QUEUED STATE MACHINE

WHAT IS A QUEUED STATE MACHINE (QSM)?

Queued State Machine is similar to the Classic State Machine, but is more advanced and allows more flexibility at a cost of some added complexity. Typically used in midsize to larger programs.

CLASSIC VS. QUEUED STATE MACHINE

In the Classic State Machine, there always exists a state such as “Wait For User Input” that tells the state machine which state to go to next when various events happen.

The Queued State Machine does the same thing, except it essentially runs the “Wait For User Input” state in a separate while loop. It uses queues so that it keeps track of any events that happen while not losing any events.

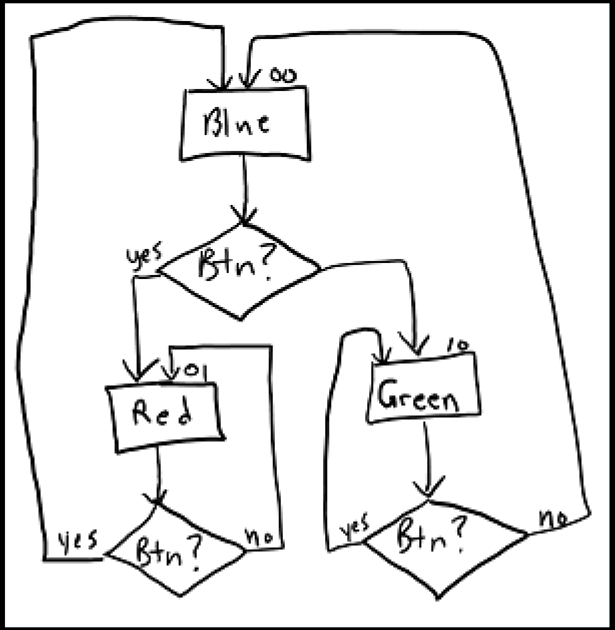
QUEUED STATE MACHINE

The Queued State Machine typically has these two while loops: Event Handler Loop and the State Machine Loop and sometimes other while loops to handle other independent processes.

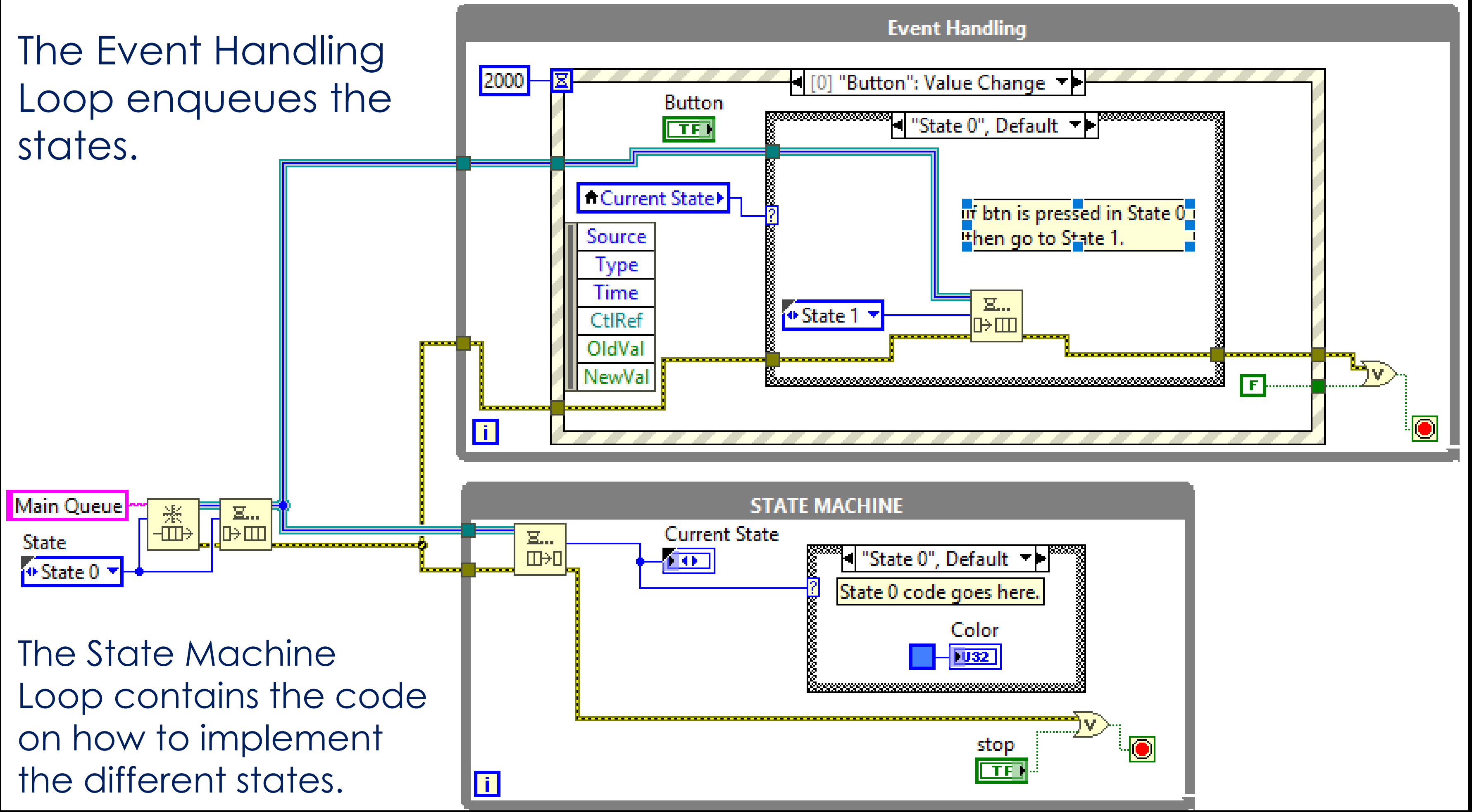
The Event Handler Loop: uses an event structure to handle any events that happen in the program by telling the state machine which states it needs to go to next. Once an event happens, the Event Handler Loop enqueues the necessary states that need to happen in order to handle that event. It uses queues so that it doesn't miss any events.

