SORCERER'S APPRENTICE™

PAGE 73
VOLUME 4
NUMBERS 4 & 5
June 1, & July 15, 1982

INTERNATIONAL COMPUTER USERS GROUP/NEWSLETTER

Price $3.00

IN THIS ISSUE -

BELLY UP! 73
ODDS & ENDS 74
PUBLIC DOMAIN - #2 75
SOUND EFFECTS 76
SB LETTER MACRO 76
NEWS FROM THE VALLEY 77
RANDOM EXPANSIONS 78
OFFICE SORCERER 80
SPREAD SHEET CONT'D 81
'SWORD' REVIEW 82
PRINT 'EM W I D E 83
CASSETTE INDICATOR LIGHT ...83
VENEZUELAN CONNECTION 84
EASY TYPING PRACTICE 89
ON-LINE 89
USEFUL POKE COMMANDS 89
ESF - ADDITIONAL INFO. 90
ESF - OPERATING TECHNIQUES..91
ROM PAC NOTEBOOK - #4 93
ROM PAC NOTEBOOK - #5 94
MAGIC SQUARES 95
REMOTE CP/M NOTES 95
T.O.S.C.A. 96
PSEUDO PRINT USING 96
THE MAKING OF 'DUEL' 97
TIPS 97
MX-80 SUPER INTERFACE 97
FINANCIAL TABULATOR 98
HARDWARE NOTES 103
GET ML ROUTINES FROM B-PAC.103
MERGE 103
RELOCATED W-PAC MODS 105
RANDOM I/O 106
4TH TIP 106

ESI BELLY UP!

by Don Gottwald

Finally, we have the correct information (we hope) regarding the status of Exidy Systems, Inc. (ESI). Mike Mazzola of Biotech is referring all calls regarding ESI to Gary Jensen of Challenge Computer Corp.

I spoke with Gary Jensen and he confirms that Exidy Systems, Inc. is **out of business.** Challenge Computers is **NOT** a replacement for ESI, nor will they support the Sorcerer in any way. For any questions you may have, he suggests you contact B.J. Freeman by mail at P.O. Box 70310, Sunnyvale, CA 94086, by phone at (408) 738-0185 or (408) 749-8871 or via the SOURCE at TCD284.

Challenge Computers has bought some of ESI's inventory. They are currently offering Model II's with 48K RAM for $700. They also have some single Floppy Disk Systems (FDS) for $600 and add-on units for $500. No S-100 Unit is required for these. A few Standard BASIC Pac's and Word Processor Pac's are available, as well as some software. If you're interested in any of these items, contact Gary Jensen at Challenge Computer Corp., 1225 Commerce Drive, Richardson, TX 75081; tel. (214) 669-1101.

It appears that ESI could not generate enough cashflow to support the products they were offering. According to Gary, about 5000 Sorcerers were sold in the United States, versus 20,000 in Europe. (Figures quoted by ESI's management were substantially higher. They said that 15,000 were sold in the U.S., versus 30,000 in Europe). Compudata, B.V. of Holland still has manufacturing rights to the Sorcerer and are currently producing them for the European market. (We will investigate the possibility of support from Compudata and report back to you).

Biotech owns all patents, copyrights and manufacturing licenses to the Sorcerer. The designer of the original Sorcerer is reportedly negotiating with Biotech for the manufacturing rights. We'll keep you posted.

(continued on page 89)

ODDS & ENDS

by Ralph LaFlamme, Editor

Greetings after a long absence from these pages. I want to thank Don Gottwald for pitching-in so well with the last issue. He got the whole thing out alone. I think he did a super job!

I must apologize to a number of you for my lack of personal contact in some time. That includes those of you who have tried to contact me via the SOURCE and MicroNET. If you have had file(s) for me to be picked up off these systems in the last few months, please contact me again via the account numbers shown on the back cover. Hopefully, I'll be better able to return your inquiries.

I have been going through a good deal of personal upheavals for the last few months. I have just moved from Troy, Michigan to the Lafayette, Louisiana area. This, coupled with problems in getting articles and ads in on time, and various equipment problems, created a good many delays. Our apologies for this and thank you for your patience. Hopefully, everything is now under control.

I have been accepted into the graduate program in Computer Science at the University of Southwestern Louisiana located in Lafayette, LA. Consequently, I have to give up my role as Editor of this Newsletter. We are in need of a new Editor. If anyone is interested in this job, please contact us by mail at our P.O. box, or by phone. Don Gottwald may be reached at (313) 286-9265 and I at (318) 856-4954. Despite all the headaches, I've enjoyed this role and have enjoyed talking and corresponding with you. I hope that you give the new Editor as much support as you have given me.

In order to try and get caught up, we are combining issues 4 and 5 in one. You are still getting an equal number of articles in this combined issue, you're just not getting a duplication of covers, ads, application form, etc.

A.S. Marland of Bois Colombes, France advises that Bob Roth's Z-80 Relocator program found in issue 3.4, had an additional bug. Address 504E reads: 0C (INC C), but should read: 03 (INC BC). This gets the length right when dealing with blocks that are a whole number of pages long.

He also adds that the new Monitor 1.1 retains a bug in the TEst program. OK still means there is a zero at the indicated bit location, while BAD still means there is a one with no indication of what the bits ought to have been. Put another way, the bad address is compared with 00 during print out, even though it was probably compared with something else when the fault was actually detected. POP BC (to recover B), PUSH bc (to put C back on the stack) need to be inserted between E963 and E964!

Bob Freeman advises that he has an Extended Basic Star Trek program containing 5 modules to run on 5 Sorcerers. The modules are not complete but are far enough along for someone to take over and complete. He has not had the time to do this himself but will be glad to provide any thoughts he has to any interested parties. He used the Book, **Starship Simulation**, by Dilithium Press as his reference source. You may contact him by phone on Weekends (408) 749-8871 or via the Source at TCD284.

Charles Boone, of Lokeren, Belgium, advises us of the following European sources for items available for the Sorcerer:

Light Pen

This pen plugs into the parallel port, comes with 5 programs to teach how to use it with BASIC and ML-programs and a demo program. This is available from: Gilbert Oegema, Florisdonk 10, 4707 Vm Roosendaal, The Netherlands.

Gebr. Van Montfort, Smedestraat 13, 6418 CR Heerlen, The Netherlands, has the following five items available:

I/O Pac

This is an EPROM Pac with 24 I/O lines, it has room for 2 EPROMs and is user programmable.

EPROM Programmer

This EPROM Programmer plugs into the I/O Pac above via a 25- pole D-connector. This includes all necessary software which can be put into the I/O Pac.

EPROM Pac

This can be used to plug in your own software.

ROM Pac Bus System

This buffered bus system allows you to software select and insert any one of eight possible Pacs without having to turn off your system.

EXPAN Monitor Expansion Program

This powerful debugger/disassembler takes the existing 13 Monitor commands and expands them with 23 new ones. This program improves the old ENTER, MOVE and SET commands and adds others such as DA (Dump Ascii), HE (Hex-calculation) and Breakpoint manipulation commands (SR, RB, DR, DS, DB, CO, ER, and EX).

Leo Gielen, Zeishof 18, 6418 JJ Heerlen, The Netherlands, has the following 3 items available:

EPROM Eraser

This inexpensive eraser takes about 45 minutes to do its thing on four EPROMS at once.

Monitor ROM Version 1.1B

This ROM fixes all known bugs while still maintaining the 1.0 jump-addresses. It has a switch available to allow switching back and forth between Versions 1.0 and 1.1B if you should so choose.

Game Interface / DA Converter

This system contains: 1) A joystick interface; 2) Music interface which is compatible with Arrington's standard; 3) An amplifier; . 4) An 8-bit DA converter based on a DA chip, not a cheap set of resistors.

ERRATA: In issues 3.8, it was erroneously stated that the Vista disk system uses soft-sectored disks. We are advised that it should have read 10 hard-sectored.

Wim Plaat at Triangle Systems advises that their Word Pac Plus program will not be released. They have had too many bugs to work out. Anyone with unresolved complaints about their order for this program, please contact Wim direct.

Anyone using CP/M 2.xx and Spellbinder have probably noted occasional problems with **lost** files. The file is 0K long when you go to load the file later. You know you wrote it to disk properly and got a message back as to how many characters were saved. What happened?

There appears to be a bug in the latest version of Spellbinder. It does its own IO calls rather than using CP/M's console input (CONIN), output (CONOUT) routines. Consequently, when you write a file to disk, it does not flush the buffer, to transmit the balance of the file and then write an end of file mark.

To prevent this loss, do a query (Q command) of the directory immediately after saving a file. This will flush the buffer and close the file properly. This has been working fine for me now. I've lost some big files by not realizing this! Lexisoft is aware of the problem and is rectifying it. We'll let you know about updates from Lexisoft as soon as we hear something.

We have several Members who are Ham operators. If any of you would like to be in contact with other Hams, please let us know (include your call sign). We'll publish a list of such members.

After trying several different cassette recorders, Steven Smouse, of Keene, Texas advises that he had the best luck with using the Sound Design model #7636 recorder using DAK Industries tapes. He found the recorder to be the most important factor in being able to LOad programs from tape with tape quality a close second.

Late last year we had requested information on your dealings with Exidy Systems, Inc. We were interested in approaching ESI with your problems to see if we couldn't help to improve the service. Since they have now gone out of business, there is no more point to this. I would like to thank all of you who took the time to respond to our request. Hopefully, just writing about it helped some of you to relieve yourselves of some pent up frustrations. There were more than a few of you!

In response to issue 3.6 article, "Galaxians Attack!", Lennart Mansson, of Smultronstigen, Sweden sends along the scores of some of his fellow students achieved while playing this game. Look at these!●

Karin Kallmark	179440
Inge Melin	140320
Anders Kullberg	51980

IN THE PUBLIC DOMAIN

by Bruce Blakeslee - CP/M & Micropolis Librarian
906 Crestwood Road - West Westfield, NJ 07090
(201) 233-3185 (Evenings)

This installment of my column will focus on the CP/M User's Group volumes. As I stated last issue, the CP/MUG has been around since 1977. To date, it has produced 59 volumes of public domain software. This is a treasure trove of useful programs and tutorial information on such topics as 8080 and Z80 assembly language programming, interpreter and compiler design and development, as well as game and utility programming.

The CP/MUG volumes are numbered 1-54 and 78-82. Volumes 55 to 77 were re-issues of early SIG/M User Group volumes. There was quite a row about CP/MUG re-releasing these volumes without the authorization of SIG/M. To my knowledge, this will not recur. I will not discuss volumes 55 to 77 as part of the CP/MUG as I will be covering them in a later column on the SIG/M volumes.

Because of space limitations, I will be giving a very general overview of each volume. For a more complete catalog of the CP/MUG and SIG/MUG volumes see the end of this column.

```
**********************************
*     Notations Used Below       *
*                                *
*  1 = A volume most will want   *
*  2 = Of interest from the      *
*      point of software         *
*      development.              *
*  3 = Of limited interest       *
**********************************
```

Volume 1 ..... Various CP/M Utility Programs (3)
Volume 2 ..... Lawrence Livermore Basic and Disk Tiny Basic. (2)
Volume 3 ..... Various Basic E Games (1)
Volume 4 ..... Actor, ML80 and Fortran-80 Code (2)
Volume 5 ..... Basic E Compiler and Programs (1)
Volume 6 ..... Utilities and Games (1)
Volume 7 ..... PILOT language (1)
Volume 8 ..... CP/M Utilities (2)
Volume 9 ..... General Ledger by Bud Shamburger (1)
Volume 10..... Lawrence Livermore Basic interfaced to CP/M (2)
Volume 11..... Disk Tinybasic and Disk Processor Technology BASIC/5. (2)
Volume 12..... Pilot Interpreters patched for CP/M. (1)
Volume 13..... BASIC-E/CBASIC and MBASIC Games (1)
Volume 14..... CP/M Utilities (2)
Volume 15..... Utilities and non-BASIC games (2)
Volume 16..... Assemblers, Utilities, and FOCAL (1)
Volume 17..... Utilities, Denver Tiny BASIC and NON-BASIC games. (2)
Volume 18..... Math Routines, Monitors and CASUAL - NO CP/M hook up. (2)
Volume 19..... Utilities - Hardware specific (2)
Volume 20..... BASIC-E and CBASIC Programs (1)
Volume 21..... Microsoft Basic Programs - Games (1)
Volume 22..... Monstrous Star Trek Games - too big for most systems but can be cut down. (1)
Volume 23..... STOIC Compiler (1)
Volume 24..... CP/M Utilities, Macro Libraries and RATFOR. (2)
Volume 25..... Utilities, CP/M STOIC (1)
Volume 26..... MBASIC and FORTRAN Games (1)
Volume 27..... MBASIC Games (1)
Volume 28..... Basic-E Utilities and Games Including a Data Base System and and ALGOL like language. (1)
Volume 29..... ASM Games and Utilities and CP/M BASIC-E V1.4 Floating Point Source. (2)
Volume 30..... CP/M BASIC-E V1.4 PLM Source (2)
Volume 31..... Tarbell Basic Manual and Source (2)
Volume 32..... Tarbell Basic Source (2)
Volume 33..... Search and Rescue Programs (3)
Volume 34..... SAM76 Language (1)
Volume 35..... FELIX - Graphics Animation Sys. (2)
Volume 36..... Assemblers, Editors, and Util.'s (2)
Volume 37..... CBASIC Programs (1)
Volume 38..... Utilities and FAST (2)
Volume 39..... Music Programs (1)
Volume 40..... Disk Cataloging System (1)
Volume 41..... Ham Radio, Chess Programs (2)
Volume 42..... Disassemblers, Utilities (1)
Volume 43..... Osborne CBASIC2 Accounts Payable and Receivable (BUGS) (1)
Volume 44..... Osborne General Ledger (BUGS) (1)
Volume 45..... Osborne Payroll w/cost accntg. (1)
Volume 46..... CP/M Utilities (1)
Volume 47..... CP/M Utilities (1)
Volume 48..... BDS-C Sampler Disk (1)
Volume 49..... Fortran and RATFOR Programs (2)
Volume 50..... Pascal Compiler, SPEED (1)
Volume 51..... Stage2 Macro Processor (1)
Volume 52..... Utilities (2)
Volume 53..... BDS-C Adventure (1)
Volume 54..... Xitan Disk Basic Programs (3)

*** Volume 55 - 77 are re-issues of SIG/M Disks

Volume 78..... Utilities (1)
Volume 79..... Modem Programs for PMMI, and Smart Modem (1)
VOLUME 80..... Cromemco Structured Basic Programs. (3)
Volume 81..... CP/M Utilities (1)
Volume 82..... North Star BIOS routines (2)

```
**********************************
* Disks May Be Ordered From Me   *
*     At The Address Above       *
*                                *
* $3.00/disk - you supply the    *
*              media FORMATTED.  *
*                                *
* $8.00/disk - I supply the      *
*              media.            *
*                                *
* Please include a minimum of    *
*         $1.50 Postage.         *
**********************************
```

I realize that the above descriptions are scanty at best. They are designed to give you a brief idea of available, low cost software. I can provide a disk with a full catalog. I will also provide the CP/MUG and SIG/M programs found on the above disks at the costs outlined above. There are now 59 CP/M disks and 69 SIG/M disks for a total of 128. That is a lot of software. Please realize that much of the software available must be converted to the Micropolis disk format because the sector skew is different from standard CP/M 8", single density disks. Also, there are often machine specific routines in much of the software which must be changed to work on the Sorcerer. The Monitor calls will be different, the video is found in a different area, and the screen size is non-standard just to mention a few of the problems. For me, however, this has been the interesting part. With the ASM files to work from, I have learned a great deal about my machine and about programming. Getting one of these programs to run is a real joy.

Don't get me wrong. Many of the programs contained in the CP/M disks will run without modification. Don't be put off, however, at having to rewrite something to make it run. It is a great way to learn and contribute. If you are like me, you will struggle at first just to make the programs run. At first, ignore the inelegance or the optimization of the program. After you have some experience, you will find yourself adding conditional IF - ENDIF program segments so those who have a Sorcerer can easily alter the program for themselves and still leave the original code untouched. Finally, you will begin to find ways to make the programs run more efficiently, correct bugs you and others discover, and make extentions to the programs to provide for more options. It's a fascinating task.

In the next issue, I will discuss the software available on the SIG/M disks. I will also announce the first volumes of the Sorcerer User Group. At this point, I have 4 volumes that I am editing. They will be made up of programs submitted by Sorcerer users and will include TIP files, BASIC Pac files, and other software designed to run on the Sorcerer. I mention this now because I would like each of you to begin thinking about submitting programs YOU have written to the SUG. Anyone who submits programs on disk will receive in return one disk of software of your choice **free.** The only thing I request, and I will be careful about this, is that you submit **only** public domain software which **you** have written. **Do not** submit software you have purchased.

If you have any questions I would love to talk to you. I can be reached most evenings at (201) 233-3185. Call me, I love to chat about software and I would love to hear what you are doing with your Sorcerer.

SOUND EFFECTS

by Thomas J. Sweeney

This is for all of those who own a Sorcerer's Voice and have any kind of space game program. The following routines were originally designed with Star Trek in mind but can be used in any of the space game programs which have been devised over the years.

Phasers

```
FOR I=1 TO 30
OUT 255,47
FOR J=1 TO 15:NEXT J
OUT 255,45
NEXT I
OUT 255,255
RETURN
```

Red Alert

```
FOR JJ=1 TO 5
FOR I=25 TO 45
OUT 255,I:OUT 255,I+1
FOR J=1 TO 10:NEXT J
NEXT I
OUT 255,255
FOR J=1 TO 30:NEXT J
NEXT JJ
RETURN
```

Klingon Return Fire

```
REM K3= # OF KLINGONS
FOR JJ=1 TO (K3*2)
FOR I=50 TO 25 STEP -1
OUT 255,I
NEXT I
OUT 255,255
REM RANDOM DELAY BETWEEN SHOTS
FOR J=1 TO (INT(RND(1)*500+5))
NEXT J
NEXT JJ
RETURN
```

Photon Torpedo

```
FOR J=1 TO 40
OUT 255,50
NEXT J
FOR I=50 TO 25 STEP -1
OUT 255,I:OUT 255,I+1
NEXT I
OUT 255,255
RETURN
```


COMPLETE LETTER MACRO FOR SPELLBINDER VERSION 5.10

by Daniel Edward Behmer

As a microcomputer consultant who does work in many parts of the world, I find that my SPELLBINDER Word Processor is one of my most frequently used pieces of computer software. It is not unusual for me to send out ten letters a day, each quite individual in character. Therefore, it was absolutely necessary that I write a Macro in SPELLBINDER's MSPEAK language to take care of most of the tedious chores.

Included at the end of the Macro you will find complete instructions. The use of this Macro has increased my own letter writing production by 100% and I hope that you find it useful as well.

LISTING

```
; A COMPLETE LETTER MACRO FOR SPELLBINDER Version 5.10
:SS 28
:%C = "1"
:%D = "2"
:%4 = 2
:%5 = $3
f
:%6 = $3
:%1 = %6 - %5 + 1
t
:%3 = %1 + %1
:pr "#1/#13/"
:pr "LETTER WRITING SYSTEM MACRO - by Daniel Edward Behmer"
:pr "#13/#13/"
:pr "TYPE (1) FOR FULL SYSTEM:    ********************** #13/"
:pr "                             *                    * #13/"
:pr "All space & Letter Options   * Note: letter must  * #13/"
:pr "File Card                    * have () after the  * #13/"
:pr "Envelope                     * address.           * #13/"
:pr "Letter                       *                    * #13/"
:pr "File Copy                    * FULL INSTRUCTIONS  * #13/"
:pr "                             * AT END OF THE MACRO.* #13/"
:pr "TYPE (2) FOR SHORT-CUT TO:   *--------------------* #13/"
:pr "                             * Options set at     * #13/"
:pr "Envelope                     * initiation:        * #13/"
:pr "Letter                       *                    * #13/"
:pr "File Copy                    * single space letter * #13/"
:pr "                             * space not expanded * #13/"
:pr "TYPE (3) FOR SHORT-CUT TO:   * your name only on  * #13/"
:pr "                             * return address of  * #13/"
:pr "Letter                       * pre-printed envelope* #13/"
:pr "File Copy                    *                    * #13/"
:pr "                             ********************** #13/"
:in "          YOUR CHOICE? " %E
:%F = "2"
:on %E%F / 54 72
:pr "#1/#13/"
:in "Type (1) for single space; (2) for double space " %C
:pr "#13/"
:in "Type (1) for expand space; (2) for regular space " %D
:pr "#13/"
:pr "Type (1) for your full name and address on envelope #13/"
:pr "Type (2) for your first name only on pre-printed #13/"
:in "          envelope --- " %4
:pr "#13/"
:in "Type (1) to file card; (2) to envelope; (3) to letter " %A
:%B = "2"
:on %A%B / 54 72
:pr "#13/"
:in "Insert the FILE CARD, then (CR)" %A
y 1 90 110 2 0 1 1 65 2 1 0 0 25 10
p%1
b%1
:pr "#13/"
:in "Insert the ENVELOPE in the Printer, then (CR) " %A
y 1 90 110 0 0 1 1 65 2 1 0 0 25 10
yt 0 2 0 0 12 12 7 3 3 0
pl/Dr. Daniel Edward Behmer    ;Insert your name here
:on %4-2 / 64 /
pl/Suite 611                   ;On the opposite PL lines
pl/5440 Cass Avenue            ;insert your return address
pl/Detroit, MI  48202          ;as in this example
:on -1  65
3 pl
pl
pl
pl
yt 0 2 0 0 12 12 6 25 25 0
y 1 90 110 0 0 2 1 65 2 1 0 0 25 10
p%3
b%1
y 1 90 110 0 0 %C 1 65 %D 1 0 0 25 10
yt 0 2 0 0 12 12 7 10 10 0
:pr "#13/"
:in "Insert PAPER for letter, then (CR) " %A
y 1 90 110 2 0 %C 1 65 %D 1 0 0 25 10
pa
ff
```

```
t
:pr "#13/"
:in "To print a FILE COPY, (CR); otherwise (ESC) to END" %A
pa
ff
t
;
;
;         *** INSTRUCTIONS ***
;
;   This Macro was written for Version 5.1Ø of
;SPELLBINDER. If you are using an older version, you
;must change the (y) and the (yt) reformat lines since
;the older version has different reformat sequences.
;   The purpose of this Macro is to increase the pro-
;duction of correspondence in an office by automating
;several steps which are part of an effective system:
;the typing of the name and address on a file card if
;desired; the typing of the envelope with option for
;full return address or one line for pre-printed envelopes;
;the typing of the letter with whatever spacing options
;have been chosen; and finally typing an additional copy
;for the files if desired.
;
;SPECIAL DIRECTIONS:
;
;   You should insert your own name and address in place
;of mine in the Macro. If you have only a three line
;address, replace my fourth line with a "PL".
;CAUTION: the lower case PL (pl) on my printer makes no
;distinction between lower case L and the numeral (1).
;The command on those lines is in fact PL in the lower
;case.
;   DO NOT type empty carriage returns for the OPTIONS.
;You must type in the number of the option you desire.
;However, when a prompting line shows a (CR), in that
;case only an empty carriage is given to indicate that
;you are ready for the Macro to continue.
;   I use a NEC Spinwriter 5510 as a printer. On the left
;side of the platen is an impression lever. I move that
;back one notch. This allows me to safely insert either
;a file card or envelope ON TOP of my tractor fed paper.
;Naturally, just before doing that I flip the paper tension
;lever on the right to ON in order to hold the card or
;envelope. Since I have not actually disengaged my paper,
;it is a simple matter to right things again and continue
;on with the printing of the letter.
;   This MACRO automatically counts the number of lines in
;the address. Therefore, it MUST have a mark (^) on the
;line following the address. Here is a sample letter format:
;
;   Mr. John Brown
;   123 Main Street
;   Cleveland, OH  46121
;   ^<  (Note the mark and the CR which creates a space!)
;   February 15, 1982
;
;   Dear Mr. Brown:
;
;        Thank you for your letter of last week. I am
;   planning to visit your office on Saturday.
;
;                          Sincerely yours,
;
;                          Your Name
;
;    After you type the Macro into SPELLBINDER using the
;    EDIT mode, you save it on your 'A' drive with an
;    appropriate name such as 'A:LET.WPM' . Please note
;    that the extension is required. When you want to use
;    the Macro at the beginning of a writing session, use
;    the COMMAND 'AD' which will ask you the file name. At
;    that point the Macro automatically initializes. For
;    subsequent use once it is in the buffer, simply use
;    the COMMAND 'A' to initialize the Macro.
;
; If you have any questions, please feel free to write
;to me by letter. My address is in the Macro.  My Source
;address is TCT534.
;                          Daniel Edward Behmer●
```

FROM THE VALLEY

by Bob Freeman

I have just spent a wonderful day going to the different little hole-in-the-wall stores here in the Santa Clara Valley. Most of them had used equipment of more or less the same quantity and quality.

A few stood out. They had parts or equipment that would delight the Sorcerer user who has a low budget, plenty of time and some technical background.

The first is South Valley Electronics, Santa Clara California. Outside of Challenge Computers (the company into which ESI was absorbed), South valley has the largest supply of logic boards, cases, and printed circuit boards (PCB) in the USA. They have unpopulated S-100 interface cards and the S-100 configuration PROM. These cards are not the S-100 box PCB, but looks just like an S-100 card. The Sorcerer connects through the 50 pin buss expansion to the top of this card. The card can occupy any slot in the S-100 mother board. All the interfacing I do to the S-100 world is with this card. A 12 slot S-100 mother board and power supply can be scraped together for under $125. The total cost would be under $225.

The second place of importance is Anchor Electronics, Santa Clara California. They are the sole supplier to the end user of California Computing S-100 bare boards. The bare board runs for under $50 and boards as Kits are under $200.

