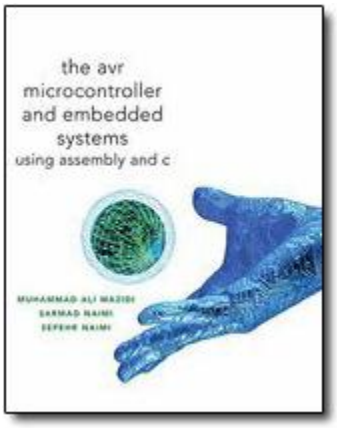
Timer/Counter with PWM



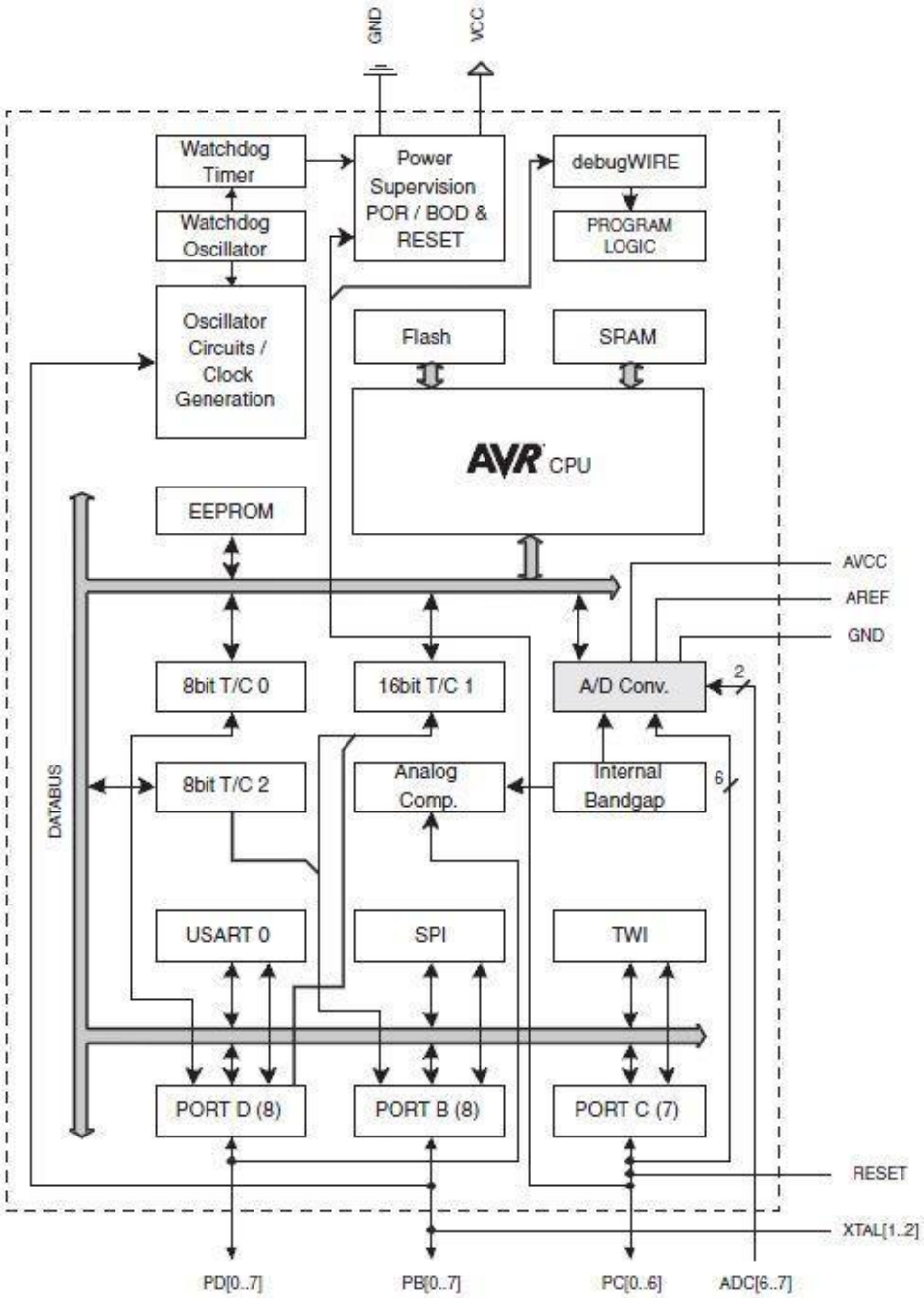
The AVR Microcontroller and Embedded Systems using Assembly and C)
by Muhammad Ali Mazidi, Sarmad Naimi, and Sepehr Naimi

