GSBASIC Release Notes

Version 1.0 Beta 4

September 10, 1987

Remember, this is a BETA software release – useful but not 100% reliable or bug free. Take appropriate precautions by saving your source programs frequently.

Important Information

GSBASIC requires a minimum of 96K free memory (including 1 whole free 64K bank for the interpreter, currently 59000+ bytes, plus a 32K data segment, plus 14 bank zero pages) to boot. The standard GSB.HELLO program attempts to expand the data segment to 65K bytes, thus increasing the minimun default requirement to 131+K bytes. The memory management and allocations in the minimum 512K system will just support a 100K data segment and allow a reasonable size program.

The Desktop environment toolbox requires an additional 192K for the toolsets, the LIBRARY dictionaries and toolset memory requirements. The Desktop demo program, PICS, will not run in a 512K system.

Known Bugs

Cosmetic features, such as indentation of FOR .. NEXT loops and spacing after keywords, are not finalized, although program execution is unaffected.

The display of a pathname is truncated to 40 characters when displayed by the CAT command.

Scrolling back in the command buffer to display lines previously entered will sometimes result in garbage displays if long (>80 column) lines have been entered.

Known Limitations

The interpreter returns an integer value of -32768 when an out of range real or larger integer is converted to a single integer IF you are using the original ROMS. SANE in the initial release ROMS does not properly report the INVALID exception back to the interpreter and so the expected

?OVERFLOW ERROR does not occur. SANE has been fixed in the GS ROM update.

GSBASIC requires ProDOS 16 V1.2 or greater (v1.3 is on the GSBASIC distribution disk). The new GetDirEntry function is required by the CATALOG,CAT, and DIR commands. The message ?ProDOS VERSION error will occur if the interpreter is run under an earlier version of ProDOS 16 that doesn't have the GetDirEntry function call.

To make BASIC programs launchable from the finder, the GSBASIC.SYS16 file must reside in the top-level directory of the boot volume.

Documentation Corrections

DATA is not allowed as an alias for the BDF **filtyp** in I/O commands.

Chapter 8 of the manual incorrectly refers to UIR() as UIR%().

GSBASIC recognizes a startup filetype for EXEC as well as RUN. Thus GSB.HELLO can be a SRC file with an AUXID of $BA and GSBASIC will execute at startup as if 'EXEC GSB.HELLO' were entered instead of 'RUN GSB.HELLO'.

The DIR command displays in NORMAL characters. The User is responsible for restoring INVERSE (if needed) after the DIR command has been executed.

Tool Programming

Tool programming from GSBASIC is not for everybody. The GSBASIC reference manual does not provide any tutorial information on tool programming; the Apple IIGS Tool Reference Manual is absolutely indispensible. In general tool call parameters are coded as you would expect given the Tool Reference Manual description. However, in some cases it may be unclear how certain data types were implemented in the GSBASIC TDF's. These release notes include a listing of the text files that were used to create the TDF's. Their format is described in the TDFBUILD appendix. The TDFBUILD utility itself is not available with this release.

The DEMOS subdirectory contains five files. Four of them are files with picture data. They are used by the GSBASIC program PICS, which is also in this subdirectory.

The program PICS is a program that has been used as a demo in other GS programming environments. It is a simple program that opens up four windows and loads pictures into them. It supports all window functions, including scrolling, dragging (reposition the window), resizing, zooming, clipping, hiding, and reshowing windows.

Here is a description of how the program works, from the top to the bottom.

- points prefix 6 at the tdf's
- clears the screen.
- grabs a large chunk of memory.
- set up the data structures that will be used later.
- set up titles for window with Pascal-type strings, accessed via the =^
- loads the quickdraw tdf-file
- calls the procedure "please.wait", which displays just that on the screen. Startup takes so long because big data structures are being initialized interpretively.
- load up the data structures by restoring the data pointer to a certain line, and calling the procedure "readtable", which loads the data from the data statements, and puts it into the byte-arrays for use in tool calls.
- set the window titles
- load the tdf files if the library file was not present
- set up error trapping
- starts up the memory manager and gets a process id.
- start up managers
- set up the menus
- set up the four graphics windows
- draw a window and put text "Loading Pictures" in it
- call the procedure to load the pictures
- hide that window
- make window 1 the current window
- turn off the quit flag
- set up the menu handling routines: about, doquit, mw1, mw2, mw3, and mw4
- set up the event handling routine for hiding a window
- turn on TASKPOLL with an event mask of $FFF

the "GetNextEvent" loop; loops until the user chooses quit
- the About procedure. Turns on the text screen and displays a cute message
- the Quit procedure. Sets the quit flag to 1 and turns off TASKPOLL so that no other events are trapped.

the Window procedures: brings the appropriate window to the front and unhilites the menu bar when done.
- Hides the selected window
- the Please.Wait procedure. Starts up graphics and prints out "One Moment Please..."
- the Msg procedure. Clears the screen and prints the string passed as a parameter
- the Deref function. Gets the word value from the address passed as a parameter
- the DoErr handler. Turns off error trapping and falls into the ShutDown handler.
- the ShutDown handler. Shuts down all of the managers, sets the screen to text, and ends the program.
- the PWin procedure. Copies a picture from an array into a window's Grafport.
- the ReadTable function. Calls DoSet to read data statements and load the data into byte arrays. Returns a pointer to the array.
- the DoSet function. Reads data statements and translates the data found into appropriately sized bytes to put into byte arrays. Has routines to handle integers, double-integers, double-precision reals, and strings. You might want to use this routine to initialize data structures in your programs. It is unbearably slow and desperately wants to be written as an invokable module. It is not the only data-initialization scheme available; it's virtue is that it allows you to easily change your data structure. Once you've got them defined, you may want to create a data file and then simply load it in from your program.
- the data for the menus
- the data for a dialog
- the data for the four windows
- miscellaneous data (size of window,...)
- the draw routines for the four windows
- the LoadPics procedure. Loads the four pictures from disk and stores them into byte arrays.

Note: it is the programmer's responsibility to correctly shut down all tools that have been initialized by an application. Failure to correctly shut down tools can result in unpredictable behavior after BASIC has been exited.

BUG REPORT

Written bug reports, describing completely REPEATABLE bugs, with example source listings, should be sent to: GSBASIC Product Management, MS 27-S, Apple Computer, Inc., 20525 Mariani Avenue, Cupertino, CA 95014 If source code required to demonstrate bug is lengthy, inclusion of disk copy is appreciated.

Problem:

Name, Address, Telephone Number:

Comments and suggestions are also welcome.