

## Peisen Hu ECE 455 Lab1 Report

So basically, the FSM framework I created is a single function as shown below:

```
void FSMFramework (unsigned int numofstates, unsigned int (*states[numofstates])(void), unsigned int init){
    unsigned int next = init;
    while (1){
        next = (*states[next])();
    }
}
```

It will take in three parameters, which are:

1. **unsigned int numofstates**: the number of all possible states in your FSM;
2. **unsigned int (\*states[numofstates])(void)**: an array of function pointers, each of which points to a function you wrote that represents a state in your state machine. For instance, states[0] may point to the function representing state 0 in your FSM.
3. **unsigned int init**: the initial state of your FSM represented by its **index in the array above**. \*\*\*Note that in practice this may be **different** from the function's name. For example, if you write two states functions:

*unsigned int state0(void);*

*unsigned int state1(void);*

but then you assign:

*states[0] = state1;*

*states[1] = state0;*

and then call the FSM framework with:

*FSMFramework (2, states, 0);*

*//note the init here is 0, representing states[0], not state0()*

then your initial state (function) will be state1(), not state0()!

A bit further explanation about how the frame works:

It will first call the initial state's function and wait for it to exit and return the index for the next state. Then it will call the next state's function and wait for it to exit and return the index for the subsequent state, and then keep looping like this on and on.

And to get this framework run, you will need to follow the two steps below:

## I. Write the functions representing the states in your FSM

For example the one below, consisting of three parts: enter code, execute code (most stuff in the infinite loop), and exit code (the return-relevant statements). Also remember to return the index of the subsequent state (which is 2 in this case) upon exit:

```
unsigned int state1(void) //State for Furnace Off (On Furnace Cool-down)
{
    //Delay_SysTick (500);
    //printf("State1\r\n");
    //return 2;
    //Enter Code
    FurnaceTimer = 0;
    //Execute Code
    while(1){
        current_temp = (-102.4)+(0.05*ReadPotentiometer());
        joy = ReadJoystick();
        if (joy != 0 && joy_flag == 0){
            joy_flag = 1;
            if (joy == 16) { //NORTH
                setpoint += 0.5;
            } else if (joy == 64) { //SOUTH
                setpoint -= 0.5;
            }
        } else if (joy == 0 && joy_flag == 1){
            joy_flag = 0;
        }
        if (PrintTimer >= 1000){
            printf ("Set Point is %.4f Celsius\r\n", setpoint);
            printf ("Current Temp is %.4f Celsius\r\n", current_temp);
            PrintTimer = 0;
        }
        //Exit Code
        if(FurnaceTimer >= DEF_OFF*1000){
            return 2;
        }
    }
}
```

