Lab 7: Timer

I. Introduction

In lab 3, we wrote a program to make an LED light blink. The LED light was turned on for a while, then turned off for a while. The effect of "for a while" was implemented by writing three nested loops doing nothing "nop". There is a more elegant way to do it – using a timer. The ATmega 2560 microcontroller includes 6 timers (two 8-bit and four 16-bit timer/counters):

- Four 16-bit timers (main focus of the lab)
  - Timer/Counter1, 3, 4, 5 are 16-bit Timer/Counter unit allows accurate program execution timing (event management), wave generation, and signal timing measurement. (P133 of the datasheet). [1]
- Two 8-bit timers
  - Timer/Counter0 is a general purpose 8-bit Timer/Counter module. It allows accurate program execution timing (event management) and wave generation. [1]
  - Timer/Counter2 is a general purpose, single channel, 8-bit Timer/Counter module.

We are going to use Timers 3, 4 and 5 in this lab. The 16-bit timer counts up or down between 0 and 65,535 (max), independent of the execution of a program. There are 65,536 (2^{16}) steps. Actually, the timer can be configured to count between 0 to a predetermined value up to 65,535, let's call it TOP, depends on how much time the user set. The counter is incremented by a user selectable clock source, in this lab, we use the same clock signal that is supplied to the CPU, 16MHz. The CPU clock is very fast, and it takes the timer 0.004 second to count from 0 to 65,535. In order to slow down the timer (note timer and counter are used interchangeably), the clock can be pre-scaled (divided) by selected powers of two. For example, we can set the pre-scaler to 1024, that is, every 1024 CPU clock cycles count as 1 in the timer. In this case, the timer counts from to 15,625 in one second. Let the TOP be a value between 0 and max. We can control the time interval by setting the proper TOP value, for example:

The CPU clock runs at 16MHz -> 16 * 10^6 CPU cycles/second, with a pre-scaler set at 1024, the timer clock cycle is:

16 * 10^6 / 1024 timer cycles per second = 15,625 timer cycles per second

If the desired time interval is 2 seconds, then we need to set the TOP value at:

TOP = 16 * 10^6 / 1024 * 2 -> round up to the closest integer as:

= (0.5 + 16 * 106 / 1024 * 2) <- take the integer part

= 31250

If the desired time interval is 0.5 second, then we need to set the TOP value at:

TOP = 16 * 10^6 / 1024 *0.5 -> round up to the closest integer as:

= (0.5 + 7812.5) <- take the integer part

= 7813

To summarize the calculations to a formula:

Let CLOCK be the CPU clock (16MHz)

PRESCALE_DIV be the value for pre-scaler
DELAY be the desired time interval in seconds
Then:
TOP=int(0.5+(CLOCK/PRESCALE_DIV*DELAY))

Since the highest number for a 16-bit timer/counter is 65,535, TOP should be between 0 and 65,535, therefore, we need to check if **TOP ≤ 65,535**

II. Configure Timer/Counter

The timers can be set to operate in different modes so that a right timer can be used in an application. The timer of the AVR can be specified to monitor several events. Some of the modes the timers can operate are:

a) Timer Overflow mode (normal mode)
b) **Compare Match** (We are going to use this mode in the lab.)
c) Input capture.

The summary of registers associated with Timer/Counter3 is listed below:

- The 16-bit timer counter value is loaded at: TCNT3H:TCNT3L. The counter will start counting up from an initial value loaded and count up to the value set in TOP. In the next clock pulse, the 16-bit counter will overflow and reset to 0x0000 and causes the OCF3A bit set in the TIFR3 register.
- The timer mode (CTC Clear Timer on Compare Match mode) is set by writing "0100" to WGM3(3:0) bits in TCCR3A and TCCR3B control registers.
- The timer can use an internal clock signal or an external signal as the source. In this lab, the timer is setup to use an internal clock. Note that the CPU is operating at 16Mhz (16 million cycles per second). We will use this clock but reduce the counting speed by setting up a pre-scaler of 1024. That is the clock signal is divided by 1024 to reduce the counting speed. This is done by using the TCCR3B control register.

