STATE MACHINES

State Machines are really easy to implement in LabVIEW. For most programs that are sizeable or control something, I highly recommend using state machines especially for medium sized programs or bigger.

State Machines allow programs to be non-sequential by giving you flexibility to move to any state you want.

From experience, every time I don't use a state machine for a larger program, I always wished I had.

Typical States

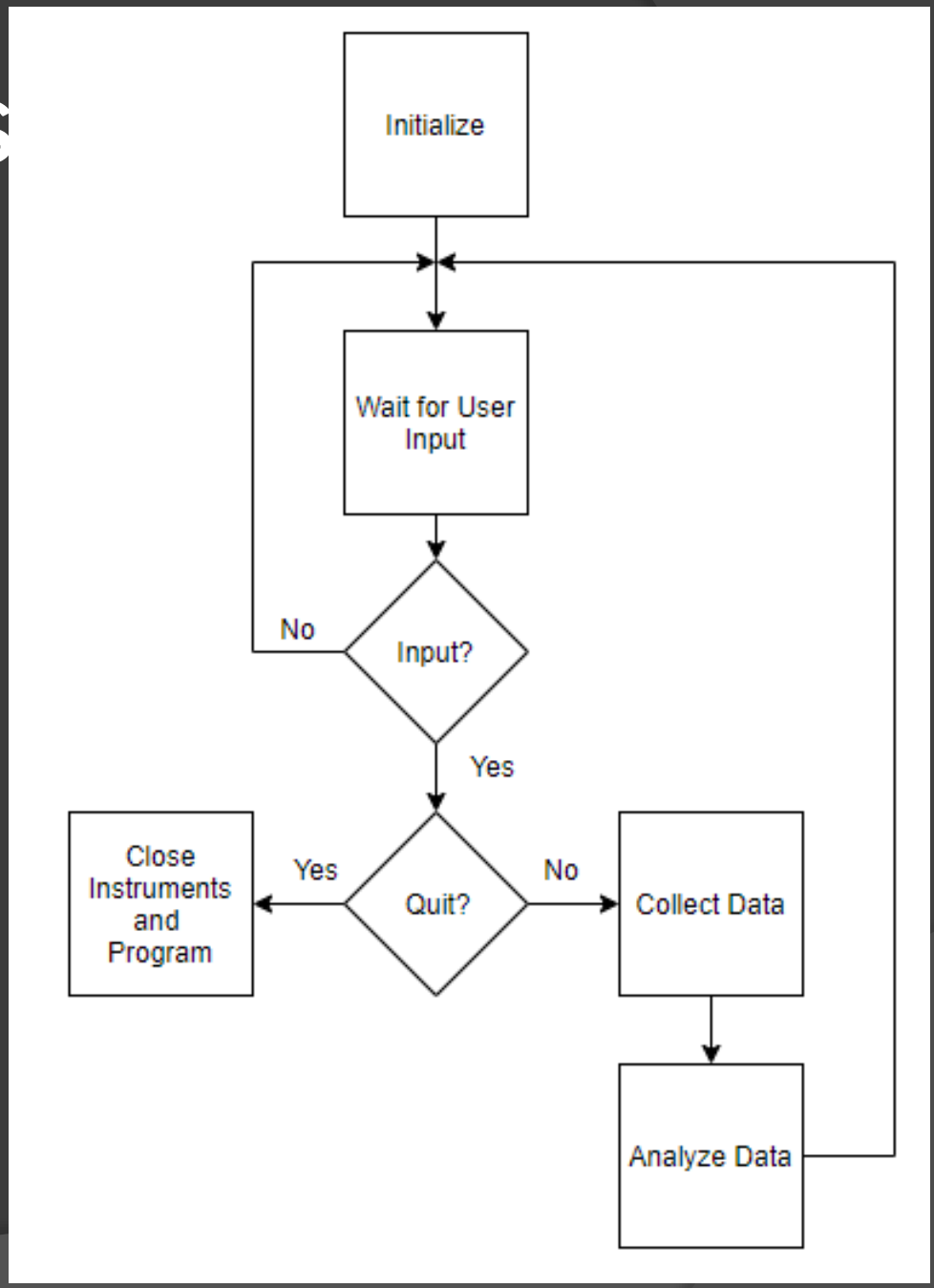
Typical states include, but not limited to …

- Initialize (set up instruments, clear indicators, etc.)
- Wait for User Input
- Collect Data
- Analyze Data
- Stop/Close (close down instruments, etc.)