I am particularly interested in the soft-sectored disk controller (CCS 2422B) which lets me use use my Exidy software. It allows anyone who now has a Micropolis Quad density hard sectored system to convert to the soft sectored and use the same drives they have now. It does require two drive boxes and a program that allows auto transfer sector by sector. I can provide that service at a $5 copy fee + disk cost (unless disks are provided). this controller's many options make it great! I can run 8" and 5 1/4" disks at the same time. Also, if desired, it will auto boot, can be polled or interrupt driven, etc., etc. In the future, if I go completely S-100, I can run most of my Sorcerer CP/M programs, with little modification.

I had to change the special 74LS-237 program PROMs and the Monitor PROM to allow it to work with the Sorcerer. I will be glad to provide the program listing for the 74LS237's free or programed PROMs for both the 74LS237's and 2716 for under $80.

The best part of this controller is the cost. The Bare board comes with excellent documentation and costs $49.95 plus tax. The Kit comes with the board and all parts and sockets for $184. It took me about 4 hours to solder and test the board. The total additional cost would be under $300, the cost of one drive.

Anchor Electronics also has Bare Boards for 32K static RAM (uses 2114's) priced at $44.95, 64K dynamic RAM kits for $174 (includes 200ns 4116's), and IO Kits with both serial and parallel ports from $165 to $185.●

RANDOM EXPANSIONS

by Bob J. Freeman

The following was precipitated by the last Random I/O column.

System crashes can and usually are caused by 74LS241 data read chips losing their pull up ability. (see ramexpan.usr)

Word Processor Pac failures in most cases are due to the EPROMS becoming UNprogramed.

The NEC 1500 and 2500 series printers will print bidirectionally by throwing switches. A user program is necessary to drive the bidirectional feature of the printer. The Exidy PPRINT program will drive the Word Processor Pac. The disadvantage is that the switch needs to be thrown back to use the Sorcerer's internal printer driver for parallel printers.

Note the NEC printers tend to hang up when first turned on or when the Sorcerer is first powered up. This can be overcome by adding a 2.2k resistor from IC8H-20 to IC8H-3 (Sorcerer I) or IC9H-20 to IC9H-3 (Sorcerer II).

The EXIDY disk controller cannot read/write hard sectored disks. It is designed around the Western Digital controller chip 1793. The 01 series of the WDC chip has programing problems. If your controller hangs up start by replacing the 1793 with the B02 series chip.

The Sorcerer CAN be run at 4 mhz. It requires the following timing modifications:

1) RAM cycle time (may require replacing RAMs)
2) RAM RAS to CAS hold time.
3) Double the CPU to 4.213MHZ
4) Use Z80A chip.
5) Replace various chips that may not meet their specs.
6) Replace the BRUCE PROM if a MMI chip.
7) If using disk rewrite the disk IO to allow double the wait time for head home command.

The Hacker's Manual is a compilation of Tech notes, engineering notes, and design changes I have made to my machines and I believe will work with others. The Manual includes the switching power supply for the Sorcerer II, 56K dynamic RAM, using the new 64K chips, Real time clock, and programable interrupt controller, and many more things. The manual is $15 with updates at least once a year for one year.

I have a Sorcerer II working at 4 mhz with soft and hard sectored controllers. I don't suggest this change be done by anyone who cannot manufacture the complete timing diagram for the Sorcerer. I don't guarantee that everyone can get their Sorcerer to work without a lot of effort. (See Hacker's Manual for hardware changes).

Keyboard bounce is caused by not enough tension on the leaf spring in the key. This can be increased by bending the leaves towards each other with care not to crease the spring. (See Hacker's manual for suggested method).

The 56K RAM change should only be done by a competent technician. It requires a lot of cutting and jumpering and has only been tested on the Sorcerer II. If there is anyone willing to offer their Sorcerer I machine for a 4-6 month period, I will do the modification and supply the parts free. (Labor is $120 and parts are about $90 as of June 82).

The switching power supply is capable of 5 amps at 5 volts with an operating temperature of 37 degrees Celsius, warm to the touch. There are no HOT components. The power supply can be purchased 3 ways: 1) Bare board with schematic (no technical support) - $35. 2) Board (not an old power supply board), parts, schematics and technical support (your phone calls) - $100. 3) Completed board with warranty - $195.

Is there a way of using more than 56K (57344 bytes decimal) of RAM with the Sorcerer? If so, how may this be accomplished?

The Sorcerer has two separate RAM blocks: 1) program, 2) display. The program portion can be a maximum of 32K (33088 bytes decimal) for the Sorcerer I and 48K (49152 bytes decimal) for the Sorcerer II of RAM addressing. Both Sorcerer I and II have 3K (3072 bytes decimal) of RAM addressing for the display portion of RAM.

For those of you without the Sorcerer Software manual, the program RAM is free for any non-ROM Pac program from address 0H (0 decimal) to within 144 bytes of the top of RAM. The last approximately 80H (144-decimal) bytes are used by the Monitor. ROM Pac programs use the lower memory 0H to 1FFH (0 to 512-decimal) for their work area also.

The disk operating program uses about 8k (8129-decimal) bytes of program memory at the top. Exidy CP/M (r) and Micropolis MDOS (r) also use the Program memory from 0H to 100H (0 to 256-decimal). MDOS Operates from 0H to about 3200H (0 to 36800-decimal). These area not available to the user.

So how does one get more memory? Well !!!! For the Serious User there are two alternatives. This first is using magnetic storage the other is to increase RAM. Both require modifying present software. And you thought there was a simple answer...fooled you!

Testing dynamic RAM can be long. Here are some hints to help you gain confidence.

On Power up do a "DU" of program RAM. For example:

```
>DU 0 FF <cr>

Addr  0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
0000: 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF
0010: 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF
0020: 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF
0030: 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF
0040: 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF
0050: 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF
0060: 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF
0070: 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF
0080: 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF
0090: 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF
00A0: 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF
00B0: 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF
00C0: 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF
00D0: 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF
00E0: 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF
00F0: 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF
```

This is the dump you would see if you had Mostek 4116 RAM chips. The key is that there is a pattern to all Dynamic RAM. Even though you may not see this exact pattern, there should be a pattern. For example, a pattern like:

```
0000: 00 7F 00 0F 00 BE 02 FF 80 F8 0B 7C 00 7F 00 7F
```

could indicate:

a) A bad 74LS241 data buffer. (IC 13C or IC 4D)
b) Low power supply voltages.
c) Possibly bad RAM chip.

Testing of the access speed, which is more realistic than access time can be done by using the TE test. The TE test uses many calls which require stack PUSHes and POPs. The PUSH and POP are the fastest instruction in the Z80 set. Start by setting the stack in the 16K RAM block to be tested for speed, then test a 256 byte block of RAM in the same 16K boundary.

EXAMPLE:

```
EN 0
>0000: 21 LL HH C3 06 E0/<cr>
>GO 0
```

The LL stands for lower 8 bits of address and the HH stands for the Higher 8 bits of address.

If you have any questions, send them to me via SOURCE at TCD284 or by mail at P.O. Box 70310, Sunnyvale, CA 94086.

THE OFFICE SORCERER

by Roger Hagan

Application Note:

Transferring Program Output From BASIC Programs to Word Processor

Since I am particularly interested in relating the Sorcerer's Word Processor Pac to business uses, I sell a disk full of utilities and techniques called "The Word Processor Augmented", which is an constant evolution. (Owners can exchange old editions for new ones free.) A recent addition to that disk is the following technique for capturing thr output of a program in ExBasic as a screen, and placing that screen into the Word Processor for incorporation into a text, as well as further dressing, labels, etc. The system requires disk capabilities as shown here. It is useful when a Basic program draws a complicated screen full of data, such as my Spreadsheet program does, or a graph, as its Viz-a-plot module does. The technique should be applicable to any program, even ones not in Basic, so long as a method of triggering the disk save without affecting the screen can be found, and the screen image is made of Ascii symbols which the Word Processor will accept. Spellbinder should work as well as the Word Processor Pac as the receiving system.

The Basic-to-WP Transfer

The Exidy Word Processor Pac is of great value as a screen editor, and is the easiest way to dress up program output to be printed. This note addresses one method to interface disk Basic program output with the Word Processor Pac. It would apply as well to a disk-based word processor like Spellbinder. It will also show how to enter a forbidden autocommand to the Word Processor in order to replace non-Ascii characters. (See The Graphic Fix below.)

One advantage of a memory mapped video system is that the screen contents at any moment may be treated as data and saved. This can then be loaded as a Word Processor file to integrate with other text. This is the approach I take with SPREADSHEET in Exbasic. There is now a "Z" command to save the screen. The trick is to give the command to save or move the screen data without disturbing the screen, and to postpone naming the file until after the save for the same reason.

In general, two approaches to the transfer of screen data could be taken. One is to work with the Word Processor Pac in place and move the screen to its work area, then load the WP disk driver program which loads an initialized work area to RAM without closing off the memory file. This approach smashes the first 16 pages of the previous program or language which produced the screen image, so a repair version of the program (or language) must be available on disk to reload these 16 bottom pages of the TPA and execute a warm start. In Exbasic a warm start is Go 100 anyway, so no changes would have to be made before saving 16 pages of Exbasic as, say, EXBASA.COM. This method is illustrated at the end. (See Direct Use).

Another approach would be to move the screen RAM into the TPA and save it under CP/M. This means the Pac need not be installed while working with the Basic program. This method is illustrated first.

It is not necessary to exit Basic to save the screen. Since we may save strings of up to 255 characters with Basic disk I/O routines, we need only fool Basic into thinking that the screen area of RAM is 8 consecutive strings, 7 of 256 and 1 of 128 characters. The following code, used in the latest version of my SPREADSHEET program, achieves this. The line numbers relate only to that program.

A. The main program initialization must include this:

```
912 EF=&H1A:FOR I=0TO7:SC$(I)="":NEXT
```

This takes care of an end-of-file mark on the tape and declaring the big strings so that VARPTR can find them.

B. At the time of the call, the screen address values must be poked into the variable pointers for size and location of the SC$ strings, and then the strings sent to disk. A temporary name is given the file automatically and changed after the file is saved, so as not to disturb the screen, since the strings are sent to the disk buffer directly from the screen RAM.

```
700 FOR I=0 TO 6:V=VARPTR(SC$(I)):POKE V,&HFF:POKE V+1,&H80:
    POKE V+2,&HF0+I:NEXT:POKE V+3,&H7F:POKE V+4,&H80:POKE V+5,&HF7: REM
    NOW SC$(0-7) POINTS TO SCREEN RAM IN 8 STRINGS
720 OPEN "O",#1,"SCREEN.WPF"
730 FOR I=0 TO 7:PRINT #1,SC$(I);:NEXT:PRINT #1,EF
740 CLOSE #1
750 PRINT:INPUT "Name for this file";NA$
760 NA$=NA$+".WPF"
770 NAME "SCREEN.WPF" AS NA$
780 PRINT:INPUT "Now";IN$:GOTO 114:REM To the menu choice processor
```

C. At the end of the screen display, the program is altered to provide for one-key input to make a command choice. (This is not essential if the cursor ends up somewhere other than at the bottom of the screen where a scroll could lose a line from screen RAM, since it will be easy to remove the command from the screen image once it is in Word Processor.)

```
30790 CURSOR 50,27:PRINT "Menu":IN$=INPUT$(I):GOTO 114: REM  114 is
      the menu choice processor.
```

A SAMPLE SCREEN WITH GRAPHIC FIX

```
        NOV DEC JAN FEB MAR APR MAY JUN JUL AUG SEP OCT
4000    ..................... 4___4___4___4___4___4___4.....
3839                          /
3678                    4___4_/
3517                     /
3195    ............./.....................................
3034              4/
2873               /
2712               |
2551               |
2390    ........  |.....................................
2229               |
2068               |
1907               |
1746               |
1585    ........  |.....................................
1424               |
1263               |
1102               |
941                |
780     ........  |.....................................
619                |
458                |
297                |
136           _9___9___9___9___9___9___9___9___9___9___9
-25        ..9_/_4/.....................................
         Menu      Two expense factors: 4=rent   9=salaries.
```

(Clearly it would be easy to get rid of the unwanted "Menu" prompt above, and to add labels and a frame if desired.)

D. THE GRAPHIC FIX

In the above example, the SPREADSHEET graphing module uses a Sorcerer graphic character for the vertical line to line up well with the slashes. This must be replaced by a vertical line from the standard Ascii set to be handled by a printer and by the Word Processor. This has been done above.

When this chart first comes on screen, therefore, an unacceptable reverse video square-0 graphic is seen in place of the vertical lines. This must be removed with a search-and-replace. But the graphic character cannot be entered on a command line -- no reverse video character can be, including underlined letters! So the command line must be entered from the Monitor, and the command initiated by a Monitor GO command, as shown next.

In This Ascii line, "100S/#//(cr)", # represents the graphic character to be replaced, whose code is 80H. (The code for underlined space is 100H, 80 plus 20, the space with high bit set.) Enter the line at 600H, WP's command line buffer.

```
>EN 600

600: 31 30 30 53 2F 80 2F 7C 2F 0D/

>GO CEE5
```

The file will return to the screen with the character replaced.

Direct Use

The lines of Basic above were brought in from the program by loading the program in Exbasic with WP Pac in place, listing the desired lines, going BYE into the Monitor and commanding:

```
>MO F080 F7FF 80F

>EN F8F

F8F: 03/

>GO 0

A>DISKDRIV
```

This brought the lines to the editing screen. They needed some cleaning up. Very often, removing a leading space at the top drops many lines into alignment. When they were clean, the rest of the program was loaded to join them in the work space and things were bloc-moved into place and resaved.

This concludes the .DOC file on the Basic-to-WP Transfer. Next issue I will show how to merge the screens to create long-line printouts of all the columns in a broad spreadsheet.

In selling word processing systems to writers, I find them especially interested in a method of counting the number of words in their manuscript. Some are paid by the word, or write to a specified length in words. It happens that the easiest utility I've seen to do it is literally given away as an extra when you purchase Staley's "SPELL" spelling checker program. It is a separate program called "FOG", the purpose of which is to show both word count and what proportion is "big words". With a text in RAM, one exits to CP/M and types FOG, and in a second or two is given his word count and related information.

The Spell program itself will be compared to SPELLCHECK (Lexisoft's SPELLGUARD) in my next column, but I can point to one respect in which it is unique - it can check a Word Processor text in RAM as well as one on disk, so short letters need not be saved first to be checked. However, it does require a disk system, since the dictionary files are called in from disk with a series of disk accesses. I like these two programs enough that I am planning to market a version of the "**WP Augmented**" disk that includes them. As you will see next issue, there are utilities in **Spellcheck** not offered in **Spell**, but the cost is a lot higher too.●

==========

THE HAGAN SPREADSHEET Copyright c 1981 by Roger Hagan Associates, 1019 Belmont Pl. E., Seattle, WA 98102 USA

(continued from page 67 of issue 4.3)

==========

```
***  ------------"L" Move left one (or 1/2) 4 col sector -----
***
10500 IF FG=1 THEN 11700
10503 SC=SC-4:IF SC<1 THEN SC=1
10510 GOTO 10275
***
*** --------------"D" Move down a 20 or 10 row sector ---------
***
11000 IF FG=1 THEN 11800
11003 SR=SR+SZ-1
11010 IF SR+SZ-1>ROWS THEN SR=ROWS-SZ+1
11015 R=SR:CH=2+J-1
11020 GOSUB 10000
11030 GOTO 10215
***
***  ---------------"U" Move up a 20 or 10 row sector ---------
***
11500 IF FG=1 THEN 11900
11503 SR=SR-SZ
11510 IF SR<1 THEN SR=1
11520 GOTO 11015
11600 SC=SC+2:IF SC+3>COLS THEN SC=COLS-3
11610 GOTO 10275
***
***  ---------------- These take over if "X" expand mode set ---
***
11700 SC=SC-2:IF SC<1 THEN SC=1
11710 GOTO 10275
11800 SR=SR+INT(SZ/2):GOTO 11010
11900 SR=SR-INT(SZ/2):IF SR<1 THEN SR=1
11910 GOTO 11015
***
***  ---------------"T" Tape routine called for ---------------
***
12000 OUT 1,28
12001 FOR I=1 TO 15:PRINT CHR$(10):NEXT
12003 OUT 1,16
12005 PRINT:INPUT "Save or Load";Q$
12007 IF LEFT$(Q$,1)="L" THEN 12500
***
***  ---------------- Tape save of sheet -----------------------
***
***  --- We will set up a Monitor save of the entire array area
***      Basic has created for the numeric arrays.
***
12008 PRINT:PRINT:INPUT"A five-letter name for this sheet";SN$
12010 PRINT:PRINT"To save the sheet data on tape, start the ";
12013 PRINT "recorder and press"
12020 INPUT "RETURN, or 'H' to return to menu instead.";IN$
12030 IF IN$="H" THEN 120
12040 PRINT:PRINT"SAVING DATA....":
12050 VA(1)=ROWS:VA(2)=COLS:VA(3)=EX:VA(4)=SEC:VA(5)=REL
12055 DATA 83,65,32: REM       "SA "
12057 DATA 76,79,32,49,32: REM "LO 1 "
12060 FOR I=1 TO REL:FORN=1TO5:IF FUNC$(I,N)=""THEN12070
12065 FUNC(I,N)=ASC(FUNC$(I,N)):REM  Convert function symbols
12070 NEXT N: REM  to a numeric array
12075 NEXT I
12080 RESTORE 12055
12090 FORI=0TO2:READ A:POKE(BU+I),A:NEXT:REM"SA "IN CMD LINE
12100 FOR I=3TO7:POKE BU+I,ASC(MID$(SN$,I-2,1)):NEXT
12110 POKE BU+8,32:POKEBU+13,32:POKE BU+18,13
***
12120 REM Now SA NAMEX & 3 spcs are in cmd line; now addresses
***
12130 POKE260,108:POKE261,00:DU=USR(0):REM "ASCHEX PUT IN"
***
***  ---The subroutine at 108 (6CHex) makes use of a Monitor
***     subroutine at E1E8H to convert the addresses for start
***     and end of array area into the four bytes Ascii form as
***     needed by the Monitor command line.  Then the Monitor's
***     SAve command processor is called in the line below.
***
12150 POKE260,56:POKE261,230:DU=USR(0):REM "SA" COMD PRCSR
***
***  ---------------- Call StringSaveOn after spacing tape -----
***
***  ---Note: The delays are to allow ample time for Basic's
***     string space cleanup operations during loading, in case
***     string space is nearly full.  They may not be adequate ***
for the worst possible case.
***
```

(continued on page 82)

(SPREADSHEET continued from page 81)

```
12234 FOR U=1 TO 2000:NEXT
12235 PRINT:PRINT"Column and row names..."
12240 POKE 260,186:POKE 261,0:DU=USR(0):REM CMOTON
12244 POKE 260,65:POKE 261,0:DU=USR(0):REM STRSVON
12247 FOR U=1 TO 500:NEXT
12248 FOR U=1 TO 100:NEXT
12250 PRINT"":FOR I=1 TO COLS:IF I=1 THEN FOR U=1 TO 200:NEXT U
12255 FOR U=1 TO 200:NEXT
12260 PRINT CN$(I)
12275 NEXT I
12277 FOR U=1 TO 400:NEXT
12280 FOR I=1 TO ROWS:PRINT RN$(I)
12290 FOR U=1 TO 200:NEXT U
12310 NEXT I
12312 FOR U=1 TO 200:NEXT
12318 FOR I=1 TO REL:FOR N=1 TO 5:PRINT CO$(I,N)
12319 FOR U=1 TO 50:NEXT U
12320 NEXT N
12325 FOR U=1 TO 50:NEXT
12335 NEXT I
12340 POKE 260,39:POKE 261,224:DU=USR(0):REM CMOTOFF
12345 POKE 260,75:POKE 261,00:DU=USR(0): REM STRSVOFF
12350 PRINT:PRINT"Tape save is finished.  Menu is listening."
12360 INPUT IN$
12370 GOTO 114
***
***  ---------- Tape load of sheet ----------
***
12500 PRINT:PRINT"To load data from tape. press RETURN and ";
12503 PRINT "start tape, or press"
12510 PRINT"H to return to the menu.":INPUT IN$
12520 IF IN$="H" THEN 120
12525 IN$=""
12530 PRINT:PRINT"Loading data arrays..."
12540 RESTORE 12057: REM  Aim data pointer at "LO 1 "
12550 FORI=0TO4:READA:POKE(BU+I),A:NEXT
12555 POKE BU+9,13
12560 POKE260,151:POKE261,00:DU=USR(0):REM PUT LO ADDR IN PLACE
12580 POKE260,138:POKE261,231:DU=USR(0):REM CALL LO CMD PRCSR
12655 PRINT:PRINT"Placing variables, translating functions..."
***
***  --------Get the sheet parameters from the VAriables array
***
12656 ROWS=VA(1):COLS=VA(2):EX=VA(3):SEC=VA(4):REL=VA(5)
12657 FORI=1TOREL:FORN=1TO5:FUNC$(I,N)=CHR$(FUNC(I,N)):NEXT N
12658 NEXT I
12659 PRINT:PRINT"Loading column and row names..."
12660 POKE 260,186:POKE 261,0:DU=USR(0):REM CMOTON
*** Above in both routines to hold baud at 1200 for Mon 1.0
12662 POKE 260,85:POKE 261,00:DU=USR(0): REM STRLDON
12665 INPUT DM$
12680 FOR I=1 TO COLS
12690 INPUT CN$(I)
12695 PRINT"Col" I "of" COLS
12705 PRINT CN$(I)
12710 NEXT
12750 FOR I=1 TO ROWS
12755 INPUT RN$(I)
12760 PRINT"Row" I "of" ROWS
12764 PRINT RN$(I)
12773 NEXT
12779 FOR I=1 TO REL:FOR N=1 TO 5:INPUT CO$(I,N)
12780 PRINT"Constant name for relation"I"stage"N"of"REL"relations"
12782 PRINT CO$(I,N)
12783 NEXT N:NEXT I
12784 POKE 260,39:POKE 261,224:DU=USR(0):REM CMOTOFF
12785 POKE 260,95:POKE 261,00:DU=USR(0): REM STRLDOFF
***
*** ----- Trim leading linefeed off the strings ----------
***
12788 FORI=1TOCOLS:CN$(I)=RIGHT$(CN$(I),LEN(CN$(I))-1):NEXT
12791 FORI=1TOROWS:RN$(I)=RIGHT$(RN$(I),LEN(RN$(I))-1):NEXT
12793 FORI=1TOREL:FORN=1TO5:IFLEN(CO$(I,N))=0THEN12795
12794 CO$(I,N)=RIGHT$(CO$(I,N),LEN(CO$(I,N))-1)
12795 NEXT N
12797 NEXT I
12800 PRINT:INPUT"Loading completed.  Type 'M' to proceed.";IN$
12810 GOTO 114
***
***  ----------"P" Turn Printer on or off ----------
***
***  --- The poke addresses below are specific to a 48K
***      Sorcerer: -16434,-16432,-16431.  For 32K size they
***      must be, in the order below, 32718, 32720, 32721.
***      (7FCE, 7FD0, 7FD1)
***
12950 IF P=0 THEN P=1:GOTO 13000
```

(continued on page 83)

SWORD - Reviewed

by Richard Stone

I would like to tell you about a very interesting cassette- based Word Processor program for the Sorcerer.

I recently purchased an Epson MX80 printer for use with my computer. After hooking it up and writing some drivers for it, of course my attention shifted to word processing. Since for me it had been a disk system OR a printer, I couldn't use any of the wonderful sounding disk-based programs like Spellbinder. So, after some looking around, I found I had these options: write a system for myself; get SWORD from NorthAmerican Software (about. $30); get VISI-WORD from Quality Software (approx. $60); or get the Word Processing Pac ($180?). Being somewhat lazy, frugal and interested in seeing state of the competition (for if I write my own), I decided to try SWORD. I'm glad I did.

I sent in my check and waited. In their ad, they promised delivery within ten days. In my case, it took a little bit longer. They were so apologetic about the delay that they gave me a free surprise. In truth, they were having duplication hardware problems which caused the delay, but their behavior showed they really care for their customers. When it did arrive, the very professional packaging job was immediately evident. Many companies have not yet learned that the packaging is very important to aid customer confidence. (If the packaging is amateurish, what about the contents?)

The documentation is perhaps the best I've seen in microcomputer products. There is an introduction section which tells how to load and start the program, then there are good (clear and understandable) descriptions of how to use the various functions. In the end, you are led through a practice session on a sample text file, also included on the cassette. There are also instructions on how to reconfigure SWORD for a custom system. As an example of what SWORD can do, the whole instruction manual was written with SWORD, as was this article.

The program can input or edit text, save or load text files to/from tape, print the text on the printer or on the screen (formatted for each of those), or change the output environment (more on this later). All commands are single keystroke responses, usually to a menu. You may almost forget where the Return key is located. In the insert/edit mode, you type continuously, with the text wrapping around to the next line. Graphic characters are used to tell the output processor when to indent (left or both), start new lines, paragraphs, pages, justify right and/or left, change the spacing, center or right adjust lines. Control characters tell when to insert or delete characters, move blocks of text, or move ahead or back in the file, or move the cursor (with the arrows). The commands are consistent (control C or E does the same thing in any mode, which is important to know).

There is an interesting feature

called the environment. This has separate lists of output parameters for the video or printer outputs. You can specify the widths of the margins and lines, the lengths of pages, the choice of page numbering or a special page header. You can program keys to have special ASCII codes, for easy printer control, etc. You can even specify the driver address of that output function. All of the environment parameters are saved with your text files on tape, so you can reclaim any special features later.

SWORD comes configured for the Centronics output routine. However, there is space reserved inside SWORD for a custom driver. This allows any printer to be used with this program. So, I put my MX80 driver in this space and changed the driver address parameters, and saved my new customized SWORD on tape. (There are instructions for all of this in the manual!) Now LOGing one file gets me a personalized word processor.

SWORD can be saved on disk (starts at 0100), and can run in an 8K machine.

I am impressed and happy with this product. It seems to have very few flaws. I have found only one bug (if you forget to tell it where to move a text block to, it will loop forever), and only one inconvenience (there should be a string search and replace command). In the inser/edit mode, the text on the screen is very dense, making it easy to get lost. I thought that would be a limiting flaw, yet I find myself writing longer and longer documents.

