EE461 Microprocessor-based Digital Design

IC16F874 Timer 1, Timer 2, and the Capture and Compare Modules

Assignment 15 Solutions

1. You are to make a PIC16F874 microprocessor generate an interrupt every 12.5 ms using Timer 2. The microprocessor will be driven by a crystal oscillator whose frequency is f_{OSC} = 20 MHz.

   If the desired outcome is possible, provide suitable settings for the Timer 2 module.

   If it is not possible, provide the best settings possible, calculate the resultant period between successive Timer 2 interrupts, and calculate the relative error.

   In either case, you may assume that whenever a Timer 2 interrupt occurs, the interrupt service routine resets the interrupt flag again in a timely fashion.

SOLUTION

Let x represent the rate at which the Timer 2 prescaler fires.

Let y = PR2 + 1 be the rate at which the TMR2 register gets reset.

Let z = T2OUTPS3:T2OUTPS0 + 1 be the rate at which the Timer 2 postscaler fires.

Then the period T_{TMR2} of Timer 2 is given by

$$T_{TMR2} = 4xyzT_{OSC}$$

$$xyz = 62\ 500 = 2^2 \times 5^6.$$

Now *x* can only take on the values 1, 4 = 2^2, or 16 = 2^4, so it cannot help in providing powers of 5: this must be done by *y* and *z.*

Furthermore, *z* can only take on values in the range from 1 to 16; of these, only 5, 10, and 15 will provide multiples of any powers of 5, and they will only provide one such power. This means that for an exact solution, *y* must provide a factor of 5^5 = 3 125 >> 256, the maximum feasible value of *y.*

It appears an approximation will be needed. If *z* and *x* are as big as they can be, they both are 16. In this case, $y = 62500/(16^2) = 244.1 = 244$. If *z* is only 15, then $y = 62500/(16 \times 15) = 260.4$: this is too big to fit into *y,* so we conclude that *z* and *x* must both be 16 and *y* must be 244.

This lets us calculate the achievable period:

$$T_{TMR2} = \frac{4xyz}{f_{OSC}}$$

$$\frac{4 \times 16 \times 244 \times 16}{20\ \text{MHz}} =$$

$$= 12.49\ \text{ms}.$$

The relative error in Timer 2's interrupt period is

= (Tacheived – Tdesired) / Tdesired = 12.49ms – 12.5ms / 12.5 ms = -0.576

- PR2 = *y* - 1 = 243,
- T2OUTPS3:T2OUTPS0 = *z* - 1 = 15, and
- T2CKPS1 = 1 (T2CKPS0 does not matter.)
- TMR2IE = 1 (enable Timer 2 interrupts)
- TMR2IF =0 (clear Timer 2 interrupt flag initially)

PEIE = 1 (enable peripheral interrupts)

GIE = 1 (enable global interrupts)

2. You are to make a PIC 16F874 microprocessor generate an interrupt every 100 ms using Timer 1 and one of the Compare modules. The microprocessor will be driven by a crystal oscillator whose frequency is f_{OSC} = 20 MHz. Timer 1 is to be driven by the internal clock.

   If the desired outcome is possible, provide suitable settings for the Timer 1 module.

   If it is not possible, provide the best settings possible, calculate the resultant period between successive Timer 1 interrupts, and calculate the relative error.

   In either case, you may assume that whenever a Timer 1 interrupt occurs, the interrupt service routine resets the interrupt flag again in a timely fashion.

SOLUTION

We can use the CCP1 or CCP2 module with the special event trigger. This will clear Timer 1 at regular intervals.

Let *u* be the number of internal clock ticks before the Timer 1 prescaler fires. It can take on values of 1, 2, 4, or 8.

Let *v* be the 16-bit value in the combined registers CCPR1H:CCPR1L. The period between successive interrupts is given by

$$Tccp = 4 * Tosc * u * v$$

$$u * v = 500000.$$

We can compute a table of possible pairs for *u* and *v:*

u	*v*
1	500000
2	250000
4	125000
8	62500

Of these, only the list is feasible because the other values of *v* are too big to fit within a 16-bit register. In hexadecimal, 62500 = F424.

We must make the following settings, then:

- T1OSCEN =0 (the oscillator is not needed)
- T1CKPS1:T1CKPS0 = 11 (1:8 prescale value)
- ~~T1SYNC~~ does not matter
- TMR1CS =0 (use the internal clock)
- TMR1ON =1 (turns Timer 1 on)
- TMR1H:TMR1L = F424
- CCP1M3:CCP1M0 = 1011 (compare mode, module 1, trigger the special event to clear Timer 1 on every interrupt)
- CCP1IE = 1 (enable CCP1 interrupts)
- CCP1IF =0 (clear CCP1 interrupt flag initially)
- PEIE = 1 (enable peripheral interrupts)
- GIE = 1 (enable global interrupts)