[7.38] Why use TI Basic?

TI Basic is the built-in programming language of the TI-89 and TI-92 Plus calculators. It is one of several programming languages you can use; the other languages are M68000 assembly language and two C compilers. "TI Basic" is not Texas Instruments' name for the built-in programming language. In fact, the TI-92 FAQ says this about it:

> ***"Programming language of the TI-92 - is it BASIC?***
> *No. There are a number of features that are similar to the BASIC programming language, but it is not BASIC."*

The *Getting Started* web page for the TI-89/92+ SDK has the only TI reference to "TI Basic" I have found, but I will use that term since it is common and well-understood in the calculator community.

Some programmers claim that TI Basic is unsuitable for coding, citing real and imagined advantages of C. TI Basic does have some serious limitations, but there remain several compelling reasons to use it:

- TI Basic is built into every calculator, and programs can be completely developed on the calculator. You don't need an external interpreter, assembler, compiler, editor or development system.
- The TI Graph Link software can be used for program development if desired. Programs can be edited on the PC, then downloaded to the calculator for execution and debugging.
- The learning curve for TI Basic is short and shallow compared to learning C or assembler.
- TI Basic is stable and robust. It is either difficult or impossible to crash the calculator with a TI Basic program.
- There is no 24K ASM limit for TI Basic programs. You need no hacks or patches to run large TI Basic programs.
- TI Basic functions may return results to other functions or programs, or to the home screen.
- TI Basic functions may be used in expressions, just like built-in functions. This can be done with C functions, but it requires a patch and a hack with AMS 2.05 and HW2 calculators.
- TI Basic can be used to add functional extensions to C or ASM programs which have reached the 24K ASM limit.
- TI Basic is extremely well documented. The user's guide thoroughly describes the language and provides many examples. The user's guide is available in many languages as both a printed document and an Acrobat PDF file. If this isn't enough, there are a few web tutorials to help you get started.
- TI Basic is 'fast enough' for many real applications. This is certainly the case when the application speed is bound by CAS or numeric operations, since both C and TI Basic use the same system calls to perform these functions.
- TI Basic is reasonably complete and sophisticated.
- TI Basic is popular, so there is a large library of existing functions and programs you can use.
- TI Basic is similar to other Basic dialects, so a huge code resource is available in books, magazines and web archives. It is trivial to translate programs from any version of Basic to TI Basic.
- Because of its popularity you can easily get help by posting on the TI discussion groups.
- TI Basic can be extended with assembler and C programs, to fill some of the more glaring functional voids.

In spite of these advantages, there are several areas in which TI Basic could stand some improvement. Loops are extremely slow. Debugging facilities are essentially nonexistent; programs cannot be

single-stepped and there are no breakpoints and no facilities for watching variables. TI Basic lacks system function calls, although these can be accomplished with ASM program extensions.

Eric S. Raymond, in *The Jargon Lexicon,* version 4.3.1, gives this definition for BASIC:

> *BASIC /bay'-sic/ n.*
>
> *A programming language, originally designed for Dartmouth's experimental timesharing system in the early 1960s, which for many years was the leading cause of brain damage in proto-hackers. Edsger W. Dijkstra observed in "Selected Writings on Computing: A Personal Perspective" that "It is practically impossible to teach good programming style to students that have had prior exposure to BASIC: as potential programmers they are mentally mutilated beyond hope of regeneration." This is another case (like Pascal) of the cascading lossage that happens when a language deliberately designed as an educational toy gets taken too seriously. A novice can write short BASIC programs (on the order of 10-20 lines) very easily; writing anything longer (a) is very painful, and (b) encourages bad habits that will make it harder to use more powerful languages well. This wouldn't be so bad if historical accidents hadn't made BASIC so common on low-end micros in the 1980s. As it is, it probably ruined tens of thousands of potential wizards.*
>
> *[1995: Some languages called `BASIC' aren't quite this nasty any more, having acquired Pascal- and C-like procedures and control structures and shed their line numbers. --ESR]*
>
> *BASIC stands for "Beginner's All-purpose Symbolic Instruction Code". Earlier versions of this entry claiming this was a later backronym were incorrect.*

Mr. Raymond finds it necessary (or at least amusing) to include Edsger Dijkstra's hoary quote with the memorable phrase 'mentally mutilated'. Even so, it is possible to write clear, coherent maintainable code in BASIC, just as it is possible to code a convoluted mess in C++ or Perl. Regardless, TI Basic *is* what we have on the calculator, so it may be more productive to make the best of the situation rather than moan about its deficiencies.

While TI Basic is particularly suited for quick, short programs and functions, it is certainly possible to use it to code significant applications: Roberto Perez-Franco's symbolic circuit simulator *Symbulator* is one good example. Applications which require maximum speed and fine control of the keyboard and display, as well as access to system functions, are best coded in C.

If you want to learn calculator programming, TI Basic is a good place to start, comments of Raymond and Dijkstra notwithstanding. You will get working results faster than trying to learn C if you have never programmed before. You may find that TI Basic is all you need. If not, it will be easier to learn C once you have some TI Basic experience.