The SANDSTONE & TERMITE RAILWAY
"Linking the Outback to the World"

PICAXE TUTORIALS

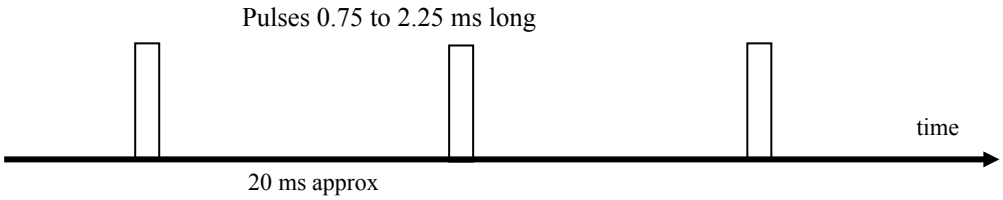
4. Using Picaxe with RC 'servo' pulses

NOTE: All these programs are for battery operated locos.

A quick Introduction to servos

Servo Motors come with three wires or leads. Two of these wires are to provide ground and positive supply to the servo DC motor. The third wire is for the control signal. These wires of a servo motor are color coded. The red wire is the DC supply lead and must be connected to a DC voltage supply in the range of 4.8 V to 6V. The black wire is to provide ground. The color for the third wire (to provide control signal) varies for different manufacturers. It can be yellow, white, brown etc.

Radio control of the model plane variety operates using what I call 'servo type' pulses. These consist of +5V pulses varying from 0.75 ms to 2.25 ms separated by about 20 ms of 'off' time. This will cause a rotation of about 200 degrees **BUT it does vary between servos**.
NOTE many RC receivers only output pulses to rotate a servo about 90 degrees!



Servos or electronic speed controllers (ESC) decode these pulse to position a servo's rotation angle, or to output a motor control voltage.

The angle is determined by the duration of a pulse that is applied to the control wire. The length of the pulse will determine how far the motor turns. For example, a 1.5 ms pulse will make the motor turn to 'neutral position'.
When a pulse is sent to a servo that is less than 1.5 ms the servo rotates to a position and holds its output shaft some number of degrees anticlockwise from the neutral point. When the pulse is wider than 1.5 ms the opposite occurs. The minimum width and the maximum width of pulse that will command the servo to turn to a valid position are functions of each servo.
When a servo is commanded to move it will move to the position and hold that position. If an external force pushes against the servo while the servo is holding a position, the servo will resist from moving out of that position. The maximum amount of force the servo can exert is the torque rating of the servo.

Normally you'd just use the servo or ESC to decode the pulses, but you can use the Picaxe to do it if you want a different type of output. You can also just use the Picaxe to control a servo without any RC.

'Reading' servo pulses

Here's an example of a program to read in 'servo' pulses.
The **PULSIN** command (no "E" in pulse) is used to detect the pulses from a RC receiver. This command will output a number corresponding to the pulse width measured in 10us units. (at 4 MHz clock) ie a 0.75ms pulse will 'measure' as 75 and a 2.25ms as 225.
Note that if no pulse is detected within 0.65 sec, the command produces a zero output. Also note that the program cannot do any other processing (except SERVO/PWMOUT) during the time it is waiting for a pulse. So if no pulse is received the programs 'hangs' for 0.65 sec. If correct pulses are being received, it can take from 20 to 40ms while the program waits.

So if the pulses are on pin0, and we define
SYMBOL timecounts=w0

Then we just look for a positive pulse on pin0
PULSIN 0,1,timecounts
IF timecounts=0 **THEN** … go do something - there is no signal being received.

Otherwise, timecounts can now be used to determine a course of action.

Example 1: Say we want the Picaxe to play a SOUND as a whistle sound output on pin1, when the input counts exceed 170. (This 170 corresponds to a Transmitter stick being moved more than halfway right from its central position – see Appendix.)

IF timecounts>170 **THEN GOTO** whistle:
If it's not then just continue at start, but if it is use:

Whistle: **SOUND** 1,(112,100) numbers around 112-118 sound like a whistle

You could use a servo signal in to produce a PWM or DC voltage out – in effect a motor ESC. In this case you would read the input signal and pass it straight to a PWMOUT command (see *Advanced Picaxe Motor control.doc*)

Controlling servos.