In summary, I think SWORD is an excellent product. It does a lot very well, with few problems, and at a low price too. I recommend it to anyone. My compliments to author R.L. Henne, and distributor NorthAmerican Software.○

PRINT 'EM W I D E

It is possible to print wider than the 63 character width allowed with the Basic ROM Pac. To do this, simply POKE (322,xx) where xx is the line length. The same can be achieved in MBasic by POKE(1929,xx).○

TAPE OPERATION INDICATOR LIGHT

by David Cooke

Here is a helpful hint that I've used with a lot of success in loading and saving tapes.

Get a small 5V LED from Radio Shack (about $2 for 3) and wire it in parallel with the one you can see through the grill in back. Next, drill a small hole and install the LED just behind the ESC key in the dark brown overlay.

This will give you an excellent indicator of tape operation that is more convenient than peering through the grill. After watching it a while, you'll be surprised how easy it is to tell how arrays are loading, when its found the header, etc. Also, the FI (FILE) command in the Monitor does the same CRC check when reading a file as when it is loaded. It is easy to verify a good CSAVE without dumping your program.○

(SPREADSHEET continued from page 82)

```
12960 IF P=1 THEN P=0:GOTO 13050
13000 IF LEFT$(Q$,1)="Y"THEN FX=1
13005 OUT 0,28:INPUT "Indent 10 from present margin";Q$
13007 IF FX=1THEN OUT0,28:PRINTTI$SPC(60-LEN(TI$));:FX=0
13010 POKE -16434,0: REM   Baud 300 (BFCE)
13020 POKE -16432,30: REM  SE O=30, addr of SRLDVR.  For Cen,
13030 POKE -16431,0: REM  put CENDRV pgm in DATA lines 940-5
***   with two leading and three trailing zeros to match length
***   of serial driver and placement of video echo bytes.
13035 IF LEFT$(Q$,1) <> "Y" THEN 13040
13038 PRINT SPC(10) CHR$(27) "9": REM  ESC 9 sets left margin
***                    (on some printers like Diablo and CItoh)
13040 RETURN
***  --------------- Turn printer off ------------------------
13050 POKE -16432,240: REM  SE O=V (E9F0,equ E01B)
13060 POKE -16431,233
13080 POKE -16434,64:REM  BAUD 1200
13090 RETURN
***
***  ---------------"F" Display formulas ----------------------
***
***  --- Starts with clear lower screen routine used also by
***      amorization program.
***
14000 OUT 0,20:FOR I=1 TO 9:PRINT SPC(62)" ":NEXT
14002 OUT 0,20:FORN=0TO62:PRINTCHR$(184);:NEXT:PRINT
14003 IF IN$="J" THEN RETURN
14004 OUT 0,21
14005 FOR A=1 TO REL:B=1       :REM  A=REL#, B=STAGE
14010 IFCN(A,B)=0ANDDC(A,B)=0ANDOC(A+1)=0ANDOC(A+2)=0THEN14250
***
***  If not constant and no 2nd operand in this relation and no
***  or zero column address for next two results then stop.
***
14020 GOSUB 700
14040 GOSUB 730
14050 IF EI(A,B)=0 THEN PRINT ".":GOTO 14250
14055 FOR B=2 TO 5
14060 IF CN(A,B)=0 AND DC(A,B)=0 THEN 14250
14070 GOSUB 700
14080 PRINT FUNC$(A,B) FAC$;
14090 IF EI(A,B)=0 THEN PRINT ".":GOTO 14250
14100 NEXT B
14250 NEXT A
14255 IF FRA(1)=0 THEN PRINT:PRINT"No addition
specified.":PRINT:GOTO 14360
14260 PRINT"Add rows";
14270 FOR S=1 TO 3
14280 IF FRA(S)=0 THEN 14300
14290 PRINT FRA(S)"to"LRA(S) "onto row" STT(S) CHR$(1)";";
14300 NEXT
14310 PRINT:PRINT "excluding ";
14320 IF EX=0 THEN PRINT "none";:GOTO 14350
14330 FOR S=1 TO EX:PRINT ","XC(S);:NEXT
14350 PRINT".":PRINT:PRINT
14360 PRINT"Review the formulas. If any derived values (left of =)";
14363 PRINT " are used"
14370 PRINT "as arguments in another formula (right of =), ";
14373 PRINT "the relation which"
14380 PRINT "derives the value must occur ahead of that ";
14383 PRINT "which uses it. (Menu)"
14390 INPUT IN$
14400 GOTO 114
15000 REM ------------------- Delete a formula ----------------
15005 PRINTCHR$(12)
15007 A=1
15010 INPUT"Number of the formula to be deleted";DL
15020 FOR I=DL TO REL
15023 IF A=1 THEN 15030
15025 FOR I=REL+1TODL+1 STEP-1
15030 OC(I)=OC(I+A):OW(I)=OW(I+A):DJ(I)=DJ(I+A):DK(I)=DK(I+A)
15040 FOR N=1 TO 5:CN(I,N)=CN(I+A,N):CO$(I,N)=CO$(I+A,N)
15050 DC(I,N)=DC(I+A,N):DR(I,N)=DR(I+A,N):FUNC$(I,N)=FUNC$(I+A,N)
15060 EI(I,N)=EI(I+A,N):NEXT N:NEXT I
15070 IF A=1THENREL=REL-1:GOTO 15085
15080 REL=REL+1
15085 GOTO 14000
16000 REM ------------------- Insert a formula ----------------
16005 PRINT CHR$(12)
16010 INPUT"At what number will te list be opened for insertion";DL
16020 PRINT:PRINT"On seeing formula #"dl"repeated as #"DL+1",
      command 'R' and"
16030 PRINT"specify #"DL"for revision, then enter the Relationship
      to insert."
16040 A=:1:GOTO 15025
***
***
```

(continued on page 84)

(SPREADSHEET continued from page 83)

```
***  ----------"R" Define relationships section ----------
***
20000 IF OC(REL)=0 THEN 20010
20005 REL=REL+1
20010 IF P=0 THEN PRINT CHR$(12)
20012 Q$="E"
20020 PRINT"Relationships are entered in the form:"
20040 PRINT:PRINT "Destination value at (column, row) is ";
20043 PRINT "the result of Data 1"
20050 PRINT"(column, row) affected by either a Constant or ";
20053 PRINT"other Data, via"
20060 PRINT"a Function (* / + or -).  If the result of this ";
20063 PRINT"calculation is"
20070 PRINT"an Intermediate element rather than the final ";
20073 PRINT"Entry for the"
20080 PRINT"destination position, queries will prompt ";
20083 PRINT"the next constants,"
20090 PRINT"data, and functions needed to complete the ";
20093 PRINT"formula."
20120 PRINT:PRINT"The relationships are numbered as you ";
20123 PRINT"enter them, and to be"
20130 PRINT"changed must be called by number and ";
20133 PRINT"re-entered completely."
20140 PRINT:PRINT ">>>>>>>> Do not enter straight columnar ";
20143 PRINT"or row  addition in"
20150 PRINT "this section.  Later queries will offer that ";
20153 PRINT"option.":PRINT
20160 PRINT "'Enter column, row' wants numbers with a ";
20161 PRINT"comma between them.":PRINT
20163 PRINT "Derived values used as arguments must be ";
20164 PRINT"derived before use."
20165 PRINT:PRINT"This will be Relationship #"REL".":PRINT
20168 Q$="N"
20170 PRINT "You will be asked column and row numbers for ";
20173 PRINT"the destination"
20174 PRINT "value and all the argument values.  You may now:"
20180 PRINT"      Delete a relationship formula,"
20181 PRINT"      Insert a formul, review the Sheet,"
20182 PRINT"      enter Column addition parameters only,"
20183 PRINT"      pass on to Enter the next relationship,"
20184 PRINT"      or enter the NUMBER of a relationship to ";
20185 INPUT"be re-defined.        ";Q$
20193 IF LEFT$(Q$,1)="I" THEN 16005
20194 IF LEFT$(Q$,1)="S" THEN GOSUB 2030
20195 IF LEFT$(Q$,1)="C" THEN GOTO 20520
20196 IF LEFT$(Q$,1)="D" THEN 15005
20197 IF Q$="" THEN Q$="E"
20198 IF ASC(Q$)<58 THEN RH=REL:REL=VAL(Q$):N=0
20199 PRINT:PRINT
20200 INPUT"Destination: (enter column, row) "; OC(REL), OW(REL)
20205 PRINT:CN=0:N=0
20210 INPUT"equals Data 1 (enter column, row) ";DJ(REL), DK(REL)
20214 PRINT
20215 N=N+1
20230 INPUT"acted on by a Constant, or other Data  (choose) ";Q$
20231 PRINT
20239 IF Q$="D" THEN CN(REL,N)=0:GOTO 20250:REM---Constant fctr
20240 PRINT"Constant (express a percentage as decimal: 10 as . 1)";
20241 INPUT CN(REL,N):PRINT
20242 INPUT "Name of constant"; CO$(REL,N)
20244 GOTO 20260: REM                          --- Data factor
20250 PRINT "Where is the other data (enter column, row) ";
20253 INPUT DC(REL,N),DR(REL,N)
20260 PRINT:PRINT "Enter the function by which the latter ";
20263 PRINT"data or constant acts on"
20270 INPUT "data 1.  *, /, +, or -";FUNC$(REL,N)
20280 PRINT:INPUT "Is result the Entry or an Intermediate";Q$
20400 IF Q$="I" THEN EI(REL,N)=1:GOTO 20410
20405 EI(REL,N) = 0:GOTO 20450
20410 PRINT:PRINT "The intermediate value in the destination";
20415 PRINT" slot is now to be": GOTO 20215
20450 PRINT:PRINT "To enter more relationships, type 'M'. ";
20453 PRINT" To define column"
20460 PRINT "addition, type 'C'.  To return to ";
20463 PRINT "sheet, type 'S'."
20470 PRINT"You may always return to define more relationships."
20472 IF RH<>0 THEN REL=RH:RH=0
20473 INPUT " ";Q$
20480 IF Q$="M" THEN 20000
20500 IF Q$="S" THEN IN$="H":GOTO 120
20505 IF Q$="" THEN 114
20510 IF Q$<>"C" THEN PRINT "Re-enter your choice.":GOTO 20450
***
***  ---------- Column addition parameters section -------
***
20520 SEC=1:PRINT:PRINT "The column may be totaled ";
20523 PRINT"entirely, with defined exceptions,"
```

(continued on page 85)

THE VENEZUELAN CONNECTION

by Roger Pfeil

Thank you very much for your fine Newsletter. It is very difficult to get Sorcerer-related information here in Venezuela. The printer-drivers and Data-save routines, which I found there, have eliminated hours of headaches. I just wish I had had all this information available right after I bought my Sorcerer in 1979.

Let me just share some of my own findings with you, which might be useful for other readers:

Serial Interface Cable

I bought both my parallel and serial interface cables from EXIDY. There was no problem with the parallel cable: I plugged it into the computer and an MX-80 printer, pressed the **P** command of my WP Pac, and off she went.

Not so with the serial cable: Again I plugged in both ends connecting the Sorcerer with my Anderson-Jacobson printer and tried the print command: --- nothing! Thinking back, I am thankful that neither the typewriter nor the computer blew up.

The serial cable consists of a flat 25-wire cable and a printed circuit board with outlets for tape saving and recording, including remote controls, as well as a 25 pin outlet for a RS-232 device.

First I found out that I had to invert wires No. 2 and 3. Still nothing!

Secondly, checking the wire connections of the Serial Cable, I noticed that the 25-line flat cable, which comes with EXIDY's Serial Connection had been placed into the plug inverted: Pin No.1 became No.13 at the other end. I opened up the plug, turned the flat cable over and inserted it again.

Now the printer worked with the WP Pac, but not with BASIC programs.

Thirdly, for some reason beyond my comprehension, the O=S command does not work with the serial port, like the O=L command does with the parallel output. I would expect EXIDY to mention this fact in their **Guided Tour of Personal Computing** and provide a Serial Driver in this Manual for the Sorcerer. Finally I found the information in the **Technical Manual**, but still had to modify the Machine Language program to include a delay, which took me, the beginning computerist, many other hours of frustrating work.

Fourthly, I hooked up my two tape recorders to the respective outlets on the printed circuit board of EXIDY's Serial Cable. After typing CSAVE and RETURN, I or rather EXIDY really blew it. The tape recorder quit working completely (a fuse had blown) and the Sorcerer refused to SAVE any more text (the transistor of the Sorcerer's Cassette output had been broken). The problem is that EXIDY has interconnected all Ground and Shield wires on the printed board, even though they warn in the **Technical Manual** (p.36): "The Mic 2 shield must be completely insulated so

that it cannot possibly touch any of the other shields." My own problem was similar, even though I was working only with mike 1: The ground of the motor control of my Sony recorder (TC-860) is different from the mike-shield. Shorting the two caused the trouble.

In **summation**: Do not waste your $98.- on a piece of equipment that is so poorly done. For less than $20.- a friend with a soldering iron could fix you a better cable following the schematics on pages 35-36 of the **Technical Manual.**

Graphics With Word Processor

The capability of the Sorcerer to use foreign letters was my main reason for purchasing this computer, since I am a Hebrew teacher at an evangelical Seminary. However, the WP Pac reloads the graphics area even if it is entered from the Monitor with a warm-start using "PP" or "GO C003".

One way of avoiding the reload is to re-enter the WP through the back-door. I am using the end of the Y-table (DA70) for this purpose:

Enter the Monitor with the X Command.

Note the location of the Shift-Graphic characters by punching the corresponding keys. Note: The letters are in alphabetical, not in QWERTY order.

Load your letters into the corresponding location.

GO DA70

Write your letter, backspace and underline. The graphic character will appear on the screen instead of the black-on-white letter.

This procedure will only display graphic or foreign characters on the screen. The output to the printer will still be the underlined original letter, unless a special printer-driver is used. Perhaps somebody could develop such a driver. It would have to take a letter and place it into a buffer. Then it would check the next letter, to find out whether it is a back-space. If it is, it would eliminate the backspace and the following underline character and enter the first letter into a special subroutine, whereby it would enter the printer into a dot-graphics mode and send the 5x8 code for the foreign letter from a look-up table. Has anyone done anything similar to this yet?

RAM Breakdown

The other day my Sorcerer got stuck in the middle of a program. No more input from the keyboard. I switched the machine off and on again. The first two letters of the sign-on could be seen, then everything was stuck again. Pressing RESET sometimes produced some interesting graphics.

Removing the BASIC Pac did not

(continued on page 86)

(SPREADSHEET continued from page 84)

```
20530 PRINT "or divided into two or three subtotals which ";
20533 PRINT"may or may not be"
20540 PRINT "combined via a 'relationship'.  If you do not ";
20543 PRINT"wish to divide the"
20550 PRINT "column, you will call it all 'Section 1'`and ";
20553 PRINT"pass the remaining"
20555 PRINT "queries.  ";
20560 PRINT "What is the first and last row to be added in"
20565 FRA(SEC)=0
20570 PRINT "Section"SEC"?  (Enter first, last row ";
20573 PRINT"numbers.)    (You may"
20580 PRINT"exclude selected rows next.  Pass with a comma.)";
20581 INPUT FRA(SEC), LRA(SEC)
20583 IF FRA(SEC)=0 THEN 20600
20585 PRINT:INPUT "On which row will total appear";STT(SEC)
20590 SEC=SEC+1:IF SEC<4 THEN 20560
20600 EX=0:PRINT
20610 PRINT "How many rows altogether will you exclude from ";
20611 INPUT "addition";EX
20615 IF EX=0 THEN 20680
20620 PRINT:PRINT "Which rows are to be excluded from ";
20623 PRINT"column addition?  (If"
20630 PRINT "none, pass.)";
20640 FOR I=1 TO EX: INPUT XC(I):NEXT
20650 PRINT:PRINT "You chose rows";:FOR I=0 TO EX
20653 PRINT XC(I);:NEXT:PRINT "  OK?"
20660 INPUT Q$
20670 IF LEFT$(Q$,1)="N" THEN 20600
20680 PRINT:PRINT"All columns will be added.  There is no ";
20683 PRINT"provision for non-"
20690 PRINT"add columns.  Now you may define more Relationships"
20700 PRINT"or make any other menu choice including Kalkulate."
20705 INPUT IN$
20710 GOTO 114
***
***  ---------------"K" Kalculations section ------------------
***
***  --- Each column is calculated 4 times, with section
***      addition between each calculation.
***
22000 C=0:K=0:Q$="":REM  300=calculation, 500=addition
22010 OUT 1,29:PRINT"Figuring column"C+1"...";:GOSUB 300
22110 SEC=1
22120 V(C+1,STT(SEC))=0
22130 GOSUB 500:REM                          Add first section
22150 GOSUB 300:REM                          Calculate 2nd time
22160 SEC=2:IF STT(SEC)=0 THEN 22227
22165 V(C+1,STT(SEC))=0
22170 GOSUB 500:REM                          Add second section
22180 GOSUB 300:REM                          Calculate 3rd time
22190 SEC=3:IF STT(SEC)=0 THEN 22227
22200 V(C+1,STT(SEC))=0
22210 GOSUB 500:REM                          Add third section
22220 GOSUB 300:REM                          Calculate 4th time
22225 GOSUB 10000:OUT 1,27:REM               Write sheet w/ results
22227 IF C+1=COLS THEN 22250
22228 IF K=1 THEN 22240
22230 INPUT " do Next col, All cols, or Stop";Q$
22235 IF LEFT$(Q$,1)="A" THEN K=1:Q$="N"
22240 IF LEFT$(Q$,1)="N" THEN C=C+1:GOTO 22010
22250 IN$="M":GOTO 120:REM  Write the sheet with move options.
***
***  ---------------" " Clear out all values, keep sheet ------
***
*** ---- This is not given a command letter because it is
***      dangerous.  To use it, one must break the program
***      and Goto 30000.  It is used to save a blank sheet with
***      its labels and formulas, ready for re-use.
***
30000 FOR I=1 TO COLS:FOR N=1 TO ROWS:V(I,N)=0:NEXT N
30003 NEXT I:IN$="":GOTO 114
***
***  ---------------"J" Amortization calculation --------------
***
30050 INPUT"Enter amount, term (months), & an. interest.";P1,MO,A
      30060 A2=A/100/12:P2=P1*(A2*(1+A2)MO)/(((1+A2)MO)-1)
30070 IF(P2-(100*P2-INT(100*P2)))<.5THENP2=INT(100*P2)/100:GOTO30090
30080 P2=(INT(100*P2)+1)/100
30090 PRINT"Monthly payment is $"P2".":INPUT IN$:GOTO 114
***
***  ---------------"S" Set or clear row totals ---------------
***
30150 IF P=0 THEN PRINT CHR$(12)
30160 IF TC<>0 THEN 30250
30200 INPUT "Which column is for row totals";TC
30205 IF TC>13 THEN PRINT "Too high.":GOTO 30200
```

(continued on page 86)

(SPREADSHEET continued from page 85)

```
30206 IF TC>COLS+1 THEN PRINT "Too high.":GOTO 30200
30207 CD=COLS:IF TC>COLS THEN COLS=COLS+1
30209 CN$(0)=CN$(TC):CN$(TC)="ROW TOTAL"
30210 FOR R=1 TO ROWS:V(0,R)=V(TC,R):V(TC,R)=0
30212 FOR C=1 TO TC-1:V(TC,R)=V(TC,R)+V(C,R):NEXT C
30215 NEXT R
30220 IN$="M":GOTO 114
***   ---------- Clear the totals column, restore original
30250 CN$(TC)=CN$(0):FOR R=1 TO ROWS:V(TC,R)=V(0,R):NEXT
30255 IF COLS<>CD THEN COLS=COLS-1
30260 TC=0:PRINT
30265 INPUT"Totals column erased.  Do you want a row totals
      column";Q$
30270 PRINT:IF LEFT$(Q$,1)="Y" THEN 30200
30275 IN$="M":GOTO 120
***
***   ----------"V" Plot a graph section ----------
***
30500 IF P=0 THEN PRINTCHR$(12)
30505 PRINT:INPUT"Print graph when done";Q$
30510 PRINT
30520 PRINT"Title for graph (appears at bottom, 60 ch.
max.)":INPUTTI$
30540 PRINT:INPUT"Plot how many rows (max. 6)";PA
30550 PRINT:PRINT"Which rows, by number (use "0" to recall them)";
30560 FOR I=1 TO PA:INPUTP(I):IF P(I)=0 THEN IN$="M":GOTO 120
30570 NEXT
30580 H=V(1,P(1)):L=V(1,P(1))
30590 FOR I=1 TO PA:FOR N=1 TO COLS:IF V(N,P(I))>H THEN H=V(N,P(I))
30600 IF V(N,P(I))<L THEN L=V(N,P(I))
30610 NEXT N
30620 NEXT I
30630 IN=INT((H-L)/25)+1
30640 L$="":PRINT:INPUT"Connecting lines";L$
30650 IF P=0 THEN PRINTCHR$(12)
30660 PRINT TAB(10);
30665 FOR I=1 TO 13
30667 IFMID$(CN$(I),4,1)=")"THENPRINTMID$(CN$(I),5,3)" ";:NEXT:GOTO
30680
30670 PRINTMID$(CN$(I),4,3)" ";:NEXT
30680 PRINT
30690 FOR N=0 TO 25:IF N/5=INT(N/5)THENPRINTH-IN*N TAB(10)DT$:GOTO
30710
30700 PRINTH-IN*N
30710 NEXT
30720 PRINT:PRINTTI$
30730 FOR I=1 TO PA:FOR N=1 TO COLS
30740 RC(N)=INT(((H-V(N,P(I)))/IN)+.5)+1:REM ..1 IF ROW 2 IS TOP
30745 RD(N)=INT(((H-V(N+1,P(I)))/IN)+.5)+1
30750 OUT N*4+6,RC(N):PRINTSTR$(P(I))
30757 NEXT N: IF LEFT$(L$,1)="N" THEN 30768
30758 FORN=1TOCOLS-
1:GOSUB800:OUTN*4+8,RC(N):PRINTJA$;JB$;JC$;JD$;JE$
30760
IFP(I)>9THENOUTN*4+6,RC(N):PRINTRIGHT$(STR$(P(I)),LEN(STR$(P(I)))-1)
30764 NEXT N
30768 NEXT I
30770 IF LEFT$(Q$,1)="Y"THEN 30800
30790 Q$="":OUT 50,27:INPUT"Menu";IN$:GOTO 114
***   ---- Before printing, cancel video echo in driver.
30802 IF PEEK(30)=0 THEN PRINT"    ";:REM ADJUST PRINTOUT
***   If skew observed on printing screen dump of plot, adjust
***   the top line of spaces.  Four are lost with some printers.
***   Line 30802 senses presence of Centronics driver and adds
***   spaces to compensate, since the problem was observed on a
***   parallel printer.
30805 FORI=-3968TO-2049:IF PEEK(I)=128 THEN PRINT "|";:NEXT
30810 PRINTCHR$(PEEK(I));:NEXT:PRINT:
30820 POKE33,205:POKE34,27:POKE35,224
30840 GOSUB 12950:GOTO 30790

***
***   For a Centronics-type parallel printer, alter the data
***   lines as follows:
***   940 DATA 0,0,245,205,27,224,254,10,40,20,245,219,255,203,
***   127,32
***   942 DATA 250,241,246,128,211,255,230,127,211,255,246,128
***   945 DATA 211,255,241,201,0,0,0
***
***   If printer needs a linefeed sent by the routine, convert
***   seventh through tenth bytes to zeroes.
```

DISASSEMBLY OF ZERO PAGE

```
0000:  E5          PUSH HL        CURSOR ADDRESSING
0001:  47          LD   B,A
0002:  CD E8 E9    CALL E9E8
```

(concluded on page 87)

(VENEZUELAN cont'd from p. 85)

help either. Only a dash showed up, no input accepted. However, the WP Pac worked almost fine, as long as I did not go into the Monitor. Text returned from the H Command changed somewhat.

Since I could not go into the Monitor I could not do any memory test. Fortunately, I had just received a set of RAM chips for a memory expansion. I exchanged the chips and eureka it worked! In my case, some of the RAM chips in the upper part of memory had broken down. Since the computer could not set up its stacks, it did not start properly. A broken-down RAM chip in the lower part of the memory will allow the Monitor or the BASIC Pac to start, but not the WP Pac.

Since the RAMs of the stack are used a lot, I would expect them to break down first. Have an extra set of RAMs on hand (they are only $16 now for the set). If your computer stops working like mine, try exchanging your RAM chips, 8 at a time. It may save you an expensive repair bill and some valuable time out.

Macros

Automatic Paragraph Composer

Apdo. 4713, Maracay, Venezuela

1The following macro will automatically compose letters for you from numbered paragraphs.

2Type Graphic-6 followed by the numbers 1 to 0.

3Type your paragraph. It can be of any length, limited only by the memory of your computer.

4Type a Graphic-9 followed by a RETURN on the next line.

5Repeat procedure from No. 1, until all paragraphs are typed. You can use up to 17 paragraphs, coded as mentioned in No.2.

6Add the following line after the Graphic-9 of your last paragraph: "Graphic-6 X Graphic-6 - (Dash) Graphic-6 - Graphic-6 - Graphic-6 - Graphic-9 RETURN" (without spaces, as shown in the last paragraph here).

7After composing all paragraphs, obtain a printout, which will serve for your reference. Since the printer ignores the Graphic-6 and the following letter, you have to write in the number of the paragraph by hand.

8Now type your letterhead, the receiver's address, and an initial phrase, anything you want to appear before the main body of the letter (See example below).

9Type a Graphic-6, followed by Graphic-9, followed by more Graphic-6 with the numbers or letters of the paragraphs desired for the first letter. The order of the paragraphs is not important.