ATMEL 8-bit AVR Microcontroller with 4/8/16/32K Bytes In-System Programmable Flash
http://www.atmel.com/dyn/resources/prod_documents/doc8161.pdf Chapter 17 “8-bit Timer/Counter 2 with PWM and Asynchronous Operation”

Table of Contents

ATmega328P Timing Subsystem

Most microcontrollers provide at least one port that has timer sub-circuitry capable of generating PWM signals on a port pin. Typically, one just needs to configure the square-wave frequency and desired duty cycle via a couple of registers. When enabled, the port pin will output a PWM signal that can be demodulated in order to provide an approximation to an analog signal. In the case of the ATmega328P, there are two 8-bit timers and one 16-bit timer, all of which are capable of generating PWM outputs.



Design Example

To provide some specificity to my explanation of the timing subsystem, I will assume the use of 8-bit Timer 2. In most design examples, I will further assume 8-bit Timer 2 operating in one of its four (4) PWM modes, with the output coming from compare register B (see Figure 1).

For example, when calculating the output frequency of one of our 6 PWM pins; in place of using the more general form $f_{OCnxPWM}$ where:

f = frequency
OC = output compare pin
n = timer/counter number 0, 1, and 2
x = output from output compare register A or B
PWM = Pulse Width Modulation mode

I will say $f_{OC2B.}$ For 8-bit timers 0 and 2, this specific design example directly translates to the more general case. This is not always true for our 16-bit Timer 1.

