only two volumes, and only after a long interval did Vol. 3 begin to appear, under a new editor.

Donald Clem, editor of The TARGET, an AIM65 newsletter, apologizes for the several month delay of the most recent issue of the bimonthly. We ourselves had to prepare a double issue last year to fill a gap, and cut down from a bimonthly to a quarterly, to make the pain not less intense, just less frequent. On the other hand, MICRO and COMPUTE, went from bimonthlies to monthlies during the past year.

Just what is the "Viability Factor"? It would appear to be a paid, not "volunteer", multi-person staff, which obviously means revenue other than from subscriptions, such as advertising, software, hardware, other publications, etc. We would never have gone into the second year, had there not been the potential of hardware/software sales to pay for Jean's time. She now has two part-time employees to help in filling orders.

On the editorial side, however, we have had no help until now. The phone calls keep us so busy that we cannot possibly keep up with the correspondence, including written requests for urgent help.

The diskettes, cassettes, and hard copy manuscripts containing items for publication and/or sale are back-logging. These include Campbell's "SARGON II for the SYM", Vrtis' "Tiny Pilot for SYM", Holt's "TECO for SYM", to list just a few examples. There is far more quality software ALMOST AVAILABLE for the SYM than we would ever have thought possible a year ago.

SYM-PHYSIS 8:20

Publishing untested programs, however, or marketing undebussed, undocumented, software would immediately release a flood of letters and phone calls full of wrath and questions. Far better to polish the items first, and get letters and calls full of thanks and praise. Incidentally, thanks for all of the ego-building letters; we have received only one critical letter since we started, and when we pointed out that the writer had somehow forgotten to read the instructions, he wrote a very nice apology.

Next year we will be traveling much of the time, thanks to a sabbatical leave, and have wondered how or even whether to continue with Volume 3, but that problem is being resolved. Denny Hall has joined us as an associate editor with this issue. He is a psychologist by training, bought an Apple to learn on, and is another migrant into the computer field for economic reasons. He is building up speed on the SYM very quickly, will handle all correspondence and letters for help, and will review and edit all software submissions.

He will be preparing a "SYM-1 User's Manual", which will be designed to help new owners with no previous electronics background to set the most from their SYMs. This manual will make the SYM a much more appropriate entry-level system for a much broader class of potential users, and will help sell more SYMs, and get more software development going, and so on and on. We feel that neither Commodore (KIM), nor Synertek (SYM) ever realized the potential of this vast market. While Commodore is going after one segment of the educational market with PET, we feel that SYM can also fill the hardware segment of the educational market. We intend to pursue this area further.

With Denny aboard we can fulfill some of the above objectives, look forward to a Volume 3, and perhaps even a return to a bimonthly publication.

SOME COMMENTS FROM DAVE KEMP

Here are some extracts from a letter from Dave which you may find of interest:

Dear Lux,

I enjoyed Hugh Campbell's letter in issue 5/6. The November or December issue of Softalk (an Apple magazine) had a good article on program protection which said, in effect, that 'to some people, protecting a piece of software with anti-copying code is like waving a red flag in front of a bull.' I couldn't agree more. I rarely actually use commercial software but copying it is loads of fun - there is nothing more satisfying than outwitting a fellow programmer.

A while back you asked in what format software should be published in SYM-physis. Of the three possible formats - hex dump, disassembly, and commented source, - the disassembly is the worst. It takes up lots of space on the page but contains no more information than a hex dump. I wish you would publish the software you now distribute on cassette in hex format and let us readers type it in and do our own disassembly. You could still offer cassettes for those who don't like to type, but I suspect that your copying workload would drop considerably.

Regards,

David

David P. Kemp

"FAIL-SAFEING" THE VIAS ON RESET

In a very brief note, Philip Kohl taught us two very important facts about the SYM, one involving the use of the VIAs, the other concerning BASIC. We'll expand on both of his lessons in this issue. Both were based on his study of the Giervic EPROM Programmer described in Issue No. 5/6

The specs for the 6522 point out that RESET sets most internal registers to zero, and sets both the A and B ports to the INPUT state. This is supposed to ensure that no spurious outputs are generated, meaning no outputs not under computer control. So why do the relays in Jack's PROM Burner reset to the closed condition, applying +5 V, and +25 V to the EPROM socket, an obviously 'risky' condition? One should not insert a chip with power-on, nor should one leave an EPROM sitting around indefinitely with an unnecessary +25 V applied.

The answer is obvious once you figure out the reason for the FF's when you enter a .V AB00 after RESET. You set the following:

```
AB00 FF FF 00 00 XX YY FF FF,ZZ
```

The 00's indicate that the ports are set as inputs, all right, but the FF's at \$AB00 and \$AB01 indicate that all ones are being input. "Floating" input lines tend high, signaling ones. These floating input lines going to the relay-driving 7404 from 3PB3 and 3PB4 are treated as if they were high outputs, and the relays close.

The solution is to "unfloat" the lines to a low state, thus allowing the relays to "float" open. Phil suggests a 1 K resistor to ground at each of the 7404 inputs. Very simple, eh? (as many of our Canadian friends would express it!). With a non-inverting buffer this problem would not occur, but it still might be safer to actually tie the line high, rather than to depend on the up-float.

We recommend installing a back biased diode across each relay coil to shunt the current which the coil inductance keeps flowing after switch-off back into itself until it peacefully decays.

Phil also replaced the 25 V relay driven by 3PB5 with a SPDT type (Mazecraft W172 DIP-1, or equivalent). He connected the 2CB2 signal (AA-5) to the normally closed contact and the +25 V supply to the normally open contact.

He indicated also how to modify the circuit to handle 2732's, by using 3PB3 to carry the A11 signal to pin 21, and using 3PB4 and 3PB5 to drive the two relays. The required software changes should be obvious, on studying the program.

SPEEDING UP BASIC PROGRAMS

Phil Kohl reports that it takes Jack Giervic's EPROM Burner Program some 04m26s to program a complete 2716 (somehow it always seems so much longer at our place!). The actual "burning" time is 01m40s (nominal). Thus the BASIC "overhead" time is 02m46s. By replacing the constants within the program with predefined variables (which need be computed only once), he reduced the total running time to 03m26s; the overhead time is now only 01m46s, a large percentage reduction.

One minute seems like a long, long, time when burning an EPROM, since there is absolutely no sign of any action, except for the flashing red LED we installed to warn us of power-on. The saving is not only in time, but in the anxiety and uncertainty of waiting, waiting, waiting..... So, before we burn our next EPROM, we'll first burn out all our constants! We can then save TWO minutes when we start burning 2732's!

ON .CT AND THE USE OF THE DISK VECTORS

RAE-1 has three "undocumented" commands intended for disk system linkage. If you have no disks you may use them for other purposes. As a matter of fact, to correct a "bug" in RAE which does not permit .CT to work as it should (see section 10.0 of the Reference Manual) one of the disk vectors (associated with L0ad) is used to vector to a patch.

The "fix" consists of setting the cassette/disk flag at \$EE to 01 (to indicate disk), which vectors the program through the DISC.IN vector at \$F6,\$F7 to the patch and then back to RAE's cassette input program. The purpose of the patch is to force the readings of the programs on the cassette in sequence no matter what the ID may be. The patch may not be needed (have not verified this) if the last PUT or GET left a 00 ID at \$0100. If the patch is present GET will always read the next file on the cassette.

