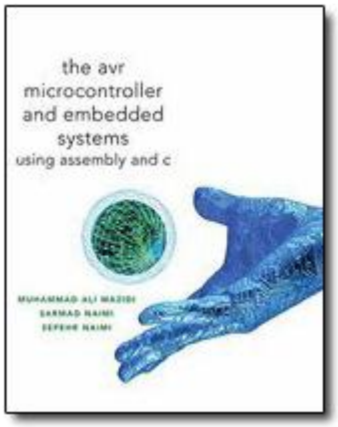
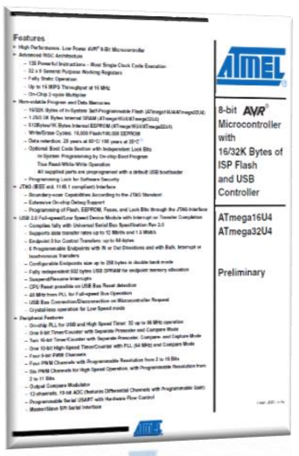
3DoT C++ Timer/Counter 4 with PWM

This article is on the motor control section of the 3DoT board using Timer/Counter 4 operating in Fast PWM mode.


the avr microcontroller and embedded systems using assembly and c
MUHAMMAD ALI MAZIDI
SARMAD NAIMI
SEPEHR NAIMI

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

Chapter 16: PWM Programming and DC Motor Control in AVR



Features

[illegible]

8-bit AVR Microcontroller with 16/32K Bytes of ISP Flash and USB Controller

ATmega16U4
ATmega32U4

Preliminary

ATMEL 8-bit AVR Microcontroller with 16/32K Bytes of ISP Flash and USB – ATmega32U4

Chapter 13 “8-bit Timer/Counter 0 with PWM,” Chapter 14 “16-bit Timers/Counters,” and Chapter 15 “10-bit High Speed Timer/Counter4.”

Table of Contents

Motor Direction Control

Figure 1 “Atmega32U4 to Motor Driver Interface” shows that we can configure 2 motors where each motor has 3 control pins and a standby (STBY) pin for the entire IC. If STBY is set low then the motors are not powered regardless of the state of the other two pins. The pins attached to AIN1/2 and BIN1/2 are digital outputs that control the rotation of the motor. The PWM inputs are connect to a pins capable of a PWM output to control the speed of the motor.

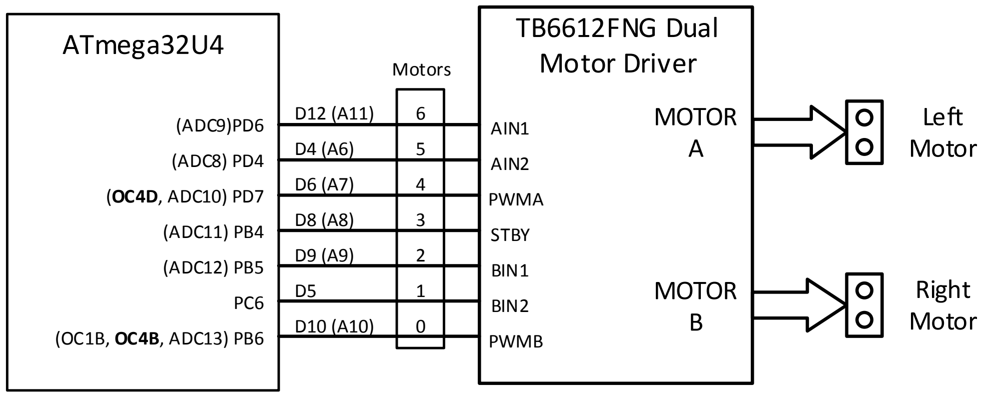


Figure 1 – Atmega32U4 to Motor Driver Interface

For the remainder of this article use Figure 1 “Atmega32U4 to Motor Driver Interface” to help you cross-reference the *tower of babel* names used by Atmel, Arduino, and Toshiba (i.e., TB6612FNG).

To prevent damage to the internal circuitry of the TB6612FNG, the IC includes clamping diodes on the inputs, a series resistor to limit in-rush current, and a weak pull-down resistor to keep the N-channel MOSFET OFF (Figure 2a Input Circuit). To prevent damage to the output circuitry internal flyback (i.e., snubber, flywheel) diodes are included (Figure 2b Flyback Diodes). Here is one of many articles on how to use a MOSFET as a Switch which goes into a little more details on these circuit elements.

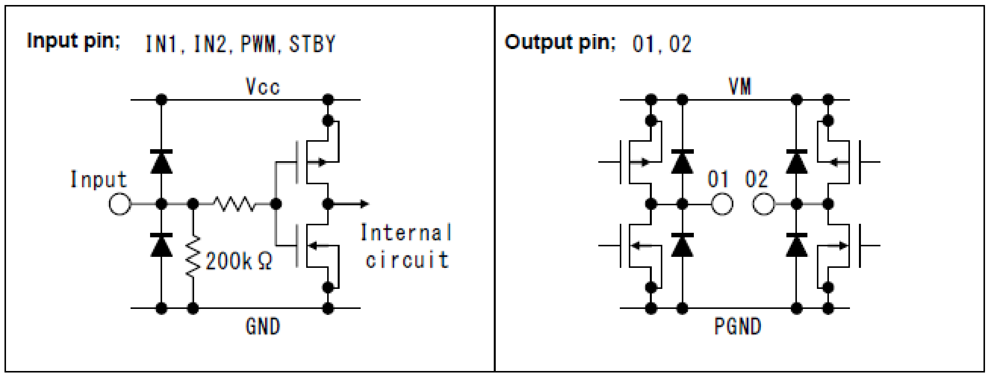
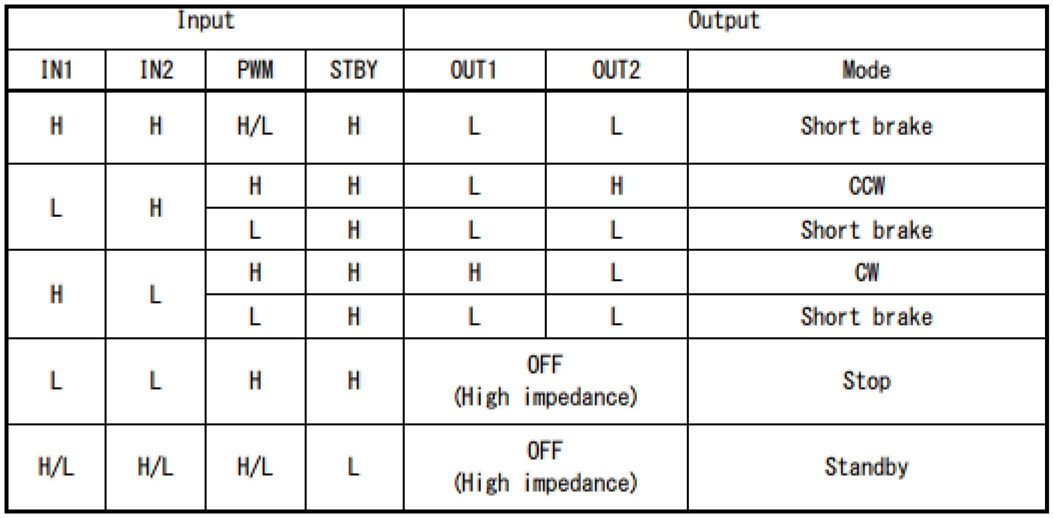


Figure 2 – TB6612FNG Input and Output Circuits

The direction in which the motors turn are defined in Table 1 and illustrated in Figure 2 “TB6612FNG H-Bridge states t1 to t5.” For example, to configure Motor A to turn clockwise (CW) you would want to set the H-Bridge to state t1 (see Figure 3). This would be accomplished by setting PD6 = 1 and PD4 = 0 (see Table 1 and Figure 1). To turn counter-clockwise (CCW) you would want to set the H-Bridge to state t5. This would be accomplished by setting PD6 = 0 and PD4 = 1.

Table 1 TB6612FNG Motor Control Truth Table



Input				Output		
IN1	IN2	PWM	STBY	OUT1	OUT2	Mode
H	H	H/L	H	L	L	Short brake
L	H	H	H	L	H	CCW
		L	H	L	L	Short brake
H	L	H	H	H	L	CW
		L	H	L	L	Short brake
L	L	H	H	OFF (High impedance)		Stop
H/L	H/L	H/L	L	OFF (High impedance)		Standby

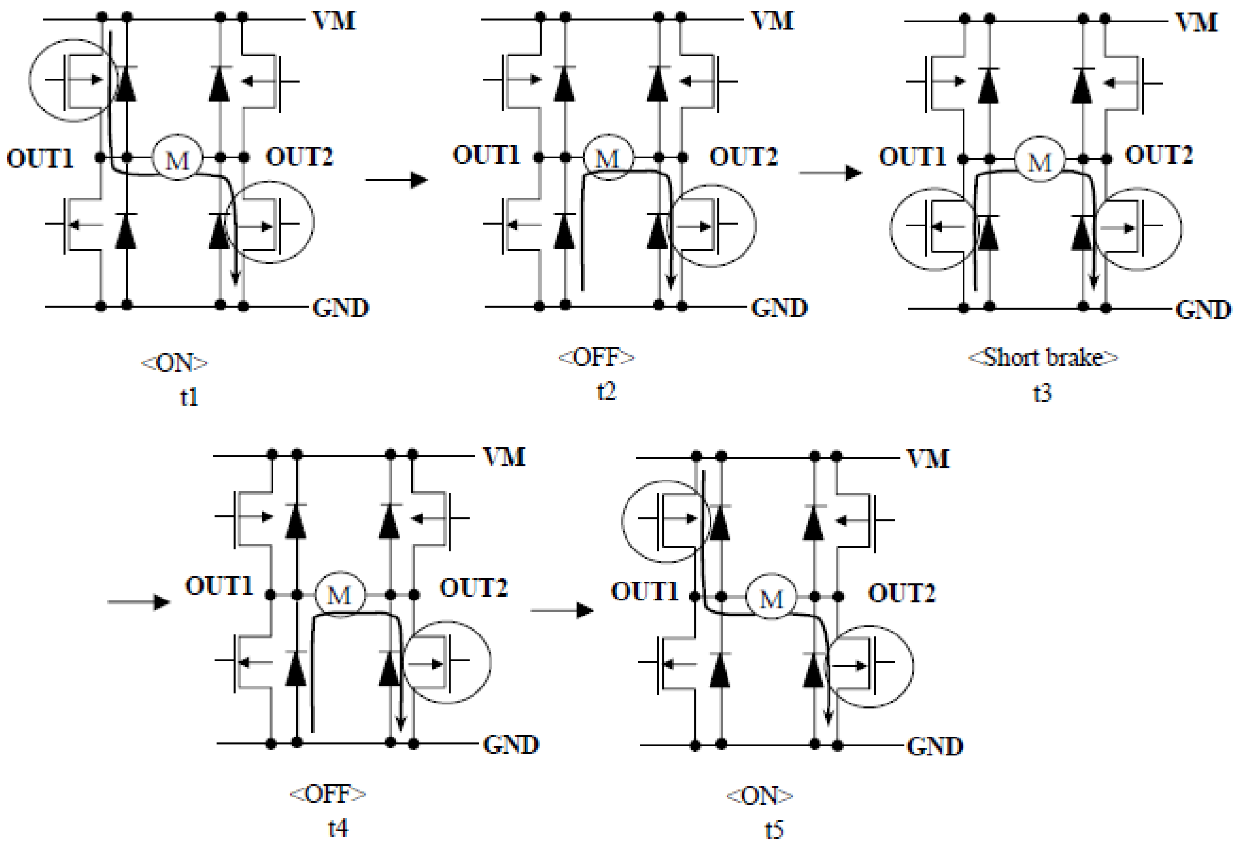


Figure 3 TB6612FNG H-Bridge states t1 to t5

To prevent a momentary short between states, it is recommended that dead times t2 and t4 be provided when switching between modes in the IC. It is unclear to the author if these modes are provided by the IC or are the responsibility of the software designer. My best educated guess – it is the responsibility of the software engineer.

