EXPERIMENTS WITH TIMER MODULE OF MSP430F149 (TI) FOR EDUCATIONAL PURPOSES

Ljudmila Rumenova Taneva

Technology School "Electronic Systems", Technical University, Sofia, Bulgaria, E-mail: ltaneva@elsys-bg.org

This paper describes the various parts of the Standard Timer Module of MCU and demonstrates some software for performing typical tasks: a simple counter; use of Timer in Output Compare (OC) mode to generate time intervals and PWM signals; using interrupts and Low-Power Modes; using Timer in Input Capture (IC) mode for detecting signals and using Watchdog Timer. The experiments give students knowledge and practice with the basic Timer functions. They have been developed for the needs of Technological School "Electronic Systems".

Keywords: Timer Module, PWM, Input Capture, Output Compare, lab exercises.

I. INTRODUCTION.

Lab experience is an important factor in teaching microcontrollers (MCU) and their subsystems. These laboratories will introduce students to the basic concepts of microcomputing. A microprocessor's systems span a vast range, from simple single-chip processors used to control appliances to 32-bit processors capable of outperforming a workstation PC. The fundamental ideas behind all of these units are about the same. Thus, by learning one microprocessor, the students will be able to move to other units with relative ease. The tool used in these experiments is SPS430 evaluation board. It is a self-contained unit needing a PC for normal operation - for editing, assembling and downloading the programs. The SPS430 board uses the MSP430F149, a 16-bit microcontroller, as main processor unit.

The Timer System is one of the most important subsystems of MCU's and is present in all of them. The other subsystems use the timer very often in their work. The Timer system could include other devices with timer functions – Watchdog Timer, PWM module, Pulse Accumulator. The educational strategy should be performed by answering some standard questions:

1) Why the study of Timer module is necessary?
2) What is to be learned?
3) How the Timer should be studied?

The answer to the first question is connected with the Timer Module importance and its use in a wide area of applications. The purpose of the timer module is to allow for time critical operations to be handled by the hardware, instead of trying to accomplish everything in software. For example, generating wave forms or measuring waveforms is fairly straightforward using the timer module. The Standard Timer Modules functions mostly involve doing things based on the current value of the programmable timer [1]. When, for example, an 'output compare' occurs, the hardware will automatically change the state of an output pin. An output compare means that the current value of the timer matches a trigger value set by the software. For another example, when an 'input capture' occurs, the current value of the timer is stored in a special register. The input capture triggers when the state of one of the input pins changes in a specified way. This allows you to capture the exact time of some external event.

The Timer modules of MCU's include one or more independent timers. They have similar functions although some of them could be very different. The basic functions are:

1) INPUT CAPTURE (IC) is the ability to capture the timer value when an external event occurs. Applications of Timer Input Capture include:

- counting pulses from external sources;
- measuring the parameters of input signals (wave form analysis) (fig.1);

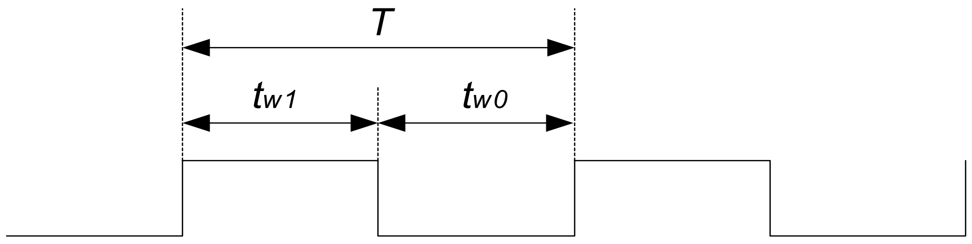

Fig.1. Using IC function for measuring the parameters of input signals.

To measure a period *T*, two successive edges of the same polarity have to be captured. To measure a pulse width *tw,* two alternate polarity edges have to be captured. For example, to measure *tw1* for a high-going pulse, the student would capture at a rising edge and subtract this time from the time captured for the subsequent falling edge. **T**he ration *tw1/T* is the duty cycle of the signal.