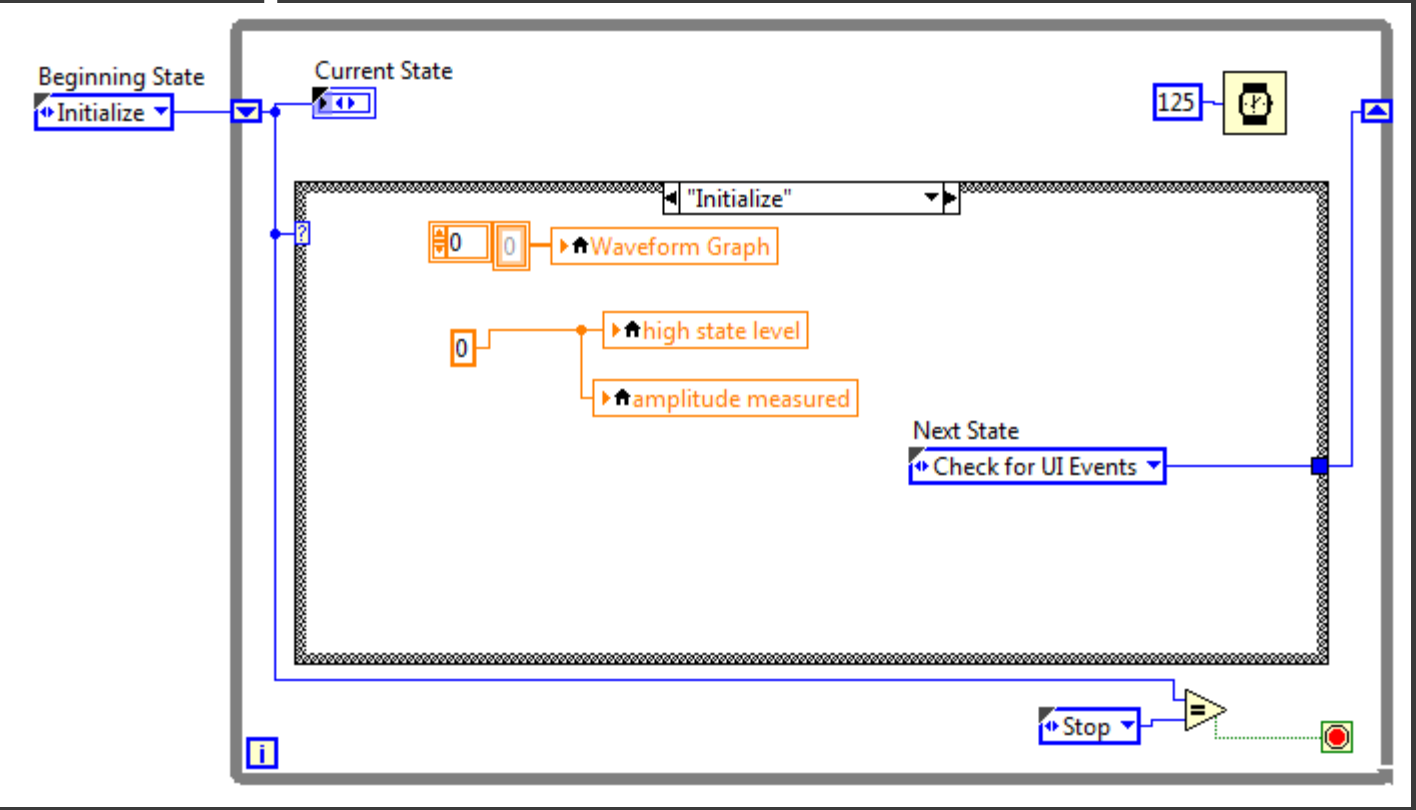
Error (handles errors – usually either shutdown program or resolve the error)

Note: States in orange are almost always used.