8-bit Timer/Counter 2 Subsystem

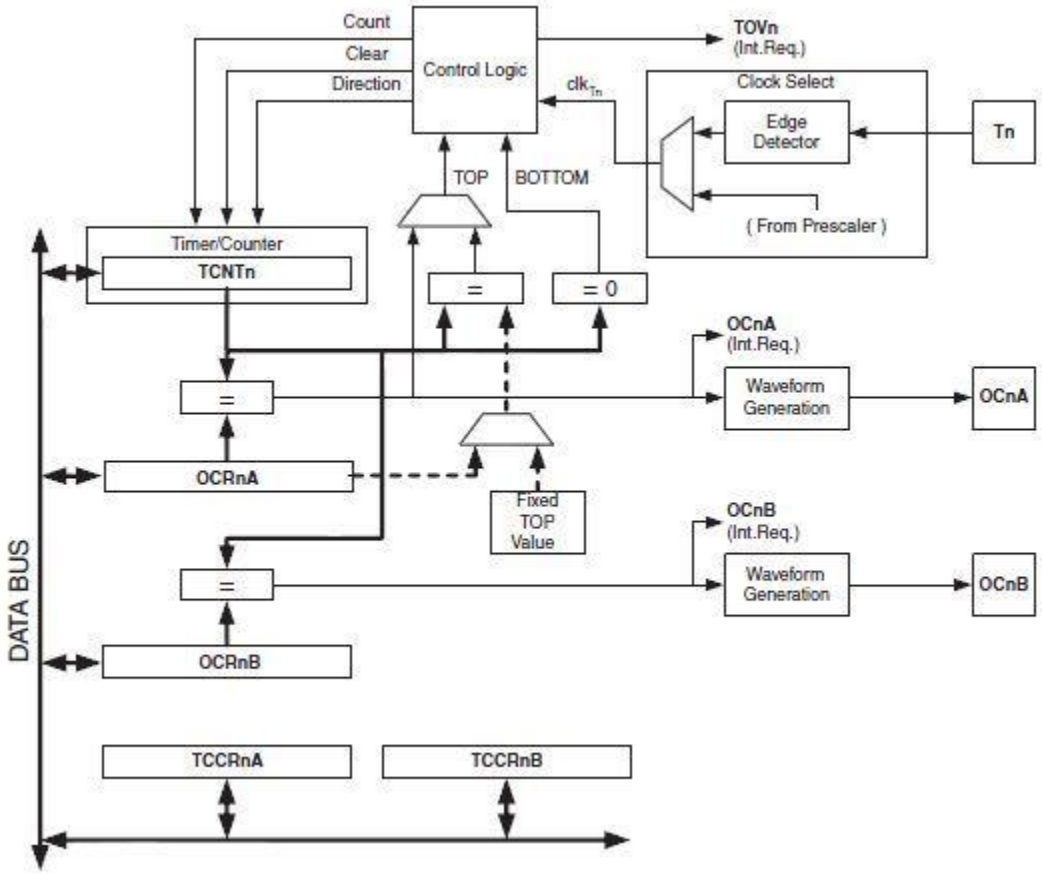


Figure 1 8-bit Timer/Counter 2 Subsystem Block Diagram

Clock Source

All our design examples will assume operation of the ATmega328P within the context of the the Arduino system. Specifically, our system clock source is a crystal input XTAL1/TOSC1 and XTAL2/TOSC2.

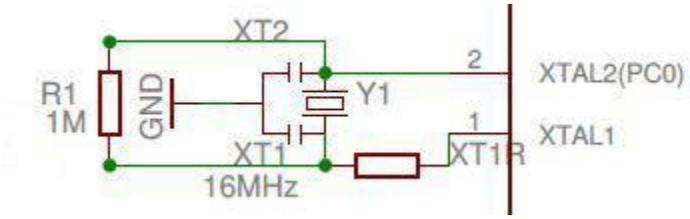


Figure 2 Arduino/ATmega328P System Clock

For this design implementation the following will always be true (The interested student is invited to read Section 17.3 “Timer/Counter Clock Sources” in the ATmega328P datasheet for why this is true).

$$clk_{SYS} = clk_{I/O} \qquad \text{eq. 1}$$

and therefore...

$$f_{CLK} = f_{I/O} = 16\ MHz \qquad \text{eq. 2}$$

Timing Terminology

Frequency

The number of times an event repeats within a 1-second period. The unit of frequency is Hertz, or cycles per second. For example, a sinusoidal signal with a 60 Hz frequency means that a full cycle of a sinusoid signal repeats itself 60 times each second.

Period

The flip side of a frequency is a period. If an event occurs with a rate of 60 Hz, the period of that event is 16.67 ms.

Duty Cycle

Duty cycle is defined as the percentage of one period a signal is ON.

Pulse Width Modulation

Several modulation methods have been developed for applications that require a digital representation of an analog signal. One popular and relevant scheme is pulse width

modulation (PWM) in which the instantaneous amplitude of an analog signal is represented by the width of periodic square wave. For example, consider the signals depicted in Fig. 3. Notice, the PWM version of the signal has a fixed frequency defining the point when a pulse begins. During the period of an individual pulse, the signal remains high for an amount of time proportional to the amplitude of the analog signal.

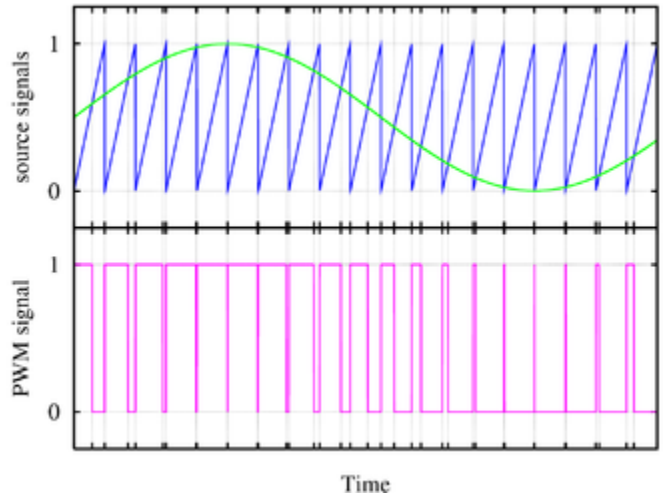


Figure 3 An example analog signal and a pulse width modulated representation.

source: http://en.wikipedia.org/wiki/Pulse-width_modulation

Bottom, Max Top

BOTTOM	The counter reaches the BOTTOM when it becomes zero (0x00).
MAX	The counter reaches its MAXimum when it becomes 0xFF (decimal 255).
TOP	The counter reaches the TOP when it becomes equal to the highest value in the count sequence. The TOP value can be assigned to be the fixed value 0xFF (MAX) or the value stored in the OCR2A Register. The assignment is dependent on the mode of operation.

Waveform Generation Modes

On the ATmega328P, three waveform generation bits exist within the two timer/counter control registers. Four of the eight possible waveform generation modes involve PWM waveform outputs, two of which are considered fast PWM while the remaining two are called phase-correct PWM.

