How to use internal timer registers of 8051(89c51,89c52) microcontroller

8051(89c51,89c52) series microcontrollers have two built-in timers Timer-0 and Timer-1. You can use them as event counters, generating Time-delays, generating Baud Rates for serial communication or as an interrupt counter. Both the timers 0 and 1 are 16-bit wide. Since the timers are 16-bit wide, the maximum range they can count is \(2^{16}\) (65535 times) after this value timers overflows. Timers actually counts the cycles that are generated by the crystal oscillator connected on pin 18(XTAL1) & 19(XTAL2). To understand the working of timer's you should first know about the registers that are associated with timers.

**Timer Registers**

- **TCON** (Timer Control register)
- **TMOD** (Timer Mode register)
- **TH0/TL0** (Timer 0, 16-bit register, High bits goes to TH0, Low bits goes to TL0)
- **TH1/TL1** (Timer 1, 16-bit register, High bits goes to TH1, Low bits goes to TL1)

### THO/TL0--------TH1/TL1

These two registers TH and TL are timer high byte and timer low byte, 0 and 1 are the timers numbers. These are 16-bit registers. We load our time delays in these registers. Since 8051(89c51,89c52) is an 8-bit microcontroller, we access these registers in two bytes, one byte for TH (timer high byte) and TL (timer low byte). TH and TL together makes 16-bits (TH 8-bit, TL 8-bit). TH0 and TL0 are byte addressable only.

**TCON (Timer control) Register**

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Bit Address</th>
<th>Explanation of Function</th>
<th>Timer</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>TF1</td>
<td>$S7h$</td>
<td>Timer 1 Overflow. This bit is set by the microcontroller when Timer 1 overflows.</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>TR1</td>
<td>$S6h$</td>
<td>Timer 1 Run. When this bit is set Timer 1 is turned on. When this bit is clear Timer 1 is off.</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>TF0</td>
<td>$S5h$</td>
<td>Timer 0 Overflow. This bit is set by the microcontroller when Timer 0 overflows.</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>TR0</td>
<td>$S4h$</td>
<td>Timer 0 Run. When this bit is set Timer 0 is turned on. When this bit is clear Timer 0 is off.</td>
<td>0</td>
</tr>
</tbody>
</table>

lower 4 bytes are IE0/IE1 & IT0/IT1

88h is SFR of TCON register

- **TR0/TR1**: Timer 0/1 (Run control flag)
- **TF0/TF1**: Timer 0/1 (Timer overflow flag)
Lower Four bits are not shown in the above figure. These bits are
- IT0/IT1: Timer Interrupts. When IT0=1 or IT1=1 it specifies interrupt on falling edge and when IT0=0 or IT1=0 it specifies interrupt on rising edge.
- IE0/IE1: Used for external Interrupts.

**Description**

TCON is an 8-bit register. Its bits are used for generating interrupts internal or external. The most important bits of the timers, TRx and TFx are also in it. TRx(timer run) and TFx(timer overflow) bits which we use in almost all our timer applications are in it. When we initialize TRx with 1, TRx=1 it means that timer is running. When the specified time is over the timer it self make TFx=1 which means that the delay value is reached. Once TFx=1 stop the timer by initializing TRx with 0 TRx=0(Stop Timer). Know if we again want to run the timer make TRx=1. In the diagram you can see the SFR for TCON register, the bit's used for interrupt handling and the timer run and timer over flow bits. If you want to access the individual bits of the registers, you can access them by their names. You can also access whole register it self by its name. At the bottom of the page their is a small example explaining it.

**TCON (Timer control)**

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Explanation of Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>GATE0</td>
<td>When this bit is set the timer will run when INT0 is high. When this bit is clear the timer will run regardless of the state of INT0.</td>
</tr>
<tr>
<td>6</td>
<td>C/T0</td>
<td>When this bit is set the timer will count events on TF0 and TF1. When this bit is clear the timer will be incremented every machine cycle.</td>
</tr>
<tr>
<td>5</td>
<td>TF0</td>
<td>Timer mode bit 0 (see below)</td>
</tr>
<tr>
<td>4</td>
<td>TF1</td>
<td>Timer mode bit 1 (see below)</td>
</tr>
<tr>
<td>3</td>
<td>Gate1</td>
<td>When Gate is 1, Timer runs only when INT0-INT1 Pin of 89c51 is high.</td>
</tr>
<tr>
<td>2</td>
<td>C/T1</td>
<td>Selects Timer as External Counter or Internal Timer. If C/T is 1 Timer as a Counter is selected if 0 Timer as a delay generator is selected.</td>
</tr>
<tr>
<td>1</td>
<td>Gate0</td>
<td>When Gate is 1, Timer runs only when INT0-INT1 Pin of 89c51 is high.</td>
</tr>
</tbody>
</table>

**TMOD (Timer mode)**

- T1M0-T1M1----T0M1-T0M0: Selects the mode of the Timer.
- C/T: Selects Timer as External Counter or Internal Timer. If C/T is 1 Timer as a Counter is selected if 0 Timer as a delay generator is selected.
- Gate0-Gate1: When Gate is 1, Timer runs only when INT0-INT1 Pin of 89c51 is high.