Typical States



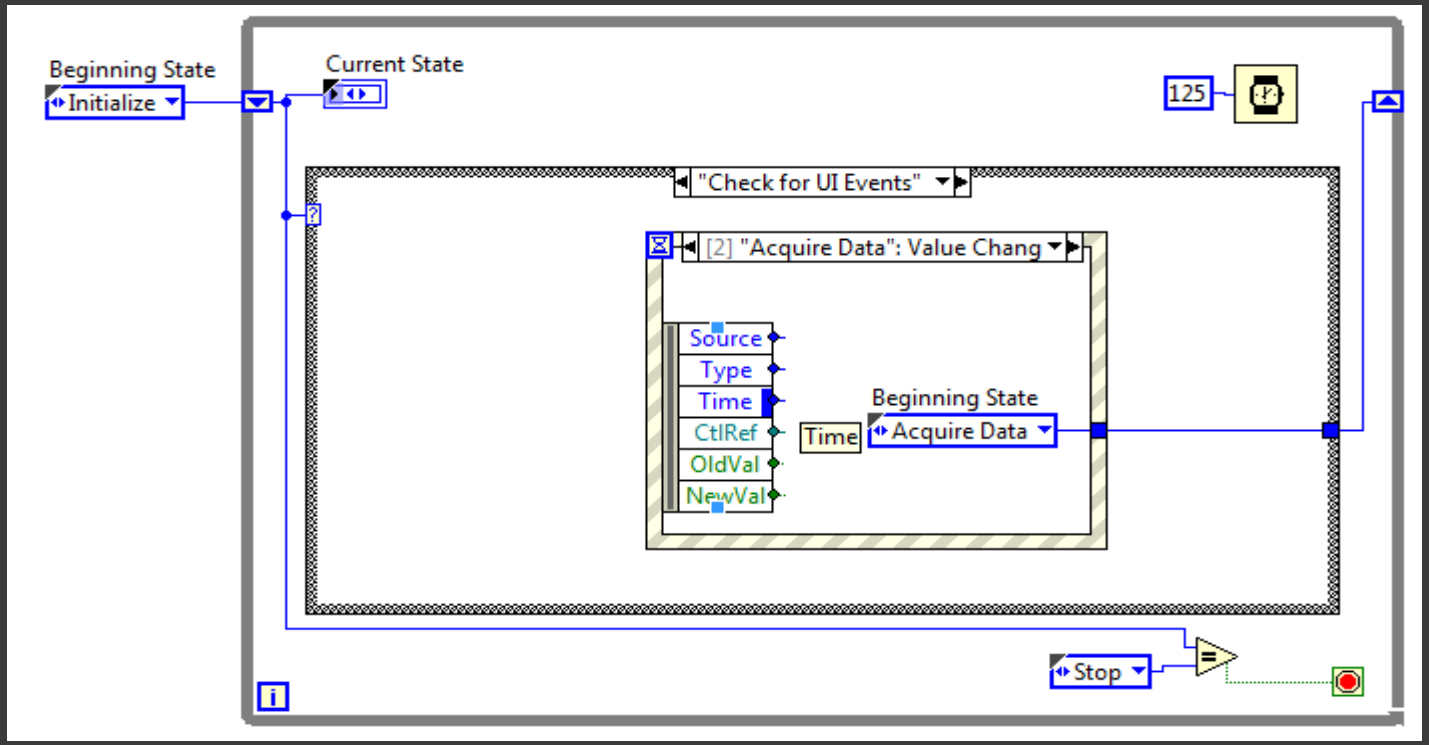
Example



Notice that the first state is “Initialize.” This is where we clear the graphs and indicators. We could also set up instruments, etc.