If you have the third printing of the manual the patch as presented may be in the middle of your source code if you change SET. Use the following instead:

```
Set $EE      to 01
Set $F6,$F7 to A0 00
Set $A0-$A7 to A9 00 8D 10 01 4C 68 EF
```

A TAPE DIRECTORY SYSTEM

What else can you do with the disk vectors if you don't have disks? Let Bill Wharrie tell you. We didn't have time to try this ourselves, but the idea looks good. The source can be assembled with .CT (see above!), in a 16K system. As noted below, our checksum after assembly did not agree with the one given by Bill.

```
0010      A Tape Directory System
0020      Bill Wharrie
0030      272 Erb St. W.
0040      Waterloo, Ont.
0050      Canada N2L 3G1
0060      Dec. 1980
0070
0080 The system described below uses the RAE commands
0090 >DC, >EN and >LO to implement a cassette tape
0100 directory. The command >DC becomes Directory Command
0110 and supports six sub-commands as detailed below.
0120
0130      COMMAND      DESCRIPTION
0140      >DC C Fnn name Create file with #nn and NAME name
0150      >DC D Fnn      Delete file #nn
0160      >DC G          Get directory from tape
0170      >DC I          Initialize (clear) directory
0180      >DC L          List directory
0190      >DC P date    Put directory to tape with optional
0200                      date code.
0210 The system assigns space on the cassette tape in
0220 a straightforward manner -- all files begin at
0230 multiples of 15 on the tape counter. It is up to
0240 the operator to correctly position the tape. The
0250 DIRECTORY file is arbitrarily assigned file #00
0260 and cassette counter 0000. The user should position
0270 the tape some distance in from the leader (I use
0280 10 counts), stop the tape, zero the tape counter,
0290 and then write the directory file. This can serve
0300 as a zero reference when using the tape later --
```

```
0310 Just load the tape, type >DC G (cr) (cr), start
0320 the tape, and zero the counter when the sync search
0330 indicator extinguishes. The Create sub-command
0340 checks to see if the file# is already in use, and
0350 puts the file in the first available space, if found.
0360 File names may be up to 13 characters in length --
0370 longer names will be truncated. The directory file
0380 structure uses 16 bytes for each file -- 1 byte for
0390 the file #, two bytes for the tape counter and 13
0400 for the name. Space is provided for 16 files, including
0410 the directory.
0420
0430 The syntax for the ENter and LOad commands is as follows:
0440
0450      >EN Fnn      ENter (write) file #nn to tape
0460      >LO Fnn      LOad (read) file #nn from tape
0470
0480 The file number MUST be supplied.
0490 Each command searches the
0500 directory for the specified file #. If it finds the file,
0510 it prints the directory line and then prompts the user
0520 as follows:
0530
0540 CUE TAPE TO nnnn
0550 THEN HIT RTRN.
0560
0570 NOTE: I am not using remote control of my cassette.
0580 Therefore the routines that prompt the user DO NOT turn
0590 on the remote control.
0600
0610      ERROR MESSAGES
0620
0630 The following error codes are used by the system
0640 The commands that may return these errors are given
0650 in parentheses following the description
0660
0670 30 Illegal sub-command (DC)
0680 31 Sub-command missing (DC)
0690 32 "F" missing before file# (EN, LO, DC C, DC D)
0700 33 Directory full (DC C)
0710 34 File# in use (DC C)
0720 35 File# missing (EN, LO)
0730 36 File not found (DC D, EN, LO)
0740
0750      USING THE SYSTEM
0760
0770 The object code occupies 3 pages of
0780 memory and the directory itself one page. The
0790 object code loads to $1B00 to $1DFE and the
0800 directory is from $1E00 to $1EFF.
0810 The checksum for .V 1B00;1DFE is $3B54.
0820 (EDITOR'S NOTE: OUR CHECKSUM WHEN ASSEMBLED WAS $3B4E!)
0830 Load a cassette in the recorder, advance the
0840 tape some distance past the leader, and zero the counter.
0850 Enter >DC I (cr), then >DC P (cr).
0860 After the prompt press record and play
0870 on the recorder and hit return -- this will mark your
0880 zero point on the tape.
0890 As you add files to the tape, use >DC C to add them
0900 to the directory. Before you remove the tape from
0910 the recorder, write the updated directory to the tape
0920 with >DC P date(cr).
```

```

0010 ; CASSETTE TAPE DIRECTORY SYSTEM
0020 ; WRITTEN BY BILL WHARRIE
0030 ; AUGUST 1980.
0040 ;
0050 ; USES RAE'S DC, EN AND LO COMMANDS
0060 ; DC LOOKS AFTER THE DIRECTORY
0070 ; EN DOES A 'PUT'
0080 ; LO DOES A 'GET'
0090 ;
0100 ; SYM MONITOR ROUTINES
0110 ACCESS .DE $88B6
0120 ASCNIB .DE $8275
0130 OUTXAH .DE $82F4
0140 OUTBYT .DE $82FA
0150 NIBASC .DE $8309
0160 CRLF .DE $834D
0170 INCHR .DE $8A1B
0180 OUTCHR .DE $8A47
0190 LOADT .DE $8C78
0200 DUMPT .DE $8E87
0210 ; SYM VARIABLES
0220 TAPDEL .DE $A630
0230 TEMP .DE $A647
0240 P1L .DE $A64E
0250 P2L .DE $A64C
0260 P3L .DE $A64A
0270 ; RAE REFERENCES
0280 ERROR .DE $B00E
0290 RAE.HOT .DE $B05E
0300 NXTFARM .DE $B502
0310 RAE.PUT .DE $EF95
0320 DCVEC .DE $EC
0330 ENVEC .DE $F0
0340 LOVEC .DE $F2
0350 GETBYT .DE $B2E6
0360 GETRUF .DE $B6A0
0370 DUCMD .DE $B6AE
0380 CRT .DE $135
0390 ;
0400 .BA $1B00
0410 .OS
0420 ; SET UP RAE VECTORS
0430 PATCH LDA #L,DC
0440 STA *DCVEC
0450 LDA #H,DC
0460 STA *DCVEC+1
0470 LDA #L,ENT
0480 STA *ENVEC
0490 LDA #H,ENT
0500 STA *ENVEC+1
0510 LDA #L,LOAD
0520 STA LOVEC
0530 LDA #H,LOAD
0540 STA LOVEC+1
0550 RTS
0560 ; DC MAIN ROUTINE
0570 ; READ SUB-COMMAND AND JUMP
0580 DC JSR ACCESS
0590 CPY ##50
0600 BNE DC1 ;SUB-CMD MISSING IF Y=50
0610 LDX ##31
0620 JMP (ERROR)
0630 DC1 LDA CRT,Y
0640 CMP #'I
0650 BNE =+4
1B00- A9 1B
1B02- 85 EC
1B04- A9 1B
1B06- 85 ED
1B08- A9 46
1B0A- 85 F0
1B0C- A9 1D
1B0E- 85 F1
1B10- A9 82
1B12- 8D F2 00
1B15- A9 1D
1B17- 8D F3 00
1B1A- 60
1B1B- 20 86 8B
1B1E- C0 50
1B20- D0 05
1B22- A2 31
1B24- 6C 0E B0
1B27- B9 35 01
1B2A- C9 49
1B2C- D0 03

