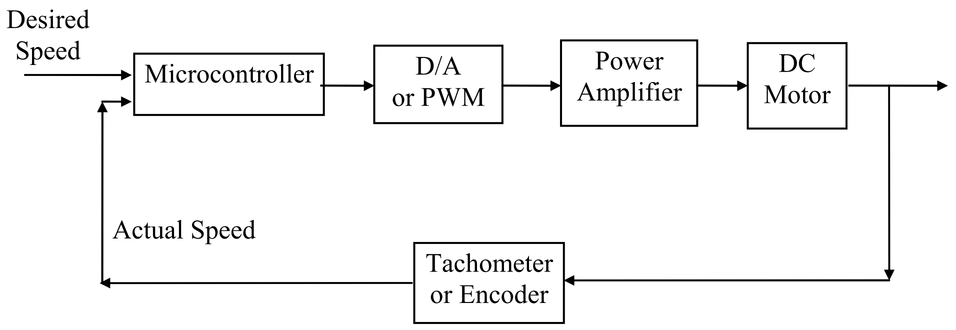
Solutions to Homework Problems
Contact: Dr. J. Yuan at 27667822

P1: Give an example in which a mechanical system is controlled by a microcontroller with the help of certain sensors, actuators, and interfacing circuits.

Solution:
An example is illustrated below, where a DC motor is controlled by a microcontroller to maintain a desired speed.



The motor speed is measured by a speed sensor (tachometer or an optical encoder). In the first case, the microcontroller needs an internal A/D converter to read in the analog signal that represents the instantaneous speed of the motor; whereas in the second case, the microcontroller needs an internal timer/counter to count the encode pulses and get the instantaneous speed of the motor. The motor speed is then compared with the desired speed that is read in from an internal parallel port. The microcontroller will generate a PWM signal or send a digital signal via a D/A converter to regulate the motor speed.

P2: The system clock of an at89c51 chip is 11,059,200Hz. An application program needs an interrupt signal of 1000Hz. Which timer mode may be selected and what should be the respective initial timer value preloaded into the timer?

Solution:
The application requires an interrupt at every 1ms, during which period a timer would count 110592/120=921.6 pulses. Since this count exceeds $2^8=256$, a timer has to be set to either 13-bit (mode 0) or 16-bit (mode 1) operation. For mode 0, the preload value should be initial_timer_value=$2^{13}-110592/120=7270$, whereas for mode 1, it is calculated by initial_timer_value=$2^{16}-110592/120=64614$.

P3: An at89c51 is programmed to generate a sequence of pulses to control a stepper motor. It reads from parallel port 0 an 8-bit integer number that represents the human instruction for the speed of the stepper motor. If the number read from port 0 is $0 \le N \le 255$, then the at89c51 must generate a sequence of output pulses to P1_0 whose frequency is $2(N+1)$ Hz.

(a) Determine the timer mode and reload value for each value of *N*, assuming that the system clock of the at89c51 is 12MHz exactly.
(b) What is the minimum frequency of the pulses under the given conditions?
(c) Can the microcontroller generate pulses with the correct frequency for all values of *N*?

Solution:

Since the system clock is 12MHz, every count by the timer lasts exactly one micro second. For all modes of operation, the reload value must be equal to or larger than zero. Since the interrupt rate is twice of the pulse frequency, one must calculate the reload value by $2^8 - 10^6/(4N+4) \ge 0$ or $2^8 \ge 10^6/(4N+4)$. The corresponding value of *N* should be $N \ge (10^6/2^{10}) - 1 \ge 976$. For mode 0, the timer has 13-bits and the reload value must also be positive. It means $2^{13} - 10^6/(4N+4)) \ge 0$, or $N \ge (10^6/2^{15}) - 1 \ge 30$. By the same token, one could calculate the value of *N* for mode 1, which is $N \ge (10^6/2^{18}) - 1 \ge 3$.

(a) If $N \ge 61$, then mode 0 may be set and the reload value should be $2^{13} - 10^6/(4N+4)$. Otherwise, if $61 > N \ge 7$, then mode 1 should be set and the reload value should be $2^{16} - 10^6/(4N+4)$. Mode 2 is not applicable to this situation.
(b) The minimum frequency of the interrupt is obtained by using mode 1 (16-bit) and a zero reload-value. The corresponding frequency of the interrupt signal is $10^6/2^{16} = 15.26$ Hz if the system clock rate is 12MHz exactly. The pulse frequency is half of that of the interrupt signal, with a minimum frequency of $10^6/2^{17} = 7.6294$ Hz.
(c) The input number is illegal if $N < 3$, and the microcontroller is unable to generate pulses with such a low frequency. In general the frequency of the interrupt alarms can only approximate the required frequency prescribed by *N*.

P4: An at89c51 is used to interface a pair of ultrasonic transducers for non-contact measurement of distance between the sensor and a mechanical target. The at89c51 uses bit 0 of port 1 to output "0". This signal triggers the ultrasonic transmitter to generate the ultrasonic signal. When the ultrasonic signal reaches the target and bounces back, the ultrasonic receiver will write a "0" to at89c51 at bit 0 of port 2. The system clock of the at89c51 is assumed to be 12MHz exactly, and timer 0 is set to mode 2 to count the flying-time.