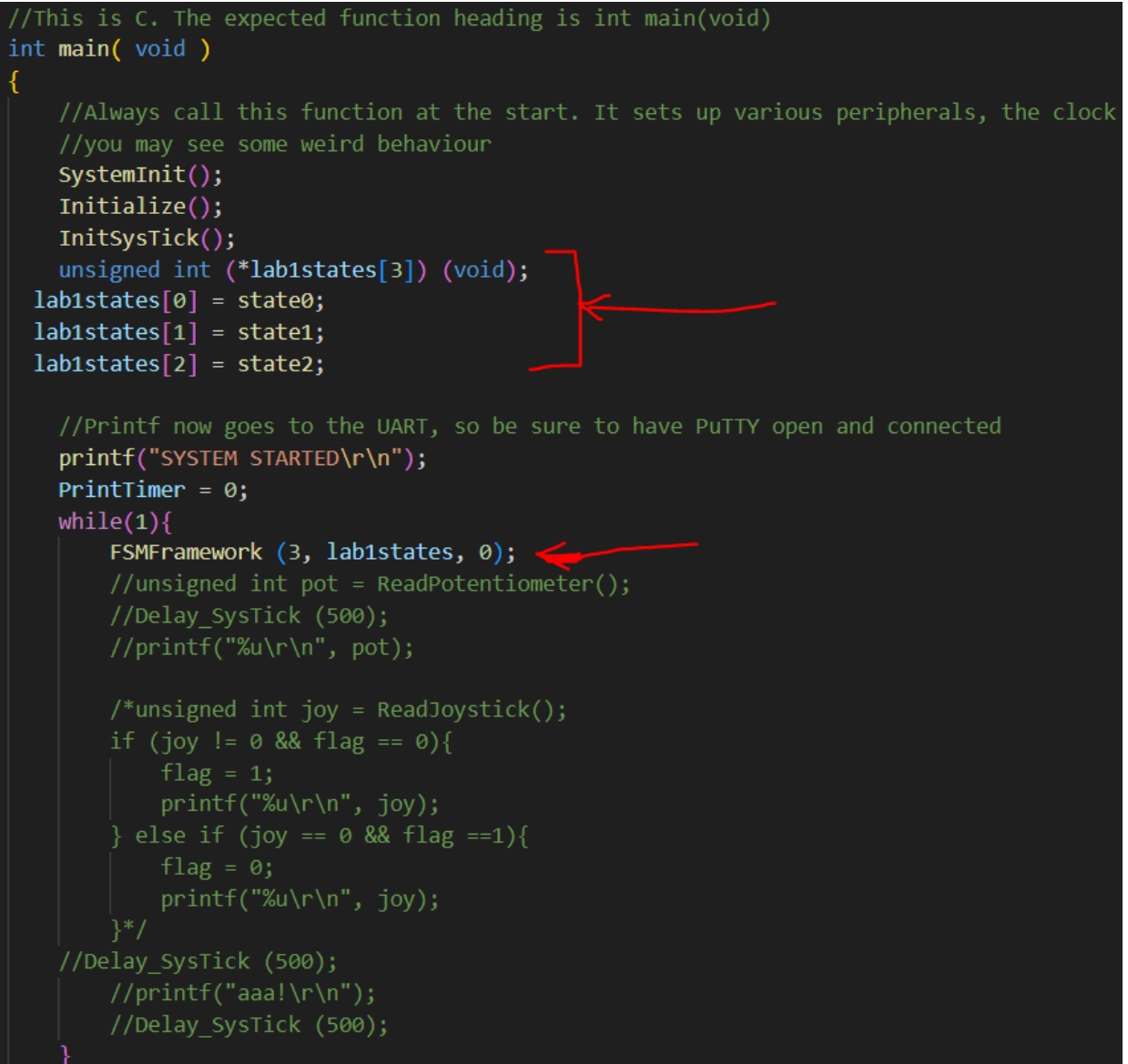
## II. Fill the array with function pointers to your functions and pass it to the FSM framework with the other parameters

This is quite straightforward. Remember to call the FSM framework in main() (or wherever you want). Below is a screenshot for your reference:

```
//This is C. The expected function heading is int main(void)
int main( void )
{
    //Always call this function at the start. It sets up various peripherals, the clock
    //you may see some weird behaviour
    SystemInit();
    Initialize();
    InitSysTick();
    unsigned int (*lab1states[3]) (void);
    lab1states[0] = state0;
    lab1states[1] = state1;
    lab1states[2] = state2;

    //Printf now goes to the UART, so be sure to have PuTTY open and connected
    printf("SYSTEM STARTED\r\n");
    PrintTimer = 0;
    while(1){
        FSMFramework (3, lab1states, 0);
        //unsigned int pot = ReadPotentiometer();
        //Delay_SysTick (500);
        //printf("%u\r\n", pot);

        /*unsigned int joy = ReadJoystick();
        if (joy != 0 && flag == 0){
            flag = 1;
            printf("%u\r\n", joy);
        } else if (joy == 0 && flag ==1){
            flag = 0;
            printf("%u\r\n", joy);
        }*/
        //Delay_SysTick (500);
        //printf("aaa!\r\n");
        //Delay_SysTick (500);
    }
}
```



And congratulations! Everything should be OK now. 😊