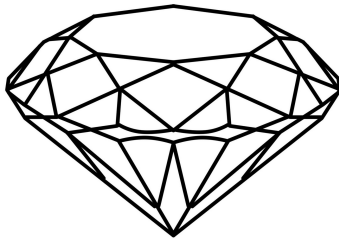# Texas Instruments
Extended BASIC 2.8
G.E.M.

by Harry Wilhelm
Extended Basic v2.7 by:Tony Knerr

09/01/20

Extended BASIC 2.8 Graphics Enhancement Module.

The original XB G.E.M.module was developed to see how much power could be packed into TI Extended BASIC using just the 5 XB groms and the standard XB roms. Three graphics utilities totaling about 18K of code were compressed and squeezed into about 12K of free space in the XB groms. There is virtually no room left in the standard size roms and groms, and further development within these limits is not feasible.

These limitations go away when you bank switch additional ROM pages in the cartridge. This made it possible to add many significant enhancements. Starting with Tony Knerr's XB 2.7, I added 15 – 8K pages of ROM memory and a number of new CALLs and other refinements to the XB 2.7 groms. The result is Extended BASIC v2.8 G.E.M.

New features in XB 2.8

Some of the existing CALLs have been modified.

CALL INIT - in addition to loading the assembly support routines it also clears the interrupt hook at >83C4. Unlike XB, CALL INIT is performed at XB startup.

CALL LOAD(filename) now uses the fast assembly language loader from the minimemory cartridge. This is about 20x faster than the normal loader used by other Extended BASICs. You can now load compressed code, use REFs and most of the other features missing from the normal XB loader. ***Do not AORG code to memory from >2006 to >2118.*** This is temporarily used by the loader. When the loader is finished these addresses are then reset to the normal XB values provided by CALL INIT.

CALL HELP now displays a detailed help menu with brief descriptions of all the new calls in XB2.7 and XB2.8. Use the space bar to advance to the next page of the help menu. The third screen has tables showing color codes and character sets. Enter a font number between 1 and 60 to display that font. Press space bar and you can press any key to see the ASCII code for that key. Press space bar and you get a number converter. Enter a decimal, hexadecimal or binary number. Space bar to exit.

CALL VERSION(X) returns a floating point number. The most recent version is 2.820200903 (2.8+year+month+date)

CALL CHAR can now define any of the characters from 30 to 159. A string up to 255 bytes long can be used as the pattern identifier. This lets you define up to 16 characters with a single CALL CHAR.

CALL CHARPAT can now return a character pattern for characters from 30 to 159.

CALL COLOR can now define the color of all the character sets from 0 to 31. Although you cannot change the pattern of the characters below 30 and above 159, by setting the foreground and background to the same color you can have an 8x8 block of one solid color. There are 15 unused character sets, so you can have blocks of all 15 colors without sacrificing *any* of the characters from 30 to 159. These can be used for borders and other graphics.

CALL COLOR(ALL,foreground,background) will set all the characters to the selected colors.

With these changes to the limit checks, XB 2.7 G.E.M. can run TI BASIC programs that use character

sets 15 and 16, and usually at a considerable increase in speed.

Added features to provide a more satisfactory programming experience.

FONTS – 60 fonts are contained in the cartridge roms. A font can be loaded to the normal XB location with CALL FONT(number). The graphics packages have new CALL LINKs to load a font to screen 2 in XB256, to the bit mapped screen of The Missing Link, or to the text mode screens in T40XB or T80XB. You can customize the look of your computer by changing the screen colors and font used in the editing mode.

RND now uses the same code as the TI BASIC random number generator. This is about 5x faster than the XB random number generator. When you use a random number seed (i.e. RANDOMIZE 99), the random numbers generated are the same as in TI BASIC. There is another very useful modification: normally a LOAD program will always produce the same random numbers when it autoruns at XB startup. This has been changed so that now a LOAD program will actually produce random numbers.