Table 1 Waveform Generation Mode Bit Description

Mode	WGM2	WGM1	WGM0	Timer/Counter Mode of Operation	TOP	Update of OCRx at	TOV Flag Set on[1][2]
0	0	0	0	Normal	0xFF	Immediate	MAX
1	0	0	1	PWM, Phase Correct	0xFF	TOP	BOTTOM
2	0	1	0	CTC	OCRA	Immediate	MAX
3	0	1	1	Fast PWM	0xFF	BOTTOM	MAX
4	1	0	0	Reserved	–	–	–
5	1	0	1	PWM, Phase Correct	OCRA	TOP	BOTTOM
6	1	1	0	Reserved	–	–	–
7	1	1	1	Fast PWM	OCRA	BOTTOM	TOP

Notes:

1. MAX = 0xFF
2. BOTTOM = 0x00
3. In normal operation the Timer/Counter Overflow Flag (TOV0) will be set in the same timer clock cycle as the TCNT0 becomes zero.
4. Whenever TCNT0 equals OCR0A or OCR0B, the comparator signals a match. A match will set the Output Compare Flag (OCF0A or OCF0B) at the **next timer clock cycle**.

Normal Mode

Normal Mode (WGM 1 bits 3:0 = 0000_2)

Bit	7	6	5	4	3	2	1	0	
(0x80)	COM1A1	COM1A0	COM1B1	COM1B0	–	–	WGM11	WGM10	TCCR1A
Read/Write	R/W	R/W	R/W	R/W	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Bit	7	6	5	4	3	2	1	0	
(0x81)	ICNC1	ICES1	–	WGM13	WGM12	CS12	CS11	CS10	TCCR1B
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	

Bit	7	6	5	4	3	2	1	0	
0x16 (0x36)	–	–	ICF1	–	–	OCF1B	OCF1A	TOV1	TIFR1
Read/Write	R	R	R/W	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

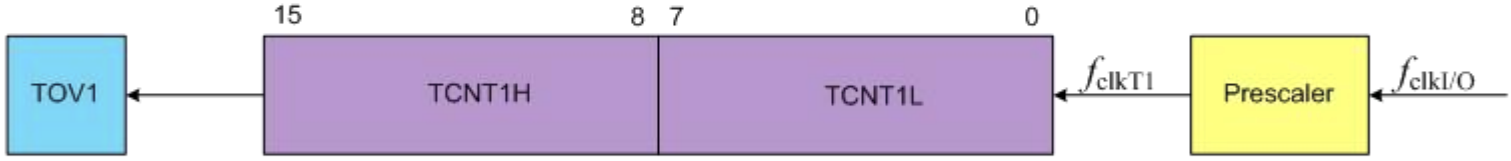


Table 13-5. Clock Select Bit Description

CS12	CS11	CS10	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 T1 pin. Clock on falling edge.
1	1	1	External clock source on T1 pin. Clock on rising edge.