2) OUTPUT COMPARE (OC) is used to program an action to occur at a specific time – when 16-bit counter reaches a specific value. The value of the Capture/Compare Register is compared to the value of the free-running counter on every bus cycle. When the comparison occurs, a programmed event takes place, such as changing the state of a pin, or generating an interrupt (fig.2). The function OC can be used for:

- generating of time intervals;
- generating a pulse or sequence of pulses with programmable parameters;
- PWM signals (fig.1);
- generating interrupts;
- programmable delays;

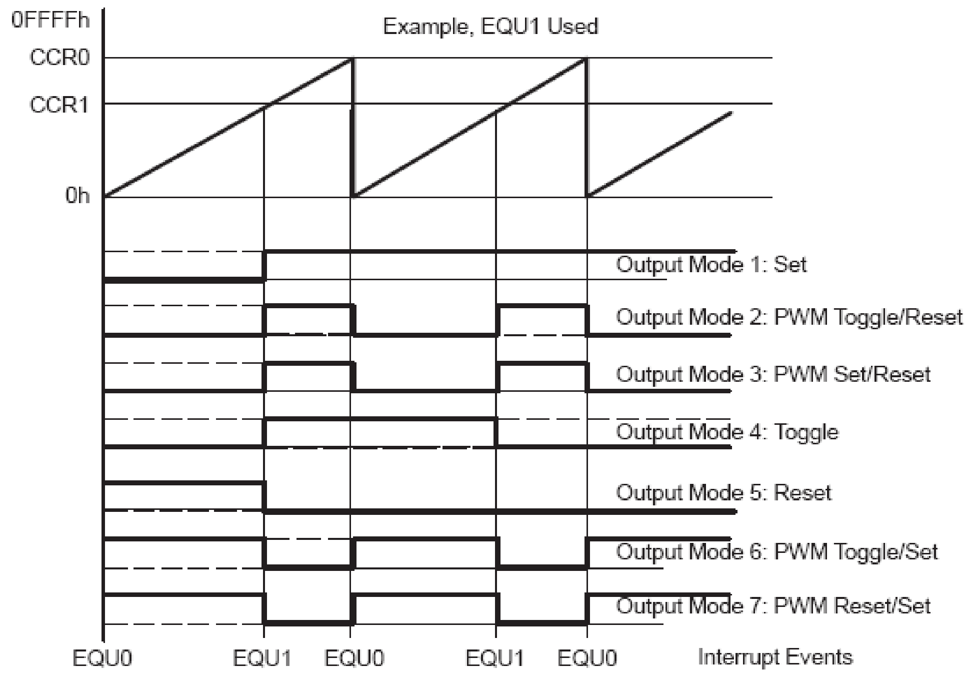

Fig.2. Using OC function for generating output waveforms and PWM signals;

The second question is connected with teaching the basic timer functions most typically applied. Actually they define the structure of the lab exercises. Beneath are examined the most typical timer applications:

- generating periodic signals with programmable parameters:

1) by program delay;
2) by timer overflow;
3) by OUTPUT COMPARE (OC) function;
4) by Pulse Width Modulation (PWM);

- a simple counter, using interrupt after overflow;

- real time clock (RTC);

- using interrupts and low power modes (LPM):

Polling pins or testing for events wastes CPU cycles and power. In this case the CPU runs normally in LPM, and interrupts by the subroutine Timer_ISR at a programmed interval. The Timer_ISR wakes up the CPU by changing the operating mode which forces the CPU to exit LPM and return to active mode. Then the mode could be changed again.

- Pulse Width Modulation of motor drive units.

One of the more commonly used digital methods for motor control is pulse-width modulation (PWM). A simple stepper motor may only require a general purpose timer. However, dc motors and brushless dc motors require a pulse width modulator (PWM) timer to control the speed of the motor. Specifically, the PWM method utilizes the microcontroller's ability to generate square waves to produce small variations in the voltage of a circuit [6].