There are changes to the line editor to make programming easier. You can enter long XB lines with up to 12 rows of code instead of the usual 5. The cursor repeat speed is faster. When editing a program line the cursor appears in the usual place, but you can go left and change the line number. The left/right/up/down keys function normally. Or you can hold down Ctrl and use the up/down/right/left keys to easily navigate a lengthy program line. Right/left work normally and up/down will go up or down one row in the line being edited. If you are on the last row, Ctrl+down goes to the end of the line. If you are on the first row, Ctrl+up goes to the beginning of the line. Fctn 3 (clear) erases the line from the cursor to the end of the line.

Instead of the * READY * message, XB 2.8 GEM gives the following messages at startup and when a program ends: * Extended Basic v2.8 G.E.M. if you have selected any XB option on the menu.

At startup or when a program breaks or ends one of the following messages will also be displayed:
* XXB * if you selected XB 2.8 + XXB
* XB256 * when XB256 is loaded and active
* T40XB * when T40XB is loaded and active
* T80XB * when T80XB is loaded and active
* TMLGA * when TML Graphics Adventure is loaded and active

You would normally use CALL CAT(devicename) to get a disk catalog, but there is also an assembly language disk catalog program in XB256, T40XB, and T80XB. CALL LINK("CAT") will start the program. Press any key to pause or resume the listing.

CALL VPEEK, VPOKE and GPEEK have been renamed PEEKV, POKEV and PEEKG. This is consistent with the convention used in Editor/Assembler BASIC, MiniMemory BASIC and RXB.

NEW CALLS IN XB 2.8

CALL INITG
INITG does the CALL INIT routine, loads two additional support routines for GPLLNK and
DSRLNK, then clears the interrupt hook at >83C4.
GPLLNK is at >24F4 and DSRLNK is at >253A.

CALL FONT(Number)
FONT is used to load one of the 60 fonts contained in the XB 2.8 G.E.M. roms. Patterns are loaded for
ASCII 30 to 127. The number must be from 1 to 60 and can be a constant or a variable.

CALL LFONT(Filename)
LFONT is used to load a font from disk. Patterns are loaded for ASCII 30 to 127.

CALL SFONT(Filename)
SFONT is used to save a font to disk. The cursor pattern is moved to ASCII 127 and then the characters
from 32 to 127 are saved in memory image (binary) format.

CALL SAVEIV("DSKn.PROGNAME").
Used to save a program to disk in IV254 format. This is the format used by XB when saving long
programs to disk. CALL SAVEIV("DSK2.TEST3") will save the currently loaded XB program as
TEST3 onto DSK2. If the program is very small, a line 1 is created to pad the program with a DATA
statement, which lets you save even the shortest XB program as IV254. (Immediate mode)
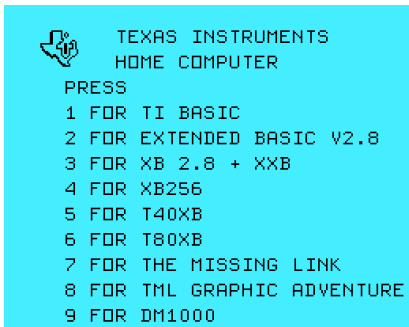
CALL RUN(Filename)
RUN is used to run an XB256 program from a running program. It does the same thing as RUN
"DSK1.PROGRAM" but you can use a string variable which is not permitted in XB. This could be
useful in a menu program or when chaining programs together.

CALL RUNL1(Filename)
RUNL1 is used to run an XB256 program from a running program. It is similar to CALL RUN but there
is a major difference. After it loads the program it runs the first line in the program, *but without doing a
prescan or resetting any of the variables of the calling program.* By chaining programs together with
RUNL1 the 24K limit disappears. You can divide a very long XB program into shorter sections and load
the segments of program as needed, with instant start up of the chained programs.

More information on using RUN, RUNL1 and SAVEIV can be found in the section starting on page 10
entitled AUTOLOADING AND CHAINING PROGRAMS.

The Graphics Packages

```
     TEXAS INSTRUMENTS
      HOME COMPUTER
 PRESS
 1 FOR TI BASIC
 2 FOR EXTENDED BASIC V2.8
 3 FOR XB 2.8 + XXB
 4 FOR XB256
 5 FOR T40XB
 6 FOR T80XB
 7 FOR THE MISSING LINK
 8 FOR TML GRAPHIC ADVENTURE
 9 FOR DM1000
```

The main purpose of XB 2.8 G.E.M. is to provide graphics packages that unshackle the crippling limits on the TMS9918A video display processor imposed by TI XB.

