
The Sorcerer User's Newsletter is published by the Sorcerer's Apprentice, P.O.Box 1131, Troy, MI, 48099

S.U.N. IS UNDER NEW MANAGEMENT-

The Sorcerer's Apprentice User Group will be publishing the S.U.N. for the remainder of Volume II. See the article on page 52 for a report from the Sorcerer's Apprentice President, Ed Heussner. The group will try to answer the back mail that was passed on by Steve Long. The remaining issues will contain information sent in by users over the past months.

The Sorcerer's Apprentice did not take over the program library. I don't know if Steve is still operating it or not. I will try to furnish this information in the next issue of S.U.N.

Steve started to put together this issue several months ago and it just got to be too much for one person to handle. I hope he doesn't mind me using his last "editorial".

> "The primary purpose of the S.U.N. is to assist people using their Sorcerers. There are at least 20,000 Sorcerer owners and more are starting out each month. I have only been able to reach about one percent of these users with the S.U.N. and have become very discouraged in the past few months. The tremendous amount of work involved is not justified by the small number of users who want this type of information." Steve Long

I have included this statement by Steve to indicate the problem that the Sorcerer newsletters have been facing. Our own Sorcerer's Apprentice almost folded when Dave Bristor couldn't devote enough time to the newsletter, the Monitor stopped publishing, the ARESCO Source went under. To make the newsletter work you need more than one person to edit, publish, distribute, and answer mail. Our user group is now organized to do this. You also need interested, participating subscribers to share experiences, discoveries, opinions or whatever you want in the newsletter. If you want a Sorcerer newsletter for 1981 sit down and drop us a line. Practice using your WP PAC or just scratch something on a postcard. Let us know if you're still computing.

USING MACHINE LANGUAGE ROUTINES FROM THE BASIC PAC- from Bill Boucher, 1740 California St. #7, Mountain View, CA, 94040

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.

(continued on page 52)

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 and display
0006  C9                  RET
```

A REPORT TO SUN SUBSCRIBERS- by Ed Heussner, President, Sorcerer's Apprentice

You probably have already detected some differences in the layout of this issue. The S.U.N. is under new management. Steve Long decided he could no longer devote the time to the S.U.N. that the newsletter required. Hence, he contacted us during the summer and asked if the Sorcerer's Apprentice was interested in finishing out Volume II of the S.U.N. After straightening out the technicalities of such an arrangement, we have agreed to complete publication of Volume II. The S.U.N. will be the Sorcerer's Apprentice effective with the January 1981 issue. Subscription rate for the United States and Canada is $12.00 (U.S.) per year and $18.00 (U.S.) for other foreign. Single copy and back issue price is $2.00 per issue. Make checks payable to "Sorcerer's Apprentice" and address all mail to:

SORCERER'S APPRENTICE
P.O. BOX 1131
TROY, MI 48099

USEFUL POKE COMMANDS- from William Cohen (age 14)

In the January 1980 issue (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 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)- for BASIC only- POKE 322,0
2. Line width control- for BASIC only- POKE 322,X (where X=1,2,3,4.....64)

This command controls the length of the line. It will be limited to X length. If the statement is longer than X characters it will scroll to the next line.

3. Printing speed control- POKE 32719,X

The larger the X the slower the print speed. The slowest allowed value of X is 255. For 8K memory use 8143 and for 16K memory use 16635.

TAPE OUTPUT SCANNER- from J.Burns, 2160 Market St. Rm. 45, San Francisco, CA, 94114

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 SA 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 order, 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).

(continued on page 54)

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 fot 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
```

Editors Note: An article describing how to make CLOADG work is in the July 1980 Sorcerer's Apprentice, Vol.2, No. 3. An article describing a way to use CLOADG will be included in S.U.N., Volume II, June 1980.

SORCERER USER'S NEWSLETTER

Volume II, Number 6, Page 55

JUNE, 1980

The Sorcerer User's Newsletter is published by the Sorcerer's Apprentice, P.O. Box 1131, Troy, MI, 48099

Elementary Program Chaining

by Bryan Lewis

Some time ago I devised a way to load and run a machine-language program without leaving BASIC. I wanted to be able to load utility routines (for example, a screen editor or a printer driver) with the CLOAD and RUN commands. That way the rest of my family could set up the system without having to learn anything about the Sorceror Monitor.

