AUTOWRITER

from

Mr T T.M.
Software

WARRANTY - If the original AUTOWRITER disk or data pack fails to function properly due to defective materials within 30 days of purchase, it will be repaired or replaced if returned to the place of purchase by the original purchaser.

I N T R O D U C T I O N

After spending nearly two hours one day looking through old newsletters and scanning old programs in search of some routines I wanted, the idea to create AUTOWRITER was born. What I needed was a program that wrote the routines I wanted without the hassle of looking for them and typing them in. A program that writes programs! All of a sudden I had a new project underway.

Although this software and documentation is meant for the person with some knowledge of programming, I tried to design them both so the beginner (who has read the SmartBASIC manual) could make good use of the routines and put a little fun into learning to program.

I must thank two organizations for their contributions and help not only in the developement of AUTOWRITER but BASICaide, TRIVIAPACK and others:

Digital Express, Inc. of Oak Hill, WV
and my original encouragement
Lyle Marschand of NIAD in Lisle, Illinois

If you have any comments, questions or suggestions, please write to:

Bob Tarnowski

Mr.T. SOFTWARE
7316 Northway Drive
Hanover Park, Il 60103

TABLE of CONTENTS

OVERVIEW

AUTOWRITER is a user friendly program that writes basic subroutines and machine code routines to a user designated storage medium. The user determines the starting line number and increment then selects the routines to be saved. After inputing a filename, the program is written to the disk or data pack as an "A" file that can be merged with existing programs or used as a base for a new creation.

AUTOWRITER will also access a file of RAM addresses which can be displayed on-screen or printed on the ADAM printer with useful suggestions, tips, fixes and enhancements that may be obtained through POKEs and CALLs. AUTOWRITER will search the file for key words such as "POKE" or "CATALOG" and display or print only those addresses that contain the key word.

LOADING AUTOWRITER

AUTOWRITER will only work with SmartBASIC V1.0. If you already have SmartBASIC V1.0 in memory, you can load AUTOWRITER by entering "RUN HELLO". If SmartBASIC V1.0 is not in memory, then pull the <RESET>. Since AUTOWRITER uses various areas of RAM to store data and changes some EOS routines, ADAM may lock up by RUNning HELLO after you have used certain utility programs. If this occurs, just pull the <RESET>. Your AUTOWRITER contains a corrected version of SmartBASIC V1.0.

SETTING THE CLOCK

The first of AUTOWRITER's two main programs sets the LOMEM to 27866, draws the title screen, pokes various routines into RAM and prompts you to set the on-screen digital clock. You're asked to indicate "am or pm", the hour, minute and second. By pressing <RETURN> at the first prompt (am or pm?), the clock setting routine is aborted and all counters are set to 0 (00:00:00am). Pressing <RETURN> at the prompt for hour, minute or second sets THAT counter to zero. The second program is then loaded and the clock is displayed. The clock/timer display is set when a key is pressed during most of the second program.

BEGINNING LINE NUMBER

After the second program is loaded, you are prompted to "ENTER THE BEGINNING LINE NUMBER FOR YOUR ROUTINES:". The entry can be any number from 10 to 60000; any other number will be rejected. You are then asked to "ENTER THE NUMBER TO INCREMENT LINES:" which must be a number from 1 to 50. For example, if your beginning line number is 20000 and the increment is 10, the first three lines will be 20000, 20010 and 20020.

AUTOWRITER MAIN SCREEN

The main control screen for AUTOWRITER is accessed from nearly any point in the program by pressing <ESCAPE> or ^C (control and C). The header (first three lines) remains on-screen throughout the program and is updated at nearly every keypress.