XXB and the five graphics packages all require the 32k memory expansion to run. If it is not detected, the message "32K NOT DETECTED" appears. Press any key to go to Extended BASIC 2.8.

The brief descriptions below describe any differences from the disk based versions. Refer to their respective manuals in the documents folder for detailed information on how to use them.

XXB 1.5 by Barry Traver. This was part of XB 2.7. XXB is one of the more complete XB extension packages. It provides many additions to XB.

XB256 – this is part of the XB game developers package. You can select from two independent screens. Screen1 is the screen normally used by Extended BASIC. It is accessed with the usual XB graphics statements. Screen2 lets you independently define 256 characters, more than twice the 112 normally available in XB. Along with these graphics you can have up to 28 double sized sprites using the patterns available in Screen1. You can toggle between the two screens as desired, preserving the graphics in each screen. Screen2 has assembly equivalents that replace CHAR, CHARPAT, COLOR, and CHARSET. Except for these subroutines, all screen access in Screen2 is by the usual Extended BASIC statements such as PRINT, SPRITE, ACCEPT, etc. You can scroll screen characters left, right, up, or down. By specifing a window area you can scroll just the characters in the window. Other routines let you scroll smoothly one pixel at a time to the left, right, up or down, or do a "text crawl" like the title scene of "Star Wars. You can hilight text, play sound lists, print text using 32 columns and much, much more.

The autoprompt feature is off by default. This is only useful to those who use the entire game developers package (XB256, compiler, loader, etc.) If desired it can be activated with CALL LINK("XB256A"). The font used in screen 1 is whatever font used by the XB editor. Screen 2 uses font 1. The option to modify grom at >6000 is no longer needed and is disabled. The sound list compilers can be found in the disk based version of XB256.

T40XB – T40XB 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. T40 is the 40 column text screen which displays 24 rows of 40 columns.. You can toggle between the two screens as desired, and the graphics on each screen are preserved. When using the T40 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.

The font used in graphics mode (G32) is the font used by the XB editor. Text mode (T40) uses font 1. . The option to modify grom at >6000 is no longer needed and is disabled.

T80XB – This is virtually identical to T40XB except it uses the 80 column text mode provided by some display devices or Classic99

The Missing Link – allows easy and versatile access to bit mapped graphics. You can plot points, lines, circles and boxes. Turtle graphics can be used with none of the ink and color restrictions found in LOGO. Up to 32 moving sprites can be placed on the screen. Text can be input from or displayed to the screen. Word wrap is used when displaying text. The character size can be changed, permitting up to 32 rows by 60 columns on the screen. Different sized text can be displayed simultaneously on the same screen.

When you select The Missing Link from the main menu, a submenu lets you configure TML. You can choose 16 or 2 colors and 0-3 disk files open. The "Texas" cursor has been replaced with the standard rectangular cursor. The default font in bit-mapped mode is font 1. CALL LINK("FONTA",N) will load a font from the cartridge. CALL LINK("LFONTA",filename) or ("SFONTA",filename) will load or save a font used by the bit mapped mode.

Since TML is loaded from the cartridge, it is possible to save programs to cassette. With 16 colors you can save a casette program up to 2048 bytes long. With 2 colors the program can be up to 5724 bytes long. You can trick XB into saving or loading even longer cassette programs. (up to 10176 bytes in 16 color or up to 13928 bytes in 2 color.) First type:
CALL LOAD(-31888,63)
Save or load the cassette program. Then type:
CALL LOAD(-31888,31) when ready to run the program.

The Missing Link Graphic Adventure – This is version of TML was modified so that bit mapped graphics pictures can be added to text adventures. It can load pictures to the top third of the screen while having text on the lower third. The default font uses 5x7 characters for a 48 column screen. The lower third of the screen will scroll automatically as you input commands to the computer. The turtle graphics have been eliminated to make room for the new routines.

Disk Manager 1000 – Press 9 to run DM1000 v7.0 This is not useful for those who use emulators, but is very handy if you are running a real TI-99/4a.

CUSTOMIZING XB 2.8 G.E.M.