My method works, and I'll describe it in a moment, but first I want to show a better way, one I learned only recently from the July Sorceror's Apprentice (thanks to the Australian users). BASIC can chain (that is, load and run automatically) either machine-language or BASIC programs, using the CLOADG command. Simply put the CLOADG command in your first BASIC program, as follows:

```
1000 CLOADG
```

That statement will load and run whatever program is next on the cassette. One surprise is that the name of the desired program doesn't have to be included in the command, as it does when you give the CLOADG command in the direct mode. Another surprise is that this will chain a machine-language (ML) program just as well.

The program to be chained must be auto-executable. For a ML program, that means you must have saved it with an execution address, using the monitor command: SE X = nnnn. For a BASIC program, follow the procedure in the July article on CLOADG.

There's one loose end: how do you chain from a ML program? How do you get back to BASIC? That's where the trick I devised originally comes in. Put the following five bytes at the end:

```
AF F5 C3 99 E7
```

That will jump to the tape-loading routine in the Sorceror Monitor, at address E799 hex. But first it pushes a zero flag onto the stack; that tells the Monitor to run the program after loading it, precisely as if you had given a LOG command.

If you wish to run a series of programs in sequence without user intervention, end each one with the proper chain command. The programs should be saved on tape in the desired order. If, however, you want the user to select the next program, as from a menu, then the CLOADG command has to include a file name. The obvious and logical way to do that is: INPUT F$: CLOADG F$. Unfortunately that doesn't work; you must provide the file name explicitly. So for a menu selection you might do the following:

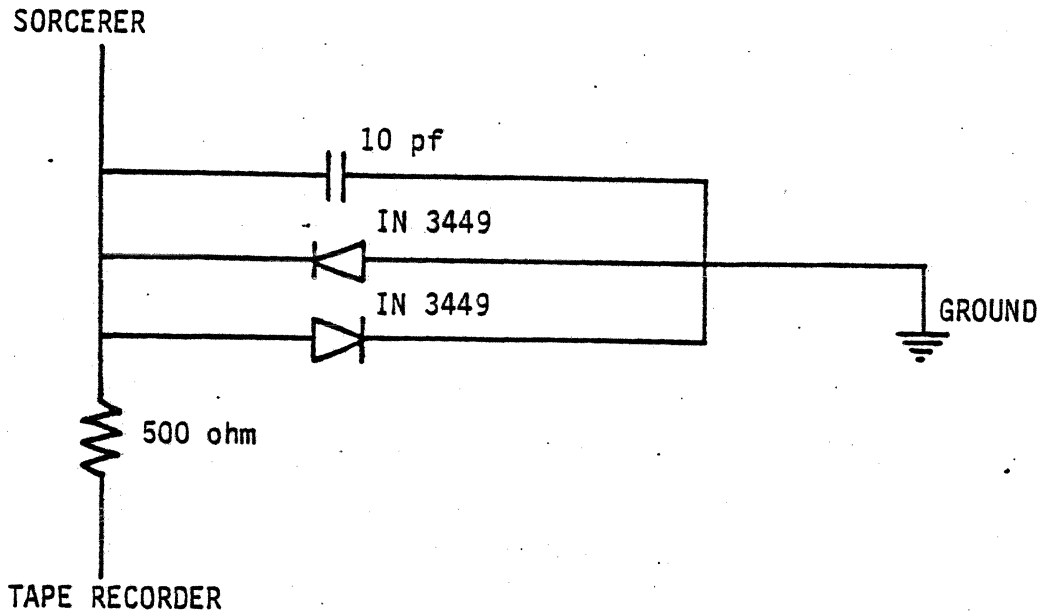
```
1000 INPUT "Which game would you like to play"; F$
1010 IF F$ = "CRAPS"      THEN CLOADG CRAPS
1020 IF F$ = "OTHELLO"    THEN CLOADG OTHLO
1030 IF F$ = "BACKGAMMON" THEN CLOADG FGAM
```

IMPROVING RECORDER RELIABILITY- from Terry Calvert

From TRS-80 MICROCOMPUTING: a way to get a tape recorder to function better for loading is to place two diodes across the "EAR" line to the Sorcerer to limit the volume. By using a 10pf capacitor in the circuit shown my Panasonic RQ-413S now works great with my Sorcerer.