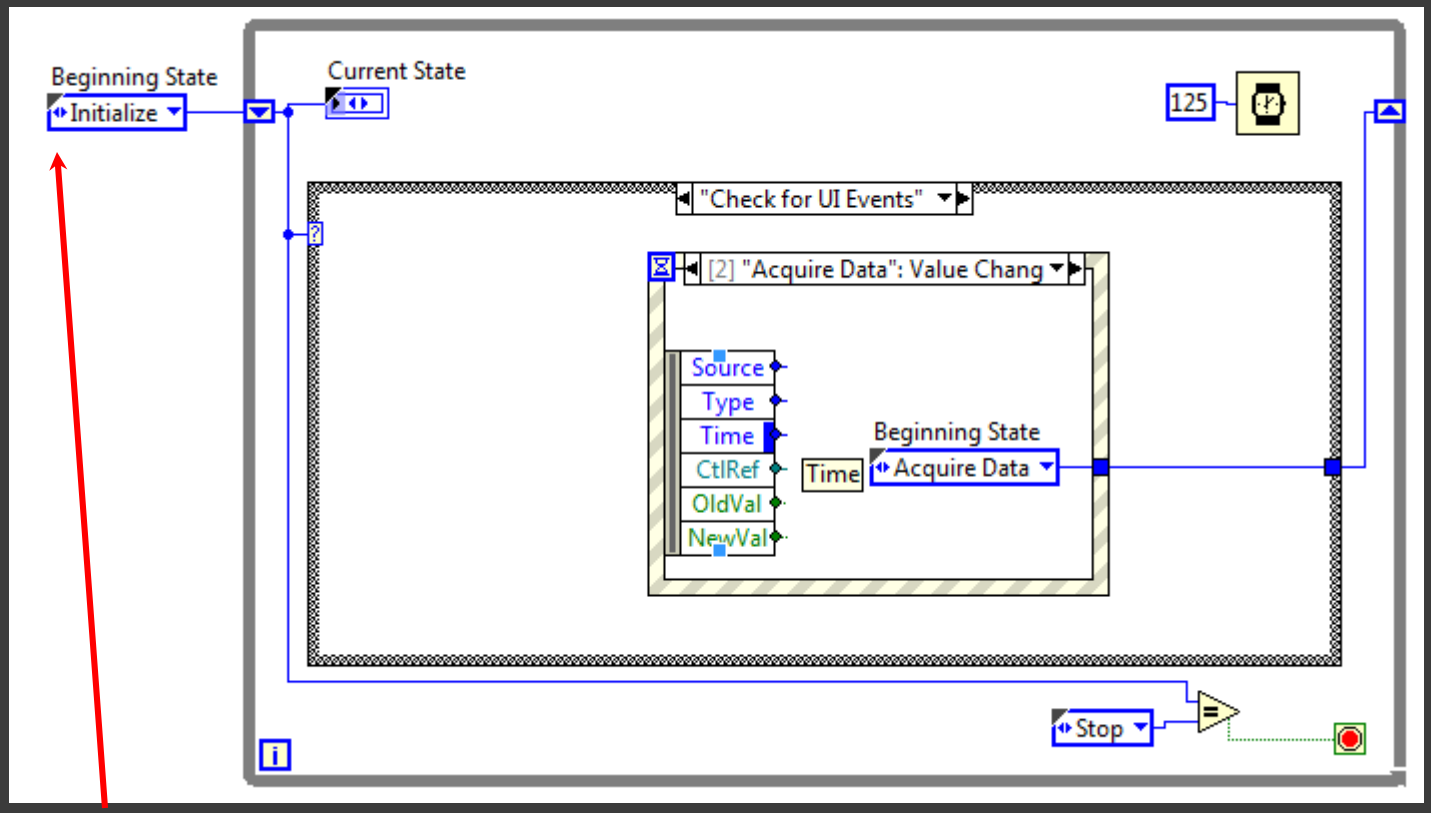
Also notice that if we go to the “Stop” state, then the while loop stops as well.

