timer:
- tcnt
- PWM.
- Output compare
- input capture.
- Interrupts (time)
  - timer overflows.
  - input capture
  - outcapture?.

Timer System and implementing Real-Time Interrupt on the ATmega2560

ME/ECE 4370/5370

Atmel ATmega2560 timer features:

- 2 -8bit, 4 – 16 bit timers.
- True 16-bit PWM
- Three independent output compare units per timer
- One input capture unit per timer
- Fast PWM, Phase-correct PWM
- Thirty independent interrupt sources (overflows, input capture, output compare),

Introduction:

The ATmega2560 timer system provides a series of clocks that can be used to time various events. These are contained in what Atmel calls its Timer/Counter system. ATmega has two 8-bit timers (Timer/Counter0, Timer/Counter2) and four 16-bit timers (Timer/Counter1, 3, 4, 5). These timers allow the program to execute accurate timing of events or measuring signals independent of software operation.

Following the Atmel datasheet notation, n is used to refer to the timer/counter number, x is used to the timer/counter unit channel. For example, OCnx – OC1A refers to the output compare on timer counter 1, channel A.

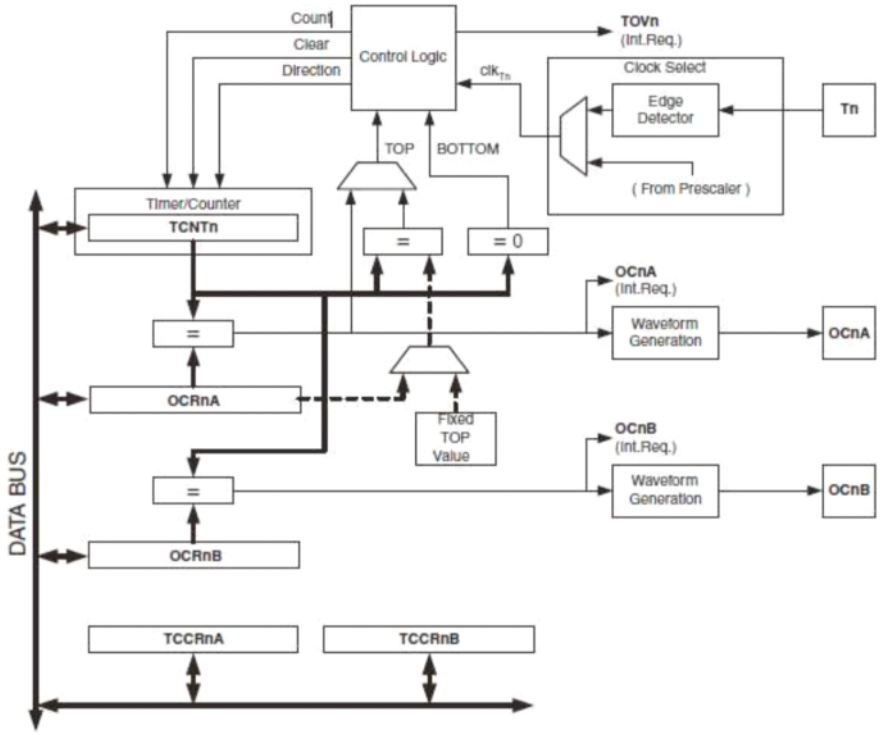
Six real-time interrupts are possible by making use of the timer overflows on each of the timers.

The block diagram below shows overview of the timer/counter system.