**Description**

TMOD is an eight bit register, it is bit addressable. Higher four bits (4 to 7) are related to Timer 1 where as the lower four bits (0 to 3) perform the exact same functions, but for timer 0. You can see their functions in the figure above.

**SFR's for timer registers**

<table>
<thead>
<tr>
<th>SFR Name</th>
<th>Description</th>
<th>SFR Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>TH0</td>
<td>Timer 0 High Byte</td>
<td>8CH</td>
</tr>
<tr>
<td>TL0</td>
<td>Timer 0 Low Byte</td>
<td>8DH</td>
</tr>
<tr>
<td>TH1</td>
<td>Timer 1 High Byte</td>
<td>8EH</td>
</tr>
<tr>
<td>TL1</td>
<td>Timer 1 Low Byte</td>
<td>8FH</td>
</tr>
<tr>
<td>TCON</td>
<td>Timer Control</td>
<td>8BH</td>
</tr>
<tr>
<td>TMOD</td>
<td>Timer Mode</td>
<td>88H</td>
</tr>
</tbody>
</table>

8051 Timers sfrs
Note we can access the registers by their names as well which make the programming easy and I am going to use them by their names and not sfr’s.

How to calculate values for TH and TL registers?

Timer Clock Cycle Duration = 6/oscillator frequency

Timer clock cycle duration is the time in which one cycle is provided by the oscillator, and in a second thousand ’s of cycle’s are provided by the oscillator. Divide the delay needed by the Timer clock cycle duration. For example I need a delay of 20ms, Divide 20ms by Timer Clock Cycle duration. Now negate the result of previous calculation from 65535, which is the maximum count by the timer with out overflow. Now transform the obtained decimal value in hexadecimal, and load the two high bytes in TH and low bytes in TL.

Example 1

Calculation of Timer 0 reload value needed to achieve timer delay of 20 ms. Oscillator frequency is 11.0592 MHz.

\[
\text{Delay Value} = \frac{\text{Timer Delay} \times 10^{-3}}{\text{Timer Clock Cycle Duration}} \\
= \frac{20 \times 10^{-3}}{6} \\
= \frac{11.0592 \times 10^5}{6} \\
= 30864 \text{ (must be rounded to the nearest Integer)}
\]

\[
\text{Timer Reload Value} = \text{Maximum Register Count} - \text{Delay Value} \\
= 65535 - 30864 \\
= 28671 \\
= 0x6FFF
\]

so Timer 0 is loaded with:

\[
\text{TH0} = 0x6F; \\
\text{TL0} = 0xFF;
\]

Calculating Delay for 8051 (89c51,89c52) microcontroller

8051 Microcontroller Delay generation Steps

1. Select Timer Mode in TMOD Register. Specify which Timer you are using.
2. Load values for TLx and THx registers.
3. Run the timer by initializing TRx with 1 TR=1.
4. Keep monitoring the timer flag (TFx). If it is raised Stop the timer by making TRx=0;
5. Clear the TFx flag for the next count.

Simple Delay function

```c
void Delay(void)
{
    TMOD = 0x01;  //Timer 0 is in use. 16-bit Timer Mode is selected.
    TL0 = 0x6F;   //Load value for TLx register
    TH0 = 0xFF;   //Load value for THx register
    TR0 = 1;      //Run Timer-0
    while(!TF0)   //Poll TFx
    TR0 = 0;      //If TF=1 stop the timer by making TR=0
    TF0 = 0;      //Make TF=0 for next counting
}
```
Some Projects Created using same Delay method shown above

Sep 8: Obama Urges Homeowners To Switch To A 15 Year Fixed
If you owe less than $625,000 on your home, use the President’s Refi Program. You’ll be shocked when you see how much you can save.

Select Your Age:

<table>
<thead>
<tr>
<th>Age</th>
<th>20</th>
<th>30</th>
<th>40</th>
<th>50</th>
<th>60</th>
<th>70</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>21</td>
<td>31</td>
<td>41</td>
<td>51</td>
<td>61</td>
<td>71</td>
</tr>
<tr>
<td>21</td>
<td>32</td>
<td>42</td>
<td>52</td>
<td>62</td>
<td>72</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>33</td>
<td>43</td>
<td>53</td>
<td>63</td>
<td>73</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>34</td>
<td>44</td>
<td>54</td>
<td>64</td>
<td>74</td>
<td></td>
</tr>
</tbody>
</table>

Calculate New House Payment

Download PDF Document for 8051(89c51,89c52) Microcontroller Timers Registers
Emulators for 8051
phyton.com
Realtime non-Intrusive emulation of virtually all 8051 microcontrollers

8051 projects  C++ projects  Arduino projects  Knowledge based  About me
Xilinx

Copyright © 2014 microcontroller-project.com. All rights reserved.
Unless otherwise indicated, all materials on these pages are copyrighted by the microcontroller-project.com. All rights reserved. No part of these pages, either text or image may be used for any purpose other than personal use.