0After some line-feeds, write your final phrase and name. You may also add some personal words,

which are not in the paragraphs, before or after the main body of text represented by the codes for the paragraphs. Terminate your letter with a Graphic-1 (Formfeed) and a Graphic-8, both followed by RETURN. Read your Macro into the Computer. Position the Cursor at the beginning of the Macro. Give the A-Command. Switch on your printer. Now type an **A1** Command to see the letter composed from the paragraphs and printed. In the EDIT mode insert a new name and address as well as new numbers for the paragraphs of your following letter. An **A1** Command will do the rest for you. If you want to do some Reading or Writing from or to Tape, first take the Macro out of its buffer with the **A0** Command. Otherwise, you will lose part of your Macro.

X----

Roger Pfeil
Apdo. 4713
Maracay,
Aragua
Venezuela,
2101A
April 10,
1982

Sorcerer's Apprentice
P.O. Box 33
Madison Heights, MI 48071

Dear Ralph,

Sincerely
yours,

Roger

```
t/
s/1/./h0/h/u
s/1/./u
t
s/2/./h0/h/u
s/2/./u
t
s/3/./h0/h/u
s/3/./u
t
s/4/./h0/h/u
s/4/./u
t
s/5/./h0/h/u
s/5/./u
t
s/6/./h0/h/u
s/6/./u
t
s/7/./h0/h/u
s/7/./u
t
s/8/./h0/h/u
s/8/./u
t
s/9/./h0/h/u
s/9/./u
t
s/0/./h0/h/u
s/0/./u
t
s/X/./f1
p
t
s/X/./h0/h1/u
s//./d/d1/u
b7
```

(SPREADSHEET continued from page 86)

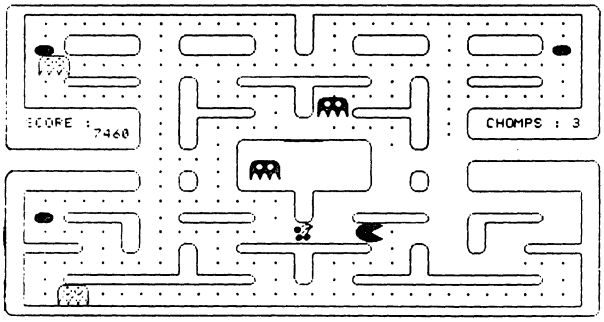
```
0005:  AF           XOR  A
0006:  FD 77 6B     LD   (IY+6B),A
0009:  CB 18        RR   B
000B:  1F           RRA
000C:  CB 18        RR   B
000E:  1F           RRA
000F:  FD 77 68     LD   (IY+68),A
0012:  FD 70 69     LD   (IY+69),B
0015:  3A 3F 01     LD   A,(013F)
0018:  FD 77 6A     LD   (IY+6A),A
001B:  E1           POP  HL
001C:  D1           POP  DE
001D:  C9           RET
001E:  C5           PUSH BC          SERIAL DRIVER
001F:  F5           PUSH AF          (Centronics driver must
0020:  F5           PUSH AF           fit in same space with..
0021:  CD 1B E0     CALL E01B         VIDEO ECHO
0024:  3E 80        LD   A,80         in same position)
0026:  D3 FE        OUT  FE
0028:  F1           POP  AF
0029:  CD 12 E0     CALL E012
002C:  DB FD        IN   FD
002E:  CB 47        BIT  0,A
0030:  28 FA        JR   Z,002C
0032:  01 00 08     LD   BC,0800
0035:  0B           DEC  BC
0036:  79           LD   A,C
0037:  B7           OR   A
0038:  20 FB        JR   NZ, 0035
003A:  78           LD   A,B
003B:  B7           OR   A
003C:  20 F7        JR   NZ,0035
003E:  F1           POP  AF
003F:  C1           POP  BC
0040:  C9           RET
0041:  11 12 E0     LD   DE,E012     STRSVON  (OUTTAPE)
0044:  FD 73 3F     LD   (IY+3F),E
0047:  FD 72 40     LD   (IY+40),D
004A:  C9           RET
004B:  11 1B E0     LD   DE,E01B     STRSVOFF (VIDEO)
004E:  FD 73 3F     LD   (IY+3F),E
0051:  FD 72 40     LD   (IY+40),D
0054:  C9           RET
0055:  11 0F E0     LD   DE,E00F     STRLDON  (INTAPE)
0058:  FD 73 41     LD   (IY+41),E
005B:  FD 72 42     LD   (IY+42),D
005E:  C9           RET
005F:  11 18 E0     LD   DE,E018     STRLDOFF (KEYBD)
0062:  FD 73 41     LD   (IY+41),E
0065:  FD 72 42     LD   (IY+42),D
0068:  C9           RET
0069:  02           LD   (BC),A      RAMWriter
006A:  03           INC  BC
006B:  C9           RET
006C:  D5           PUSH DE          ;Subroutine - CSAVE* substitute.
006D:  E5           PUSH HL          ;Places start and end address of
006E:  F5           PUSH AF          ;array bloc in Monitor command
006F:  C5           PUSH BC          ;buffer after name, for SAve.
0070:  FD 36        LD   (IY+3F),69  ;Make outaddr eq Ramwriter
0074:  FD 36        LD   (IY+40),00  ;
0078:  ED 5B B9 01  LD   DE,(01B9)   ;DE points to start of arrays
007C:  01 9A BF     LD   BC,BF9A     ;BCpoints to part of cmd line
007F:  CD E8 E1     CALL E1E8        ;Call Send Ascii Hex of 2 bytes
0082:  03           INC  BC
0083:  ED 5B BB 01  LD DE,(01BB)     ;DE points to end of arrays
0087:  CD E8 E1     CALL E1E8        ;Call Send Ascii Hex again
008A:  FD 36        LD   (IY+3F),1B  ;Make outaddr eq Video
008E:  FD 36        LD   (IY+40),E0
0092:  C1           POP  BC
0093:  F1           POP  AF
0094:  E1           POP  HL
0095:  D1           POP  DE
0096:  C9           RET
0097:  D5           PUSH DE          ;Subroutine - CLOAD* substitute.
0098:  E5           PUSH HL          ;Places start - arrays address
0099:  F5           PUSH AF          ;in Monitor command line buffer
009A:  C5           PUSH BC          ;after "LO 1" for a LOad.
009B:  FD 36        LD   (IY+3F),69  ;Make outaddr eq Ramwriter
009F:  FD 36        LD   (IY+40),00
00A3:  ED 5B B9 01  LD   DE,(01B9)   ;DE points to start of arrays
00A7:  01 96 BF     LD   BC,BF96     ;BC points to correct position
00AA:  CD E8 E1     CALL E1E8        ;in command line.Call Send Ascii
00AD:  FD 36        LD   (IY+3F),1B  ;Make outaddr eq Video
00B1:  FD 36        LD   (IY+40),E0
00B5:  C1           POP  BC
00B6:  F1           POP  AF          ;Note that BF9A and BF96 are
00B7:  E1           POP  HL          ;  specific to a 48K Sorcerer.
00B8:  D1           POP  DE
00B9:  C9           RET
```

ENSIGN SOFTWARE

FREE CATALOGUE

CHOMP $19.95

is an absolute must for every game enthusiast. It is better then the Pac-Man arcade game where you are being chased through a maze of alleys by four Monsters who will eat you if they can catch you.



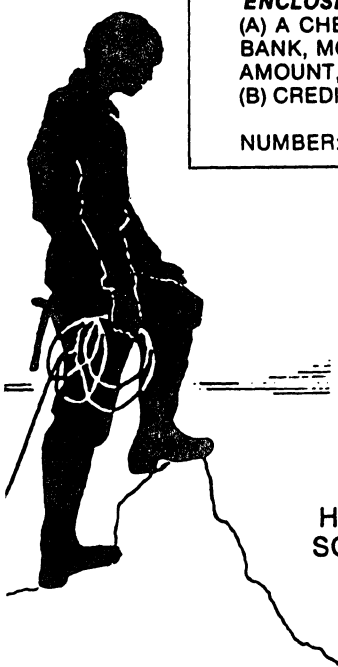
DISASSEMBLER $17.95

is a Z-80 machine language two-pass disassembler whose output format is directly compatible with the Development Pac. The Z-80 assembly language source (input to assembler) listing can be sent to Video, Cassette or Printer. The cassette file produced is a source file for the Editor/Assembler and can be read directly into the editor of the Development Pac. The disassembler has a displacement function which allows any program residing anywhere in memory to be decoded, whether it is at its normal address or has been moved to be decoded.

PRODUCT	PRICE
OVERSEAS POSTAGE: ADD $3 NO POSTAGE CHARGE IN USA & CANADA	
VISA MasterCard TOTAL	

ENCLOSE:
(A) A CHECK IN U.S. DOLLARS DRAWN ON A U.S. BANK, MONEY ORDER OR CASH FOR THE ABOVE AMOUNT, OR
(B) CREDIT CARD, EXPIRATION DATE_____

NUMBER:.....



MAIL YOUR ORDER TO:

ENSIGN SOFTWARE
2312 N. COLE RD., SUITE E
BOISE, IDAHO 83704 U.S.A.

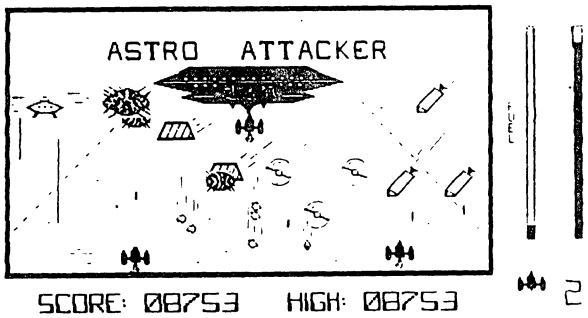
PHONE: (208) 378-8086

HIGH QUALITY SOFWARE AT A LOW PRICE

THE NEW ARRINGTON SOFTWARE SERVICE

ASTRO ATTACKER $21.95

is similar to the arcade game called "ASTRO BLASTER". This action game for the Sorcerer is far superior to all other Sorcerer games because of its high resolution graphics, sound, variety and playability. Astro Attacker graphics are extremely advanced. The display is of the console inside your astro fighter craft. In your console window you see the enemy ships placed against a background of continuously moving stars. Gauges also indicate the amount of fuel remaining and the temperature of your lazer cannons. If you fire too frequently you can overheat the lazers, or if you move recklessly you may run out of fuel. Your challenge is to survive and destroy the Spinners, the Lazer Ships, the Rockets, the Flame Throwers, and the Meteor shower. Docking with the mother ship is crucial to survival as this restores your shield strength and fuel, and cools your lazer cannon. With each succeeding level of play, survival becomes more difficult as the enemy ships attack with greater frequency and quickness. Superb sound too.



DATABASE SYSTEM II $29.95

is a RAM-based general purpose database system for handling alphanumeric data. It is written in Z-80 machine language and is suitable for use in 32K or 48K Sorcerer microcomputers. Files may be stored on cassette or on disk under the CP/M 1.4 or later operating system. This database is useable by both cassette based systems and/or disk based systems.

Commands available in functional groupings are:
A: File Definition: CREATE
B: File Input/Output: LOAD, MERGE, SAVE
C: File Alteration: ADD, DELETE, EDIT, SORT
D: File Listing: LIST, REPORT, TOTAL
E: System Parameters: PRINTER, SPACE, TABSET
F: Program Exit: CPM, MONITOR

Allowance is made for up to 750 records which may consist of 1-9 fields. Each field is given a name by the user when creating the database and this is used as an aid in manipulating the file. The space available for record storage is approximately 9K less than the available RAM in cassette mode and 14K less than CP/M system size in disk mode. A field may contain up to 56 characters. It is best to divide the record into small fields which are useful for sorting and searching. For example, a file of names and addresses, the fields could be:

NAME, STREET, CITY, and PHONE.

The software is sent on cassette tape, but is easily transfered to your CP/M disk. Use the Monitor >LO command, boot your disk, and then type A> SAVE 27 DATABASE.COM.

(BELLY UP! continued from page 73)

This brings us to a very important question. Where do the Sorcerer owners go from here? Do we just dump the system we have and invest in something new or do we continue to try to make the best of the situation? The answer really depends on what you are using your system for. If the system meets your immediate and short term future needs, hang on to it, because you can't get enough for it to justify the purchase of a new system with equal or better specifications. If the system does not meet your needs then upgrade it or dump it!

If you have the S-100 unit, you can upgrade your system by purchasing a dumb terminal and a new S-100 board that meets your specifications. Dumb terminals are quite reasonable in price (Televideo, ADDS, ADM etc.). Look at Priority One's Spring 1982 Engineering Selection Guide, and you'll see the large variety of S-100 boards available, most of which will work with the Sorcerer without modification. With a new CPU, Video display, and Memory card, plus an inexpensive terminal, you can have a new system with a lot more capability than you now have. The price will probably be less than a $1000, if you shop carefully. You can still use your old disk system and printer. Of course, if you want to upgrade to a 68000 microprocessor, the price will be higher. Priority One has a MC 68000 CPU card available for $1075 (page 3 of the Spring guide). The expansion and upgrade possibilities of an S-100 system are virtually endless. With the Exidy S-100 unit, or any S-100 unit with the Exidy S-100 card (available from South Valley Electronics), you'll never be obsolete. To upgrade, merely buy a new S-100 card (most are quite reasonably priced) and you've got a new system. Your imagination and money are the only limitations. So, there is really no need to give up the Sorcerer. Add CP/M and most of your software problems are solved. If you are contemplating building a business around the Sorcerer, you may not get very far. For all practical purposes, unless something develops wth either Compudata or the licensing deal currently under negotiations, the Sorcerer is dead.

What happens to the users group? Well, that will depend upon the membership at large. There is a large base in Europe and Australia, but not in the United States. (Gary Jensen estimated there to be about 1000 active users in the U.S.) To continue bringing you the Newsletter in its present format, we will need 100% renewals for next year at current prices. We can always reduce the number of pages, or do away with stapling, three ring binder holes, use lower quality paper etc. to continue the Newsletter for fewer members. We're going to need your input regarding the continuance of the Newsletter and the group. Please send your suggestions and comments to us by the end of September, so we may plan ahead for next year. In the meantime, hold on to your Sorcerer, it is still a very good computer! We still have expert service available from Jack MacGrath and B.J. Freeman. There is a lot of software available from Ensign (Arrington) Software, System Software, Dayspring Computer Enterprises and a lot of individual authors. For CP/M users, there is a constantly growing library of software available. O

EASY TYPING PRACTICE

by Larry Kobylarz

The following BASIC program allows you to use the Sorcerer for typing practice. Use carriage return and line feed at the end of each line. Use Control H for backspace. Control C gets you out of the program.

```
10 POKE 318,195: POKE 320,224
20 A=INP(9): IF A=0 THEN 20
30 PRINT CHR$(A);: GOTO 20
```

ON-LINE

by Robert Hageman, System Operator

In this issue, I will cover mapping operating systems, moving files between different systems in the same machine, and moving ROM Pac Basic files from tape to disk.

Mapping An Operating System

1. Clean memory by filling it with all one value, i.e. all nulls or all FFs. One method is to ENter "00" at address 0000 and MOve 0000 to fill all memory except the Monitor work area (MWA). In a 52K machine the move command is: MO 0 CF7E 1.
2. Boot the system to put a fresh image in memory. This is usually a GO command, as in GO D000 to boot a disk controller at address D000.
3. Escape to the Monitor. Hitting both RESET keys will work, but if you can have the system execute code by a jump to an address, try to enter the Monitor at E003.
4. Dump memory. Don't cheat, start at 0000 and work through to your highest RAM address. Note addresses for the following:
   a. Areas occupied by the system.
   b. Areas unoccupied by the system. Try to be specific. If possible, locate boot loaders, main operating system, and data areas used by the system.

+++ +++ +++

Moving Files Between Different Systems In The Same Machine

1. Map both operating systems following the directions given above.
2. Determine the "safe" areas, i.e. those areas neither system uses and those areas each system leaves untouched.
3. Load the file to be transferred from the source system. It is best if you can just load the file without executing it.
4. Escape to the Monitor.
5. MOve the file to a "safe" area for the destination system.
6. Boot the destination system. Hopefully, it will load without touching your file.
7. Escape to the Monitor again.
8. Move the file to a suitable location for this system.
9. Reenter the destination system. If possible, do this through it's warm entry point.
10. Save the file.

+++ +++ +++

Basic Pac Files To Disk

It is possible to get the Basic ROM Pac to write an ASCII tape of the Basic program in memory. This tape can be read by the Monitor serial input routine from CP/M using PIP to put the file on disk.

1. ENter at 0000: CD 12 E0 CD 1B E0 C9.
2. SEt Output = 0.
3. Return to the ROM Pac with PP.
4. Type LIST.
5. Start the recorder.
6. Push RETURN.
7. After READY, type several control-Zs. The "Reader" function of your CP/M BIOS should call or jump to the Monitor serial input routine (like CD 0F E0 C9 or C3 0F E0).
8. Boot your CP/M.
9. PIP the file from the reader to disk: PIP FILENAME.ASC=RDR:. If the file is larger than 16K, you should use PIP's block move option: PIP FILENAME.ASC[B], which tells PIP to buffer the input until it receives a control-S or control-Z. O

USEFUL POKE COMMANDS

by William Cohen

(Reprinted from the May, 1980 issue of the S.U.N.)

In the January 1980 issue (S.U.N. Vol II, page 6) the NEWVIDEO program caused a jump of a line at every line feed. The same operation can be accomplished much easier with the use of a BASIC POKE command. Below are listed a few POKE commands which I found which might be of interest to other users.

1. Double space printing (same as NEWVIDEO) - POKE 322,0
2. Line width control- for BASIC only- POKE 322,xx (where xx = 1,2,3,4....64). This command controls the length of the line. It will be limited to xx length. If the statement is longer than xx characters it will scroll to the next line.
3. Printing speed control- POKE 32719,xx. The larger the xx the slower the print speed. The slowest allowed value of xx is 255. For 8K memory use 8143 and for 16K memory use 16635.

SORCERER STRINGY FLOPPY
Additional Information

by Alan H. Schmid

Issue 4.1 of the Sorcerer's Apprentice, had a review of the Exatron Stringy Floppy (ESF) modified by ASP Microcomputers for use with the Sorcerer. This same issue carried an informative ad by ASP. I do not intend to duplicate this same information, but to build upon it.

I have been using the ASP ESF for eight months. The controller and two drives have seen daily use supporting my stock option trading business. Since I will discuss both the features and the faults, and will probably go into the faults in the greatest detail, it should be stated in the beginning that I am very pleased with the ESF.

This is not going to be a crusade to prove that the ESF is better than a pair of floppy disks. However, if cost is a factor, the ESF is a valid middle ground in the large void that exists between audio cassettes and disks.

Who Should Consider the ESF?

If you already have two disks, go read another article!

If you have only one disk, then save your money for a second, so you can easily maintain backup records.

If you have 48K memory, but only cassettes, you certainly need either disks or ESF.

If you have a Sorcerer I with its maximum factory internal capacity of 32K, plus two cassettes with motor controls, you are a prime candidate for ESF.

If you have a 32K Sorcerer I, and a single cassette without motor control, you have a small current investment, and will benefit the most from the ESF.

If you have less than 32K, the ESF will work well with your system, and use practically none of the computer's precious memory capacity.

Now that we have told the disk owners where to go, let's get on with the details.

ESF Features and Limitations

Please go read the referenced article and ad. I want to spend our time covering new ground.

Concerning SPEED. The ESF operates at approximately 9600 baud verses 1200 baud or 300 baud for the cassette. To illustrate what this means in the real world, I ran a test of loading the SYSTEM 3 Toolkit at both 1200 baud from cassette and with the ESF.

	TIME: Sec.	
TASK	Cass.	ESF
Rewind	15	-
Search for begin. of record	-	1
Load complete - Total elapsed	75	6.8

The ESF increase in transfer rate over the cassette is 9600b/1200b, a factor of eight. However, the system increase in transfer rate is a factor of 11 when the time for human button pushing is included! That could really cut into your day-dreaming time!

Next COST. This is a trickier subject than speed, with many subjective items. The best I can do is make a clear case as I see it, with enough data that you may convert it to your case and point of view.

First the ground rules:

Baseline System - Sorcerer I or II with 32K, 1 or 2 cassettes, Standard BASIC and Word Processing Pacs.

Minimum user available RAM, 28K, after operating system and BASIC loaded.

Minimum of 2 disks required so backup disks can be conveniently made.

While ESF backups do not require 2 drives to make backup records, I will specify 2 to save arguments.

Using, "Disk Notes", in the December, 1981 SA for cost data, the best one could do would be $1200 for two drives. There would still be the cost for increasing 32K to 48K to provide room for the disk operating system and disk BASIC. Also, there would still be the problem of getting the Word Processor back in service. However, let's not get caught up in details. The point is that ESFs are not better than disks, rather the ESF is a valid medium cost alternative!

ESF Costs:

Controller and drive	$339
Second drive	157
WP Driver	50
Total	$546

Note: 1. List price is stated in Australian dollars. However, at writing time, USA and Australian dollars are equal.

Note: 2. I personally feel the second drive and the WP Driver could be omitted from the system (details later).

RELIABILITY of the ESF is as advertised! CSAVE and >SA are verified automatically after recording. While this takes additional time, I have never lost a record. The worst that has happened is a parity error on loading that required a second attempt, or a head cleaning.

While reliability of the hardware is excellent, there is a QUALITY problem with the wafers. A new wafer is certified before initial use, i.e. test data are first recorded, and then read over the entire length of the tape. The Certify command does this once, or a Certify Loop command will do it 10 times in succession. I have found that in a new batch of wafers, approximately 10% will fail the certification. These are usually the longer wafers, 35 and 50 foot long. Repeated attempts at certification will allow a few marginal wafers to pass, but the remaining are only good for obtaining replacements from Exatron.

Once a wafer has been initially certified, it is a very reliable wafer. The initial certification failures are a significant annoyance, but do not affect the system reliability.

The certification problem seems to be wafer internal friction as the endless loop of tape spirals inward in the wafer, rubbing upon itself. The friction is also sensitive to temperature. My office is heated by wood stove, and drops as low as 40°F in the winter. After soaking at this temperature some wafers may give parity errors on the initial read attempts. Warming the wafer slightly, will eliminate this problem. I suspect most of you would not even have this situation, but mention it only to illustrate how some wafers are very sensitive to internal friction.

The documentation supplied with the ESF is good. It is particularly clear in the section concerning two minor modifications to the printed circuit board, and the insertion of the new Monitor ROMs. I had no problems with the modifications.

The manual details the changes incorporated in the Monitor ROMs. I will only mention four that I find affect the user.

1. The Monitor Work Area (MWA) has been extended 10 bytes further downward into user RAM.

2. The "SE T = " commands to set the baud rate are like the Exidy 1.1 Monitor,and therefore, the reverse of the Exidy 1.0 Monitor you may have.

3. The keyboard debounce routine has been improved. I used to be plagued with double 8's during long input sessions. This has been eliminated.

4. My printer would consistently refuse the first software commands of the day to print. This problem has been eliminated.

Now for the "good/bad news" items. A program is included wih the ESF for saving string arrays. However, the CSAVE* command for saving numerical arrays was not implemented on the ESF. Personally, I find this a bad tradeoff. The CSAVE* and CLOAD* are still available in the cassette mode. There are ways to work around this limitation, but each of you should evaluate this limitation carefully in relation to your use of the Sorcerer before you purchase. Since this is a software limitation, a solution may still appear.

In the next article, I will explain some ways around the CSAVE* problem.

The CSAVE* problem illustrates one of the virtues of the ESF, it augments rather than replaces the cassette. The manual explains how the system "wakes up" in the ESF mode. One can transfer to cassette with the Monitor command "SE M=C" and back to the ESF with "SE M=S". A significant exception to this rule is not documented. While in the ESF mode, the system will save or load data to cassette at 300b.

For me this is a significant

convenience. On a weekly basis I spend an hour doing data input, and then turn the computer loose to generate an updated 20K numerical array. A printout is made for review, and the array saved for historical purposes. I am able to make backup records on the ESF during data input, and later save the array on cassette at 300b without having to remember to "SE M=C".

The ESF uses very little of the Sorcerer's RAM:

The MWA is increased by 10 bytes as noted above.

The Word Processing Driver, if used, occupies 0H to 2FFH.

The ESF control ROM uses addresses B800H to BFFFH. Therefore, in the case of Sorcerers with 48K of RAM, they become effectively, 46K memories.

Earlier, I made reference to only minor changes being required in a program to make it compatible with the ESF. There are basically two:

1. Since the Monitor Work Area has been extended 10 bytes downward, POKEs into the MWA must have their addresses changed. For example, with a 32K memory, POKEs into 32720 & 32721 to turn a printer ON or OFF, would be changed to 32710 & 32711, respectively.

2. Any program references to CSAVE* at 1200b may only be done if the Sorcerer is in the cassette mode. I do not find this a significant limitation since the Sorcerer cannot reliably save data at 1200b.

In case of the Word Processor Pac with the ESF, one has a choice, both acceptable. If one loads the WP Driver program, the ESF will read and write text to the ESF using the standard commands. In addition, all standard cassette operations are still available. The one change is that the ESF files are numbered rather than named. The files are loaded in the same manner as cassettes under motor control, i.e. in 256 byte blocks, and without verification. The use of the WP Driver would be desireable, if the operator was not familiar with Sorcerer Monitor commands.

The alternate method of saving WP files involves the use of the Monitor >SA command. It has the advantages of:

1. Saving a larger file on a given size wafer.

2. Verifying the file when saved.

3. Not requiring the WP Driver Program.

I will cover operational details in the next article. The main point is that the WP Driver is not required to use the ESF with the WP Pac.

Finally, here is the approximate capacity for the available wafer tape lengths:

Wafer Length	Bytes: >SA or CSAVE	Bytes: WP Driver	Run Time sec.
5	6000	2700	6.5
10	12000	5400	13
20	24000	10800	26
35	42000	18000	45
50	60000	27000	65

At present, the ASP ESF controller is manufactured by ASP Microcomputers in Australia. The ESF drives are standard drives manufactured by Exatron in Sunnyvale, CA, and modified by ASP in Australia. I personally appreciate the effort that ASP put into the project, so that there is a mass memory option that filled the gap between cassettes and floppy disks.

