-Punch along this edge for insertion into a binder-

SYM-PHYSIS

THE SYM-1 USERS' GROUP NEWSLETTER

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

SYM-PHYSIS is a quarterly publication of the SYM-1 Users' Group, P. O. Box 315, Chico, CA 95927. SYM-PHYSIS and the SYM-1 Users' Group (SUG) are in no way associated with Synertek Systems Corporation (SSC), and SSC has no responsibility for the contents of SYM-PHYSIS. SYM is a registered trademark of SSC. SYM-PHYSIS, from the Greek, means the state of growing together, to make grow, to bring forth.

Editor/Publisher: H. R. "Lux" Luxenberg
Business/Circulation: Jean Luxenberg
Associate Editors: Thomas Gettys, Jack Brown
Jack Gieryic

SUBSCRIPTION RATES (1981):

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

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

EDITORIAL POLICY

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

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

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

HELP US TO HELP YOU

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

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

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

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

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

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

OUR SUPPORT POLICY

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

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

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

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

MORE ON SOFTWARE 'THEFT'

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

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

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

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

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

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

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

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

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

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

RAE now signals us audibly on error messages, and also lets us know when we are near the end of an input line (72 characters). We can now PRINT CHR$(7) from BASIC, instead of LET X = USR(&H8972",0) when we want a beep. Our only complaint is that the Bell Enable signal is too long, nearly two seconds. One fix for this is to install a 'flasher LED' in series with the beeper; this will produce several short beeps instead of the one long beep. The beeper volume may be reduced in intensity with either a series resistor or a piece of masking tape over the opening. To disable the oscillator cut the long trace just above R 23, and jumper the left end of this trace to the right end of the long trace just below U 38. This cut and jumper will disconnect pin 5 of U 38 from the on-board oscillator and permanently ground it.

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

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

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

CONTROLLING I/O FROM BASIC

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

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

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

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

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

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

ANOTHER CASSETTE PROBLEM AND FIX

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

THE RAE USer FUNCTION

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

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

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

WIGGLE YOUR CHIPS AND FLEX YOUR BOARDS

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

MORE ON DISK SYSTEMS AND COMMUNICATIONS

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

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

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

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

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

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

SYMPHYSIS 1979-80 INDEX AVAILABLE

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

SPEAKING SOFTLY (ON SOFTWARE)

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

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

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

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

>>>SUPERMON EXTENSIONS<<<

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

>>>A SYMBOLIC DISASSEMBLER<<<

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

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

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

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

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

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

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

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

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

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

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

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

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

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

(Continued on page 7:17)

A KANSAS CITY STANDARD TAPE DUMP

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

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

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

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

Dear Lux:

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

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

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

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

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

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

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

Yours sincerely,

John Newman

ASSEMBLE LIST

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

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



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

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

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

3031- 20 A0 8A

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

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

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



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

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

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

30FB- 20 D5 31

                    3910            DS (TREAD INVEC+1)

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

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

3121- 20 A0 8A

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

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

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

                4420            T1S

3170- 20 C5 31

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

3174- 20 C5 31

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

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

                4550            T1S

3182- 20 C5 31

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



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

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

                4830            DS (WORKBF+256 TEND)

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

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

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

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

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

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

(Continued from page 7:8)

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

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

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

A SIMPLE DATA SAVE AND DATA LOAD PROGRAM FOR BASIC

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

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

ASSEMBLE LIST(DSAV2)

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

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