Figure 4 Normal Mode

PWM Waveform Generation Modes

PWM Types

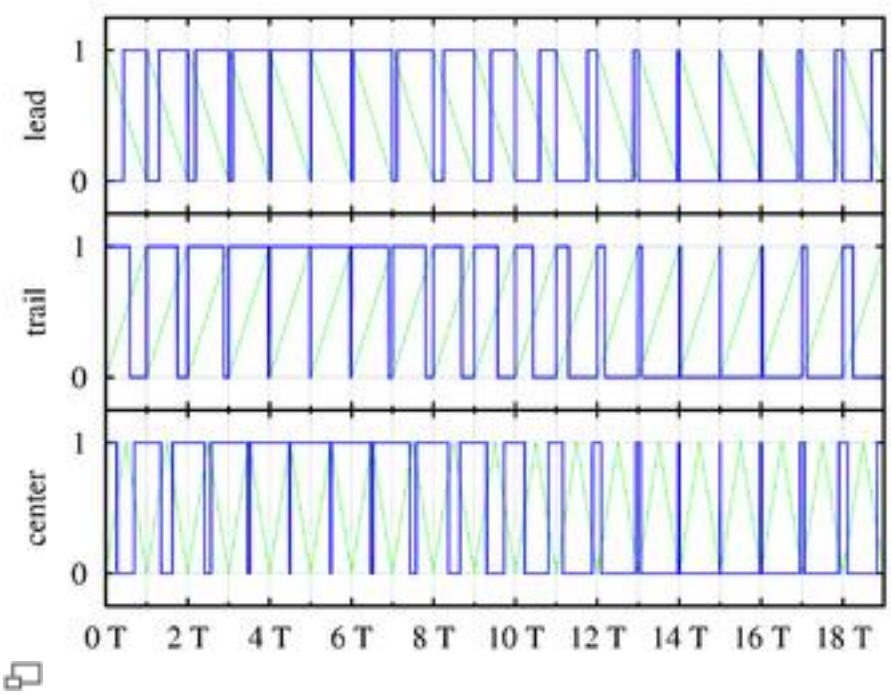


Figure 5 Three types of PWM signals (blue): leading edge modulation (top), trailing edge modulation (middle) and centered pulses (both edges are modulated, bottom). The green lines are the sawtooth waveform (first and second cases) and a triangle waveform (third case) used to generate the PWM waveforms using the intersective method.

Four types of pulse-width modulation (PWM) are possible:

1. The tail edge can be fixed and the **lead edge** modulated. *ATmega328P Fast PWM inverting modes 3 and 7.*
2. The lead edge can be fixed and the **tail edge** modulated. *ATmega328P Fast PWM non-inverting modes 3 and 7.* This is the mode used by the **Adafruit Motor Shield**.
3. The pulse center may be fixed in the center of the time window and both edges of the pulse moved to compress or expand the width. *ATmega328P PWM Phase Correct modes 1 and 5.*
4. The frequency can be varied by the signal, and the pulse width can be constant. However, this method has a more-restricted range of average output than the other three. *ATmega328P CTC mode 2*

Note: *ATmega328P modes 4 and 6 are reserved (i.e., undefined)*

ATmega328P 8-bit PWM Modes

Shown in Fig. 6 and Fig. 7 are the four different output waveforms given the specified waveform configurations.

