STAR (Super TI Assembly Routines) Version 1.1 Revision 12/17/86

Instructions

INTRODUCTION:

STAR is a package containing 53 TMS9900 assembly language routines to be used in conjunction with your TI Extended Basic programs. The routines are accessed via the CALL LINK statement. These routines perform tasks that would be difficult or impossible to do with Extended Basic code. Because all of the routines are written in the native tongue of the computer, they execute extremely faster than they would if the code was written in Extended Basic. Also, these routines reside in an area of memory that is not used by your Extended Basic programs, so they do not reduce the maximum size of your programs, and, by performing tasks with one statement instead of several, STAR saves memory. Routines are provided in such areas as character sets, sounds, cassette control, color, screen access, VDP memory access, control, keyboard, character definitions, disk access, string handling, and text mode (40 columns).

FAIRWARE INFORMATION:

STAR was written by Michael Riccio, COM-LINK Enterprises, and was released to the public as Fairware. Fairware is the "try before you buy" concept. This is how it works: You get a copy of the program FREE, then you take it home and use the program. If you find the program useful, you send your payment to the address in the program. The program IS NOT FREE; the author is relying on the honesty of the public to pay for the program.

You may copy the program and distribute it as often as you wish, as long as you follow the following rules:

- NO FEE may be charged for the program. The only exception to this is a copying fee/disk fee, but ONLY IF this fee is charged by a User Group or similar organization as a standard library fee. This fee MAY NOT EXCEED $2. This fee is NOT a substitute for the Fairware donation.

- When the disk is copied, ALL FILES MUST BE INCLUDED. This includes the documentation, as well as any sample programs.

- The files MAY NOT BE ALTERED in any way. This includes the documentation and title screens. However, additional files may be added to the disk, such as expanded documentation or sample programs.

- STAR, its documentation, and sample programs are copyright (C)1986 by Michael Riccio. All rights reserved. Any violation of the above rules is a violation of the Federal Copyright Act.

The requested Fairware donation for this program is $10 to $15. You may send this amount, or any other which you think is fair. Keep in mind that this is not a high price to ask, considering that a comparable package would cost upwards of $50. Send payment to this address:

COM-LINK Enterprises
953 Fillmore Street
Phila, PA 19124

Please include a note explaining that you are paying for STAR.
IMPORTANT: MAKE ALL CHECKS PAYABLE TO MICHAEL RICCIO.

LOADING:

The program loads extremely fast through an Extended Basic LOAD program.

IMPORTANT: For the program to load correctly, files LOAD and LOAD1 must not be altered, and their names must not be changed. Line 1 of the program LOAD2 may be changed (this will be discussed shortly), but the filename must not be altered.

STAR requires a minimum system of the following to operate:

- TI-99/4A console (some routines will not work on a TI-99/4 console)
- Monitor or TV set and RF Modulator (a color set is preferred)
- TI Extended BASIC (STAR has not been tested with third-party versions)
- 32K Memory Expansion (WILL work with Myarc 128K/512K and with RAMDISK)
- TI Disk Controller (has not been tested with CORCOMP or Myarc controllers)
- One or more disk drives (any configuration)
- Cassette recorder and cable optional for cassette control routines.
- All other hardware is optional, but should work without conflicts.

To load STAR, place the disk in drive 1 (must be in drive 1). If you are not in Extended Basic, you may now enter it, and STAR will load automatically. If you are in Extended Basic, type RUN "DSK1.LOAD" from command mode and press ENTER (you can also have a running program execute this statement.)

After the program loads, you will see a graphic title screen. When you are done looking at it, press any key. You will now see the Fairware information screen. When you are done reading it, press any key. The screen will clear, and you will be placed in command mode of Extended Basic. No program will be in memory, so it will be just like you entered Extended Basic from the title screen. STAR is now loaded into low memory expansion, where it will stay loaded unless you execute CALL INIT or turn off the computer. Entering NEW, BYE, or pressing QUIT (Function-equals) will not affect STAR.