AUTOWRITER MAIN SCREEN

```
04:21:32 pm           LOMEM: 27407
# BYTES: 0            FREE RAM: 6965
# LINES: 0            NEXT LINE: 20000

      > > > Press SmartKey < < <

          I    MACHINE CODE ROUTINES
         II    BASIC ROUTINES
        III    POKES and CALLS
         IV    SAVE LINES IN MEMORY
          V    PRINT OPTIONS
         VI    QUIT

   I     II     III     IV      V      VI
 Mcode Basic   Pokes   Save   Print   Quit
```

The CLOCK displays the current time when a key is pressed. If the clock reset was aborted, the clock becomes a timer letting you know how much time you spent creating your routines or searching the Pokes and Calls.

The LOMEM indicates the lomem setting that would be written into your subroutine program if you were to save the routines at that point.

BYTES indicates the length of the routines currently stored in memory in bytes and gives you an idea of how long your program would be if stored at that point. 1024 bytes can be stored in each block on your disk or tape.

FREE RAM lets you know how much memory is left for storing your routines. AUTOWRITER will not let you store any more routines after the FREE RAM goes below 600.

LINES indicates the number of lines currently stored in memory.

The NEXT LINE indicator lets you know the line number that will be used to start storing the next routine. If you intend on merging your new program with an existing one, use this indicator to prevent duplication of line numbers.

I MACHINE CODE ROUTINES

Pressing this SmartKey clears the center screen and begins the display of machine code routines one at a time. At the bottom of the screen, you will be given the following options:

```
     I       II      III      IV       V       VI
   Store                     Pass             Done
```

If you wish to store the machine code routine indicated in the center of the screen just press Roman numeral "I". By pressing "IV", the screen is cleared and the next routine is displayed. After all the routines have been displayed, you are returned to the main screen. Pressing "VI" at any point will also return you to the main screen. If you attempt to store more routines than memory can hold, you will be advised to save the routines currently in memory.

After each routine is stored, the header will be completely updated. If the routine stored does not use any addresses above lomem, LOMEM will not change. Keep an eye on the FREE RAM. If you wanted to store all of the routines in AUTOWRITER, you would have to create two programs and merge them.

II BASIC ROUTINES

Follow the same procedures for handling these routines as described above for machine code routines. It will be easier to work with your new program if you do not intermingle machine code and basic routines. Ideally, the basic routines would be grouped first followed by the machine code routines.

III POKES and CALLS

Pressing "III" at the main screen will clear the screen and offer the following options at the bottom of the screen:

```
I     II     III     IV       V      VI
                   search   list   escape
```

Pressing "VI" takes you back to the main screen. If you wish to view the entire list of addresses, press "V". A message will tell you to "press SPACEBAR to pause", AUTOWRITER will access the file, then the addresses will begin scrolling upward in the center of your screen. At any point, you may press "VI" to stop the display and close the file.

If you press "IV", you will be asked to "ENTER THE WORD FOR SEARCH:". The first time you use AUTOWRITER, you should view the entire list of addresses to see how it is put together. This will help you determine what words to search for to find an applicable address. Key words to try are POKE, CALL, BUFFER, FIX, SCREEN, COLOR, etc. You can pause the display by pressing the SPACEBAR or close the file and exit the routine by pressing "VI" escape.

IV SAVE LINES IN MEMORY

Pressing "IV" during the main screen display will clear the screen and ask you to "ENTER FILENAME FOR ROUTINES". At the prompt you can enter up to 10 letters or numbers for your program. If you attempt more than 10 characters or you use illegal characters your entry will be rejected. After entering your filename, the bottom of the screen will indicate those drives containing a disk or tape as follows:

```
  I      II     III     IV      V      VI
Disk1  Disk2  Tape1  Tape2   Scan   Escape
```

If a drive does not contain a medium, there will be a blank below the appropriate Roman numeral. If you change a medium or add one to a drive, press "V" to scan the drives and reset the bottom display. YOU CANNOT STORE ROUTINES ON YOUR AUTOWRITER MEDIUM. This is to prevent the accidental overwriting of any of the programs or files. If you have only one drive, you will have to replace the AUTOWRITER medium with one that has enough room to store your routines. If you prefer to abort this function, press "VI" to return to the main screen, otherwise, press the SmartKey that corresponds to the drive that contains the medium for storage.