(a) If the speed of sound is 340 m/s, what is the maximum range of the ultrasonic sensor?
(b) What should be the reload value of the timer?
(c) Write a C statement so that the at89c51 can detect the arrival of the ultrasonic echo signal (An input bit is assumed to be "1" unless a "0" is written to it).

(d) Ignoring the settings of the timer, list the necessary actions of the at89c51 in correct order, such that the ultrasonic sensor is controlled to take one sample of the distance measurement.

Solution:

(a) The maximum range is directly proportional to the maximum measurement of the flying-time. If a timer is set to mode 2, the maximum number of counts is 2^8, and the maximum flying-time is $2^8/10^6$=256 μs. When the speed of sound is 340 m/s, then the maximum range is $340\times100\times2^8/10^6$= 8.704 cm for a one-way path, or 4.352 cm for a two-way path. If a timer is set to mode 1, the maximum number of counts is 2^{16}, and the maximum flying-time is $2^{16}/10^6$=65.54 ms. When the speed of sound is 340 m/s, then the maximum range is $340\times2^{16}/10^6$=22.3 m for a one-way path or 11.15 m for a two-way path.

(b) When used to count the flying-time, timer 0 is not used as an interrupt (alarm) source. Its reload value should be zero. One may use timer 1 to generate an alarm signal for other timing operations, the reload value of timer 1 depends on the interrupt rate of timer 1.

(c) The simplest way to detect the arrival of the echo signal is a while-loop: "while(P2.0==1) {}:" The at89c51 will execute the next statement immediately after the arrival of the echo signal to bit 0 of port 2.

(d) The microcontroller should take the following actions: (1) pre-stop the ultrasonic transmitter by P1_0=1; (2) initialize the timer settings (details omitted); (3) start the ultrasonic transmitter by P1_0=0; (4) start timer 0 by TR0=1; (5) wait for P2_0=0; (6) stop timer 0 immediately after receiving the signal from bit 0 of port 2 by TR0=0; (7) stops the ultrasonic transmitter by P1_0=1; (8) calculates the time elapse between starting ultrasonic transmitter and receiving the input pulse (from bit 0 of port 2); (9) converts the time elapse to distance (Dist) assuming the speed of sound is 340m/s, display the distance value to port 0 by P0=~Dist;

P5: An economic way to control DC motors is by means of pulse-width modulation (PWM) signals. You are required to use an at89c51 to output a PWM signal to bit 0 of port 1 with the following specifications:

(1) the period of each PWM is 256 ms;
(2) the width of each PWM is $0\leq N\leq255$ ms;
(3) where N is an 8-bit integer input from port 0.

Modify the sample codes listed in Chapter 1 of the lecture notes to meet the above requirements.

Solution:

You should modify the reload value such that the interrupt signal has a frequency of 1000Hz. You should then modify the main function in the following way:

```
void main()
{
    char pWidth=0;
    timer0_initialize();                          // must initialize timer first
```

```
        while(1)
        {
                pWidth = P0;                    // read pulse width from port 0
                P1.0 = 1;                       // generate the rising edge
                timer0_delay(pWidth);           // keep high for pWidth
                P1.0 = 0;                       // generate the falling edge
                timer0_delay(256-pWidth);       // keep low for the rest of period
        } /* end while */
} /* end main */
```

P6: In the same application as described in **P5**, a tachometer is added to measure the speed of the DC motor. The voltage reading of the tachometer is converted into an 8-bit digital signal and written to port P2. At the same time, the expected tachometer digital output will be input from port P0. You are required to modify the codes of **P5** such that the system meets the following additional requirements:

(1) read from P0 and P2 regularly;
(2) compare the readings of P0 and P2;
(3) increase pWidth (which should be pre-set to zero) by 1 if the input of P2 is smaller than the input of P0 and explain why.
(4) decrease pWidth by 1 if the input of P2 is larger than the input of P0 (why?).
(5) keep pWidth unchanged if the inputs of P0 and P2 are the same.
(6) Check the resultant value of pWidth; force pWidth=255 if it is found larger than 255 or force pWidth=0 if it is found to be negative (why?).

Solution:

The sample codes have already set the timer and its ticker for you. All you need to do is to modify the main function in the following way:

```
void main()
{
    char pWidth=0, ref_s, act_s;
    timer0_initialize();                        // must initialize timer first
        while(1)
        {
                ref_s = P0;                     // read reference speed from P0
                act_s = P2;                     // read actual speed from P2
                if(ref_s > act_s) pWidth++;     // requirement 3
                if(ref_s < act_s) pWidth--;     // requirement 4
                if(pWidth > 255) pWidth=255;    // requirement 6
                if(pWidth < 0) pWidth=0;        // requirement 6
                P1.0 = 1;                       // generate the rising edge
                timer0_delay(pWidth);           // keep high for pWidth
                P1.0 = 0;                       // generate the falling edge
                timer0_delay(256-pWidth);       // keep low for the rest of period
        } /* end while */
} /* end main */
```

Explanations:

- For requirement (3), the DC motor speed is lower than expected and the at89c51 must increase the pulse width to speed up the DC motor.
- For requirement (4), the DC motor speed is higher than expected and the at89c51 must decrease the pulse width to slow down the DC motor.
- During the increase/decrease operation, there are possibilities that the value of pWidth may either exceed 255 or lower than zero. Requirement 6 is therefore imposed to keep the value of pWidth within the reasonable range.