Example



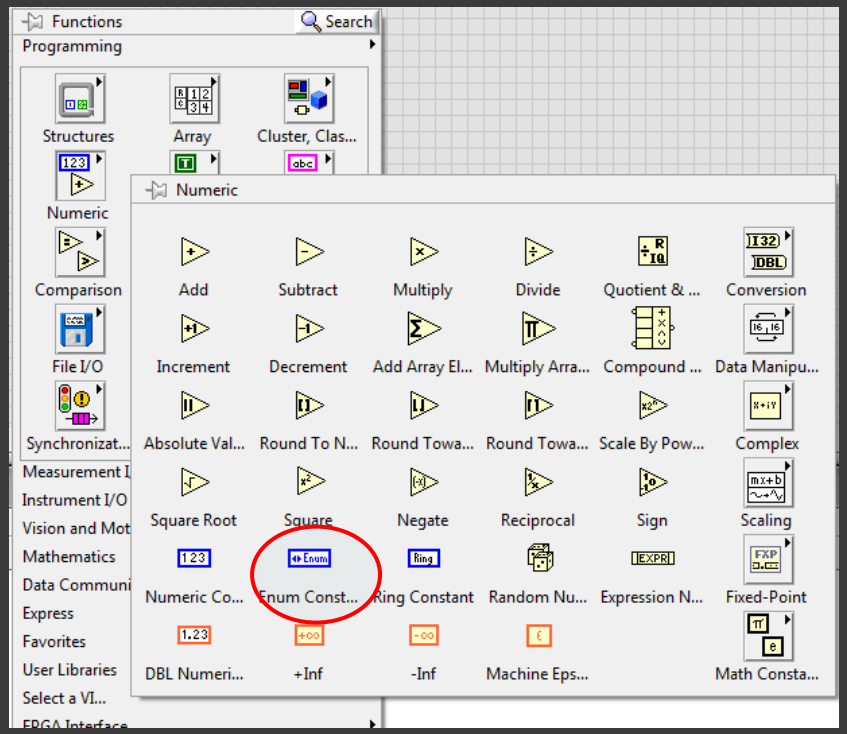
Here we are waiting for user input. If the “Acquire Data” button is pressed, then we go to the “Acquire Data” state. Most of time you probably won’t have an Acquire button. Typically, we’ll use an event structure here. The advantage to this is that the computer doesn’t waste computational time polling for user inputs. So the computer really doesn’t have to work hard until there is some user input.