Later on in these instructions, you will learn how to modify the LOAD program so it will do something other than return to command mode (such as run a program that needs the STAR routines.) Now that STAR is loaded, you can learn how to use each of its 53 powerful routines.

PROGRAM ROUTINES:

IMPRORTANT: When the syntax is shown for a routine, it will follow these guidelines:

Items in CAPITALS are to be typed exactly as seen. Also, any punctuation except brackets ([]), ellipses (...), or dashes (-) is to be typed as seen.

Items in lower case describe a value, variable, or expression that is to be substituted where indicated.

Items in [brackets] are optional, and may be omitted.

Items followed by ellipses (...) may be repeated.

These are the same guidelines used in the Editor/Assembler manual.

All of STAR's 53 routines are accessed through the CALL LINK statement. This Extended Basic command has the following syntax:

CALL LINK("program-name"[,parameter1 ... ,parameter15])

The program-name is from 1 to 6 characters long, and identifies the routine you are accessing. Parameters 1 through 15 are optional, and vary from routine to routine. The program-name MUST be enclosed in quotation marks, and parameters MUST be separated by commas. The program-name and parameter(s) MUST be enclosed in parentheses. All program names are in capital letters, and two contain the numeral 1 in their names. You must type the program-name exactly as given for the routine to work correctly. Consult the error section of these instructions if you receive an error message when executing a routine.

All of the routines may be executed from a program or from command mode. However, upon return to Extended Basic, the computer resets many video items to the defaults, therefore you may not obtain the expected results. The following routines will not work as expected from command mode: LOW, LGCAPS, FLASH, SCROFF, GETSCR, POKEV, VDPREG, COLORS, MAG, INVERT, ROTATE, FLIP, MIRROR, COPY, TEXT, PRINT, and SCREEN. There is a way to defeat this problem. If you follow the CALL LINK with :: ACCEPT A$, and press FUNCTION-4 when the computer asks for input after the routine executes, the defaults will not be restored. For example, if you enter: CALL LINK("LOW") the lower case letters will be redefined briefly, but then Extended Basic will redefine them. If you enter: CALL LINK("LOW"):: ACCEPT A$ the lower case letters will be redefined, then Extended Basic will be waiting for your input. When you press FUNCTION-4, you will be returned to command mode, and Extended Basic will not redefine the letters. NOTE: You do not have to do this if you use the routines inside a program.

CHARACTER SETS:

The four character set routines allow you to use different character sets other than the standard Extended Basic set.

CALL LINK("SMCAPS")

Loads the standard upper case character set (characters 32-95). Same as CALL CHARSET, but does not reset character colors.

CALL LINK("LGCAPS")

Loads the title screen upper case character set (characters 32-95). The title screen characters are larger than the standard character set.

CALL LINK("STDLOW")

Loads the standard lower case character set (characters 96-126). The lower case letters appear as small capitals. CALL CHARSET does not reset these characters.

CALL LINK("LOW")

Loads a true lower case character set (characters 97-122). Also redefines the zero to be slashed, and makes the capital O round.

SOUNDS:

The three sound routines allow you to generate commonly used sounds with one command.

CALL LINK("BEEP")

Emits the valid response tone. It is the input tone used in Extended Basic.

CALL LINK("HONK")

Emits the bad response tone. It is the error tone used in Extended Basic.

CALL LINK("CHIMES")

Plays a chimes sound. It is the same as the one used in the FAST-TERM program.

CASSETTE CONTROL:

The two cassette control routines allow you to turn the motor on or off on the cassette in port 1. This is done through the small black plug. You can control music or other taped material through your programs.

CALL LINK("CS1ON")

Turns on the motor control for cassette port 1.

CALL LINK("CS1OFF")

Turns off the motor control for cassette port 1.

COLORS:

The four color routines allow you to control the colors of different character sets in new ways.

CALL LINK("COLORS",foreground,background)

Changes the colors of all 15 character sets (0-14) to foreground and background. Foreground and background are numbers from 1 to 16, which represent the Extended Basic color code.