I hope this helps with someone else's problems. A problem I need help with is:

When I use my SHIFT LOCK the letters p,t,y,u,2 are printed lower case instead of the desired shifted character. Anyone have any suggestions?



SORCERER'S APPRENTICE USER GROUP PROJECTS-

Details on subscriptions to the Sorcerer's Apprentice can be found on page 52. Subscribers automatically become members of the SA User Group and as such can use the services offered by the group. Reports on special projects will be available through the group's library. Some of the special projects are:

1. Relocating the WP PAC and the DEV PAC and recording them on cassette so that they can be used at the same time BASIC or another PAC is in the machine. This has been done by the User Group in Doncaster, Australia. I have a copy of the documentation and I have made a copy of my relocated WP PAC.
2. Running TRS-80 software in the Sorcerer- The TRS-80 operating system and Level II BASIC have been transplanted into a Sorcerer via a cassette recorded, modified copy and TRS-80 software can be run with it. Loading tapes is a problem because they are recorded at 500 baud. A hardware project is being tested to accomplish this.

Documentation on these projects will be available later this year. A library catalog will be sent to SA subscribers as it becomes available.

SIMULATION GAMING AND WAR GAMES- from Adrian Pett, 10 Burgundy Drive, Doncaster, Victoria 3108, Australia

Dear Editor,

You should shortly receive two items in the mail; the September issue of the "Lone Warrior" and a collection of photocopies. The photocopies are of articles dealing with the subject of simulation gaming, wargaming, military simulations, hobby gaming with respect to the microcomputer.

Among the articles is a review of "Computer Bismark" and for comparison two reviews of the Avalon Hill boardgame "Bismark". Also included is a review of some of the Science Fiction and Fantasy games available, which may help to explain why some of the writers consider the majority of existing computer games to be of a lower standard. Most of the other articles are about the possibilities of using microcomputers to enhance simulation gaming.

"Lone Warrior" is the Journal of the Solo Wargamers' Association and could prove to be of interest due to the ideas presented for workable solitaire systems. Coincidently this particular issue also contains an article that I wrote called "Wargame + Microcomputer = ?".

It would be appreciated if you could inform me whether any of your readers are using their microcomputers to enhance simulation gaming and if so what approaches are they taking?

Editors Note: Copies of the articles mentioned by Adrian are available from the Sorcerer's Apprentice library for the cost of copying and postage. If you are interested in this aspect of computing write to Adrian direct and send a note along to the Sorcerer's Apprentice and we will try to include more articles of this type in future issues.

AVALON HILL WAR GAMES- by L.Kobylarz, Editor, Sorcerer's Apprentice

After receiving Adrian Pett's letter (see above article) on simulation gaming, I found war games for computers by Avalon Hill in a local hobby store which sells boardgames and periodicals of simulation games, war games, and fantasy games (Dungeons and Dragons). Titles included were:

B1 Nuclear Bomber
Midway Campaign
North Atlantic Convoy Raider
Nuke War
Planetary Miners

The price was $15.00 each and they were packaged in a "bookshelf" style box with instructions and one cassette with the program recorded once for TRS-80, PET, and Apple on the same tape. I intend to get one of the games and convert the TRS-80 version for the Sorcerer. I will write a review in a future issue of the S.U.N. or Sorcerer's Apprentice.

USR Routines without Poking
by Bryan Lewis

To include machine-language USR routines in a BASIC program, the usual method is to POKE the bytes into memory. A brief example:

```
10 FOR ADDR = 0 TO 2     :REM Put in low memory.
20 : READ BYTE
30 : POKE ADDR, BYTE
40 NEXT ADDR
50 DATA 195, 3, 224      :REM In hex, C3 03 E0.
```

That method is not very elegant, though. Your program listing gets cluttered, and there's an unnecessary delay each time the program runs.