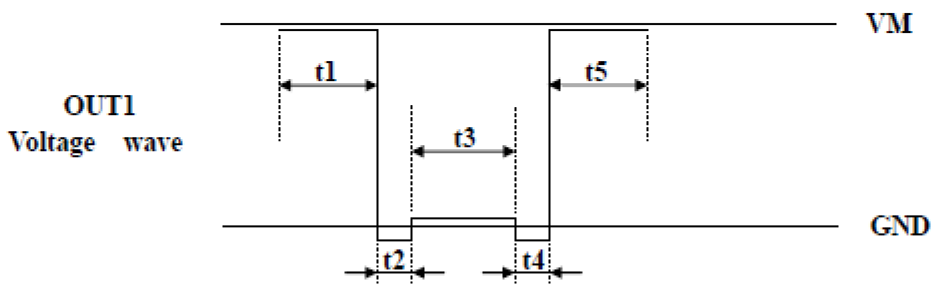
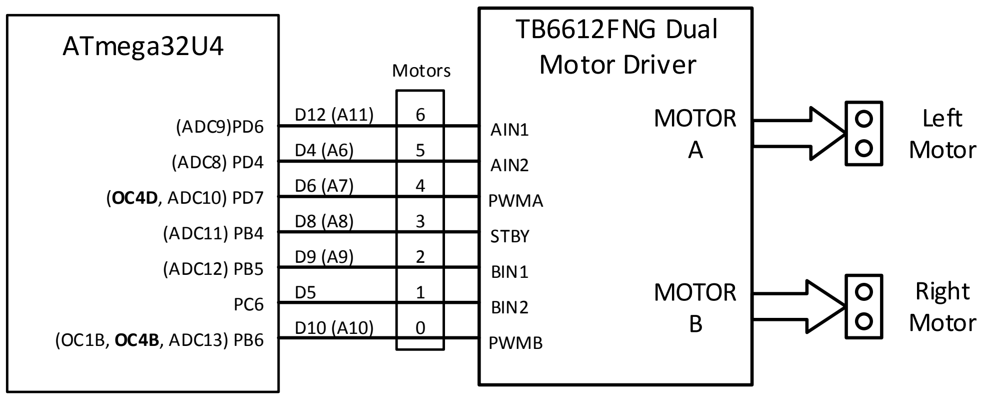


Figure 4 How to program switching between states to prevent short circuit conditions.

Sample C++ Code to Configure GPIO Ports



```
//MOTOR PINS
//Motor A PD6,4,7
DDRD |= _BV(PD7) | _BV(PD6) | _BV(PD4);       // 0xD0
// Default to Low output
PORTD &= ~(_BV(PD7) | _BV(PD6) | _BV(PD4));

//Motor B PB5,6 and STBY = PB4
DDRB |= _BV(PB6) | _BV(PB5) | _BV(PD4);       //0x70
PORTB &= ~(_BV(PB6) | _BV(PB5 | _BV(PD4)));

//And for motor B PC6 BIN2
DDRC |= _BV(PB6) ;                              //0x40
PORTC &= ~(_BV(PB6));
```

Motor Speed Control

Overview

The speed of the DC motors is controlled using pulse-width-modulation (PWM). The idea of PWM is to control the power to a motor using a high-frequency square wave. When the square wave signal is high the motor is powered ON, and when the signal is low the power is turned OFF. The speed of the motor is controlled by the fraction of time the controlling signal is ON (duty cycle = Th/Tp %, where Th = time high and Tp = clock period). The Arduino UNO can generate square waves for PWM on digital pins 3, 5, 6, 9, 10, 11.

The speed of the motors A and B are controlled by changing the duty cycle of pins PWMA and PWMB respectively. With reference to Figure 1 “Atmega32U4 to Motor Driver Interface,” the speed of motor A will be controlled by Timer 4 register OC4D and motor B by Timer 4 register OC4B. The mnemonic OCnx stands for Output Compare register nx, where n is the Timer number (0, 1, 3, and 4) and x is the Compare register (Timer 4 has four (4) output compare registers designated A, B, C, and D). We will be operating our timer using “Fast Pulse Width Modulation.” *I will tell you more about these registers and modes in the coming paragraphs.*

ATmega 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 our design the characteristics of the motor circuit act to demodulate the PWM signal.*

ATmega32U4 Timing Subsystem

The ATmega32U4 processor has 4 timer/counter (TC) modules that can be used to generate a PWM signal. They are numbered 0, 1, 3, and 4. I will use TCx convention from now on.

Timer/Counter0 is an 8-bit Timer/Counter module, with two independent Output Compare Units, and with PWM support. The Arduino uses Timer 0 to implement the millis() function.

Timer/Counter1 and **Timer/Counter3** are 16-bit Timer/Counter units with three independent double-buffered Output Compare Units.

Timer/Counter4 is the only 10-bit high speed timer on the ATmega32U4 and has a lot of advanced features, including a high precision mode, double buffering (no glitches), dead time (break before make), fault protection with noise canceling (motor stall monitoring), and even support for brushless dc motors. To keep things simple we will not be using any of these features.

Looking at Figure 1 “Atmega32U4 to Motor Driver Interface” again we see the PWMA and PWMB are associated with OC4D and OC4B respectively. OC4X denotes an output compare with TC4. The 32U4 has a more extensive timer system than the 328p and this timer (TC4) has 3 OCRs attached to it. Conveniently for us, both of these utilize the same timer making configuration a little more straightforward.

Take note though that TC4 in this microcontroller is actually a 10-bit timer. Meaning, if desired, a 1024 bit resolution could be achieved for more refined motor control. The OCRx however, is only 8-bits long and will only compare to the low 8-bits of the timer giving an easy 256 bit resolution which is consistent with arduino implementation. This means we can safely ignore the high 2 -bits of the timer as far as PWM generation is concerned.

What is Fast Pulse Width Modulation

The timing diagram for the fast PWM mode is shown in Figure 15-13. The counter is incremented until the counter value matches the **TOP** value. The counter is then cleared at the following timer clock cycle. The **TCNTn** value is in the timing diagram shown as a histogram for illustrating the single-slope operation. The diagram includes the Waveform Output in non-inverted and inverted Compare Output modes. The small horizontal line marks on the TCNTn slopes represent Compare Matches between OCRnx and TCNTx. Figure 15-3 is true for Timer/Counter 4 operation. The only difference for Timer/Counters 0, 1, and 3, is the mnemonic OCWnx, which is replaced simply by OCnx.

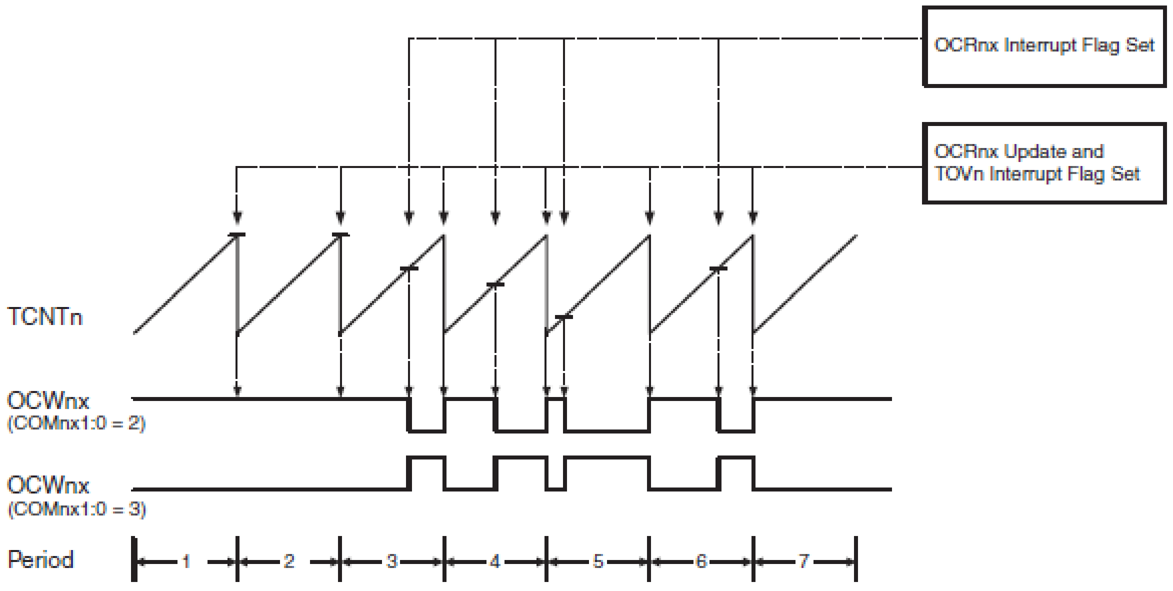
The Timer/Counter Overflow Flag (**TOVn** bit) is set each time the counter reaches TOP. In fast PWM mode, the compare unit allows generation of PWM waveforms on the OCnx pins. In our case **OC4D** for Motor A and **OC4B** for Motor B.

Table 15-1 Definitions

BOTTOM	The counter reaches the BOTTOM when it becomes 0.
MAX	The counter reaches its MAXimum value.
TOP	The counter reaches the TOP value.

For Timer/Counter 4 the **OCR4C** holds the Timer/Counter **TOP** value, i.e. the clear on compare match value. The Timer/Counter4 High Byte Register (TC4H) is a 2-bit register[1] that is used as a common temporary buffer to access the MSB bits of the Timer/Counter4 registers, if the 10-bit accuracy is used (Section 15.2.3 Registers).

Figure 15-3 Fast PWM Mode, Timing Diagram



[1] Enhanced PWM mode adds an additional 3rd bit to the TC4H register.

Fixed PWM Output Frequency

Timer/Counter 4[2]

For Timer/Counter 4 the PWM output frequency can be calculated by the following equation (Section 14.8.3).

$$f_{OCnxPWM} = \frac{f_{\text{clk\_I/O}}}{N \cdot (1 + TOP)} \qquad \text{Equation 1.0}$$

The frequency f_{OC4X} as defined by equation 1 is a function of the system clock (8 MHz), the prescaler, and TOP. The N variable represents the prescale divider and is defined in **TCCR4B** CS43:CS40 (stopped, 1, 2, 4,…16384). For our design N = 1 (no prescaler). Our TOP is defined by **OCR4C** and is fixed at its default value of 0xFF, with the 2 bits in **TC4H** set to zero.

$f_{OC4X} = f_{CLK}$ / 256 = 8 MHz / 256 = 31.25 KHz $\simeq$ 32 KHz

Calculating the PWM Duty Cycle

We will be operating our 10-bit timer/counter 4 as an 8-bit timer in Fast PWM mode.

- TOP will be defined as 0xFF = 255_{10}
- The most significant 2-bits contained in register TC4H will always be zero.
- TCNT4 will be compared to OCR4D (Motor A) and OCR4B (Motor B).
- Therefore, the **Duty Cycle = OCR4x/255,** where x equals D or B

[2] I believe the equation defined in Section 15.8.2 "Fast PWM Mode" is incorrect and have replaced with equation used for calculating the frequency throughout the rest of the datasheet.

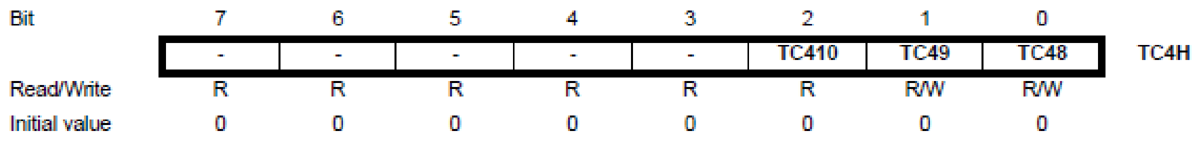
10-bit Timer/Counter 4 Register

Timer 4 is a 10-bit timer/counter. Special considerations need to be taken when writing to or reading from a 10-bit register. To write to a 10-bit register, write the most significant 2 bits to **TC4H** first, followed by the least significant byte (for example TCNT4). The TC4 register is shared by all 10-bit registers in Timer/Counter 4. One consequence of this common register, is that when you read a 10-bit register, the most significant 2-bits are saved to TC4H. Consequently, any subsequent 8-bit write operation to the least significant byte of a 10-bit register, will have this new TC4H value written to the high order bits. Again, this potentially unintended consequence can be avoided by always writing to TC4H first. For more on working with a 10-bit register read Atmel Document 7766 “8-bit AVR Microcontroller with16/32K Bytes of ISP Flash and USB Controller,” Section 15.11 “Accessing 10-bit Register.”