The State Machine Loop: contains the algorithmic state machine. It waits around until the Event Handler Loop tells it what states need implementing.

EXAMPLE OF WHAT A QSM LOOKS LIKE

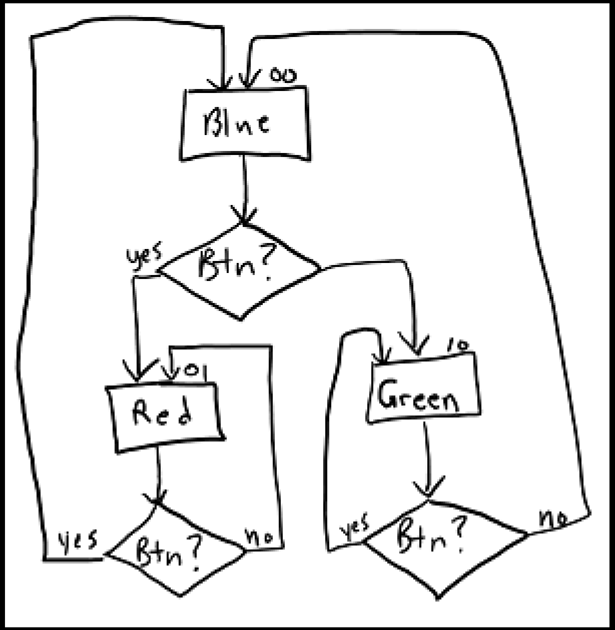


The Event Handling Loop enqueues the states.



The State Machine Loop contains the code on how to implement the different states.

