CS120B – Lecture Exercises 02-07-2022

Ivan Neto

Exercise 1

```
#include "RIMS.h"

volatile unsigned char TimerFlag = 0;

void TimerISR() { TimerFlag = 1; }

enum GP_STATES {...} GP_State;
void GP_Tick() {

    static unsigned char i, pulse; // First blank fill in

    ... // Standard switch statments for SM
}

enum RE_States {...} RE_State;
void RE_Tick() {

    static unsigned char echoes; // Second blank fill in

    ... // Standard switch statements for SM
}

void main() {
    B = 0;
    TimerSet(200);
    TimerOn();
    GP_State = GP_Start;
    RE_State = RE_Start;

    while (1) {
        GP_Tick(); // Third blank fill in
        RE_Tick(); // Fourth blank fill in
        while (!TimerFlag) {}
        TimerFlag = 0;
    }
}
```

Exercise 2

```
#include "RIMS.h"

volatile unsigned char TimerFlag = 0;

void TimerISR() { TimerFlag = 1; }

enum GP_States { ... } GP_State;
void GP_Tick() { ... }

enum RE_States { ... } RE_State;
void RE_Tick() { ... }

void main () {

    uint16_t period = 50;

    uint16_t GP_Elapsed = 200;
    uint16_t RE_Elapsed = 50;

    B = 0;
    TimerSet(50);
    TimerOn();
    GP_State = GP_Start;
    RE_State = RE_Start;

    while(1) {

        if (GP_Elapsed >= 200) {
            GP_Tick();
            GP_Elapsed = 0;
        }
        if (RE_Elapsed >= 50) {
            RE_Tick();
            RE_Elapsed = 0;
        }

        while(!TimerFlag) {}
        TimerFlag = 0;

        GP_Elapsed += period;
        RE_Elapsed += period;

    }
```

```
}
```

Exercise 3

```
#include "RIMS.h"

typedef struct task {
    int state;
    unsigned long period;
    unsigned long elapsedTime;
    int (*TickFunc)(int);
} task;

task tasks[taskNum];

const unsigned char tasksNum = 2;
const unsigned long tasksPeriodGCD = 50;
const unsigned long periodGeneratePulse = 200;
const unsigned long periodRecordEchoes = 50;

enum GP_States { ... } GP_State;
int GP_Tick(int state) { ... }

enum RE_States { ... } RE_State;
int RE_Tick(int state) { ... }

void TimerISR() {
    unsigned char i;

    for (i = 0; i < tasksNum; ++i) {

        if (tasks[i].elapsedTime >= tasks[i].period) {
            tasks[i].state = tasks[i].TickFunc(tasks[i].state);
            tasks[i].elapsedTime = 0;
        }

        tasks[i].elapsedTime += tasksPeriodGCD;
    }
}

int main() {
    unsigned char i = 0;

    tasks[i].state = GP_Start;
    tasks[i].period = periodGeneratePulse;
    tasks[i].elapsedTime = tasks[i].period;
    tasks[i].TickFunc = &GP_Tick;
```

```
    i++;
    tasks[i].state = RE_Start;
    tasks[i].period = periodRecordEchoes;
    tasks[i].elapsedTime = tasks[i].period;
    tasks[i].TickFunc = &RE_Tick;

    TimerSet(tasksPeriodGCD);
    TimerOn();

    while (1) { Sleep(); }
    return 0;
}
```