```

```

1B2E- 4C CD 1B 0660
1B31- C9 43 0670
1B33- D0 03 0680
1B35- 4C 59 1B 0690
1B38- C9 4C 0700
1B3A- D0 03 0710
1B3C- 4C F4 1B 0720
1B3F- C9 50 0730
1B41- D0 03 0740
1B43- 4C 5B 1C 0750
1B46- C9 47 0760
1B48- D0 03 0770
1B4A- 4C DC 1C 0780
1B4D- C9 44 0790
1B4F- D0 03 0800
1B51- 4C 06 1D 0810
1B54- A2 30 0820
1B56- 6C 0E B0 0830
0840 ; >DC C Fnn
0850 ; CREATE FILE #nn
0860 CREATE INY
0870 INY ;POINT TO ARGUMENTS
0880 LDA CRT,Y
0890 CMP #'F ;FILE# PRECEDED BY 'F'
0900 BEQ CR1
0910 LDX ##32
0920 JMP (ERROR) ;'F' NOT FOUND
0930 CR1 INY ;POINT TO FILE #
0940 LDX ##A
0950 JSR GETBYT
0960 LDA #10A
0970 CR2 PHA ;SAVE FILE #
0980 JSR FINDFILE ;SEARCH DIRECTORY
0990 BCC CR3 ;CARRY CLEAR IF OK
1000 TAX
1010 JMP (ERROR)
1020 CR3 PLA ;RECOVER FILE #
1030 STA DRCTRY,X !X POINTS TO NEXT FREE SPACE
1040 INX
1050 INX ;POINT TO LO HALF OF COUNTER
1060 CLC
1070 SED
1080 LDA DRCTRY-16,X !GET CNTR# OF PREVIOUS FILE
1090 ADC ##15 ;ALLOW COUNT OF 15 BTWN FILES
1100 STA DRCTRY,X ;STORE NEW CNTR#
1110 DEX
1120 LDA DRCTRY-16,X
1130 ADC #0 ;PROPAGATE CARRY
1140 STA DRCTRY,X
1150 CLD
1160 INX
1170 INX ;POINT TO FILE NAME
1180 INY ;POINT TO FILE NAME IN BUFFER
1190 FNLOOP LDA CRT,Y ;MOVE FILE NAME FROM BUFFER...
1200 STA DRCTRY,X ; TO DIRECTORY
1210 INY
1220 INX
1230 TXA
1240 AND ##F ;DO UNTIL X IS A
1250 BNE FNLOOP ;MULTIPLE OF 10
1260 JMP RAE.HOT ;RETURN TO RAE
1270 ; LOOK FOR FREE SPACE IN DIRECTORY
1280 FINDFILE JSR LOOKUP
1290 BCC ERROR34
1300 LDX #0
1B7A- 68 1020
1B7B- 9D 00 1E 1030
1B7E- EB 1040
1B7F- EB 1050
1B80- 1B 1060
1B81- FB 1070
1B82- BD F0 1D 1080
1B85- 69 15 1090
1B87- 9D 00 1E 1100
1B8A- CA 1110
1B8B- BD F0 1D 1120
1B8E- 69 00 1130
1B90- 9D 00 1E 1140
1B93- D8 1150
1B94- EB 1160
1B95- EB 1170
1B96- CB 1180
1B97- B9 35 01 1190
1B9A- 9D 00 1E 1200
1B9D- CB 1210
1B9E- EB 1220
1B9F- BA 1230
1BA0- 20 0F 1240
1BA2- D0 F3 1250
1BA4- 4C 5E B0 1260
1BA7- 20 2F 1D 1280
1BA8- 90 19 1290
1BAC- A2 00 1300

```

```

1BAE- BD 00 1E 1310 FLOOP2 LDA DRCTRY,X
1BB1- C9 FE 1320 CMP ##FE
1BB3- F0 0D 1330 BEQ OKEXIT ;FOUND FREE ENTRY
1BB5- C9 FF 1340 CMP ##FF
1BB7- F0 09 1350 BEQ OKEXIT ;INSERT AT END OF DIRECTORY
1BB9- 8A 1360 TXA
1BBA- 18 1370 CLC
1BBB- 69 10 1380 ADC ##10
1BBD- B0 0A 1390 BCS ERROR33 ;DRCTRY FULL
1BBF- AA 1400 TAX
1BC0- D0 EC 1410 BNE FLOOP2 ;ALWAYS
1BC2- 18 1420 OKEXIT CLC
1BC3- 90 07 1430 BCC FEXIT
1BC5- A9 34 1440 ERROR34 LDA ##34
1BC7- D0 02 1450 BNE =+3
1BC9- A9 33 1460 ERROR33 LDA ##33
1BCB- 38 1470 SEC
1BCC- 60 1480 FEXIT RTS
1490 ;
1500 .CT F2

0010 ; TAPE DIRECTORY - FILE 2
0020 ;
0030 ; >DC I -- INITIALIZE DIRECTORY
0040 ; FILL DIRECTORY FILE WITH SPACES
0050 ; SET FIRST LINE
1BCD- A2 00 0060 INIT LDX #0
1BCF- A9 20 0070 LDA #'
1BD1- 9D 00 1E 0080 ILOOP1 STA DRCTRY,X
1BD4- E8 0090 INX
1BD5- D0 FA 0100 BNE ILOOP1 ;FILL WITH SPACES
0110 ;
1BD7- A2 10 0120 LDX ##10
1BD9- A9 FF 0130 ILOOP2 LDA ##FF
1BDB- 9D 00 1E 0140 STA DRCTRY,X ;MARK END-OF-DIRECTORY
1BDE- BA 0150 TXA
1BDF- 18 0160 CLC
1BE0- 69 10 0170 ADC ##10 ;MARK EVERY 16TH LOCATION
1BE2- AA 0180 TAX
1BE3- D0 F4 0190 BNE ILOOP2
0200 ;
1BE5- A2 08 0210 LDX #11
1BE7- BD D3 1D 0220 ILOOP3 LDA FILE0,X ;SET FIRST LINE
1BEA- 9D 00 1E 0230 STA DRCTRY,X
1BED- CA 0240 DEX
1BEE- 10 F7 0250 BPL ILOOP3
1BF0- 18 0260 CLC
1BF1- 4C 5E B0 0270 JMP RAE.HOT
0280 ;
0290 ; >DC L -- LIST DIRECTORY
0300 ;
1BF4- 20 4D 83 0310 LIST JSR CRLF
1BF7- A2 00 0320 LDX #0
1BF9- BD BC 1D 0330 LOOP1 LDA HEADER,X ;TYPE HEADER
1BFC- 30 06 0340 BMI LSTART ;##FF MARKS END-OF-MESSAGE
1BFE- 20 47 BA 0350 JSR OUTCHR
1C01- E8 0360 INX
1C02- D0 F5 0370 BNE LOOP1 ;ALWAYS
0380 ;
1C04- A2 00 0390 LSTART LDX #0
1C06- BD 00 1E 0400 LOOP2 LDA DRCTRY,X
1C09- C9 FE 0410 CMP ##FE
1C0B- F0 0D 0420 BEQ LSKIP ;SKIP OVER DELETED ENTRIES