The full address for ASP is:

ASP Microcomputers
797 Dandenong Road
East Malvern 3145
Victoria, Australia

SORCERER STRINGY FLOPPY

Operating Techniques

by Alan H. Schmid

In the previous article, I provided information concerning the Exatron Stringy Floppy (ESF), modified by ASP Microcomputers to operate on the Sorcerer. That article was to answer the questions that remain after a typical product review, but before one is prepared to make a buy decision. This article will describe a few operating techniques, hopefully, urging others to submit new or improved ideas for our mutual use.

Saving WP Text W/Monitor >SA

This suggestion came from the Melbourne University, Australia, via the ASP Microcomputers, **"Stringy Floppy Newsletter"**. Paul Stuart, of ASP, deserves thanks for passing on a technique that allows anyone willing to use the Sorcerer Monitor, to use the WP Pac and the ESF without purchasing the WP Driver from ASP.

The advantages of the Monitor >SA method are: approximately twice as fast, more than twice as much text per wafer, automatic verification of a write, and no patch program needed. The disadvantages are: writing a file is slightly more complicated, incoming text will not automatically merge with existing text in RAM and will over-write it. A method to overcome this is included below.

Saving a WP File

1. Exit WP Pac to Monitor: Command X.

2. Dump addresses 074A and 074B: DU 074A 074B. These two bytes contain the end of text address. Note that 074B contains the first part of the address and 074A the second. For example, if 074A contains 1E and 074B contains 12, the end address is 121E.

3. Set the auto execution address to the warm start address for the WP Pac (C003): SE X=C003. (Note: I have not found the above step necessary.)

4. Save the file on wafer using the usual Monitor syntax: SA n 80F 121E. 80F is the starting address for all WP files.

Loading a WP Monitor File

1. Exit WP Pac to Monitor: Command X.

2. Load file with the command LO n, where n is the file number. The file will load and then return control to the WP Pac in the EDIT mode.

Merging Two WP Monitor Files

1. Put existing text into holding buffer using usual WP commands.

2. Load second file as above.

3. Restore first file from holding buffer by moving cursor to the desired location and unloading buffer using command U.

This method of saving text takes less time than reading the instructions. The automatic verification of the text is an important feature.

In my previous article, I discussed the problem of the CSAVE* command not being implemented on the ESF. A program for DATA I/O was included with the ESF for recording string arrays, not numerical arrays. This did not fill my need to store large (20K) numerical arrays, since string arrays for decimal numbers occupy more memory. Also, the program must be enlarged to handle the conversion to decimals and back to strings so that calculation can be made.

Two variations of the Monitor >SA method described above will work with BASIC data. The difference between the two methods is the starting address of the SAVE. Again, I heard of the first method from Paul Stuart of ASP Microcomputers.

The following BASIC memory addresses are significant to the procedures:

BEGIN BASIC:

Work Area	100
Program	105
Prog. Variables	(1B7 & 1B8)
Arrays	(1B9 & 1BA)

END BASIC:

Arrays	(1BB & 1BC)

The addresses in () are pointers to the location listed. For example, the BASIC Arrays start at the address contained in (1B9 & 1BA), and end at the address contained in (1BB & 1BC). In normal Z-80 fashion, the address bytes are in reverse order.

Saving Program & Variables In One Record

This method is convenient for programs with data that are periodically updated, or you are interrupted during a computing session, and want to later restart.

This method is not convenient

during program development, since the program cannot be changed without dumping the variables to zero. Also, it takes more tape since the program is recorded every time.

Saving a Program with Variables

1. Exit to Monitor: BYE.

2. Dump addresses 01BB and 1BC: DU 01BB 01BC. These two bytes contain the end of arrays address. Note that 01BC contains the first part of the address and 01BB the second. For example, if 01BB contains 1E and 01BC contains 62, the end address is 621E.

3. Save the file on wafer using the usual Monitor syntax: SA n 100 621E. The starting address is 100 so that all program counters will be included along with the program and variables.

Loading Program with Variables

1. Exit to Monitor: BYE.

2. Load file with the command LO n, where n is the file number.

3. Return to BASIC: PP.

Since you have just loaded the program with all variables, you MUST NOT start with a RUN or RUN nn. This would result in all variables being dumped to zero. You must also enter the program after any array dimensioning statements to avoid "REDIMENSIONED ARRAY" error.

I personally start all program menus at step 100. It is then automatic with me to restart all programs with GOTO 100.

If a program revision must be made, the arrays will first have to be saved separately using the method below, and later merged with the revised program.

Saving BASIC Arrays

The advantage of this method over saving the program and all variables together is that it takes less tape, and allows revision of the program after arrays are saved.

The disadvantage is that the address for the start of arrays must also be found in the BASIC Work Area, and the reloaded arrays must be moved to the correct address after loading to complete the merger with the program.

Saving Arrays

(Note: This method saves all arrays if there are more than one.)

1. Exit to Monitor: BYE.

2. Dump addresses 01B9 to 01BC: DU 01B9 01BC.

3. Save the file on wafer using the usual Monitor syntax: SA n (1BA & 1B9) (1BC & 1BB). The actual save will use the address for the beginning and end of arrays contained in the addresses shown in ().

Loading & Merging Arrays w/Program

1. RUN the program to reset pointers. Program should be stopped after arrays are dimensioned.

2. Exit to Monitor: BYE.

3. Dump pointers to beginning of array area: DU 1B9 1BA.

4. Load file with the command LO n, where n is the file number. Hold down CONTROL C during loading so that the file header prints on the screen.

5. Move the arrays downward in memory to the new array start address: >MO (file header start address) (file header end address) (1BA & 1B9).

6. Return to BASIC: PP.

If the program is revised by adding to it after the arrays are saved on ESF, a problem may exist if the program has grown into the previous array area. For minor program revisions, this is not a problem. When the program is restarted with RUN, the variables are dumped to zero. If the program additions use less memory than the old variable area, there will be room to load the arrays and move them downward to merge with the program.

Contrary to loading from cassette, the ESF does not have provisions to load Monitor files to a different address than from which it was saved. Also, a file may not be moved up in memory unless the new starting address lies above the existing ending address. Therefore, there is not sufficient memory, in many cases, to move large arrays upward in memory, after loading from the ESF.

If you are worried about this problem, pad your program with some dummy statements that can be removed if the program needs to be shortened after revision.

WARNING

The next section is a detailed description of how to recover from the problem caused by extensive revision of the program after the arrays are saved on the ESF. SKIP TO THE "******" below unless you have need of this detail.

If you get trapped after the arrays have been saved:

1. CSAVE the program on the ESF.

2. Go to the Monitor: BYE.

3. >LO the arrays from the ESF.

4. Set memory to cassette: SE M=C.

5. Save the arrays on cassette using above method, except the file is named rather than numbered.

6. SE M = S.

7. Go to BASIC and reload program from ESF. This is necessary since its back end was cut off by the array load in Step 3.

8. Do steps 1, 2, & 3 in basic method above, i.e. RUN program, go to Monitor, and dump pointers.

9. SE M = C.

10. Using method described on page 26 of, "A Guided Tour to Personal Computing", load the arrays higher in memory than they were recorded.

11. SE M=S.

12. Finish task by completing steps 5 & 6 in basic method above, i.e. >MO arrays and go to BASIC.

With a little planning, you will never have to do the above, but here it is just in case.

*****

The above procedures are crutches to get around the lack of a CSAVE* on the ESF. They cannot be done under program control, since they require Monitor commands.

So much for saving arrays. Now for some miscellaneous tips and facts.

Do your program development using the ESF. In addition to the speed, there are other advantages. Use two tapes that you alternate in drive 0. Make a CSAVE about every five minutes. A simple GRAPHIC D will give a one stroke CSAVE command. Alternate the tapes before each save. The ESF will default to file #1 on drive 0. This way a power failure will not cost you more than 5 minutes work.

The ESF does not add a CRC byte to the end of the program. Using the ESF for program development does not pad the end of the program with accumulated garbage that you have to laboriously delete if memory gets short.

You may store up to 127 files on a wafer. This is only practical if the files are added sequentially, and an earlier file never revised. Any slowing of the drive motor, or enlarging of the file, will cause the beginning of the following file to be overwritten.

There is a way around this problem. To save file 5, the ESF searches for the END of file 4; it does not care if file 4 has a beginning. Therefore, separate files on a tape with dummy files that are long enough to cover minor revisions and drive speed variations.

A typical sequence would be:

#1 Program/#2 Dummy/#3 Program/ #4 Dummy/.....

The dummy could be anything. One suggestion would be to (>SA n 0 1FF). This would provide a 512 byte buffer between programs.

While I have a few more goodies, I will end this article. I'm sure some of you have ideas that could make my comments look naive. Please do so by sending short items to the Sorcerer's Apprentice, showing all us ESF users how to do it better. I will even offer a blank un-certified wafer to the first one to show how to save a numerical array under program control.

ROM PAC NOTEBOOK #4

by John de Rivaz

As promised for this issue, here is the program that saves the entire array area using the Monitor routine. If an error is found on loading, instead of just stopping in the usual defeatest manner, the routine goes on to load the next file. Therefore, if you have saved two copies of a file, even if there is an error on loading the first copy, the program automatically tries to load the second. You can save as many copies as you wish depending on the reliability of your cassette system.

This routine is used to save arrays in RTL's Stocks and Shares Management program, which is under test at the time of writing. Special features of the program include inflation linking of the stocks, to show whether those gains are for real or whether they are just paper gains. This will give a graphic demonstration of the need to modify laws in capital taxation to adjust the cost price for inflation before calculating taxable gains. The program includes printout on an Epson MX80F/T of graphs of share performance on a time axis and also on one of market movements. A frivolous extra is a routine to play "tunes" and tones based on a company's price movements. The routines used to generate medium resolution graphics and to print them with the MX80 will be given in future articles in Sorcerer's Apprentice.

In order to get the POKEs for BASIC, load in the machine code with the Monitor, and then get BASIC to print out PEEKs of the required memory. POKE 260,0. In order to record a file, POKE 261,0 and call A=USR(0). To load a file, POKE 260,39 and again call A=USR(0). Before these operations, it is usual to Print a line telling the operator to switch his machine on and press any key when he has done so. This program records every single numeric array present. RTL sells a program called STR which also loads string arrays, but this is more complicated and time consuming, as every single string has to be recorded separately.

An alternative method of saving strings is used in the program on stocks and shares. Each string is split up into groups of three letters, which are coded into six digit numbers and saved in numeric arrays. The strings recorded require only ten percent of the large numeric arrays used, so this method was considered quicker than using the string loading program. A further alternative would be to record both the array area and the string area.

As a footnote to the last article, it may be noted that the file made by the assembler had to start at 4000H because the assembler overwrote it if it started at 3000H.

The routine uses code taken from the Monitor and modified to suit the required purpose.

```
EXIDY Z-80 ASSEMBLER
  ADDR  OBJECT    ST

                  0001 ;
                  0002 ;ARRAY LOAD FOR SHARE PROGRAM
  >01B9           0003 ARRAY   EQU     01B9H
  >01BB           0004 END     EQU     01BBH
  >E1A2           0005 GETIY   EQU     0E1A2H
  >E65C           0006 SAVE    EQU     0E65CH
  >E453           0007 CARRET  EQU     0E453H
  >E28A           0008 CMOTON  EQU     0E28AH
  >E2AF           0009 CMOTOF  EQU     0E2AFH
                  0010 ;FINDS GETIY AND LOADS UP
                  0011 ;MONITOR STACK FOR FIRST
                  0012 ;SAVING
 '0000  ED4BB901  0013         LD      BC,(ARRAY)
 '0004  1153E4    0014         LD      DE,CARRET
 '0007  2ABB01    0015         LD      HL,(END)
 '000A  FD7150    0016         LD      (IY+50H),C
 '000D  FD7051    0017         LD      (IY+51H),B
 '0010  FD364741  0018         LD      (IY+47H),'A'
 '0014  FD364852  0019         LD      (IY+48H),'R'
 '0018  FD364952  0020         LD      (IY+49H),'R'
 '001C  FD364A41  0021         LD      (IY+4AH),'A'
 '0020  FD364B59  0022         LD      (IY+4BH),'Y'
 '0024  C35CE6    0023         JP      SAVE
                  0024 ;here to load array
  >E6DE           0025 HEDPRT  EQU     0E6DEH
  >E759           0026 TAPWT   EQU     0E759H
  >E6A9           0027 BLKADJ  EQU     0E6A9H
  >E408           0028 CRCMSG  EQU     0E408H
  >E1BA           0029 MSGOUT  EQU     0E1BAH
  >E205           0030 CRLF    EQU     0E205H
 '0027  CDA2E1    0031 COMAND  CALL    GETIY
  >E2DA           0032 SERIN   EQU     0E2DAH
 '002A  FD363D40  0033         LD      (IY+3DH),40H
 '002E  0601      0034         LD      B,1
 '0030  CD8AE2    0035         CALL    CMOTON
 '0033  CD59E7    0036         CALL    TAPWT   ;modified GETHED
 '0036  FDE5      0037         PUSH    IY
 '0038  DDE1      0038         POP     IX
 '003A  0610      0039         LD      B,10H
 '003C  CDDAE2    0040 GETHD1  CALL    SERIN
 '003F  2843      0041         JR      Z,LOAD10-$
 '0041  DD7757    0042         LD      (IX+57H),A
 '0044  DD23      0043         INC     IX
 '0046  10F4      0044         DJNZ    GETHD1-$
 '0048  FD4646    0045         LD      B,(IY+46H)  ;CRC chkbit
 '004B  CDDAE2    0046         CALL    SERIN
 '004E  B8        0047         CP      B
 '004F  2802      0048         JR      Z,CONTIN-$
 '0051  1826      0049         JR      ERROR-$
 '0053  CDDEE6    0050 CONTIN  CALL    HEDPRT
                  0051 ;Tape will only load into array area
                  0052 ;set by BASIC, not area specified by
                  0053 ;the header on the tape.
 '0056  2AB901    0054         LD      HL,(ARRAY)
 '0059  FD5E5E    0055         LD      E,(IY+5EH)
 '005C  FD565F    0056         LD      D,(IY+5FH)
                  0057 ;DE now > end of array area
                  0058 ;HL now > start of array area
 '005F  CD59E7    0059         CALL    TAPWT
 '0062  CDA9E6    0060 LOAD8   CALL    BLKADJ
 '0065  281D      0061         JR      Z,LOAD10-$
 '0067  CDDAE2    0062 LOAD9   CALL    SERIN
 '006A  2818      0063         JR      Z,LOAD10-$
 '006C  77        0064         LD      (HL),A
 '006D  23        0065         INC     HL
 '006E  10F7      0066         DJNZ    LOAD9-$
                  0067 ;CRC check - load next if error
 '0070  FD4646    0068         LD      B,(IY+46H) ;CRC byte
 '0073  CDDAE2    0069         CALL    SERIN
 '0076  B8        0070         CP      B
 '0077  28E9      0071         JR      Z,LOAD8-$
 '0079  2108E4    0072 ERROR   LD      HL,CRCMSG
 '007C  CDBAE1    0073         CALL    MSGOUT
 '007F  CD05E2    0074         CALL    CRLF
 '0082  18A3      0075 STAGE1  JR      COMAND-$;next if error
 '0084  C3AFE2    0076 LOAD10  JP      CMOTOF
ERRORS=0000

ARRAY    01B9 BLKADJ   E6A9 CARRET   E453
CMOTOF   E2AF CMOTON   E28A COMAND   0027
CONTIN   0053 CRCMSG   E408 CRLF     E205
END      01BB ERROR    0079 GETHD1   003C
GETIY    E1A2 HEDPRT   E6DE LOAD10   0084
LOAD8    0062 LOAD9    0067 MSGOUT   E1BA
SAVE     E65C SERIN    E2DA STAGE1   0082
TAPWT    E759
```

<<< CLASSIFIED ADS >>>

$1/line $1/line

==========

FOR SALE: 32K Sorcerer Model I, W/ BASIC and Development Pacs, cassette and monitor. Contact A.L. Cavalieri, 401 Bustleton Pike, Churchville, PA 18966.

==========

ROM PAC NOTEBOOK #5

by John de Rivaz

In my last article, I mentioned the printing of graphs on the MX80F/T from Basic, as used in RTL's stocks and shares program - the one that takes inflation into account. In order to generate graphics that are similar to those used in the MX80, a short machine code program is used. This is a task easier to carry out in machine code, and the routine can be POKEd in, used, and the space then used for another machine code program later on. The graphics are small blobs that reduce the resolution of each pixel from the Sorcerer's 8 x 8 to the TRS80's 2x3. The fit isn't exact, so the middle ones are one dot smaller. However this doesn't show in the printed results.

These graphics are also used in RTL's DRAW2 program, which enables medium resolution graphic pictures to be drawn on the screen and added to Basic programs without re-typing lots of POKEs. (DRAW1 is similar but high resolution).

The routine works by counting up in register B and then testing bits to determine which numbers are to be loaded into the graphic defining memory. The IX register points to the graphic defining memory, and a subroutine "SET" is called each time this is to be set.

In order to print the screen out on the MX80F/T, the graphics characters have to be assigned new codes. In addition, a line is left at the top to allow non-printed messages to be passed to the operator. Each bit of screen memory is tested in turn. If it contains a byte whose 7th bit is set, then it is a graphic not a standard ASCII character. Therefore it has 20H subtracted so as to comply with the MX80's graphic set, and is sent to the printer. Ordinary ASCII characters are printed as usual. The program would be confused if there were any other graphics apart from TRS80 pixels on the screen, so it is left up to the operator to ensure this does not take place. The routine also inserts carriage returns at the ends of lines. The printer driver itself is a standard one which provides line feeds. If your printer is switched to auto line feed, replace lines 32-33 inclusive with NOPs.

With care, these routines can be used with other printers which have TRS80 type pixel graphics. There is, of course, the MX80F/T2 high resolution plotting feature available on these printers, but the Sorcerer is limited by programable graphic space as to how much high resolution plotting can be done on screen, and these medium resolution programs are perfectly adequate for many purposes.

Here is the printer driver routine to transfer all but the first line from the screen:

```
EXIDY Z-80 ASSEMBLER
   ADDR  OBJECT    ST

                   0001 ;
                   0002 ;This sets up 6 pixel graphics.
                   0003 ;
  '0000  2100FE    0004          LD      HL,0FE00H
  '0003  010000    0005          LD      BC,0
  '0006  3E00      0006 START    LD      A,0
  '0008  CB41      0007          BIT     0,C
  '000A  2802      0008          JR      Z,X1-$
  '000C  3EF0      0009          LD      A,0F0H
  '000E  CB49      0010 X1       BIT     1,C
  '0010  2802      0011          JR      Z,X2-$
  '0012  C60F      0012          ADD     A,0FH
  '0014  77        0013 X2       LD      (HL),A
  '0015  23        0014          INC     HL
  '0016  77        0015          LD      (HL),A
  '0017  23        0016          INC     HL
  '0018  77        0017          LD      (HL),A
  '0019  23        0018          INC     HL
  '001A  3E00      0019          LD      A,0
  '001C  CB51      0020          BIT     2,C
  '001E  2802      0021          JR      Z,X3-$
  '0020  3EF0      0022          LD      A,0F0H
  '0022  CB59      0023 X3       BIT     3,C
  '0024  2802      0024          JR      Z,X4-$
  '0026  C60F      0025          ADD     A,0FH
  '0028  77        0026 X4       LD      (HL),A
  '0029  23        0027          INC     HL
  '002A  77        0028          LD      (HL),A
  '002B  23        0029          INC     HL
  '002C  3E00      0030          LD      A,0
  '002E  CB61      0031          BIT     4,C
  '0030  2802      0032          JR      Z,X5-$
  '0032  3EF0      0033          LD      A,0F0H
  '0034  CB69      0034 X5       BIT     5,C
  '0036  2802      0035          JR      Z,X6-$
  '0038  C60F      0036          ADD     A,0FH
  '003A  77        0037 X6       LD      (HL),A
  '003B  23        0038          INC     HL
  '003C  77        0039          LD      (HL),A
  '003D  23        0040          INC     HL
  '003E  77        0041          LD      (HL),A
  '003F  23        0042          INC     HL
  '0040  0C        0043          INC     C
  '0041  79        0044          LD      A,C
  '0042  FE40      0045          CP      64
  '0044  C8        0046          RET     Z
  '0045  18BF      0047          JR      START-$
ERRORS=0000

START       0006 X1          000E X2          0014
X3          0022 X4          0028 X5          0034
X6          003A

EXIDY Z-80 ASSEMBLER
   ADDR  OBJECT    ST No.

                   0001 ;
                   0002 ;A screen printer for MX80FT
                   0003 ;with 128 x 90 resolution.  The
                   0004 ;Sorcerer's special graphics must
                   0005 ;already have been set up to get the
                   0006 ;correct screen image, and no other
                   0007 ;characters with bit 7 set must be
                   0008 ;present on screen.
                   0009 ;
  >F0C0            0010 STOP     EQU     0F0C0H
                   0011 ;allows for a top message not to be
                   0012 ;printed out.
  >F7FF            0013 SBOT     EQU     0F7FFH
  >E205            0014 CRLF     EQU     0E205H
                   0015          ORG     0C4H
  '00C4  21C0F0    0016          LD      HL,STOP
  '00C7  11FFF7    0017          LD      DE,SBOT
  '00CA  0E20      0018          LD      C,20H    ;MX80 displacement
  '00CC  1820      0019          JR      CARRET-$
  '00CE  7E        0020 START    LD      A,(HL)
  '00CF  CB7F      0021          BIT     7,A
  '00D1  2801      0022          JR      Z,OUT-$
  '00D3  91        0023          SUB     C
  '00D4  F5        0024 OUT      PUSH    AF
  '00D5  DBFE      0025 PRINT    IN      A,(0FEH)
  '00D7  CB77      0026          BIT     6,A
  '00D9  28FA      0027          JR      Z,PRINT-$
  '00DB  F1        0028          POP     AF
  '00DC  D3FF      0029          OUT     (0FFH),A
  '00DE  FE0D      0030          CP      0DH
  '00E0  2004      0031          JR      NZ,NEXT-$
  '00E2  3E0A      0032          LD      A,0AH
  '00E4  18EE      0033          JR      OUT-$
```

```
  '00E6  23        0034 NEXT    INC     HL
  '00E7  E5        0035         PUSH    HL
  '00E8  ED52      0036         SBC     HL,DE
  '00EA  E1        0037         POP     HL
  '00EB  C8        0038         RET     Z
  '00EC  10E0      0039         DJNZ    START-$
  '00EE  3E0D      0040 CARRET  LD      A,0DH
  '00F0  0641      0041         LD      B,65D
  '00F2  2B        0042         DEC     HL
  '00F3  18DF      0043         JR      OUT-$
ERRORS=0000

CARRET      00EE CRLF       E205 NEXT       00E6
OUT         00D4 PRINT      00D5 SBOT       F7FF
START       00CE STOP       F0C0
```

MAGIC SQUARE

by Emiliano De Laurentiis

(This article is reprinted from the Oct/Nov, 1980 issue of the S.U.N.)

What is a magic square? A magic square is a square matrix of odd numbered sides, where each cell of the matrix holds an integer from 1 to n, where n is the number of cells in the matrix. Furthermore, all the columns, all the rows, and all the diagonals must add to the same number. An example of a 3x3 magic square is presented below. The following program will produce any nxn magic square. I present it here as an illustration of the algorithm (called a production system in the program) which will produce such a square. The size of the magic square you produce is only limited by the size of your Sorcerer's memory.

```
3 x 3 MAGIC SQUARE

 8  1  6  =  15
 3  5  7  =  15
 4  9  2  =  15
--------
15 15 15   diagonals=15
```

PROGRAM LISTING

```
1 REM *** This program will draw any size magic square ***
2 REM ***** By Emiliano De Laurentiis *****
3 REM ================================================
4 PRINT CHR$(12)
5 INPUT "Size of proposed magic square";M:DIM C(M,M)
6 PRINT CHR$(12):PRINT:PRINT
8 :
9 REM N=No. in square; X=Vertical Axis; Y=Horizontal Axis
10 N=0:X=1:REM Initialize values
15 Y=INT(M/2+.5):REM find middle position of co-ordinates (1,Y)
16 REM solution must start at co-ordinates (1,Y) where Y is center
20 :
30 N=N+1:C(X,Y)=N:REM set value N at position X,Y
35 :
36 REM ==========PRODUCTION SYSTEM ==========
40 IF N=M*M THEN 100
50 IF X-1 < 1  AND Y+1 > M THEN X=X+1:GOTO 30
60 IF X-1<1 THEN X=M: Y=Y+1: GOTO 30
70 IF Y+1 > M THEN Y=1:X=X-1:GOTO 30
80 IF C(X-1,Y+1) <> 0 THEN X=X+1:GOTO 30
90 X=X-1:Y=Y+1:GOTO 30
95 :
98 REM print matrix
99 :
100 FOR A=1 TO M: FOR B=1 TO M
105 D=D+C(A,B)
106 IF C(A,B)<100 THEN PRINT " ";
107 IF C(A,B)<10 THEN PRINT " ";
110 PRINT C(A,B);" ";
120 NEXT B:PRINT "=";D:D=0:NEXT A
125 :
126 FOR I=1 TO M*6:PRINT "-";:NEXT I:PRINT
127 :
130 FOR B=1 TO M:FOR A=1 TO M
140 D=D+C(A,B)
142 NEXT A
145 IF D<100 THEN PRINT " ";
147 IF D<10 THEN PRINT " ";
150 PRINT D;" ";:D=0:NEXT B
155 PRINT ""
160 PRINT
170 FOR A=1 TO M:B=1
180 D=D+C(A,B)
190 B=B+1: NEXT A: PRINT "DIAGONAL '\'=";D;
191 PRINT "    ";
192 D=0
193 FOR A=1 TO M: B=M
194 D=D+C(A,B)96 B=B-1: NEXT A: PRINT "DIAGONAL '/'=";D
200 PRINT CHR$(17):RUN 5:REM Go home and start run at line 5
210 REM                      RUN COMMAND CLEARS MEMORY>
220 END
```

NOTES FROM RCPM-SORCERER

by The Sysop

++++++

Exidy CP/M Modification

Does anyone know how to modify Exidy CP/M ver. 1.42/3 from Micropolis Mod I to Mod II operation? So far I've found that changing address 0123EH in MOVCPM from 47H to 9BH and changing address 01B4H in FORMAT from 26H to 4EH accomplishes most of the change. I still need to modify something to get CP/M to write to the whole disk. R.D. HAUN

++++++

Joystick Standard

Arrington's standard, described in the January Sorcerer's Apprentice Newsletter (3.1), doesn't permit concurrent operation of the keyboard and joystick #2. Movement while firing is also prohibited. I believe this should be changed! Using the keyboard should make it replace **only** joystick #1, leaving #2 fully functional. Since the keypad is on the right, joystick #1 should be the right stick and #2 should be the left one. I believe games have more realism if the "gun" doesn't have to stop while it's firing. But, Arrington's standard has FIRE and LEFT/RIGHT controls sharing the same pins; they can't be used together. Since games make far more use of LEFT/RIGHT than they do of UP/DOWN, the FIRE button should be moved to the UP/DOWN pins! Then, you can fire on invaders as you keep sliding across the screen, and shoot asteroids as you swing your ship around. G. King

++++++

Monitor Memory Test Show

The Monitor's TEST command can produce an interesting display. If you >TEST screen RAM, you'll see the patterns it uses. To get things going, type in these two lines:

```
SE O=E044       (this turns off
                 screen output)

TE F080 F800 C (this runs the show
                continuously)
```

Impress your friends with the next best thing to blinking lights! G. King

++++++

Sorcerer Basic Converter

Bob, what is needed on the RCPM is an interchange of Sorcerer Basic programs. Would you agree? I've got a nifty CP/M program that converts ASCII Basic to Sorcerer compacted Basic and vice-versa. It's great for working with your E-Basic programs, Word Processor created programs, etc. Is this something that would be good to have on your system? Dan Conway

I told Dan we would be glad to have his program on the system. We now have SORBAS.OBJ, and DOC on drive A:. Sysop

(Editor's Note: In his article, "Restoring Lost Link Addresses", found in issue 4.3, James Canning refers to the 'TOSCA' program found in the May, 1980 issue of the S.U.N. Since many of you never subscribed to this publication, and back-issues are no longer available, that article is being reprinted below. We will also be reprinting other articles, that may be of interest to you.)

