

Chapter 18

Algorithms: Time

Measuring time with any 8051-family microcontroller can be very little burden on the processor. Since you have a crystal oscillator already running, it is just a matter of relating the an interval to the number of oscillator cycles. Unfortunately, the first approach taught to most students is to make a soft-

ware loop and poll the input until the time expires. Nothing else can happen while the software loop measures the time between pulses. Unless there is *absolutely* nothing else that needs to be done, you are better off measuring time with a real-time clock running off the timer. See “Timer Uses” on page 116.

Measuring Frequency or Period

Making measurements is best done with the timer functions of the 8051 family. The best approach varies depending on the expected period and the needed accuracy. Very fast events are best measured directly with a timer/counter or PCA, while very slow events are best measured with a real-time clock. There are also issues of jitter and frequency of obtaining readings.

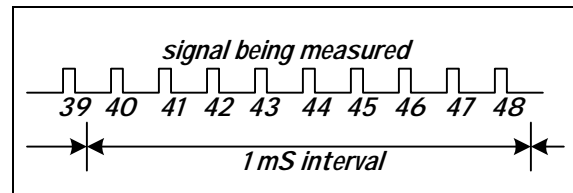


Figure 18.1: High Frequency Measurement by Counter and Real-time Clock

Fast Events and High Frequencies

These are the signals that are best handled by their own counter or capture register. As shown in Figure 18.1, the fast signals are counted in the hardware for a fixed time. Then that value in the counter is subtracted from the value in the counter just before the interval started. In Figure 18.1 the initial count would be 39 and the final count 48, so the software would determine it saw $48 - 39 = 9$ pulses per millisecond or a frequency of 9000Hz. The system must have a separate timer providing the fixed time interval, typically a real-time clock running off an interrupt. See the discussion of real-time clocks beginning on page 116.

```
#include <reg51.h>
unsigned int msecs; //global time counter
uint count;
void initialize(void){
    TMOD=0x51; //timer0, counter1in16-bit mode
    TH0--(1000/256); //initial load timer0
    TLO--(1000%256);
    IE=0x82; //unmask timer0 int & global interrupt
    TCON=0x50; //start timer0, counter1
}
void msecint(void) interrupt 1 using 1{
    uint static newcount, oldcount;
    TH0--(1000/256); //reload timer 0
    TLO--(1000%256);
    msecs++;
    newcount=TH1<<8|TL1;
    count=newcount-oldcount;
    oldcount=newcount;
}
void main(void){
    initialize();
    for(;;){
        //background stuff here
    }
}
```

Figure 18.2: Measuring Fast Events

Resolution From Figure 18.1 you can see that there is a resolution problem with such a slow signal or such a short interval—the readings would be to only the nearest 1000Hz. Using a 1mS interval, at a frequency of 100kHz, readings would have a 1% resolution. If a higher resolution is needed, let the counter accumulate for a longer time. Keeping the 1mS clock, if you subtracted readings every 100th interrupt (100mS), the 100kHz could build up to a count of 10,000 or a resolution at 100kHz of 0.01%. That may be unnecessary accuracy, but it *would* give

a 1% resolution for a 1kHz frequency. Notice that