III. Use Timer/Counter3 in AVR ATmega 2560 as Example

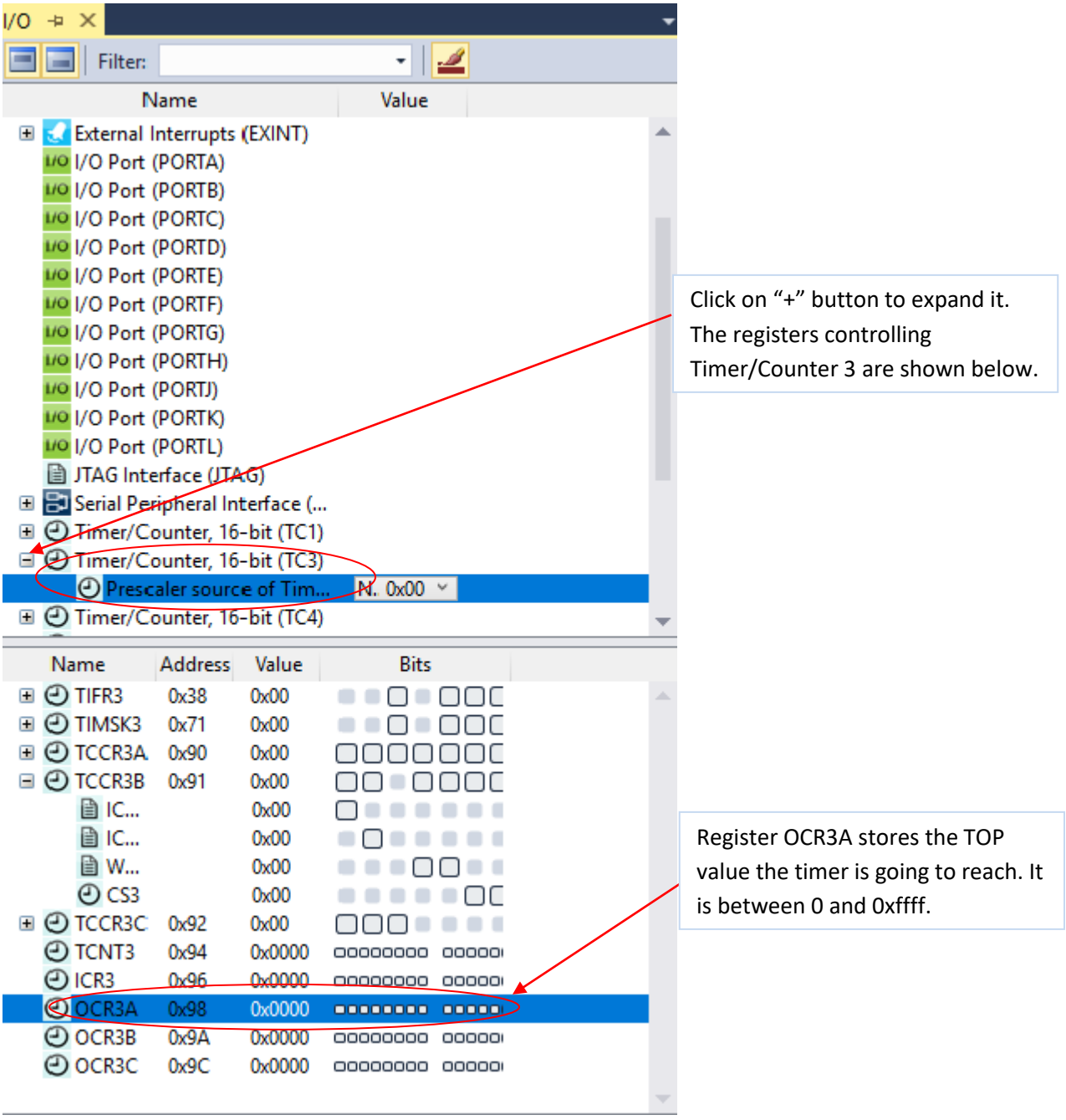
In this lab, the 16-bit Timer/Counters 3, 4 and 5 are used. The following is an example of Timer 3. The names and addresses of the registers associated with Timer 3 are:

.equ OCR3A = 0x98 ;**O**utput **C**ompare **R**egister of Timer/Counter**3**, channel **A (stores TOP)**
.equ TCCR3A = 0x90 ;**T**imer/**C**ounter3 **C**ontrol **R**egister **A**
.equ TCCR3B = 0x91 ;**T**imer/**C**ounter3 **C**ontrol **R**egister **B**
.equ TCCR3C = 0x92 ;**T**imer/**C**ounter3 **C**ontrol **R**egister **C (not used in this lab)**
.equ TCNT3H = 0x95 ; high byte of the Timer/Counter3
.equ TCNT3L = 0x94 ; low byte of the Timer/Counter3
.equ TIFR3 = 0x18(0x38) ;**T**imer/**C**ounter3 **I**nterrupt **F**lag **R**egister
.equ TIMSK3 = 0x71 ;**T**imer/**C**ounter3 **I**nterrupt **M**ask **R**egister **(not used in this lab)**

OCR3A - Output Compare Register of Timer/Counter3, channel A (page 159 of Datasheet[1])

It is a 16-bit register pair: OCR3AH:OCR3AL. The TOP value is stored in this register if the timer is configured to reach a number other than the maximum (0xFFFF). See page 160 of the datasheet[1] for more details. Refer to page 135 for the definition TOP.

