```c
/**
  ******************************************************************************
  * @file    main.c
  * @author  Harsh Savla & TJ Wiegman
  * @version V1.1.0
  * @date    2022-10-24
  * @brief   Main program body for ME586 Homework 7
  ******************************************************************************
  */

/* Includes ------------------------------------------------------------------*/
#include "ME586.h"

/* Private typedef -----------------------------------------------------------*/
/* Private define ------------------------------------------------------------*/
#define OUTPUT_MAX 10
#define OUTPUT_MIN -10
/* Private macro -------------------------------------------------------------*/
/* Private variables ---------------------------------------------------------*/
int* ram_ptr;       // address of storage start
int* adc_ptr;       // address of last stored value
int ram_allocated;  // amount of RAM that has actually been allocated
int ram_size = 0;
float time_period = 100;  // in milliseconds
float Kp = 1;
float Ki = 0;
float Kd = 0;
float setpoint = 0;
float error_prior = 0;
float integral = 0;
/* Private function prototypes -----------------------------------------------*/
/* Private functions ---------------------------------------------------------*/


int main(void) {
    // Setup
    char input_choice;
    char input_mode = 1;
    int stored_actual = 0;
    disable_timer_interrupt();  // start without interrupts
    initcom();  // initialize serial
    initadc();  // initialize analog in
    initdac();  // initialize analog out

    // User input
    while (input_mode == 1) {
        printf("\nChange parameters?\n");
        printf(" [S] Provide a setpoint\n");
        printf(" [G] Input new gains\n");
        printf(" [T] Set the sampling period\n");
        printf(" [P] Change RAM storage size\n");
        printf(" [R] Run the controller\n");
        printf(" [D] Output stored values\n");
        printf(" [X] Exit\n");
```

```
55          printf("Input: ");
56          WaitForKeypress();
57          input_choice = getchar();
58          printf("\n\n")
59
60          switch (input_choice) {
61              case 'S' : // provide a setpoint
62                  printf("Setpoint: ");
63                  setpoint = getfloat();
64                  printf("\n");
65              break;
66
67              case 'G' :  // change gains
68                  printf("Kp: ");
69                  Kp = getfloat();
70                  printf("\nKi: ");
71                  Ki = getfloat();
72                  printf("\nKd: ");
73                  Kd = getfloat();
74                  printf("\n");
75              break;
76
77              case 'T' :  // timer period
78                  printf("Time period (ms): ");
79                  time_period = getfloat();
80                  printf("\n");
81              break;
82
83              case 'P' : // change RAM storage size
84                  printf("Number of values to store: ");
85                  ram_size = getnum();
86                  printf("\n");
87              break;
88
89              case 'R' : // run the controller
90                  // Allocate RAM storage size
91                  if (ram_ptr == NULL) {
92                      ram_ptr = (int*) calloc(ram_size * sizeof(int));
93                  } else {
94                      free(ram_ptr);
95                      ram_ptr = (int*) calloc(ram_size * sizeof(int));
96                  }
97                  ram_allocated = ram_size;
98                  adc_ptr = ram_ptr;
99
100                 // Report allocated RAM to user
101                 if (ram_size == 0) {
102                     printf("Not recording measured values.\n")
103                 } else if (ram_ptr == NULL) {
104                     printf("Error! Failed to allocate memory for measured values.\n")
105                 } else {
106                     printf("Recording first ");
107                     shownum(ram_allocated);
108                     printf(" measured values.\n");
109                 }
```

```
110
111                 // Start running controller
112                 printf("Running...\n");
113                 inittime(time_period);
114                 restore_timer_interrupt();
115             break;
116
117             case 'D' : // output data from RAM
118                 disable_timer_interrupt(); // don't want data to change mid-readout
119                 stored_actual = (adc_ptr - ram_ptr) / sizeof(int);
120                 printf("First ");
121                 shownum(stored_actual);
122                 printf(" measured values:\n");
123                 for (int* out_ptr = ram_ptr; i <= adc_ptr; out_ptr += sizeof(int)) {
124                     // start at ram_ptr, count up to adc_ptr, in increments of sizeof(int)
125                     shownum(*out_ptr);
126                     printf("\n"); // each number gets a new line
127                 }
128                 restore_timer_interrupt();
129             break;
130
131             case 'X' : // exit
132                 input_mode = 0;
133                 disable_timer_interrupt();
134                 printf("Quitting...\n");
135             break;
136
137             default:
138                 printf("Input not recognized.\n")
139             break;
140         }
141     }
142     while (1) {};
143 } //end of main program
144
145
146 void timehand(void) {
147     // Setup
148     int measured_ADC;
149     float error;
150     float derivative;
151     float output;
152
153     // PID Controller
154     measured_ADC = a_to_d(0);
155     error = setpoint - measured_ADC;
156     integral = integral + (error * time_period);
157     derivative = (error - error_prior) / time_period;
158     output = (Kp * error) + (Ki * integral) + (Kd * derivative);
159
160     if (output > OUTPUT_MAX) output = OUTPUT_MAX;
161     if (output < OUTPUT_MIN) output = OUTPUT_MIN;
162     error_prior = error;
163     d_to_a(0, output);
164
```

```
    // Store PID values
    if (adc_ptr < (ram_ptr + (ram_allocated * sizeof(int))) {
        *adc_ptr = measured_ADC;
        adc_ptr = adc_ptr + sizeof(int);
    }
}

void inthand(void){
}

/**********************END OF FILE****/

```