Matrix Knowledge Exchange

Controlling a Servomotor with Timer0

by Martin Whitlock, July 2011

Abstract

This article describes a method of controlling a model servo motor using a low pin count microcontroller running from its internal RC oscillator.
A timer interrupt is then used to maintain timing accuracy while allowing the micro-controller to perform other tasks.

Requirements

Software:

- Any licence of Flowcode v3 or v4 for PIC.

Hardware:

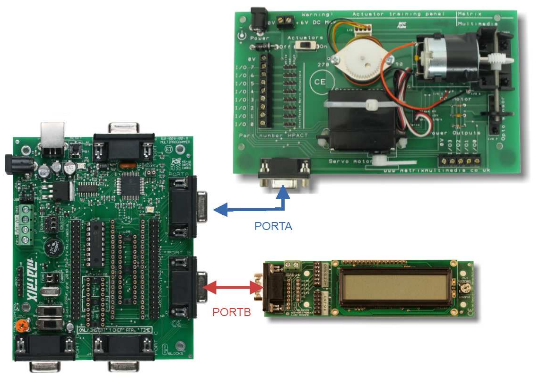
- HPACT Motors board (or other suitable servomotor)
- EB006 multiprogrammer with 16F88 chip
- Various E-blocks (LCD-EB005, Sensors-EB003)

Hardware Set-Up

- Connect the EB003 Sensors Board E-block (or any other suitable value variable resistor (5k-100k) to PORTA1 of the multiprogrammer (requires +5V), the jumper on the EB003 should be connected to J5 (default position) allowing the variable resistor to supply a voltage to PORTA1 from 0 to 5V. A patch lead is required from P2(0) of the EB003 to I/O 4 of the HPACT Motors board to supply the servo PWM signal.

- HPACT Motors board powered by a 6V DC supply. A common 0V connection is required between the multiprogrammer and the HPACT board.

- EB005 LCD E-block connected to PORTB of the multiprogrammer (requires +5V).



Operation

In order to drive a standard RC type servo motor, a PWM signal with a frequency of 50Hz (period = 20ms) is required. Whatever the starting position of the servo is, (when power is first connected) a pulse duration of 1.0ms every 20ms will drive the servo to its full clockwise position.
Note: The servo is a position controlled device, so if it is already in its full clockwise position, on power start up, then it will not move (unlike a normal DC motor, which will keep rotating if drive is pre-sent).

Likewise a pulse duration of 2.0ms every 20ms will drive the servo to its full counter clockwise posi-tion. Again if the servo is already in its full counter clockwise position on power start up, then the servo will not move since it is already in the position the input signal is demanding.

Interrupts

The servo requires a reliable and accurate method of generating a 1.0 to 2.0ms pulse every 20ms. A timer interrupt is used to prevent timing inaccuracies which could be caused by the microcontroller performing other tasks:

I am using a target device with an internal oscillator set at 8MHz. Timer0 is set up to generate an interrupt every 0.1ms. If you look that the timer0 interrupt on the flowchart, you will see that when using a 8MHz oscillator, the Interrupt frequency will be 7812.5 Hz, giving a period of 0.128ms.

We need to find a way of increasing the interrupt frequency from 7.8KHz to 10KHz. Timer interrupts work by continuously counting via an internal register (tmr0 for 8bit, or tmr0l with 16bit timer) until 1 is added to 255 (8-bit timer mode). At this point rollover occurs as the internal counter goes from 255 to 0 and the interrupt is triggered. This triggering temporarily halts the program at its current position and allows the interrupt macro to be run.

After the interrupt macro has been completed, the main program continues from where it was halted. The most Important thing to note is that the timer interrupt macro must be completed before the timer0 interrupt is triggered again!

E.g. Avoid delays and loops in the interrupt macro—unless they can be guaranteed to be completed within a suitable time period; Avoid calls to other component macros that might contain delays (LCD etc).

If we now go back to increasing the interrupt frequency. As stated earlier the interrupt macro is triggered when tmr0 rolls over from 255 to 0. If the first instruction in the interrupt macro is to load the tmr0 register with a pre-set value, the overflow period can be reduced.

e.g. a C code block with the following instructions reduced the interrupt period from to 0.1ms.

```
tmr0=+81;
```

Note: By calculation the timer register should be incremented by 56, leaving 200 clock cycles (at 2MHz clock) between interrupts. The use of the RC clock might require individual tuning of this value.

So:

20ms = 20/0.1 = 200 lots of 0.1 = Count value of 200.

At the start of a 20ms cycle portA0 goes from low to high.