Page 2

Monday, March 27, 2017 4:12 PM

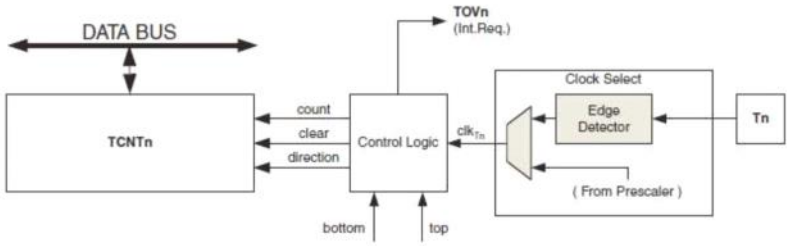
Figure 16-1. 8-bit Timer/Counter Block Diagram



The primary idea of the Timer/Counter (TC) is as follows. The TC generates a clock called the timer clock which is used to index (or decrement) a corresponding counter, TCNTn. TCNTn can be accessed at any time to determine the current time in timer clock cycles. When the timer is full (256 for 8 bit, 65536 for 16 bit), the timer sets and overflow flag, potentially generates an interrupt, resets and starts counting again from 0. The timer counter is also used to generate PWM signals and can be used in input capture.

The main part of the TC is the counter unit (TCNTn). TCNTn can be incremented, decremented or cleared at each clock cycle. In general we will assume an incrementing TCNTn. The counter is based on the system clock plus a prescaler.

Figure 16-2. Counter Unit Block Diagram

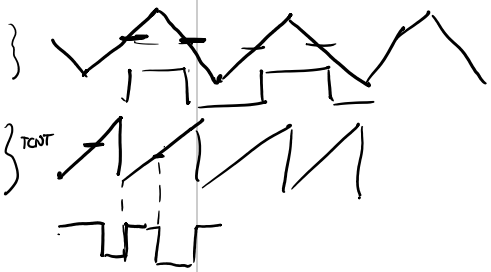


Modes of Operation: The TC has several models of operation as defined in the table below:

Table 17-2. Waveform Generation Mode Bit Description[1]

Mode	WGMn3	WGMn2 (CTCn)	WGMn1 (PWMn1)	WGMn0 (PWMn0)	Timer/Counter Mode of Operation	TOP	Update of OCRnX at	TOVn Flag Set on
0	0	0	0	0	Normal	0xFFFF	Immediate	MAX
1	0	0	0	1	PWM, Phase Correct, 8-bit	0x00FF	TOP	BOTTOM
2	0	0	1	0	PWM, Phase Correct, 9-bit	0x01FF	TOP	BOTTOM
3	0	0	1	1	PWM, Phase Correct, 10-bit	0x03FF	TOP	BOTTOM
4	0	1	0	0	CTC	OCRnA	Immediate	MAX
5	0	1	0	1	Fast PWM, 8-bit	0x00FF	BOTTOM	TOP
6	0	1	1	0	Fast PWM, 9-bit	0x01FF	BOTTOM	TOP
7	0	1	1	1	Fast PWM, 10-bit	0x03FF	BOTTOM	TOP
8	1	0	0	0	PWM, Phase and Frequency Correct	ICRn	BOTTOM	BOTTOM
9	1	0	0	1	PWM,Phase and Frequency Correct	OCRnA	BOTTOM	BOTTOM
10	1	0	1	0	PWM, Phase Correct	ICRn	TOP	BOTTOM
11	1	0	1	1	PWM, Phase Correct	OCRnA	TOP	BOTTOM
12	1	1	0	0	CTC	ICRn	Immediate	MAX
13	1	1	0	1	(Reserved)	–	–	–
14	1	1	1	0	Fast PWM	ICRn	BOTTOM	TOP
15	1	1	1	1	Fast PWM	OCRnA	BOTTOM	TOP

Note: 1. The CTCn and PWMn1:0 bit definition names are obsolete. Use the WGMn2:0 definitions. However, the functionality and location of these bits are compatible with previous versions of the timer.

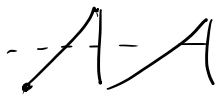


Normal Mode (mode 0): in Normal mode, the TCNTn is always incremented, and the counter is only reset when it reaches its maximum, 8 or 16 bit value. When reaching the max value, an overflow flag (TOVn) is set, and an interrupt can be generated.

