PWM On The ATmega328

**INTRODUCTION:**

    This tutorial is a mirror of the ATmega8 tutorial.  Because the registers for this feature are completely different I have split it into 2 complete chapters.

    We know that the AVR lets us turn outputs on and off, but, what if we want to control the brightness of an LED or the speed of a motor?  Well the easiest way to do this is to change the voltage of the output but we can fake it.  If we take a voltage source and connect it for a second and then disconnect it  for a second it will .... well be on for a second then off for a second. BUT, if we manage to speed this up so that we connect/disconnect the power supply over 1000 times a second we can fool the power supply into thinking that our signal is half the value of the power supply. This is called Pulse Width Modulation or PWM for short.

**THEORY OF OPERATION:**



*Figure 1: ATmega168/328 PWM Pins*

    When you take a square wave, its on for a while and off for a while.  If we divide the on by the off and multiply by 100% we will get what is called a duty cycle.

Duty\_Cycle = [ON\_time / (ON\_time + OFF\_time) ] \* 100

    So if we are on for 1ms and then off for 1ms we will end up with a 50% duty cycle;  If we are on for 1ms and off for 3ms we end up with a 25% duty cycle.

Output\_Voltage = Duty\_Cycle \* Input\_Voltage

    Now if we take our duty cycle and multiply it by our voltage we will get the output voltage.  So if we have a 5V power supply and we activate a PWM on a 25% duty cycle we will make an analog device behave as if it was receiving a 1.25V signal. Cool eh!?

    Much like the counter functions PWM can be simulated in software however, the hardware version is preferred because it just sort of does its own thing and, with very little coding you can get a constant square wave going.

    Remember the Prescaler? well its back in the PWM.  And much like in the counter, its roll is to slow things down.  This is good because it allows us to run the PWM at different frequencies.  This is important because some devices are sensitive to PWM speeds.

A motor for example will get hot if the PWM waveform is too fast, and will jitter if the PWM is too slow.  Since I already planted the question in your head, the answer is start at 10kHz.  Different motors like different frequencies but 10kHz will get you into the ball park.

    The ATmega168/328 has 6(!!!) PWM outputs, 2 are located on each timer/counter.

    The AVR's PWM timer is capable of running in 3 different modes: Fast PWM, Phase Corrected PWM and Phase and Frequency Phase Corrected PWM.

*Fast PWM mode:*

    Fast PWM works the same way as the normal counter.  The Control Logic receives the signal and increments the TCNTn register.  When a match is detected the OCFnx flag is set and signal is send to the Waveform Generator.  The Waveform Generator then changes the state of the OCnx pin (the state is determined by the selected mode).  When the TCNTn register passes the TOP value (0xFF or OCRnA) it simply overflows (or overruns) back to 0, at the same time the OCFnx flag is set. The OCFnx flag can be configured to trigger an interrupt.  The OCFnx flag can be cleared by software, but as always is cleared automatically when an interrupt request is triggered.

    Due to the high frequency of this mode is best used for DAC, fading LEDs, rectification and Power regulation.

    The Frequency of the fast PWM can be calculated by the following equation.

PWM\_fequency = clock\_speed / [Prescaller\_value \* (1 + TOP\_Value) ]

*Phase Corrected PWM mode:*

    The phase corrected mode is a bit strange, it counts up until it hits the TOP value (FIXED, OCRnA or ICRn) then starts to count down until it hits the BOTTOM (0). The Control Logic receives the signal and increments the TCNTn register.  When a match is detected the OCFnx flag is set and signal is send to the Waveform Generator.  The Waveform Generator then changes the state of the OCnx pin (the state is determined by the selected mode).  When the TCNTn register hits the TOP value (FIXED, OCRnA or ICRn) the OCFnx flag is set. The OCFnx flag can be configured to trigger an interrupt.  The OCF1x flag can be cleared by software, but as always is cleared automatically when an interrupt request is triggered.

    This mode can be inverted or none-inverted.  In none-inverting mode, the OCn pin is LOW(GND) on the Compare Match between TCNTn and OCRnx while up-counting, and HIGH(VCC) on the Compare Match while down-counting.  In inverting mode, the OCn pin is HIGH(VCC) on the Compare Match between TCNTn and OCRnx while up-counting, and LOW(GND) on the Compare Match while down-counting.

    This mode is recommended for motor control.

    The frequency of the Phase Corrected PWM can be calculated by the following equation.

