RC PWM Control

Associated Files:

- RC_Control.qcp
- RC_Control_feedback.qcp

Background

Radio Control is commonly used for live action stage performances, allowing an operator or operators, using RC transmitters to remotely operate mechanics on the stage. RC control can also be used in other remotely controlled operations, where un-tethered operation is desired.

The RC transmitter multiplexes multiple channels into a data stream that is transmitted. The receiver demultiplexes these channels producing multiple outputs of logic (3.3v or 5v) pulse width control channels. Common pulse width are 1 ms to 2ms ms to approximately 0.5 ms to 2.5ms. The spacing between pulses is dependent upon the longest pulse width, and the number of channels being multiplexed. (The old standard was a 50Hz pulse rate, but newer systems shorten this time). Typically, the shorter pulse width represents Counterclockwise operation and the longer pulse indicates Clockwise operation, but this may vary in actual application. The mid value would indicate stopped or centered. (Note that a throttle which may only has positive values could map the whole pulse width differently!).

A couple of popular servo systems: JR servos use a 3 pin connector (1=Brown = Negative supply, 2=Red = Positive supply, 3=orange = Signal), Futaba (1=black = Negative supply, 2=Red=Positive supply, 3=White=Signal).

Reading Pulse Width into a SilverDust

A new mode has been added via the Set Mode Command. The Set Mode Command (SMD) with a mode of 7 allows the secondary encoder channel hardware to be used to capture two pulse width inputs. IO6 captures the pulse width into register 200 and IO5 captures the pulse width into register 201. The values are limited to a range of 0 to 65535. The time base is adjustable via the third (setting) parameter of the SMD command. The clock time base is set to 40MHz / 2^n where n is between 1 and 7 (inclusive). (The mode may be exited by setting n=0.)

Note: The SilverDust code revision must be at least 53-1x to support this command. Request this revision when you order or contact Support to update in the field.

A value of n=1 provides a full scale value of approximately 50,000 counts for a 2.5ms pulse.

This data is available to the program running within the SilverDust. RC_Control.qcp shows a simple way to scale the output to produce the desired range of motion from the servo. The PIM command is used to scale the commanded range from the RC counter into actual position units.

RC_Control_feedback.qcp uses the VIM command to compare a commanded position with a voltage from a feedback potentiometer to provide absolute position control (with reference to the feedback potentiometer) from the RC control counter.

Simple Position Control

RC_Control.qcp

Line# Oper	Label	Command
1:REM		RC control of a servo system Input is taken from a standard RC receiver. This is a LVTTL signal of 3.3v or 5v with a pulse width of 0.7 to 2.3 ms. The pwm signal is connected into IO6, with the ground connected to logic ground. Note a functioning system may also need to use a home routine to set the initial center position.
2:REM		Enable the pulse width read command. The setting makes the clock into the counter: Clk = 40MHz / 2^n where n is the setting value. A value of 1 appears to work welll for the RC This nominally produces a count range of 14000 to 46000 in register 200. This value is updated every 7.5 to 20ms (typical RC transmitter).
3:SMD		Set Mode: Mode=7 Setting=1
4:REM		Set up the PIM command to map the control signal onto the desired range of positions.
5:REM		We will feed the measured commanded position into Register 12, with a zero centered value. This will be done in a loop at the bottom.
6:REM		Set the saturation limit on the input. This sets the input scale that will correspond to the full scale output value. Our input can vary from 14000 to 46000, a range of 32000 counts. If we center mean it in the loop, then the value written to register 12 will range from -16000 to + 16000 counts
7:WRP		Write 16000 to "User or Maximum Scale|Limit[15]" Register
8:REM		Set the output motion to be coverd by the input - in this case 5 revolutions of the motor
9:WRP		Write 5 rev to "User or Maximum Output Scale[16]" Register
10:REM		Set the maximum velocity for the motor - we will pick 10 revolutions per second or 600 RPM
11:WRP		Write 10 rps to "User or Output Rate of Change[18]" Register
12:REM		Enable the loop to operate while the PIM motion is active
13:EMT		Enable Multi-Tasking
14:PIM		Position Input Mode:
15:REM		We will make sure that we have an input pulse before calculating
16:JLE	LOOP	Jump to "LOOP" When "External Encoder Position[200]" <= 5000
17:REM		Offset the pulse to its mid point
18:CLD		Accumulator[10] = External Encoder Position[200] - 30000
19:REM		Limit the value to the wanted output range just to be safe (use this to prevent out of range pulse from exceeding the desired position range. Not that the saturation on the input to the PIM command will also limit the range symmetrically, these limits may be used to independently limit the two directions.
20:REM		enforce the lower limit
21:CLD		Accumulator[10] = Max of Accumulator[10] or -16000
22:REM		enforce the upper limit - save to the target value
23:CLD		User or Input Source Data[12] = Min of Accumulator[10] or 16000
24:REM		Clear the pulse - we will wait for the next valid value
25:CLC		Clear External Encoder Position[200]
26:JMP		Jump to "LOOP"

Line 3 starts the pulse width measurement mode. The setting of 1 corresponds to a 20MHz counting rate.

Line 7 sets the input range that will correspond to the output motion range.

Line 9 sets the distance corresponding to each direction of saturation for the input range. That is +/- 16000 counts will be mapped onto +/- 5 revolutions of the motor from home.

Line 11 sets the maximum motor speed to use when moving. Here it is 10 revolutions per second.

