# T80XB

"Altair"

Eighty Column Utilities for TI Extended BASIC

by Harry Wilhelm



05/22/18

# INTRODUCTION

T80XB is a collection of assembly language subroutines that give the Extended BASIC programmer easy access to the 80 column screen mode offered by the F18A video upgrade and other 80 column upgrades. No knowledge of assembly language is required to use T80XB. Programs are written completely in Extended BASIC, so they are both easy to write and easy to understand.

T80XB lets you select from two independent screens. G32 is the default screen when a program starts running.. This is the 32 column graphics mode screen normally used by Extended BASIC. It is accessed using the usual XB graphics statements. T80 is the 80 column text screen which offers 24 rows of 80 columns.. You can toggle between the two screens as desired, preserving the graphics on each screen. When using the T80 screen there are assembly equivalents that replace PRINT, CLEAR, COLOR, INPUT, CHAR, HCHAR, VCHAR plus routines that will scroll the screen and invert text on the screen.

# EQUIPMENT REQUIRED

T80XB has only been tested with the Classic99 emulator. (In Classic99, you should “enable 80 column hack” under Video) If you use vintage hardware, it should work using the TI-99/4A console, Extended BASIC cartridge, 32K memory expansion, a disk drive system and an 80 column upgrade such as the F18A.. It is compatible with the CF7 expansion. I have not yet received confirmation that it is compatible with the 9938 video chip

# DIFFERENCES FROM EXTENDED BASIC

The T80 text mode has certain limitations compared to the normal G32 graphics mode. Sprites cannot be displayed. You can only use two colors. As implemented in T80XB only the 96 characters from 32 to 127 can be defined, and 127 is reserved for the cursor. The 96 characters from 127 to 223 are inverse video copies of the characters from 32 to 127.

When a program starts running in T80XB it defaults to the G32 mode. The program can then select the T80 mode when desired. The G32 screen is unmodified but hidden while the T80 screen is displayed. When you break a program by pressing <Fctn 4> the display will automatically revert to the G32 screen. After breaking a program, you can type CON to continue. If you were using the T80 screen, T80XB will display that screen just as it was when the program was interrupted.

The memory available for programming is 24488 bytes which is the normal size for XB. The stack space is reduced from 11840 bytes to 8138 bytes (with the default 3 disk files). The stack is primarily used to contain string data and subprogram names and there should be enough room for most programming needs.

Disk file space is allocated in the usual manner using CALL FILES(n)

## LOADING T80XB

T80XB is an XB extension that resides in low memory, separate from where XB programs are stored. Once loaded, you can load, write or save XB programs with no effect on T80XB. To load it, select Extended BASIC from the TI master title screen. Place the program disk into a disk drive (drive number "n"), type RUN "DSKn.T80XB" and press <Enter>. T80XB loads into low memory, starts up, and then the loader erases itself so you can start programming. In immediate mode, the colors are changed to black on gray to show that T80XB is loaded and active. T80XB can be renamed LOAD for autoloading from drive 1.

## USING T80XB

T80XB contains 17 assembly language subroutines.

Except for the commands T80ON and T80OFF, all these subroutines should be called from within a running Extended BASIC program. No error message results when the subroutines are called from the immediate mode, but nothing useful will result. They only have an effect on the T80 screen. You can use the T80 routines in the G32 mode or XB graphics instructions in the T80 mode but the results will only appear when you change the screen mode.

## COMMANDS

**CALL LINK("T80ON")**
Turns the T80XB interrupt routine on This command is contained in line 10 of the T80XB loader. This can be used in the immediate mode to reactivate T80XB if you turned it off with CALL LINK("T80OFF") as described below. (Immediate mode)

**CALL LINK("T80OFF")**
Turns off the interrupt routine for T80XB. This turns off the interrupt routine without having to return to the master title screen. T80XB is still loaded can can be reactivated with CALL LINK("T80XB") (Immediate mode)

## THE SUBROUTINES

The subroutines are described below. The first line of each description shows the correct syntax to use when calling the subroutine. Most of the subroutines require that additional information be included after the name of the subroutine. This information is supplied in the form of a parameter list. Be careful to include these parameters in the order described, and not to mix strings and numbers. Sometimes there are optional parameters. These optional parameters are shown enclosed in brackets. The purpose of each of the parameters in the list is fully described. Numbers and strings can be constants, variables, or elements of an array.

**CALL LINK(“G32”)**
Selects G32, which is the normal XB screen. This has no effect if you are already in the G32 mode..

**CALL LINK(“T80”)**
Selects the T80 mode which is the 80 column text mode.. This has no effect if you are already in the T80 mode. When a program first calls this the T80 screen is cleared, the standard character set is loaded and the colors are set to white foreground on a dark blue background.