EXAMPLE OF WHAT A QSM LOOKS LIKE

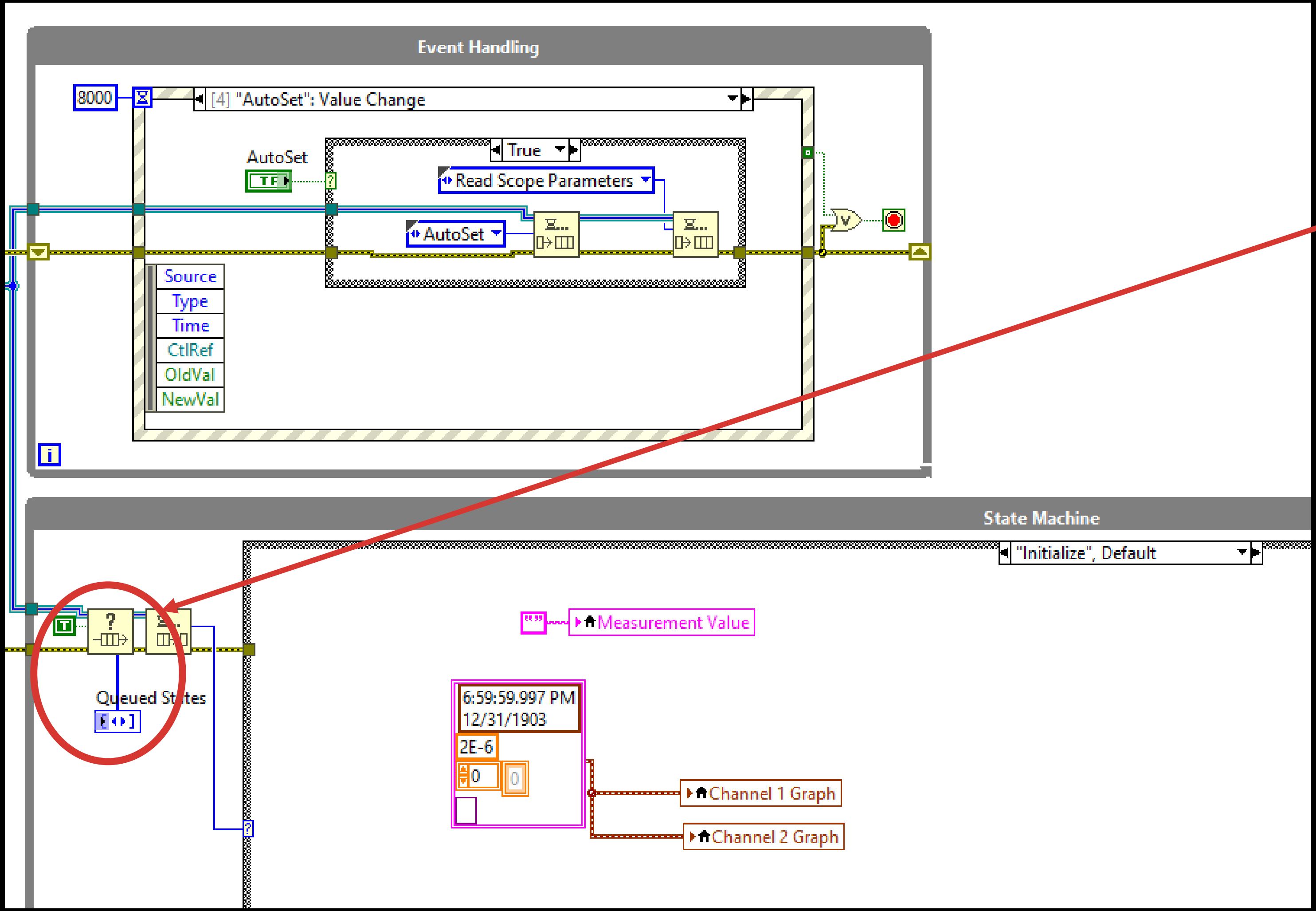


In the Classic State Machine LabVIEW model, we typically had a "Wait for User Input" state that would handle any events and we always cycled back to this state.

However, this would be hard to implement with this state machine. Notice that "Btn" = yes could put you in State 01, State 00, or State 10. If you were in the "Wait for User Input" state, then you wouldn't necessarily know which state you were currently in which would be a problem as to where to go to next.

