SystemOuts.C

```
// File deals with the controller selection and desired transient
charicteristics
#include <stdio.h>  // Including stdio.h header file
#include <math.h>   // Including math.h header file
#include <tgmath.h> // Including tgmath.h header file

void controllerSelection(struct controllers *c2, int i); // Declaration
of the function to select which controllers to design
float outputParams(struct parameters *p2, int i);        // Declaration
of the desired transient parameters function
void buffer(void);                                       // Declares the
function to clear the input buffer

struct controllers // Defining the contollers structure
{
    int PID;  // Value states if there's going to be a PID controller
    int PI;   // Value states if there's going to be a PI controller
    int PD;   // Value states if there's going to be a PD controller
    int Lead; // Value states if there's going to be a Phase Lead
Compensator
    int Lag;  // Value states if there's going to be a Phase Lag
Compensator
};

struct parameters // Defining the parameters structure
{
    float ss_gain;              // Defining the steady state gain of the
compensated system
    float settling_time;        // Defining the compensated systems'
settling time
    float percentage_overshoot; // Defining the percentage overshoot for
the compensated system
    float stepVal;              // Defining the step value for the
compensated system
};

void controllerSelection(struct controllers *c2, int i) // Deining the
controller selection function
{
    printf("You can choose up to five contoller types to design,
PID/PI/PD/Phase lead/Phase Lag.\n"); // gives info to the user
    printf("Would you like to design a PID controller? (Yes = 1, no =
0)\n");                          // asks the user if they want a PID
controller
    do
// loops at least once
    {
        if (scanf("%d", &c2[i].PID) != 1 || (c2[i].PID != 1 && c2[i].PID
!= 0)) // prompts user and runs if answer not in range
        {
            printf("Invalid input, please enter a valid answer.\n"); //
error message
        }
        buffer(); // clears the buffer
    } while (c2[i].PID != 1 && c2[i].PID != 0); // loops through until a
valid answer is entered
```

```
    printf("Would you like to design a PI controller? (Yes = 1, no =
0)\n"); // asks the user if they want a PI controller
    do
    {
        if (scanf("%d", &c2[i].PI) != 1 || (c2[i].PI != 1 && c2[i].PI !=
0)) // prompts user and runs if answer not in range
        {
            printf("Invalid input, please enter a valid answer.\n"); //
error message
        };
        buffer(); // clears the buffer
    } while (c2[i].PI != 1 && c2[i].PI != 0); // loops through until a
valid answer is entered
    printf("Would you like to design a PD controller? (Yes = 1, no =
0)\n"); // asks the user if they want a PD controller
    do
    {
        if (scanf("%d", &c2[i].PD) != 1 || (c2[i].PD != 1 && c2[i].PD !=
0)) // prompts user and runs if answer not in range
        {
            printf("Invalid input, please enter a valid answer.\n"); //
error message
        };
        buffer(); // clears the buffer
    } while (c2[i].PD != 1 && c2[i].PD != 0); // loops through until a
valid answer is entered
    printf("Would you like to design a Phase lead compensator? (Yes = 1,
no = 0)\n"); // asks the user if they want a Phase lead compensator
    do
    {
        if (scanf("%d", &c2[i].Lead) != 1 || (c2[i].Lead != 1 &&
c2[i].Lead != 0)) // prompts user and runs if answer not in range
        {
            printf("Invalid input, please enter a valid answer.\n"); //
error message
        };
        buffer(); // clears the buffer
    } while (c2[i].Lead != 1 && c2[i].Lead != 0); // loops through until
a valid answer is entered
    printf("Would you like to design a Phase lag compensator? (Yes = 1,
no = 0)\n"); // asks the user if they want a Phase lag compensator
    do
    {
        if (scanf("%d", &c2[i].Lag) != 1 || (c2[i].Lag != 1 && c2[i].Lag
!= 0)) // prompts user and runs if answer not in range
        {
            printf("Invalid input, please enter a valid answer.\n"); //
error message
        };
        buffer(); // clears the buffer
    } while (c2[i].Lag != 1 && c2[i].Lag != 0); // loops through until a
valid answer is entered
};

float outputParams(struct parameters *p2, int i) // defining the output
parameters function
{
    do // Loops through at least once, as the do while is initiated
```

```
    {
        printf("Please enter the value of the step input:\t");
// Asks the user to enter the value of their step input
        if (scanf("%f", &p2[i].stepVal) != 1 || (p2[i].stepVal < -10000
|| p2[i].stepVal > 10000)) // Prompts the user for an answer and checks
the answer is numerical
        {
            printf("Invalid input, please enter a number between -10000
and 10000\n"); // Error message thrown if the user enters an invalid
input
        };
        buffer(); // Clears the buffer
    } while (p2[i].stepVal < -10000 || p2[i].stepVal > 10000); // Keeps
looping through
    do // Loops through at least once, as the do while is initiated
    {
        printf("Please enter the desired steady state gain of the
system:\n");                    // Asks the user to enter the desired
steady state gain of the system
        if (scanf("%f", &p2[i].ss_gain) != 1 || (p2[i].ss_gain > 10000 ||
p2[i].ss_gain < -10000)) // Prompts the user for an answer and checks the
answer is numerical
        {
            printf("Invalid input, please enter a number between -10000
and 10000\n"); // Error message thrown if the user enters an invalid
input
        };
        buffer(); // Clears the buffer
    } while (p2[i].ss_gain > 10000 || p2[i].ss_gain < -10000); // Keeps
looping through until a valid answer is entered
    do // Loops through at least once, as the do while is initiated
    {
        printf("Please enter the desired settling time of the system (in
seconds):\n");                              // Asks the user to enter the
desired settlement time of the system in seconds
        if (scanf("%f", &p2[i].settling_time) != 1 ||
(p2[i].settling_time > 10000 || p2[i].settling_time < -10000)) // Prompts
the user for an answer and checks the answer is numerical
        {
            printf("Invalid input, please enter a number between -10000
and 10000\n"); // Error message thrown if the user enters an invalid
input
        };
        buffer(); // Clears the buffer
    } while (p2[i].settling_time > 10000 || p2[i].settling_time < -
10000); // Keeps looping through until a valid answer is entered
    do // Loops through at least once, as the do while is initiated
    {
        printf("Please enter the desired percent overshoot of the system
(As a decimal value between 0 and 1):\n");             // Asks the user
to enter the desired percent overshoot of the system as a decimal value
        if (scanf("%f", &p2[i].percentage_overshoot) != 1 ||
(p2[i].percentage_overshoot < 0 || p2[i].percentage_overshoot > 1)) //
Prompts the user for an answer and checks the answer is numerical
        {
            printf("Invalid input, please enter a number between 0.0 and
1.0\n"); // Error message thrown if the user enters an invalid input
        };
```

```
        buffer(); // Clears the buffer
    } while (p2[i].percentage_overshoot < 0 || p2[i].percentage_overshoot
> 1); // Keeps looping through until a valid answer is entered
    return p2[i].stepVal; // returns the step value for the system
};

void buffer(void) // Function definition for the buffer function
{
    int c = 0; // Defining the input variable used to clear the buffer
    do         // Runs at least once
    {
        c = getchar(); // Gets the next character in the buffer
    } while (c != '\n'); // loops through until it reaches a new line
}; // End of function definition for clearing
```