Running Servo's from QL Based Systems

By Ian Burkinshaw

Originally Published in QL Today Vol 17, Issue 4, June-Oct 2013

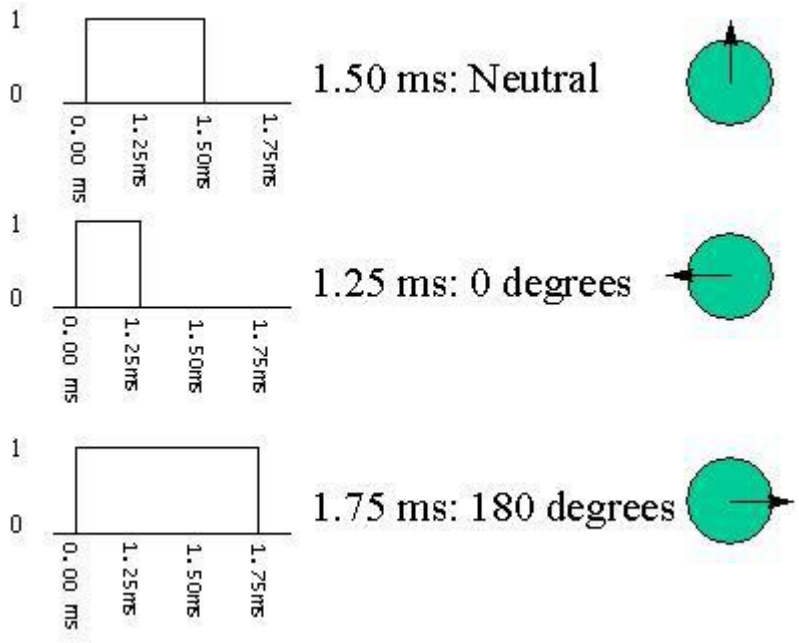
For my final article for QL Today I thought I would build on the theme of interfaces that can be connected to either the USB port on PC based system as well as serial (RS232) ports. I have done this with my series on the I2C interface, offering different solutions. Also my review of the PS2 mouse interface in Vol 17 issue ? of QLToday which uses the CTL ports. Without having to delve too deeply into the PC or QL hardware.

In this article I will be looking at driving servo's, like the ones used by radio modellers. These in fact can be used for all sorts of applications, such as robotics, automation and remote control as well as all sorts of modelling. For example model railway points, which give a far more realistic action to points going over. So not just radio controlled cars and aircraft. In fact I use two of these servo's to remotely control a loop antenna, one servo used for rotating the antenna itself and one for tuning the antenna.

So what is a servo ? Simply a device that converts a control signal into a physical angular position. It can also set a speed for an electric motor, there are motor speed controllers that use the same control signals as positional servo's.

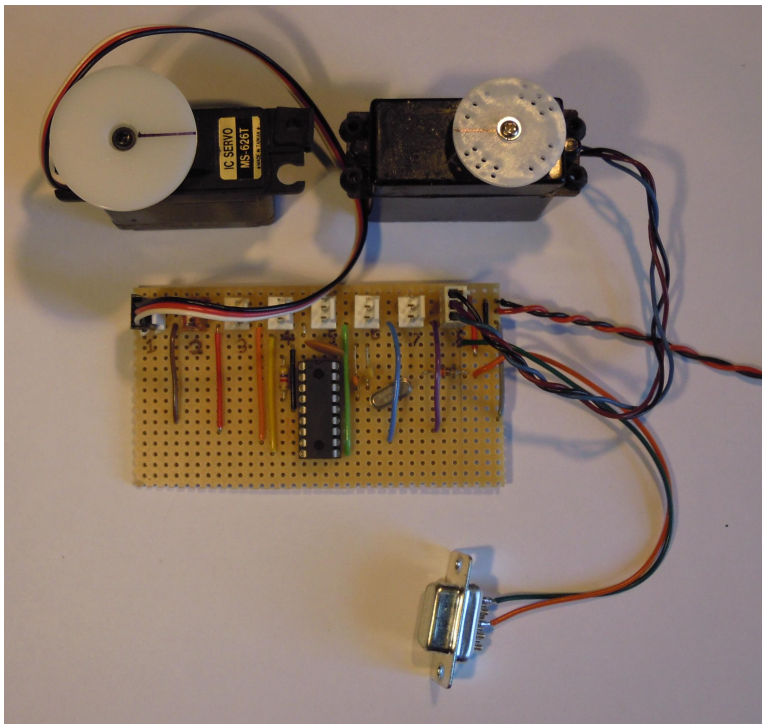
In the main, analogue servo's have three wires, power, ground and signal input. The power requirements of most servo's is between 4.5V and 6V. So the power supply for this interface and the servo's can be the same. However there are servo's around that have different power requirements, for example HV types. These use the same signal as the 5V type servo's, but require a higher supply voltage. In the main these HV servo are more powerful. So it is worth checking the power requirements of the servo's you wish to use. Also ensure your power supply can supply sufficient current to drive all the servo's (motor controllers) at the same time, this can add up.