The Queued State Machine LabVIEW model fixes this problem. ☺

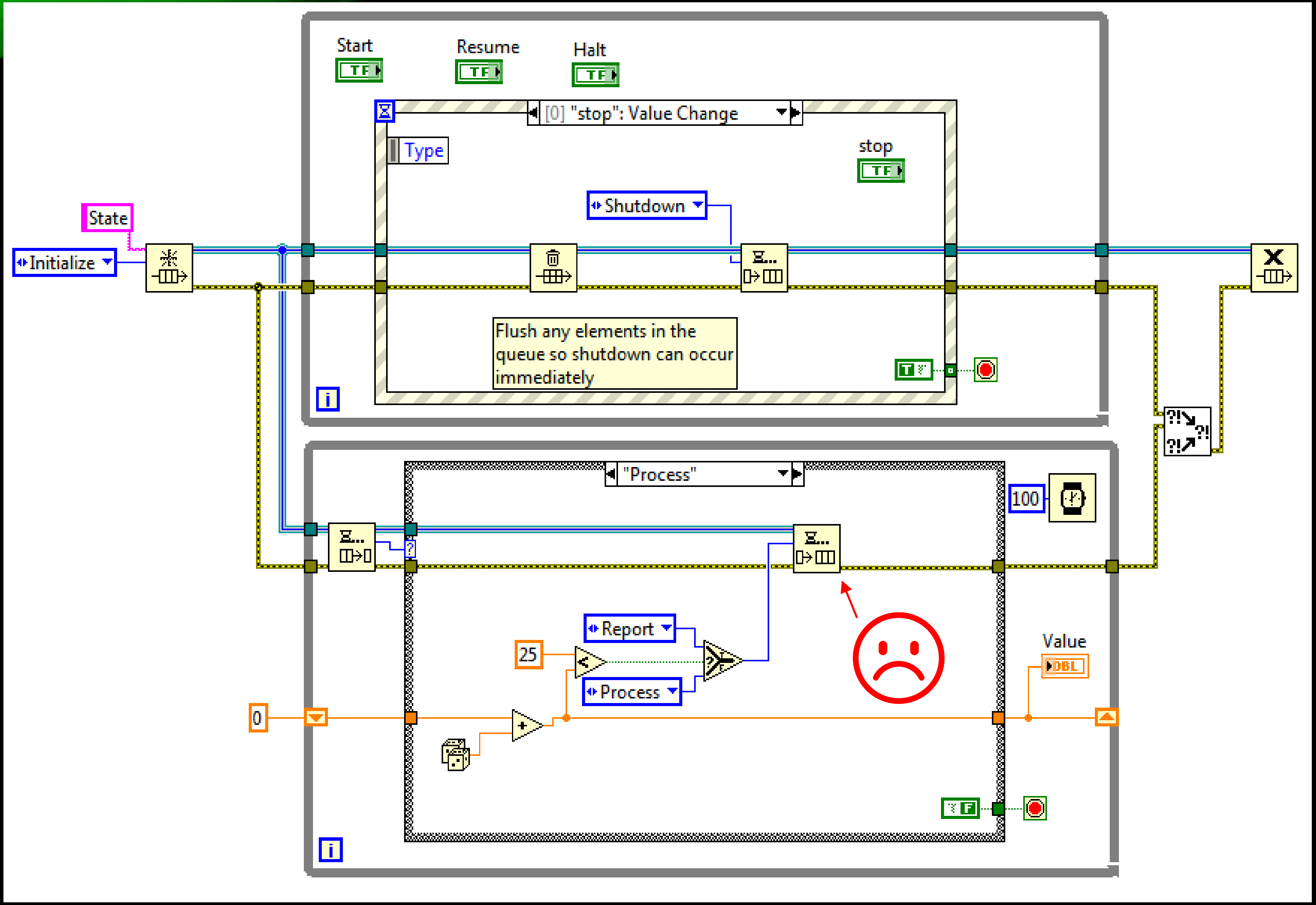
Displaying All States in the Queue



You can use "Get Queue Status" with "return elements" set to True to display the array of the elements in the queue.

This will show the user which states are waiting to be implemented.

BAD EXAMPLE FROM NI FORUMS . . .



The problem with this example is that the State Machine Loop (SML) is enqueueing states, which could happen at the same time that the Event Handler Loop is enqueueing states. This means that the program could end up with states being thrown in during the middle of a sequence of states.

It is typically best practice to only have the Event Handler loop enqueuing states.