```

SYM-PHYSIS 8:27

```

1C0D- C9 FF 0430 CMP ##FF
1C0F- F0 11 0440 BEQ ENDLIST
1C11- 20 25 1C 0450 JSR PRTLINE ;TYPE CURRENT DIRECTORY LINE
1C14- E0 00 0460 L2 CPX #0 ;CHECK IF END-OF-DRCTRY REACHED
1C16- F0 0A 0470 BEQ ENDLIST
1C18- D0 EC 0480 BNE LOOP2
1C1A- 8A 0490 LSKIP TXA
1C1B- 18 0500 CLC
1C1C- 69 10 0510 ADC ##10
1C1E- AA 0520 TAX
1C1F- 4C 14 1C 0530 JMP L2
1C22- 4C 5E B0 0540 ENDLIST JMP RAE.HOT
0550 ; PRINT CURRENT DIRECTORY LINE
1C25- 20 4D 83 0560 PRTLINE JSR CRLF
1C28- A9 46 0570 LDA #'
1C2A- 20 47 BA 0580 JSR OUTCHR ;TYPE 'F'
1C2D- BD 00 1E 0590 LDA DRCTRY,X
1C30- 20 FA B2 0600 JSR OUTBYT ;TYPE FILE NUMBER
1C33- A9 20 0610 LDA #'
1C35- 20 47 BA 0620 JSR OUTCHR
1C38- E8 0630 INX
1C39- BD 00 1E 0640 LDA DRCTRY,X
1C3C- 20 FA B2 0650 JSR OUTBYT ;TYPE CASSETTE COUNTER
1C3F- E8 0660 INX
1C40- BD 00 1E 0670 LDA DRCTRY,X
1C43- 20 FA B2 0680 JSR OUTBYT
1C46- A9 20 0690 LDA #'
1C48- 20 47 BA 0700 JSR OUTCHR
1C4B- E8 0710 PRTL0OP INX
1C4C- 8A 0720 TXA
1C4D- 29 0F 0730 AND ##0F ;DO UNTIL X IS A...
1C4F- F0 09 0740 BEQ PRTEXIT ; MULTIPLE OF 16
1C51- BD 00 1E 0750 LDA DRCTRY,X
1C54- 20 47 BA 0760 JSR OUTCHR
1C57- 4C 4B 1C 0770 JMP PRTL0OP
1C5A- 60 0780 PRTEXIT RTS
0790 ; >DC P date -- WRITE (PUT) DIRECTORY TO TAPE
0800 ; ADD date TO HEADER IF SUPPLIED
1C5B- C8 0810 PUTD INY
1C5C- 20 02 B5 0820 JSR #B502 ;POINT TO NEXT NON-SPACE IN BUFFER
1C5F- C0 50 0830 CPY ##50
1C61- F0 12 0840 BEQ P1 ;NO DATE ENTERED
1C63- A2 00 0850 LDX #0
1C65- B9 35 01 0860 FLOOP1 LDA CRT,Y ;MOVE DATE INFO TO HEADER
1C68- C9 20 0870 CMP #' ;END IF SPACE READ
1C6A- F0 09 0880 BEQ P1
1C6C- 9D 0C 1E 0890 STA DRCTRY+12,X
1C6F- C8 0900 INY
1C70- E8 0910 INX
1C71- E0 04 0920 CPX #4
1C73- D0 F0 0930 BNE FLOOP1
1C75- A2 00 0940 P1 LDX #0 ;POINT TO FILE 0
1C77- 20 AB 1C 0950 JSR CUE ;PROMPT USER
0960 ; SET UP PARAMETERS FOR SYM SAVE
1C7A- A9 00 0970 LDA #0 ;FILE NUMBER 0
1C7C- 8D 4E A6 0980 STA P1L
1C7F- 8D 4F A6 0990 STA P1L+1
1C82- A9 00 1000 LDA #L,DRCTRY
1C84- 8D 4C A6 1010 STA P2L
1C87- A9 1E 1020 LDA #H,DRCTRY
1C89- 8D 4D A6 1030 STA P2L+1
1C8C- A9 00 1040 LDA #L,ENDADD
1C8E- 8D 4A A6 1050 STA P3L
1C91- A9 1F 1060 LDA #H,ENDADD
1C93- 8D 4B A6 1070 STA P3L+1

```

SYM-PHYSIS 8:28

```

1C96- A9 04      1080      LDA #4
1C98- 8D 30 A6   1090      STA TAPDEL ;LONG SYNC
1C9B- A0 80      1100      LDY ##80
1C9D- 20 87 8E   1110      JSR DUMPT
1CA0- A9 01      1120      LDA #1
1CA2- 8D 30 A6   1130      STA TAPDEL
1CA5- 4C 5E B0   1140      JMP RAE.HOT
                1150 ;
                1160      .CT F3

                0010 ; CASSETTE TAPE DIRECTORY - FILE 3
                0020 ; CUE
                0030 ; TYPE DIRECTORY LINE,
                0040 ; THEN TYPE USER PROMPT,
                0050 ; WAIT FOR USER.
1CAB- 20 25 1C   0060 CUE      JSR PRTLINE ;TYPE DIRECTORY LINE
1CAB- 20 4D 83   0070      JSR CRLF
1CAE- 8A         0080      TXA
1CAF- 3B         0090      SEC
1CB0- E9 10      0100      SBC ##10 ;RESET LINE POINTER
1CB2- AA         0110      TAX
1CB3- EB         0120      INX ;POINT TO CASSETTE COUNTER
1CB4- EB         0130      INX
1CB5- A0 03      0140      LDY #3 ;XFER COUNTER TO MESSAGE BUFFER
1CB7- BD 00 1E   0150 FILLOOP  LDA DRCTRY,X ;GET HEX BYTE
1CBA- 20 09 83   0160      JSR NIBASC ;CONVERT LO NIBBLE
1CBD- 99 EB 1D   0170      STA CUEMSG+12,Y
1CC0- 88         0180      DEY
1CC1- BD 00 1E   0190      LDA DRCTRY,X
1CC4- 4A         0200      LSR A
1CC5- 4A         0210      LSR A
1CC6- 4A         0220      LSR A
1CC7- 4A         0230      LSR A
1CC8- 20 09 83   0240      JSR NIBASC ;CONVERT HI NIBBLE
1CCB- 99 EB 1D   0250      STA CUEMSG+12,Y
1CCE- CA         0260      DEX
1CCF- 88         0270      DEY
1CD0- 10 E5      0280      BFL FILLOOP
1CD2- 20 F5 1C   0290      JSR MSG ;TYPE PROMPT MESSAGE
1CD5- 20 1B 8A   0300      JSR INCHR ;WAIT FOR USER INPUT
1CD8- 20 4D 83   0310      JSR CRLF ;ECHO CRLF TO ANY CHAR ENTERED
1CDB- 60         0320      RTS
                0330 ;
                0340 ; >DC G -- GET DIRECTORY FROM TAPE
1CDC- A2 00      0350 GETD      LDX #0
1CDE- 20 A8 1C   0360      JSR CUE ;PROMPT USER
1CE1- A9 00      0370      LDA #0
1CE3- 8D 4E A6   0380      STA P1L ;GET FILE 00
1CE6- A0 80      0390      LDY ##80
1CE8- 20 78 BC   0400      JSR LOADT
1CEB- 90 05      0410      BCC OKLOAD
1CED- A2 17      0420      LDX ##17
1CEF- 6C 0E B0   0430      JMP (ERROR)
1CF2- 4C 5E B0   0440 OKLOAD  JMP RAE.HOT
                0450 ;
1CF5- 20 4D 83   0460 MSG      JSR CRLF
1CF8- A2 00      0470      LDX #0
1CFA- BD DF 1D   0480 MSGLOOP  LDA CUEMSG,X
1CFD- 30 06      0490      BMI MSGEXIT
1CFF- 20 47 BA   0500      JSR OUTCHR
1D02- EB         0510      INX
1D03- D0 F5      0520      BNE MSGLOOP
1D05- 60         0530 MSGEXIT  RTS
                0540 ;
1D06- CB         0550 DELETE  INY ;DELETE FILE