Servos can be used to operate leveling crossing gates or booms, or as turnout 'motors', or to operate semaphore signals.
We can use the SERVO command to control a servo. (but cannot use it at the same time as a PWMOUT command). The SERVO command must be used on pin2 in an 8pin Picaxe.

Example 2: – lower a level crossing boom arm by 90 degrees when a contact on pin3 closes.
Assume that raised arm is counts and lowered is counts. These will have to be determined by trial and error, as they depend greatly on the servo. Also, I have assume lowering needs a reduced number of counts. This may be the opposite depending on the physical arrangement.
Define
SYMBOL raisedcounts=200
SYMBOL loweredcounts=100
Then the program flow will look like this:

```
Start:
PAUSE 50            'no need to do this too fast, so have a rest!

IF pin3=0 THEN GOTO raise
'So pin3 must be a 1 and we want to lower the boom.
SERVO 2, loweredcounts
GOTO start

Raise:  SERVO 2, raisedcounts
GOTO start
```

It is possible to control more servos even though there is only one pin that supports the SERVO command. In this case we use the PULSOUT command to generate the pulses and then ensure they are resent every 20 to 80 ms. So if in the above example we also want to control a servo on pin0 with an input on pin4, we can use:

```
SYMBOL raisedcounts1=200
SYMBOL loweredcounts1=100
SYMBOL raisedcounts2=180
SYMBOL loweredcounts2=90
```

Then the program flow will look like this:

```
Start:
PAUSE 50            'provides the space between the pulsout commands 20 to 80 seems to work

IF pin3=0 THEN GOTO raise1
'if we get to here, pin3 must be a 1 and we want to lower the boom.
SERVO 2, loweredcounts1
GOTO second

Raise1:  SERVO 2, raisedcounts1

Second:             'process the 2nd input
IF pin4=0 THEN GOTO raise2
PULSOUT 0, loweredcounts2
GOTO start
Raise2:
PULSOUT 0,raisedcounts2
GOTO start
```

I find that the PULSOUT command gives a more 'jitter-free' servo operation.

Complete listing for:-
twoservos.bas

```
'example program to control two servo motors,
'one using SERVO command and one using PULSOUT.
'08M and 08M2 types
'G Hunter    20/4/14            30 bytes

'pin0 is 2nd servo output
```

```
'pin2 is 1st servo output
'pin3 is 1st servo control input - 0 to raise, 1 to lower
'pin4 is 2nd servo control input as above

SYMBOL raisedcounts1=200
SYMBOL loweredcounts1=100
SYMBOL raisedcounts2=180
SYMBOL loweredcounts2=90


'---------------------------------------------
Start:
        PAUSE 50                'provides the space between the pulsout commands

        If pin3=0 then raise1
'So pin3 must be a 1 and we want to lower the boom.
        SERVO 2, loweredcounts1
        Goto second

Raise1:
        SERVO 2, raisedcounts1

Second: 'now check the 2nd input
        if pin4=0 then raise2
        PULSOUT 0, loweredcounts2
        GOTO start
Raise2:
        PULSOUT 0,raisedcounts2
        GOTO start
```

Appendix A: Servo pulse widths and Transmitter stick positions

measured Hobbyking Rx pulse widths in msec
transmitter stick position

centred	1.50	centred	1.5
left direction	1.05	full up	1.85
right direction	1.95	full down	1.1

Other transmitter/receiver combinations may be different.

Appendix B: Byte usage in these example programs

word	byte	Steam chuff	diesel	Motor control
w0	b0		Seed (shift register)	Timecounts (pulsin)
	b1			
w1	b2	Speed input voltage	Speed input voltage	PWMspeed output
	b3			
w2	b4	chufftime	pausetime	
	b5	chuffcounter	Lookuptable ?	
w3	b6	offtime (ms)		
	b7			
w4	b8	soundpin	soundpin	
	b9	Temporary in cutoff calcs		
w5	b10	oldspeed1	oldspeed1	
	b11	oldspeed2	oldspeed2	
w6	b12	PWMvolume (PWM 15kHz)	PWMvolume (PWM 15kHz)	
	b13			