Now as to the signal which controls the servo, this is a pulse sent every 20mSec. It does not have to be this way you can send the pulse just once and the servo will move and stay where it is put. What is important is the width of the pulse, since it is this that sets the angular position of the servo as shown below.



Searching the internet I found a PIC project that is ideal for these servo's. The interface can drive up to 8 Servo's. As it uses the standard RS232 serial interface, so can be used with Black Box QL's or with a USB to RS232 serial converter, can be used with PC based systems for example QPC2. The interface uses a PIC to translate the command from the computer to the variable pulse width signal required by the servo's. The PIC used is PIC16F84, as has been used in some of my other projects. I discussed the topic of programmers in my review of the PS2 Mouse to Games Port Converter in QLT Vol 17, Issue 2, page 6 and in my I2C article in QLT Vol 17, Issue 3, page 48. So if you have made the investment in a programmer, as have discussed in the past, then you should be OK for this project. Only a minimal number of components are required for this project. A clock crystal , capacitors and resistors. A 5 Volt power supply is required.
I will not waste space here with the circuit, since it is available from the following web site. Also the required HEX files to program the PIC is also available as well from this site.

http://www.rentron.com/serialservo.htm



As you can see from my example shown above, this really is very simple circuit and lend themselves to be constructed on strip (Vero) board.

These are typical connections for some popular servo's

Manufacturer	Positive	Negative	Signal
Airtronics (Obsolete)	RED	BLACK (in the middle)	BLACK, WHITE or BLUE
Airtronics / Sanwa (Obsolete)	RED	BLACK	WHITE or YELLOW
Airtronics / Sanwa	RED	BLACK	BLUE or YELLOW
Futaba	RED	BLACK	WHITE
Hitec	RED	BLACK	YELLOW
Japan Radio	RED	BROWN	ORANGE
Tower Hobbies	RED	BLACK	WHITE
Kyosho / Pulsar	RED	BLACK	YELLOW

So having constructed the interface, here is how to use it.

The protocol for the RS232 version is very simple. Just print to the serial port chosen, first the servo CHR$(0 to 7) followed by CHR$(0 to 255) which sets the position of the servo. Note do not send the line feed (CHR$(10)) or carriage return (CHR$(13)), to stop these, put a semicolon (;) at the end of the print line, as shown below.

Below is a simple test routine to test the interface. This program will first run the servo 10 times from one end of it travel to the other end and then back for each servo in turn. Then it will increment the servo one count at a time. This shows the minimum movement than can be achieved. Again it does this for each servo in turn.

If you find that the servo is being over or under exercised, then adjust the lower_limit and/or upper_limit variables. Not all servo's are the same in the way they respond to a given pulse width.

```
10 BAUD 2400:REMark Baud rate of the PIC16F84 Servo Controller board
20 OPEN#3;ser1i:REMark open serial port one, no handshaking
30 lower_limit=60:REMark value of lower limit of servo, need to be adjust for the servo used
40 upper_limit=215:REMark value of upper limit of servo, need to be adjusted for the servo used
50 FOR s=0 TO 7:REMark step servo's
60 FOR a=1 TO 10:REMark count full deflection test cycles
70 servo_drive s,lower_limit:REMark drive servo to lower limit
80 PAUSE 50:REMark wait
90 servo_drive s,upper_limit:REMark drive servo to upper limit
100 PAUSE 50:REMark wait
110 NEXT a
120 NEXT s
125 REMark second test, to step though the deflection range
130 FOR s=0 TO 7:REMark step servo's
140 FOR a=lower_limit TO upper_limit:REMark count servo step's
150 servo_drive s,a:REMark set servo
160 PAUSE 50:REMark wait, needs this time for the data to be transmitted, the PIC process the data and the servo itself to repond.
170 NEXT a
180 NEXT s
190 CLOSE#3:REMark close serial port
200 DEFine PROCedure servo_drive(servo,POSITION)
210 PRINT#3;CHR$(servo);CHR$(POSITION);:REMark drive data to servo
220 AT 0,0:PRINT "Servo #";servo+1;" Servo Position ";POSITION;"    ":REMark print to screen data set to servo's
230 END DEFine
```

So this is the end for QLToday. Like everybody else I am sorry to see it go. But on the other hand it was inevitable this was going to happen sooner or later. I fully understand Jochen and Geoff's decision. I have been amassed that it has lasted 17 years, just shows the strength of support from all the editors and contributors over the years, including you the readers. Without which there would have been no point. I have enjoyed contributing, and should a replacement come along, in whatever form, I would be happy to continue to contribute. I do hope I have given you some ideas of what can be done in hardware terms with your QL systems. I do plan to continue and contribute to Quanta, also I may put all my projects together into some form of book, most likely in PDF form. Since I had started to prepare further articles for QLToday both hardware and software based, so they may well still see the light of day. That is assuming there is any interest. So thank you all for reading my articles hope you enjoyed them. Bye for now.