Clear Timer on Compare Match (CTC) mode: Clears time when successful compare with OCnx is made.

Fast PWM Mode: provides PWM signals with counter starting at the bottom, running to top and then resetting at the bottom. The output is set or cleared on a successful compare.

3




Phase Correct PWM mode: Counter counts from bottom to Top and then back to bottom. The output is set or cleared on a successful compare when counting up, and then cleared or set on a successful compare when counting back down.
Atmel claims that the symmetric feature of the dual-slope PWM mode makes it preferred for motor control applications.

Phase and Frequency Correct PWM mode: Same as Phase Correct PWM except the time when OCRnx register is updated.

The remainder of these notes focus on normal mode and generating real-time interrupts.

Register Description: (for 16 bit timers)

Step 1. Set Timer/Counter output compare settings and mode.
TCCRnA (0x24)– Timer/Counter Control Register A, n=1,3,4,5

Bit 7	6	5	4	3	2	1	bit 0
COMnA1	**COMnA0**	**COMnB1**	**COMnB0**	**COMnC0**	**COMnC0**	**WGMn1**	**WGMn0**
Reset 0	0	0	0	0	0	0	0
All R/W							

With
COMnx1:0 Compare Match Output A Mode:
These control the Output Compare pin (OCnx with x=A, B or C_ behavior and depend on WGM01:0 bit settings – here we assume normal port operation, see chart:

Table 17-4. Compare Output Mode, Fast PWM

COMnA1 COMnB1 COMnC1	COMnA0 COMnB0 COMnC0	Description
0	0	Normal port operation, OCnA/OCnB/OCnC disconnected
0	1	WGM13:0 = 14 or 15: Toggle OC1A on Compare Match, OC1B and OC1C disconnected (normal port operation). For all other WGM1 settings, normal port operation, OC1A/OC1B/OC1C disconnected
1	0	Clear OCnA/OCnB/OCnC on compare match, set OCnA/OCnB/OCnC at BOTTOM (non-inverting mode)
1	1	Set OCnA/OCnB/OCnC on compare match, clear OCnA/OCnB/OCnC at BOTTOM (inverting mode)

Note: A special case occurs when OCRnA/OCRnB/OCRnC equals TOP and COMnA1/COMnB1/COMnC1 is set. In this case the compare match is ignored, but the set or clear is done at BOTTOM. See "Fast PWM Mode" on page 146. for more details.

WGMn1:0 Waveform Generation Mode
These are combined with the WGM02,3 bit found in the TCCR0B Register, are used to set mode.

Page 5

Monday, March 27, 2017 4:12 PM

Table 17-2. Waveform Generation Mode Bit Description[1]

Mode	WGMn3	WGMn2 (CTCn)	WGMn1 (PWMn1)	WGMn0 (PWMn0)	Timer/Counter Mode of Operation	TOP	Update of OCRnx at	TOVn Flag Set on
0	0	0	0	0	Normal	0xFFFF	Immediate	MAX
1	0	0	0	1	PWM, Phase Correct, 8-bit	0x00FF	TOP	BOTTOM
2	0	0	1	0	PWM, Phase Correct, 9-bit	0x01FF	TOP	BOTTOM
3	0	0	1	1	PWM, Phase Correct, 10-bit	0x03FF	TOP	BOTTOM
4	0	1	0	0	CTC	OCRnA	Immediate	MAX
5	0	1	0	1	Fast PWM, 8-bit	0x00FF	BOTTOM	TOP
6	0	1	1	0	Fast PWM, 9-bit	0x01FF	BOTTOM	TOP
7	0	1	1	1	Fast PWM, 10-bit	0x03FF	BOTTOM	TOP
8	1	0	0	0	PWM, Phase and Frequency Correct	ICRn	BOTTOM	BOTTOM
9	1	0	0	1	PWM,Phase and Frequency Correct	OCRnA	BOTTOM	BOTTOM
10	1	0	1	0	PWM, Phase Correct	ICRn	TOP	BOTTOM
11	1	0	1	1	PWM, Phase Correct	OCRnA	TOP	BOTTOM
12	1	1	0	0	CTC	ICRn	Immediate	MAX
13	1	1	0	1	(Reserved)	–	–	–
14	1	1	1	0	Fast PWM	ICRn	BOTTOM	TOP
15	1	1	1	1	Fast PWM	OCRnA	BOTTOM	TOP