The bytes at cpu memory -2 let you customize the colors and font used by XB 2.8 G.E.M. when in the edit mode. If they are both 0 then XB 2.8 G.E.M. defaults to black on cyan and font 1. You can change this with CALL LOAD(-2,colors,font) The color should be foreground*16+background-17. The font should be a number from 1-60.

Normally XB tries to RUN "DSK1.LOAD" at startup. But with XB 2.8 G.E.M., when selecting an option from the main menu, you can hold down any key for about 2 seconds to defeat autoloading. (Easiest is to keep pressing the key used to make the selection.) By adding 128 to the font number you can invert that. If your finger is *off* a key XB *will not* autoload. If you hold a key down for about 2 seconds then XB *will* try to autoload.

TIPI users can add 64 to the font number. Then when you enter BYE, XB will try to RUN "TIPI.FC.FCMDXB" which should bypass the title screen and go directly to Force Command.

Example: You want dark blue characters on a gray background, font 17 and autoload defeated. Color is 5*16+15-17 which is 78. Font = 17+128 which is 145.
From the immediate mode you can CALL LOAD(-2,78,145). Or you can make this a 1 line program and SAVE DSK1.LOAD. Now when XB starts it will autoload your preferences for colors and font and defeat autoload. (unless you are holding down a key.)

EXTENDED BASIC VERSION 2.7

Documentation by Tony Knerr with additions for XB 2.8 by Harry Wilhelm

There are several new commands in Extended Basic v2.7 which support floppy disk formatting from the command line of Extended Basic. It is only a limited implementation, as I have very few bytes of program space left in the gram banks. The new commands are: CALL SSSD, CALL DSSD, CALL DSDD, and CALL DSQD.

They have been tested with the TI, Corcomp, and Myarc Floppy disk controllers and the Myarc Hard and floppy disk controller. The disk is formatted as it would be using Disk Manager 1000 or Disk Utilities and choosing N(o) when prompted to verify. The only sectors checked are 0 and 1 as these sectors are written to at the end of the formatting process.

Extended Basic v2.7 is compatible with TI Extended Basic with the following modifications:
The default colors in edit mode are the standard black on cyan, although this can be customized. Regardless of the colors in edit mode, when a program runs the colors revert to the standard black on cyan.

The default character set is one with true lower case. This can be customized
The command CALL CHARSET has been modified to restore both upper and lower case characters to the original TI character set and also resets screen, foreground, and background colors to Black on Cyan.
The CALL INIT command executes a little faster and clears the interrupt hook at >83C4.
By holding down any key after selection of XB v2.7, the user can bypass the auto-search for DSK1.LOAD and go directly to edit mode. This can be inverted as described on page ??.

The following CALLs are resident in version 2.7:

(edit) CALL ALLSET is replaced by CALL FONT(7). Used to reset the characters 32 through 126 to their original TI definitions. Works like CALL CHARSET but does not affect screen, foreground, or background colors.

CALL BYE will reset to the title screen from a running program.

CALL BEEP will sound the familiar accept tone.

CALL CAT("pathname.") will catalog a device while in command mode. Has been tested with TI, Corcomp, and Myarc floppy controllers as well as the Myarc HFDC and Horizon Ramdisk. The pathname must end in a period. Directories on the HFDC of up to a total length of 11 characters are acceptable. Example – CALL CAT("WDS1.ABCDE.") is the longest pathname that will work.

CALL CHARA1 is replaced by CALL FONT(2). Restores the resident XB v2.7 character set if it has been re-defined.

CALL CHIME will generate a chime sound.

CALL CLSALL will close all open files.

CALL COLR13, COLR15, COLR17, COLR18, COLR19, COLR1A, COLR1B, COLR1F, COLRF1, COLRF2, COLRF4, COLRF6, COLRFC are replaced by.CALL COLOR(ALL,foreground,background).

CALL DSDD will format a 1280 sector disk in drive 1 if you are using a Myarc floppy controller with a 40 track Eprom. A 1440 sector disk will be formatted if using a Corcomp floppy controller, a Myarc floppy controller with an 80 track Eprom, or a Myarc hard and floppy disk controller. You will be prompted to insert a disk in drive 1 and press enter to continue. Pressing any other key aborts the formatting process. Using this command with a TI controller will only be a waste of your time as the floppy controller will go through the whole formatting process only to give you an error at the end. Don't say I didn't warn you!