Timer Modes 3 and 1

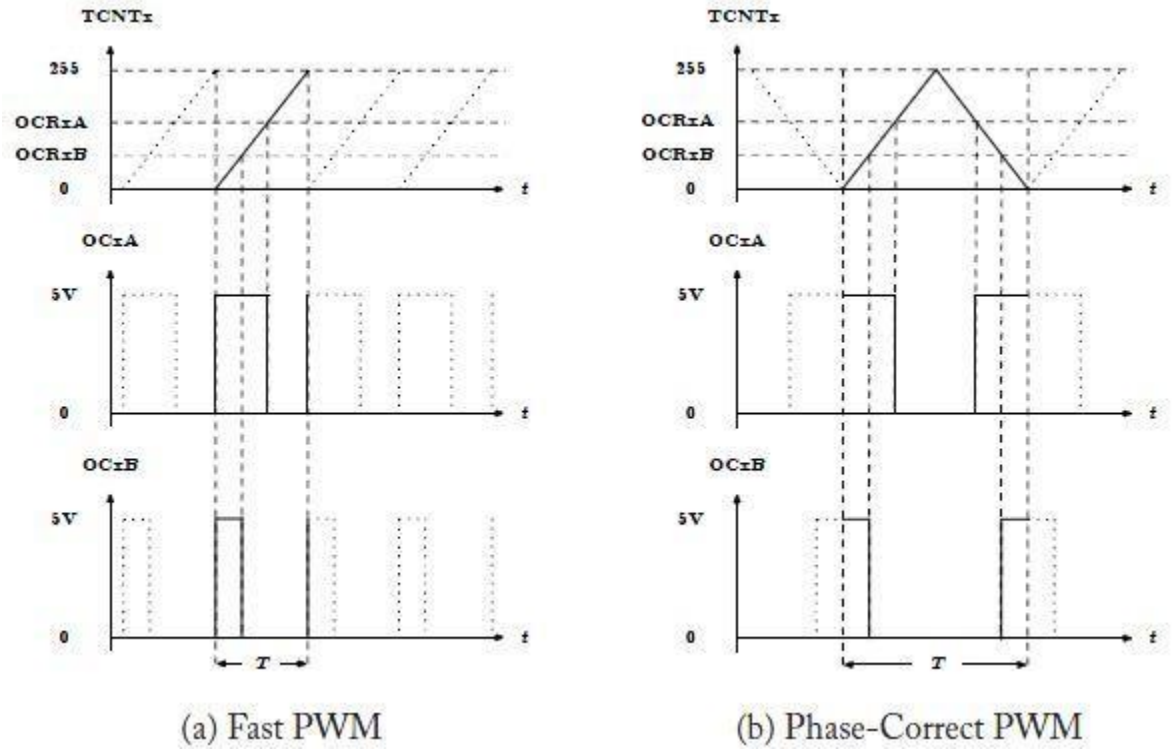


Figure 6 Non-inverting Timer Modes 3 and 1

In general, the PWM generation circuitry operates based on the 8-bit or 16-bit count register TCNT which updates its current value every time there is a clock pulse. As long as the TCNT value is below the value stored in the output compare register OCRnA or OCRnB, then the associated output pin OCnA or OCnB will remain in a specific state, for example, set high. Once the TCNT value becomes greater than the compare register value, the output pin will switch to the opposite state, for example clear low. This operation will continue until the timer is disabled.

Mode 3 Fast PWM

The first output mode shown in Fig. 6(a) represents the waveforms generated given a fast PWM setting where the TOP value is fixed at the maximum 8-bit value of 255. In this mode, two different output compare register values can be set independent of each other, each affecting a different output pin (OCnA, OCnB). For our design example, two separate PWM waveforms may be generated on pin 17 (PB3 MOSI/OC2A) and pin 5 (PD3 OC2B/INT1).

From Figure 1 “8-bit Timer/Counter 2 Subsystem Block Diagram” and Figure 17-12 “Prescaler for Timer/Counter2” it is seen that:

$f_{T2} = f_{I/O}/N$ eq. 3

The N variable represents the prescale factor (1, 8, 32, 64, 128, 256, or 1024). This will be covered in more detail in the Register section of this document.

Given that in this PWM setting (mode 3) that the TOP value is fixed at the maximum 8-bit value of 255 and that the OC2B output is changed on the next clock cycle it can further be shown that:

$f_{OC2B} = f_{T2} / 256$ eq. 4

Combining equations 1, 3, and 4 we see that the PWM frequency for the output can be calculated by the following equation:

$f_{OC2B} = f_{CLK} / N * 256$ eq. 5

The general form given by the equation:

$$f_{OCnxPWM} = \frac{f_{clk\_I/O}}{N \cdot 256}$$ eq. 6 (Section 17.7.3 Fast PWM Mode)

The Adafruit motor shield design example sets N = 1:

$f_{OC2B} = f_{CLK} / 256 = 16\ MHz / 256 = 62.5\ KHz \simeq 64\ KHz$ eq. 7

AFMotor

To get sidetracked for a moment. In the AFMotor header file (i.e., `C:\Program Files\arduino-0022\libraries\AFMotor`) you will find the following definitions:

```
#define MOTOR12_64KHZ _BV(CS20)  // no prescale
```

where CS20 is further defined as equal to zero, and therefore:

```
MOTOR12_64KHZ = 0b00000001
```

Here is the line of code that instantiates the Adafruit motor shield

```
AF_DCMotor motor(2, MOTOR12_64KHZ); // create motor #2, 64KHz pwm
```

The first parameter is used to set the static property `motornum`. The second parameter (freq) is used to initialize Timer/Counter Configuration Register `TCCR2B`.

```
TCCR2B = freq & 0x7;
```

Putting this all together we have the prescalar set to 1 (no prescalar) and a corresponding output frequency of approximatly 64 KHz. Please read the companion lecture “Adafruit Motor Shield - Part 2” for more information.

The period of the PWM waveform is therefore $T_{OC2B} = 256/f_{CLK}$, which is a little more than half the period of the phase-correct version (i.e., it is faster) - which brings us to the next section.

Mode 1 Phase Correct PWM

The second output mode shown in Fig. 6(b) represents the waveforms generated given the phase-correct PWM setting where the TOP value is also fixed at the maximum 8-bit value of 255. As in the fast PWM case, two different output compare register values can be set independent of each other, each affecting their own output pin. As can be seen, this mode alters the TCNT register behavior in that once the counter reaches the TOP value of 255, it begins counting backwards toward 0. The benefit has to do with the phase of the modulated carrier. In particular, notice the narrower pulses of OCnB as compared to that of OCnA in both Fig. 6(a-b). In the fast PWM non-inverted case, the **front edges line up**, whereas in the phase-correct case, the **center of the pulses line up**; that is, the phase of the OCnA and OCnB waveforms are equivalent.