Note: 1. The CTCn and PWMn1:0 bit definition names are obsolete. Use the WGMn2:0 definitions. However, the functionality and location of these bits are compatible with previous versions of the timer.

Step 2: Set the mode, prescale for the timer clock:

TCCRnB (0x25)– Timer/Counter Control Register B, n=1,3,4,5

Bit 7	6	5	4	3	2	1	bit 0
ICNCn	**ICESn**		**WGMn3**	**WGMn2**	**CSn2**	**CSn1**	**CSn0**
Reset 0	0	0	0	0	0	0	0
All R/W							

With

ICNCn: Input Capture Noise Canceler

Noise canceler, the input from input capture pin is filtered. It requires four samples of the ICPn pin to charge output and therefore delays input capture by four oscillator cycles.

ICESn: Input Capture Edge select

Selects which edge on the ICPn that is used to trigger an event (zero is falling, one is rising)

WGMn3:2 Waveform Generation Mode

These are combined with the WGMn1:0 bit found in the TCCRnA Register, are used to set mode.

Page 6

Monday, March 27, 2017 4:12 PM

CS02:0 Clock Select
Three clock select bits select the clock source to be used by the Timer/Counter as shown in the table below.

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

TCCRnC – Timer/Counter n Control register C (n= 1,3,4,5)
Only used in non-pwm mode to force compare match. We will not use this at this time.

Register Location of Timer counter

TCNTnH,L– Timer/Counter Register H/L

Bit 7	6	5	4	3	2	1	bit 0
TCNT15							
Reset 0	0	0	0	0	0	0	0
All R/W							

Bit 7	6	5	4	3	2	1	bit 0
TCNT07							
Reset 0	0	0	0	0	0	0	0
All R/W							

This register contains the current count of the 16 bit Timer clock.

OCRnxH/L Output Compare Register (n=1,3,4,5, x=A,B,C)

Bit 7	6	5	4	3	2	1	bit 0
OCR0A15-0							
Reset 0	0	0	0	0	0	0	0

Page 7

Monday, March 27, 2017 4:12 PM

All R/W

OCRnAH/L contains 16-bit value to compare with TCNTnHL, a match generates the output compare flag.

Step 3: Enable Interrupts for the timer

ICRnH/L: Input Capture Register n
This 16 bit register contains the value of TCNT when an input capture occurs for each timer/counter.

TIMSKn Timer/Counter Interrupt Mask Register (n=1,3,4,5)

Bit 7	6	5	4	3	2	1	bit 0
		ICIEn		**OCIEnC**	**OCIEnB**	**OCIEnA**	**TOIEn**
Reset 0	0	0	0	0	0	0	0
All R/W							

ICIEn Input Capture Interrupt Enable:
When this bit is set as 1, (and global interrupt enable is on), the input capture interrupt is enabled for T/Cn.

OCIE0x Timer/Counter Output Compare Match x (x=A, B, C) Interrupt Enable
When set, this bit enables the Output Compare Match B/A interrupt, allowing the interrupt to be executed if a successful output compare match occurs.

TOIE0 Timer/Counter0 Overflow Interrupt Enable
When this bit is set, (and the clobal interrupt enable is on), the timer overflow interrupt is enabled for T/Cn.