After selecting the drive, you are given the following options at the bottom of the screen:

```
  I     II     III     IV     V     VI
Write                               Escape
```

To abort, press "VI" Escape. Pressing "I" will write the routines in memory to the selected medium. Any I/O error will return you to the main screen. It is suggested that you check your storage medium BEFORE booting AUTOWRITER to make sure it has enough room (up to 6 blocks).

<u>V PRINT OPTIONS</u>

Pressing "V" at the main screen will the provide the following options:

```
  I     II     III     IV     V     VI
Memry Pokes                         Escape
```

PUT PAPER IN YOUR ADAM PRINTER FIRST then select the option of choice. Pressing "I" will start printing the routines in memory. To pause the printing, press the SPACEBAR, to abort the printing process, press "VI" at any point. ADAM will finish printing the line and exit to the main screen. If you press "II", you will be given the following options:

```
  I     II     III     IV     V     VI
                     search  list  escape
```

Select the option in the same manner as you would to view the Pokes and Calls (see POKES and CALLS on page 8). Again, to pause press the SPACEBAR, to abort press "VI". If you abort, AUTOWRITER will finish printing the address information and exit to the main screen.

<u>VI QUIT</u>

Pressing "VI" will terminate the program IMMEDIATELY and leave you in basic, however, I suggest that you reboot basic before running any programs containing pokes and calls. AUTOWRITER changes many addresses in RAM that will conflict with other programs.

MACHINE CODE ROUTINES

These routines are written so you can GOSUB to the first routine at the beginning of your program. All programs written by AUTOWRITER contains the LOMEM and changes the poke limit in line 1:

```
       1  LOMEM :27407 :POKE 16149, 255 :POKE 16150, 255
```

The LOMEM is automatically set above any routines as required. In the following explanations of the routines, those addresses that are determined by AUTOWRITER are shown in brackets i.e. CALL [27407]. Those addresses which do not vary are written without brackets i.e. CALL 11707. Each program explanation is written as if the program is the only one saved. Each routine begins with a REM statement identifying the routine and indicating the actual POKE and CALL numbers.

REM INSTANT BACKGROUND COLOR CHANGE - POKE [27408],color: CALL [27407]

POKE the desired color (0-15) then CALL the routine for an instant background color change without clearing the screen. This routine is demonstrated by AUTOWRITER if an error occurs.

REM INSTANT TEXT/SCREEN COLOR CHANGE - POKE [27408],color: CALL [27407]

POKE the desired text/background color then CALL the routine. You can use the following formula where tc=text color and bc=background color: POKE [27408], tc*16+bc. The routine will instantly change the colors without clearing the screen.

REM INSTANT UNDERSCORED TEXT FONTS - POKE 17126, PEEK(17115): TEXT: CALL [27407]

The POKE changes the INVERSE colors to the same as NORMAL text colors. After CALLing the routine, all inverse fonts will be displayed as underscored. Entering TEXT restores INVERSE fonts to standard form.

REM SOLID COLORED INVERSE FONT BLOCKS - CALL [27407]

CALLing this routine enables you to create colorful text screens by changing the INVERSE fonts to solid colored blocks. Enter TEXT to return to normal fonts.

REM CHANGE CURSOR TO SOLID BLOCK - CAll [27407]

CALL this routine to instantly change the cursor to a solid block.

REM 40 COLUMN TEXT - CALL 11740 / 31 COLUMNS - CALL 11707

To go to 40 column mode, just CALL 11740 in the immediate mode or within a program. The text/background color is determined by the value in 17059 which is the address for background color. Be sure to poke a text/background color into 17059 before CALLing the routine. For example, POKE 17059,(31, 33, 245, or 79). CALLing 11707 will put you back in 31 column mode where the colors are determined by the values in 17115 for NORMAL and 17126 for INVERSE. In the 40 column mode, if you want to HTAB beyond column 31, you'll have to POKE 26198,39 - default is 31.