Line 13 permits the loop to operate while the PIM command is still active

Line 14 starts up the Position Input Mode command.
Line 16 delays until a pulse is received.
Line 18 offsets the raw pulse width count so as to make it span +/- 16000 counts.
Lines 21 and 23 limit the input range if needed to limit the motion range asymmetrically.
Finally Line 25 clears the last read pulse value and 26 loops to read the next pulse.

RC Control with Feedback

Line# Oper	Label	Command
1:REM		RC control of a servo system with resistive feedback potentiometer Input is taken from a standard RC receiver. This is a LVTTL signal of 3.3v or 5v with a pulse width of 0.7 to 2.3 ms. The pwm signal is connected into IO6, with the ground connected to logic ground. Note a functioning system may also need to use a home routine to set the initial center position.
2:REM		Enable the pulse width read command. The setting makes the clock into the counter: Clk = 40MHz / 2^n where n is the setting value. A value of 1 appears to work welll for a standard RC controller. This nominally produces a count range of 14000 to 46000 in register 200 (IO6) and for register 201 (IO5). This value is updated every 7.5 to 20ms (typical RC transmitter).
3:SMD		Set Mode: Mode=7 Setting=1
4:REM		This program implements a secondary absolute resistive feedback from a potentiometer coupled to the output of an actuator. The motor also uses its own encoder feedback to commutate the motor and close the inner control loop. Feedback potentiometer is connected into IO7 (analog4). The input must be between 0 and 3.3v. A series resistor may be used to divide down the 5v provided at the SMI connector down to the needed 3.3v.
5:REM		We will feed the control signal into register 12; we will subtract the analog input by reading it into Register 13. We will do an initial read of the analog input into register 12 so that the motor will not move until a pulse is present to calculate
6:ARI		Analog Read Input: "User or Input Source Data[12]" = "Analog Channel #4"
7:ACR		Analog Continuous Read: "User or Input Offset[13]" = Analog Channel #4
8:REM		Set the deadband which will allow the servo to settle when "close enough"
9:WRP		Write 100 to "User or Input Dead Band[14]" Register
10:REM		Set the saturation limit on error, this error will produce the maximum velocity. A smaller value here will map the maximum velocity to a smaller error, effectively raising the gain of the control loop
11:WRP		Write 3000 to "User or Maximum ScaleLimit[15]" Register
12:REM		Set the maximum motor speed;
13:WRP		Write 10 rps to "User or Maximum Output Scale[16]" Register
14:REM		Set the acceleration limits
15:WRP		Write 1256 rps/s to "User or Output Rate of Change[18]" Register
16:REM		Set the Velocity input mode (VIM) to prodoce a motion according to the input and given parameters
17:EMT		Enable Multi-Tasking
18:VIM		Velocity Input Mode:
19:REM		We will make sure that we have an input pulse before calculating
20:JLE	LOOP	Jump to "LOOP" When "External Encoder Position[200]" <= 5000
21:REM		We will offset the input, then scale up , then offset to the output, then limit the value
22:REM		Offset the pulse to its mid point, reversing the sign In our test set, 29654 was the pot centered counter value value.
23:CLD		Accumulator[10] = 29654 - External Encoder Position[200]
24:REM		fractional multiply to adjust the scale factor to get the desired output range from the measured input range. 1.8846 is the needed scale factor from the system tested. 1.8846 * 65536 = 123513 When we right shift by 16 places we are dividient by 65536 , leaving the desired 1.8846 scale factor
25:CLD		Accumulator[10] = (Accumulator[10] * 123513)>>16
26:REM		offset to the desired output midpoint
27:CLD		Accumulator[10] = Accumulator[10] + 15500
28:REM		Limit the value to the wanted output range just to be safe
29:REM		enforce the lower limit
30:CLD		Accumulator[10] = Max of Accumulator[10] or 1000
31:REM		enforce the upper limit - save to the target value the commanded position will be based on this result
32:CLD		User or Input Source Data[12] = Min of Accumulator[10] or 30000
33:REM		Clear the pulse - we will wait for the next valid value
34:CLC		Clear External Encoder Position[200]
35:JMP		Jump to "LOOP"

RC_Control_feedback.qcp

This program controls a dual loop system, incorporating both the motor-encoder feedback and an analog absolute feedback from an actuator. This allows for a system that does not need homing on power-up.
Line 3 starts the pulse width measurement.

Line 6 takes an initial measurement of the analog feedback from the external potentiometer. This will be used as the initial requested position until a pulse is received.

Line 7 starts a continuous read of the external analog signal.

Line 9 sets the dead-band.

Line 11 effectively sets the system gain by setting the input error corresponding to maximum speed. A lower value here raises the gain.

Lines 13 and 15 set the maximum velocity and acceleration.
Line 17 allows the loop to operate while the VIM command, Line 18, is active.

Line 23 calculates the encoder center range. This would be 30000 if the controller were accurately adjusted. Varying it can be used to calibrate the system.

Line 25 allows the commanded secondary loop position , corresponding to the analog signal read from the feedback potentiometer, to be scaled as desired. The actual scale factor is multiplied by 65536 to allow a fractional effective value in line 25.

Line 27 offsets the result to the center of the actuator feedback range.

Lines 30 and 32 are used to limit the requested position to those available and/or desired by the output actuator. The actual feedback potentiometer is normally not moved to its limits, so limiting the requested range to somewhat less than the actuator can reach will prevent driving the actuator against its mechanical limits.

Line 34 clears the last pulse so we can see the next one. Finally line 35 loops back to process the next control pulse.