**CALL LINK("COLOR”,foreground-color,background-color)**

This is used to change the text colors in the T80 mode. The color codes are the same as in Extended BASIC

**CALL LINK("CHAR”,character-code,pattern-identifier[,...])**

This is the T80 equivalent of CALL CHAR in XB. It is used to change the patterns of the characters used in the T80 mode.. *Character-code* specifies the ASCII code of the character you wish to define. It must be a value from 32 to 127. *Pattern-identifier* can be a hexadecimal string of up to 255 bytes. (This is much longer than XB allows.) 16 bytes are used to define each character. If the length of *pattern identifier* is not a multiple of 16, then zeros are added to the final character definition to make it 16 bytes long. CHAR also automatically redefines the inverted characters.

```
CALL LINK(“CHAR”,65,”0123456789ABCDEFF”) defines
character 65 to be “0123456789ABCDEF” and character 66
to be “F000000000000000”)

CALL LINK(“CHAR”,65,”1234”,80,”5678”) defines character
65 to be “1234000000000000” and character 80 to be
“5678000000000000”
```

**CALL LINK("INVERT”,row,column,length[,...])**

INVERT is used to toggle text to inverse video or back to normal. *Row* and *column* determine the first character you want to invert and *length* is the number of characters to invert. *Row* is from 1 to 24. *Column* is from 1 to 80. *Length* is from 1 to 920. INVERT will stop at the lower right of the screen even if the combination of *row, column* and *length* should go off the bottom of the screen. A character in the area being inverted with an ASCII of less than 128 will have 96 will be added. A character in the area being inverted with an ASCII greater than 127 will have 96 subtracted. Because characters 128 to 223 are set to inverse video when T80 is called, the hiliting takes place automatically. If you want to restore text back to normal, do a second CALL LINK(“HILITE”,row,column,length).

**CALL LINK(“SCROLL”[,repeats])**

SCROLL will scroll the screen up one line and fill the bottom line with spaces. Include an optional number from 1 to 24 to scroll that number of times.

**CALL LINK(“SCROLI”[,repeats])**

Identical to SCROLL except the spaces are in inverse video.

**CALL LINK(“CLS”)**

Fills the T80 screen with spaces. (ascii 32)

**CALL LINK(“CLSI)**

Fills the T80 screen with inverted spaces. (ascii 128)

**CALL LINK(“HCHAR”,row,col,character-code[repeats,.....])**

HCHAR is the T80 equivalent of HCHAR in XB. *Row* is from 1 to 24; *col* is from 1 to 80; *character-code* is from 32 to 223. As in XB you can add an optional number to repeat. You can do up to 4 HCHARs per call by adding additional *row,col,character-code,repeats.* If you want to use this feature you must include the optional repeat so that 4 values are passed per HCHAR.

```
CALL LINK(“HCHAR”,2,2,42,10,4,4,65) will display 10
asterisks (ascii 42) starting at row 2, column 2. It
then displays a single “A” (ascii 65) at row 4, column 4
```

**CALL LINK(“VCHAR”,row,col,character-code[repeats,.....])**

Identical to HCHAR above except it displays characters vertically like HCHAR in XB.

**CALL LINK(“INPUT”,row,col,string-or-numeric-variable[,length,prompt-string])**

INPUT is used by your program to input a string or numeric variable. *Row* is a number from 1 to 24; c*ol* is a number from 1 to 80. The line will be blanked to the right until column 80 is reached. Include an optional length of -80 to 80.if you do not want to blank the entire line. A negative length will not erase the line if you want to suggest an input. If you use the optional prompt string you must also use the optional length.

```
CALL LINK(“PRINT”,2,2,”Hello World”)::CALL
LINK(“INPUT”,2,2,A$,-15) will input the string variable
A$ at r2,c2 using “Hello World” as a prompt.

CALL LINK(“INPUT”,4,4,X,20,”3.14159265”) will input the
variable X at r4,c4 using the value of Pi as a prompt.
```

```
CALL LINK("INPUT",10,76,A$,80) will input the string
variable A$ starting at r10,c76. You can only input 5
characters from c76 to c80 because the right hand column
has priority over the specified 80 character length.
```

**CALL LINK("INPUTI",row,col,character-code[repeats,.....])**

Identical to CALL LINK("INPUT") except it inputs text in inverse video.

**CALL LINK("PRINT",row,col,string or number[,length,.....])**

PRINT lets you print a string or number on the T80 screen.. *Row* is a number from 1 to 24; *col* is a number from 1 to 80, optional *length* is a number from 1 to 80. Printing starts at row and col, and goes to the right until it reaches column 80. There is one exception: If it is printing on row 24 and is not done printing the string when it reaches the lower right corner (r24, c80), then it will scroll up one line and continue printing until finished.

If you omit the optional length then it will fill the remainder of the line with spaces. Including the optional length forces PRINT to use just that number of characters. Printing will stop at column 80 even if the optional length would be past that column.