REM 31 COLUMN INSTANT INVERSE COLOR CHANGE - POKE [27408, color]: CALL [27407]

After POKEing the color and CALLing the routine, all inverse fonts are instantly changed to the new color without clearing the screen.

REM 40 COLUMN INSTANT INVERSE FONTS - CALL 200

AFTER CALLing 11740 to get into 40 column mode, CALL 200 to enable inverse fonts. The font color and background will be the inverse of the NORMAL 40 column colors. In other words, if NORMAL is black letters and cyan background, INVERSE will be cyan letters and black background like the AUTOWRITER main screen.

REM PRINT STRINGS IN HGR

This machine code routine includes a basic subroutine that is jumped over during the initial set up. To use the subroutine, find the line number that begins with "INPUT txt$:..."; let's say that line number is 20030. To print in HGR or HGR2, assign a horizontal value to ht and a vertical value to vt then GOSUB the routine. Example:

```
     100  vt = 10: ht = 4: GOSUB 20030
```

A question mark (?) will appear (in HGR). After entering the input, enter <RETURN> and the fonts will be drawn on the hi-res screen.

A more practicle method to use in your programs would be to assign a string to the variable "txt$", omit "INPUT txt$" from line 20030 then GOSUB the routine. Example:

```
     100  vt=10: ht=4: txt$="Mr.T. SOFTWARE": GOSUB 20030
```

This routine is used in the AUTOWRITER title screen.

REM AUTOMATIC RANDOM NUMBERS (RND)

Immediately after this routine is poked into memory, the RND function will provide true random numbers every time the program is run. (This routine must not be used when the on-screen digital clock routine is used.)

REM FAST CLEAR OF RAM ADDRESSES - na=start address nb=# of addresses to clear nc=fill value

This machine code contains a basic subroutine which is jumped over in the initial set up. To use, assign the appropriate values to the above variables and GOSUB to the basic routine. We'll use 20030 for the line number of the basic routine for the following example:

```
     100  na=28000: nb=1000: nc=255: GOSUB 20030
```

In the example RAM addresses 28000 to 28999 would almost instantly be changed to contain the value 255.

REM SCAN DRIVES FOR MEDIA - CALL [27407]

This routine determines the status of the four ADAM drives. Addresses 65532 to 65535 are used to store the status value of each drive after the routine is CALLed.

```
        65532=DISK 1 status     65534=TAPE 1 status
        65533=DISK 2 status     65535=TAPE 2 status
```

PEEK the above addresses to find the following values:

1	indicates the drive does not exist
155	indicates the power in the drive is off
255	indicates there is no medium in the drive
*	if a medium is in the drive, the drive # will be found: 4=Disk 1, 5=Disk 2, 8=Tape 1, 24=Tape2

This example determines the status of Tape drive 1:

```
   100  CALL [27407]: If PEEK(65534)=8 THEN PRINT
        " THERE IS A TAPE IN DRIVE 1"
```

This routine is used in AUTOWRITER to scan the drives before saving your new program.

REM SOUND ROUTINE #1 (PROMPT) - CALL [27407]

After the initial setup, just CALL the routine to get the same prompt sound as is used by AUTOWRITER.

REM SOUND ROUTINE #2 (KEYCLICK) - POKE [27408],note: POKE [27422],duration: CALL [27407]

This routine used as is will return a keyclick sound when the routine is CALLed. The routine can, however, be customized to fit many needs by POKEing the noted addresses with different values. Experiment with different notes and durations (0-255). Example:

```
   100  POKE [27408],35: POKE [27422],40: CALL [27407]
```

REM SOUND ROUTINE #3 (NOISE) -POKE [27408],note (224-240): POKE [27412],volume (240-255): POKE [27417],duration: CALL [27407]

This is another routine you can customize as with SOUND #2. With all the sound routines, if something goes awry and the sound doesn't turn off just CALL 64851 in the immediate mode or within a program to turn all sound off. This routine makes the badkey sound in AUTOWRITER.