TAPE OUTPUT SCANNER

by Jim Burns

(This article is reprinted from the May, 1980 issue of the S.U.N.)

The following is a relocatable routine that acts as a cassette image dump, bypassing exclusion of headers, CRC bytes, etc. and which does not bomb out in the middle of the tape because it has one block that doesn't CRC check properly.

```
11 zz ww          LD DE,wwzz   ;finish address
21 xx yy          LD HL,yyxx   ;start address
CD 8A E2          CALL E28A    ;call CMOTON
CD DA E2  LOAD    CALL E2DA    ;call INTAPE
C8                RET Z        ;on ESC, CNTRL-C, etc.
77                LD (HL),A
23                INC HL
E5                PUSH HL
B7                OR A         ;clears carry
ED 52             SBC HL,DE
E1                POP HL
C8                RET Z        ;fin add= strt add
18 F2             JR LOAD      ;LOAD-$ in some
                                assemblers
```

For a more sophisticated version, that does strip headers, and CRC bytes for SAve and CSAVE files (not CSAVE* files), see the next listing called, "TOSCA" (Tape Output Scan). TOSCA is useful for loading BASIC, or machine language files into the right position in memory. For BASIC files, you will have to manually load the end address of the file (which should be at the end of three consecutive zeroes which mark the EOF for the BASIC program) in memory locations 1B7-1B8, 1B9-1BA, and 1BB-1BC. Nine times out of ten, the header information in front of each BASIC statement will be intact and you can LIST the file to find out where the line with the CRC error is, and correct it by typing in the line correctly.

If you get the line numbers out of sequence (like a 65404 in the middle of the 1000's), you probably will be able to type that line over again with the correct line number. However, if you get garbage on your listing, then the link bytes have been damaged. These are the first two bytes after the zero at the end of each BASIC statement. The third and fourth bytes after the zero are the line number in hex. If they are damaged, you will have to trace the links to the very first one at 01D5-01D6, making sure they point to the next link in the sequence, that each is preceded by a zero, and that no other zeroes occur in the core area occupied by the BASIC file. The last link points to the 00 00 00 EOF mark.

For more information on header statements, memory pointer locations and functions, and tape file formats, get a copy of the, "**Software Internals Manual for the Sorcerer**", put out by Quality Software.

The CSAVE* format is not explained in that manual. The CSAVE* format is: four D2H's followed by the core image of the values in the array. It does not have any information of it's dimensions, block length, or name. That is why the array has to be dimensioned **before** you CLOAD* it. Since the tape image contains no information on the array's name, the first array you position to on the tape, will be loaded into the array specified in the CLOAD* command. NOTE: Since RUN re-initializes the variable space, you will have to use GOTO (line number) to start the program (BASIC does patch it's pointers to remember it made space for the loaded array).

Some final notes on the two drivers:

NOTE 1: The call to CMOTON is necessary even if you don't use motor control for your cassette. Exidy's procedure at the end of tape handling, or errors (at least in the Monitor), is to call CMOTOF. One of CMOTOF's actions, is to reset the UART to 300 baud regardless of the SET parameter. CMOTON sets the UART to the desired baud rate.

NOTE 2: The first byte dumped from tape using these routines, may be shifted left one or two bits, as the UART may not be in sync yet. Example: The first D2H (1101 0010), in the first array I dumped, came out A4H (1010 0100). This is the same problem people have had with the serial printer driver in the, "**Technical Manual**". It works fine for block output from the Sorcerer, but gets out of sync (sporadically) with keyboard input. Exidy software only verifies that part of the leaders are correct, so this doesn't cause any problems.

One parting question, and then the second listing. CLOADG has no more effect that CLOAD in my Sorcerer. Is anyone using CLOADG for autoload/execute? And, oh yes, I appreciate the Monitor listing printed in the S.U.N. I wrote a Z80 disassembler in BASIC, but alas, no printer! Even so, I had transcribed the video output for the first 1K by hand. Along came the S.U.N. with the other 3K to placate my weary hand and brain.

TOSCA- A Routine to Scan Tape Output

```
(06 nn            LD B, N)     ;optional for motor
                                 control
CD 8A E2          CALL E28A    ;call CMOTON
06 02             LD B,2       ;for both 101 byte
                                 hdrs
CD 59 E7          CALL E759    ;call HEADERCK
10 FB             DJNZ,-3      ;to CALL HEADERCK
21 xx yy          LD HL, yyxx  ;start address
11 zz ww          LD DE, wwzz  ;finish address
CD DA E2  LOAD    CALL E2DA    ;call INTAPE
C8                RET Z        ;(ESC) returns to
                                 caller
77                LD(HL), A
23                INC HL
E5                PUSH HL
B7                OR A
ED 52             SBC HL, DE
E1                POP HL
C8                RET Z        ;done
10 F2             DJNZ LOAD    ;load 256 bytes
CD DA E2          CALL E2DA    ;dummy CRC read
18 ED             JR LOAD      ;repeat
```

Jim Burns, 2160 Market St. Rm. 45
San Francisco, CA, 94114

PSEUDO 'PRINT USING'

by Neil Barbu

The routine listed below is one that I have used to overcome the Sorcerer's lack of 'PRINT USING'.

My entry routines are usually from cassette but I have used keyboard input and loops, then branch from line 7190 to a columnar printout.

```
5 REM - SAMPLE ENTRY ROUTINE
10 LET C=.25
20 FOR Q=1 TO 10
30 LET A=INT(100*(Q*C)+.5)/100
35 REM - NORMAL PRINTOUT
40 PRINT A;
45 REM - FORMATTED PRINTOUT
50 GOSUB 7000
60 NEXT Q
70 STOP
7000 REM SUBROUTINE - FORMATTED PRINTOUT
7010 LET A$=STR$(A)
7020 LET L=LEN(A$)
7030 IF L=2 GOTO 7130
7040 IF MID$(A$,(L-1),1)="." GOTO 7100
7050 IF MID$(A$,(L-2),1)="." GOTO 7170
7060 GOTO 7130
7100 LET B$="0"
7110 GOTO 7150
7130 LET B$=".00"
7150 LET A$=A$+B$
7170 LET L=LEN(A$)
7190 PRINT TAB(30-L);A$
7199 RETURN
```

The Making of, "DUEL: A DOGFIGHT IN SPACE"

by Bill Boucher

In response to Don Gottwald's review of, "DUEL: A Dogfight In Space", this article describes the techniques I used to produce the high-resolution displays in the game. From the start, I wanted to write a video game that would fully use the Sorcerer's graphics capabilities. Realistic action, and an overall good look were my prime considerations.

DUEL is a game for two players, where each player controls a spaceship, and earns points by blowing up his opponent's ship. The players, therefore, do not compete with the machine, but with each other. The ships can rotate right, or left in 22 1/2° degree increments, accelerate forward, and fire missiles, with each ship having up to five missiles on the screen at one time. Because there is no gravity or friction, once your ship is moving, there are only two ways to stop it; you can apply thrust in the reverse direction, or get blown up by your opponent. A ship encountering the edge of the screen wraps around to the other side. A space-probe traverses the screen at random intervals. Crashing into it, costs a player five points, while destroying it, earns the player a five-point bonus. The adjustable parameters in each game, rotate speed, ship speed, missile speed, and firing rate, can change the nature and strategy of the game. For instance, with fast ships and slow missiles, the ships can outmaneuver enemy missiles, which is not possible with an opposite set up. The ships, positioned on the full 240 x 512 pixel grid, move smoothly and realistically, and because the background starfield remains intact as the ships pass over it, there is no visual evidence of the character grid. The explosions are detailed, and animated with flying debris.

Because of the complexity of the program, it would be impractical to describe the workings of the program in every detail. So, instead I will give a general description of the key routines. This, I hope, may give the reader some programming ideas to pursue.

The program redefines the graphics character used for each ship every time it positions a ship on the screen, thus making possible the preservation of the background stars, and the bit position of the ship. Whenever the program moves a ship on the screen, the following takes place:

1. From the desired location on the screen, the program copies into memory a portion of the starfield (a 3 x 3 block of the characters).
2. The program positions the ship image, stored as a 2 x 2 block of characters, within the 3 x 3 starfield block. It does this by shifting the ship image bytes to the correct bit position and ORing them into the copy of the starfield.
3. The program loads the resulting image into the correct area of the user-character set.
4. The program then puts the redefined graphics characters onto the screen.

With this technique, the location of the ship is not hindered by character boundaries; the ship's center can be placed on any one of the 240 x 512 pixels on the screen. The code that accomplishes this task is the heart of the program. It demonstrated to me how software can squeeze the most out of a given piece of hardware.

The explosions are accomplished in a similar way. There are two nine-frame animated explosions in the game. The program successively ORs the nine images of the explosion into the starfield, and then puts them on the screen, resulting in a starfield continually preserved behind the explosion.

The basic construction of the program is a big loop of several subroutines. Each function, such as ship movement, missile advancement, explosions, etc., has its own subroutine. This, however, created a timing problem because the time to complete the loop varied as the amount of subroutine activity varied. To solve this problem, the program uses the serial output port as a timer: at the beginning of the loop, it sends a byte to the serial output port; at the end, it waits for the "transmit buffer empty" bit to change before returning to the start of the loop. Thus, a constant loop time of about 30 milliseconds was achieved.

As a final note, I want to thank Dave Grigsby who provided some important assistance in the creation of the high-resolution ship positioning routines, solved my timing problem, and developed the random-number generator used to form the starfield. Thanks also to Ron Julian of Dayspring Computer Enterprises who conceived the game's introduction.

TIPS

by Kent Regal

1. TIP - When inputting many print statements in a BASIC program, it is not necessary to use closing quotation marks at the end of each statement. Closing quotes are needed at the end of the last PRINT statement only. Such as:

```
400 PRINT"Words
405 PRINT"that you
410 PRINT"want to print
415 PRINT"such as text in a
420 PRINT"program that needs
425 PRINT"multiple lines "
```

   This method saves the hassle of putting in the extra quotes and probably saves program space as well.

2. TIP - The Exidy Word Processor instructions for the command mode commands imply that they are to be lower case letters. The commands can be in either lower or upper case on my system.

3. TIP - With all due respect to Arkay Engravers, Inc, I've chosen to use small embossed plastic label letters. The initials I us, remind me of what key does what.

4. TIP - When using random numbers, the binary string (i.e. 1, 2, 4, 8, 16, 32, etc.) produces the same random number sequence. This can be a problem when trying to get a different strings of random numbers.

5. REQUEST - I'd like to obtain S.A. Volumes 1 & 2 on a loan basis since these are now out of print. I would pay postage both ways and guarantee they would be returned in the same condition as received, by sending them in a heavy duty mailer envelope.

   Kent Regal, P.O. Box 478,
   Marshall, WI 53559 ●

A SUPER PRINTER INTERFACE

by Ernest E. Bergmann

I shall describe how I have interfaced the Epson MX-80 printer (any Centronics compatible printer could have been used). The software should be of particular interest because it can be installed for use with both the BASIC and Development ROM Pacs. The software provides facilities to turn on/off the printing of program output, to stop and resume the program's output, to type the current contents of the video screen (without disturbing the display or the program), to send special control characters to the printer (but not to the program!), and, even, to jump to the Monitor at will.

I shall divide my description into several parts: how to use the interface; how the software works; the wiring of the MX-80 printer; and, making your own cassette of the software.

Using The Interface

Turn on the computer and printer and enter the Monitor. If no ROM Pac is inserted, you will be in the Monitor to begin with. If you are in BASIC, type: BYE. If you are using the Development Pac, "execute" E003.

From the Monitor, issue the command, "LOG", and read in the cassette that contains the interface software. Since the file is very short (less than one "page"), the program loads in only a few seconds. One is back again in the Monitor (but indeed the interface has been installed). If you need to return to the ROM Pac, you should now use the Monitor command, "PP".

You could do what you did before, and not see any difference.

Listing

Use a control-P to turn the printing on/off of the output that is going to the video screen. For example, suppose you are using BASIC and want to print the listing of your program. Type: LIST <control-P>, carriage return <CR>. The listing that appears on the screen is appearing also on the printer. When the listing is complete, you may wish to type another control-P to prevent further output from going to the

printer.

Starting & Stopping

Pressing the key combination, control-S, will stop output. Your program will resume activity as soon as another key is struck (or another control-S). This feature is a great convenience even without a printer; it sure beats having to keep the RUN/STOP key constantly pressed to freeze a BASIC program!

During the pause generated by the control-S, you can use other features of the interface software, such as the control-P or typing the current screen, see below:

Screen Dump

It often happens that I want to record on paper what is on the screen as accurately as possible; to take a "snap shot" of the screen. I need to type a control-T, which causes the characters at each location on the screen to be examined in turn, and output to the printer.

Of course, graphic characters will not be handled properly by the printer, but what can you expect? The eighth bit, is not sent, to preserve compatibility with Centronics printers. Graphic symbols are not standardized in any case.

Exiting

A control-^ causes a jump to the Monitor warm entry point (0E003H). It is useful in places where continuing to execute the program would be painful. For example, the editor in the Development Pac does not have a "fast exit", so I type control-^ and then issue the "PP" command to re-enter the ROM Pac.

Special Codes

There are quite a few special character sequences that can be sent to the MX-80 printer to achieve special effects such as condensed, expanded, emphasized, and double printed letters. Sequences also exist to control vertical (line) spacing. These sequences are detailed in the operation manual for the printer.

Usually, I want to send these sequences to the printer but not to the program. To do so, I type a control-E. A "blob" should appear on the screen as a "prompt", indicating a readiness for input of the special character sequence that is destined for the printer only. Then I type in the sequence; it can be zero to five characters in length and may include any control codes except control-M [a carriage return] or a delete [shift-RUB]. The sequence is sent to the printer when either a carriage return is keyed or five characters have been entered. If I have mistyped the sequence, I can redo it before it is sent to the printer by keying a shift-RUB.

For example, I might wish to type a listing that is more than 80 columns wide, so I choose to use condensed type that supports lines up to 132 columns wide. According to the MX-80 manual, the printer should be sent a control-O, so I type control-E, control-O and return. That is all that is needed!

Because typing "blind" is inconvenient and error prone, I have written the software so that the characters that are to be sent to the printer are shown on the screen as they are keyed in. Since the control characters are usually not displayed, all control codes are converted for display purposes to an up-arrow (^) and a printable character. Thus, the control-O of the above example would be represented on the screen as (^).

Software

I wrote the software so that the machine code would fit exactly in memory between 3BH and 0FFH on page 0. The software uses memory also from 33H to 37H, inclusive, as a buffer that is used for the "special codes", described above. This choice was made to be compatible with space that is normally not utilized in the Sorcerer when the BASIC or Development Pac is in use. To provide so much function in so little space, I needed to "squeeze" the code as much as possible; readability and modularity have had to be compromised somewhat.

For the software to be useable, two routines, CHIN (character in) and CHOUT (character out), need to be patched into the Sorcerer's I/O. Although this could be achieved by:

SET I=8A

SET O=65

I was able to squeeze in the routine, IOINIT (I/O INITIALIZATION) , which achieves the same results automatically.

It is a commendable feature of the original Sorcerer software to provide patch points for the I/O. Any software that is written for the Sorcerer should call locations 0E009H (RECEIVE) and 0E00CH (SEND). Such software then has its I/O automatically patched by the Sorcerer Monitor to the current input and output devices.

By having control over I/O, we can use all the "services" described above. Let us see how it is done:

CHIN

This character input routine must behave in a manner analogous to other Sorcerer input routines. If no character is available yet, it should return with the Z-flag set. If a character is, or has been, received, the Z-flag should be reset, and the character should be returned in the A register.

CHIN begins by calling the KEYBRD routine in the Monitor. If no character is pressed then KEYBRD returns with the Z-flag set and CHIN returns also with the Z-flag set. But if a character is pressed, KEYBRD returns it in the A register and CHIN proceeds to see if the character might be one of the special

(continued on page 99)

EIGHT DIGIT FINANCIAL TABULATOR

by Frank Voss

Sorcerer's ROM Pac BASIC, having only 6 significant figure accuracy, is a severe handicap if one wants to utilize it for financial programming. To maintain dollars and cents accuracy, the maximum value that can be accommodated is $9999.99. Hardly useful for many purposes in these times of increasing monetary figures. Through the use of some programming techniques, I have been able to expand the range of financial figures that can be accommodated to 8 significant digits, thus permitting the maximum amount to be increased to $999999.99 with full dollars and cents accuracy.

The method that I've used to achieve 8 digit accuracy for financial figures is as follows:

a) INPUT of an amount to the program is as a string instead of a numerical value. [A$]

b) The A$ is separated into its dollars and cents components. [D$ & C$]

c) The D$ & C$ are converted to their value amounts. [D & C]

d) The appropriate mathematical functions are performed.

e) The resultant values are converted back t string functions. [TD$ & TC$]

f) The TC$ function is formatted for missing 0's.

g) The final assembled string functions are made available for display or printing. [TD$;".";TC$]

I've prepared 2 programs as part of this article.

1) The FINANCIAL TABULATOR SUBROUTINE is for individuals who would like to write a more expanded program.

2) The FINANCIAL TABULATOR is a formatted, working program that may be used with a printer. (Output to Centronics bus.)

The programs as presented here will add and subtract amounts and yield a resultant total. Multiplication and division of financial figures can be done using the same method I've outlined above and is left to be done by the reader.

```
0 REM          FINANCIAL TABULATOR SUBROUTINE
1 REM
2 REM               Frank Voss          1982
3 REM               Box 43
4 REM               Wyandotte, MI.      48192
5 REM
6 REM  This program is to be used as a subroutine for a larger
7 REM  tabulating program.  It permits 8 digit accuracy for
```

```
8 REM  financial data.  This means that values to $999999.99
9 REM  can be accommodated without error.  Subtractions are
10 REM entered as a negative number.  The total is obtained
11 REM by entering T for the amount.
12 REM
20 INPUT"  ENTER AMOUNT";A$    :REM  Enter AMOUNT as a string
250IF LEFT$(A$,1)="T" OR LEFT$(A$,1)="t" THEN 150 :REM Do Total
30 LET L=LEN(A$)
35 LET D$=LEFT$(A$,L-3         :REM  Extract dollars from string
40 LET C$=RIGHT$(A$,2)         :REM  Extract cents from string
45 LET D=VAL(D$):C=VAL(C$)     :REM  Convert to numerical values
50 IF LEFT$(A$,1)="-" THEN C=-C
55 L=LEN(D$)
60 PRINTTAB(40-L)"$";D$;".";C$
80 LET TD=TD+D:TC=TC+C         :REM  Do mathematical operations
85 IF TC>99 THEN TC=TC-100:TD=TD+1:GOTO 100
86 IF TC<-99 THEN TC=TC+100:TD=TD-1
87 IF TC<0 AND TD>0 THEN TC=TC+100:TD=TD-1
88 IF TC>0 AND TS<0 THEN TC=TC-100:TD=TD+1
100 GOTO 20                    :REM Loop for another input
150 LET TD$=STR$(TD):LET TC$=STR$(TC)  REM  Back to strings
160 IF TC=0 THEN TC$="00"      :REM Format TC$ for 0's
162 IF TC>0 AND TC<10 THEN TC$="0"+TC$
163 IF TC<0 AND TD=0 THEN TD$="-0"
165 LET TC$=RIGHT$(TC$,2)
180 PRINTTAB (30)"-----------------":REM Display total
185 PRINT" TOTAL AMOUNT =";
190 L=LEN(TD$)
195 PRINTTAB(40-L);"$";TD$;".";TC$
200 RETURN                     :REM  BACK TO MAIN PROGRAM


0 REM            FINANCIAL TABULATOR
1 REM
2 REM            Frank Voss     1982
3 REM            Box 43
4 REM            Wyandotte, MI.48192
5 REM
6 REM A formatted, working program for use in tabulating
7 REM financial figures to $999999.99
8 REM Subtractions are entered as a negative number.
9 GOSUB 250
10 PRINTCHR$(17):PRINT
15 N=N+1
20 PRINTCHR$(17):PRINT
25 INPUT"  ENTER AMOUNT";A$
26 IF LEFT$(A$,1)="T" OR LEFT$(A$,1)="t" THEN 150
27 IF LEN(A$)<3 THEN GOTO 225
30 LET L=LEN(A$)
35 LET D$=LEFT$(A$,L-3):LET C$=RIGHT$(A$,2)
40 LET D=VAL(D$):C=VAL(C$)
45 IF LEFT$(A$,1)="-" THEN C=-C
50 L=LEN(D$)
60 FOR X=1 TO N:PRINTCHR$(10);:NEXT X
70 POKE M,147:POKE M+1,233          :REM Output to printer
71 PRINTTAB(25-L)"$";D$;".";C$
72 POKE M,27:POKE M+1,224           :REM Output to video
80 LET TD=TD+D:TC=TC+C
85 IF TC>99 THEN TC=TC-100:TD=TD+1:GOTO 100
86 IF TC<-99 THEN TC=TC+100:TD=TD-1
87 IF TC<0 AND TD>0 THEN TC=TC+100:TD=TD-1
88 IF TC>0 AND TD<0 THEN TC=TC-100:TD=TD+1
100 PRINTCHR$(17):PRINT
105 PRINTTAB(30)" "
110 PRINTCHR$(17):PRINT
120 GOTO 15
150 LET TD$=STR$(TD):LET TC$=STR$(TC)
160 IF TC=0 THEN TC$="00"
162 IF TC>0 AND TC<10 THEN TC$="0"+TC$
163 IF TC<0 AND TD=0 THEN TD$="-0"
165 LET TC$=RIGHT$(TC$,2)
170 FOR X=1 TO N:PRINTCHR$(10);:NEXT X
175 POKE M,147:POKE M+1,233         :REM Output to printer
180 PRINTTAB(17)"-----------------"
185 PRINT" TOTAL AMOUNT = ";
188 L=LEN(TD$)
190 PRINTTAB(25-L;"$";TD$;".";TC$
195 PRINT:PRINT:PRINT
200 POKE M,27:POKE M+1,224          :REM Output to video
201 INPUT"  Press RETURN for another tabulation."Q$
205 TC=0:TD=0:N=2:GOTO 10
210 END
225 PRINTCHR$(23);"  IMPROPER ENTRY, DO AGAIN !"
230 FOR T=1 TO 500:NEXT T
235 PRINTCHR$(23);TAB(35)" "
240 PRINTCHR$(23);:GOTO 20
250 N=2
260 M=PEEK(-4095)*256+PEEK(-4096)
270 IF M>32767 THEN M=M-65536
275 M=M-47
280 RETURN ©
```

(SUPER PRINTER cont'd from p. 98)

"commands" which start with control-P, control-T, control-^, or control-E. If the character is one of these four possibilities, CHIN returns with the character in A and the Z-flag reset.

Now let us consider what happens if one of the four special control characters had been keyed as four special cases:

If control-P (printer on/off) is pressed, control passes to the code labelled CP. CP examines a two byte piece of memory called OPATCH ("output patch") that should contain either the address 0E01BH (which is the Monitor's video output routine, that I have named NOLIST) or the address 0E993H (which is the Monitor's "Centronics driver", that I have named LIST). Whichever address is found, it is replaced with the other address at OPATCH. Having switched from LIST to NOLIST, or vice versa, CP returns control to CHIN to get the next character (control-P "doesn't count"). In order to preserve the HL register pair, it was necessary to save it with PUSH HL; the corresponding POP HL is to be found at CHINM1 ("CHIN-1).

If a control-T had been selected (to type all of the screen), control would have passed to CT. Since this code has the potential of messing up both the BC and HL register pairs, they are both pushed upon entry to CT. CT ends by running into CHINM2 ("CHIN-2") which pops these registers and becomes CHIN to fetch another character.