For our robots, the good news is that we never read a 10-bit register. Specifically, the Timer/Counter4 high byte (**TC4H**) will be always be kept at its default value of zero (0x00).

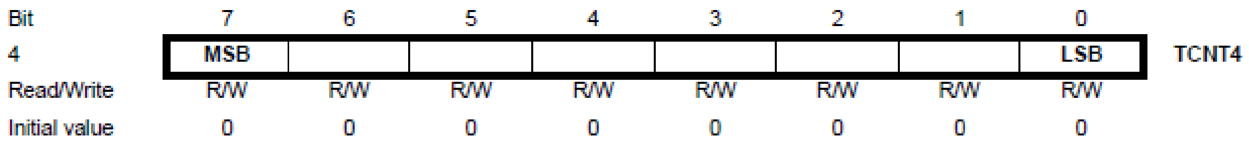
If you were wondering, **TC410** (Bit 2) is an “optional” accuracy bit for 11-bit accesses in Enhanced PWM mode. The enhanced PWM mode allows to get one more accuracy bit while keeping the frequency identical to normal mode. For more information on this topic see Section 15.6.2 “Enhanced Compare/PWM mode Timer/Counter 4” in the ATmega32U4 Datasheet.

TC4H – Timer/Counter4 High Byte



Bit	7	6	5	4	3	2	1	0	
	-	-	-	-	-	TC410	TC49	TC48	TC4H
Read/Write	R	R	R	R	R	R	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	

TCNT4 – Timer/Counter4



Bit	7	6	5	4	3	2	1	0	
4	MSB							LSB	TCNT4
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	

Although Timer 4 is a 10-bit timer/counter, we will be operating it as an 8-bit timer.

Configuring Timing/Counter4

For our *fast* PWM implementation we need:

1. To enable the Fast PWM Mode
2. define output waveform shape
3. set an appropriate timer frequency and...
4. duty cycle by configuring the OCR registers

Step 1 – Enable Fast PWM mode

Modes of operation supported by the Timer/Counter4 are: Normal mode (counter), Fast PWM Mode, Phase and PWM6 Modes as defined in Table 15-19.

Table 15-19 Waveform Generation Mode Bit Description

PWM4x	WGM41..40	Timer/Counter Mode of Operation	TOP	Update of OCR4x at	TOV4 Flag Set on
0	xx	Normal	OCR4C	Immediate	TOP
1	00	Fast PWM	OCR4C	TOP	TOP
1	01	Phase and Frequency Correct PWM	OCR4C	BOTTOM	BOTTOM
1	10	PWM6 / Single-slope	OCR4C	TOP	TOP
1	11	PWM6 / Dual-slope	OCR4C	BOTTOM	BOTTOM

We will be operating in the Fast PWM mode, so we need to set the bits to match row #2.

- PWM4x (PWM4x where x = D and B) is set in **TCCR4A** (bit 0 = 1) and **TCCR4A** (bit 0 = 1) respectively.
- WGM41 and WGM40 **TCCR4D** bits 1 and 0 are cleared (default)

Timer/Counter4 Control Register D (Section 15.12.4)

Bit	7	6	5	4	3	2	1	0	
	FPIE4	FPEN4	FPNC4	FPES4	FPAC4	FPF4	WGM41	WGM40	TCCR4D
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	

```
           7654_3210
TCCR4D = 0b0000_0000 = 0x00
TCCR4D &= ~(_BV(WGM41) | _BV(WGM40));
```

Timer/Counter4 Control Register C (Section 15.12.3)

Bit	7	6	5	4	3	2	1	0	
	COM4A1S	COM4A0S	COM4B1S	COMAB0S	COM4D1	COM4D0	FOC4D	PWM4D	TCCR4C
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	

Timer/Counter4 Control Register A (15.12.1)

Bit	7	6	5	4	3	2	1	0	
	COM4A1	COM4A0	COM4B1	COM4B0	FOC4A	FOC4B	PWM4A	PWM4B	TCCR4A
Read/Write	R/W	R/W	R/W	R/W	W	W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	

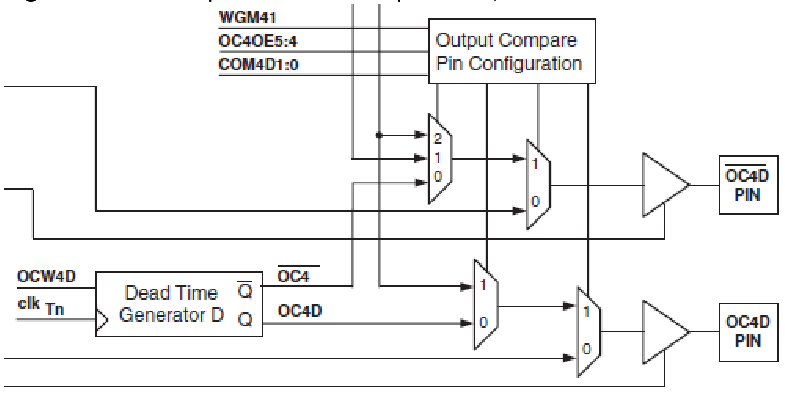
All bits shaded in blue are kept at their default value of zero (0).

Step 2 – Define output waveform shape

Timer/Counter4 Control Register A and C – Comparator Output Mode bits

To simplify the definition of the Comparator Output Mode bits located in TCCRA and TCCR4C, I am going to be define COM4D1:0 in TCCR4A. The discussion is directly applicable to the definition of COM4B1:COM4B0 in TCC4C

Figure 15-9. Compare Match Output Unit, Schematic



Comparator D Output Mode (COM4D1:COM4D0) TCCR4A **bits 3 and 2**, control the behavior of the Waveform Output (OCW4D) and the connection of the Output Compare pin (OC4D). If one or both of the COM4D1:0 bits are set, the OC4D output overrides the normal port functionality of the I/O pin it is connected to. The complementary OC4D output is connected only in PWM modes when the COM4D1:0 bits are set to “01”. Note that the Data Direction Register (DDR) bit corresponding to the OC4D pin must be set in order to enable the output driver. The function of the COM4D1:0 bits depends on the PWM4D and WGM40 bit settings. Table 15-17 shows the COM4D1:0 bit functionality when the PWM4D bit is set to a Fast PWM Mode.

Table 15-17. Compare Output Mode, Fast PWM Mode

COM4D1..0	OCW4D Behavior	OC4D Pin	$\overline{\text{OC4D}}$ Pin
00	Normal port operation.	Disconnected	Disconnected
01	Cleared on Compare Match. Set when TCNT4 = 0x000.	Connected	Connected
10	Cleared on Compare Match. Set when TCNT4 = 0x000.	Connected	Disconnected
11	Set on Compare Match. Clear when TCNT4 = 0x000.	Connected	Disconnected

Timer/Counter4 Control Register C and A: C++ Code Example

```
           7654_3210
TCCR4C = 0b0000_1001 = 0x09
TCCR4A = 0b0010_0001 = 0x21
```

```
//Setting COMD and PWM4D
TCCR4C |= (_BV(COM4D1)| _BV(PWM4D));
TCCR4C &= ~(_BV(COM4D0));
TCCR4A |= (_BV(COM4B1)| _BV(PWM4B));
TCCR4A &= ~(_BV(COM4B1));
```

Step 3 –Set an appropriate timer frequency

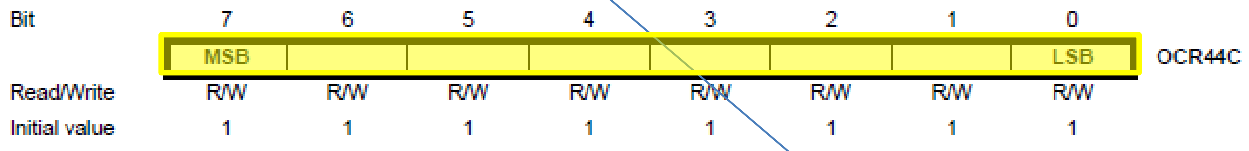
For Timer/Counter 4 the PWM output frequency can be calculated by the following equation (Section 14.8.3).

$$f_{OCnxPWM} = \frac{f_{\text{clk\_I/O}}}{N \cdot (1 + TOP)} \qquad \text{Equation 1.0}$$

The frequency f_{OC4X} as defined by equation 1 is a function of the system clock (8 MHz), the prescaler, and TOP. The N variable represents the prescale divider and is defined in **TCCR4B** CS43:CS40 (stopped, 1, 2, 4,…16384). For our design N = 1 (no prescaler). Our TOP is defined by **OCR4C** and is fixed at its default value of 0xFF, with the 2 bits in **TC4H** set to zero.

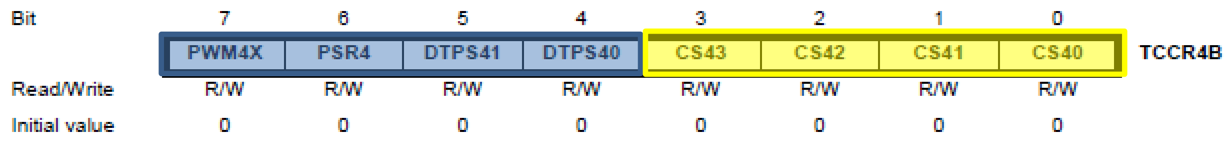
$f_{OC4X} = f_{CLK}$ / 256 = 8 MHz / 256 = 31.25 KHz ≃ 32 KHz

Timer/Counter4 Output Compare Register C (Section 15.12.10) *correct mnemonic is OCR4C*



Bit	7	6	5	4	3	2	1	0	
	MSB							LSB	OCR44C
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	1	1	1	1	1	1	1	1	

Timer/Counter4 Control Register B (Section 15.12.2)



Bit	7	6	5	4	3	2	1	0	
	PWM4X	PSR4	DTPS41	DTPS40	CS43	CS42	CS41	CS40	TCCR4B
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	

TCCR4B bits 7 to 4 are kept at their default value of zero (0). As defined in Table 15-15, we set the clock prescaler to divide by 1 by setting CS43 to CS40 equal to 0001_2.