REM TO RESTORE SPECIFIC LINE NUMBERS - RESTORE x

This routine creates a new command "RESTORE" which enables your program to restore a specific line number containing a data statement. The best explanation for this one is an example:

```
          10  DATA AUTOWRITER
          20  DATA from
          30  DATA Mr.T. SOFTWARE
          40  FOR x=1 TO 3: READ a$(x): NEXT
          50  RESTORE 30
          60  READ a$(4): PRINT a$(4)
```

Using this routine does have a drawback - you can't use RESTORE without a line number but the pluses are tremendous.

REM SHOW DELETED FILES IN CATALOG

After RUNning this routine, all deleted files on your media will appear in the CATALOG with an inverse "D" to the left of the file type. The Blocks Left also reveals the actual number left.

REM ONSCREEN REALTIME DIGITAL CLOCK - CALL 27700

This is the routine used by AUTOWRITER to display the clock/timer. Next to the last line in the routine is a line that begins with a REM statement - REM POKE 27600,dy: POKE 27601,hr... If the routine is run as is, the counters are set to zero and the clock effectively becomes a timer. To set the clock with the current time, first remove the REM in the program line. Before the initial setup for the routine, assign values to the following variables:

```
        dy = 0 for "am" or 1 for "pm"
        hr = the current hour (1 to 12)
        mi = the current minute (1 to 12)
        se%= the current second (1 to 59)
```

Then GOSUB to the routine and the clock will be set. To display the clock, CALL 27700. To update the display, CALL 27700. This routine also automatically results in real random numbers (RND).

The machine code routines end with a "RETURN". Again, for best results, group all machine code routines together following any basic subroutines and GOSUB the first machine code routine at the beginning of the program. After your program is complete, the REM statements can be removed to shorten the program.

BASIC SUBROUTINES

These routines are written so you can GOSUB to each subroutine as needed individually. In the following explanations of the subroutines, those addresses or line numbers that are determined by AUTOWRITER are shown in brackets ([20030]). Those addresses that do not change are not shown in brackets (POKE 12374,148). All subroutine explanation assumes it is the first subroutine starting at line number 20000. In your programs you create with AUTOWRITER, each subroutine begins with a REM statement indicating the actual POKE, GOSUB and GOTO numbers. For best results, BASIC SUBROUTINES should be grouped first ahead of machine code routines.

REM KEYBOARD ENHANCEMENTS

This routine changes certain controlkey functions to a single keypress. Control key functions (example: ^C requires <CONTROL> to be held down while <C> is pressed and causes a program "break") are changed as follows:

```
      ^N = <INSERT> key = moves text to right of cursor
      ^O = <DELETE> key = moves text to left towards cursor
      ^L = <CLEAR> key = clears screen
      ^C = <ESCAPE> key = causes a break in program
      ^S = <WILDCARD> key = causes a print pause
      ^P = <PRINT> key = prints screen on ADAM printer
```

This routine could be placed at the beginning of your program - just LIST the program, change the program line number and run the cursor across the program line and press <RETURN>, then delete the old line by entering the old line number and pressing <RETURN>. LIST the program again and the routine will be with the new line numbers. To use as is as a subroutine, add ":RETURN" to the end of the last line of the routine then GOSUB to the routine as desired.

REM SMARTKEY INPUT SUBROUTINE - RETURNS WITH WITH VARIABLE k%=1 to 6

To use SmartKey input, GOSUB to the first line of the routine (20010 GET k$...) as in the following example:

```
   100  GOSUB 20010: ON k% GOTO 100,200,300,400,500,600
```

If the value returned as k%=1 the program will goto line 100; if it is k%=2 then the program will jump to line 200 etc. The subroutine is set up so that if ^C or <ESCAPE> is pressed, the program will end. If any other key is pressed, the keypress is rejected with an error sound.

