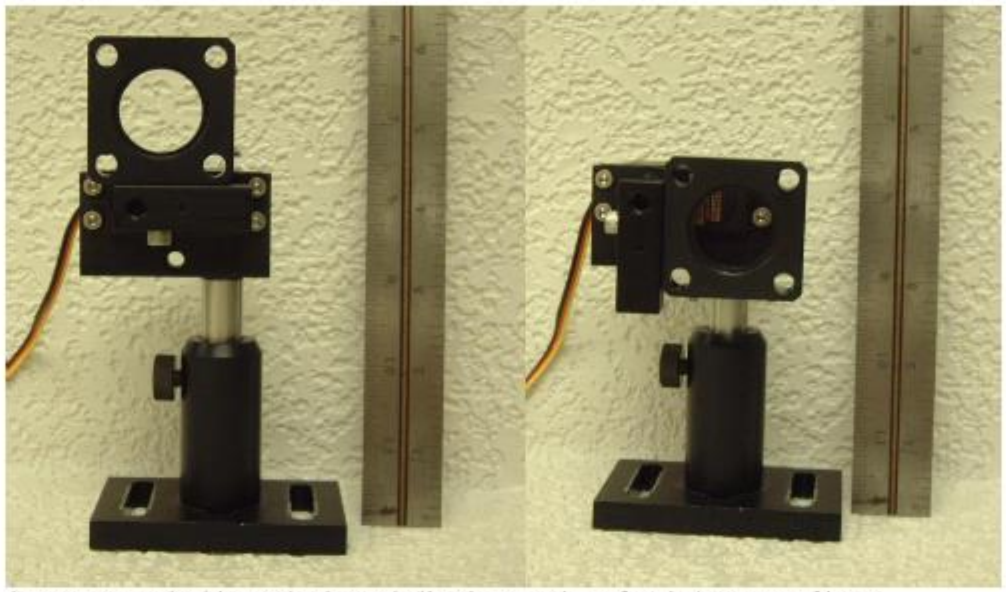
**Background**:

There is often a need to move laser optics in and out of the beam path. In the past we would use pneumatic cylinders to move the devices, but this was not precise and had a huge overhead of air supply, solenoids, guides, etc. In 2002 I came up with the idea of using a high quality model airplane servo to move the optics devices. The servos that I prefer to use are digital metal geared servos (Hitec HS-5625MG). The digital aspect allows the servo to be custom programmed for end positions, travel speed, deadband, and direction. The metal gears ensure a long and reliable life for this application. If you are interested, the full invention disclosure for the universal optics servo in the attached CDROM and also on the jlab group directory Jlabgrp/inj_group/hansknecht/motion/RC servos/servoboard.


Servo mounted with arm horizontal allowing rotation of optic in or out of beam

**Driver Board Requirements:**

The driver board has been designed to drive 4 digital servos simultaneously. It is powered by a single CAT5 standard power input. It has a connector for EPICS or other control system interface to issue position commands and obtain position read back. It has phone jack (RJ-11) connectors for the servos which have an adapter cable that mates with the standard model servo. The driver board provides 5 volts DC, Ground, and a pulse width modulated (PWM) signal to each servo. Servos work to position the output hub to a position based on the PWM signal received. The industry standard range is 900usec to 2100usec pulses at a repetition rate of 60Hz. If a 900usec pulse is received, the hub will be taken to one end limit. If 2100usec is received, the hub will travel to the other end limit. Pulse lengths between these two limits will position the servo hub to a relative intermediate position. My design goal for this board was to accept a bistable input (24VDC or 0VDC) from the control system and then position each servo to an end stop position based on the input state. The board will also provide a bistable feedback (12VDC or 0VDC) to provide the control system with a positive indication that the servo drive board has responded to the command and positioned the servo accordingly. A bistable control is sufficient for most of our optics because we usually want them completely in the beam path or completely out. Another feature of the driver is a movement with a deceleration near the end of each movement. This feature helps ensure that the optics arrive at their final resting position in a more controlled fashion.