PWM\_frequency = clock\_speed / (2 \* Prescaller\_value \* TOP\_value )

*Phase and Frequency Corrected PWM mode:*

    Phase Corrected and Phase and Frequency Corrected PWM modes function the same way if we are not planning on changing our TOP value once the PWM mode is started.  The only difference that I could see on the data sheet is that the Phase and Frequency Corrected mode updates its TOP value when it hits Bottom while the Phase Corrected mode updates its TOP value when it hits the TOP.

**This mode is recommended for motor control.**

    The frequency of the Phase and Frequency Corrected PWM can be calculated by the following equation.

PWM\_frequency = clock\_speed / (2 \* Prescaller\_value \* TOP\_value )

**TIMER0 (8BIT PWM):**



*Figure 2: ATmega328 Timer0 (8bit)*

    Timer/Counter0 has 2 outputs, OC0A and OC0B.  Since both of these outputs run off the same timer and waveform generators both OC0A and OC0B are synchronized, this make the timer perfect for making tank robots (I love tank robots).

    Timer/Counter0 does not have a 32 or 128 devision in its prescaler.  Because of this it is somewhat limited to the frequencies that it could produce.  If you need a specific frequency on your PWM use Timer/Counter2 which has all of its prescaler values.

    Timer/Counter0 is capable of running on 4 modes the Fast PWM with a max TOP (0xFF), a Fast PWM mode with a variable TOP (OCR0A), a Phase Corrected PWM mode with a max TOP (0xFF) and a Phase Corrected PWM mode with a variable TOP (OCR0A).  Each of these modes can be inverted or none-inverted.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|   | **7 bit** | **6 bit** | **5 bit** | **4 bit** | **3 bit** | **2 bit** | **1 bit** | **0 bit**  |
| **TCCR0A** | COM0A1 | COM0A0  | COM0B1 | COM0B0 | - |  - |  WGM01 | WGM00  |

*Timer/Counter Control Register 0 A*

|  |  |  |
| --- | --- | --- |
| **COM0A1** | **COM0A0** | **DESCRIPTION** |
| 0 | 0  |  OC0A disabled |
| 0 | 1  |  WGM02 = 0: Normal Port Operation, OC0A Disconnected WGM02 = 1: Toggle OC0A on Compare Match |
| 1 | 0  |  None-inverted mode (HIGH at bottom, LOW on Match) |
| 1 | 1  |  Inverted mode (LOW at bottom, HIGH on Match) |

*Applies only to PWM modes*

|  |  |  |
| --- | --- | --- |
| **COM0B1** | **COM0B0** | **DESCRIPTION** |
| 0 | 0  |  OC0B disabled |
| 0 | 1  |  Reserved |
| 1 | 0  |  None-inverted mode (HIGH at bottom, LOW on Match) |
| 1 | 1  |  Inverted mode (LOW at bottom, HIGH on Match) |

*Applies only to PWM modes*

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|   | **7 bit** | **6 bit** | **5 bit** | **4 bit** | **3 bit** | **2 bit** | **1 bit** | **0 bit**  |
| **TCCR0B** | FOC0A | FOC0B | - | - | WGM02 |  CS02 |  CS01 | CS00  |

*Timer/Counter Control Register 0 A*

|  |  |  |  |
| --- | --- | --- | --- |
| **CS02** | **CS01** | **CS00** | **DESCRIPTION** |
| 0 | 0  | 0  |  Timer/Counter2 Disabled  |
| 0 | 0  | 1  |  No Prescaling |
| 0 | 1  | 0  |  Clock / 8 |
| 0 | 1  | 1  |  Clock / 64 |
| 1 | 0  | 0  |  Clock / 256 |
| 1 | 0  | 1  |  Clock / 1024 |

*CS bits*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **MODE** | **WGM02** | **WGM01** | **WGM00** |  **TOP** | **DESCRIPTION** |
| 0 | 0 | 0  | 0  |   |  Normal  |
| 1 | 0 | 0  | 1  |  0xFF |  PWM Phase Corrected |
| 2 | 0 | 1  | 0  |  OCRA |  CTC |
| 3 | 0 | 1  | 1  |  0xFF |  Fast PWM  |
| 4 | 1 | 0 | 0 |  - |  Reserved  |
| 5 | 1  | 0 | 1  |  OCR0A  |  PWM Phase Corrected  |
| 6 | 1  | 1  | 0  |  -  |  Reserved |
| 7 | 1  | 1  | 1  |  OCR0A  |  Fast PWM  |