REM KEYBOARD INPUT CONTROL - vt%=VTAB ht%=HTAB sh%= MINIMUM LINE LENGTH lo%=MAX LINE LENGTH

First, the variables noted in the REM statement must be assigned values, then GOSUB to the first line of the subroutine. Starting at the VTAB and HTAB cursor position, lo% number of dashes appear on the screen and the cursor is at the first dash. You can then enter your line and press <RETURN>. If the line contains a ^C or <ESCAPE> was pressed the program will end. If the line is less than sh% or longer than lo% the input is rejected. Example:

```
   100  vt%=10: ht%=5: sh%=6: lo%=20: HOME: GOSUB 20010
```

REM PRINT LINES FROM CENTER SUBROUTINE - txt$=LINE TO PRINT vt%=VTAB

Assign a value to vt% and a string to txt$ then GOSUB the subroutine for an interesting print display. Example:

```
   100 vt%=4: txt$="AUTOWRITER (C) from Mr.T. SOFTWARE":
        HOME: GOSUB 20010
```

REM ALPHABETICAL SORT SUBROUTINE - it%=# of items st$(x)=character strings

Assign values to it% and st$(x) then GOSUB the sort routine. Example:

```
   100 DATA JOHN, PETER, MARY, CAROL, CINDY, BOB, TOM
   110 FOR i=1 TO 7: READ st$(i): NEXT
   120 it% = 7: GOSUB 20010
   130 FOR i=1 TO 7: PRINT st$(i): NEXT
```

NOTE - Remember, any time you add DATA statements, make sure they will be read in the proper sequence. If the program contains other DATA statements that are read at the initial set up, then the above DATA line would have to be on a line AFTER the other DATA lines.

REM ERROR TRAPPING SUBROUTINE - AT THE START OF THE PROGRAM - ONERR GOTO [20010]

By using this routine, if an arror is encountered in your program the program will branch to this routine, end and display the error number and the line the error occured in. To use, place the ONERR GOTO statement at the beginning of your program - example:

```
   10  ONERR GOTO 20010
```

That simple! If you don't want the program to necessarily stop, you could remove the END statement and replace it with - ...CLRERR: FOR delay=1 TO 3000: NEXT: GOTO (line #) The screen will display the error message for a moment then GOTO the line indicated.

REM CORRECT GR/HGR COLOR TABLES

This routine simply corrects the GR and HGR color table so it is the same as the TEXT table:

0 =	Transparent	8 =	Medium Red
1 =	Black	9 =	Light Red
2 =	Medium Green	10 =	Dark Yellow
3 =	Light Green	11 =	Light Yellow
4 =	Dark Blue	12 =	Dark Green
5 =	Light Blue	13 =	Magenta
6 =	Dark Red	14 =	Gray
7 =	Cyan	15 =	White

REM CHANGE STRINGS TO UPPER CASE - k$=STRING

This subroutine will take a string of lower case letters and convert it to upper case. Example:

```
   100  k$="There are 16 colors in ADAM": GOSUB [20010]
   110  PRINT k$
```

After your program is complete, the REM statements can be removed to shorten the program.

POKES and CALLS

ADAM has 65,536 memory locations or addresses that contain Z80 machine code routines, text data represented by ASCII number codes (65=A, 66=B, 32=space, etc.), and space available for programs. To see what value is in a location you would PEEK that location; example: PEEK(16953) would reveal the ASCII character for the cursor - 95 the underscore. You can see these values by PEEKing them in the immediate mode:

```
   PRINT PEEK(16953) - ADAM would respond with "95"
```

The POKE command is used to change the values in an address. The normal value for the text color at 17115 for instance is 240 (text color 15*16=240 plus the background color 0 transparent). To change the color, POKE a different value into that address:

```
   POKE 17115,33: TEXT
```

Now the screen is medium green text (2*16) with a black background (1).