First, CT sends a form feed character (you may which to save paper by substituting a carriage return, 0DH, here at location 75H), to the printer and then enters a pair of nested loops. The innermost loop (lines 74-7), reads and prints 64 consecutive memory locations (one line) of the screen RAM. The outer loop (lines 73-80), invokes the inner loop and then prints a carriage return; it repeats 30 times, the number of lines on the screen.

If a control-^ was pressed, CHIN jumps to 0E003H, the warm reentry point of the Monitor.

Last, if a control-E is pressed (to send special character sequences to the printer), control passes to CE. First, CE saves the registers that would otherwise be messed up. Then (starting at CE1, line 95), a "blob" character is put on the screen to prompt the user to enter the special sequence. This sequence will be placed in a buffer first so that I can correct an error before it reaches the printer. The address of the buffer is placed in HL. B contains the maximum length of the buffer (which is 5) and C contains an ASCII carriage return. At CC (line 99) a loop is established which calls KEYBRD until a character is received. Ignoring for the moment that the character might be a carriage return or delete (shift-RUB), the character is placed into the buffer (line 105). By comparing the character with a blank (line 107), CE determines

whether or not the character is a control code. If not, a jump is made to NCTRL which echoes the character to the video display and loops back to CC (unless this was the fifth character placed in the buffer). If the received character had been a control code, then lines 109-113 are carried out; a '^' is then echoed, and the character is supplemented by 40H which converts the control code to a printable character.

If, during input, a 7FH (RUB) is encountered, then a jump back to CE1 is performed to start on filling BUFFER all over again. If a CR is encountered, it indicates that we finished filling the buffer and want to transmit its contents to the printer; a jump to FIN (line 117) accomplishes that.

A useful entry point, that is used by the above software, is PRINT (line 49), which is the routine that outputs the character in A to the printer only; all registers are preserved. No such routine exists in the Sorcerer's Monitor software. The closest semblence is at LIST (0E993H):

```
E993  F5       PUSH AF
E994  CD1BE0   CALL 0E01BH
                    ;video out
E997  ...
```

Although one can call LIST, one cannot call 0E997H successfully in order to drive the printer exclusively, because the stack level was altered at 0E993H. Too bad the CALL and PUSH instruction had not been interchanged!

CHOUT

CHOUT has the responsibility of outputting the character in the A register to the video and, if in the "listing state", should drive the printer as well. Further, it should check the keyboard for a control-S (to immobilize output). In all cases, all registers must be preserved.

CHOUT starts by calling CTRLS, which does not save the accumulator or flags. CTRLS has the specific task of scanning the keyboard and holding up the works whenever a control-S is sensed. CTRLS returns almost instantly, if no control-S is pressed.

After control is returned from CTRLS, a jump (line 64) to LIST (0E993H) or to NOLIST (0E01BH), takes place, completing the function CHOUT. The jump address is at OPATCH, which is over-written by CP (when a control-P is pressed) alternatively with the two possible addresses. This method, known as "improper coding", is chosen because we need to conserve memory space at all cost.

CTRLS

Obviously, CTRLS (line 130) should be described further. I form a control-S by first pressing the CTRL key, and then, the S-key, without releasing the first key. CTRLS outputs first a 3 to the keyboard scanning port (0FEH), and tests bit 2 from that same port. If that bit is set, then the S-key is not depressed and CTRLS returns (line 135). Otherwise, a 0 is output and the same bit 2 is tested. If the bit is set, the CTRL key hasn't been depressed and

```
ADDR   0  1  2  3   4  5  6  7   8  9  A  B   C  D  E  F

0030:                                    2A  00 F0 11 D1
0040:  FF 19 97 36  65 23 77 23  36 8A 23 77  C9 3E 0D F5
0050:  C3 97 E9 E5  21 93 E9 3A  6B 00 BD 20  03 21 1B E0
0060:  22 6B 00 18  24 F5 CD E3  00 F1 C3 1B  E0 E5 C5 21
0070:  80 F0 0E 1E  3E 0C CD 4F  00 06 40 7E  23 CD 4F 00
0080:  10 F9 CD 4D  00 0D 20 F1  C1 E1 CD 18  E0 C8 FE 10
0090:  28 C1 FE 14  28 D7 FE 1E  CA 03 E0 FE  05 C0 E5 C5
00A0:  3E 7F CD 1B  E0 21 33 00  01 0D 05 CD  18 E0 28 FB
00B0:  B9 28 19 FE  7F 28 E9 77  23 FE 20 30  09 C6 40 F5
00C0:  3E 5E CD 1B  E0 F1 CD 1B  E0 10 E0 79  CD 1B E0 3E
00D0:  0A CD 1B E0  4D 21 33 00  7D B9 28 AC  7E 23 CD 4F
00E0:  00 18 F5 3E  03 D3 FE DB  FE E6 04 C0  3E 00 D3 FE
00F0:  DB FE E6 04  C0 CD 18 E0  28 FB CD 8A  00 28 FB C9
```

Dump of the machine code for the MX-80 printer interface.

```
EXIDY Z-80 ASSEMBLER
ADDR    OBJECT  ST

                0001 ;************************************
                0002 ;*                                  *
                0003 ;* MX-80 INTERFACE by E.E. BERGMANN *
                0004 ;*       May 24, 1981               *
                0005 ;*                                  *
                0006 ;************************************
                0007          PSECT   ABS
                0008 ;
>007F           0009 BLOB     EQU     7FH
>0033           0010 BUFFER   EQU     33H       ;RST6+3
>0005           0011 BUFLEN   EQU     5
>0500           0012 BUFB     EQU     500H      ;BUFLEN*100H
>000D           0013 CR       EQU     0DH
>E018           0014 KEYBRD   EQU     0E018H
>00FE           0015 KEYPRT   EQU     0FEH
>000A           0016 LF       EQU     0AH
>000C           0017 FORMFD   EQU     0CH
>E993           0018 LIST     EQU     0E993H    ;VIDEO+MX80
>E01B           0019 NOLIST   EQU     0E01BH    ;VIDEO ONLY
>F080           0020 SCREEN   EQU     0F080H
>E003           0021 WSTART   EQU     0E003H    ;WARM RESTART
                0022 ;
                0023 ; USEFUL ENTRY POINTS:
                0024          GLOBAL  CHOUT
                0025          GLOBAL  CHIN
                0026          GLOBAL  IOINIT
                0027          GLOBAL  PRINT
                0028 ;
                0029          ORG     3BH       ;AFTER RST7
                0030 ;
>003B           0031 IOINIT   ;WILL PATCH THE I/O WITH
                0032          ;CHIN AND CHOUT
                0033          ;IN THE INTERESTS OF SPACE,
                0034          ;IOINIT DOES NOT SAVE REGISTERS
003B    2A00F0  0035          LD      HL,(0F000H)
003E    11D1FF  0036          LD      DE,0FF92H+3FH
0041    19      0037          ADD     HL,DE
0042    97      0038          SUB     A         ;PAGE 0
0043    3665    0039          LD      (HL),CHOUT
0045    23      0040          INC     HL
0046    77      0041          LD      (HL),A
0047    23      0042          INC     HL
0048    368A    0043          LD      (HL),CHIN
004A    23      0044          INC     HL
004B    77      0045          LD      (HL),A
004C    C9      0046          RET
                0047 ;
004D    3E0D    0048 CRPR     LD      A,CR
004F    F5      0049 PRINT    PUSH    AF        ;MX-80 ONLY
0050    C397E9  0050          JP      LIST+4
                0051 ;
0053    E5      0052 CP PUSH  HL      ;CTRL-P
0054    2193E9  0053          LD      HL,LIST ;TOGGLE PRINT
0057    3A6B00  0054          LD      A,(OPATCH)
005A    BD      0055          CP      L
005B    2003    0056          JR      NZ,STHL-$
005D    211BE0  0057          LD      HL,NOLIST
0060    226B00  0058 STHL     LD      (OPATCH),HL
0063    1824    0059          JR      CHINM1-$
                0060 ;
0065    F5      0061 CHOUT    PUSH    AF
0066    CDE300  0062          CALL    CTRLS
0069    F1      0063          POP     AF
006A    C31BE0  0064          JP      NOLIST  ;OVERWRITTEN
>006B           0065 OPATCH   EQU     $-2
                0066 ;
006D    E5      0067 CT       PUSH    HL        ;CTRL-T
006E    C5      0068          PUSH    BC        ;TYPES SCREEN
006F    2180F0  0069          LD      HL,SCREEN
```

```
0072   0E1E    0070           LD      C,30    ;#LINES INSCREEN
0074   3E0C    0071           LD      A,FORMFD
0076   CD4F00  0072           CALL    PRINT
0079   0640    0073 CTL1      LD      B,64    ;# CHARS/LINE
007B   7E      0074 CTL2      LD      A,(HL)
007C   23      0075           INC     HL
007D   CD4F00  0076           CALL    PRINT
0080   10F9    0077           DJNZ    CTL2-$
0082   CD4D00  0078           CALL    CRPR
0085   0D      0079           DEC     C
0086   20F1    0080           JR      NZ,CTL1-$
0088   C1      0081 CHINM2    POP     BC
0089   E1      0082 CHINM1    POP     HL
008A   CD18E0  0083 CHIN      CALL    KEYBRD
008D   C8      0084           RET     Z
008E   FE10    0085           CP      10H     ;CTRL-P?
0090   28C1    0086           JR      A,CP-$
0092   FE14    0087           CP      14H     ;CTRL-T?
0094   28D7    0088           JR      Z,CT-$
0096   FE1E    0089           CP      1EH     ;CTRL-|?
0098   CA03E0  0090           JP      Z,WSTART
009B   FE05    0091           CP      5       ;CTRL-E?
009D   C0      0092           RET     NZ
009E   E5      0093 CE        PUSH    HL
009F   C5      0094           PUSH    BC
00A0   3E7F    0095 CE1       LD      A,BLOB
00A2   CD1BE0  0096           CALL    NOLIST
00A5   213300  0097           LD      HL,BUFFER
00A8   010D05  0098           LD      BC,BUFB+CR
00AB   CD18E0  0099 CC        CALL    KEYBRD
00AE   28FB    0100           JR      Z,CC-$
00B0   B9      0101           CP      C
00B1   2819    0102           JR      Z,FIN-$
00B3   FE7F    0103           CP      7FH     ;REDO?
00B5   28E9    0104           JR      Z,CE1-$
00B7   77      0105           LD      (HL),A
00B8   23      0106           INC     HL
00B9   FE20    0107           CP      ' '
00BB   3009    0108           JR      NC,NCTRL-$
00BD   C640    0109           ADD     A,40H
00BF   F5      0110           PUSH    AF
00C0   3E5E    0111           LD      A,'|'
00C2   CD1BE0  0112           CALL    NOLIST
00C5   F1      0113           POP     AF
00C6   CD1BE0  0114 NCTRL     CALL    NOLIST
00C9   10E0    0115           DJNZ    CC-$
00CB   79      0116           LD      A,C
00CC   CD1BE0  0117 FIN       CALL    NOLIST
00CF   3E0A    0118           LD      A,LF
00D1   CD1BE0  0119           CALL    NOLIST
00D4   4D      0120           LD      C,L     ;SAVE POSITION
00D5   213300  0121           LD      HL,BUFFER
00D8   7D      0122 NXPR      LD      A,L
00D9   B9      0123           CP      C       ;AT OLD POSITION?
00DA   28AC    0124           JR      Z,CHINM2-$
00DC   7E      0125           LD      A,(HL)
00DD   23      0126           INC     HL
00DE   CD4F00  0127           CALL    PRINT
00E1   18F5    0128           JR      NXPR-$
               0129 ;
>00E3          0130 CTRLS                     ;CHK FOR CTRL-S
00E3   3E03    0131           LD      A,3     ;IF SO WAIT
00E5   D3FE    0132           OUT     (KEYPRT),A;ELSE RETURN
00E7   DBFE    0133           IN      A,(KEYPRT)
00E9   E604    0134           AND     4
00EB   C0      0135           RET     NZ
00EC   3E00    0136           LD      A,0     ;CTRL?
00EE   D3FE    0137           OUT     (KEYPRT),A
00F0   DBFE    0138           IN      A,(KEYPRT)
00F2   E604    0139           AND     4
00F4   C0      0140           RET     NZ
00F5   CD18E0  0141 YES       CALL    KEYBRD  ;WAIT NOW
00F8   28FB    0142           JR      Z,YES-$
00FA   CD8A00  0143 WT        CALL    CHIN
00FD   28FB    0144           JR      Z,WT-$
00FF   C9      0145           RET             ;STILL PAGE 0
               0146 ;
ERRORS=0000

BLOB    007F   BUFB            0500 BUFFER        0033
BUFLEN  0005   CC              00AB CE            009E
CE1     00A0   CHIN    [INT]   008A CHINM1        0089
CHINM2  0088   CHOUT   [INT]   0065 CP            0053
CR      000D   CRPR            004D CT            006D
CTL1    0079   CTL2            007B CTRLS         00E3
FIN     00CC   FORMFD          000C IOINIT [INT]  003B
KEYBRD  E018   KEYPRT          00FE LF            000A
LIST    E993   NCTRL           00C6 NOLIST        E01B
NXPR    00D8   OPATCH          006B PRINT  [INT]  004F
SCREEN  F080   STHL            0060 WSTART        E003
WT      00FA   YES             00F5
```

a return is made (line 140).

Thus if a control-S is keyed, the point labelled YES (line 141) is reached. This point is a loop that keeps calling KEYBRD until all keys are released. Flow continues to a second loop labelled WT ("wait").

The program is immobilized here, until a character that is passed on by CHIN is received. The placement of CHIN in this loop enables me to use the commands activated by control-E, -^, -T, and -P.

MX-80 Hardware Interface

The Epson MX-80 printer appears to be a very versatile printer, and consequently, more confusing to connect. I made up a parallel interface cable with a 57-30360 (Amphenol or Cinch) plug at the printer end, and a DB-25P plug at the computer end, for the parallel interface socket. Between the plugs, I used a six foot length of 12 conductor cable that I had lying around. These are the connections I chose finally:

EXIDY	MX-80	REMARKS
1,8,25	9,14,19-30	Ground
2	10	MX-80 ack.
4	1	Exidy data str.
5	8	Bit 6
6	7	Bit 5
7	6	Bit 4
16	2	Bit 0
17	3	Bit 1
18	4	Bit 2
19	5	Bit 3

The grounding of Exidy pin 25, means that there is never a "BUSY" signal. The MX-80 pin 9 must be grounded, because the printer expects an 8th data bit (normally a zero), whereas the Sorcerer provides only 7 data bits with its Centronics interface. The MX-80 pin 14 is grounded so that the printer will line feed automatically after receiving a carriage return character; the Sorcerer driver filters out the LF character.

The Cassette

The preferable way, of course, to generate the machine code, is to start with the assembly code and assemble it. Since you may not have the patience nor the assembler, I have produced a dump of the machine code that can be entered using the Monitor command, ENter. After entering the hex values into the memory locations 3BH through 0FFH, issue the commands:

```
SET X=3B

SAVE MX80 3B FF
```

Of course, your cassette recorder should be set up to record the saved program. While you are at it, why not record the program several times over. That way you are less likely to "lose" the program, and you will not need to rewind the cassette so frequently. Incidentally, installing motor control for the cassette recorder(s) is well worth the effort!

The recorded cassette can be used with a simple "LOG" command. This printer software will become your constant companion!.

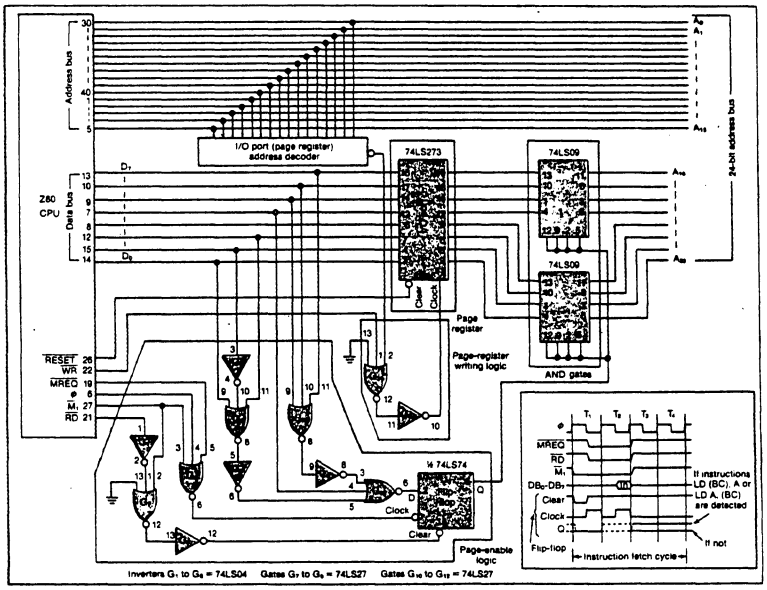
HARDWARE NOTES

by Russell Frew

Several issues back I wrote a column about the S-100 Box and its signals. I mentioned that I was working on generating the PHANTOM/ signal for the Sorcerer. I have since received many requests for information on how this signal might be used to expand memory beyond 64k.

The best method I've seen is an improvement on the traditional paging method. The technique I'll describe frees up the most significant address bit and therefore allows each page to have a full 64k of RAM. Since the technique adds 8 bits to the address bus, up to 256 pages can be addressed. Most importantly, the hardware is transparent to existing software.

Let me state right up front that this is no minor modification. It is intended for the experienced computerist with a strong hardware background and the right equipment. It involves at least 8 additional chips that must be timed and buffered correctly so as not to interfere with the Sorcerer's normal operation. I outline it here as a "think piece" only. A suitable modification for the main logic board and S-100 box would still have to be worked out.



Since the most significant address bit is not required to define local or main memory, this hardware modification will permit the addressing of up to 64 kbytes by the Z80's 16-bit address (a). Only two instructions are required to address the entire main memory, as shown in the timing diagram (b).

Diagram reprinted with permission from **Electronic Design**, Vol. 29, No. 20; copyright Hayden Publishing Co., Inc., 1981.

The technique centers on the bit pattern of two instructions, and the M1/ cycle. If you look at both the timing diagram and the schematic, you will see that when the clock, M1/, & MREQ/ are all low, the flip-flop will be set, 00000010 or 00001010 is on the data bus. Every time the M1/ signal is logic 0, we know that an instruction is being fetched. The two binary numbers correspond to the two instruction codes for LD A,(BC) and LD (BC),A. Once these instructions are fetched, the contents of the LS273 page-register are added to the Z-80's 16 bit address bus extending it to 24 bits. The next instruction is then executed at the newly defined page. When M1/ goes to logic 0 at the next T1, the flip-flop is cleared and the program resumes at page 0 in main memory. This technique does still confine program memory to the 56K of RAM that the Sorcerer has at page 0, but it allows over 16 megabytes of data memory with very little hardware overhead.

Design considerations that should be kept in mind include the need to pick-up most of these signals inside the Sorcerer. While the S-100 box has the free address lines to extend the addressing to 24 bits, those 8 additional lines will have to be carried out of the Sorcerer on another flat cable and buffered into the Expansion Box. Also, don't forget that the expansion bus is non-selected whenever the main logic board is active.

I hope this approach helps some of you who are trying to add more memory. This method of memory paging for the Sorcerer, seems to involve the least chip count and has the distinct advantage of causing the least disruption to existing software.

This concept was originally developed by Henrique Malvar, Department of Electrical Engineering, University of Brazil and was originally discussed in Electronic Design magazine. Our thanks to him for sharing his ideas.●

USING ML ROUTINES FROM THE BASIC PAC

by Bill Boucher

(This article is reprinted from the May, 1980 issue of the S.U.N.)

For those of us into machine or assembly language, I found several useful subroutines in BASIC.

A routine at D015 will output to the screen the ASCII string starting at the memory location specified by HL. The string must be terminated with a 00.

Example routine calling D015:

```
0000 21 07 00   SAYHI   LD     HL,MSG1   ;Set HL to string
0003 CD 15 D0           CALL   D015      ;Call output subr.
0006 C9                 RET
0007 48         MSG1    "H"              ;Start of string
0008 45                 "E"
0009 4C                 "L"
000A 4C                 "L"
000B 4F                 "O"
000C 00                 NOP              ;End of string
```

Another routine at D7BB converts the 16 bit number in the HL register pair to decimal and outputs it to the screen.

Example calling the routine at D7BB:

```
0000 21 01 01   DECOUT  LD   HL,0101H ;HL=257 decimal
0003 CD BB D7           CALL D7BB     ;Convert & displ.
0006 C9                 RET
```

Bill Boucher, 1740 California St. #7,
Mountain View, CA, 94040

MERGE

by James E. Cooper Jr.

I wrote the following routine, to be able to use programs like Tom Bassett's modification of Devin Trussel's RENUMber program (S.A. Vol.1, No.5), without having to type it in every time. Since I only had 8K of memory at first, using Tom's MERGE program (in BASIC), plus RENUM, left me with practically no room for my own programs.

The whole reason for using named files is to make keeping track of them easy. With every MERGE program on which I have obtained information, unless it was disk-based, you had to position the tape at the beginning of the file to be MERGED. Last but not least, that extra typing needed, such as:

EN 1 B 7 <CR> and etc.

is for the birds. So I came up with my own MERGE.

Loading Instructions:

From BASIC: type --- CLOADG MERGE, <CR>.

From the Monitor: type - LOG MERGE, <CR>.

MERGE loads from 10H to F3H, and sets the input vector to 10H, the same as typing SE I=10 while in the Monitor. I chose for MERGE to load at 10H because my Sorcerer likes to "chew up" the first few bytes of memory when I push the RESET buttons.

Operating Instructions:

I made this part as simple as I could (mainly because I'm lazy). When you load MERGE, it takes over with its keyboard input routine, which just checks for three CTRL

keys which aren't used for anything, and returns everything else.

MERGE looks for the following keys:

CTRL-B : Disables MERGE. You must use this key before you run any program which POKEs into low memory. Otherwise, strange things happen, eventually causing BASIC to crash if you POKE in the wrong place.

CTRL-R : Recovers a program after you have typed NEW, CLOAD, or hit RESET. This works only if your program hasn't been destroyed by random POKEs or stack overflow.

NOTE: If you want to MERGE something to a program you've loaded from a tape, I suggest you use CTRL-R to readjust the end-of-program pointer before you MERGE anything to it. The reason for this is that standard BASIC adds a byte to the end of a program every time you CSAVE it. In other words, if you save a program five times, the last copy will have four extra bytes at the end. These extra bytes keep MERGE from working correctly. CTRL-R drops these extra bytes.

CTRL-E : Extends the program by MERGEing. When you press CTRL-B, there won't be any visible change except that CTRL-B, CTRL-R, and CTRL-E will give, ?SN ERRORS, as if MERGE had never been loaded. When you press CTRL-R, the screen should show a small flicker, however brief, but no other visible signs that it worked. However, when you press CTRL-E, you will see:

Unit # , Name ? -

displayed on the screen (or whatever your input device happens to be). You now have several options:

1) Hit <CR>. The defaults take over and MERGE the first program from tape drive #1.

2) TYPE: 1, <CR>. Same effect as 1.

3) TYPE: 2, <CR>. MERGEs the first program from tape drive #2.

4) TYPE: 1, 'file name' <CR>. This will MERGE the program 'file name' from tape drive #1.

EXAMPLE: To MERGE PROG1 from drive #1, TYPE: 1, PROG1 <CR>

5) TYPE: 2, 'file name' <CR>. Same option as #4, except it uses tape drive #2. If you use a file name, you must include the drive unit #!

After you hit <CR>, MERGE will turn on the specified (or defaulted) drive unit. Then, if a file name was given, it will search the tape for the correct file and MERGE it with the program in memory. Otherwise, it will simply MERGE the next file on tape to the program in memory.

MERGE adjusts ALL pointers in memory, so there's no more worrying about typing: 0 REM, then delete line 0 - then trying to remember whether or not you already HAD a line 0.

Cautions

I said that MERGE will recover your program after you push RESET. This is true **only** if you give input control back to MERGE. You can do this in one of three ways:

1) Go to the Monitor and SET I = 10H, the start of MERGE.

2) Go to the Monitor and type GO E7. MERGE will set the input vector for you.

3) From BASIC type the following line (substitute numbers for the n's): POKE nnnn, 16: POKE nnnn+1,0 <CR>.

IMPORTANT: the above **must** be on one line, or your Sorcerer will hang up, meaning you will have to RESET it again.

For 8K, nnnn = 8146
For 16K, nnnn = 16338
For 32K, nnnn = 32722
For 48K, nnnn = 16430

You must be careful about the programs you merge. MERGE doesn't care what type of program you tell it to get, so you must keep track, or you'll waste a lot of time re-typing destroyed programs.

Whatever program you merge MUST have line numbers greater than those already in memory, or the merge will be useless and will use up precious memory.