*Waveform Generator Mode bits*

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|   | **7 bit** | **6 bit** | **5 bit** | **4 bit** | **3 bit** | **2 bit** | **1 bit** | **0 bit**  |
|  **TIMSK0** |  - | -  | -  | -  | -  |  OCIE0B |  OIE0A | TOIE0  |

*Timer/Counter Interrupt Mask Register*

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|   | **7 bit** | **6 bit** | **5 bit** | **4 bit** | **3 bit** | **2 bit** | **1 bit** | **0 bit**  |
|  **TIFR0** |  - | -  | -  | -  | -  |  OCF0B |  OCF0A | TOV0  |

*Timer/Counter Interrupt Flag Register*

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|   | **7 bit** | **6 bit** | **5 bit** | **4 bit** | **3 bit** | **2 bit** | **1 bit** | **0 bit**  |
|  **TCNT0** |   |  |  |  |  |  |  |   |

*Timer/Counter Register (stores the counter value)*

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|   | **7 bit** | **6 bit** | **5 bit** | **4 bit** | **3 bit** | **2 bit** | **1 bit** | **0 bit**  |
|  **OCR0A** |   |  |  |  |  |  |  |   |

*Output Compare Register A (stores the compare value)*

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|   | **7 bit** | **6 bit** | **5 bit** | **4 bit** | **3 bit** | **2 bit** | **1 bit** | **0 bit**  |
|  **OCR0B** |   |  |  |  |  |  |  |   |

*Output Compare Register B (stores the compare value)*

*ATmega168/328 Code:*

// this code sets up counter0 for an 8kHz Fast PWM wave @ 16Mhz Clock

#include <avr/io.h>

int main(void)
{
    DDRD |= (1 << DDD6);
    // PD6 is now an output

    OCR0A = 128;

    // set PWM for 50% duty cycle

    TCCR0A |= (1 << COM0A1);

    // set none-inverting mode

    TCCR0A |= (1 << WGM01) | (1 << WGM00);

    // set fast PWM Mode

    TCCR0B |= (1 << CS01);
    // set prescaler to 8 and starts PWM

    while (1);
    {
        // we have a working Fast PWM

    }
}

**TIMER1 (16BIT PWM):**



*Figure 3: ATmega328 Timer1 (16bit)*

    Timer/Counter1 has 2 outputs, OC1A and OC1B.  Since both of these outputs run off the same timer and waveform generators both OC1A and OC1B are synchronized, this make the timer perfect for making tank robots (I love tank robots).

    Timer/Counter1 is capable of running in 3 modes the Fast PWM mode, the Phase Corrected PWM mode and, Phase and Frequency Corrected mode.  Each of these modes can be inverted or none-inverted.  Just like Timer/Counter0 Timer/Counter1 has several options for controlling the TOP value of the PWM unlike Timer/Counter0 however, the TOP value can be fixed, stored in the OCR1A register or the ICR1 register (see table below).

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|   | **7 bit** | **6 bit** | **5 bit** | **4 bit** | **3 bit** | **2 bit** | **1 bit** | **0 bit**  |
| **TCCR1A** | COM1A1 | COM1A0 | COM1B1 | COM1B0 | - |  - |  WGM11 | WGM10  |

*Timer/Counter Control Register 1 A*

|  |  |  |
| --- | --- | --- |
| **COM1A1COM1B1** | **COM1A0COM1B0** | **DESCRIPTION** |
| 0 | 0  |  Normal port operation, OC1A/OC1B disconnected. |
| 0 | 1  |  Mode 9,11,14,15 only: Enable OCR1A only (OC1B disconnected) |
| 1 | 0  |  None-inverted mode (HIGH at bottom, LOW on Match) |
| 1 | 1  |  Inverted mode (LOW at bottom, HIGH on Match) |

*Applies only to PWM modes*

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|   | **7 bit** | **6 bit** | **5 bit** | **4 bit** | **3 bit** | **2 bit** | **1 bit** | **0 bit**  |
| **TCCR1B** | ICNC1 | ICES1 | - | WGM13 | WGM12 |  CS12 |  CS11 | CS10  |

*Timer/Counter Control Register 1 B*

|  |  |  |  |
| --- | --- | --- | --- |
| **CS12** | **CS11** | **CS10** | **DESCRIPTION** |
| 0 | 0  | 0  |  Timer/Counter2 Disabled  |
| 0 | 0  | 1  |  No Prescaling |
| 0 | 1  | 0  |  Clock / 8 |
| 0 | 1  | 1  |  Clock / 64 |
| 1 | 0  | 0  |  Clock / 256 |
| 1 | 0  | 1  |  Clock / 1024 |
| 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 |