To go to a machine code routine in memory from basic, you would CALL the routine, for example, CALL 11090 is the same as the HOME command. A WORD OF WARNING - THE SLIGHTEST ERROR WHEN WORKING WITH POKES AND CALLS CAN LOCK UP YOUR COMPUTER. Try to know what the results will be BEFORE POKEing or CALLing.

Since ADAM has only an 8 bit processor, each byte can only contain a value up to 255. In order to work with higher numbers, ADAM uses a system of high bytes and low bytes. This method is used in machine code routines for numbers like addresses. The low byte always preceeds the high byte and are calculated as follows:

```
   Lets take 16953 again - the cursor character address
   high byte = INT(16953/256) = 66
   low byte = 16953-(66*256) = 57
```

Now if 16953 were to be expressed in machine code, the first address would contain 57, the second would have 66. To convert it back, multiply the high byte by 256 and add the low byte: (66*256)+57=16953

The listing of POKES and CALLS in AUTOWRITER gives you all of the common addresses and then some plus many corrections, alternatives and enhancements to help you to really customize your programs

MERGING PROGRAMS

Let's say that you already have a program called "GOODPGM" that you would like to add some of AUTOWRITER's routines to. Let's say the last line in your program is 5300. Here's how to do it. Boot AUTOWRITER and enter 10000 for the beginning line number for your AUTOWRITER routines. Select your routines and save them to the same medium that contains "GOODPGM" and call the routines program "RTNS". Quit AUTOWRITER and reboot Basic.

Now load "GOODPGM", then in the immediate mode, POKE 6356,201 which prevents ADAM from erasing "GOODPGM". Then, load "RTNS" and LIST what you have in memory. You'll find AUTOWRITER's line #1 which contains the LOMEM, "GOODPGM" lines from 2 to 5300 and "RTNS" lines starting at 10000. POKE 6356,205 to restore the NEW command, put in a line at the beginning of your new program to GOSUB the new routines and SAVE "BETTERPGM".

NEW PROGRAMS

AUTOWRITER routines can serve as a great base for your new programs you are creating. It is suggested that your machine code routines be placed high in your program. In other words start them at line 30000 or so to give yourself plenty of room.

Basic routines run faster if they're placed towards the middle of the program. With that in mind, you may want to consider the following method for creating new programs. First boot AUTOWRITER and select the BASIC ROUTINES that you want to use and place them around line 5000. Save that program then reboot AUTOWRITER or RUN HELLO and set the beginning line for the machine code routines at 30000. Save the machine code routines to the same medium as the basic and exit AUTOWRITER.

Now boot Basic and load the BASIC ROUTINES into memory. The basic routines do not change the LOMEM setting so delete line # 1 and prepare to merge the MACHINE CODE ROUTINES (see page 17 - MERGING PROGRAMS). Next load the machine code program with the basic program. The net result will be a base where the basic subroutines start at line 5000 and the one time used machine code routines start at line 30000. Line # 1 will be from the machine code routines and will set the LOMEM as needed.

In any case it's very important to keep it simple by saving the basic subroutines ahead of the machine code routines. On page 19 is a sample program that demonstrates a product of AUTOWRITER. The lines added by the programmer are lines 10 through 100 and 20300 through 20330. Try writing the sample program to get used to AUTOWRITER and run it. Break it at some point to see how the error trapping routine works and when you're done, run a CATALOG with a medium containing deleted files.