Table 15-15. Timer/Counter4 Prescaler Select

CS43	CS42	CS41	CS40	Asynchronous Clocking Mode	Synchronous Clocking Mode
0	0	0	0	T/C4 stopped	T/C4 stopped
0	0	0	1	PCK	CK
0	0	1	0	PCK/2	CK/2
0	0	1	1	PCK/4	CK/4
0	1	0	0	PCK/8	CK/8
0	1	0	1	PCK/16	CK/16
0	1	1	0	PCK/32	CK/32
0	1	1	1	PCK/64	CK/64
1	0	0	0	PCK/128	CK/128
1	0	0	1	PCK/256	CK/256
1	0	1	0	PCK/512	CK/512
1	0	1	1	PCK/1024	CK/1024
1	1	0	0	PCK/2048	CK/2048
1	1	0	1	PCK/4096	CK/4096
1	1	1	0	PCK/8192	CK/8192
1	1	1	1	PCK/16384	CK/16384

Timer/Counter4 Control Register B and Output Compare Register C: C++ Code Example

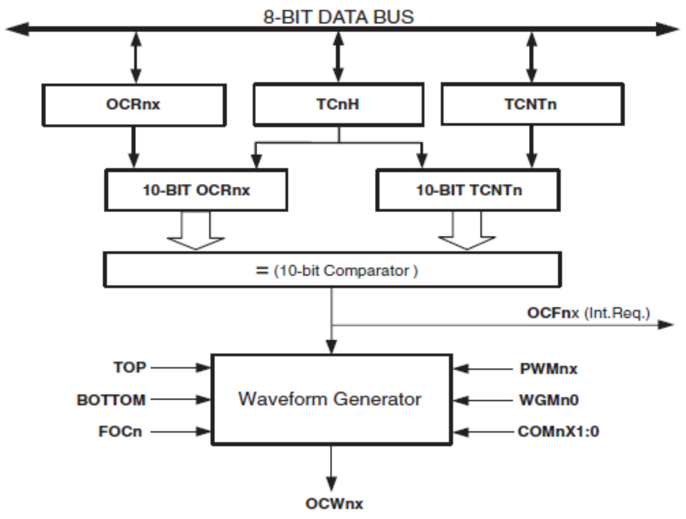
```
             7654_3210
TCCR4B = 0b0000_0001 = 0x01

OCR4C = 0xFF;

//CS4 4:0 = 0b0001;
TCCR4B |= _BV(CS40);
TCCR4B &= ~(_BV(CS43)| _BV(CS42)|_BV(CS41)); // Clear prior settings from Arduino.
```

Step 4 –Set the duty cycle by configuring the OCR registers

Calculating the PWM Duty Cycle

As illustrated in Figure 15-4, the 8-bit Timer/Counter Output Compare Registers OCR4x (where x = B or D) are compared with Timer/Counter4. On compare match the OC4x pin is cleared to 0 (see Table 15-17 Compare Output Mode, Fast PWM Mode). Write to this register to set the duty cycle of the output waveform. A compare match will also set the compare interrupt flag OCF4B after a synchronization delay following the compare event.

Figure 15-4. Output Compare Unit, Block Diagram



The **Duty Cycle = OCR4x/255,** where x equals D or B

OCR4B – Timer/Counter4 Output Compare Register D (Section 15.12.9)

Bit	7	6	5	4	3	2	1	0	
	MSB							LSB	OCR4B
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	

OCR4D – Timer/Counter4 Output Compare Register D (Section 15.12.11)

Bit	7	6	5	4	3	2	1	0	
	MSB							LSB	OCR4D
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	

Timer/Counter4 Output Compare Register: C++ Code Example

```
void setPWM(char Pin, uint8_t val){

// Only using 8-bit mode so value is from 0-255 as normal.
// This assumes proper values passed, if block should protect from bad inputs
//if (Pin == 'A'| Pin == 'B') (Pin == 'A' ) ? (OCR4D = val) : (OCR4B = val) ;
// looking at more stuff this ternary probably won't work ):
```

```
// wanted something more elegant than switch, but it works.
// would have to make own methods like Arduino has it set up to be really clean.
switch (Pin){
  case 'A':  // Due to ASCII, this is case sensitive, change if you wish.
    OCR4D = val;
    break;
  case 'B':
    OCR4B = val;
    break;
  default:
    Serial.println("Invalid Motor Pin");
    break;
  }
}
```

Fast PWM Assembly Code Example

The following code example shows how to configure timer/counter 4 for Fast PWM operation, at a frequency of 31.25 KHz.

```
Reset:
/* Test code for motor A */
 clr r0          // r0 = 0x00
 clr r1
 com r1          // r1 = OxFF
/* Test code for motors A and B */
 cbi PORTD, 7    // outputs 0 to Motor A PWM pin when timer/counter 4 disconnected
 cbi PORTB, 6    // outputs 0 to Motor B PWM pin when timer/counter 4 disconnected
                 // see section 15.11 Accessing 10-bit Register
 sts TC4H, r0    // most significant 2-bits
 sts TCNT4, r0   // 10-bit write TC4H:TCNT4 = 0x000
                 // frequency = (8MHz/prescaler)/OCR4C = 31.372 KHz (default)
 sts OCR4C, r1   // 10-bit write TC4H:OCR4C = 0x0FF
                 // duty cycle = OCR4D/OCR4C = 100%
 sts OCR4D, r1   // 10-bit write TC4H:OCR4D = 0x0FF (Motor A)
 sts OCR4B, r1   // 10-bit write TC4H:OCR4B = 0x0FF (Motor B)

 sts TCCR4B, r0  // all configuration bits to default with prescalar = 0, OFF
 ldi r16, 0x09
 sts TCCR4A, r16 // clear to manually control Motors A PWMD
 ldi r16, 0x2
 sts TCCR4A, r16 // clear to manually control Motors B PWMB

 sts TCCR4D, r0  // mode = fast PWM (default)

end of initialization

Walk:
 push r16
 ldi r16, 0x01   // configure inversion mode, reset, dead time to default = 0
 sts TCCR4B, r16 // with prescaler = 1. Motors A and B Timer/Counter 4 ON
 pop r16
 ret

Stop:
 push r16
 clr r16         // configure in fast PWM mode with prescaler = 0
 sts TCCR4B, r16 // Motors A and B Timer/Counter 4 OFF
 pop r16
 ret
```

Fast PWM C++ Code Example

Handling PWM in C++ will be divided into two functions(or sections).

1. Configure the Timer and OCR
2. Set OCR value as needed to change PWM

```
void RobotPWM(){
// Configure Motor GPIO Port Pins
//Motor A PD6,4,7
DDRD |= _BV(PD7) | _BV(PD6) | _BV(PD4);    // 0xD0
// Default to Low output
PORTD &= ~(_BV(PD7) | _BV(PD6) | _BV(PD4));

//Motor B PB5,6 and STBY = PB4
DDRB |= _BV(PB6) | _BV(PB5) | _BV(PD4);         //0x70
PORTB &= ~(_BV(PB6) | _BV(PB5 | _BV(PD4)));

//And for motor B PC6 BIN2
DDRC |= _BV(PB6) ;                             //0x40
PORTC &= ~(_BV(PB6));

// Configure Timer4 for PWMs
// Motor A on PD7 (OC4D)
// Motor B on PB6 (OC4B)
// Ignore 10-bit mode for ease of use

// Need to configure Timer4 for fast PWM
// PWM4D and PWM4B set with WGM4 1:0 = 0b00
// Setting WGM = 00
TCCR4D &= ~(_BV(WGM41) | _BV(WGM40));
// Set PD7 and PB6 as outputs
// I have also added digital pins since they are part of the same system
// If I want the PWM then I want the digitals also
//Setting PWM4B and COMB
TCCR4A |= (_BV(COM4B1)| _BV(PWM4B));
TCCR4A &= ~(_BV(COM4B1));
//Setting PWM4D and COMD
TCCR4C |= (_BV(COM4D1)| _BV(PWM4D));
TCCR4C &= ~(_BV(COM4D0));                ← Error in Thomas C++ Code

//SetPrescaler - turn on timer
//Assumes *Mhz external with default fuses (making Fio = 1Mhz)
//TB66612FNG says wants PWM Freq <= 100k

OCR4C = 0xFF;                            ← Missing in Thomas C++ Code

//CS4 4:0 = 0b0001;                      ← Error in Thomas C++ Code
TCCR4B |= _BV(CS40);
TCCR4B &= ~(_BV(CS43)| _BV(CS42)|_BV(CS41)); // Clear prior settings from Arduino.
}
```

With the configuration ready we can make a function to set our PWM by change the OCR threshold.

```
void setPWM(char Pin, uint8_t val){

// Only using 8-bit mode so value is from 0-255 as normal.
// This assumes proper values passed, if block should protect from bad inputs
//if (Pin == 'A'| Pin == 'B') (Pin == 'A' ) ? (OCR4D = val) : (OCR4B = val) ;
// looking at more stuff this ternary probably won't work ):
```

```
// wanted something more elegant than switch, but it works.
// would have to make own methods like Arduino has it set up to be really clean.
switch (Pin){
  case 'A':  // Due to ASCII, this is case sensitive, change if you wish.
    OCR4D = val;
    break;
  case 'B':
    OCR4B = val;
    break;
  default:
    Serial.println("Invalid Motor Pin");
    break;
  }
}
```

Appendix A: Timer/Counter 4 Register Summary

Timer/Counter4 is a monster with five (5) control registers for configuring the timer/counter.

- TCCR4A – Timer/Counter4 Control Register A
- TCCR4B – Timer/Counter4 Control Register B
- TCCR4C – Timer/Counter4 Control Register C
- TCCR4D – Timer/Counter4 Control Register D
- TCCR4E – Timer/Counter4 Control Register E

Two registers used to make the 10-bit timer/counter.

- TC4H – Timer/Counter4 High Byte
- TCNT4 – Timer/Counter4

Four output compare registers

- OCR4A – Timer/Counter4 Output Compare Register A
- OCR4B – Timer/Counter4 Output Compare Register B
- OCR4C – Timer/Counter4 Output Compare Register C
- OCR4D – Timer/Counter4 Output Compare Register D

Two register to support polling and interrupts

- TIMSK4 – Timer/Counter4 Interrupt Mask Register
- TIFR4 – Timer/Counter4 Interrupt Flag Register

And one register unique to timer/counter4

- DT4 – Timer/Counter4 Dead Time Value

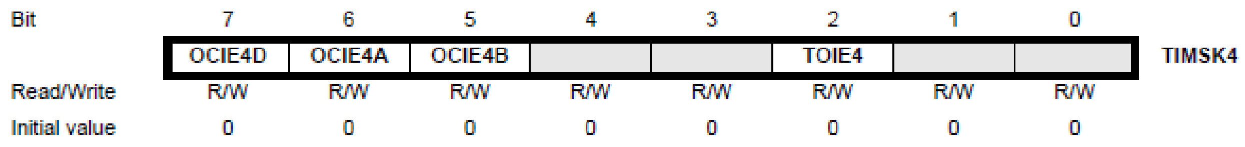
Unused Registers

The following registers are not used in our application and are kept at their default values (0x00).

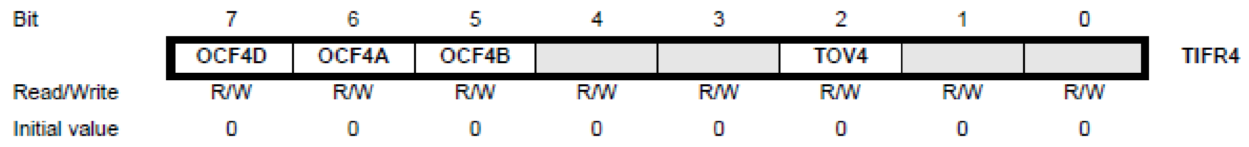
- TCCR4E – Timer/Counter4 Control Register E
- OCR4A – Timer/Counter4 Output Compare Register A
- TIMSK4 – Timer/Counter4 Interrupt Mask Register
- TIFR4 – Timer/Counter4 Interrupt Flag Register
- DT4 – Timer/Counter4 Dead Time Value

Timer/Counter4 Interrupt Mask and Flag Registers (14.10.17, 14.10.19)

The OCF4B and OCF4D flag bits in the TIFR4 register will be set on compare match. As currently configured and defined in section "Timer/Counter4 Control Registers" the OCF4B and OC4D pins are cleared to 0 on compare match.



Bit	7	6	5	4	3	2	1	0	
	OCIE4D	OCIE4A	OCIE4B			TOIE4			TIMSK4
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	



Bit	7	6	5	4	3	2	1	0	
	OCF4D	OCF4A	OCF4B			TOV4			TIFR4
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	

TIMSK4 address = (0x72)
TIFR4 address = 0x19 (0x39)

Appendix B: Design Example – Timer/Counter 0

This section takes a closer look at ATmega32U4 timing subsystem. To simplify the discussion, I will assume the use of an 8-bit Timer. This is a reasonable simplification because all 3DoT timer/counters are programmed to operate as 8-bit timer/counters. In most design examples, I will further assume 8-bit Timer 0 operating in one of four (4) PWM modes, with the output coming from an output compare register (OCnx).

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 $f_{OC0A.}$ for the frequency of the 8-bit timer 0 output compare register A. This specific design example translates nicely to the more general cases used to configure 16-bit timer/counter 1 and 10-bit timer/counter 4 .