**Schematic overview:**

J5 is the Standard Cat5 power input. The power input must be referenced to a ground that is common with the command and readback voltages of J6.

J6 is the Command I/O connection. Since we are driving and reading the position of 4 servos, there was no room on this connector for a reference ground. This is why the CAT5 power input must have a ground that is common with these J6 connections. The servo readbacks will provide 15V or 0V based on the position requested. The commands require at least 10V to actuate to the opposite state. The input commands pass through a voltage divider to limit the input voltage applied to the PIC microprocessor. The output readbacks come from the PIC via a UDN2981A driver that will pass or block 15VDC.

J1,J2,J3,J4 are the four servo connectors.

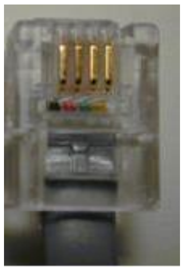
TP1,2,3,4 are currently spare I/O connections reserved for future use.

MCU1 is a PIC16F628 or PIC16F628A microcontroller. It handles all position requests and provides the PWM signals and TTL logic signals for report back to the control system.

**NOTE:** All servos should be programmed for a speed setting of 3 or lower with the Hitec Servo programmer. This will prevent a large current surge upon power up and actuation.

**Cabling issues:**

The 4 servo cables when viewed from the contact end should be crimped with black, red, green, yellow from left to right as shown below. This will result in the black wire as ground, the red wire as 5VDC, and the yellow wire as the PWM signal. This color code then matches the servo connector directly. Green is unused.



The CAT5 standard power is a standard that I have come up with for my own electronics in the Polarized Source. The pinout is as follows:

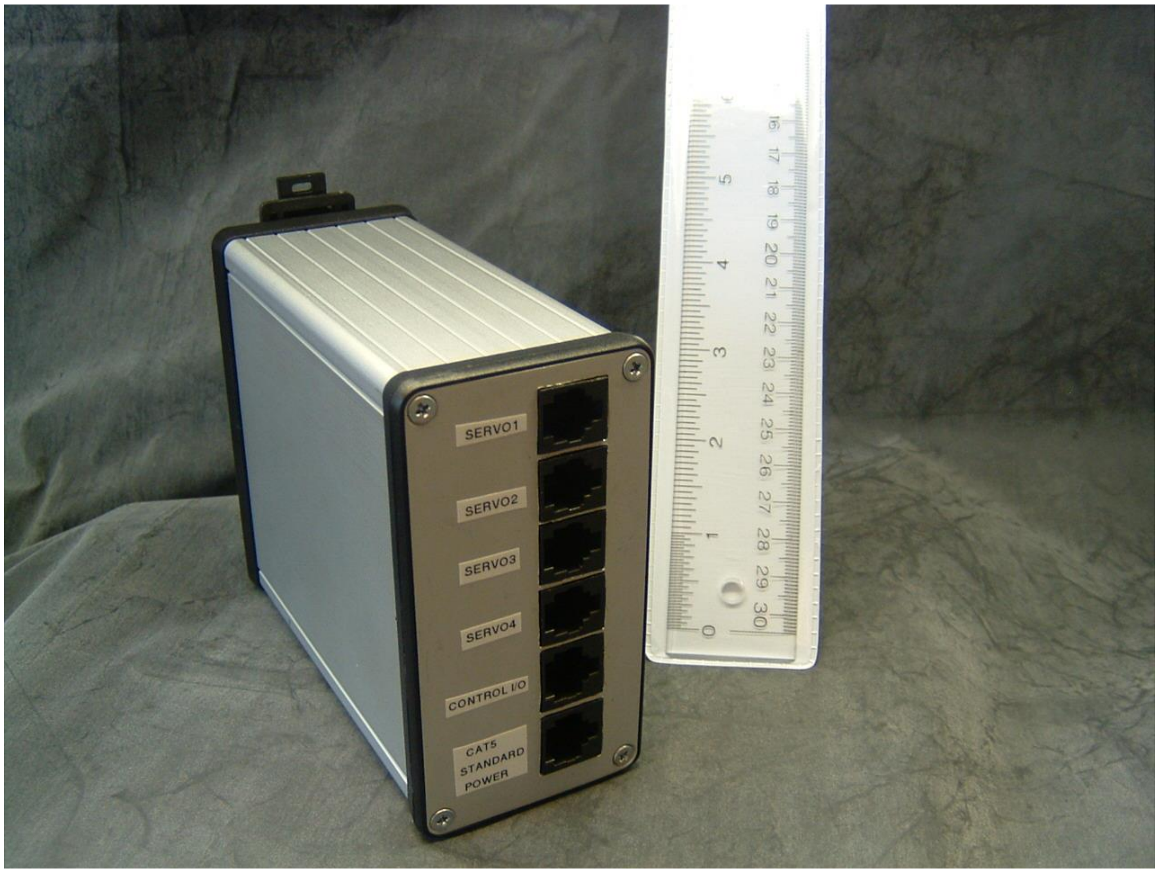
RJ-45 connection wired to the normal CAT5 color code:

