INTRODUCTION TO NOVASOFT PROGRAMMING

NOVASOFT-G PROGRAMMING:

Novasoft is GUI (Graphical User Interface) based programming environment in which you can create programs using a graphical notation (connecting functional blocks). It is an interactive program development and execution system designed especially for students and enthusiastic in Robotics, to program as part of their learning and Working. The software provides a visual programming language for writing simple programs and downloading them to the Novabot Brick. This means that rather than requiring users to write lines of code, they instead can use flowchart like "blocks" to design their program.

However, Novasoft is much more than a text based programming languages. The Novasoft development environment works on computers running Windows, Mac OS. Using the very powerful graphical programming language that many Novasoft users affectionately call "G" (for *graphical).* Programs that take long to write using conventional programming languages can be completed quickly using Nova Soft because it is specifically designed to take measurements, analyze data, and present results to the user. And because Novasoft has such a versatile graphical user interface and is so easy to program with, it is also ideal for simulations, presentation of ideas, general programming, or even teaching basic programming concepts.

Dataflow and the Graphical Programming Language:

The Novasoft program development environment is different from standard C and Java development systems in one important respect: While other programming systems use text-based languages to create lines of code,

Novasoft uses a graphical programming language, to create programs in a pictorial form called a *block diagram*.

Graphical programming eliminates a lot of the syntactical details associated with text-based languages, such as where to put your semicolons and curly braces. Graphical programming allows you to concentrate on the flow of data within your application, because its simple syntax doesn't obscure what the program is doing.

Novasoft uses the better terminology, icons, and ideas familiar to Students and Trainers. It relies on graphical symbols rather than textual language to define a program's actions. Its execution is based on the principle of dataflow, in which functions execute only after receiving the necessary data. Because of these features, you can learn Novasoft even if you have little or no programming experience. However, you will find that knowledge of programming fundamentals is very helpful.

How Does Novasoft Work?

A Novasoft program consists of one or more virtual instruments (VIs) blocks. These are called such because their appearance and operation often imitate actual physical instruments. The *block diagram* is the component source code, constructed in Novasoft graphical programming language. The block diagram is the actual executable program. The components of a block diagram are lower-level VIs, built-in functions, constants, and program execution control structures. You drag blocks to connect the appropriate objects together to define the flow of data between them. Front panel objects have corresponding terminals on the block diagram so data can pass from the user to the program and back to the user.

G Programming Language

- Shorter learning curve than traditional text-based programming

- Naturally represents data-driven applications with timing, sensitivity and Speed.
- Intuitive, flowchart-like dataflow programming model

Using it, you can quickly tie together data acquisition, analysis, and logical operations and understand how data is being modified. From a technical standpoint, G is a graphical dataflow language in which blocks (operations or functions) operate on data as soon as it becomes available, rather than in the sequential line-by-line manner that most programming languages employ. You lay out the "flow" of data through the application graphically connecting the output of one block to the input of another. The practical benefit of the graphical approach is that it puts more focus on logic and the operations being performed on that logic, and abstracts much of the administrative complexity of computer programming such as memory allocation and language syntax.