```

SYM-PHYSIS 8:29

```

1D07- CB         0560      INY ;POINT TO FILE NUMBER
1D08- B9 35 01   0570      LDA CRT,Y
1D0B- C9 46      0580      CMP #'F
1D0D- F0 05      0590      BEQ D1
1D0F- A2 32      0600 DE1     LDX ##32
1D11- 6C 0E B0   0610      JMP (ERROR) ;'F' NOT FOUND
1D14- C8         0620 D1      INY
1D15- A2 0A      0630      LDX ##A
1D17- 20 E6 B2   0640      JSR GETBYT ;GET BYTE FROM BUFFER
1D1A- AD 0A 01   0650      LDA $10A
1D1D- 20 2F 1D   0660      JSR LOOKUP ;LOOK UP FILE#
1D20- B0 08      0670      BCS NOTFOUND
1D22- A9 FE      0680      LDA ##FE ;OVERWRITE FILE NUMBER
1D24- 9D 00 1E   0690      STA DRCTRY,X ;X POINTS TO ENTRY
1D27- 4C 5E B0   0700      JMP RAE.HOT
1D2A- A2 36      0710 NOTFOUND LDX ##36
1D2C- 6C 0E B0   0720      JMP (ERROR) ;FILE NOT FOUND
                0730 ;
                0740 ; LOOK UP FILE # IN DIRECTORY
                0750 ; CARRY SET IF FILE NOT FOUND
                0760 ; CARRY CLEAR IF FILE FOUND AND
                0770 ; X POINTS TO LINE IN DIRECTORY FILE
1D2F- A2 00      0780 LOOKUP   LDX #0
1D31- DD 00 1E   0790 LKLOOP   CMP DRCTRY,X
1D34- F0 0C      0800      BEQ FNDFILE
1D36- 48         0810      PHA ;SAVE FILE # BEING SEARCHED FOR
1D37- 8A         0820      TXA
1D38- 18         0830      CLC
1D39- 69 10      0840      ADC ##10 ;POINT TO NEXT ENTRY
1D3B- B0 07      0850      BCS LRET ;END OF DIRECTORY
1D3D- AA         0860      TAX
1D3E- 68         0870      PLA
1D3F- 4C 31 1D   0880      JMP LKLOOP
1D42- 18         0890 FNDFILE  CLC
1D43- 60         0900      RTS
1D44- 68         0910 LRET     PLA
1D45- 60         0920      RTS
                0930      .CT F4

                0010 ; CASSETTE TAPE DIRECTORY - FILE 4
                0020 ;
                0030 ; ENTER COMMAND
                0040 ; LOOK UP FILE IN DIRECTORY
                0050 ; CUE OPERATOR
                0060 ; WRITE TO TAPE IN RAE FORMAT
0070 ENT        CPY ##50
0080          RCS ENERR1
0090          JSR GETBUF
0100          CMP #'F
0110          BNE ENERR2
0120          INY
0130          LDX ##A
0140          JSR GETBYT ;GET FILE# FROM RAE BUFFER
0150          LDA $10A
0160          JSR LOOKUP ;SEARCH DIRECTORY FOR FILE
0170          BCC EN1
0180          LDX ##36
0190          JMP (ERROR) ;FILE NOT FOUND
0200 EN1        JSR CUE
0210          LDY #0 ;RESET BUFFER POINTER
0220          LDA #'P
0230          STA CRT ;CHANGE INPUT BUFFER...
0240          LDA #'U ; TO A 'PUT' COMMAND
0250          STA CRT+1
0260          LDX #0 ;RESET X FOR COMMAND SEARCH
0270          STX ##EF ;CLEAR PUT FLAG

```

SYM-PHYSIS 8:30

```

1D77- 4C AE B6 0280      JMP DOCMD      ;LET RAE DO THE WORK
1D7A- A2 35 0290 ENERR1  LDX ##35
1D7C- D0 E3 0300      BNE ENERR      ;FILE# MISSING
1D7E- A2 32 0310 ENERR2  LDX ##32
1D80- D0 DF 0320      BNE ENERR      ;'F' NOT FOUND
0330 ;
0340 ; LOAD COMMAND
0350 ; GET FILE NUMBER FROM BUFFER
0360 ; PROMPT USER
0370 ; READ (GET) FILE FROM TAPE

1D82- C0 50 0380 LOAD    CPY ##50
1D84- B0 2E 0390      BCS LERR1      ;TEST FOR ARGUMENT
1D86- 20 A0 B6 0400      JSR GETBUF     ;GET NEXT CHAR FROM RAE
1D89- C9 46 0410      CMP #'F
1D8B- D0 2B 0420      BNE LERR2      ;IS IT AN 'F'?
1D8D- C8 0430      INY
1D8E- A2 0A 0440      LDX ##A
1D90- 20 E6 B2 0450      JSR GETBYT     ;GET FILE NUMBER
1D93- AD 0A 01 0460      LDA #10A
1D96- 20 2F 1D 0470      JSR LOOKUP     ;LOOK UP FILE# IN DIRECTORY
1D99- 90 05 0480      BCC L01        ;CARRY SET IF FILE NOT FOUND
1D9B- A2 36 0490      LDX ##36
1D9D- 6C 0E B0 0500 LERR   JMP (ERROR)
1DA0- 20 A8 1C 0510 L01    JSR CUE        ;PROMPT USER
1DA3- A0 00 0520      LDY #0
1DA5- A2 00 0530      LDX #0
1DA7- A9 47 0540      LDA #'G        ;CHANGE RAE BUFFER TO...
1DA9- BD 35 01 0550      STA CRT        ; A 'GET' COMMAND.
1DAC- A9 45 0560      LDA #'E
1DAE- BD 36 01 0570      STA CRT+1
1DB1- 4C AE B6 0580      JMP DOCMD      ;LET RAE DO IT
0590 ;
1DB4- A2 35 0600 LERR1   LDX ##35
1DB6- D0 E5 0610      BNE LERR
1DB8- A2 32 0620 LERR2   LDX ##32
1DBA- D0 E1 0630      BNE LERR
0640 ;
0650 ; MESSAGE BLOCKS
0660 ;
1DBC- 46 49 4C 0670 HEADER .BY 'FIL CNTR ----NAME-----' $FF
1DBF- 20 43 4E
1DC2- 54 52 20
1DC5- 2D 2D 2D
1DC8- 2D 4E 41
1DCB- 4D 45 2D
1DCE- 2D 2D 2D
1DD1- 2D FF
1DD3- 00 00 00 0680 FILE0 .BY 0 0 0 'DIRECTORY'
1DD6- 44 49 52
1DD9- 45 43 54
1DDC- 4F 52 59
1DDF- 43 55 45 0690 CUMSG .BY 'CUE TAPE TO 0000' $0D $0A
1DE2- 20 54 41
1DE5- 50 45 20
1DE8- 54 4F 20
1DEB- 30 30 30
1DEE- 30 0D 0A
1DF1- 54 48 45 0700 .BY 'THEN HIT RETN' $FF
1DF4- 4E 20 48
1DF7- 49 54 20
1DFA- 52 45 54
1DFD- 4E FF
0710 ; DIRECTORY FILE IS HERE
0720 .BA $1E00
0730 DRCTRY .DS 256
0740 ENDADD .BI =
0750 .EN SYM-PHYSIS 8:31
1E00-