| | |
|---|---|
| 1- Wht/Org | + 80Vdc |
| 2- Org | +5Vdc |
| 3- Wht/Grn | -15 Vdc |
| 4- Blu | Serial TX |
| 5- Wht/Blu | Serial RX |
| 6- Grn | +15 Vdc |
| 7- Wht/Brn | Gnd |
| 8- Brn | Gnd |

Of course in this servo driver we are only using pins 2,6,7,8. for +5, +15, and Gnd.

J6  Command and readback connector pinout (RJ45)

| | |
|---|---|
| 1- Wht/Org | Servo1 readback 15V or 0V provided to indicate current position |
| 2- Org | Servo2 readback |
| 3- Wht/Grn | Servo3 readback |
| 4- Blu | Servo4 readback |
| 5- Wht/Blu | Servo1 command  +24V or 0V to move servo |
| 6- Grn | Servo2 command |
| 7- Wht/Brn | Servo3 command |
| 8- Brn | Servo4 command |



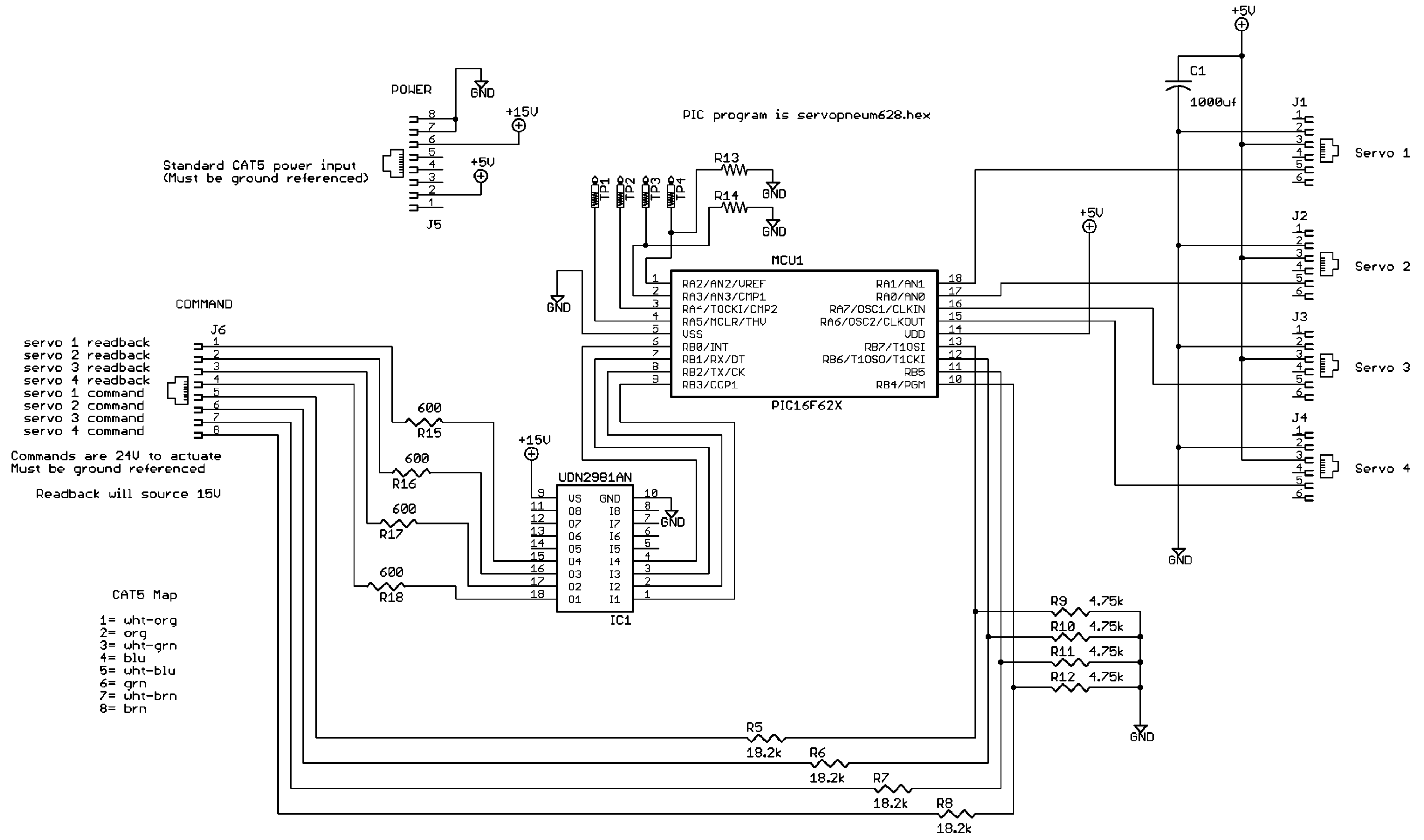
**Software:**
The program that resides in the PIC is called Servopneum628.hex and is compiled from the program servopneum628.c using the PCWH compiler (PIC C compiler).