At least a couple of alternatives exist. The best one is to store the machine-language routine on tape at the same time as the BASIC program. You can't use the CSAVE command for that -- it saves only the BASIC area of memory beginning at 01D5 hex. Use the monitor's SAve command instead, to write to tape the entire block of memory from 0000 up. Like so:

```
Load the program and USR routine by any method.
RUN              To set up the correct parameters in the BASIC control
                 area, 100-1D4 hex.
<CTRL-C>         Interrupt the run if you wish.
BYE
>DU 1B7-1B8      To find the end of the program area. 1B7 contains the
                 low byte of the address (call it LL), and 1B8 the high
                 byte HH.
>SA <NAME> 0 HHLL
```

The beauty of this is that a later CLOAD command will restore the whole thing. No going back to the monitor and no POKEs.

If you've been saving your Exidy BASIC programs on disk, you're already using the block-memory-saving method. You certainly should remove any USR poking loops.

A second alternative is to store the bytes right inside the BASIC program, in the non-executable space provided by a REM statement. Watch this:

```
1 REM <bunch of spaces here>
2 <rest of program...>
BYE
>DU 1D0-1FF
    You'll see the byte 8F (the token for REM) followed by the bunch of
    spaces, 20's. If you've typed it just as above, the spaces will
    begin at 01DA.
>EN 1DA
    Enter the machine-language routine in place of the spaces.
>PP
```

Your program can now use 01DA for the USR address: POKE 260,218 : POKE 261,1. If you LIST the program, you'll see some strange characters in line 1.

A couple of warnings:

(1) Don't alter the REM line or any line before it, lest you move the USR starting address. That's why I put it as early in the program as possible.

(2) The USR code can't contain any 00 bytes. BASIC will regard the 00 as an end-of-line marker.

Because of these difficulties, this method is really useful only when you've run out of room below 100 hex.

SORCERER USER'S NEWSLETTER

Volume II, Number 7, Page 59

July, 1980

The Sorcerer User's Newsletter is published by the Sorcerer's Apprentice, P.O.Box 1131, Troy, MI, 48099

DEVELOPER

No. 1

by Bryan Lewis

This new column will demonstrate the use of the Sorceror's Development Pac. It will start by showing you how to write assembly language programs. In theory it shouldn't be necessary to show you how, once you've bought the Pac and read its User's Manual, but the manual is not all that easy to understand if you've never been exposed to assembly and machine language programming before. This column will also demonstrate the other features of the DevPac, and will point out a few minor bugs and how to get around them.

There's a lot of ground to cover in this first column. I'd like to show a concrete example of a program written with the DevPac. In order to get to a real example, I'm forced to assume that you have some familiarity with machine language. You should know how to enter hexadecimal code with the monitor's ENTER and GO commands. You should be aware of the purpose of an assembler -- allowing you to think in terms of mnemonics and to use labels and symbols, both of which save you from having to use that same hex code. You should know that there are many useful routines already written in the Sorceror's monitor.

If you aren't comfortable with those ideas, get yourself a book on Z-80 programming, such as Barden's "The Z-80 Microcomputer Handbook." Follow Dave Bristor's column on the Exidy Monitor. Once you get deeper into it, you'll especially want to have Exidy's "Sorceror Software Manual," which gives a complete listing of the Power-On Monitor.

I've found that an excellent foundation for assembly language programming is experience with a programmable calculator. Such a calculator is in effect an instant assembler; it takes your keystrokes (labelled with English-like mnemonics) and converts them to numbers (instruction codes) in memory. Your thinking processes are similar as well; every variable must be deliberately saved in some location, for example.

Enough introduction. I apologize to readers who are already bored.

The first several programs will be input/output drivers. They're perfect for the purposes of this column: useful and short. With I/O drivers you can change almost any idiosyncrasy of the Sorceror that annoys you. You can customize the keyboard, for instance, programming individual keys with special functions.