- Generating a dead time. The up/down mode of the MSP430F149 supports applications that require dead times [1] between output signals (fig.3). If two outputs must never be in a high state simultaneously, the t_{dead} is:

$$t_{dead} = t_{timer} \times (CCR1 - CCR3) \qquad (1)$$

with:

t_{dead}: Time during which both outputs need to be inactive

t_{timer}: Cycle time of the timer clock

CCRx: Content of capture/ compare register x

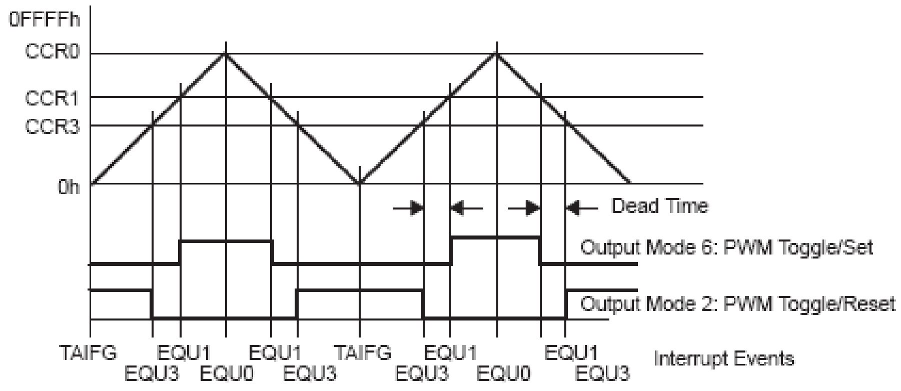

Fig. 3. Realizing Dead time in Up/Down mode.

The third question includes both didactic methodology and educational software.

II. Developing lab exercises with TIMER MODULE of MSP430F149 (TI).

2.1. TIMER MODULE of MSP430F149.

TI microcontroller MSP430 has a 16-bit RISC architecture, 16 integrated registers on the CPU, built-in hardware multiplication, communication capability using asynchronous (UART) and synchronous protocols, 12 bit analog-to-digital converter, two timers, Watchdog timer, 60kB Flash memory and 2kB RAM.

A digital –controlled oscillator, together with the frequency lock-loop, provides a fast wake up from the low power mode to the active mode in less than 6µs.

The MSP430F149 Standard Timer Module consists of two 16-bit programmable timers – Timer_A and Timer_B and Watchdog timer (WDT). Timer_A has 16-bit counter with 4 operating modes (Up, Up/Down, Continuous and Stop mode), selectable clock source, three independently configurable capture/compare registers with configurable inputs, three individually configurable output modules with 8 output modes. Each capture/compare register has hardware support for implementing serial communications (UART protocol). Timer_B is almost identical to Timer_A except for a few differences: the length of Timer_B is programmable to be 8, 10, 12, or 16 bits; Timer_B has seven capture/compare registers and hasn't implemented UART protocol.

Watchdog timer (WDT) is a special module whose primary function is to perform a controlled-system restart after a software problem occurs. If the WDT function is not needed in an application, the module can work as an interval timer, to generate an interrupt after the selected time interval [2][3].

2.2. EXPERIMENTS FOR TIMER MODULE OF MSP430F149 (TI).

1) A Counter (Program Timer Module.c).

Timer_A runs in UP mode and configured for OC . Four-digit BCD counters are provided and they increment after timer interrupt (every 1ms). A single BCD digit counts from 0-9 and all digits are displayed on the indication every 50 ms. The maximum value is 20 000 (20s).

The Up mode is used if the timer period must be different from 65 536 clock cycles. The capture/ compare register CCR0 data define the timer period. The counter counts up to the content of CCR0. When the timer value and the value of the compare register CCR0 are equal, the timer restarts counting from zero.