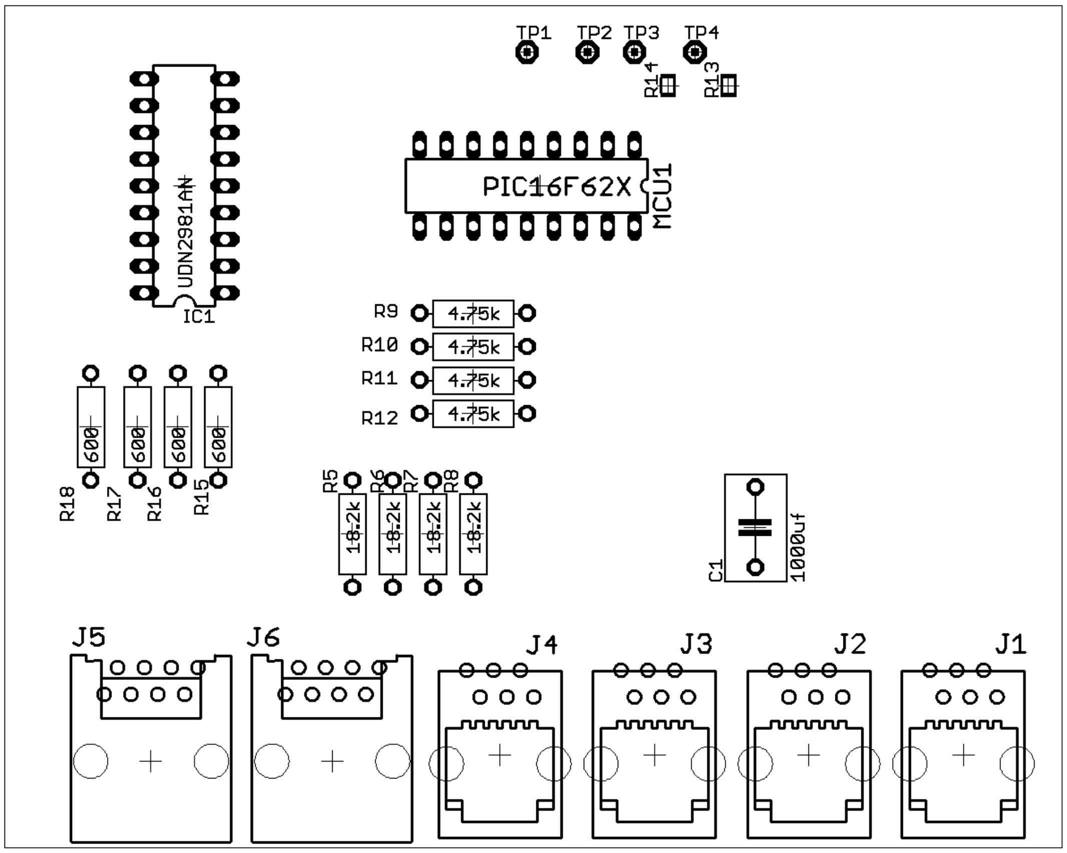
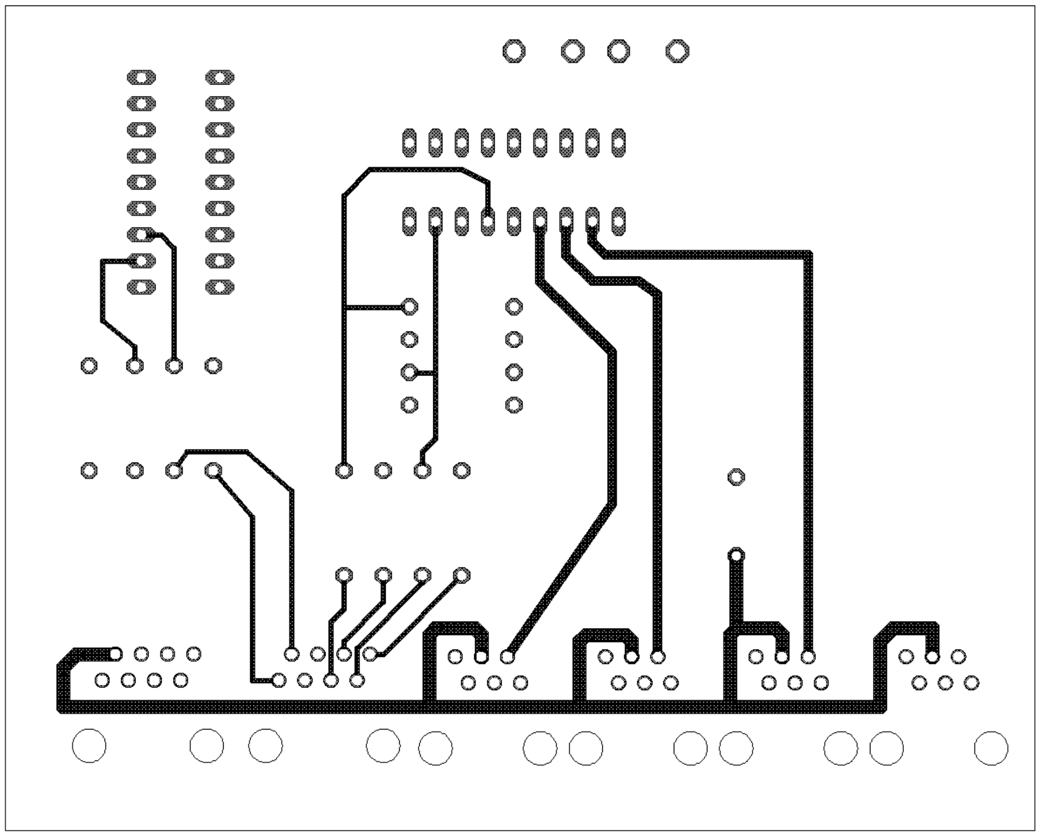
All files are stored on the accompanying CDROM and in the following directory:

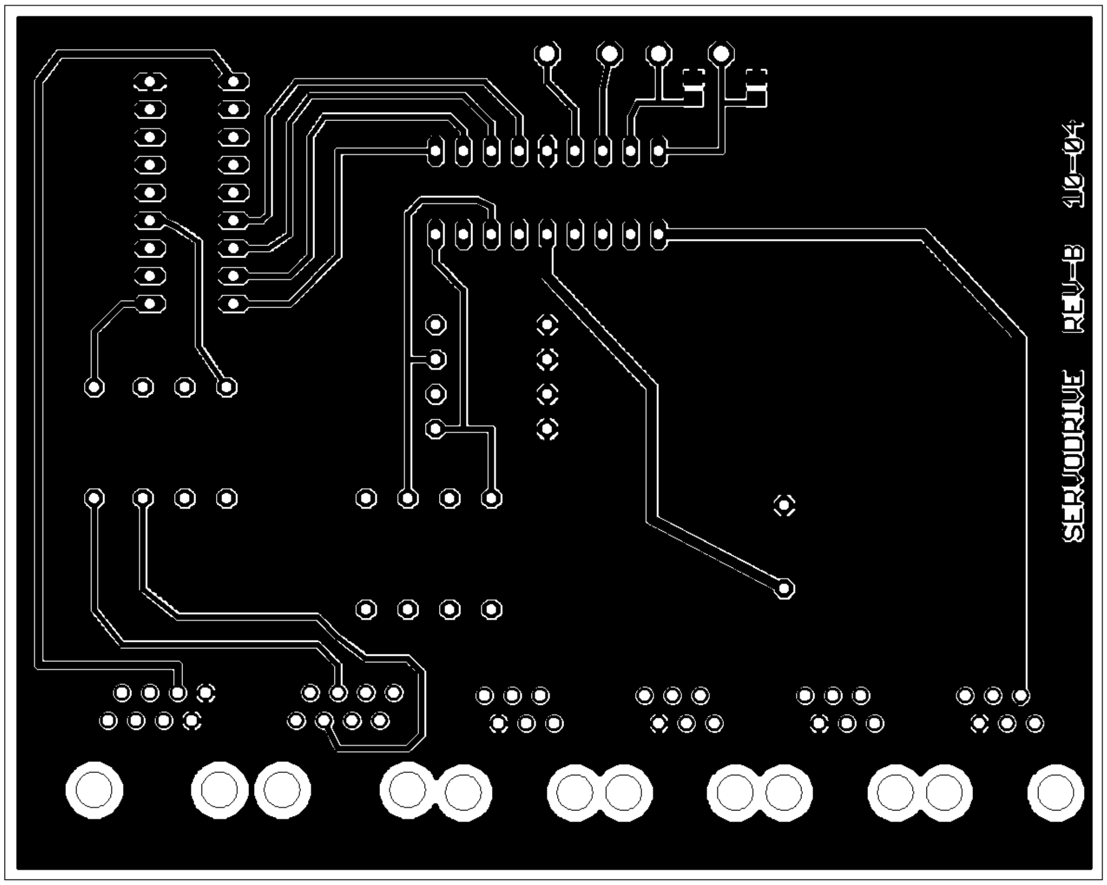
M:/Jlabgrp/inj_group/hansknecht/motion/RC servos/servoboard.

```c
//Servopneum628.c                                   John Hansknecht 5/2004
// Target processor is PIC16F628.  Internal oscillator at 4Mhz.
// Do not use watchdog timer or master clear function.
// Program performs a multi-tasking of 4 pulse width modulated signals
// to control the position of 4 separate servos.

#include <16F628A.h>
#use delay(clock=4000000)
#fuses NOWDT,INTRC_IO, NOPUT, NOPROTECT, BROWNOUT, NOMCLR, LVP, NOCPD

long curpos1 = 100;  // initialize the 4 servo current positions so they
long curpos2 = 100;  // can be tracked globally in the program.
long curpos3 = 100;
long curpos4 = 100;
long i;
/* Pulse function.  Accepts an integer binary value for the pin to be
   driven (pin) and the current pulse width (curpos) that should be placed to
   the output pin */

void pulse(int pin,long curpos)
  {
  curpos = curpos / 4;          //divide curpos by 4
  output_a(pin);                //port A is addressed as a whole, but only one
                                //pin of the port is accessed at a time.
  for (i=0;i<=curpos;++i)       //The pin is taken high for this delay.
         {
         delay_cycles(1);
         }
     output_low(PIN_A1);        //The pin is taken low again
  output_a(0);                  //We leave with the whole port low.
  }

/* Drop function.  Called when a servo is being commanded to drop its
   current position.  If we are early in the movement, the drop is fast and the
   servo moves fast.  Near the end of motion the units only drop by 1 unit
   at a time and the servo inches to its final resting postion. */

long drop (long temp)
  {
   if (temp >330)     //when > 330 we drop by 10 units each time through.
   temp = temp - 9;
   if (temp > 230)    //when > 230 but < 330 we drop by 1 unit.
   temp = temp -1;
   return temp;       //we exit with the updated count.
  }
/* Raise function.  Called when servo is being asked to raise its position.
   It is identical in operation to the drop function, but counts up instead. */

 long raise (long temp)
  {
  if (temp <= 440)    //if <440 we raise by 10 units each time through.
   temp = temp + 9;
   if (temp < 540)    //when >440 but <540 we raise by 1 unit.