8-bit Timer/Counter 0 Subsystem

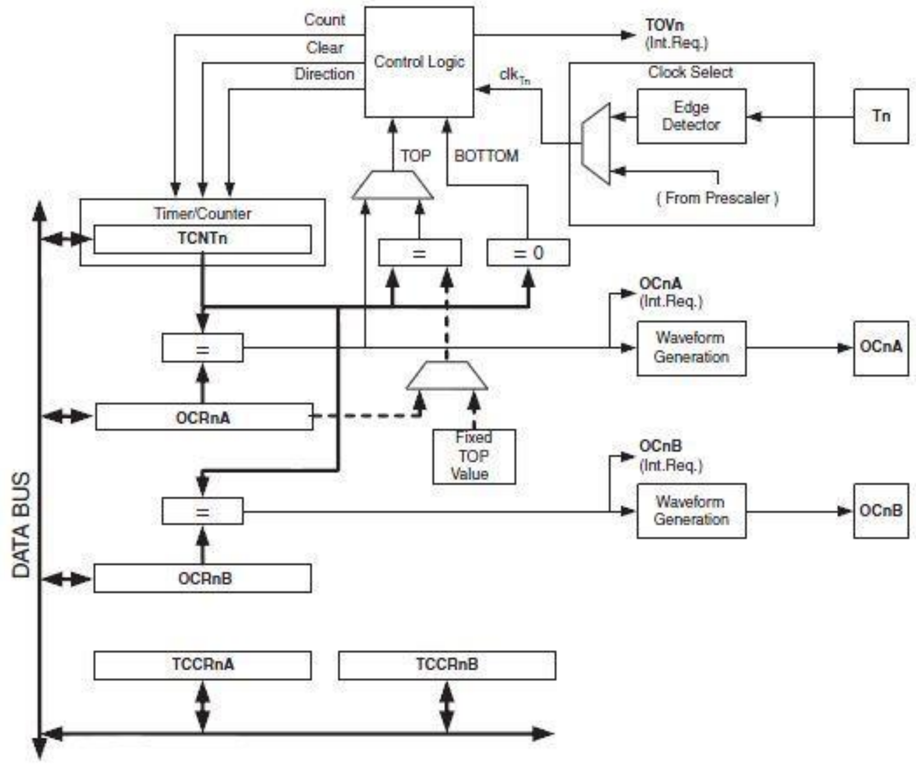


Figure 1 8-bit Timer/Counter 0 Subsystem Block Diagram

Clock Source

All our design examples will assume operation of the ATmega32U4 within the context of the Arduino system. Specifically, our system clock source is a crystal input XTAL1/TOSC1 and XTAL2/TOSC2.

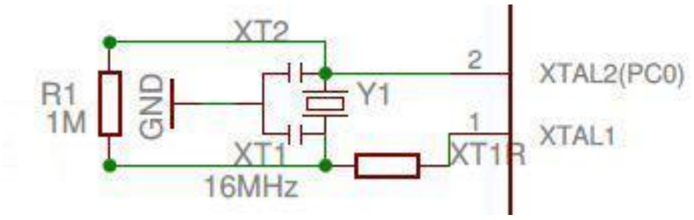


Figure 2 Arduino/ATmega32U4 System Clock

For this design implementation the following will always be true (The interested student is invited to read Section 14.3 “Timer/Counter Clock Sources” in the ATmega32U4 datasheet for why this is true).

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

and therefore...

$f_{CLK} = f_{I/O} = 16\ MHz$ 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.

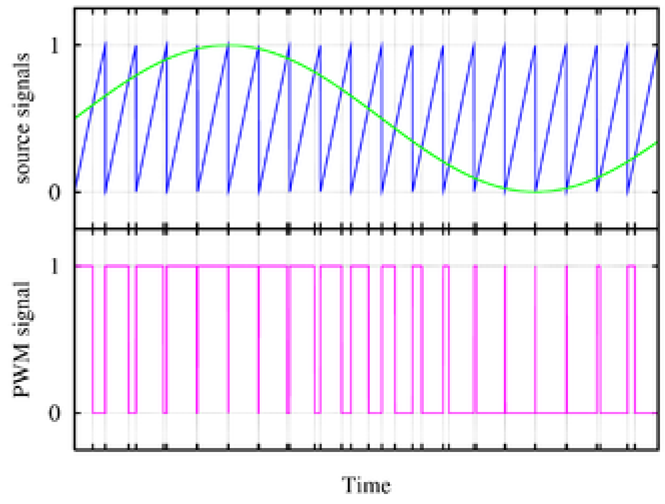
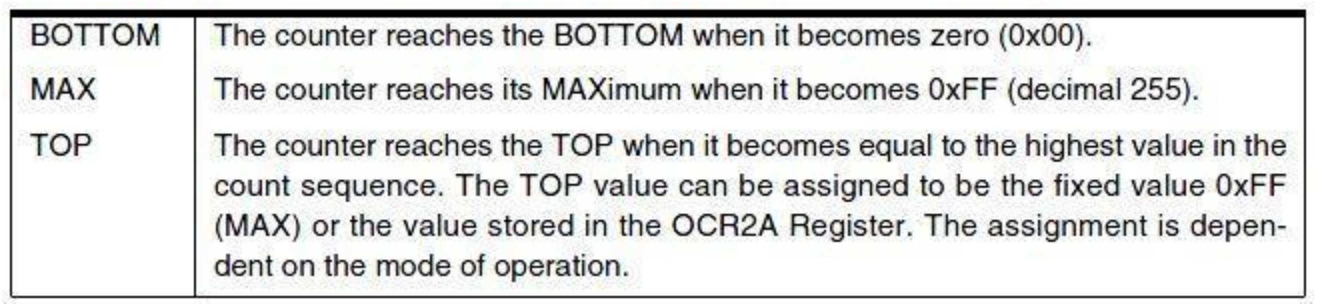


Figure 3 An example analog signal and a pulse width modulated representation.

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

Bottom, Max Top

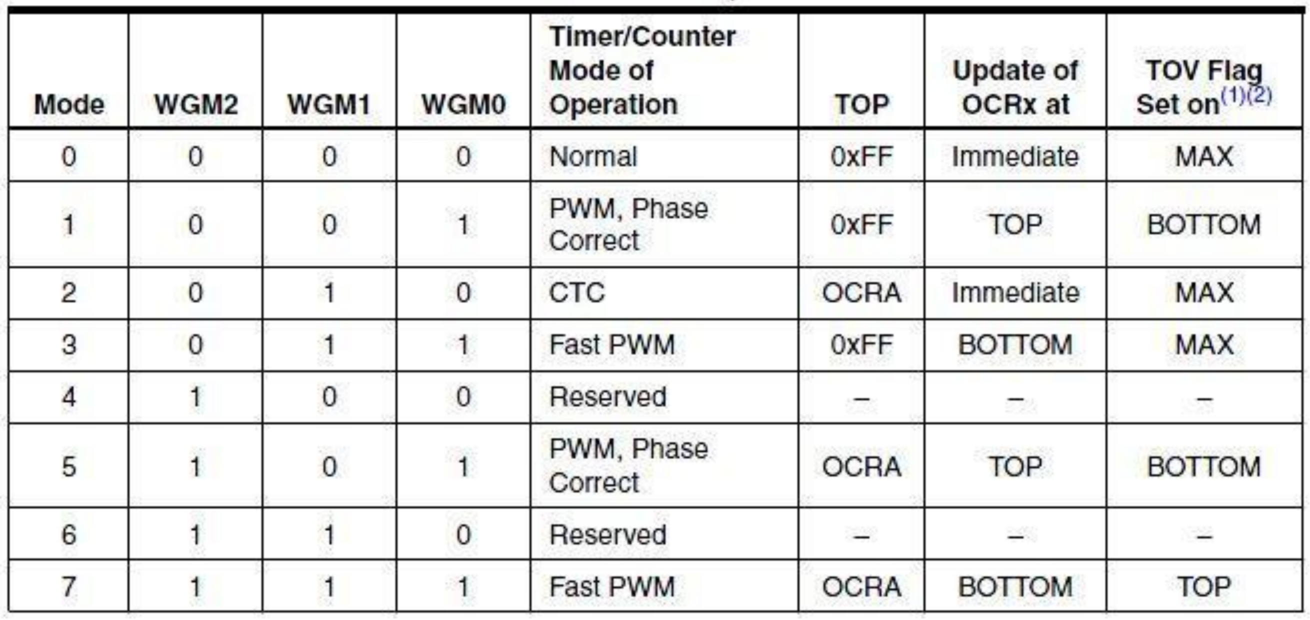


BOTTOM	The counter reaches the BOTTOM when it becomes zero (0x00).
MAX	The counter reaches its MAXimum when it becomes 0xFF (decimal 255).
TOP	The counter reaches the TOP when it becomes equal to the highest value in the count sequence. The TOP value can be assigned to be the fixed value 0xFF (MAX) or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

Waveform Generation Modes

On the ATmega32U4, 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



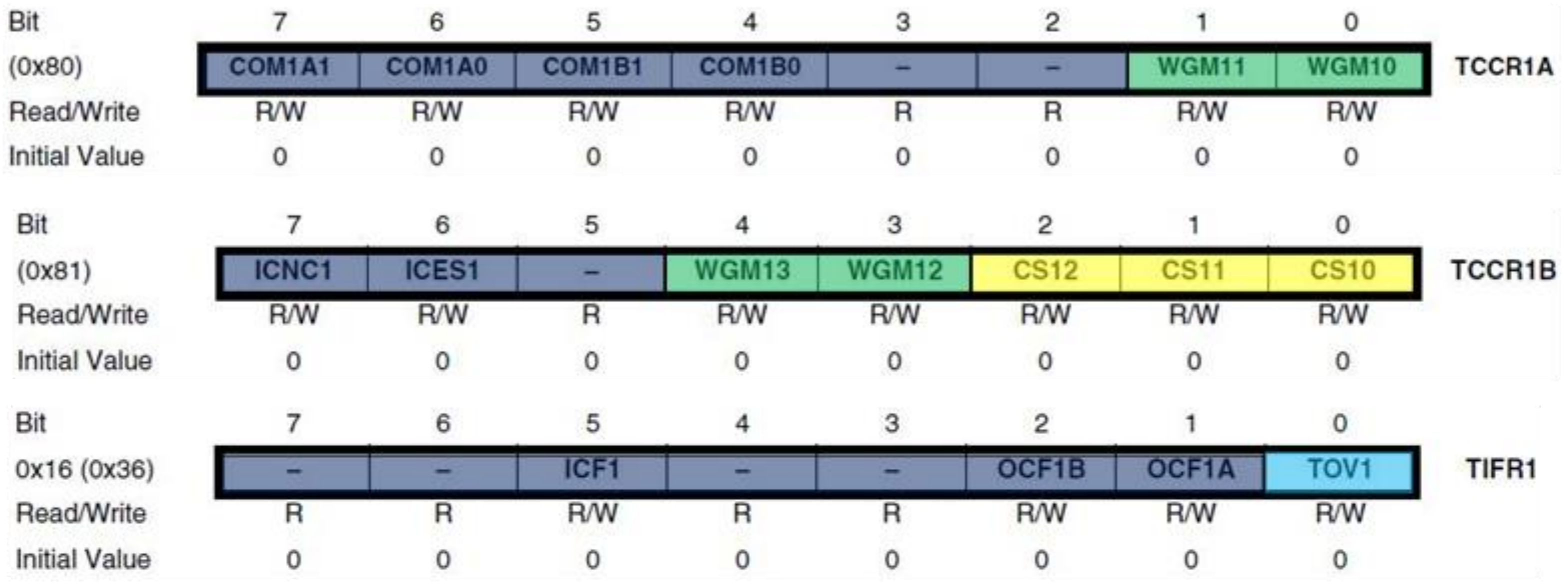
Mode	WGM2	WGM1	WGM0	Timer/Counter Mode of Operation	TOP	Update of OCRx at	TOV Flag Set on(1)(2)
0	0	0	0	Normal	0xFF	Immediate	MAX
1	0	0	1	PWM, Phase Correct	0xFF	TOP	BOTTOM
2	0	1	0	CTC	OCRA	Immediate	MAX
3	0	1	1	Fast PWM	0xFF	BOTTOM	MAX
4	1	0	0	Reserved	–	–	–
5	1	0	1	PWM, Phase Correct	OCRA	TOP	BOTTOM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