CALL LINK("BURST",cycles)

Rapidly changes the screen color through all possible colors, creating an unusual optical illusion. Upon completion, the screen color will be white. Cycles can be a number from 1 to 30000, with 1 being very short, and 30000 being a very long duration.

CALL LINK("FLASH",foreground,background)

Copies the character definitions for the upper case letters (65-90) on to the lower case letters (97-122). In other words, the lower case letters appear as upper case. Any lower case letters on the screen will flash with foreground and background colors. Flashing will continue without further program control until NORMAL is executed (see below.)

CALL LINK("NORMAL")

Stops the flashing caused by the above routine, resets TI's standard lower case letters, and resets the colors to black on transparent.

VDP ACCESS:

The three VDP routines allow you to directly access VDP (Video Display Processor) RAM memory. For details as to what this memory contains, consult the Editor/Assembler and Explorer manuals.

CALL LINK("PEEKV",address,variable1[,variable2 ... ,variable15])

Allows you to read up to 15 consecutive VDP memory addresses. Address should be a number from 0 to 16383.

CALL LINK("POKEV",address,byte1[,byte2 ... byte15])

Allows you to write to up to 15 consecutive VDP memory addresses. Address should be a number from 0 to 16383, and byte1 through byte15 should be values from 0 to 255.

CALL LINK("VDPREG",register,byte)

Writes to a VDP write only register. Register should be a number from 0 to 7, and byte should be a value from 0 to 255.

SCREEN ACCESS:

The seven screen access routines allow you to control what is on the screen.

CALL LINK("SCROFF")

Disables screen display. All that is visible on the screen is the screen color. This is the same as when a key is not pressed for five minutes. After disabling the screen display, you can build your screen, piece by piece, then enable the screen display (see below) to make it all appear at once.

CALL LINK("SCRON")

Enables the screen display.

CALL LINK("ALL",character)

Fills the entire screen with given character code. It is the same as CALL HCHAR(1,1,character,768) but it executes much faster.

CALL LINK("RDSCR",string-array)

Reads in the contents of the screen into string-array, assigning one line to each element. If you are in text mode, 21 elements must be dimensioned, otherwise, 24 elements must be dimensioned. Each array element will contain one line of characters from the screen. Element zero is not used by this routine.

CALL LINK("WTSCR",string-array)

Writes the contents of the string-array to the screen, with each element of the array producing one line on the screen. All elements must have the same length as the lines on the screen. Element zero is not used by this routine. This routine may be used with the one above to save and restore screens using string arrays as temporary storage locations.

CALL LINK("SAVSCR",filename[,screen-color])

Saves the contents of a screen, including color, definitions, sprites, and motion to the device specified by filename. The Myarc 128K/512K RAM-DISK may be used as a valid filename for rapid screen saving. If screen-color is included, the color of the screen (1-16) will also be saved to the device, otherwise it will not be saved.

CALL LINK("GETSCR",filename)

Restores a screen saved by the above routine. If the screen color was saved, it will be restored, otherwise it will remain unaltered. The Myarc 128K/512K RAM-DISK is a valid filename for rapid screen displaying.

CONTROL ROUTINES:

The seven control routines allow you to perform tasks normally not possible in Extended Basic.

CALL LINK("BYE")

Returns to the master title screen. Functions the same as the command BYE, but this routine may be used in a program. Note: This routine does NOT close files like the BYE command does, and all unsaved data will be lost. This routine will not affect STAR (you do not have to re-load.)

CALL LINK("NEW"):: END

Erases the current program in memory. Functions the same as the command NEW, (without clearing the screen) but this routine may be used in a program. Note: This routine MUST be followed :: END or unpredictable results may occur. This routine will not affect STAR (you do not have to re-load.)

CALL LINK("NOQUIT")

Disables the QUIT key (function-equals). When QUIT is pressed, nothing will happen.

CALL LINK("QUIT")

Enables the QUIT key (function-equals). When QUIT is pressed, the computer will return to the master title screen. Do not confuse this routine with BYE.