The PWM frequency for the output can be calculated by the following equation:

$$f_{OCnxPCPWM} = \frac{f_{clk\_I/O}}{N \cdot 510}$$

Section 17.7.4 Phase Correct PWM Mode

The N variable represents the prescale factor (1, 8, 32, 64, 128, 256, or 1024). This will be covered in more detail in the Register section of this document. For our design example we will set N = 1.

$f_{OC2B} = f_{CLK}$ / 510 = 16 MHz / 510 = 31.3725 MHz

The period of the PWM waveform is therefore $T_{OC2B} = 510/f_{CLK}$, and the period of the phase correct PWM waveform is nearly doubled from that of the fast PWM waveform.

You may be asking why 510 and not 512 (2 x 256)? In the fast PWM case, the counter follows the sequence {0, 1, ..., 254, 255, 0}, which means there are 256 values in a single period. In the phase-correct case, the counter follows the sequence {0, 1, ..., 254, 255, 254, ..., 1, 0}, which means there are (510 = 255 + 255) values in a single period.

Timer Modes 7 and 5

The final two output modes shown in Fig. 7 represent the fast and phase-correct PWM waveforms when the TOP value is set to the 8-bit value stored in OCRnA.

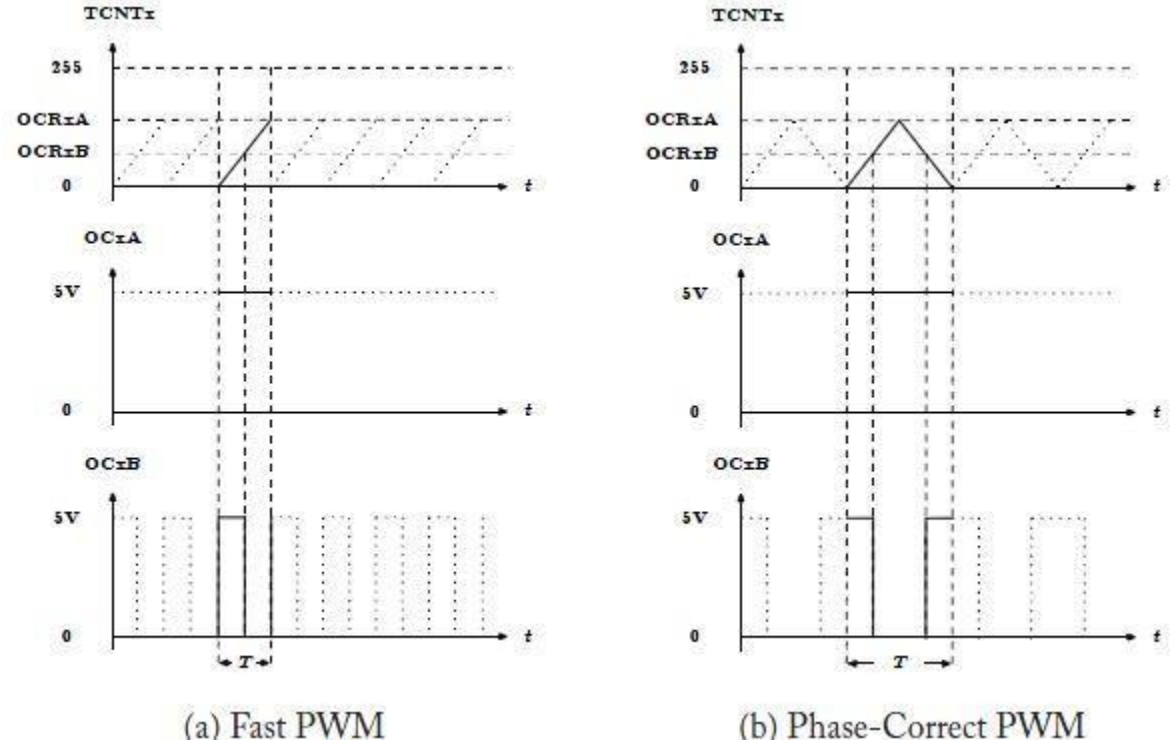


Figure 7 Non-inverting Timer Modes 7 and 5

Both of these modes effectively disable the OCnA pin functionality at the benefit of increasing the PWM frequency dramatically. In both cases, the TCNT register will count up to the OCRnA value, and then either reset to 0 or start counting down toward 0. The only comparison that matters is that to OCRnB, which will affect the OCnB pin as in the previous cases. One significant impact is that for a value of X loaded into OCRnA, the total resolution of the duty cycle output is reduced from 256 to X + 1 for fast PWM and 510 to $2X$, where X is a 8-bit number.