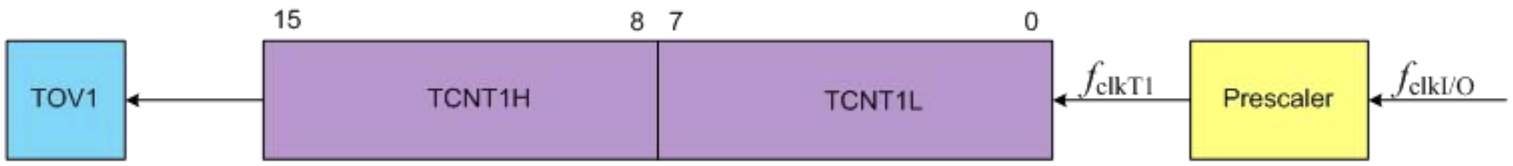
Normal Mode (WGM 1 bits 3:0 = 0000_2)



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	



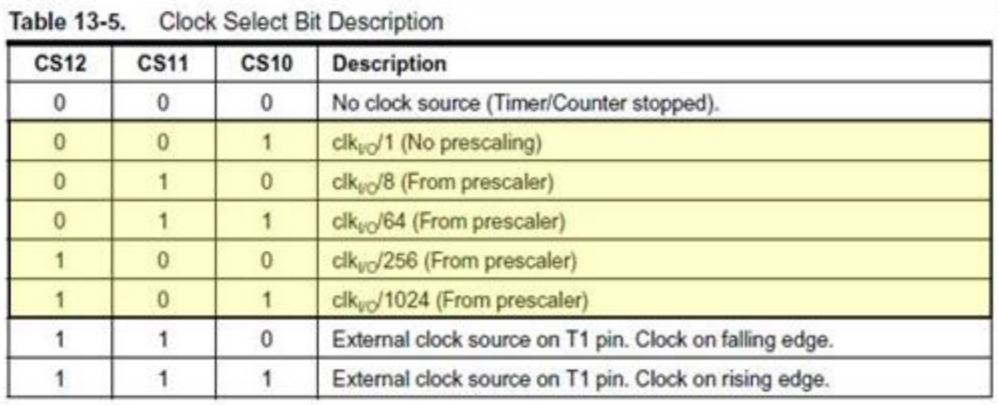


Table 13-5. Clock Select Bit Description

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

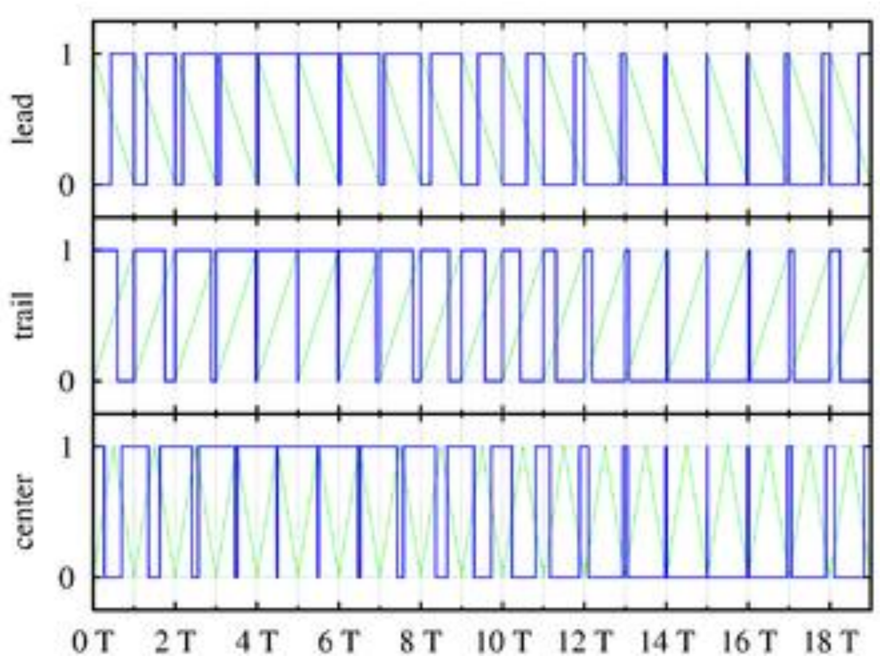


Figure 5 Three types of PWM signals (blue): leading edge modulation (top), trailing edge modulation (middle) and centered pulses (both edges are modulated, bottom). The green lines are the sawtooth waveform (first and second cases) and a triangle waveform (third case) used to generate the PWM waveforms 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. *ATmega32U4 Fast PWM inverting modes 3 and 7.*
2. The lead edge can be fixed and the **tail edge** modulated. *ATmega32U4 Fast PWM non-inverting modes 3 and 7.*
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. *ATmega32U4 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. *ATmega32U4 CTC mode 2*

Note: *ATmega32U4 modes 4 and 6 are reserved (i.e., undefined)*

ATmega32U4 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

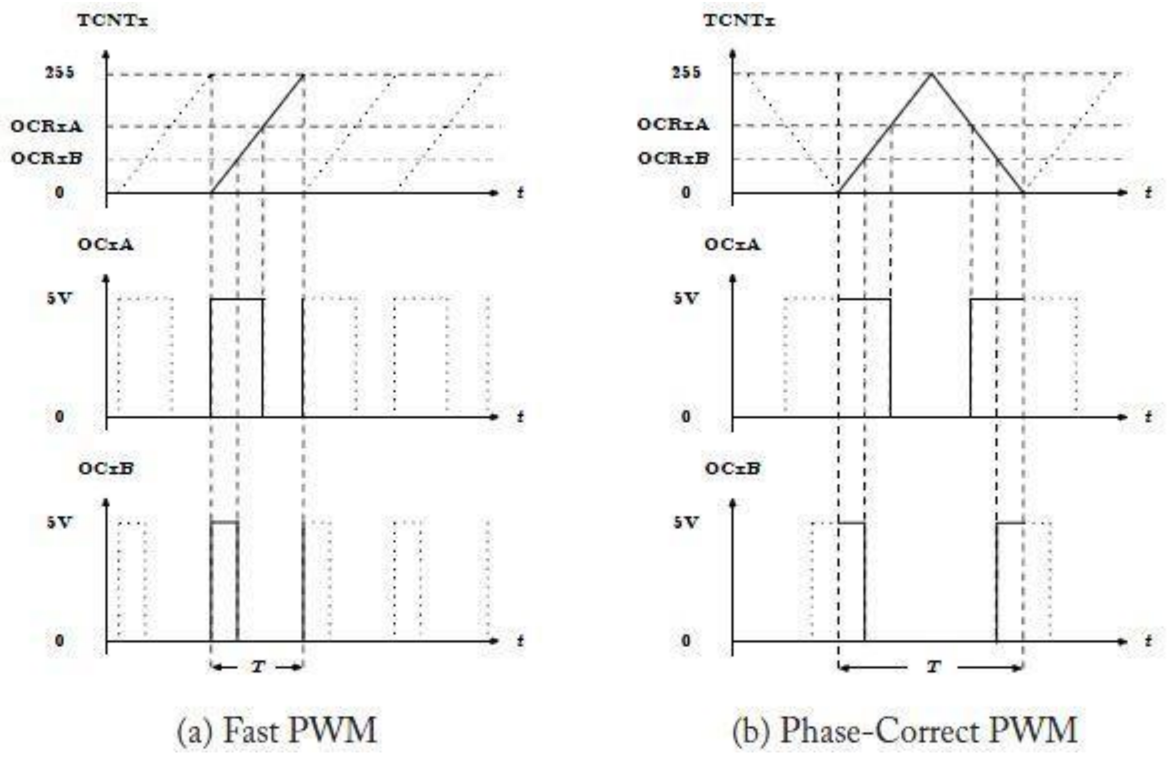


Figure 6 Non-inverting Timer Modes 3 and 1

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 0 Subsystem Block Diagram” it is seen that:

$f_{T0} = f_{I/O} / N$ 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 OC0B output is changed on the next clock cycle it can further be shown that:

$f_{OC0B} = f_{T0} / 256$ eq. 4

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

$f_{OC0B} = f_{CLK} / N*256$ eq. 5

The general form given by the equation:

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

The 3DoT motor design sets N = 1:

$f_{OC0B} = f_{CLK} / 256 = 8 \text{ MHz} / 256 = 31.25 \text{ KHz} \simeq 32 \text{ KHz}$ 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 $T_{OC0B} = 256/f_{CLK}$, 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:

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

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.

$f_{OC0B} = f_{CLK} / 510 = 8\text{ MHz} / 510 = 15.686\text{ KHz}$

The period of the PWM waveform is therefore $T_{OC0B} = 510/f_{CLK}$, 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.

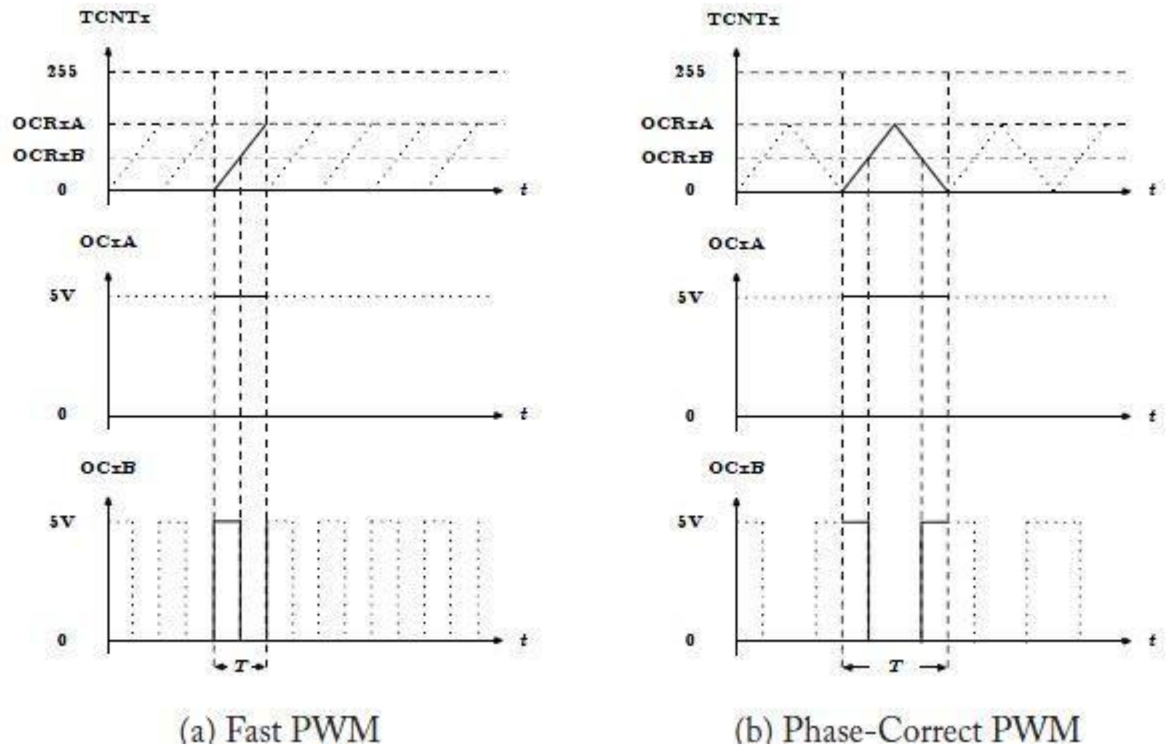
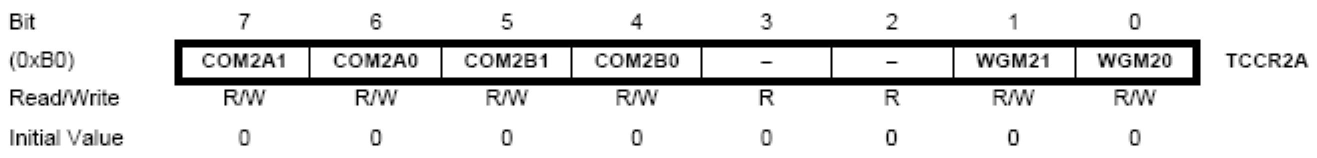