ADVANTAGES OF USING A QSM OVER A REGULAR STATE MACHINE

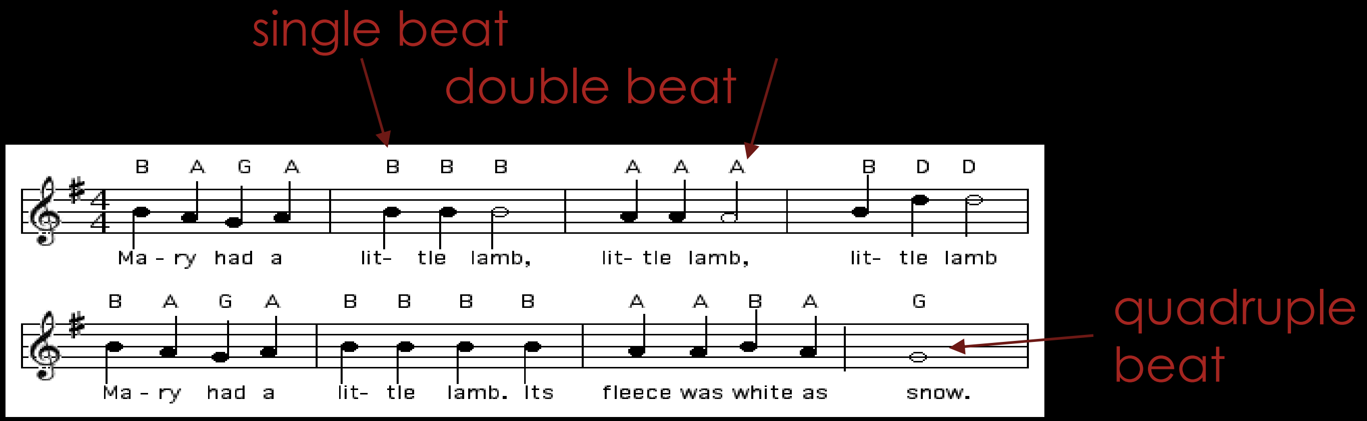
- **Easier for the programmer to perform a sequential task or a set of sequences.** For instance, if you had 8 valves and you needed to open valve number 4, then valve 7, then valve 1, etc. then you would want to implement a QSM. Also use a QSM if you need to repeat something multiple times such as opening and closing a valve 50 times.
- Useful for applications where **multiple tasks occur in parallel**, often at different rates. For example, if our program was acquiring a signal from an instrument and acquiring a signal from a DAQ board simultaneously, then without a QSM we would need a separate queue for each loop.
- Allows the programmer to handle events without always having to return back to the “Wait for User Input” state.

USING VARIANTS

A **variant** is a variable that doesn't have a defined type. Queued State Machines use variants to enqueue data that goes with a particular state.