   temp = temp +1;
   return temp;       // we exit with the updated count.
  }

/* Main routine.  Looks at the four input requests and tracks the current
   position of each servo.  If a move is requested it will go out and obtain
   the proper move step size by calling a drop or raise function.  It will
   then address the pin by calling the Pulse function */
void main()
{
while (true)   //infinite loop from here on.
```

```
      {
       if(input(PIN_B7))               //if PIN B7 is high then
       curpos1 = drop(curpos1);        //drop its current postion
       else curpos1 = raise(curpos1);  // otherwise, raise its current position.
          pulse(0b00000010,curpos1);   // send the pulse out to servo.
      if (curpos1 > 500)               // if near max position,
         output_high (PIN_B0);         // give the readback 15V.
      if (curpos1 < 260)               // if near min position,
         output_low (PIN_B0);          // give the readback 0 V.

       if(input(PIN_B6))               // same as above, but for servo 2
        curpos2 = drop(curpos2);
       else curpos2 = raise(curpos2);
          pulse(0b00000001,curpos2);
          if (curpos2 > 500)
         output_high (PIN_B1);
      if (curpos2 < 260)
         output_low (PIN_B1);

       if(input(PIN_B5))               // same as above, but for servo 3
        curpos3 = drop(curpos3);
       else curpos3 = raise(curpos3);
          pulse(0b10000000,curpos3);
      if (curpos3 > 500)
         output_high (PIN_B2);
      if (curpos3 < 260)
         output_low (PIN_B2);

       if(input(PIN_B4))               // same as above, but for servo 4
        curpos4 = drop(curpos4);
       else curpos4 = raise(curpos4);
          pulse(0b01000000,curpos4);
         if (curpos4 > 500)
         output_high (PIN_B3);
      if (curpos4 < 260)
         output_low (PIN_B3);
          /* we have now gone through and updated the pulse period for all
             four servos.  They have each received a pulse of the correct
             pulse width for the position they should be moving to.  Now we
             must delay a short period of time and repeat the pulses again
             to form a true PWM signal. */

       delay_ms(16);  // adjust this value to achieve a 60hz refresh rate
      }
}
```