Let's write a short I/O driver with the DevPac. Take the simple case of disabling the CLEAR key. (Let's say you want to make sure you can't mess up a graphics-oriented game by accidentally clearing the screen.) This will be an

input driver; whenever the CLEAR key is pressed, we want to ignore it. In "structured English," our super-simple program is:

```
Check the keyboard for input.  Was a key pressed?
   If not, keep checking.
   If so, was it CLEAR?
      If not, return with the value of the key.
      If so, ignore it; return with nothing.
```

Now let's rephrase this in assembly language:

```
UNCLR   CALL    KEYBRD  ;LOOK FOR A KEY FROM KEYBOARD.
        JP      Z,UNCLR ;IF NO KEY PRESSED, KEEP LOOKING.
        CP      CLEAR   ;IS IT CLEAR?  SET ZERO FLAG IF SO.
        RET             ;DONE; RETURN TO WHATEVER CALLED US.
```

Note the label UNCLR in the left column. This is for our convenience, so that we can write an instruction like JP Z,UNCLR without worrying about the hex address. The assembler will figure that out for us.

The next column contains the instructions. You should have no trouble understanding them with the help of the above English version and the comments in the right-hand column. Clear comments are almost mandatory in assembly language programming, again for YOUR convenience in later debugging or modifying. Note that the comments begin with a semi-colon; that key will be recognized by the assembler as a comment marker, just as Basic recognizes REM.

Four more loose ends (or nasty details, if you wish):

(1) The Development Pac can't know what the English words KEYBRD and CLEAR mean. We have to tell it. That's the purpose of the EQUate instruction:

```
KEYBRD  EQU     0E018H  ;PREDEFINED ADDRESS IN MONITOR.
CLEAR   EQU     0CH     ;ASCII VALUE OF THE CLEAR KEY.
```

To be precise, these lines are called pseudo-operations, or pseudo-ops for short. They're "pseudo" because they won't result in any machine code; they're only there for (you guessed it) your convenience.

We've used hexadecimal values for these constants, and we said so by adding an "H" to the ends. If you prefer, you can use decimal numbers instead; no suffix is necessary then, such as: CLEAR EQU 12. Also note the leading zeroes. The assembler demands a digit, not a letter, as the first character in a number.

(2) The assembler allows a program section to use absolute addressing (simpler) or relative addressing (more flexible). We'll get into the reasons for this later, but for now we only need to know that we need another pseudo-op:

```
        PSECT   ABS
```

This tells the assembler that the Program SECTion uses ABSolute addressing.

(3) We also have to tell the obedient assembler where to put the program. I should say, where to put the origin of the program. Let's start at the bottom, location 0000. This area is out of the way of any other programs we might have in memory. The pseudo-op is:

```
ORG     0000H
```

or simply ORG 0, the same thing.

(4) Back to the idea of clear documentation. We will (in a moment) preface the program with a description of its purpose, how it works, what comes in, and what goes out. Again, we'll use semicolons to signify comments.

FINALLY, we can turn on the Sorceror, with the Development Pac inserted. We can't just start typing at once, however, as we could if this was the Word Processor Pac. We must specify what kind of input and output the Assembler will use -- one more instance of a slight loss in simplicity for the sake of flexibility. We do that as follows, with our entries underlined:

```
.M :SI
:SI xxxx  :BI
:SO xxxx  :SV
xxxx  xx  .
```

Translation: the first period is the DevPac's prompt character. We enter M (the command to examine memory) and :SI (the predefined symbol for source input), then a carriage return. The Sorceror responds with the information that the address assigned for source input is some random garbage, whatever was in memory after we turned the unit on. We correct this by typing :BI, the symbolic address for the "B" input buffer. In other words, our source code will be held in memory (not read in from tape).

Next we specify the Source Output address in a similar manner, selecting :SV for Sorceror Video. That is, just display the code on the screen without bothering to save it on tape.

That's all we need now. On the next line (more garbage) we call it quits by typing just a period.

Now, we invoke the text editor:

```
.E :ED
```

These symbols stand for Execute EDitor. The Sorceror responds with an asterisk, the editor's prompt. We tell it we want to Insert text by typing I, even though we had no previous text to insert into. No prompt character appears this time, just a blank line waiting for us to type.