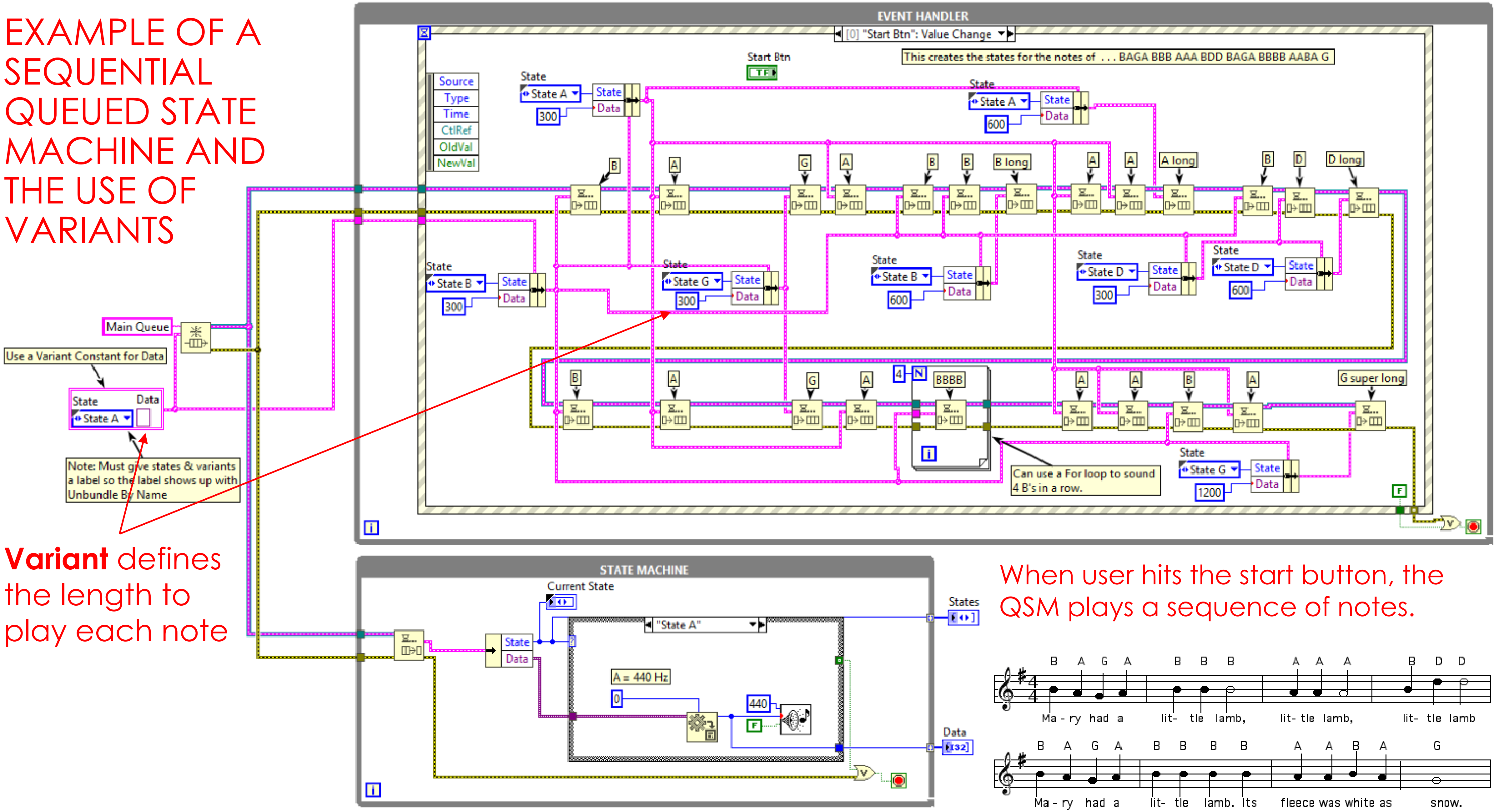
For example, in the next slide a QSM is demonstrated that plays "Mary Had a Little Lamb." Each state in the state machine played a different note. Sometimes I wanted the states to play notes for a single beat and sometimes I wanted the notes to play longer for two beats. I used a variant to tell how long each state should play each note.

The following QSM also shows how **sequential** states are implemented.


single beat
double beat
quadruple beat
B A G A B B B A A A B D D
Ma - ry had a lit- tle lamb, lit- tle lamb, lit- tle lamb
B A G A B B B B A A B A G
Ma - ry had a lit- tle lamb. Its fleece was white as snow.

Mary Had a Little Lamb.vi

EXAMPLE OF A SEQUENTIAL QUEUED STATE MACHINE AND THE USE OF VARIANTS



This creates the states for the notes of . . . BAGA BBB AAA BDD BAGA BBBB AABA G

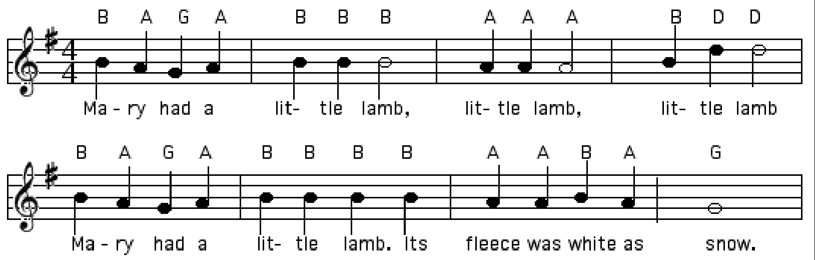
Use a Variant Constant for Data

Note: Must give states & variants a label so the label shows up with Unbundle By Name

Can use a For loop to sound 4 B's in a row.

Variant defines the length to play each note

When user hits the start button, the QSM plays a sequence of notes.


B A G A B B B A A A B D D
Ma - ry had a lit- tle lamb, lit- tle lamb, lit- tle lamb
B A G A B B B B A A B A G
Ma - ry had a lit- tle lamb. Its fleece was white as snow.