CALL LINK("NOMOVE")

Disables sprite motion. When this routine is executed, all sprites will stop. Any sprites created with motion, or any sprites told to move will not do so. Motion may be enabled with MOVE (see below.)

CALL LINK("MOVE")

Enables sprite motion. All sprites will move when directed to. This routine, along with NOMOVE above, may be used to create large moving objects made up of several sprites. First, execute NOMOVE. Place the sprites on the screen, telling them to move as desired. When all sprites are set, execute MOVE. All sprites will begin to move at the same instant, so no gaps will exist between them.

CALL LINK("ERROR",error-code)

Issues an Extended Basic error or warning message designated by error-code. DO NOT use the codes in the Extended Basic manual, but use a code from this list:

2	Numeric Overflow	24	Line Too Long
3	Syntax Error	25	Can't Continue
4	Illegal After Subprogram	26	Command Illegal In Program
5	Unmatched Quotes	27	Only Legal In Program
6	Name Too Long	28	Bad Argument
7	String-Number Mismatch	29	No Program Present
8	Option Base Error	30	Bad Value
9	Improperly Used Name	31	Incorrect Argument List
10	Image Error	32	Input Error
11	Memory Full	33	Data Error
12	Stack Overflow	34	File Error
13	NEXT Without FOR	35	
14	FOR-NEXT Nesting	36	I/O Error (gives meaningless code)
15	Must Be In Subprogram	37	Subprogram Not Found
16	Recursive Subprogram Call	38	
17	Missing SUBEND	39	Protection Violation
18	RETURN Without GOSUB	40	Unrecognized Character
19	String Truncated	41	Warning: Numeric Overflow
20	Bad Subscript	42	Warning: String Truncated
21	Speech String Too Long	43	Warning: No Program Present
22	Line Not Found	44	Warning: Input Error
23	Bad Line Number	45	Warning: I/O Error
		46	Warning: Line Not Found

KEYBOARD DETECTION:

The five keyboard detection routines allow you to check for key presses that would be difficult or impossible to do from Extended Basic.

CALL LINK("LOCK",variable)

Checks to see if the ALPHA LOCK key is down (on). If it is, then variable will equal one. If it is up (off), then variable will equal zero. This is useful for game programs that require joysticks, because the joysticks will not operate properly if the ALPHA LOCK key is down.

CALL LINK("SHIFT",variable)

Checks to see if either of the two SHIFT keys are down. If it is, then variable will equal one. Otherwise, variable will equal zero.

CALL LINK("CTRL",variable)

Checks to see if the CONTROL key (marked "CTRL") is down. If it is, then variable will equal one. Otherwise, variable will equal zero.

CALL LINK("FCTN",variable)

Checks to see if the FUNCTION key (marked "FCTN") is down. If it is, then variable will equal one. Otherwise, variable will equal zero.

CALL LINK("KEY",valid-keys,key-code)

Scans the keyboard waiting for a key press. Valid-keys is a string containing a list of acceptable key presses. If this is a null string (""), then all keys will be allowed. The key code of the key pressed will be returned in the variable key-code.

CHARACTER DEFINITIONS:

The six character definition routines allow you to modify the definitions of characters in unique ways. All of the character definition routines allow you to access characters 30 through 143. Character 30 is the cursor, and character 31 is the edge character. Normally in Extended Basic, it is not possible to modify these characters, but now you can.

CALL LINK("COPY",character1,character2)

Copies the character definition from character1 over to character2 so they appear identical. This would be the same as: CALL CHARPAT(character1,X$):: CALL CHAR(character2,X$) except this routine is faster and saves memory. To change the definition of the cursor (character 30), define an unused character, then COPY the definition to character 30. Character1 and character2 are character codes from 30 to 143.

CALL LINK("MAG",character1,character2)

Magnifies character1 to twice its size, and defines character2 and the next 3 characters as this magnified version. This comes in handy for using magnification factors 3 and 4 with sprites. Character1 is magnified in the same way that sprites are. Character1 is a character code from 30 to 143. Character2 is a character code from 32 to 140, and it must be evenly divisible by four.

