Tech Info Library

GS BASIC: Overview of Features

Revised: 10/19/87
Security: Everyone

GS BASIC: Overview of Features

==========

This article last reviewed: 15 October 1987

GS BASIC vs Applesoft BASIC

Unlike Applesoft BASIC, GS BASIC gives programmers access to all of the new features of the Apple IIGS: Super hi-res graphics, the Ensoniq sound chip, expanded memory, the new Tools, and other enhanced ROM facilities.

GS BASIC is an entirely RAM-based language. It does not rely on Applesoft, nor is it related to it in any way except that they are both BASIC languages. In fact, because of advanced features like access to all of GS RAM, more powerful I/O instructions, and access of external routines, GS BASIC is not 100% compatible with Applesoft. The number of commands is increased from 120 to over 200 (see List of GS BASIC Commands), many Applesoft BASIC commands do different things in GS BASIC, and some Applesoft commands do not appear in GS BASIC at all (PR#, IN#, and HGR for example). Applesoft and GS BASIC files are stored differently on disk, and are not interchangeable. Finally, GS BASIC is designed to run under ProDOS 16, whereas Applesoft can run only under ProDOS 8. Otherwise, GS BASIC would not be able to take advantage of the GS's powerful new features.

GS BASIC gives the programmer twice the number of variable types that Applesoft has. In addition to Integer, Real, and String types, GS BASIC provides Double Real, Double Integer, and Long Integer.

GS BASIC is not as fast as Applesoft. Applesoft runs the following program in 4 seconds, while GS BASIC takes 43 seconds:

```
10 FOR i = 1 TO 10000
60 NEXT i
```

With longer programs, the difference is less. This program takes Applesoft 48 seconds and GS BASIC 217 seconds:

```
10 FOR i = 1 TO 10000
```

```
   20 a=i-i
   30 a=i+i
   40 a=i/i
   50 a=i*i
   60 NEXT i
```

Access to External Routines

GS BASIC's main advantages is its powerful library and external routine facilities. Provided with GA BASIC are many ToolBox Definition Routines (TDFs) that let programmers call the GS's ToolBox routines (QuickDraw II, Menu Manager, Window Manager, Sound Manager, and so on). These routines are setup with the LIBRARY command, and called with the CALL and EXFN commands.

Programmers can also define and call their own library routines. These are loaded with the INVOKE command, and called with the PERFORM and EXFN commands.

Display Format

GS BASIC offers improved formatting of listings. Users can specify how much space appears between a line number and the first statement, and how far to indent when listing program statements within a FOR...NEXT or DO...WHILE...UNTIL loop. See the List of Built-in Constants and Reserved Variables.

Sound

Sound is created through Sound Manager routines.

Built-in Constants and Variables

GS BASIC includes pi calculated to 20 digits. Also, GS BASIC stores many system parameters in reserved variables. Much of the system's status can be determined and changed by examining and setting these variables. For example, the cursor's screen position is stored in the variables VPOS and HPOS. You can move the cursor simply by assigning new values to these variables.

Labels

GOTO, GOSUB, and ON xxx commands can point to labels instead of line numbers, making programs much easier to follow. Instead of:

```
   1234 GOSUB 50: REM Read a character from the modem
```

the programmer can type:

```
   1234 GOSUB READ_MODEM
```

Looping

GS BASIC includes three looping structures not in Applesoft:

```
   IF...THEN...ELSE
   FOR...NEXT...STEP
   DO...WHILE...UNTIL
```

These structures, common in other languages such as Pascal and Fortran, give the programmer greater flexibility in program control. In addition, the IF...THEN structure has multi-line capabilities. This means that all statements that appear between the IF and the THEN do not have to appear on the same line.

PRINT USING and INPUT USING

PRINT USING and INPUT USING have long been a features of BASICs other than Applesoft. GS BASIC implements them with a vengeance, making them nearly as powerful as FORTRAN format statements. A programmer has many ways to specify how to print data to a screen and read it from a keyboard or other device.

When defining a USING statement, a programmer can specify it in many ways. The first way is to specify the format on the same line, right after the USING statement. The second way is to assign the formatting string to a variable (such as FMT$), and then enter that variable after the USING statement. This allows the programmer to use the same formatting in many PRINT statements. Or, if the programmer doesn't want to use variable space to store a formatting string, he can use the IMAGE command to define it. The programmer can then simply refer to the line on which it appears in the PRINT USING statement. All of these options apply to INPUT USING as well.

PRINT and and INPUT USING don't merely have to work with the text screen. With optional device extensions, a programmer can PRINT or INPUT to or from a file, or PRINT to a printer or even the Super-Hi-res graphics screen.

Finally, GS BASIC offers many formatting commands that are to be placed in the formatting strings. These are divided into three groups: String Spec, Literal Spec, and Numeric Spec. Numeric Spec is further divided into three sub-sections: FIXSPEC, SCISPEC, and ENGRSPEC. The SCALE command can be used to accurately reposition a decimal point in a floating point number.

Tech Info Library Article Number:798