2) Generating Time Intervals in Continuous Mode (Program Intervals.c)

This program uses Timer_A CCRx units and overflow to generate four independent timer intervals. Timer_A runs in Continuous mode and configured for OC (OUTMOD_4). For demonstration, CCR0, CCR1 and CCR2 output units are optionally selected with port pins P5.0, P5.1 and P5.2 in toggle mode. As such, these pins will toggle when respective CCRx registers match the TAR counter. Interrupts are also enabled with all CCRx units, software loads offset to next interval only - as long as the interval offset is added to CCRx, toggle rate is generated in hardware.

3) Measuring the parameters of input signals (program Detect.C)**.**

This experiment introduces the interrupt driven input capture feature of the MSP430x development boards. Timer_A runs in Continuous mode and P1.1 (TA0) is configured for IC. There is a connection between P1.1 (TA0) and P2.0/ACLK. CPU is normally off and used only during TA_ISR when Input Capture of ACLK occurs. RAM array consist the current value of the timer for 16 captures. Using scope the data could be compared with the parameters of the signal. The program that finds the period, the width and the duty cycle of an unknown pulse could be written.

4) Pulse Width Modulation and DC Motor Control (Program PWM1.c)**.**

This experiment demonstrates control of the DC motor speed with generating PWM signals. Timer_A runs in Up mode and P1.2/1.3 are configured for OC. Two PWM signals are generated in P1.2/1.3 using output mode 7. There is DC motor connected to the P1.2 and this output is used to control the direct current motor. The speed of rotation depends on the ratio of the PWM signal duty cycle. The content of CCR0 defines the period (512) and the contents of CCR1, CCR2 define the ratio of the PWM signal's duty cycle (CCR1 = 384 - 75% PWM, CCR2 = 128 - 25% PWM) (fig.6). Increasing the ratio of duty cycle we have to obtain as a result higher speed of rotation of the DC motor. The students could find out the connection (2) between frequency of DC motor rotation *n* and DC current *Ia*:

$$n = f(Ia) \qquad (2)$$

5) Using Watchdog Timer.

Another possible experiment involves performing tests using a Watchdog Timer. A Watchdog Timer is a safe guard available in many microcontrollers. It is designed to prevent uncontrolled infinite loops or suspended operation of the CPU. It is an interrupt mechanism that requires the microcontroller program to periodically check-in with the "Watchdog". If a program fails to check in after a certain amount of time the Watchdog Timer generates a non-maskable interrupt that forces the CPU to start executing a special recovery routine.

Program WDT1.c demonstrates the operation of a Watchdog Timer. The program performs some visible tasks with LED's, like "running zero" on port 5, and then initiate the Watchdog Timer. The program fail to check-in and the result is uninterrupted lightening of the LED's. The purpose of this experiment is to use WDT like in real practice. In laboratory program usually WDT is stopped.

Program WDT2.c demonstrates the operation of a Watchdog Timer as an interval timer. The CPU runs normally in Low Power Mode and is interrupted by the WDT at a programmable interval (30ms). The interrupt service routine Wakes up the CPU, which completes program by toggling two pins with connected LED and buzzer. The result is audible (with definite frequency) and visible (lightening LED). The source clock of WDT is DCO ~ 800kHz.

This mode gives application the possibility to use a third timer if Timer_A and Timer_B are not sufficient.

III. RESULTS.

Here are reported several experiments with proven software to provide practice with Timer Module of the MSP430 F149. All the source code for the experiments provided in this report can be completed with the IAR EW IDE. The programs are tested on the SPS430 demo board and could be performed on others with some corrections. The experiments with Timer Module of the MSP430 F149 are good teaching tools and they offer knowledge that cannot be learned at classical lectures. The students have the opportunity to gain experience in basic Timer functions.

The experiments could be upgraded with adding some components and the area of applications could be extended.

REFERENCES:

[1] Application Report SLAA120 -2000, TI, *PWM DC Motor Control Using Timer_A of the MSP430x.*

[2] www.ti.com

[3] www.motorola.com

[4] Computer Science 20/30/40. *Outline of Foundational and Learning Objectives.*

[5] www.silabs.com

[6] Kloth A., *PWM of DC Motor Drive Units.*