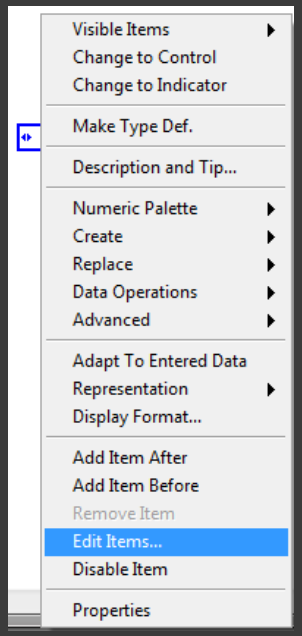
Enumerated Constants

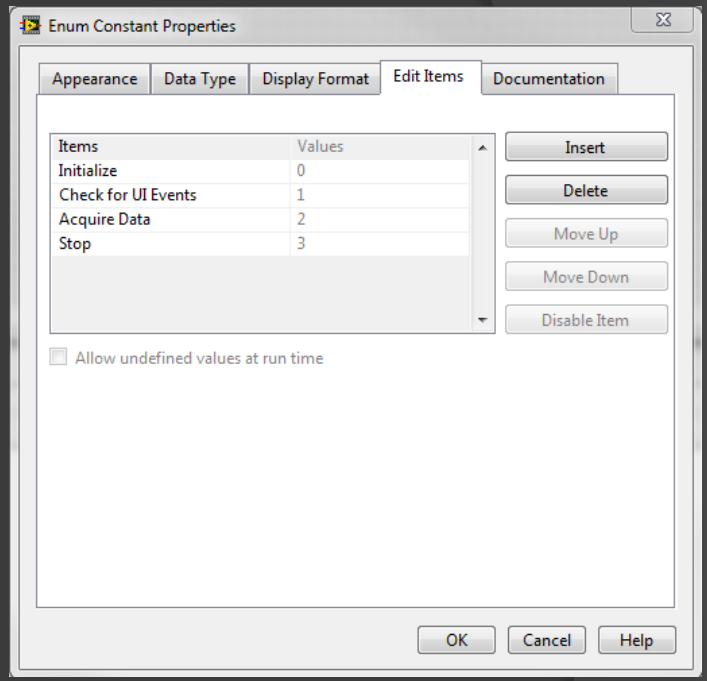


Also notice that we are using an enumerated type. An enumerated constant is simply a set of text values with each value having a number associated with it. So “Initialize” has a value of 0, “Check for UI Events” has a value of 1, etc. A benefit to using enumerated constants is readability: it connects a label to a number.

Enumerated Constants

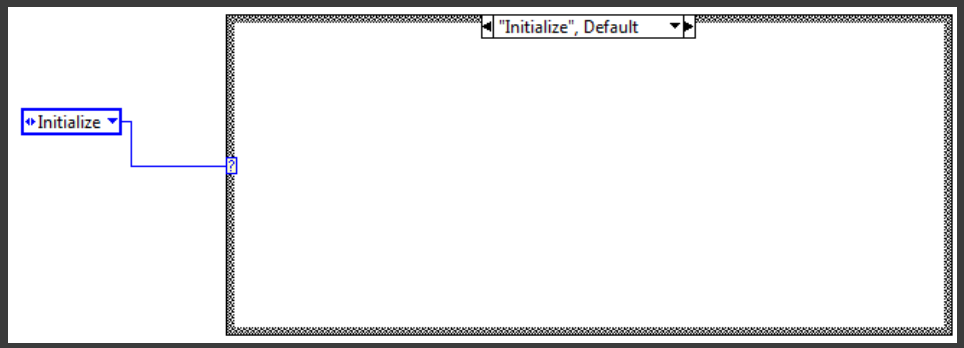


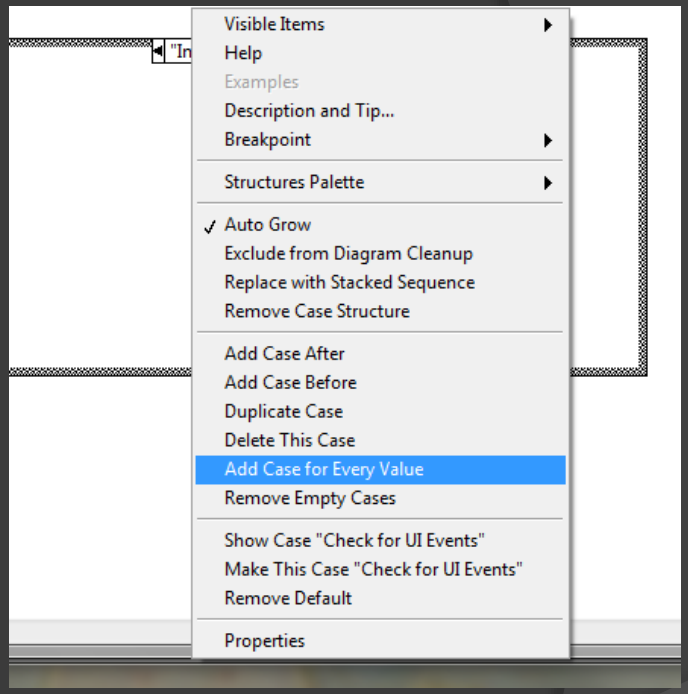




Enumerated Constants are found in the Numeric Palette.
Once it is placed on the block diagram, you can Edit Items.
In this example, the enumerated constant had items . . .
Initialize, Check for UI Events, Acquire Data, Stop

