KIT 102. SERVO-MOTOR DRIVER

Servo motors are used in radio-controlled models (cars, planes), robotics, theme park special effects, test equipment, industrial automation. At the hobbyist end of the market they are small, compact and relatively inexpensive at around $US20. The motors themselves are black boxes which contain a motor, gearbox and decoder electronics. Three wires go into the box; 5V, ground and signal. A short shaft comes out of the motor which usually has a circular interface plate attached to it Most servos will rotate through about 100 degrees in less than a second according to the signal input. This Kit will control up to 4 servo motors simultaneously.

ASSEMBLY

Check the components in the kit against the Components List. Some of the resistors stand up on the board. Make sure to get the electrolytic capacitor and the IC1 around the correct way.

To complete the kit between one and four 5K - 10K potentiometers are required to produce the input signal. Connect each pot as a voltage divider with the center pin going to the signal input. Servo motors are required. They have not been included in this kit because users will usually have their own particular servos they wish to control.

CIRCUIT DESCRIPTION

All the work controlling the servos is done in the preprogrammed PIC micro-controller (uC). As such the kit provides a text-book example of how a uC can replace a handfull of IC's & other glue chips. Everything is done in software. Connect a 5V power supply capable of delivering an amp.

The input signals are between 0 - 5V delivered by connecting up the potentiometers as voltage dividers. Inside the PIC an AD converter (multiplexed when there is more than one input signal) changes the voltage signal into the Pulse Code Modulation system used by servo motors. This signal is a 5V pulse between 1 and 2 msec long repeated 50 times per second. That is, a 20msec frame rate. The width of the pulse determines the position of the server. Most servos will move to the center of their travel when they receive a 1.5msec pulse. One extreme of motion generally equates to a pulse width of 1.0msec; the other extreme to 2.0msec with a smooth variation throughout the range, and neutral at 1.5msec. The period between the pulses is used to synchronise the receiver.

Servos are closed loop devices. They are constantly comparing their position (proportional to the pulse width) to their actual position (proportional to the signal voltage input.) If there is a difference between the two the servos electronics will turn the motor to adjust the difference error. This also means that servos will resist forces which try to change their position. When a servo is unpowered or not receiving positioning pulses the output shaft can be easily turned by hand.

Kit 102 Components		
Resistors 1/4W, 5%:		
470K	R1 to R5	5
470R	R6 to R9	4
0.1uF (104)	C4	1
15pF ceramic capacitor....................	C1 C2	2
2200uF/16V electrolytic capacitor .	C3	1
3.579MHz crystal	XTAL	1
Programmed PIC16C71-04/P	IC1	1
18 pin IC socket................................	....	1
2 pole terminal block	....	1
K102 PCB	....	1
Potentiometers & servo motors not supplied.		

```
; PROGRAM: SERVO.SRC

; This program generates pulse width modulation from sampled voltages.
; The PIC 16C71 has four inbuilt ADC converters (actually one
; ADC which is multiplexed) which are set up in this case to read 0 - 5V
; as the binary values 0 - 255.
; The ADC results are loaded into a delay routine which is implemented
; using the real time clock counter (RTCC). Basically the RTCC counts
; up from the loaded value until it reaches 255 and then rolls over to
; zero, triggering an interrupt.

; As the program is intended to drive servos, there is also a fixed delay
; of about 0.8 milliseconds included. The controller thus raises the
; appropriate output pin for 0.8 msec plus the variable delay and then
; drops it again. The maximum pulse width is about 2.2 msec.

; Note that the four ADC's sample and output one at a time. Once all four
; have had a turn the controller is put to SLEEP which shuts everything
; down except the watch dog timer (WDT). When the WDT times
; out (in about 18 msec) it completely resets the controller and
; the process starts all over. Thus, in the case of all 0V inputs, the
; cycle takes 4*0.8+18 equals about 21 msec to complete.

; The following constants set the ADC clock source/ speed. Uncomment
one.

;AD_clk      =    0          ;PIC oscillator period x 2  (<= 1 MHz).
;AD_clk      =    64         ;PIC oscillator period x 8  (<= 4 MHz).
;AD_clk      =    128        ;PIC oscillator period x 32 (<= 16 MHz)
AD_clk       =    192        ;Independent RC oscillator, 2-6 us.

; The following constants select a pin for ADC input. Uncomment one.

AD_ch        =    0          ;ADC channel 0 (Ain0, pin 17).
;AD_ch       =    8          ;ADC channel 1 (Ain1, pin 18).
;AD_ch       =    16         ;ADC channel 2 (Ain0, pin 1).
;AD_ch       =    24         ;ADC channel 3 (Ain0, pin 2).
AD_ctl       =    AD_clk | AD_ch  ;Logical OR.

; The following constants determine which pins will be usable by the ADC
; & whether Vdd or ra.3 will serve as the voltage reference. Uncomment
one.

AD_ref       =    0          ;ra.0 through 3 usable, Vdd reference.
;AD_ref      =    1          ;ra.0 through 3 usable,ra.3 reference.
;AD_ref      =    2          ;ra.0/1 usable, Vdd reference.
;AD_ref      =    3          ;All unusable--digital inputs only.

              device
pic16c71,hs_osc,wdt_on,pwrt_off,protect_on
              id     'ADC1'

counter1     =    10h
counter2     =    11h
integer1     =    12h
integer2     =    13h
```

