```
//
// @file    test_servo.c
//
//
// Will prove we can talk to I2C devices
//
// Need access to std i/o routines
//

#include <stdio.h>
#include <robotcontrol.h>
#include "servo_driver.h"

// Simple main program which gets a reading from the SRF02

int main(void) {
   int i ;

// Check for existing processes

    if(rc_kill_existing_process(2.0)<-2) return -1;

// Create a lock file

    rc_make_pid_file();

// Start signal handler so we can exit cleanly

        if(rc_enable_signal_handler()==-1){
                fprintf(stderr,"ERROR: failed to start signal handler\n");
                return -1;
        }

// Initialize the i2c bus

    rc_i2c_init(I2C_BUS, SERVO_I2C_ADDR) ;

// Turn on the servo power rail

   rc_servo_init() ;
   rc_servo_power_rail_en(1) ;

// Reset the servo driver

    resetServoDriver() ;

// Set rep rate to 50 Hz

    setServoFREQ(50.0) ;
```

```
// Turn servo one direction and then back the other

    for (i=150; i<=350; i+=50) {
       setServoPW(0, i) ;
       rc_usleep(900000) ;
       rc_usleep(900000) ;
     }

   resetServoDriver() ;

// Cleanup

    rc_i2c_close(I2C_BUS) ;

// Turn off the servo power rail

   rc_servo_cleanup() ;


// Remove the lock file

   rc_remove_pid_file();

   return(0);
}
```

```
// Need access to std i/o routines

#include <stdio.h>

// Need access to i2c library funcitions

#include <robotcontrol.h>

// Need access to servo_driver.h

#include "servo_driver.h"

// Routine to set the PWM freqeuncy
// Routine accepts a freqeuncy between 40 Hz and 1000 Hz
// and sets the pre-scaler value in the PCA9685 servo controller

#define   DEBUG    0

unsigned int resetServoDriver(void) {

// Set the I2C device address

   rc_i2c_set_device_address(I2C_BUS, SERVO_I2C_ADDR) ;

// Write to MODE1 register

   rc_i2c_write_byte(I2C_BUS, PCA9685_MODE1, 0x00) ;

   return(TRUE);
}

// ****************************************************************************

unsigned int setServoFREQ(double freq) {
   double         tmp ;
   uint8_t        pre_scale_value ;
   uint8_t        old_mode ;
   uint8_t        new_mode ;

// Set up a read buffer

   unsigned char rd_buf[BUF_SIZE] ;

// Make sure frequency is in the allowed range (40 Hz to 1 kHz)
// If not, then return an error

   if ((freq < 40.0) || (freq > 1000)) return(FALSE) ;

// Convert freqeuncy to a pre-scaler value
```

```
   tmp = (25.0e6 / (4096.0 * freq)) + 0.5 ;
   pre_scale_value = ((unsigned char) tmp)  - 1 ;
   if (DEBUG) printf("Pre-scaler value: %d\n", (int) pre_scale_value) ;

// Select the device we want to talk to

   rc_i2c_set_device_address(I2C_BUS, SERVO_I2C_ADDR) ;

// Need to read current mode

   rc_i2c_read_byte(I2C_BUS, PCA9685_MODE1, rd_buf) ;
   old_mode = rd_buf[0] ;
   new_mode = (old_mode & 0x7f) | 0x10 ; // sleep mode

// Write out new mode and put PCA9685 to sleep

   rc_i2c_write_byte(I2C_BUS, PCA9685_MODE1, new_mode) ;

//  Write out the prescaler value

   rc_i2c_write_byte(I2C_BUS, PCA9685_PRESCALE, pre_scale_value) ;

// Restore old mode setting

   rc_i2c_write_byte(I2C_BUS, PCA9685_MODE1, old_mode) ;

// Wait 5 ms

   rc_usleep(5000) ;

// Turn on auto increment

   rc_i2c_write_byte(I2C_BUS, PCA9685_MODE1, (old_mode | 0xa1) ) ;

   return(TRUE);
}

// ****************************************************************
// $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
// ****************************************************************

// Routine to set the pulse width in ticks
// 0.5 ms to 2ms is 100 to 400

unsigned int setServoPW(int chan, int pw) {
   uint8_t   ms_byte, ls_byte ;
   uint8_t   cr_addr ;

// Set up a write buffer
```

```
   uint8_t  wr_buf[BUF_SIZE] ;

// Make sure we have a valid channel number

   if ((chan < 0) || (chan > 15)) return(FALSE) ;

   if (DEBUG) printf("Channel #:  %d\n", chan) ;

// Select the device we want to talk to

   rc_i2c_set_device_address(I2C_BUS, SERVO_I2C_ADDR) ;

// Control register base address for a particular servo
// can be computed by taking 4 * channel number + 6

   cr_addr = (unsigned char) (4 * chan + SERVO_0_ON_L);
   if (DEBUG) printf("Control register address is %d\n", (int) cr_addr) ;

// Need to break the pulse with up into two bytes

   ms_byte = (unsigned char) (pw / 256) ;
   ls_byte = (unsigned char) (pw % 256 );
   if (DEBUG) printf("ms_byte = %d, ls_byte = %d\n", (int) ms_byte, (int) ls_byte) ;

// Base address for control register

   wr_buf[0] = cr_addr ;

// Pulse turned on when counter equals $000

   wr_buf[1] = 0x00 ;
   wr_buf[2] = 0x00 ;

// Pulse turned off when counter equals ...

   wr_buf[3] = ls_byte ;
   wr_buf[4] = ms_byte ;

   rc_i2c_write_bytes(I2C_BUS, wr_buf[0], 4, &wr_buf[1]) ;

   return(TRUE);
}
```

```
#define         I2C_BUS                 1

// Base address for the servo driver module

#define         SERVO_I2C_ADDR          0x40

// Write buffer size

#define         BUF_SIZE                12

// PCA9685 registers

#define         PCA9685_SUBADR1         0x02
#define     PCA9685_SUBADR2     0x03
#define     PCA9685_SUBADR3     0x04
#define         PCA9685_MODE1           0x00
#define         PCA9685_MODE2           0x01
#define         PCA9685_PRESCALE        0xfe

#define         SERVO_0_ON_L            0x06
#define         SERVO_0_ON_H            0x07
#define         SERVO_0_OFF_L           0x08
#define         SERVO_0_OFF_H           0x09

#define         ALL_SERVO_ON_L          0xfa
#define         ALL_SERVO_ON_H          0xfb
#define         ALL_SERVO_OFF_L         0xfc
#define         ALL_SERVO_OFF_H         0xfd

// Useful defines

#define         TRUE    1
#define         FALSE   0

//
// Function declaration

// Set the servo pulse repetition rate

unsigned int setServoFREQ(double) ;

// Set the pulse width of one of the servo channels

unsigned int setServoPW(int, int) ;

unsigned int resetServoDriver(void) ;
```