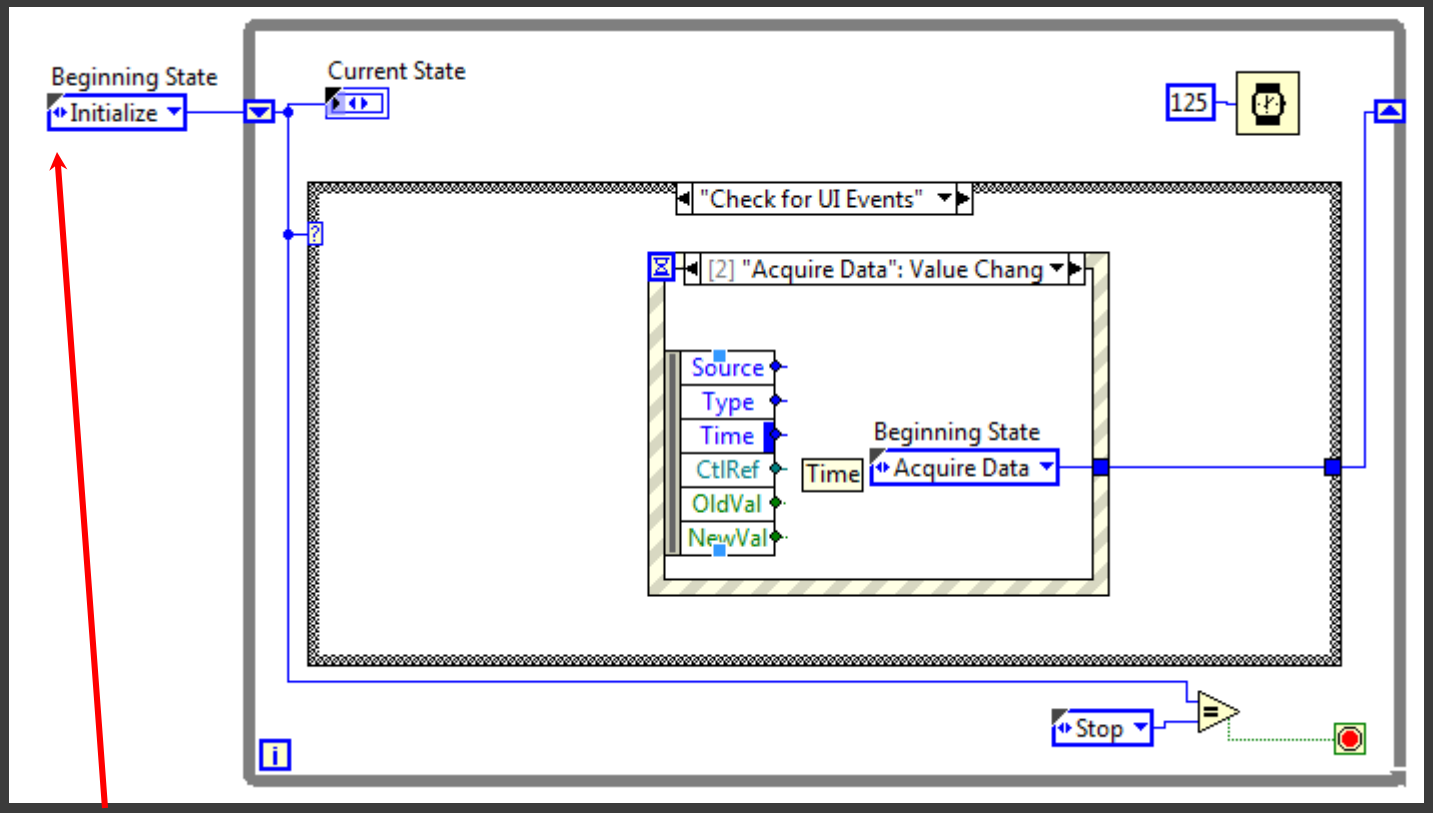
Enumerated Constants and Case Structure





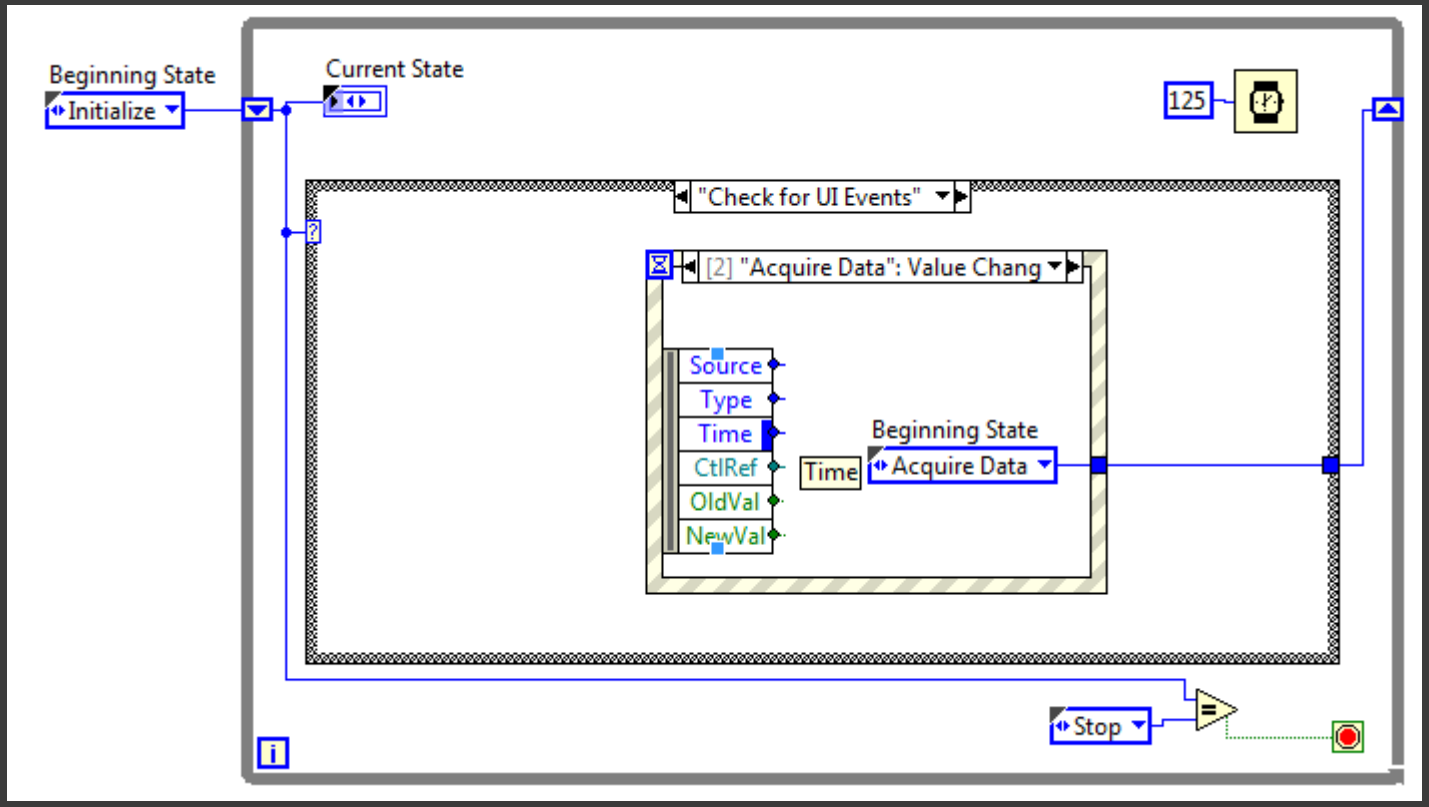
Once you connect the enumerated constant to the Case Structure, the different cases show up automatically. If some cases don't appear, then you have to choose "Add Case for Every Value."

Type Definitions



Notice that the enumerated constant has a small black triangle in the top left corner. This means that this enumerated constant has been converted to a type definition. A type definition forces the control to be the same everywhere it is used in this program (or if it is used in other VIs, it will be the same for those programs as well). In this slide, I see 3 instances of the type definition.

Type Definitions



So if you change the enumerated constant, it will change all the other enumerated constants. The Type Def. is created as a separate file with a .ctrl extension. You have to save this separate file for the type definition change to take effect.

LabVIEW Project

Ideally, you should create a project when you have multiple files associated with one program. You would include the enumerated constant .ctrl file as well as all of your subVIs.