We type in our complete source program now: prefatory comments, pseudo-ops, and instructions. (There's no room in this first article to go into the editing commands, so I'll assume you've read and understood that section of the User's Manual. That chapter is the easiest to follow, by the way, because it gives an illustrative example.)

```
;
;****************   UNCLR   ********************
;
;THIS INPUT DRIVER SIMPLY IGNORES THE CLEAR KEY.  IF
;CLEAR IS PRESSED, THIS ROUTINE SETS THE ZERO FLAG, AS
;IF NOTHING WAS PRESSED.
;INPUT TO THIS ROUTINE:  NONE.
;OUTPUT:  A CHARACTER IN THE A REGISTER, OR ELSE A SET
;ZERO FLAG IF THE CLEAR KEY WAS PRESSED.
;
;
        PSECT   ABS
        ORG     0000H
;
KEYBRD  EQU     0E018H  ;PREDEFINED ADDRESS IN MONITOR.
CLEAR   EQU     0CH     ;ASCII VALUE OF THE CLEAR KEY.
;
;

UNCLR   CALL    KEYBRD  ;LOOK FOR A KEY FROM KEYBOARD.
        JP      Z,UNCLR ;IF NO KEY PRESSED, KEEP LOOKING.
        CP      CLEAR   ;IS IT CLEAR?  SET ZERO FLAG IF SO.
        RET             ;DONE; RETURN TO WHATEVER CALLED US.
```

Whew! At least we've finished a real example of a source program. The details of assembling and loading this will have to wait till the next column. Just so you won't be left hanging, the end result is the following hexadecimal machine code starting at location 0000:

CD 18 E0 CA 00 00 FE 0C C9

Nine bytes! That shows you how much of all that text really was just for convenience.

If you want to test our little program, ENter it via the monitor, then assign it as the input driver: SET I=0. Your CLEAR key will be kaput.

SORCERER'S APPRENTICE
P.O. BOX 1131
TROY, MICHIGAN 48099

!!! SUBSCRIBE NOW !!!

To ensure receipt of the 1981 SORCERER'S APPRENTICE newsletter (vol. 3) send this application form along with $12 (USA and Canada), $16 all others (payable in US funds). This also entitles you to a one (1) year membership in the SORCERER'S APPRENTICE Users Group. (Make checks/money orders payable to: SORCERER'S APPRENTICE)

NAME: _____
(last) (first) (middle)

ADDRESS: _____
(number) (street) (apt. #)

_____
(city) (state/nation) (zip/postal code)

PHONE NUMBER: (H) _____ ; (W) _____
(area code) (number) (area code) (number)

Your responses to the following information will be held in confidence. Please circle the appropriate responses where applicable.

WHICH OF THE FOLLOWING DO YOU HAVE?:

Sorcerer: Model I II ; 8K 16K 32K 48K >48K ; Expansion Unit

Disk system: _____ ; CP/M : Printer: _____

Other peripherals? _____

Do you consider yourself a:(beginner Intermediate Expert) computerist?

Is your primary interest: Hardware Firmware Software ?

Is your primary application: Business Personal both ?

What types of software most interests you? _____

_____

What topics/articles would you most like covered? _____

_____

Comments about the newsletter: _____

_____

_____

The SORCERER'S APPRENTICE is published eight times a year by the SORCERER'S APPRENTICE USER GROUP. The subscription rate for the United States and Canada is $12.00 (U.S.) per year and $18.00 (U.S.) for other foreign. Subscribers automatically become members of the User Group which entitles them to the additional services of library, on-line bulletin board, technical bulletins, etc. as they become available.

Prospective advertisers, please contact Thomas E. Bassett, Advertising Manager in C/O this publication. Classified ads are accepted at the rate of $1.00 per line.

Newsworthy items can be submitted as hand written copy, typed copy or on word processor cassettes (either of the latter two preferred). Original material will be returned if requested and accompanied by SASE.

BACK ISSUES: SUN (Steve Long's Newsletter Volume I complete) $10.00; THE SOURCE issues 1-5 (published by ARESCO) $7.50 SORCERER'S APPRENTICE $2.00 each issue and Volume I $10.00 (limited supply on all items).

NEXT MEETING: 7:00 p.m., November 20, 1980 at Lyceum, Inc. Microcomputers, 28657 Hoover Rd., Warren, Michigan 48093, just one mile North of I-696, easy on, easy off.

SEND ALL CORRESPONDANCE TO:

SORCERER'S APPRENTICE
P. O. Box 1131
Troy, Michigan 48099