```

ARCADE TYPE GAMES

We have had numerous requests for "real-time" arcade games, and have received several such programs, some for the KTM-2/80, and some for the MTU Visible Memory. There is not enough room available to print any one of them in toto, but we present below the instruction sheet which goes with Jack Gieryic's "HI-RES LASER GUN" game, JBF-6, to give you an idea of the kinds of things which can be done with the Visible Memory.

from JACK BUILT PROGRAMS

```

*****
*
* HI-RES LASER GUN *
*
*****

```

This 1K machine language program presents the user with 8 targets traveling across the top of the screen and a movable laser gun on the bottom. Gun positioning and fire control is via the hex keypad on the SYM.

Hardware Requirements: 1K of RAM at \$1C00
 Micro Technology's 8K Visible Memory
 General Instrument's Programmable Sound Generator
 (AY-3-8910/8912) wired per PSG Demonstrator article

No, this isn't just another entertaining game. I wrote this program to gain experience with MTU's Visible Memory. I wanted to see just how interactive the SYM could be and how close I could get to "real time". This program should be viewed as an idea generator rather than an amusement device.

First a note on the Visible Memory. This board provides 8K of additional RAM for your SYM and can be jumper selected at any 8K boundary. If you have Blalock's 4K memory expansion board on your SYM, then the Visible Memory will provide you with your next 8K of RAM. The VM board provides a composite video signal which can be fed into a monitor. The monitor display consists of 200 horizontal lines with each line consisting of 320 dots. That's 64,000 dots all under software control! Each dot can be turned on or off by simply changing a bit in memory. Each byte in the VM controls 8 consecutive dots. The first forty words on the memory represent the top line (left to right), the second 40, the 2nd line and so on down thru the 200th line. When any number of consecutive dots are turned on, the result is a solid white line on that part of the monitor. Changing the display is simply storing data into memory.

The Laser Gun program has been assembled to load at location \$1C00. Begin execution with G 1C00 CR. To move the sun left(right) simply press the number 1(3) on the hex keypad. From any one position you may fire 10 shots and the sun becomes inactive. To reactivate it, just move the sun right or left. To fire the sun, press the number 4. Too heavy a hand will fire all 10 shots in a flash. To restart the program, press the number 7.

The PSG will let you hear the gun fire as well as provide an explosion when one of the targets is hit. It's interesting to observe how the addition of sound can change a normally silent video display. Sight is not the only sense which can be stimulated by your computer.

When an explosion occurs the entire monitor screen is inverted for an instant and then placed back in normal video. My 30MHz bandwidth,

(continued on page 8:38)

SYM-PHYSIS 8:32

ANOTHER SOFTWARE SOURCE

Kin-Ping Kwok, Flat A, 10/F, 20-22 Tung Choi St., Tat Mins Bldg., Mongkok, Kowloon, Hong Kong, several of whose ideas and programs have been published in earlier issues, sent us his version of a Disassembler-into-RAE, with permission to publish freely, at just about the same time we were putting the finishing touches on the Hissink version published in this issue.

His version contains one convenience feature not present in the Hissink version. If insufficient space has been allocated for the source code, the Kwok version stops to permit entry of a .CT, dumping to cassette, and resumption of disassembly from that point.

He has used his disassembler to prepare 'relocatable source codes' for BASIC and RAE, and has sent us cassettes containing his results. These programs would be extremely useful for anyone wishing to replace the BAS-1 and RAE-1 ROMs with RAM, and reassemble both programs into the same address space. We reproduce below, that portion of his correspondence giving his prices. We highly recommend these programs; please write him directly for any additional information.

Dear Dr. Luxenburg,

On the tape enclosed, there are four programs (each was recorded twice). They are programs related to the programs I sent you last time with the letter dated 4th May, 81.

The first program on the tape is a RAE file with some necessary internal address definitions for the reassemble of the SYM BASIC assembly listings. Documentation on the relocation of SYM BASIC is also enclosed.

The second program on the tape is the BASIC assembly listing generator executed in \$299-\$4FF. The use of it is described in last letter.

The third program is the relocated SYM BASIC executed in \$1000-\$2FFF.

The last program on tape is the RAE relocater. Documentation is also enclosed.

I would like to say again that the programs can be obtained from me.

RAE relocater including documentation -- U.S.\$15.00.

Commented source listings of the disassembler, object code and documentation (I sent you last time) -- U.S.\$15.00

BASIC assembly listing generator including a file with necessary internal address definition, documentation and cross-reference -- U.S.\$25.00

Comments and suggestions are always welcome.

Yours sincerely,

Kin-Ping Kwok

NEW PRODUCT - THE SPIKE SPIKER DELUXE POWER CONSOLE

We finally found what we had been looking for these many years, a way to "organize" the maze of 3-prong power cords coming from a collection of computer connected devices. This is the "Spike-Spiker" Deluxe Power Console, made by KALGO Electronics Co., Inc.

In addition to transient absorption and RF "hash" filtering, this neat little "black box" provides EIGHT individually-switched, 120 Volt, 60 Hertz, power outlets, a line fuse, a pilot light, and a master switch. We like it so much, that we have become dealers, and can offer it at a discount. See the addendum mailed with this issue for the price.

SYM-PHYSIS 8:33

CORRECTION TO THE DIS PROGRAM

We knew of one very minor bug to DIS, but prepared the copy for the printer anyway, just in case. The fix arrived just in time. Here 'tis: Delete lines 3440 and 3450 and insert the following lines instead:

```
3440 LSR A          ;TEST FOR ZERO PAGE FORMAT
3445 BCC NZERO     ;CODES 01,05,11
3450 CMP #09      ;ELIMINATE 21,59,4D,1D
3455 BCS NZERO
```

After the disassembly is completed verify that the source file has been properly terminated. It should contain a \$00 byte at the end-of-file address + 2. The first number on the second line of the SET message printed after disassembly is complete is the end-of file address. Then insert a .BA \$XXXX and a .CE at the front of the source and a .EN at the end, as enter AS. External address references will be flagged with error messages.

