Using Multiple RC type (PPM) servos with the TICkit 63/74

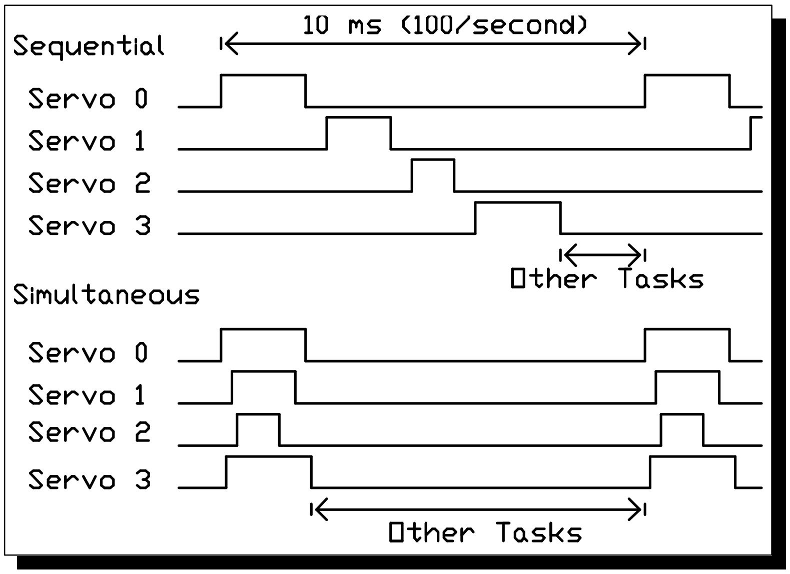
Submitted by:

Glenn Clark - Protean Logic Inc.

Introduction

The Radio Control (RC) servo used in model aircraft, cars, and boats is a very popular electromechanical actuator for machine control applications. This is not surprising considering the vast range of inexpensive yet capable servos available in this control format. RC servos are positioned with a pulse train of varying width, which repeats approximately 100 times per second. The pulses range in duration from 1 millisecond to 2 milliseconds which corresponds to angular deflection. Usually this movement occurs over a 90 degree arc.

Many designers have used simple pulse output statements, like the pulse_out_high() function of the TICkits, to generate control signals for RC servos. Although this method works, it wastes a great deal of processor time when more than one servo is to be controlled at a time. For example, in a system that uses 4 servos, each positioned mid way in their range, each servo will require a pulse of 1.5 ms. The total time to generate a pulse for each servo is 6 ms. Because the pulses must repeat 100 times a second to prevent straying, the pulses must repeat within 10 ms. This leaves only 4 ms of processing time during each pulse cycle. In other words, between 40 and 80 percent of the processors time is consumed generating pulses.



The TICkit 63 and 74 have a special function called servos() which much improves this situation. The servos() function generates these pulses on 4 output pins, simultaneously. So the same 4 servos would take 2 ms or less out of 10 ms to be controlled. So, only 20 percent of the processors time is consumed generating the pulse train. Furthermore, the TICkit's periodic interrupt capability can be used to generate these pulses exactly 100 times per second completely in a background process. The same 20 percent of the processors time is still consumed by the interrupt routine, but the main program does not need to keep track of the 10 ms cycle, and is completely unaware of the interrupt processing.

How the servos() function works

The servos function uses information passed to it in two arrays, each with 4 entries. The first array contains the pin numbers used to communicate to the servos. If fewer than 4 servos are used, a value of 255b can be used for the unused pin numbers, which inhibits the pulse generation for that array entry. The second array contains values that range from 0 to 255. Each of these values specifies the width of the pulse to be generated on the corresponding pin. A value of 0 produces a pulse width of 1ms, while a value of 250 produces a pulse width of 2 ms. Important to note, is the background time keeping in the TICkit works even while these pulses are being generated, so no time keeping is lost during the execution of this function. Even if no pulse width if 2 ms in length, the servos function requires a full 2 ms to execute.

The rising and falling edges of the pulses are staggered by 1 count's time. This is done to ensure accurate timing regardless of the other pulse width values, and to even out the startup current draws by the servos, by staggering the rising and falling edges of the pulses, no two servo motors should ever be starting or stopping at exactly the same instant. This reduces surge current requirements in the circuit.

The function does a few more things behind the scenes to make sure the pulses are stable. For example, it sorts the order of the pulse starts from longest to shortest and uses the staggering just mentioned to ensure that the pulses are exactly the length specified, regardless of whether or not another servo has the same pulse period. Each count for the position resolution is exactly 4 microseconds, which is very precise for this speed of processor.

If absolute stability is desired, for example in photographic positioning, you might consider turning the power off on the servo when not actively changing position. This will eliminate the minute adjustments of motion that take place upon every position refresh pulse.