CALL DSQD will format a 2880 sector disk in drive 2 if you are using a Myarc floppy controller with an 80 track Eprom or a Myarc HFDC properly configured for 80 tracks. You will be prompted to insert a disk in drive 2 and press enter to continue. Pressing any other key aborts the formatting process. Using this command with a TI or /Corcomp controller will only be a waste of your time as the floppy controller will go through the whole formatting process only to give you an error at the end, and may damage your disk and/or drive. Don't say I didn't warn you!

CALL DSSD will format a 720 sector disk in drive 1. You will be prompted to insert a disk in drive 1 and press enter to continue. Pressing any other key aborts the formatting process.

CALL CRASH will generate a crash sound.

CALL GPEEK is now CALL PEEKG(address, numeric variable list) – this will read the contents of addresses in GROMS. The function is similar to CALL PEEK.

CALL HELP has been modified to display help screens showing the commands in XB 2.8. You can view a table of color codes, character sets, display any font, find ascii codes and convert numbers.

CALL HONK will generate the familiar error tone.

CALL LRGCPS is replaced by CALL FONT(13). Loads the large capitals character set from the TI title screen.

CALL MLOAD("filename", mode) will load a program image file (E/A option 5) into memory and run it if MODE=1. Example - CALL MLOAD("DSK1.UTIL1",1) will load and run an E/A option 5 file named UTIL1 from floppy drive #1.

CALL MOVE(mode, start address, target address, # of bytes) will move a block of memory. The four modes available are: 1 - from VDP RAM to VDP RAM, 2 - from VDP RAM to CPU RAM, 3 - from CPU RAM to VDP RAM, 4 - from CPU RAM to CPU RAM

CALL MSAVE("filename", start address, # of bytes) will save a portion of memory in program image format.

CALL NEW is the same as the NEW command except that it can be used within a running program.

CALL NYANYA generates the NYANYA sound our children love to do so often.

CALL QUIT is the same as CALL BYE, that is, it will return to the title screen from a running program.

CALL QUITOF disables the function of the QUIT key (FCTN =).

CALL QUITON restores the function of the QUIT key (FCTN =) after a CALL QUITOF.

CALL RESTORE(numeric variable) is the same as the RESTORE statment except that a variable may be used. Example - 100 A=500 :: CALL RESTORE(A)

CALL SCREENOF turns of the screen.

CALL SCREENON turns on the screen again after a CALL SCREENOF.

CALL SPROF stops the motion of all sprites.

CALL SPRON restarts the motion of all sprites after a CALL SPROF.

Call SSSD will format a 360 sector disk in drive 1. You will be prompted to insert a disk in drive 1 and press enter to continue. Pressing any other key aborts the formatting process.

CALL VPEEK is now CALL PEEKV(address, numeric variable list) – this will read the contents of addresses in VDP RAM. The function is similar to CALL PEEK.

CALL VPOKE is now CALL POKEV(address, numeric variable list) – this will write values to VDP RAM. The function is similar to CALL LOAD.

CALL WAIT(duration) causes a delay. DURATION must be a value from 0 to 16382. One second of delay equals a duration value of 50.

CALL XB will re-start XB v2.7 and autoload DSK1.LOAD if no key is held.

CALL XXB will load Barry Traver's XXB version 1.5 into lower memory. It is recommended that CALL INIT be executed immediately prior to CALL XXB to insure all memory is configured properly. All XB v2.7 and XXB v1.5 commands will then be available to the user, making for a very powerful Extended Basic environment. Docs for XXB v1.5 are included in the DOCS folder.