A USEFUL CASSETTE TIP

Here is a helpful suggestion from Dick Albers, Placentia, CA, followed by some additional comments from the editor:

"SYM-1's tape LOAD program uses the same memory locations as 'M' to keep track of 'current memory location' in which to store the next byte. If you have trouble loading a tape, after the 'ER XX' message, enter 'M <cr>' and SYM will display the last address loaded + 1. This can be used to adjust your volume and tone controls."

We suggest that you study, with simple examples, how RAE-1 and BAS-1 mark the ends of their files, and how and where the end pointers are set. Do this with a readable cassette. Then, when you find that you can read most, but not all, of a long cassette dump, you can use this knowledge, plus the tip above to set the pointer and end marker correctly, to rescue at least part of your cassette load.

This tip is also helpful for readable cassettes when you just want to know the ending address.

MISCELLANEA

TRI-TEK, 7808 N. 27th Ave., Phoenix, AZ 85021, lists and describes in their new catalog, the National Semiconductor complementary chip pair, the LM1871 Radio Control Encoder/Transmitter, and the LM1872 Radio Control Receiver/Decoder. The normal configuration for this pair permits the remote transfer of two analog and two binary (on/off) signals, but this may be modified. Price for each chip is \$7.45, for each spec sheet is \$0.40. Anyone tried them yet with the SYM?

MATT WILSON, author of 'HEXTOASCII' on page 8:2, has designed an inexpensive (he calls it cheap) 625 line, 50 Hz video board for SYM. He will send plans to anyone interested.

DAVE KEMP, East Coast Micro Products, Odenton, MD (ESP-1 Speech Synthesizer), along with many others, has sent us versions of APPLE SAVE and DUMP routines. We intend publishing an integrated version combining the best features of all of the many programs we have received.

DON DEPEW, Spartan Electronics, Jackson, MI, suggests identifying each program you write with a unique identifier. An example might be 8106141645TITLE/LUX. The number indicates the year, month date, and time, and the title and author follow. This makes it easy to identify the version (is this the latest, how many have there been?) and allows several tabs for organizing your programs.

SYM-PHYSIS 8:34

FULLY COMMENTED SOURCE CODE FOR MICROSOFT BASIC?

Microsoft holds its source codes to be Proprietary, and BAS-1 logs-on with a copyright notice, but presumably it is a legitimate research effort to analyze object code to determine its inner workings, and to generate, through one's own efforts, a recreation of source code. Such research could certainly also be a team effort. A purchaser would seem to be free to modify and enhance a purchased product, and indeed, to freely market such enhancements as a new product. If the "enhancements" are so "complete" that a purchaser of the enhancement need not also purchase the original product, that is another matter, indeed. We quote from a recent letter from John Hissink in this regard:

"I have been disassembling BAS-1 and have generated relocatable text. I would like to fully disassemble BAS-1 as part of a (small?) team effort, mainly out of interest in seeing how the language works, and learning the programming tricks used. It's quite well commented now, and I sit down every now and then with the printout and do some more, but it will take a year, unless I drop all other projects. Any interest with the guys in your neighborhood? If so, I'll send the latest version on a disk (it takes 33 tracks!) and we can start from that and exchange updates."

We know of a number of users who have pretty much annotated their printouts of BAS-1 disassemblies, but very few who have done so for "reassemblable" disassemblies. John, Steve Cole, and Arthur Richards have been working together to reorganize BAS, RAE, FODS, and MON (and everything else they can get their hands on) in order to create a really coherent system package.

Since each of these programs was developed independently, they have differing syntaxes. One uses spaces as delimiters, two use commas, one uses none. Command words also differ. For example, BASIC uses SAVE and LOAD for cassette operations (disk is not supported), while RAE-1 uses PUT and GET for cassette, and ENTER and LOAD for disk. Since RAE was cast in ROM, while FODS was not, we replaced FODS' SAVE with ENTER for consistency.

Note that BASIC permits no abbreviations, FODS requires three letter abbreviations, and RAE permits two letter abbreviations or even extensions of the word. In fact, what we called LOAD above was called LOOKUP in the original RAE manual (no, this is not documented in the RAE-1 Reference Manual as issued by Synertek!).

Their system will be easier to learn and use than the "standard", but at the cost of program transportability. They are carrying the concept of a "personal" computer to its ultimate limit! In practice, however, the syntax and vocabulary differences mentioned above are no real problem. All good computerists must be multilingual, anyway.

But, to get back to the main point: If you are interested in joining John's team (you will need FODS to participate), sign-on with him at 29 Knox Road, Havant, Hants, PO9, INP, England. The end result of your participation will be a fully commented source code of a version of Microsoft BASIC into which you can incorporate any enhancements you wish. You'll also end up with the most powerful DISK BASIC available!

MORE ON FORTH

FORTH is fast becoming our own personal favorite computer language. We
SYM-PHYSIS 8:35

have decided that we will become our University's resident FORTH "expert" while everyone else is setting into Pascal. We strongly recommend that you subscribe to Saturn Software, Ltd.'s (SSL) "Saturn Softnews", even if you have not yet seen FORTH, as the best way of introducing yourself to the language.

In the first issue, Jack Brown publishes, and explains, in detail, the screens (the standard page-format for FORTH source code) necessary to convert the original fis-FORTH model to the new 79-STANDARD. He also provides screens for a real-time clock, sound-effects generation, KTM-2/80 graphics, and MTU Visible Memory graphics. We feel that the best way to become fluent in any language is by reading lots of well written material in that language, and Softnews provides that material.

We have been testing, with great pleasure, Saturn Software's latest FORTH product, a disk-based 79-STANDARD FORTH, with 8 "resident" screens, and 104 (!!!) "virtual-memory" disk-resident screens, all on-line at once. With diskette changes, the useful memory is unlimited. It's the penultimate system!

We will soon be receiving a still newer product, a fully disk-linked Extended SYM-BASIC (ESB-2), for evaluation, testing, and review. ESB-1 and ESB-2 will also be supported in Saturn Softnews. We wish Jack much success in his venture. SSL's address is 8246 - 116A St., Delta, B.C., Canada V4C 5Y9.

ANOTHER RS-232 PORT FOR THE SYM

The SYM has a 20 mA current loop port which works at TTL level (+5 V), but you may need to add a 1 K resistor from the base of Q28 to ground, to improve the keyboard input reliability. We see no need ever to bring in a -Vn for current loop usage.

The so-called RS-232 port is actually an inverted TTL port, with 0 V representing a '1', and a +5 V (nominal) representing a '0', as far as outputs go. From the input standpoint, the SYM will accept the full range of RS-232 levels, from < -5 V to > +5 V.

Most of the modern RS-232 equipments will accept the inverted TTL signals; for those that do not, you will need to bring in a -Vn and change some jumpers.

It is very simple to convert the 20 mA loop to a non-inverted TTL port by shunting the output to the printer with 150 to 250 ohm resistor (use the existing R110, 150 ohm resistor, for this purpose, by "local" rewiring). If you wish to invert the TTL signal to make this port RS-232 compatible, cut the appropriate traces and insert inverters in series. For your information, there are three unused inverters on the SYM which you are free to use for this, or any other purpose. These are to be found in the 74LS04 at U9 (pins 3,4,5,6) and in the 7404 at U2 (pins 1,2). In addition, pins 3 and 4 of the open-collector hex inverter 7416 at U38 are also free. If you purchased the early version of RAE-1 (with the A suffix), you are already using the inverter in U2, to correct for a masking error in the ROM CS (chip select) pin polarity.