PIC program is servopneum628.hex
POWER
Standard CAT5 power input
(Must be ground referenced)
J5
GND
+15V
+5V
COMMAND
J6
servo 1 readback
servo 2 readback
servo 3 readback
servo 4 readback
servo 1 command
servo 2 command
servo 3 command
servo 4 command
Commands are 24V to actuate
Must be ground referenced
Readback will source 15V
CAT5 Map
1= wht-org
2= org
3= wht-grn
4= blu
5= wht-blu
6= grn
7= wht-brn
8= brn
MCU1
PIC16F62X
RA2/AN2/VREF
RA3/AN3/CMP1
RA4/TOCKI/CMP2
RA5/MCLR/THV
VSS
RB0/INT
RB1/RX/DT
RB2/TX/CK
RB3/CCP1
RA1/AN1
RA0/AN0
RA7/OSC1/CLKIN
RA6/OSC2/CLKOUT
VDD
RB7/T1OSI
RB6/T1OSO/T1CKI
RB5
RB4/PGM
TP1
TP2
TP3
TP4
R13
R14
UDN2981AN
IC1
VS
GND
O1 O2 O3 O4 O5 O6 O7 O8
I1 I2 I3 I4 I5 I6 I7 I8
R15 600
R16 600
R17 600
R18 600
R5 18.2k
R6 18.2k
R7 18.2k
R8 18.2k
R9 4.75k
R10 4.75k
R11 4.75k
R12 4.75k
C1
1000uf
J1
J2
J3
J4
Servo 1
Servo 2
Servo 3
Servo 4