Command Summary

CTRL-B = Disable MERGE and return to BASIC
CTRL-R = Recover program and reset pointers
CTRL-E = Extend program by MERGEing

```
;
;  Merging with unit # and name.
;  Works with Monitor Ver 1.0,
;   and Standard BASIC Ver 1.0
;
;  (Other versions of Monitor and
;   BASIC not tested as yet.)
;     Copyright (C) 1981 by
;          James Cooper Jr.
;
;  Equates, in Alphabetical order:
;
BLADJ   EQU     0C3E1H  ;BASIC LINE POINTER ADJUST
BPEND   EQU       1B7H  ;END-OF-PROGRAM STORAGE
BPSTO   EQU       1D5H  ;BASIC PROGRAM STORAGE
CNAME   EQU     0E264H  ;FIND FILE NAME IN INPUT
CONV    EQU     0E23DH  ;ASCII TO HEX IN DE
CR      EQU        0DH  ;CARRIAGE RETURN
CRLF    EQU     0E205H  ;SEND CR THEN LF TO OUTPUT
CTRLB   EQU         2H  ;CONTROL-B
CTRLE   EQU         5H  ;CONTROL-E
CTRLR   EQU        12H  ;CONTROL-R
FOUND   EQU     0E4CAH  ;FOUND MESSAGE
GETHD   EQU     0E71BH  ;GET FILE HEADER FROM TAPE
GETIY   EQU     0D1A2H  ;FIND MWA, PUT IT IN IY
HDPRT   EQU     0E6DEH  ;PRINT FILE HEADER
KEYBD   EQU     0E018H  ;KEYBOARD INPUT
MONIN   EQU     0E13AH  ;MONITOR INPUT
MOTON   EQU     0E28AH  ;CASSETTE MOTOR ON
PRINT   EQU     0E1BAH  ;MONITOR 'SENDLINE'
SKIPF   EQU     0E734H  ;SKIP FILE ON TAPE
T2MEM   EQU     0E7F8H  ;TAPE TO MEMORY TRANSFER
;
        ORG       10H
START   EQU     $
        CALL    KEYBD           ;KEY PRESSED?
        JR      Z,START-$       ;NO, TRY AGAIN
        CP      CTRLE           ;CONTROL-E?
        JR      Z,MERGE-$       ;YES,MERGE
        CP      CTRLR           ;CONTROL-R
        JR      Z,RECOV-$       ;YES, RECOVER PROGRAM
        CP      CTRLB           ;CONTROL-B?
        RET     NZ              ;NO,RETURN TO BASIC
;
RSETI   CALL    GETIY           ;FIND MWA
        LD      HL,KEYBD        ;KEYBOARD INPUT ROUTINE
        LD      (IY+41H),L      ;LOW BYTE
        LD      (IY+42H),H      ;HIGH BYTE
        RET                     ;INPUT IS NOW 'KEYBD'
;
RECOV   LD      A,0             ;BYTE TO FIND
        LD      HL,BPSTO+4      ;WHERE TO LOOK
        LD      BC,0            ;LOOK 'TIL FOUND
        CPIR                    ;DO IT!
        LD      (BPSTRO),HL     ;SECOND LINE START
CHECK   LD      A,H
        OR      L               ;PROG END?
        JR      Z,PNTR-$        ;YES, RESTORE POINTER
        LD      E,(HL)
        INC     HL
        LD      D,(HL)
        EX      DE,HL           ;POINT TO NEXT LINE
        JR      CHECK-$
PNTR    EX      DE,HL           ;GETEND IN HL
        INC     HL              ; +1 BYTE
        JR      STORE-$         ;ADJUST POINTERS & RETURN
;
MERGE   EQU     $
        CALL    CRLF            ;SKIP A LINE
        LD      HL,MSG          ;START OF MESSAGE
        CALL    PRINT           ;PRINT IT
        CALL    GETIY           ;NEED FOR MONIN
        PUSH    IY              ; & LATER
        CALL    MONIN           ;GET UNIT # & NAME,
                                ; IF ANY
        POP     HL              ;MWA IN HL
        LD      E,1             ;CASSETTE 1 IS DEFAULT
        LD      A,(HL)          ;INPUT
        CP      CR              ;ANY?
        CALL    NZ,CONV         ;YES, GET DRIVE # IN E
        LD      B,E             ;OTHERWISE, DRIVE # 1
        CP      CR              ;IS THAT ALL THE INPUT?
```

```
        JR      Z,NEXT-$        ;YES, NO NAME
        PUSH    BC              ;CNAME DESTROYS B
        CALL    CNAME           ;GET NAME FOR FILE
                                ; TO MERGE
        POP     BC              ;GET DRIVE # BACK
NEXT    LD      DE,(BPEND)
        DEC     DE
        DEC     DE              ;GET END OF PROG
        PUSH    DE              ; & SAVE FOR LATER
        LD      HL,RETPT        ;RETURN POINT FROM
                                ; TAPE LOAD
        PUSH    HL              ;SAVE IT
        EX      AF,AF'          ;PROTECT FLAGS
        OR      1               ;RESET Z FLAG
        PUSH    AF              ;MEANS - DON'T AUTO
                                ; EXECUTE
        EX      AF,AF'          ;GET FLAGS BACK
        PUSH    AF              ; JUNK, BUT NECESSARY
        PUSH    AF              ;Z MEANS NO NAME
        CALL    CRLF            ;SKIP A LINE
        CALL    MOTON           ;START DESIRED TAPE DRIVE
FLOOP   PUSH    DE              ;SAVE LOAD ADDR
        CALL    GETHD           ;GET HEADER FROM TAPE
        LD      A,(IY+5CH)      ;CHECK FILE TYPE
        OR      A               ;SET FLAGS
        JR      Z,LOAD-$        ;SKIP HEADER PRINT
        PUSH    HL              ;SAVE FOR LATER
        LD      HL,FOUND        ;PRINTS "FOUND -"
        CALL    PRINT
        CALL    HDPRT           ;IDENTIFY FILE TO USER
        POP     HL
LOAD    POP     DE              ;RESTORE LOAD ADDR
        POP     AF              ;NAME?
        PUSH    AF              ;SAVE AGAIN
        JR      Z,TAPIN-$
        JR      C,TAPIN-$       ;NO, MERGE THIS ONE
        PUSH    IY              ;PUT IY
        POP     IX              ; IN IX
        LD      B,5             ;NAME LENGTH
NLOOP   LD      A,(IX+47H)      ;COMPARE INPUT
                                ; TO NAME WANTED
        CP      (IX+57H)
        INC     IX
        JR      NZ,FSKIP-$      ;WRONG NAME, SKIP FILE
        DJNZ    NLOOP-$         ;OK SO FAR, KEEP CHECKING
TAPIN   JP      T2MEM           ;MATCH - MERGE PROGRAMS
;
RETPT   CALL    CRLF            ;SKIP ANOTHER LINE
        POP     HL              ;FIRST PROG END
        LD      E,(IY+5EH)
        LD      D,(IY+5FH)      ;GET # OF BYTES IN 2nd
                                ; PROGRAM
        DEC     DE              ; -1 FOR MEMORY ADDR
        ADD     HL,DE           ;END OF MERGED PROGRAMS
                                ; IN HL
STORE   LD      (BPEND),HL      ;START OF VARIABLES
        LD      (BPEND+2),HL    ;START OF ARRAYS
        LD      (BPEND+4),HL    ;START OF FREE MEMORY
        LD      HL,(BPSTO)      ;ADDR OF 2nd LINE
        POP     DE              ;ADJUST STACK, WE
                                ;WON'T RETURN
        JP      BLADJ           ;ADJUST LINE POINTERS AND
                                ;RETURN TO BASIC
                                ;COMMAND LEVEL
;
FSKIP   PUSH    DE              ;SAVE LOAD ADDR
        CALL    SKIPF           ;SKIP FILE
        POP     DE              ;RESTORE LOAD ADDR
        JR      FLOOP-$         ;FILE LOOP
;
MSG     EQU     $
        DEFB    'U'
        DEFB    'n'
        DEFB    'i'
        DEFB    't'
        DEFB    ' '
        DEFB    '#'
        DEFB    ','
        DEFB    ','
        DEFB    'N'
        DEFB    'a'
        DEFB    'm'
        DEFB    'e'
        DEFB    '?'
        DEFB    ' '
        DEFB    0
SETI    CALL    GETIY           ;FIND MWA
        LD      HL,START        ;BEGINNING OF MERGE
        LD      (IY+41H),       ;LOW BYTE
        LD      (IY+42H),H      ;HIGH BYTE
        RET                     ;INPUT NOW 'START'
```

RELOCATED W-PAC MODIFICATION

by Terry L. Calvert

This is a short modification for the relocated WP program that allows users to store and recall text on cassette. The beauty of this modification is that it is shorter than the existing READ and WRITE commands in the WP, and stores text in Exidy's standard program format, rather than the WP format.

To use the modification:

1) Enter the new codes at the addresses shown. (For a WP at 8000H, the first digit would be '9').

2) You may zero out the rest of the command areas: 6BB9-6BFB and 6C83-6CAD. These areas can now be used for other commands, additional workspace, or left blank.

3) Only Unit #1 is used to save and read programs, but a unit no. must still be entered after the prompt. This can be changed by changing these locations:

```
6C60: 21  B0  6C   : delete the '#/' from
                     prompt
6C6F: 21  B0  07   : points to beginning
                     storage input of name.
```

4) Use the W and R commands in the WP as normal and enjoy quicker response from your Sorcerer.

Credit belongs to R. Henne for pointing out the use of the AUTOLOAD and AUTOSAVE routines in the Monitor in his, "Mini-Word Processor", in the **Book of Sorcery**, Vol. V.

```
6B9C:   F5              PUSH AF
6B9D:   C5              PUSH BC
6B9E:   D5              PUSH DE
6B9F:   E5              PUSH HL
6BA0:   DD E5           PUSH IX
6BA2:   FD E5           PUSH IY
6BA4:   CD 60 6C        CALL 6C60
6BA7:   21 78 6C        LD   HL,6C78
6BAA:   E5              PUSH HL
6BAB:   21 F6 5D        LD   HL,5DF6
6BAE:   06 02           LD   B,02
6BB0:   3E 01           LD   A,01
6BB2:   B8              CP   B
6BB3:   F5              PUSH AF
6BB4:   37              SCF
6BB5:   3F              CCF
6BB6:   C3 2D E0        JP   E02D

6C36:   F5              PUSH AF
6C37:   C5              PUSH BC
6C38:   D5              PUSH DE
6C39:   E5              PUSH HL
6C3A:   DD E5           PUSH IX
6C3C:   FD E5           PUSH IY
6C3E:   CD 60 6C        CALL 6C60
6C41:   CD A2 E1        CALL E1A2
6C44:   FD 36           LD   (IY+4D),D7
6C48:   ED 5B 4A 07     LD   DE,(074A)
6C4C:   01 00 07        LD   BC,0700
6C4F:   FD 71 50        LD   (IY+50),C
6C52:   FD 70 51        LD   (IY+51),B
6C55:   21 78 6C        LD   HL,6C78
6C58:   E5              PUSH HL
6C59:   C5              PUSH BC
6C5A:   21 F6 5D        LD   HL,5DF6
6C5D:   C3 2A E0        JP   E02A
6C60:   21 AE 6C        LD   HL,6CAE
6C63:   CD 89 5C        CALL 5C89
6C66:   21 B0 07        LD   HL,07B0
6C69:   CD 77 5A        CALL 5A77
6C6C:   11 D8 7F        LD   DE,7FD8
6C6F:   21 B2 07        LD   HL,07B2
6C72:   01 05 00        LD   BC,0005
6C75:   ED B0           LDIR
6C77:   C9              RET
6C78:   FD E1           POP  IY
6C7A:   DD E1           POP  IX
6C7C:   E1              POP  HL
6C7D:   D1              POP  DE
6C7E:   C1              POP  BC
6C7F:   F1              POP  AF
6C80:   C3 1B 5F        JP   5F1B
6C83:   00              NOP
6C84:   00              NOP
```

! ! ! HELP ! ! !

SORCERER'S APPRENTICE

Needs:

☆ **Articles!** ☆

☆ **Advertisements!** ☆

☆ **A NEW EDITOR!** ☆

If you can help with any of the above, please contact us at P.O. Box 33, Madison Heights, MI, U.S.A., or by phone: Don Gottwald (313) 286-9265; or new editor only, Ralph LaFlamme (318) 856-4954.

ASTRONOMY PROGRAMS

MARS - Distance and angular diam. of Mars for any date; date and details of next opposition following any date. $10

MVEN - Phase, distance and angular diam. of Mercury and Venus for any date and next elongation after any date. $10

MERVE- Graphical display of Mercury and Venus relative to Sun for series of time intervals; distances, etc. for any date. $10

PRISE- Risings or settings of Mercury and Venus before or after the Sun for any location and date. $10

RISES- Times of rising, transit, and settings for any planet, Sun, or Moon for any location and date. $10

SSTAR- Dates, radiants, etc. of annual meteor showers and graphical display for selected month. $20

JSATS- Displays configurations of Jupiter's satellites for any date and time or series of dates and times, N or S at top. $10

ECLIP- Gives date and magnitude of next umbral eclipse of Moon, starting any year and continuing for as long as requested. $5

PLTTN- Ask for any planet or Sun on any date and program selects and displays a star map and plots planet plus any others and Moon if in same region. With or without RA and Dec grid, and plots a series for selected time intervals. Identifies stars. Indicates phase of Moon. $20

RADEC- Gives RA and DEC for planets, any date. $15

RADCM- Gives RA and DEC for Moon, any date. $10

SKYPN- Plots stars, planets, Sun, and Moon visible above horizon at any time and date in Northern or Southern Hemisphere to 85 deg. lat. $25

BOOK of listings of **20 astronomical programs** with photos of screen displays $25 plus $3.50 postage and handling ($7 overseas).

(A self-addressed envelope for details. Available for Astrologers also.)

(Overseas, add $2 per order, for Airmail)

Eric Burgess. F.R.A.S.,13361 Frati Lane, Sebastopol, CA 95472

RANDOM I/O

by Don Gottwald

Jack MacGrath of Tercentennial Technical (see his ad elsewhere in this issue) is willing to provide Level II BASIC on cassette or Micropolis Mod II formatted diskettes for $3 plus postage. Send him a cassette or a FORMATTED disk along with a money order or check for $3. Don't forget to include postage.

C. Richard Stelling of 1501 Dial Court, Springfield, IL 62704, is looking for good prices on used Micropolis Mod II drives. He also needs CP/M for the Vista drive.

Computer Dynamics, 105 S. Main St., Greer, SC 29651 (803) 877-7471, is offering 5 Megabyte Shugart SA-1002 Winchester Hard Disk Drives for only $600. Supposedly, it is a snap to interface with the Western Digital WD1000-80 Controller, which they sell for $500. Interfaces are available for the S-100, IBM, STD Bus and others. CP/M BIOS is also available. Call or write (SASE) Bill Star at CDI. If anyone does interface the Sorcerer to one of these drives, please let us know so we can publish the results.

Kevin McCabe, a member of our group, has co-authored a book on Sorcerer programs. The title is, "32BASIC Programs for the Exidy Sorcerer". It is written by Tom Rugg, Phil Feldman, and Kevin McCabe, and is published by Dilithium Press. It's a very good reference book for beginners and experts alike. Highly recommended for your library.

There are several new books on the Z-80 microprocessor on the market. If you would like to know more about the hardware aspects of the Z-80, and how it relates to Assembly Language programming, try, "**Z-80 Users Manual**", by Joseph J. Carr, (published by Reston; price $10.95). There is also, "**Z-80 Microcomputer Design Projects**", by William Barden Jr., (Howard Sams Book; price $13.95). This book takes you step by step through the construction of a minimal system using the Z-80 microprocessor. In addition to learning about the Z-80 hardware, you'll also learn how to program in Assembler. There are many projects (most under $50.00) listed using just a Z-80 and a few other chips. This book, although not specifically about the Sorcerer, will greatly aid your understanding of the hardware and software aspects of the Sorcerer.

Have you tried the game of 'DUEL' yet? With the new version, you do not have to 'LO' and then 'GO 100', you can just 'LOG' it and it'll be ready for two players to live out their Starwars fantasy. The graphics in this game really are astounding. (See ad elsewhere in this issue).

Have you been getting keybounce lately? Do you hit a key and no letter appears on the screen? Both of these problems may be due to dirty key contacts. To clean them, disconnect the power, then very carefully, pry off the keycaps using a small screwdriver. With a fine burnishing tool (available at most hobby stores), gently burnish the contacts while holding the key down with a screwdriver. (NOTE: To prevent damage to the fingers on the contacts, make sure the screwdriver only touches the plastic frame and not the contacts themselves). A few strokes straight up and down should do it. Do not twist the contacts. Replace the keycap and you should be back in business minus the annoyances.

Newmar Computer Supplies, 55 Salem Street, Lynnfield, MA 01940, tel. (617) 245-7960, has some very good prices on a large variety of diskettes. Call or write Frank Stout for the latest prices.

Cary Stewart, 529 S. Beachwood Drive, Burbank, CA 91506, tel. (213) 843-1101, (editor of the Southern California Sorcerer User Group Newsletter -SASE for copy-) is organizing a group purchase of parts for blank S-100 cards to be purchased from either Bryan Wagner or South Valley Electronics. Contact him if you're interested in participating. A preliminary estimate places the cost in the area of $50.

By the way, South Valley Electronics of Santa Clara, CA will also do Exidy repairs. See page 71 of SA Vol. 4, issue #3 for their address and telephone number.

We would like to encourage our friends, both inside and outside the U.S., to submit sources for repairs, parts and other hardware and software items of interest to Sorcerer owners. We want to compile a listing of where members can obtain service and supplies.

Dr. William Mahaffey of P.O. Box 756, Norfolk, VA 23501, wants to sell his new and untried Australian Stringy Floppy with Sorcerer PROM for $175. Write to him for particulars.

Allyn H. Fisher writes that his experiences paralleled Eric Zorawowicz's (see, "The Computer System Evolution", in issue 4.3), in almost every detail. Allyn has had a good deal of trouble with his Vista V-200 drives. They have only worked about two weeks out of the 20 months that he has had them! He thanks Jack MacGrath for his help in getting and keeping his system running.

4th TIP

by Timothy Huang

Managing FORTH(.COM) Files

As many of you already know, the FORTH concept is very different from other computer languages to which you have been exposed. I think, therefore, that it is a good time to answer one reader's question about how to use my, "And So FORTH" package. As I see the question, it should be re-phrased to: "How does one manage FORTH files?", or "How do you use FORTH?"

With other languages, such as BASIC, FORTRAN, or PASCAL, we have the language (either Interpreter or Compiler), various utilities (Editor, Assembler, Debugger, Linker/-Loader, etc.), and the programs. There are clear cut differences between them. No one will mistakenly use the utility module to interpret a program. No one can mess up the

application program(s) or the language with which they were written. This, however, is not the case with FORTH.

As a matter of fact, we never use FORTH to write any "program", because there is no FORTH program. There is no clear cut difference between FORTH itself or its extensions (Screen Editor, Debugger, Assembler and applications). All of the extensions are simply layers of FORTH, just like the onion adding another layer while its growing. Every time we write something with FORTH, we are not writing something external or foreign to FORTH but an extension of it. Therefore, we don't write "programs", rather, we extend FORTH. But for most of you who still possess strong residual feelings about the traditional sense of "program", I'm willing to make the following compromise:

> A FORTH "program" is an extension of itself, and is written in FORTH.

With this modified definition, we can still use the term "program" to illustrate our points.

In the "And So FORTH" package, there are two FORTH files. The first is a bare bones FORTH(.COM). This is the kernel portion. The other is the extended BIGFORTH(.COM). In order to avoid possible confusion between the two, I'll call the first FORTH and the second BIGFORTH.

The current version of FORTH is a machine executable program operating under CP/M. The names are appended with the memory size and are called FORTH32K.COM and FORTH-48K.COM. This file should be kept as small as possible. Anything which is not absolutely essential should be left out to avoid unnecessary overhead. You'll see the reasons for this later. The FORTH files are each about 8K long. Of course they can be trimmed further, if necessary.

The BIGFORTH file is an enlarged FORTH file whose memory size is indicated as B48FORTH.COM or B32-FORTH.COM. On top of the FORTH file, I added (pre-compiled) the following "programs": (a) Screen Editor, (b) DIS-forther, (c) TRACEr, and (d) misc. words.

BIGFORTH is primarily intended as a development system rather than a final "program". As you can see, those additions are handy tools for the development of final application programs, but have nothing to do with whatever the application programs are. The Screen Editor enables us to write source programs (screens), and after LOADing the screens, DIS-forther and TRACEr allow us to debug, disassemble the compiled words, and trace the execution of the program(s). Other utilities can be added as needed. Thr Assembler is one good candidate, if we need the speed of execution. Thus, we can have many versions of BIGFORTH, each customized for a particular development requirement. Unfortunately, due to the short sighted CP/M file naming convention, we are limited to only 8 characters for the file name. You may have to be creative to maintain separate BIG-FORTH file identities.

Assuming that you were commissioned to write a Super Duper Word Processor program, here is how you could use the "And So FORTH" package:

1. Boot up BIGFORTH containing the Screen Editor.
2. Type in the Super Duper Word Processor source screens to a new disk, assuming that you have already developed the source program.
3. LOAD the Super Duper Word Processor on top of BIGFORTH. While the source is compiling, any typo or undefined word will terminate the compilation (loading). If necessary, go back to the Screen Editor and make all the syntax corrections.
4. LOAD the source again on top of BIGFORTH and try to execute the Word Processor. You may discover some logic bugs. Use the TRACEr and DIS-forther to help debug the program. Go back to the Screen Editor to make the logic corrections or enhancement modifications. After this step, you have a debugged version of the Super Duper Word Processor.
5. Now, you can boot up the FORTH file and then LOAD the debugged source program on top of it. Remember, the Word Processor "program" will be compiled on top of the bare bones FORTH.
6. Now, to save your new creation. The most straight forward way is to adjust the boot-up parameters and save the FORTH plus WP by LOADing Screen 40 (set cold start parameters) and 41 (provide CP/M SAVE information) from the Screen Disk I. You can now create a WORDPROC.COM file for CP/M.
7. If you examine the practice described in step 6, you'll see that your product contains an 8K FORTH overhead. Since the Word Processor portion may only contain 4K, you wind up with a "program" three times as long as the application alone. Besides being a possible copyright infringement of the FORTH package, this combination also gifts someone a FORTH which a novice may then unintentionally get all fouled up.
8. One intermediate solution to this dilemma is to hide the FORTH overhead in the application program. The definition and usage of the word HIDE is provided in Chapter 14 of "And So FORTH". By doing so, the danger of the novice's messing up FORTH is removed. One word of caution, however. If this trick is not used properly, you can be sealed out too. The system may also crash without warning.
9. Even with the FORTH overhead hidden in the application program, it still carries a lot of unnecessary dead weight. This is the reason for keeping the FORTH file as small as possible. Your dilemma is having the convenience of as many words available as possible verses the size of the final overhead. A compromise must be found. The FORTH file contains those words that I thought were essential for general applications with the Sorcerer. You may have a different requirement.
10. The ultimate solution is to strip off all unnecessary portions from the total program package. The name fields and link fields of all words can be removed as well as all unused FORTH words. Essentially, what you have left in this optimized package are: the FORTH inner (address) interpreter, the Word Processor words and some absolutely required I/O words. The outer (text) interpreter also may not be needed. Thus, the final overhead can be as small as several hundred bytes (such as the one used in the Craig language translator). With this approach, you can even ROM the package. On top of that, if anyone tries to disassemble your code, they will have a lot of fun trying to figure out exactly what's going on. (Think about trying to disassemble a bunch of routine addresses!) This process is called Meta-compiling, Target-Compiling, or Cross-Compiling. It's a more advanced subject, and we'll not worry about it now. The final product of meta-compiling could no longer be FORTH but a pure application program. Of course, we can generate a better, newer version of FORTH or BIGFORTH by this same process. This is something that other languages cannot do!

Some of you may ask, "Why use FORTH to write program(s) in the first place? Why not use Assembly language or something on a lower level?" The answer is very simple. Since FORTH is a very high level language, it is much easier to accomplish the job faster than with a low level language. The execution speed will be only minimally sacrificed (about 1-2 times longer than pure assembly equivalent, while BASIC programs run about 30-50 times slower), and it will not require other run time utility programs to execute. By contrast, in using a language such as BASIC, or even CBASIC, our application program cannot be made stand-alone. We always need something else to run the program (CP/M to run a compiled BASIC program, a BASIC interpreter to run an interpretive BASIC program, CRUN to run programs written in CBASIC). In addition, these overhead utilities are getting bigger and bigger. (The Exidy Extended BASIC takes up 20+K of memory! With that kind of memory, FORTH can handle a 2-user system!).

FORTH can be thought of as a user expandable high level macro assembler. If we think about it, our familiar Z80 CPU, was provided with a fixed set of instructions. We have no choice but to accept the instructions provided even though some of them may be useless to us. On the other hand, FORTH is a virtual language, with a set of instructions which can be expanded. Assembly language is very powerful, but a pain in the neck to use. FORTH overcomes the pains but keeps the power of a macro assembler. With this different concept of programming, we can understand why we never write FORTH "programs" in the traditional sense. We don't have to. We just expand to include those desired instructions! This is beautiful!●

Members of the **Sorcerer's Apprentice User's Group** are entitled to 8 issues of the group's Newsletter, the **SORCERER'S APPRENTICE**; the services of the library; access to its on-line CP/M based Computer Bulletin Board Service; other services as they become available.

MEMBERSHIP RATES for 1982: USA - bulk postage - $18, 1st class postage in an envelope - $24; Canada & Mexico - $24; single issues $3; all others - air mail - $32, single issues $4.

BACK ISSUES:	Sorcerer's Apprentice Vol III (1-8)	$12.00
	S.A. Vol. III (per issue)	$ 2.50
PROGRAMS:	Personal Accounts Payable System (PAPS) (3 diskettes, Micropolis Mod II format)	$50.00
	Sorcerer Telecommunications System (STS) (Tape modem program to access and transfer CP/M programs. (See issue 4.2, p. 32)	$30.00

Overseas orders for back issues add $4 per volume or $1 per issue to cover additional air mail postage and handling.

Commercial advertisers, please contact us for advertising rates. **Non-commercial classified ads** are accepted at the rate of $1 per 35-column line or part-line.

Make checks or money orders (only in US funds drawn on a US bank) payable to: **SORCERER'S APPRENTICE.**

Newsworthy items may be submitted via the MiniCBBS on the Sorcerer-based RCPM at (313) 535-9186R (ringback), the SOURCE (TCF656), or MicroNET (70150,365), on Word Processor cassettes or CP/M Word Processor/Editor files on Micropolis Mod II hard- or soft- sectored diskettes (any of these preferred) or hardcopy. Magnetic media returned upon request. Hardcopy will be returned if requested and accompanied by SASE.

SEND ALL CORRESPONDENCE TO:

SORCERER'S APPRENTICE
P.O. Box 33
Madison Heights, Michigan 48071
U.S.A.

BULK RATE
U.S. POSTAGE
PAID
WYANDOTTE
MICHIGAN 48192
PERMIT NO. 23

RETURN AND FORWARDING POSTAGE GUARANTEED