To find out the memory addresses of the registers, go to the menu -> "Debug" -> "Start Debugging and Break", from the "I/O" panel, click on the "+" button on "Timer/Counter, 16-bit (TC3)":



The 16-bit register stores the TOP value that the timer is going to reach. The TOP value is between 0 and 0xffff (highest value for a 16 unsigned integer). Since the data bus is 8 bits, Special procedures must be followed when read/write to the 16-bit register. There is a single 8-bit register

for temporary storage of the high byte. Accessing the low byte triggers the 16-bit "read" or "write" operation. When the CPU writes to the 16-bit register, it must write the high byte first. For a 16-bit read, the low byte must be read first. In this lab, OCR3A is set at 7813, which means 0.5 second delay.

TCCR3A - Timer/Counter 3 Control Register A (page 154 of Datasheet[1])

Bit	7	6	5	4	3	2	1	0	
(0x90)	COM3A1	COM3A0	COM3B1	COM3B0	COM3C1	COM3C0	WGM31	WGM30	TCCR3A
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Set at 0b00. Means normal port operation. P155 of Datasheet[1]

Combine with WGM_33: WGM_32 in TCCR3B, set timer3 to CTC mode (non-PWM) mode.

TCCR3B - Timer/Counter3 Control Register B (page 156 of Datasheet[1])

Bit	7	6	5	4	3	2	1	0	
(0x91)	ICNC3	ICES3	–	WGM33	WGM32	CS32	CS31	CS30	TCCR3B
Read/Write	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Combine with two bits in TCCR3A, $WGM_33:0$ = 0b0100 set the timer mode of operation to CTC (Clear Timer on Compare Match mode)

The three clock select bits select the clock source to be used by timer3 and set the prescaler to slower down the counter.

WGM stands for Waveform Generation Mode.

In this lab, the $CS_32:0$ = 0b101, which means the CPU clock is used as source and the pre-scaler is set to 1024.

- **Bit 2:0 – CSn2:0: Clock Select**

The three clock select bits select the clock source to be used by the Timer/Counter, see Figure 17-10 and Figure 17-11 on page 152.

Table 17-6. Clock Select Bit Description

CSn2	CSn1	CSn0	Description
0	0	0	No clock source. (Timer/Counter stopped)
0	0	1	$clk_{I/O}$/1 (No prescaling
0	1	0	$clk_{I/O}$/8 (From prescaler)
0	1	1	$clk_{I/O}$/64 (From prescaler)
1	0	0	$clk_{I/O}$/256 (From prescaler)
1	0	1	$clk_{I/O}$/1024 (From prescaler)
1	1	0	External clock source on Tn pin. Clock on falling edge
1	1	1	External clock source on Tn pin. Clock on rising edge

If external pin modes are used for the Timer/Countern, transitions on the Tn pin will clock the counter even if the pin is configured as an output. This feature allows software control of the counting.

The least significant three bits of TCCR3B are used to slow down the timer in our example. Instead of counting 1 per CPU clock cycle, it counts 1 every 1024 CPU clock cycles in the example.

TCCR3C - Timer/Counter3 Control Register C

Bit	7	6	5	4	3	2	1	0	
(0x92)	FOC3A	FOC3B	FOC3C	–	–	–	–	–	TCCR3C
Read/Write	W	W	W	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

Not used in this example.

TIMSK3 – Timer/Counter3 Interrupt Mask Register

Bit	7	6	5	4	3	2	1	0	
(0x71)	–	–	ICIE3	–	OCIE3C	OCIE3B	OCIE3A	TOIE3	TIMSK3
Read/Write	R	R	R/W	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Not used in this example.

TIFR3 – Timer/Counter3 Interrupt Flag Register

Bit	7	6	5	4	3	2	1	0	
0x18 (0x38)	–	–	ICF3	–	OCF3C	OCF3B	OCF3A	TOV3	TIFR3
Read/Write	R	R	R/W	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Output Compare A Match Flag. It is set after the timer3 value matches the TOP (TCNT3 == OCR3A). In this lab, when OCF3A is 1, toggle LED, then clear bit OCF3A of TIFR3 by writing a logic one to its bit location (p163 of the Datasheet[1]).

IV. Exercises:

- Create project lab7.
- Download multi_timer.asm, and move it to the folder where main.asm is.
- Add multi_timer.asm to project lab7.
- Copy upload.bat from the previous labs to the Debug folder, modify upload.bat.
- Implement delay by timer4 and timer5 such that LED4 and LED5 blink every 1.5 and 3 seconds respectively.

V. References:

1. Section 17 (page 133) of the ATmega2560-datasheet.pdf at http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-2549-8-bit-AVR-Microcontroller-ATmega640-1280-1281-2560-2561_datasheet.pdf