CALL LINK("ROTATE",turns,character[,number])

Rotates one or more character definitions, starting at character, and continuing for the specified number. Turns is a number from 1 to 3 which means the following: 1: Character(s) will be rotated 90 degrees clockwise. 2: Character(s) will be rotated 180 degrees. 3: Character(s) will be rotated 270 degrees clockwise (or 90 degrees counter-clockwise, which is the same.) Character is the starting character code, and is a number from 30 to 143. Number is the number of characters to rotate. If number is omitted, it is assumed to be one, and only character is rotated.

CALL LINK("INVERT",character[,number])

Inverts (changes "on" dots to "off", and "off" dots to "on", making a "negative" image) one or more character definitions, starting at character, and continuing for the specified number. Character is the starting character code from 30 to 143. Number is the number of characters to invert. If number is omitted, it is assumed to be one, and only character is inverted.

CALL LINK("FLIP",character[,number])

Flips one or more character definitions upside down (NOTE: This is NOT the same as rotating them 180 degrees, there is a difference!), starting at character, and continuing for the specified number. Character is the starting character code from 30 to 143. Number is the number of characters to flip. If number is omitted, it is assumed to be one, and only character is flipped.

CALL LINK("MIRROR",character[,number])

Creates a sideways mirror image of one or more character definitions (a sideways flip), starting at character, and continuing for the specified number. Character is the starting character code from 30 to 143. Number is the number of characters to mirror. If number is omitted, it is assumed to be one, and only character is mirrored.

DISK ACCESS:

The six disk access routines allow you to access many of the features of the Disk Manager module from the Extended Basic environment, without losing the program in memory. All of these disk routines will work with the Myarc 128K RAM-DISK if it is emulating a disk drive.

CALL LINK("CAT"[,drive])

Catalogs a disk to the screen. Drive is the drive number from 1 to 5. If drive is omitted, it will catalog the most-recently cataloged drive. The disk will be cataloged to the screen in the same format as Disk Manager uses. The listing can be paused by pressing a key, and it can be resumed by pressing a key again. The catalog may be aborted by pressing BREAK (FCTN-4).

CALL LINK("PRO",drive,filename)

Places write protection on filename on the disk in the specified drive. With this "Disk Manager" protection, the file can not be written to, changed, or deleted. Drive is the disk drive number from 1 to 5. Filename is the name of the file to be protected. It may not be longer than 10 letters, and it may not contain any periods or spaces. This protection may be removed with UNPRO (see below.)

CALL LINK("UNPRO",drive,filename)

Removes write protection on filename on the disk in the specified drive. Without this "Disk Manager" protection, the file may be written to, changed, or deleted. Drive and filename are the same as for PRO (see above.)

CALL LINK("RENAME",drive,old-filename,new-filename)

Renames the file called old-filename on the disk in the specified drive as new-filename. Drive is the disk drive number from 1 to 5. Both old-filename and new-filename may be up to 10 letters long, and may not contain any periods or spaces.

CALL LINK("RSEC",drive,sector,string-variable1,string-variable2)

Reads a sector off the disk in the specified drive, and places its contents into the string variables. This routine should be used with extreme caution. Drive is the disk drive number from 1 to 5. Sector is the sector number (sectors are numbered starting with zero.) After execution, String-variable1 will contain the first 128 bytes of the sector, and string-variable2 will contain the last 128 bytes of the sector. For more information on the contents of disk sectors, consult the Advanced Diagnostics manual.

CALL LINK("WSEC",drive,sector,string1,string2)

Writes a sector to the disk in the specified drive. This routine should be used with extreme caution, for ruining one sector can destroy an entire disk! Drive is the disk drive number from 1 to 5. Sector is the sector number to write to (sectors are numbered starting with zero.) String1 contains the first 128 bytes to be placed in the sector, and string2 contains the last 128 bytes. Both string1 and string2 MUST be 128 bytes long.