Using REF in an assembly language program

You no longer have to use EQU when writing an assembly program. The new loader in XB 2.8 G.E.M. can process REF correctly. Below is a table showing the that will be recognized by REF.

```
UTLTAB   EQU >7020      VMBW     EQU >2024      SMUL     EQU >0E8C
PAD      EQU >8300      VSBR     EQU >2028      SDIV     EQU >0FF8
GPLWS    EQU >83E0      VMBR     EQU >202C      CSN      EQU >11AE
SOUND    EQU >8400      VWTR     EQU >2030      CFI      EQU >12B8
VDPRD    EQU >8800      DSRLNK   EQU >253A      FCOMP    EQU >0D3A
VDPSTA   EQU >8802      LOADER   EQU >25AE      NEXT     EQU >0070
VDPWD    EQU >8C00      GPLLNK   EQU >24F4      COMPCT   EQU >0000
VDPWA    EQU >8C02      NUMASG   EQU >2008      GETSTR   EQU >0002
SPCHRD   EQU >9000      NUMREF   EQU >200C      MEMCHK   EQU >0004
SPCHWT   EQU >9400      STRASG   EQU >2010      CNS      EQU >0006
GRMRD    EQU >9800      STRREF   EQU >2014      VPUSH    EQU >000E
GRMRA    EQU >9802      ERR      EQU >2034      VPOP     EQU >0010
GRMWD    EQU >9C00      FADD     EQU >0D80      ASSGNV   EQU >0018
GRMWA    EQU >9C02      FSUB     EQU >0D7C      CIF      EQU >0020
SCAN     EQU >000E      FMUL     EQU >0E88      SCROLL   EQU >0026
XMLLNK   EQU >2018      FDIV     EQU >0FF4      VGWITE   EQU >0034
KSCAN    EQU >201C      SADD     EQU >0D84      GVWITE   EQU >0036
VSBW     EQU >2020      SSUB     EQU >0D74
```

GROM space available in XB 2.8 G.E.M.

Much code and data was moved to the cartridge ROMs and this freed up a considerable amount of memory in the five XB 2.8 groms. This could be used for future modifications and improvements to the GPL code. 10520 bytes of free grom are available in the following locations:

```
>B6F0->B7F0      >100 bytes
>B9E8->BFFF      >618 bytes
>D588->D617      >90 bytes
>D7E0->DBFF      >420 bytes
>DEB0->DFFF      >150 bytes
>E400->FFFF      >1C00 bytes
TOTAL            >2918 bytes (10520)
```

AUTOLOADING AND CHAINING PROGRAMS

Let's say you have written a complex program that needs to be in two or more parts. You have put all the character definitions, into the first program and then want that program to run a second program while preserving the characters you just defined. Simple enough, just add RUN "DSKn.PARTTWO to the first program and it will load and run PARTTWO, or you can use CALL RUN(A$) if the disk name is a string variable. If you are using one of the graphics packages in XB 2.8 G.E.M. and want to preserve sound tables, the screen2 or text mode definitions and colors then PARTTWO **must** be in IV254 format.

Internal, Variable 254 is the format used by Extended BASIC when saving or loading large programs that are longer than the available stack space. If the program is large enough it will be saved in this format automatically. If you don't know what format the program has been saved in, you can CALL CAT("pathname.") to catalog the disk and this will display the file types.

CALL SAVEIV("DSKn.PROGNAME")

XB 2.8 G.E.M. has a subprogram that will to force XB to save a program of any length in IV254 format. In the immediate mode, with your program loaded, simply type:CALL SAVEIV("DSKn.PROGNAME") The program will be saved in IV254 format. If the program is very short a line 1 will automatically be added. This is a DATA statement that does nothing other than adding enough bytes to permit saving even the shortest programs in IV254 format. This does not happen   if the program is larger than 256 bytes. This should be compatible with a CF7 device, but has not been tested.

CALL RUN(A$)

RUN is used to run an XB program from a running XB program. The normal way to do this is with RUN "DSK1.PROGRAM", which normally works fine. The only reason for using CALL RUN would be to run a file name contained in a string variable:
10 D$="DSK1.PROGRAM"
20 CALL RUN(D$)   (XB cannot do this)

As long as there is no break in the programs you can chain programs together this way and when the new program runs the screen, colors, character definitions, sprites, etc. will be preserved. But variables are *not* preserved. When the new program runs, prescan will reset all numeric variables to zero and string variables are to a null string. If you want to pass variables from one program to the next you have to store them with CALL LOAD and retrieve them with CALL PEEK, or save them to disk and then read them with the new program. Wouldn't it be nice if there was a way to preserve the variables and automatically pass them along to the new program? Well, there is.