Please direct any questions or comments to:
Bob Tarnowski
Mr.T. SOFTWARE
7316 Northway Drive
Hanover Park, Illinois 60103

SAMPLE PROGRAM

```
    1 LOMEM : 27461: POKE 16149, 255: POKE 16150, 255
   10 ONERR  GOTO 20100: GOSUB 20120: CALL 27419
   20 HOME: FOR i=1 TO 9: READ st$(i): HTAB 5:PRINT st$(i):NEXT
   30 VTAB 20: PRINT "    press any key": GET k$: CALL 27432
   40 it%=9: GOSUB 20070
   50 HOME: FOR i=1 TO 9: vt%=i+4: txt$=st$(i)
   60 GOSUB 20010: CALL 27432: NEXT
   70 FOR de=1 TO 2500: NEXT: POKE 27408, 79: CALL 27407
   80 GET k$: PRINT:PRINT CHR$(4); "catalog": POKE 27408, 241:CALL 27407
  100 END
20000 REM PRINT LINES FROM CENTER SUBROUTINE - txt$=LINE TO PRINT vt%=VTAB
20010 IF LEN(txt$)<=31 THEN  x9$=txt$:  txt$="": GOSUB 20040: RETURN
20020 FOR x=2 TO 32: IF MID$(txt$, x, 1)=" " THEN  y=x-1
20030 NEXT: x9$=LEFT$(txt$, y): txt$=RIGHT$(txt$, LEN(txt$)-y-1): GOSUB
      20040: GOTO 20010
20040 x9=LEN(x9$): IF x9/2<>INT(x9/2) THEN  x9$=x9$+" ":  x9=x9+1
20050 FOR z=1 TO x9/2: VTAB vt%: HTAB 16-z: PRINT LEFT$(x9$, z);
      RIGHT$(x9$, z); : NEXT: vt%=vt%+1: RETURN
20060 REM ALPHABETICAL SORT SUBROUTINE - it%=# OF ITEMS  st$(x)=STRINGS
20070 FOR x=1 TO it%-1: x2$=st$(x): z=x: FOR y=x+1 TO it%+1: IF
      x2$>st$(y) THEN  x2$=st$(y): z=y
20080 NEXT: x1$=st$(x): st$(x)=st$(z): st$(z)=x1$: NEXT: RETURN
20090 REM  ERROR TRAPPING SUBROUTINE - AT START OF PGM - ONERR GOTO 20100
20100 er%=ERRNUM(0): ln=PEEK(16124)+256*PEEK(16125): ln=ln-4:
      lx=PEEK(ln)+256*PEEK(ln+1)
20110 TEXT: PRINT "  ERROR: "; er%:PRINT "  LINE: "; lx: END
20120 REM INSTANT TEXT/SCREEN COLOR CHANGE -  POKE 27408,color: CALL 27407
20130 DATA 62,0,17,32,0,33,0,32,205,38,253,201
20140 FOR x=27407 TO 27418: READ ml: POKE x, ml: NEXT
20150 REM CHANGE CURSOR TO SOLID BLOCK - CALL 27419
20160 DATA 33,1,0,17,248,2,1,8,0,205,26,253,201
20170 FOR x=27419 TO 27431: READ ml: POKE x, ml: NEXT
20180 REM SOUND ROUTINE #1 (PROMPT) - CALL 27432
20190 DATA 6,24,62,128,211,224,120,211,224,62,146,211,224,17,
      0,3,27,122,179,32,251,5,16,234,62,159,211,224,201
20200 FOR x=27432 TO 27460: READ ml: POKE x, IN CATALOG -
20220 POKE 21256, 0: POKE 21257, 0: POKE 21403, 195: POKE 21404, 135:
      POKE 21405, 228
20230 DATA 203,127,40,4,62,42,24,7,203,87,202,160,83,62,196,195,218,46
20240 FOR x=58503 TO 58520: READ ml: POKE x, ml: NEXT
20260 RETURN
20300 DATA Houston Texas,Chicago Illinois, Ames Iowa
20310 DATA Los Angeles California,Detroit Michigan
20320 DATA Miami Florida,New Orleans Louisiana
20330 DATA Reno Nevada,Atlanta Georgia
```