TEXT MODE:

The four text mode routines allow you to access 40 columns from Extended Basic. In 40 column mode, only lines 1 through 21 may be used, and there are 40 characters per line. Only two colors are allowed in text mode: text color and screen color. When defining characters for text mode, they are only 6X8 instead of 8X8, so the right-most two dots of each dot-row are ignored. Sprites are not available in text mode. The routines SCRON, SCROFF, SAVSCR, GETSCR, RDSCR, and WTSCR will work correctly in text mode. These routines should only be used within a program for proper operation. If your program stops while in text mode (because of an error, break, etc.), you MUST execute CALL LINK("GRAPH") IMMEDIATELY, or the text mode routines may not function as expected.

CALL LINK("TEXT")

Enters text mode (40 columns). The screen is cleared, and the screen colors are set to white on dark blue. If you were already in text mode, the screen is cleared, but your screen colors are retained. Line 22 is always left blank in text mode. Lines 23 and 24 contain valuable Extended Basic system information. This was retained so your program may stop without crashing the computer. You MUST ALWAYS return to graphics (32 column) mode when you are done using text mode.

CALL LINK("GRAPH")

Returns to graphics mode. The screen is cleared, and its color is set to cyan. If you are already in graphics mode, nothing happens. When you return from text mode, all character set colors will be unaltered, but all sprites will be destroyed.

CALL LINK("PRINT",row,column,string)

Displays string on the screen, starting at row and column in text mode. This routine will do nothing in graphics mode. If the string goes off the edge of the screen, it will wrap-around to the next line. If it goes off the bottom of the screen, the screen will scroll up one line. Row is the starting row number from 1 to 21. Column is the starting column number from 1 to 40. String is the string to be printed.

CALL LINK("SCREEN",text-color,screen-color)

Changes the text color and screen color in text mode. This routine will do nothing in graphics mode. Both text-color and screen-color are color codes from 1 to 16.

STRING HANDLING:

The two string handling routines change strings in convenient ways that are cumbersome and slow in Extended Basic.

CALL LINK("CAPS",string-variable)

Changes all the lower case letters in string-variable to upper case letters. All the other characters in the string-variable remain unaffected.

CALL LINK("REVRSE",string-variable)

Reverses the order of the characters in string-variable. For example, if string-variable contained "ABCDE", after execution, it would contain "EDCBA".

These are the 53 routines contained in the STAR package. I hope these descriptions enable you to use them to their fullest potential. If you have any questions concerning the operations of these routines, feel free to write me at the address on the title screen. Be sure to include a self-addressed stamped envelope for a reply. I am sorry to say that I cannot reply to letters which do not contain a self-addressed stamped envelope.

ERROR MESSAGES:

The 53 STAR routines issue 4 error messages, their descriptions follow:

SUBPROGRAM NOT FOUND

This error will be issued if you misspelled the name of the routine (all routine names are in capital letters.) Also, this error will be issued if the package was not loaded correctly (see the beginning of this documentation.)

FILE ERROR

This error is issued by routines which access devices. It is impossible to get Extended Basic to issue an I/O error with the proper error code, so a file error is issued instead. The first digit of the I/O error code does not matter, since no files were opened in the first place, but the second digit may be obtained by executing the following IMMEDIATELY after the error:

```
CALL PEEK(-24576,X):: PRINT X
```

The second digit of the I/O error code will then be printed (for a description of error codes for the disk routines, consult the disk controller manual.) The RSEC and WSEC routines return a zero for the error code, but the error code should be a six, because a sector cannot be read only when the disk is not initialized.

BAD VALUE

The routines issue this error when you supply a value that is beyond the acceptable limits. This error is also given when the string length for WSEC is not 128.

STRING TRUNCATED

This error is issued when a supplied string is beyond the maximum acceptable length. Note that this is an error, not a warning.

For other errors, consult the Extended Basic manual, for they were not issued by STAR. You may issue any Extended Basic error or warning by use of the ERROR routine. You can use an ON ERROR statement to handle these error conditions. Note: CALL ERR will return the Extended Basic error code for an error, not the code listed for the ERROR routine.

