# LabVIEW Graphical Programming
(491 words)

National Instruments LabVIEW$^{TM}$ is a revolutionary programming language that depicts program code graphically rather than textually. LabVIEW excels in data acquisition and control, data analysis, and data presentation, while delivering the complete capabilities of a traditional programming language such as Microsoft Visual C. One major benefit of using graphical programming rather than text-based languages is that you write programs simply by connecting icons. In addition, graphical programming solutions offer the performance and flexibility of text-based programming environments but conceal many programming intricacies like memory allocation and syntax.

With the LabVIEW graphical programming environment, you build virtual instruments (VIs) instead of writing programs. VIs consist of two distinct parts – a front panel user interface for interactive control of the software system and a block diagram, the natural design notation for engineers and scientists that specifies the function of the system.

## Creating the Front Panel
To create the user interface for a VI, you place the controls and data displays for your measurement system on the front panel by choosing objects from the Controls palette, such as numeric displays, knobs, meters, gauges, thermometers, tanks, LEDs, charts, and graphs. Then, you control your system at runtime by simply operating the various objects on your front panel, whether it be moving a slide, zooming in on a graph, or entering a value from the keyboard.

## Constructing the Graphical Block Diagram
You can construct a block diagram to define the behavior of a VI without worrying about the many syntactical details of conventional programming. You select objects, or icons, from the Functions palette and connect them with virtual wires to pass data from one block to the next. These blocks range from simple arithmetic functions to advanced acquisition and analysis routines, to network and file I/O operations.

## Dataflow Programming
LabVIEW employs a patented dataflow programming model that frees users from the linear architecture of text-based languages. Because it is the flow of data between objects on a block diagram, and not sequential lines of text, that determines execution order in LabVIEW, you can create diagrams that simultaneously execute multiple operations. Consequently, LabVIEW is a multitasking system capable of concurrently running multiple execution threads and multiple VIs.

## Modularity and Hierarchy
Because LabVIEW VIs are modular in design, any VI can run on its own or operate as part of another VI (subVI). With this modularity, you can design a whole hierarchy of VIs and subVIs that serve as building blocks in any number of applications. You can then modify, interchange, and combine VIs with ease as application needs change.

## Graphical Compiler
In many applications, execution speed is a critical consideration. LabVIEW is the only graphical programming system with a compiler that generates optimized code with execution speeds comparable to compiled C programs. To further improve performance, you can analyze and optimize time-critical sections of code with the built-in Profiler. In this way, you increase productivity with graphical programming without sacrificing execution speed.