Step 4: Clear the timer flags:

TIFRn – Timer/Counter 0 Interrupt flag register (n=1,3,4,5)

Bit 7	6	5	4	3	2	1	bit 0
		ICFn		**OCFnV**	**OCFnB**	**OCFnA**	**TOVn**
Reset 0	0	0	0	0	0	0	0
All R/W							

ICFn Timer/Counter, Input Compare x Match Flag (n=1,3,4,5)

Page 8

Monday, March 27, 2017 4:12 PM

This bit is set (as a flag) when a successful input capture event occurs. This bit is cleared by hardware when the corresponding interrupt handling vector is executed (i.e., interrupt is enabled). Otherwise the flag can be cleared by writing a one to it.

OCFnx Timer/Counter Output Compare x Match Flag (n=1,3,4,5. X = A, B,C)
This bit is set (as a flag) when a successful compare event occurs match occurs. This bit is cleared by hardware when the corresponding interrupt handling vector is executed (i.e., interrupt is enabled). Otherwise the flag can be cleared by writing a one to it.

TOVn Timer/Counter0 Overflow Flag (n=1,3,4,5)
This bit is set as a flag when when TCNTn overflows from 0xFFFF to 0x0000. It is cleared by hardware if the interrupt is executed or can be cleared by writing a 1 to it.

Page 9

Monday, March 27, 2017 4:12 PM

Summary: Timer Programming as real-time interrupts

0. Select timer and channel
.5 ~~Clear Global interrups~~
1. Set corresponding Output Compare pin to no outpt (TCCRnA)
2. Set waveform generation mode to normal mode, set clock prescale (TCCRnB)
3. Enable local timer overflow interrupt (TIMSK1)
4. Clear timer overflow Flag (TIFRn) ?
5. ~~Enable global interrupts - SEI~~
6. Write Timer overflow interrupt code in: ISR(Timer1_OVF_vect)

Example Program

```
// Simple 16 bit Timer example
// Tristan Hill - January,14,2016

volatile uint16_t ofcnt=0;
volatile float   time_sec=0;


void setup() {
  // put your setup code here, to run once:

  // setup Timer 1
  TCCR1A = 0b00000000; // mode 0 (normal mode), output setings
  TCCR1B = 0b00000001; // mode, timer prescale = 1 (no prescale)
  TIMSK1 = 0b00000001; // overflow interrupt enable

  // setup serial monitor
  Serial.begin(9600);         //  setup serial
  while (! Serial);
  Serial.println("16-bit timer example");

}

void loop() {
  // put your main code here, to run repeatedly:

  delay(500);
  Serial.print("Time: \n");
  Serial.print(time_sec,DEC);
  Serial.print("\n");
}
```

```
//
**********************************************************************
********
// Interrupt Routines
//
**********************************************************************
********
// timer1 overflow
ISR(TIMER1_OVF_vect) {
   ofcnt++; // overflow counts
   time_sec=time_sec+.004096; // 1/16e6*65536
}
```

TCNT:

0 → FFFF

$\frac{1}{16e6\ \text{p.s.}}$ sec. * 65536

---

stepper → $\omega = 1\ \text{rad/s}$ → full stepping mode

200 step / rotation.

$\omega' = 1\ \text{rad/s} * \frac{200\ \text{step}}{\text{rotation}} * \frac{1\ \text{rotation}}{2\pi\ \text{rad.}} = \frac{100}{\pi}\ \text{step/s.}$

$\frac{1}{\omega'} = \frac{\pi}{100}\ \frac{\text{sec}}{\text{step.}} = .0314\ \text{sec./step}$

Over flow occurs when TCNT1 → $65,535+1$ counts / overflow. * 16e6 counts/sec. * $\frac{1}{1024}$

$(65535+1)\ \text{counts} * \frac{1\ \text{sec.}}{16e6\ \text{counts}} * \frac{1024}{1}$

= 4.2 sec.

10 1.05

,5

$= \frac{.032768}{.004096\ \text{sec.}}$ ← 8 ← Prescale of 1.

1024

256

128