TP1
TP2
TP3
TP4
R14
R13
UDN2981AN
IC1
PIC16F62X
MCU1
R9
R10
R11
R12
4.75k
4.75k
4.75k
4.75k
600
600
600
600
R18
R17
R16
R15
R5
R6
R7
R8
18.2k
18.2k
18.2k
18.2k
C1
1000uf
J5
J6
J4
J3
J2
J1




| Part | Value | Device | Package |
|---|---|---|---|
| C1 | 1000uf 50V 7.5 mm lead spacing Radial (Vishay 515D108M050EK6A) (Allied part# 507-0043) | | |
| IC1 | UDN2981AN (Newark Part# 09F1706) $1.56 ea | | |
| J1 | AMP 520250-3 (allied part# 512-5770) | | |
| J2 | AMP 520250-3 (allied part# 512-5770) | | |
| J3 | AMP 520250-3 (allied part# 512-5770) | | |
| J4 | AMP 520250-3 (allied part# 512-5770) | | |
| J5 | AMP 520251-4 (allied part# 512-5774) | | |
| J6 | AMP 520251-4 (allied part# 512-5774) | | |
| MCU1 | PIC16F628 or 628A (Newark part# 92C4532) $2.44ea | | |
| R5 | 18.2k | | |
| R6 | 18.2k | | |
| R7 | 18.2k | | |
| R8 | 18.2k | | |
| R9 | 4.75k | | |
| R10 | 4.75k | | |
| R11 | 4.75k | | |
| R12 | 4.75k | | |
| R13 | | R-US_M0805 | M0805 |
| R14 | | R-US_M0805 | M0805 |
| R15 | 600 | | |
| R16 | 600 | | |
| R17 | 600 | | |
| R18 | 600 | | |