Figure 7 Non-inverting Timer Modes 7 and 5

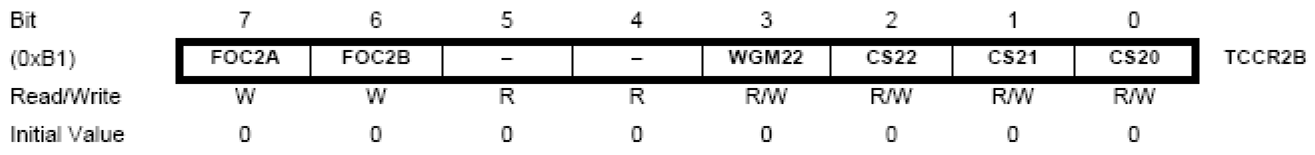
Both of these modes effectively disable the OCnA pin functionality at the benefit of increasing the PWM frequency dramatically. In both cases, the TCNT register will count up to the OCRnA value, and then either reset to 0 or start counting down toward 0. The only comparison that matters is that to OCRnB, 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_{OC0B} 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	

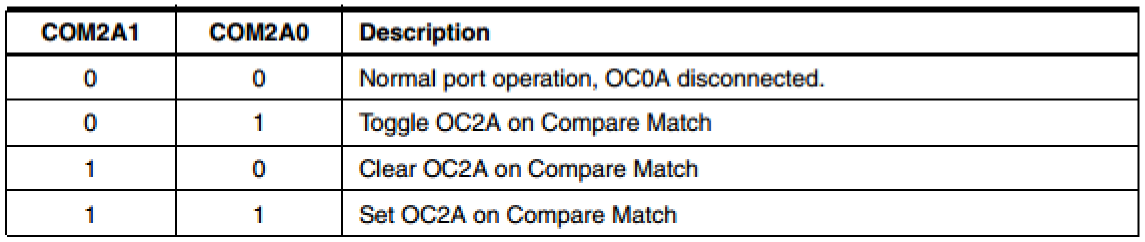


Bit	7	6	5	4	3	2	1	0	
(0xB1)	FOC2A	FOC2B	–	–	WGM22	CS22	CS21	CS20	TCCR2B
Read/Write	W	W	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Timer/Counter Control Register A (TCCR0A)

Inverting versus Non-inverting Modes (COM0A1, COM0A0 and COM0B1, COM0A0)

Compare Output Mode bits (COM0A1 and COM0A0 -- Timer/Counter 0 Output Compare Register A used as an example) define if the mode is non-inverting (COM0A1 = 1, COM0A0 = 0) or inverting (COM0A1 = 1, COM0A0 = 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 (TCCR0B)

Timer/Counter Prescaler (CS02, CS01, CS00)

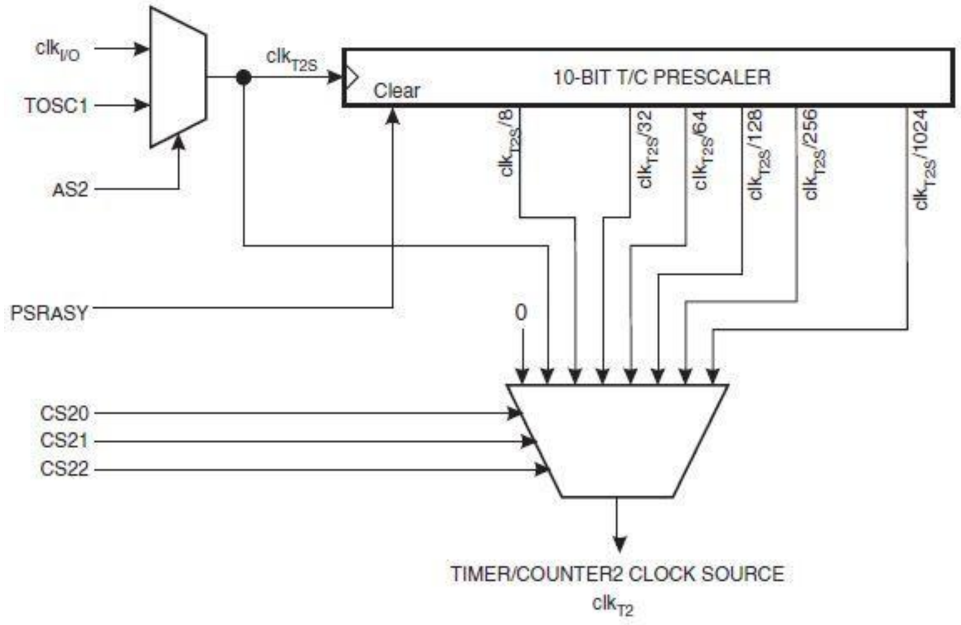


Figure 17-12 Prescaler for Timer/Counter0

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 (FOC0A and FOC0B)

The FOC0A and FOCB bits are only active when the WGM bits specify a non-PWM mode.

Appendix C: Motor Control Using ATmega32U4 16-bit Timer/Counter 1

What is Fast Pulse Width Modulation

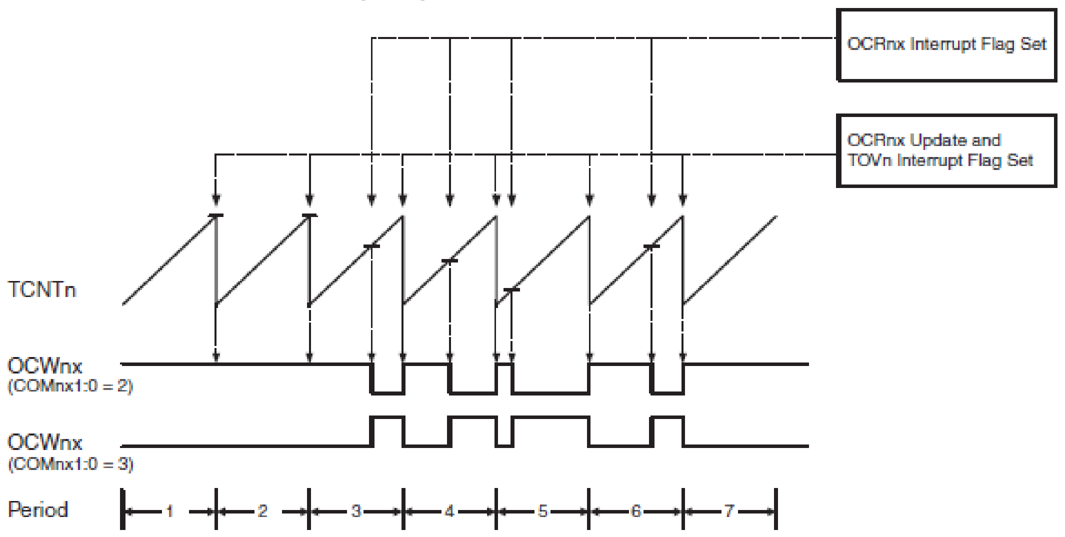
The timing diagram for the fast PWM mode is shown in Figure 15-13. The counter is incremented until the counter value matches the **TOP** value. The counter is then cleared at the following timer clock cycle. The **TCNTn** value is in the timing diagram shown as a histogram for illustrating the single-slope operation. The diagram includes the Waveform Output in non-inverted and inverted Compare Output modes. The small horizontal line marks on the TCNTn slopes represent Compare Matches between OCRnx and TCNTx. Figure 15-3 is true for Timer/Counter 4 operation. The only difference for Timer/Counters 0, 1, and 3, is the mnemonic OCWnx, which is replaced simply by OCnx.

Table 15-1 Definitions

BOTTOM	The counter reaches the BOTTOM when it becomes 0.
MAX	The counter reaches its MAXimum value.
TOP	The counter reaches the TOP value.

For Timer/Counter 1 the **TOP** value is defined by the OCR1A 16-bit register or given a fixed value of **0x00FF**, 0x01FF, or 0x03FF (WGMn3:0 = 5, 6, or 7).

Figure 15-3 Fast PWM Mode, Timing Diagram



Fixed PWM Output Frequency

Timer/Counter 1

For Timer/Counter 1 the PWM output frequency can be calculated by the following equation (Section 14.8.3).

$$f_{OCnxPWM} = \frac{f_{\text{clk\_I/O}}}{N \cdot (1 + TOP)} \qquad \text{Equation 1.0}$$

The N variable represents the prescaler divider (1, 8, 64, 256, or 1024). For our design N = 1 (no prescaler). Our TOP is fixed at 0xFF.

$f_{OC2B} = f_{CLK}$ / 256 = 8 MHz / 256 = 31.25 KHz ≃ 32 KHz

Calculating the PWM Duty Cycle

We will be operating all our timer/counters as 8-bit timers in Fast PWM mode.

- For all timers TOP will be defined as 0xFF
- For 16-bit timer/counter 1
    - The most significant 8-bits contained in TCNT1H will always be zero.
    - TCNT1L will be compared to OCR1AL
    - Therefore, the **Duty Cycle = OCR1AL/255**

Fast PWM Code

The following code example shows how to configure the 16-bit timer/counter 1 and 10-bit timer/counter 4 for Fast PWM operation, at a frequency of 31.25 KHz.

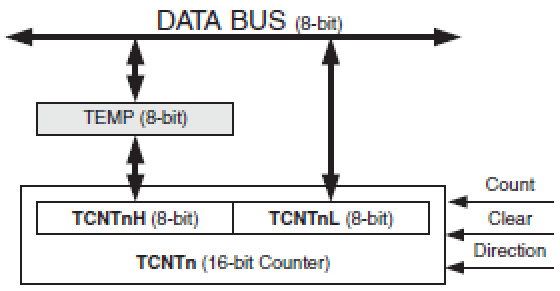
```
Reset:
/* Motor Test Code */
 clr r0          // r0 = 0x00
 clr r1
 com r1          // r1 = OxFF
 sts TCNT1H, r0  // most significant 8-bits (write to TEMP register)
 sts TCNT1L, r0  // 16-bit write TCNT1H:TCNT1L = TEMP:TCNT1L = 0x0000
                 // duty cycle = OCR1AL/0xFF = 100%
 sts OCR1AL, r0  // 16-bit write OCR1AH:OCR1AL = TEMP:OCR1AL = 0x00FF)
 ldi r16, 0x81   // clear output on compare match, configure in fast PWM
 sts TCCR1A, r16
 ldi r16, 0x08   // configure in fast PWM mode with prescaler = 0 OFF
 sts TCCR1B, r16
 sts TIMSK1, r0  // disable local timer/counter1 output compare match
                 // interrupt enable. Enabled by Arduino bootloader for
                 // servo library on reset.

end of initialization

Walk:
 push r16
 ldi r16, 0x09   // configure in fast PWM mode
 sts TCCR1B, r16 // with prescaler = 1. Motor A Timer/Counter 1 ON
 pop r16
 ret

Stop:
 push r16
 clr r16         // configure in fast PWM mode with prescaler = 0
 sts TCCR1B, r16 // Motor A Timer/Counter 1 OFF
 pop r16
 ret
```

Register Definitions

Timer/Counter 1 Registers

Timer 1 is a 16-bit timer/counter. Special considerations need to be taken when writing to or reading from a 16-bit register. To do a 16-bit write, the high byte must be written before the low byte (see Figure 14-2). For a 16-bit read, the low byte must be read before the high byte. For more on working with a 16-bit register read Atmel Document 7766 "8-bit AVR Microcontroller with16/32K Bytes of ISP Flash and USB Controller," Section 14.2 "Accessing 16-bit Register."