CALL RUNL1(A$)

RUNL1 is used to run an XB256 program from a running program. It is similar to CALL RUN but with a major difference. After it loads the program it runs the first line, *but without doing a prescan or resetting any of the variables of the calling program.* By chaining programs together with RUNL1 you can create a very long XB256 program, and the neat thing is that bypassing the prescan means super fast start up time and that all the variables are automatically carried over into the newly loaded program. You don't have to save them before leaving one program and retrieve them in the next program.

RUNL1 plays some tricks on the XB interpreter and to avoid problems, there are a couple of things that are important to know.

1 – The initial program can be saved in either program format or IV254 format.

2 – A program being loaded with RUNL1 must be in IV254 format

3 – A program being loaded with RUNL1 cannot be longer than the initial XB program at the beginning of the chain. A "Memory Full" error message will be issued if the chained program is longer than the original program. If it is in IV254 format you can use RUNL1 to load and return to the initial program.

4 – The length of the initial program is stored at >24F2. *In the first program only*, include CALL LOAD(9458,0,0) to set this to zero. When a program does CALL RUNL1, if this value is zero, the starting address of the program is stored there. This value is used to check whether subsequent portions of the program are too long.

5 – Because prescan is not performed, the chained programs cannot introduce new variables. They must all be contained in the first program. If the second program uses HCHAR, and the variables X,Y,Z and A$ and they are not used in the first, the first program should include them this way:

```
10 GOTO 100::X,Y,Z,A$::CALL HCHAR
```

This causes prescan in XB to reserve space for the four variables and HCHAR. This line doesn't give a syntax error because XB never gets to the second part of the line.

6 – If the chained programs use DATA statements, they need to include RESTORE or RESTORE line-number so the program knows where to find the DATA.

7 – DEF, SUB and SUBEND (named subprograms) can only be used in the initial program. GOSUB can be used in any of the programs.

Here is a short demonstration showing how to chain programs in XB 2.8 G.E.M.

Enter this program and save as DSK1.DEMO1.

```
100 ! 1st part of demo
105 CALL LOAD(9458,0,0)
110 DIM SR(100)
120 CALL CLEAR :: CALL SCREEN(4)
130CALL CHAR(96,"0103070B0A1B2A2B2A2B4A4B4A7B0E0A0080C0A0A0B0
A8A8A8A8A4A4A4BCE0A0")
140 CALL MAGNIFY(4):: CALL SPRITE(#1,96,5,100,120,-7,0)
150 FOR I=1 TO 100 :: SR(I)=SQR(I):: NEXT I
160 INPUT "ENTER A STRING: ":A$
170 RUN "DSK1.DEMO2"
180 CALL RUNL1("DSK1.DEMO2")
```

Save as DSK1.DEMO1, then enter this program:

```
100 ! 2nd part of demo
110 CALL CLEAR :: CALL SCREEN(12)
120 PRINT A$
130 FOR I=1 TO 100 :: PRINT SR(I):: NEXT I
140 GOTO 140
```

Save this with CALL SAVEIV("DSK1.DEMO2")   You can use SIZE to verify that DEMO1 is larger than DEMO2

Now RUN "OLD DSK1.DEMO1". A sprite appears, then some time elapses while the program generates 100 square roots and puts them in the array SR(). Then you are prompted to enter a string. When you enter the string, line 170 loads and runs DEMO2. The sprite remains on the screen, but then the program breaks with BAD SUBSCRIPT IN 130. You can see that none of the variables were passed. The prescan set the string to "", the variables to 0 and the default array size to 10. This shows that you chaining

programs this way does not preserve strings or variables.

Let's see if we can fix that by changing DEMO1. Type OLD DSK1.DEMO1, delete line 170 and then SAVE DSK1.DEMO1. Now quit and select XB. When the program comes to line 180, it will use RUNL1 to load DEMO2 and go to the first line. But XB has been duped; it has no way to know that another program has been loaded, so no prescan is done. This time A$ and all 100 square roots were retained in memory and passed to DEMO2 where they could be printed. All this happens automatically and with no delay for prescan or to retrieve the variables.