0.1ms is the maximum resolution for the duration of the pulses generated every 20ms. E.g. Every 20ms you can have a pulse duration of: 1.0ms, 1.1ms, 1.2ms etc, up to a final duration of 2.0ms.

The positional resolution is determined the characteristics of the servo (which can differ slightly between models and manufacturers).

To generate a pulse of correct duration every 20ms, which is determined by the pot setting (0 to 255) the following formula was used:

In this case 0 = 1.0 ms and 255 = 2.0ms

I have gone by the standard spec of servo, but it appears that most servos allow more moment if required.

To change the lower and upper ms values:

1st constant = max req ms(x10)- min req ms(x10)
last constant = min req ms(x10)

E.g If you want ADC@0 = 0.6ms and ADC@255 = 2.2ms

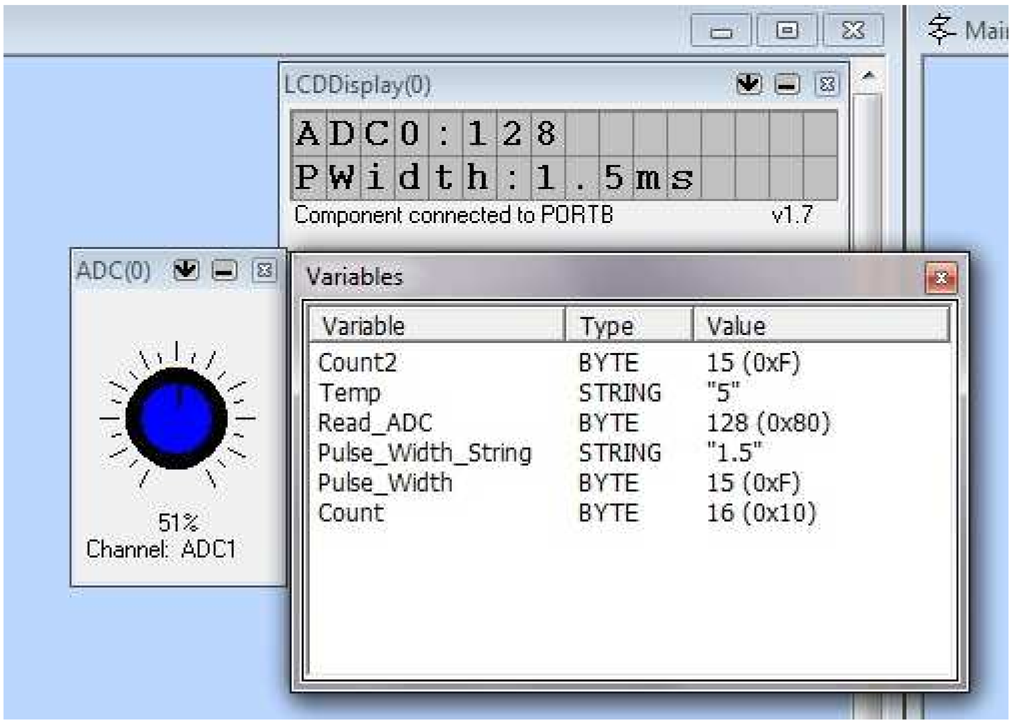
Then Pulse_Width=(22-6)*Read_ADC/255+6
=
Pulse_Width=16*Read_ADC/255+6

E.g If 1.5ms pulse duration is required:

After PortA0 goes high, Count2 is incremented every time the interrupt macro is accessed (every 0.1ms). So 1.5ms has elapsed after Count2 has been incremented 15times. (0.1*15 = 1.5ms)
Count 2 is compared with the value derived from the formula. (in this case is 15). For this to happen the pot has to be set so the ADC1 conversion reads 128 (mid position). Then if using:

Pulse_Width=10*Read_ADC/255+10
= 10*128/255+10
= 15

So when count2=15 portA0 changes from High to low.



```
ADC0:128
PWidth:1.5ms
```

Variable	Type	Value
Count2	BYTE	15 (0xF)
Temp	STRING	"5"
Read_ADC	BYTE	128 (0x80)
Pulse_Width_String	STRING	"1.5"
Pulse_Width	BYTE	15 (0xF)
Count	BYTE	16 (0x10)

Further reading

Below are some links to other resources and articles on related subjects, and technical documentation relating to the hardware used for this project...

Flowcode:	http://www.matrixmultimedia.com/flowcode.php
Learning Centre:	http://www.matrixmultimedia.com/lc_index.php
User Forums:	http://www.matrixmultimedia.com/mmforums
Product Support:	http://www.matrixmultimedia.com/sup_menu.php