Figure 14-2 TEMP High byte Register



For our robots, the good news is that we never read a 16-bit register. Specifically, the Timer/Counter1 high byte (**TCNT1H**) will be always be kept at its default value of zero (0x00).

Timer/Counter 1

Bit	7	6	5	4	3	2	1	0	
	TCNT1[15:8]								TCNT1H
	TCNT1[7:0]								TCNT1L
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	

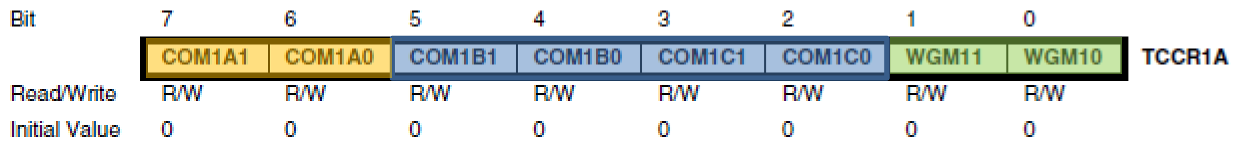
As previously mentioned, we will only be working with TCNT1L (TCNT1H = 0x00).

Timer/Counter1 Output Compare Register A (Section 14.10.9)

Bit	7	6	5	4	3	2	1	0	
	OCR1A[15:8]								OCR1AH
	OCR1A[7:0]								OCR1AL
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	

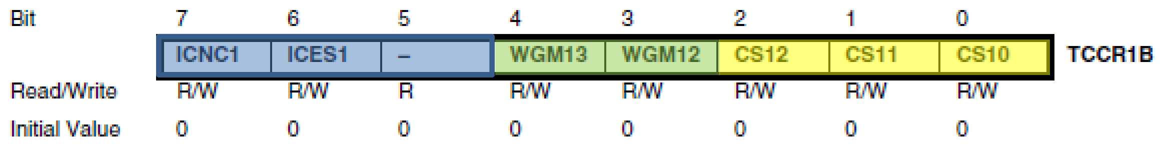
The 8-bit Timer/Counter Output Compare Register A contains data to be continuously compared with Timer/Counter1. Write to register OCR1AL to set the duty cycle of the output waveform. A compare match will also set the compare interrupt flag OCF1A after a synchronization delay following the compare event.

Timer/Counter1 Control Register A and B (14.10.1, 14.10.3)



Bit	7	6	5	4	3	2	1	0	
	COM1A1	COM1A0	COM1B1	COM1B0	COM1C1	COM1C0	WGM11	WGM10	TCCR1A
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	

```
             76543210
TCCR1A  =  0b10000001  =  0x81
```

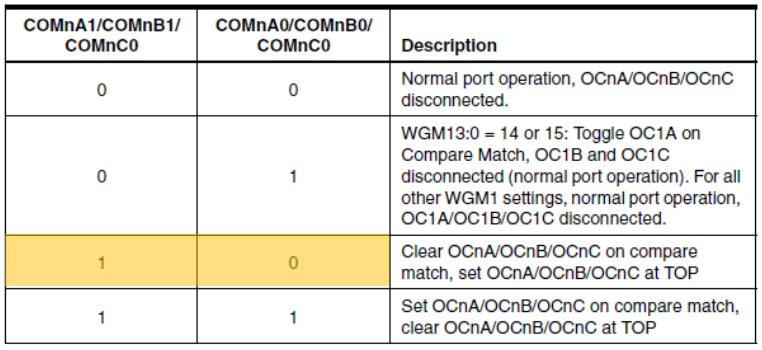


Bit	7	6	5	4	3	2	1	0	
	ICNC1	ICES1	–	WGM13	WGM12	CS12	CS11	CS10	TCCR1B
Read/Write	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

```
             76543210
TCCR1B  =  0b00001001  =  0x09
```

Timer/Counter 1 Comparator A Output Mode (**COM1A1:COM1A0**) TCCR1A **bits 7 and 6**, control the behavior of the Output Compare pin (OC1A) as defined in Table 14-3. For Fast PWM modes the COM1A1:0 bits are set to "**10_2**".

Table 14-3. Compare Output Mode, Fast PWM

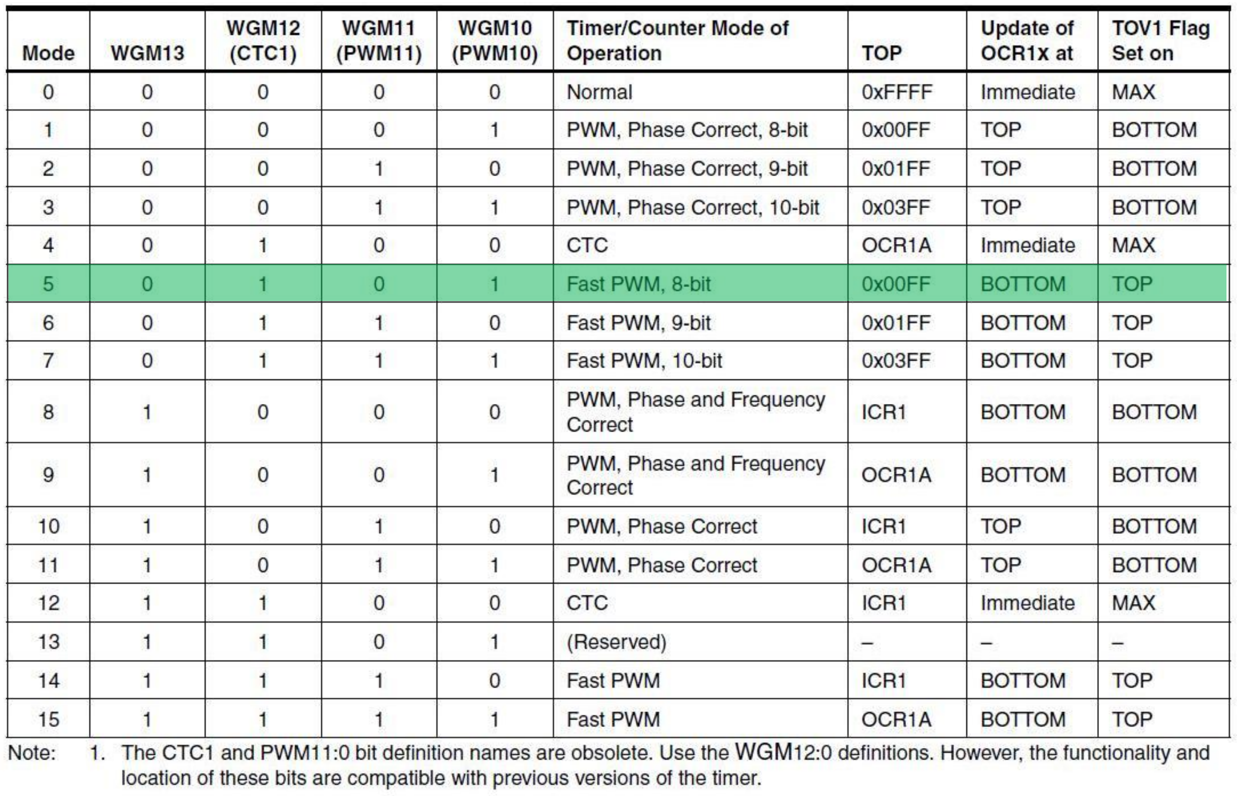


COMnA1/COMnB1/ COMnC0	COMnA0/COMnB0/ COMnC0	Description
0	0	Normal port operation, OCnA/OCnB/OCnC disconnected.
0	1	WGM13:0 = 14 or 15: Toggle OC1A on Compare Match, OC1B and OC1C disconnected (normal port operation). For all other WGM1 settings, normal port operation, OC1A/OC1B/OC1C disconnected.
1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

TCCR1A **bits 5 to 4** and TCCR1B **bits 7 to 5** are kept at their default value of zero (0).

Timer/Counter 1 Waveform Generation Mode (**WGM13:0**) TCCR1A **bits 1 and 0**, and TCCR1B **bits 4 and 3** configure the timer/counter for operation in one of 16 modes as defined in Table 14-5. To configure as a "Fast PWM, 8-bit" timer the WGM13:0 bits are set to "**0101_2**".

Table 14-5. Waveform Generation Mode Bit Description

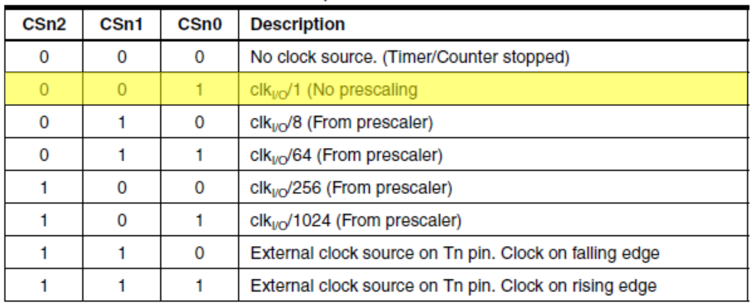


Mode	WGM13	WGM12 (CTC1)	WGM11 (PWM11)	WGM10 (PWM10)	Timer/Counter Mode of Operation	TOP	Update of OCR1x at	TOV1 Flag Set on
0	0	0	0	0	Normal	0xFFFF	Immediate	MAX
1	0	0	0	1	PWM, Phase Correct, 8-bit	0x00FF	TOP	BOTTOM
2	0	0	1	0	PWM, Phase Correct, 9-bit	0x01FF	TOP	BOTTOM
3	0	0	1	1	PWM, Phase Correct, 10-bit	0x03FF	TOP	BOTTOM
4	0	1	0	0	CTC	OCR1A	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	ICR1	BOTTOM	BOTTOM
9	1	0	0	1	PWM, Phase and Frequency Correct	OCR1A	BOTTOM	BOTTOM
10	1	0	1	0	PWM, Phase Correct	ICR1	TOP	BOTTOM
11	1	0	1	1	PWM, Phase Correct	OCR1A	TOP	BOTTOM
12	1	1	0	0	CTC	ICR1	Immediate	MAX
13	1	1	0	1	(Reserved)	–	–	–
14	1	1	1	0	Fast PWM	ICR1	BOTTOM	TOP
15	1	1	1	1	Fast PWM	OCR1A	BOTTOM	TOP

Note: 1. The CTC1 and PWM11:0 bit definition names are obsolete. Use the WGM12:0 definitions. However, the functionality and location of these bits are compatible with previous versions of the timer.

Timer/Counter 1 Clock Select (CS12:0) TCCR1B bits 2, 1, and 0 set the clock prescaler as defined in Table 14-6. To run at the frequency of the system clock (no prescaling) the CS12:0 bits are set to "001_2."

Table 14-6. 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

Timer/Counter1 Interrupt Mask and Flag Registers (14.10.17, 14.10.19)

When are the motors off?

As discussed in section "Timer/Counter1 Output Compare Register A" the OCF1A flag bit in the TIFR1 register will be set on compare match. As currently configured and defined in section "Timer/Counter1 Control Register A and B" the OC1A pin is cleared to 0 on compare match. Consequently, the motors will be turned off.

Bit	7	6	5	4	3	2	1	0	
	–	–	ICIE1	–	OCIE1C	OCIE1B	OCIE1A	TOIE1	TIMSK1
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	

Bit	7	6	5	4	3	2	1	0	
	–	–	ICF1	–	OCF1C	OCF1B	OCF1A	TOV1	TIFR1
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	

TIMSK1 address = (0x6F)
TIFR1 address = 0x16 (0x36)

The OCF1 flag is set in the timer clock cycle after the counter (TCNT1 value matches the Output Compare Register A (OCR1A). OCF1A is automatically cleared when the Output Compare Match A Interrupt Vector is executed. Alternatively, OCF1A can be cleared by writing a logic one to its bit location.

Unused Registers

The following registers are not used in our application and are kept at their default values (0x00).

- TCCR1C – Timer/Counter1 Control Register C
- TCNT1H – Timer/Counter1 High Byte
- OCR1B – Timer/Counter1 Output Compare Register A
- OCR1C – Timer/Counter1 Output Compare Register B
- ICR1H and ICR1L – Input Capture Register 1