MUSIC, MUSIC, MUSIC

We now have the MTU Musical Instrument Synthesis Song Pack on our system, with over 40 selections. We have automated the system to re-boot itself after each selection (for some pieces, the instrument synthesis tables overlay, and destroy, the DOS) and ask the listener to enter the number of his desired selection. Our favorites include "Lara's Theme," "The Entertainer," "Maple Leaf Rag," and "Dueling Banjos," but there are some excellent Tchaikovsky, Bach, Handel, Chopin selections thrown in.

SYM-PHYSIS 8:36

STILL MORE MISCELLANEA

CARL MOSER, author of RAE-1, based on his previous ASSM/TED for the 6502, has long had available an ASSM/TED 6800, a cross-assembler from 6502 to 6800. He has been selling this to industrial users with full documentation and consulting support for \$500. Since RAE-1 users already have the essential documentation, and since we have agreed to provide any required support, Carl has allowed the Users' Group to market ASSM/TED 6800 object code on diskette or cassette for \$75. We hope that Carl will soon have an ASSM/TED 6809 available. Since most SYM-1/69 users will also have a SYM-1 available, the cross-assembler approach should be acceptable.

TERRY TRIFFET, Engineering Dean, University of Arizona, has come up with an interesting concept, that of publishing books on cassettes, readable on any computer system with Microsoft Basic. Terry has written a text editor in BASIC to permit organizing the text and dumping it to cassette a page at a time. A typical full-length book could be contained on a single 60-90 minute cassette. When tapes are duplicated automatically, publishing costs are below those for hard copy editions. A second BASIC program permits the user to control the read-in and formatting of successive pages. The user specifies the terminal width (from 40 to 66) and the display is right-justified, with long words hyphenated, if required. Terry's first book is a novel, "The One-fold Way". We have read the first three chapters and are anxious to read the rest. The formatter program and the complete novel are available directly from Terry, at 6935 Stardust Circle, Tucson AZ, 85718, for \$25. As a convenience to overseas subscribers who are ordering other items and would prefer to send only a single check, the Users' Group will handle overseas orders.

WILLI KUSCHE, Wilserv Industries, P. O. Box 456, Bellmawr, NJ 08031, sent us a copy of the User's Manual for KMMM (Kusche Magnetic Media Monitor), a 6502 system DOS (Disk Operating System). KMMM is written to be used with the SD Systems Versafloppy I Floppy Disk Controller, which itself is designed to be used with the S-100 Bus. It was John Blalock who called our attention to KMMM by sending us a preprint of an article submitted to MICROCOMPUTING, describing his method of interfacing the SYM to the Versafloppy Controller. John has high praise for K-M cubed, and sent along copies of his links between BAS-1/RAE-1 and KMMM. He also sent along a schematic of a simple 6502 to S-100 bus signal converter to drive the Versafloppy FDC. Since MICROCOMPUTING has a long lag time for articles, contact Willi for additional information, don't wait for the article. Our feelings about KMMM? It is a well conceived and well implemented DOS, and since it can read CP/M generated disks, it should be possible to transfer ASCII text files to/from 8080/Z-80 based systems, using 8" drives.

PROTRONICS, 1516 E. Tropicana, Suite 7A815, Las Vegas, NV 89109, sent us a copy of the Owner's Manual for a 32K Dynamic RAM Board for the KIM-4 Bus. The board also includes four ROM/EPROM sockets. The 32K can be assigned either above or below \$8000, and individual 4K blocks can be disabled in case of address space conflicts. The ROM sockets will hold the 2-chip versions of BAS-1 and RAE-1/2 or 4 EPROMS. Contact PROTRONICS directly for additional information.

JOE HOBART, 3465 N. Andes Dr., Flagstaff, AZ 86001, has sent us for review a copy of PIRATE'S ADVENTURE, adapted from the TRS-80 Level II BASIC version printed in BYTE, December 1980. As the author, Scott Adams, of Adventure International (AI), points out (and as we have confirmed by testing), BASIC is far too slow, and requires lots of memory. Mr. Adams states that all current AI games are written in assembly language, and gave Joe authorization to distribute his BASIC version royalty-free. The game requires a FULL 32K system, really! Send Joe \$15 for a cassette copy.

SYM-PHYSIS 8-39

LOG-OUT

Well, another issue will be ready to go to the printer's when these closing paragraphs are stored on disk (ENT LOG), the printer turned on (DC PDN), and the hard copy prepared for publication (DC PUB).

We have published three programs, all in assembly language, one each at the beginning, intermediate, and advanced levels. This does not mean that we did not set enough BASIC programs, or that we are anti-BASIC. We like BASIC; we regard it as the lingua franca of the computer world, and feel it should be everyone's at least second language.

Actually, we did receive many more BASIC programs than we could find time to try, and Tom Gettys has given us a whole diskful. We think we have convinced Tom to compile the listings into a book, which we could call "The First BASIC Book of SYM", and offer for sale in the next issue, and provide on cassette also. If you would be interested in spurring Tom on, write to him, in care of us. Our aim should be to build the SYM software availability up to where it will become the STRONGEST selling point for SYM.

In our next issue we will be publishing a neat BASIC utility program submitted by Jim Pensra, of Whitman College, Walla Walla, WA. This program searches a BASIC text file and picks out and lists all variable names. Unfortunately, it arrived too late for review and publication in this issue.

We hope also to find room for John Hissink's 9600 baud terminal patch program, which will practically seem to throw the data onto your CRT screen. Several readers, including Jean Paris, of Paris, France, who visited us last weekend, on a motorcycle trip through the western US and Canada, work at 9600 baud regularly, by simply having replaced the 1.0 MHz crystal with a 2.0 MHz one.

We regret to announce that the miniature ultra-violet lamp which forms the basis for our cheap EPROM Eraser has been discontinued. We are contacting all possible places we can think of which may have some still in stock. Thanks to Bill Cramer, Atlanta, GA, who found us a small batch, we still have a few Erasers available, but at a slightly increased price. See addendum for new price.

Our Microprocessor Fundamentals class at Chico State drew 26 students from as far away as Michigan, Utah, and Washington State. In addition a number of Chico State's foreign students from Algeria and Saudi Arabia were registered. Six of the students were previous "SYMmers". These included Phil Kohl, Dick Albers, Ken Kartchner, Larry Briggs, Joe Fierstein and Paul Walker. It was good to meet with these "old friends", and to be able to compare notes with them. There was quite a profitable interchange of ideas. In addition, they provided valuable guidance and inspiration to the raw beginners. We will be offering this course again at University of California, Davis (near Sacramento), on Sept 11-13. The fee is \$295, or \$495 with a SYM included. Call (916) 752-0880 for additional information.

We will be on a sabbatical leave from Chico State during the spring semester of 1982, and plan to spend several weeks each in Europe and the South Pacific area. If any of our readers associated with universities or technical schools would wish to have us as a "Visiting Professor" for a few days, in exchange for local travel and lodging expenses, please write. We look forward to meetings with many of our overseas readers during these trips.

We thank all who have submitted material, and apologize that so much had to be left out; it was not easy to decide, and there was more than enough for an 80-page issue.

SYM-PHYSIS 8-40