EEE 447 MICROPROCESSOR LABORATORY

EXPERIMENT 5 - TIMERS

Objective: In this experiment, it is aimed to learn Timer programming for PIC18F452 in C language.

Theory

The PIC18F452 has 4 timers referred to as Timer 0,1,2,3 and 4. They can be used either as timers to create a time delay or as counters to count events happening outside the microcontroller. Every timer needs a clock pulse to tick. The clock source can be internal or external. If we use the internal clock source, then 1/4th of the frequency of the crystal oscillator on the OSC1 and OSC2 pins (Fosc/4) is fed into the timer. By choosing the external clock option, we feed pulses through one of the PIC18F452's pins: this is called a counter. In this experiment we discuss the PIC18 timer.

Many of the PIC18 timers are 16 bits wide. Because the PIC18 has an 8-bit architecture, each 16-bit timer is accessed as two separate registers of low byte (TMRxL) and high byte (TMRxH). Each timer also has the TCON(timer control) register for setting modes of operation.

Timer0 can operate as a timer or as a counter.

Timer mode is selected by clearing the T0CS bit. In Timer mode, the Timer0 module will increment every instruction cycle (without prescaler). If the TMR0L register is written, the increment is inhibited for the following two instruction cycles. The user can work around this by writing an adjusted value to the TMR0L register.

Counter mode is selected by setting the T0CS bit. In Counter mode, Timer0 will increment, either on every rising or falling edge of pin RA4/T0CKI. The incrementing edge is determined by the Timer0 Source Edge Select bit (T0SE). Clearing the T0SE bit selects the rising edge.

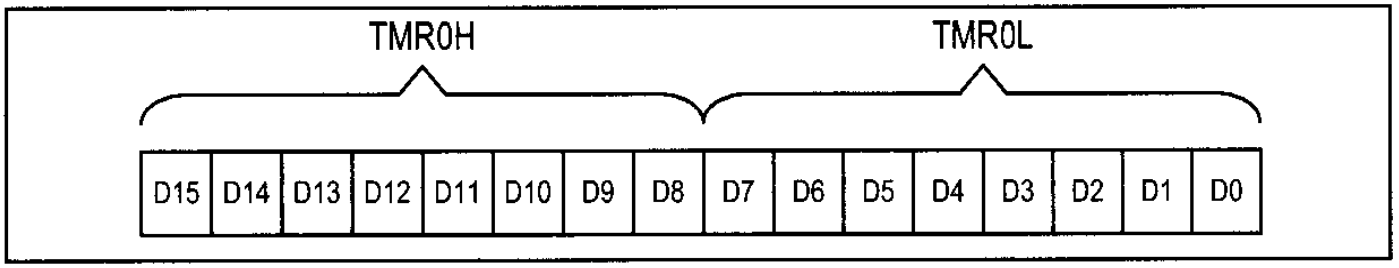

TABLE 10-1: REGISTERS ASSOCIATED WITH TIMER0

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Value on All Other RESETS
TMR0L	Timer0 Module Low Byte Register								xxxx xxxx	uuuu uuuu
TMR0H	Timer0 Module High Byte Register								0000 0000	0000 0000
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	0000 000x	0000 000u
T0CON	TMR0ON	T08BIT	T0CS	T0SE	PSA	T0PS2	T0PS1	T0PS0	1111 1111	1111 1111
TRISA	—	PORTA Data Direction Register							-111 1111	-111 1111

Legend: x = unknown, u = unchanged, - = unimplemented locations read as '0'. Shaded cells are not used by Timer0.

REGISTER 10-1: T0CON: TIMER0 CONTROL REGISTER

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
TMR0ON	T08BIT	T0CS	T0SE	PSA	T0PS2	T0PS1	T0PS0
bit 7							bit 0

bit 7 **TMR0ON:** Timer0 On/Off Control bit
1 = Enables Timer0
0 = Stops Timer0

bit 6 **T08BIT:** Timer0 8-bit/16-bit Control bit
1 = Timer0 is configured as an 8-bit timer/counter
0 = Timer0 is configured as a 16-bit timer/counter

bit 5 **T0CS:** Timer0 Clock Source Select bit
1 = Transition on T0CKI pin
0 = Internal instruction cycle clock (CLKO)

bit 4 **T0SE:** Timer0 Source Edge Select bit
1 = Increment on high-to-low transition on T0CKI pin
0 = Increment on low-to-high transition on T0CKI pin

bit 3 **PSA:** Timer0 Prescaler Assignment bit
1 = TImer0 prescaler is NOT assigned. Timer0 clock input bypasses prescaler.
0 = Timer0 prescaler is assigned. Timer0 clock input comes from prescaler output.

bit 2-0 **T0PS2:T0PS0:** Timer0 Prescaler Select bits
111 = 1:256 prescale value
110 = 1:128 prescale value
101 = 1:64 prescale value
100 = 1:32 prescale value
011 = 1:16 prescale value
010 = 1:8 prescale value
001 = 1:4 prescale value
000 = 1:2 prescale value

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

Step to program Timer0 in 16-bit mode

1. Load the value into the T0CON register indicating which mode (8-bit or 16-bit mode) is to be used and the selected prescaler option.

2. Load register TMR0H followed by register TMR0L with initial count values.

3. Start the timer

4. Keep monitoring the timer flag (TMR0IF) to see if it is raised. Get out of the loop when TMR0IF becomes high.

5. Stop the timer.

6. Clear the TMR0IF flag for the next round.

7. Go back to step 2 to load TMR0H and TMR0L again.

Experimental Work

Ex.1: Write a C program to toggle all the bits of PORTC continuously every 1s. Use Timer0, 16 bit mode, and the 1:256 prescaler to create the delay.

Ex.2: Write a C program to toggle only the PORTC.3 bit continuously every 4s . Use Timer0, 16-bit mode, the 1:256 prescaler to create the delay.