You can do up to 4 prints per call, but you must include the optional length

```
CALL LINK("HCHAR",1,1,42,1920)::CALL
LINK("PRINT",2,2,"Hello",10,4,4,"Hello World",
10,6,1,3.14159265)
fills the screen with asterisks;
then at r2,c2 prints "Hello" followed by 5 spaces;
then at r4,c4 prints "Hello Worl" (the first 10
characters of  "Hello World");
then at r6,c1 prints Pi with the rest of
the line filled with spaces because there was no
optional length.
```

**CALL LINK("PRINTI",row,col,string or number[,length,.....])**

Identical to CALL LINK("PRINT") except it prints in inverse video.

## 16K VDP RAM MEMORY MAP WITH T80XB

| | | |
|---|---|---|
| 0 – 767<br>>0000->02FF | Screen Image Table | VDP address is given by (Row-1)*32+Column-1 |
| 768 – 879<br>>0300->036F | Sprite Attribute Table | Room for 28 sprites – each sprite needs 4 bytes<br>vertical position-1,horizontal position,<br>char #+96,color-1(+128 for early clock) |
| 1008 – 1919<br>>03F0->077F | Pattern Descriptor Table<br>for G32 | 8 bytes per character – char 30 starts at 1008 |
| 1920 – 2031<br>>0780->07EF | Sprite Motion Table | 4 bytes per sprite.<br>Vertical velocity, horizontal velocity; sys use; sys use |
| 2048 – 2079<br>>0800->081F | Color Table for G32 | 1 byte per character set – char set 0 = 2063<br>(foreground-1)*16+background-1 |
| 2432 – 3071<br>>0900 - >0BFF | Pattern descriptor table for characters 32 to 127 (Normal patterns) | |
| 3072 – 4095 | Pattern descriptor table for characters 128 to 223 (Inverse patterns) | |
| 4096 – 6016<br>>1000 - >177F | Screen Image Table for T80 mode | 1920 bytes long |
| 6016 - 14295<br>>1780 - >37D7 | Value stack | Used by XB for strings, etc. CALL LINK(T80XB,n)<br>will reserve N bytes starting at 6176 |
| 14296 – 16383<br>>37D8 - >3FFF | Disk Buffering Area for CALL FILES(3) | |

## FILES INCLUDED

| | |
|---|---|
| README | Start here. |
| T80XBdocs.pdf | Documentation for the T80XB package |
| T80XB | XB loader program for T80XB |
| T80XBHM | High memory version of T80XB |
| T80DEMO | Demo program |
| T80DEMOHM | Demo program packaged with T80XBHM in one file |
| MYWATCH | Demo with Mark Twain short story. |

## PACKAGING T80XB WITH AN XB PROGRAM

Usually you would want to load T80XB into low memory as described above. Having the XB program and T80XB in distinct segments is the easiest way to develop programs using T80XB. But after a program is developed you may find it useful to combine T80XB with the XB program, making a nice, neat one program package. T80XBHM is a high memory version of T80XB that makes it easy to do this.

Using T80XB, develop your 80 column XB program as normal. (But do not use lines 1-3). Once you are happy with it, follow these steps, which assume the T80 disk is in drive 1:
1 – SAVE DSK1.PROGRAM
2 – SAVE DSK1.PROGRAM-M,MERGE (Or you can LIST "CLIP" with Classic99)
3 – CALL LINK("T80OFF") (Or you can quit and restart XB)
4 – OLD DSK1.T80XBHM
5 – MERGE DSK1.PROGRAM-M (Or you can PASTE XB with Classic99)
6 – SAVE DSK1.PROGRAMHM
That's all there is to it. When it starts, the program turns on T80XB, then runs the XB code.

---

Now for some details:

Because T80XB is combined with the XB program, the available program space is slightly reduced. There are 21502 bytes available instead of the usual 24488 bytes.

When T80XBHM is active there is an interrupt routine running in high memory. If you load another XB program when this interrupt is active the computer will crash. To load another program, first turn off T80XB with CALL LINK("T80OFF") or else quit and restart.

The three lines of T80XBHM XB code are:

```
1 !T80XB high memory version   by Harry Wilhelm
2 CALL INIT :: CALL LOAD(8192,255,178):: CALL LINK("T80ON"):: RUN 3
3 !@P-              (this has to be the third line!)
```

Line 3 turns off prescan which lets the program start up quickly without a time consuming prescan of the XB program. When CALL LINK("T80ON") is performed, it turns on the T80 interrupt routine and changes the third line of the program to enable prescan: 3 !@P+
Then RUN 3 starts the program again which initializes the stack and variables so that T80XB can work.

When T80XB is activated, you can safely make changes to the XB program. But do not resequence it, as that can modify the embedded T80XB code. Be sure to change line 3 back to !@P- before saving. If you are making major changes it is best to return to the standard T80XB and make them there.