File needed

```
#include <c8051_SDCC.h>

#include <stdlib.h>// needed for abs function

#include <stdio.h>

#include <i2c.h>
```

8051 initialize functions

```
void Port_Init(void);

void PCA_Init (void);

void SMB_Init (void);

void Interrupt_Init(void); void PCA_ISR ( void ) __interrupt 9;

void read_accel (void); //Sets global variables gx & gy

void set_servo_PWM (void);

void set_drive_PWM(void);

void updateLCD(void);

void set_gains(void); // function which allow operator to set feedback gains


//define global variables

unsigned int PW_CENTER = ____;

unsigned int PW_RIGHT = ____;

unsigned int PW_LEFT = ____;

unsigned int SERVO_PW = ____;

unsigned int SERVO_MAX= _____;

unsigned int SERVO_MIN= _____;

unsigned int  heading;

unsigned int  range;

unsigned int  light;

int compass_adj = 0;          // correction value from compass
int range_adj = 0;           // correction value from ranger
unsigned char r_count;        // overflow count for range
unsigned char h_count;        // overflow count for heading
unsigned char print_count;     // overflow count for printing


__sbit __at ____ RUN // a slide switch
__sbit__at ____ BILED0
__sbit__at ____ BILED1
Main function
Declare local variables
    None
Funcion initialization
Do infinite while loop
    Print battery voltage for check
    if run out of battery
        charge the battery
    else
        if (run switch is off)
            Set the motor stop
            Set the steer parallel to the car
            BILED is red
        Else if (run switch is on)
            Set gain first (only once)
            If (enough overflows to update accel)
                read_accels();
                set_servo_PWM(); // set the servo PWM
                set_drive_PWM(); // set drive PWM
                new_accels = 0; //set the flag off
                a_count = 0; //clear the accel counts
                if (enough overflows to update LCD)
                    updateLCD(); // display values
                    new_lcd = 0;
                    lcd_count = 0;
        finish the loop
end main function


void PCA_ISR ( void ) __interrupt 9 {
        if (CF) {
                CF = 0; // clear overflow indicator
                a_count++;
                if(a_count>=____) {
                        new_accel=1;
                        a_count = 0;
                }
                lcd_count++;
                if (lcd_count>=____) {
                        new_lcd = 1;
                        lcd_count = 0;
                }
                PCA0 = PCA_start;
        } // handle other PCA interrupt sources
        PCA0CN &= 0xC0;
}
```