Displays.C

```
// File deals with the display of the created controllers
#include "CalcsCt.C" // Including the CalcsCt.C file

void controller_display(struct systemOuts *s2, struct controllers *c2,
struct parameters *p2, int i); // Declaration of the controller_display
function

void controller_display(struct systemOuts *s2, struct controllers *c2,
struct parameters *p2, int i) // defining the controller_display function
{
    printf("The system has the following transient parameters, as
specified.\n");                                                    //
tells the user what they're viewing
    printf("A natural frequency of %2.0f rad/s\n", s2[i].naturalFreq);
// the natural frequency
    printf("A damping ratio of %f\n", s2[i].damping);
// the damping ratio
    printf("A settling time of %3.0f seconds\n", p2[i].settling_time);
// the settling time
    printf("A percentage overshoot of %2.0f percent\n",
(p2[i].percentage_overshoot) * 100);
// the percentage overshoot
    printf("For a step input of %3.0f, as specified, the system has a
steady state value of %2.0f \n", p2[i].stepVal, p2[i].ss_gain); // the
steady state vlaue
    printf("Controllers designed, as specified.\n");
    if (c2[i].PID == 1) // Runs if a PID controller is selected
    {
        printf("\nContoller type: PID\n");
// details that it's a PID controller
        printf("%f (s + %f)(s + 0.1)\n", s2[i].a.kp, (s2[i].a.kd) * -1);
// Sets the controllers transfer function's numerator
        printf("-----------------------------\n");
// Adds a division line
        printf("            %1.0f * s\n", s2[i].a.kid);
// sets the denominator value
    };
    if (c2[i].PI == 1) // Runs if a PI controller is selected
    {
        printf("\nController type: PI\n");        // Tells the user the
type of controller they're viewing
        printf("%f (s + 0.1)\n", s2[i].b.kp);     // Sets the controllers
transfer function's numerator
        printf("-----------------\n");            // Adds a division line
        printf("      %1.0f * s\n", s2[i].b.kid); // sets the denominator
value
    };
    if (c2[i].PD == 1) // Runs if a PD controller is selected
    {
        printf("\nController type: PD\n");                     // Tells
the user the type of controller they're viewing
        printf("%f (s + %f)\n", s2[i].c.kp, (s2[i].c.Kd) * -1); // Sets
the controllers transfer function's numerator
    };
    if (c2[i].Lead == 1) // Runs if a phase lead compensator is selected
    {
```

```
        printf("\nController type: Phase Lead Compensator\n"); // Tells
the user the type of controller they're viewing
        printf("%f (s + %f)\n", s2[i].d.K, s2[i].d.r);         // Sets
the controllers transfer function's numerator
        printf("------------------------\n");                  // Adds a
division line
        printf("     (s + %f)\n", s2[i].d.ra);                 // sets
the denominator value
    };
    if (c2[i].Lag == 1) // Runs if a phase lag compensator is selected
    {
        printf("\nController type: Phase Lag Compensator\n"); // Tells
the user the type of controller they're viewing
        printf("%f (s + %f)\n", s2[i].d.K, s2[i].d.r);        // Sets the
controllers transfer function's numerator
        printf("------------------------\n");                  // Adds a
division line
        printf("     (s + %f)\n", s2[i].d.ra);                 // sets the
denominator value
    };
};
```