ECE2036: Lab 4 : State Machine Design with mBED

Instructions: This is related to your lab 4 – I would like for you to use a state machine design in the control of your system.

New C++ Element: Notice the “enumerated” types declared globally in the following code with keyword enum. This can be a great way to make your code more readable (i.e. self-documenting).

```
#include "mbed.h"
#include "Speaker.h"
#include "PinDetect.h"


//declare objects for pins used with pushbuttons
PinDetect pb1(p15);
PinDetect pb2(p16);
PinDetect pb3(p18);

//declare a speaker object
Speaker mySpeaker(p21);

enum InputType {FWD, BACK,STAY};
enum StateType {Q0, Q1, Q2, Q3};

InputType input = STAY;
StateType state = Q0;

// Callback routine is interrupt activated by a debounced pb3 hit
void pb3_hit_callback (void)
{
// ADD CODE HERE THAT YOU WHAT TO RUN WHEN INTERUPT IS GENERATED
input = STAY;
}

// Callback routine is interrupt activated by a debounced pb1 hit
void pb1_hit_callback (void)
{
// ADD CODE HERE THAT YOU WHAT TO RUN WHEN INTERUPT IS GENERATED
input = FWD;
}

// Callback routine is interrupt activated by a debounced pb2 hit
void pb2_hit_callback (void)
{
input = BACK;
}


int main() {


   pb1.mode(PullUp);
   pb2.mode(PullUp);
   pb3.mode(PullUp);

   // Delay for initial pullup to take effect
   wait(.01);

   // Setup Interrupt callback functions for a pb hit
   pb1.attach_deasserted(&pb1_hit_callback);
   pb2.attach_deasserted(&pb2_hit_callback);
   pb3.attach_deasserted(&pb3_hit_callback);
```

```
  // Start sampling pb inputs using interrupts
  pb1.setSampleFrequency();  //default is 20KHz sampling
  pb2.setSampleFrequency();
  pb3.setSampleFrequency();
  // pushbuttons now setup and running


  while(1) {

    switch(state)
    {

    case(Q0):
      //Produce output for this state
      mySpeaker.PlayNote(200.0,0.5,0.05);
      //calculate next state
      if (input == FWD)
        state = Q1;
      else if (input == BACK)
        state = Q3;
      else //input should be stay
        state = Q0;
      break;

    case (Q1):
     //Produce output for this state
      mySpeaker.PlayNote(300.0,0.5,0.05);
      //calculate next state
      if (input == FWD)
        state = Q2;
      else if (input == BACK)
        state = Q0;
      else //input should be stay
        state = Q1;

      break;

    case (Q2):
     //Produce output for this state
      mySpeaker.PlayNote(400.0,0.5,0.05);
      //calculate next state
      if (input == FWD)
        state = Q3;
      else if (input == BACK)
        state = Q1;
      else //input should be stay
        state = Q2;
      break;

    case (Q3):
     //Produce output for this state
      mySpeaker.PlayNote(500.0,0.5,0.05);
      //calculate next state
      if (input == FWD)
        state = Q0;
      else if (input == BACK)
        state = Q2;
      else //input should be stay
        state = Q3;
      break;

    }//end switch
    wait (0.5);
   }
}
```