Exercise: Write the equation for f_{OC2B} with prescale factor N (1, 8, 32, 64, 128, 256, or 1024) for both Modes 7 and 5.

Registers

Bit	7	6	5	4	3	2	1	0	
(0xB0)	COM2A1	COM2A0	COM2B1	COM2B0	–	–	WGM21	WGM20	TCCR2A
Read/Write	R/W	R/W	R/W	R/W	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Bit	7	6	5	4	3	2	1	0	
(0xB1)	FOC2A	FOC2B	–	–	WGM22	CS22	CS21	CS20	TCCR2B
Read/Write	W	W	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Timer/Counter Control Register A (TCCR2A)

Inverting versus Non-inverting Modes (COM2A1, COM2A0 and COM2B1, COM2A0)

Compare Output Mode bits (COM2A1 and COM2A0 -- Timer/Counter 2 Output Compare Register A used as an example) define if the mode is non-inverting (COM2A1 = 1, COM2A0 = 0) or inverting (COM2A1 = 1, COM2A0 = 1).

COM2A1	COM2A0	Description
0	0	Normal port operation, OC0A disconnected.
0	1	Toggle OC2A on Compare Match
1	0	Clear OC2A on Compare Match
1	1	Set OC2A on Compare Match

Timer/Counter Control Register B (TCCR2B)

Timer/Counter Prescaler (CS22, CS21, CS20)

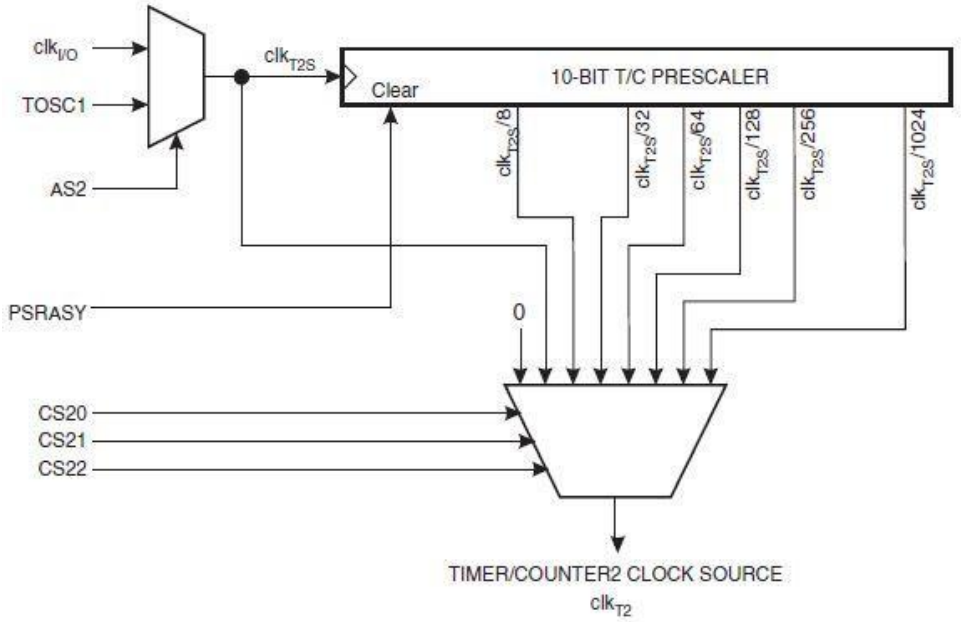


Figure 17-12 Prescaler for Timer/Counter2

Table 17-9 Clock Select Bit Description

CS22	CS21	CS20	Description
0	0	0	No clock source (Timer/Counter stopped).
0	0	1	clk_{T2S}/(No prescaling)
0	1	0	clk_{T2S}/8 (From prescaler)
0	1	1	clk_{T2S}/32 (From prescaler)
1	0	0	clk_{T2S}/64 (From prescaler)
1	0	1	clk_{T2S}/128 (From prescaler)
1	1	0	clk_{T2S}/256 (From prescaler)
1	1	1	clk_{T2S}/1024 (From prescaler)

Force Output Compare A (FOC2A and FOC2B)

The FOC2A and FOCB bits are only active when the WGM bits specify a non-PWM mode.