How to Execute the servos() function in Background at 10 ms Intervals Automatically

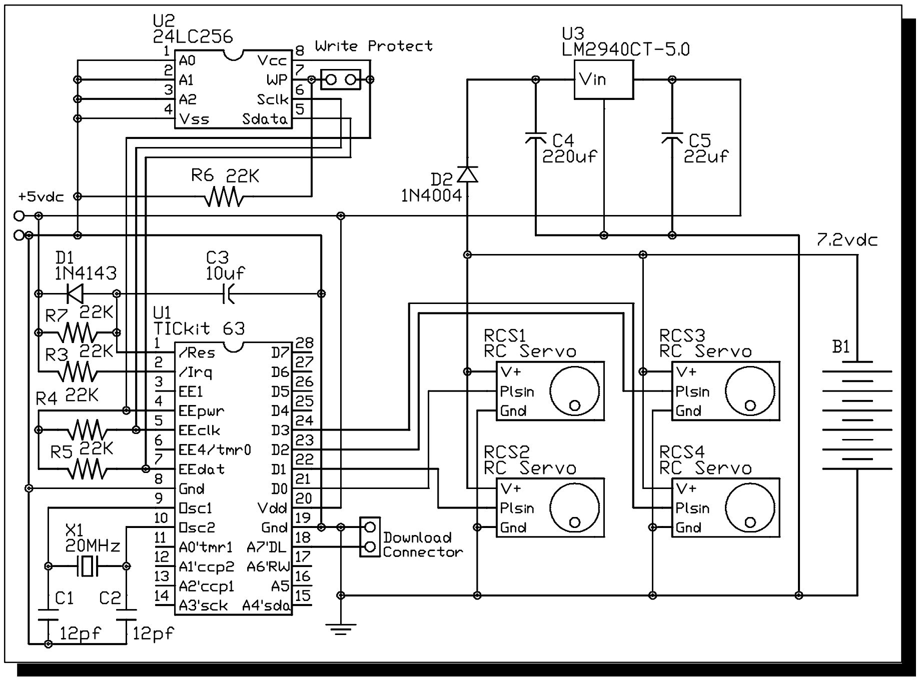
As mentioned earlier, the TICkit 63/74 have the ability to execute functions in background automatically at set intervals of time. This is called the periodic interrupt capability of the Real Time Clock (RTC). By setting a few registers and control bits, the TICkit 63/74 will call a function named timer_interrupt every time the period specified elapses. In the case of the RC servos, the interval is 10 ms for a 100 times per second refresh rate. An example program to setup the interrupts and the interrupt routine itself is shown below. The main routine only needs to change the pulse width values in the proper array to inform the servo control routine of the proper positions. The next time the interval elapses, the updated position information is used to generate the pulses. Of course, several calls to servos with different pin and position arrays could be made within the same timer_interrupt routine to control more servos. Obviously, no more than 16 servos can be controlled per TICkit, due to the limitation in processing time, and the fact that only 16 general purpose I/O pins are available on the TICkit interpreters.

```
; TICkit 63 RC Servo Array test

DEF tic63_i
LIB fbasic.lib

GLOBAL byte servo_times[4b]
GLOBAL byte servo_pins[4b]
GLOBAL byte my_count 0b


FUNC none timer_int
BEGIN
    servos( servo_times[0b], servo_pins[0b] )
    irq_on()
ENDFUN


FUNC none delay_int() ; just delays, but allows for interrupts during delay
    PARAM word in_count
BEGIN
    WHILE in_count
        --( in_count )
    LOOP
ENDFUN
```

```
FUNC none main
BEGIN
    rs_param_set( debug_pin )
    =( servo_pins[0b], pin_d7 )
    =( servo_pins[1b], pin_d6 )
    =( servo_pins[2b], pin_d5 )
    =( servo_pins[3b], pin_d4 )

    pin_low( pin_d7 )
    pin_low( pin_d6 )
    pin_low( pin_d5 )
    pin_low( pin_d4 )

    ; initialize the RTC timer for background servo refresh
    tmr1_cont_set( tmr1_con_on | tmr1_con_pre1 )
    rtc_intdiv_set( 5b )
    rtc_cont_set( rtc_enable | rtc_intm )

    irq_on()
    REP
        =( servo_times[0b], 0b )
        =( servo_times[1b], 0b )
        =( servo_times[1b], 0b )
        =( servo_times[1b], 0b )

        ; servo 1
        REP
            =( servo_times[0b] +( servo_times[0b], 1b ))
            delay_int( 100 )
        UNTIL not( servo_times[ 0b ] )

        ; servo 2
        REP
            =( servo_times[1b] +( servo_times[1b], 1b ))
            delay_int( 100 )
        UNTIL not( servo_times[ 1b ] )

        ; servo 3
        REP
            =( servo_times[2b] +( servo_times[2b], 1b ))
            delay_int( 100 )
        UNTIL not( servo_times[ 2b ] )

        ; servo 4
        REP
            =( servo_times[3b] +( servo_times[3b], 1b ))
            delay_int( 100 )
        UNTIL not( servo_times[ 3b ] )
    LOOP
ENDFUN
```

Electrical Considerations when Using RC Servos

We have already mentioned the surge current requirements of RC servos. One common problem with microprocessors and heavy switched loads, like servos, is that the sudden high current draws can cause short power brownouts. These short, but severe power fluctuations can reset the controller unless proper steps are taken. The power supply below shows a very capable regulator circuit that can prevent microprocessor reset. A capacitor stores sufficient charge to power the processor for a short duration while the input power is pulled down. The series diode prevents the capacitors energy from being used by the load. This simple circuit can virtually eliminate the brownout reset problem. However, if the primary power supply is not capable of handling the loads placed on it by the servo motors, you will probably see little jitters in all the servos every time one servo motor starts. This problem can only be solved by using a power supply with lower series impedance, or by placing sufficiently large capacitors on the power supply to bolster the supply during intermittent heavy loads.



Concluding Remarks

Using the servos() function is a very effective means of controlling multiple RC servos on a TICkit. This opens up use for robotics and other mechanical control applications. Also, TICkit ICs can be used as slaves for other controllers in which case they serve as intelligent RC control nodes.