MODIFING THE LOAD PROGRAM:

The files LOAD and LOAD1 should not be altered in any way. Doing so will cause improper loading of the STAR package. Line 1 of the file LOAD2 may be altered to suit your needs. Altering line 0 will cause improper loading of STAR. Line 1 currently reads:

```
1 CALL LINK("NEW"):: END
```

As you know, this will end the program and erase it from memory. You can change this line to do anything you want, such as run another program that needs the STAR routines. You may add lines to the end of this program, but do NOT modify line 0 or resequence the program, or STAR will not load correctly.

IMPORTANT: --------- Programs that use the STAR routines will not run properly unless the STAR package is loaded and in memory.

SOURCE CODE:

The original source code, which is fully documented line-by-line and is 74 pages long, is available from COM-LINK Enterprises. It is of the utmost value to beginners and experts alike, because it provides many programming tips, short-cuts, and methods of interfacing to Extended Basic programs. The source code is NOT available as Fairware, and may only be obtained by purchase.

The source code costs $20 if you send a blank disk, mailer, and postage, or $25 if you do not. Make sure you include a note indicating that you are purchasing the STAR source code, and be sure to include your name and address.

IMPORTANT: --------- The fee for the source code is in ADDITION to the Fairware donation of $15. If you have not paid for STAR, your request will be denied.

MAKE ALL CHECKS PAYABLE TO MICHAEL RICCIO. Send your payment to the address on the title screen.

SAMPLE PROGRAMS:

Several sample programs have been provided on the disk to illustrate the use of the STAR routines.

HELP

Prints out these instructions. Routines used: None.

MULTI

Demonstrates multi-color mode. Routines used: LGCAPS, LOW, SMCAPS, STDLOW, VDPREG, and ALL.

SCREENS

Displays 8 screens, waiting for a key press before going on to the next one. The screens are from the files SCREEN1 through SCREEN8. Routines used: KEY and GETSCR.

SIDEWAYS

Prints items sideways on the screen. Routines used: LOW, CHIMES, ROTATE, KEY, and COPY.

SPEECH

A comical demonstration of the computer with speech. Routines used: GETSCR, KEY, and COLORS.

SPRITES

Demonstrates the use of MOVE and NOMOVE. Routines used: MOVE and NOMOVE.

VDPUTIL2

A modified version of VDP UTILITY. To use it, MERGE it in with a TI Basic program. The program will now run correctly in Extended Basic, because character sets 15 and 16 will be available.

I hope you enjoy STAR, and I would like to see many Extended Basic programs make use of the routines. I hope you find them helpful. If I get a good response from this release, I might write another package with many more useful routines. Remember, feel free to write to me! Enjoy it!

UPDATES FROM VERSION 1.0:

This release of STAR has several improvements over the first release. They are listed here so that you can note these changes where appropriate.

CAT

The CAT routine will now work with the Myarc 128K/512K RAM-DISK. Also, the messages are in upper and lower case.

REVRSE:

There was a bug in Version 1.0 that caused a lock-up if a null string was passed to this routine. This has now been corrected.

LOW:

The definition of "w" was slightly incorrect. This has been corrected.

Source Code:

All changes for Version 1.1 are clearly marked in the source code by "* V1.1 *" after the comment. This allows you to better understand the development of STAR and the debugging of the routines.

If you notice any bugs in this release, please write to me and let me know. I will try to correct them as soon as possible. If you have already paid for a copy of Version 1.0, you do not need to pay for this update. Thank you for your continued support.

Try calling our bulletin board, COM-LINK BBS at (215) 289-4948, 24 hours, 300 baud, for the latest information on COM-LINK Enterprises. Our board features both amazing speed and the ultimate flexibility. It has 8 message bases and XMODEM file transfers. Any person who has made a Fairware donation for STAR will receive instant validation. This is the fastest way to talk directly to me. I hope to be talking to you soon!

Michael Riccio
COM-LINK Enterprises