KIT 102. SERVO-MOTOR DRIVER

```
dummy     =    14h
flag      =    15h
servo0    =    rb.5
servo1    =    rb.4
servo2    =    rb.3
servo3    =    rb.2

                org    0
                jmp    start

                org    4            ;Interrupt jumps here
                clrb   RTIF
                setb   flag.0
                reti

start                  mov    !ra, #255     ;Set ra to input.
                mov    !rb, #0        ;Set rb to output.
                clr    rb             ;Clear port rb
                mov    dummy,#255
                mov    intcon, #0     ;Turn interrupts off.
                mov    adcon0,#AD_ctl    ;Set AD  clock  and
channel.
                setb   rp0            ;Enable register page 1.
                mov    adcon1,#AD_ref  ;Set usable pins, Vref.
                mov    option,#00001000b  ;WDT on, no prescale
                clrb   rp0            ;Back to register page 0.
                setb   adon           ;Apply power to ADC.

                setb   go_done        ;Start conversion.
not_done        snb    go_done        ;Poll for 0 (done).
                jmp    not_done       ;If 1, poll again.
                mov    counter2,adres ;Move ADC result into
counter.
                mov    integer1,#3    ;Offset constant
                mov    integer2,#5    ;ADC multiplier
                setb   servo0         ;Output pulse to servo 0
                call   delay
                clrb   servo0

                call   pause          ;ADC settling delay
                clrb   rp0            ;Ensure reg page 0
                clrb   chs1           ;Select channel 1
                setb   chs0           ; Ain 1
                mov    dummy,#255     ;Reload dummy variable
                clrb   adres          ;Make sure
                setb   go_done        ;Start conversion.
not_done1        snb    go_done        ;Poll for 0 (done).
                jmp    not_done1      ;If 1, poll again.
                mov    counter2,adres  ;Move ADC result into
counter.
                mov    integer1,#3    ;Offset constant
                mov    integer2,#5    ;ADC multiplier
                setb   servo1         ;Output pulse to servo 1
                call   delay
                clrb   servo1

                call   pause
                clrb   rp0            ;Ensure reg page 0
                setb   chs1           ;Select channel 2
                clrb   chs0           ; Ain 2
                mov    dummy,#255
                clr    adres
                setb   go_done        ;Start conversion.
not_done2        snb    go_done        ;Poll for 0 (done).
                jmp    not_done2      ;If 1, poll again.
                mov    counter2,adres ;Move ADC result into
counter.
                mov    integer1,#3    ;Offset constant
                mov    integer2,#5    ;ADC multiplier
                setb   servo2         ;Output pulse to servo 2
                call   delay
                clrb   servo2

                call   pause
                clrb   rp0            ;Ensure reg page 0
```

```
                setb   chs1           ;Select channel 3
                setb   chs0           ; Ain 3
                mov    dummy,#255
                clr    adres
                setb   go_done        ;Start conversion.
not_done3        snb    go_done        ;Poll for 0 (done).
                jmp    not_done3      ;If 1, poll again.
                mov    counter2,adres ;Move ADC result into
counter.
                mov    integer1,#3    ;Offset constant
                mov    integer2,#5    ;ADC multiplier
                setb   servo3         ;Output pulse to servo 3
                call   delay
                clrb   servo3

                sleep
                jmp    start   ;Time out after 18 msec

; The number of loops this delay routine makes is dependent on the result
of
; the AD conversion. The higher the voltage, the longer the delay.

delay           clrb   rp0            ;Page 0
                mov    intcon,#10100000b    ;Enable  RTCC
interrupt

;**************** Fixed delay part of routine ****************

delay1          mov    RTCC,#55       ;Fixed delay
wait1           jnb    flag.0,wait1    ; of 200 till interrupt
                clrb   flag.0         ;Flag set on interrupt
                djnz   integer1,delay1 ;Three times through


;**************** Variable delay part of routine **************

                sub    dummy,counter2  ;RTCC counts UP!
load            mov    RTCC,dummy     ;Load RTCC
wait2           jnb    flag.0,wait2   ;Note infinite loop
                clrb   flag.0
                djnz   integer2,load  ;Five times through

                mov    intcon,#0      ;Disable interrupt
                ret

pause           mov    counter1,#120  ;Adds a short settling
settle                 djnz    counter1,settle     ; time to the
ADC
                ret

***********************************************************
*
```

KIT 102. SERVO-MOTOR DRIVER

POTENTIOMETER
INPUTS

INPUT 1
+5

INPUT 2
+5

INPUT 3
+5

INPUT 4
+5

R5
470K

+5

+5

C4
.1uF

C3
2200uF/16V

R1
470K

R2
470K

IC1
16C71-04/P

1	18
2	17
3	16
4	15
5	14
6	13
7	12
8	11
9	10

R4
470K

R3
470K

XTAL
3.579

C1
15P

C2
15P

R6
470   ○ 1

R7
470   ○ 2

R8
470   ○ 3

R9
470   ○ 4

SERVOMOTOR
SIGNAL
OUTPUTS