*CS bits*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **MODE** | **WGM13** | **WGM12** | **WGM11** | **WGM10** | **DESCRIPTION** |  **TOP** |
| 0 | 0 | 0  | 0  | 0  |  Normal  |  0xFFFF |
| 1 | 0 | 0  | 0  | 1  |  PWM, Phase Corrected, 8bit |  0x00FF |
| 2 | 0 | 0  | 1  | 0  |  PWM, Phase Corrected, 9bit |  0x01FF |
| 3 | 0 | 0 | 1  | 1  |  PWM, Phase Corrected, 10bit  | 0x03FF  |
| 5 | 0  | 1  | 0  | 1  |  Fast PWM, 8bit  | 0x00FF  |
| 6 | 0  | 1  | 1  | 0  |  Fast PWM, 9bit  | 0x01FF  |
| 7 | 0  | 1  | 1 | 1  |  Fast PWM, 10bit  | 0x03FF  |
| 8 | 1  | 0  | 0  | 0  |  PWM, Phase and Frequency Corrected  | ICR1  |
| 9 | 1  | 0  | 0  | 1  |  PWM, Phase and Frequency Corrected  | OCR1A  |
| 10 | 1  | 0  | 1  | 0  |  PWM, Phase Correct  | ICR1  |
| 11 | 1  | 0  | 1  | 1  |  PWM, Phase Correct  |  OCR1A |
| 14 | 1  | 1  | 1  | 0  |  Fast PWM  | ICR1  |
| 15 | 1  | 1  | 1  | 1 |  Fast PWM | OCR1A  |

*Waveform Generator Mode bits (Abbreviated)*

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|   | **7 bit** | **6 bit** | **5 bit** | **4 bit** | **3 bit** | **2 bit** | **1 bit** | **0 bit**  |
| **TCCR1C** |  FOC1A | FOC1B  | - | - | - | - | - |   |

*Timer/Counter Control Register C*

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|   | **7 bit** | **6 bit** | **5 bit** | **4 bit** | **3 bit** | **2 bit** | **1 bit** | **0 bit**  |
|  **TIMSK1** | - | -  | ICIE1  | -  | -  |  OCIE1B |  OCIE1A | TOIE1  |

*Timer/Counter Interrupt Mask Register*

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|   | **7 bit** | **6 bit** | **5 bit** | **4 bit** | **3 bit** | **2 bit** | **1 bit** | **0 bit**  |
|  **TIFR** |  OCF2 | TOV2  | ICF1  | OCF1A  | OCF1B  |  TOV1 |  - | TOV0  |

*Timer/Counter Interrupt Flag Register*

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|   | **7 bit** | **6 bit** | **5 bit** | **4 bit** | **3 bit** | **2 bit** | **1 bit** | **0 bit**  |
| **TCNT1H** |   |  |  |  |  |  |  |   |
|  **TCNT1L** |   |   |   |   |   |   |   |   |

*Timer/Counter Register (stores the counter value, 16 bit)*

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|   | **7 bit** | **6 bit** | **5 bit** | **4 bit** | **3 bit** | **2 bit** | **1 bit** | **0 bit**  |
| **OCR1AH** |   |  |  |  |  |  |  |   |
| **OCR1AL** |   |   |   |   |   |   |   |   |

*Output Compare Register A (stores the compare value, 16 bit)*

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|   | **7 bit** | **6 bit** | **5 bit** | **4 bit** | **3 bit** | **2 bit** | **1 bit** | **0 bit**  |
| **OCR1BH** |   |  |  |  |  |  |  |   |
| **OCR1BL** |   |   |   |   |   |   |   |   |

*Output Compare Register B (stores the compare value, 16 bit)*

*ATmega168/328 Code:*

// this code sets up counter1 A output at 25% and B output at 75%

// using ICR1 as top (16bit), Fast PWM.

#include <avr/io.h>

int main(void)
{
    DDRB |= (1 << DDB1)|(1 << DDB2);
    // PB1 and PB2 is now an output

    OCR1 = 0xFFFF;

    // set TOP to 16bit

    OCR1A = 0x3FFF;

    // set PWM for 25% duty cycle @ 16bit

    OCR1B = 0xBFFF;

    // set PWM for 75% duty cycle @ 16bit

    TCCR1A |= (1 << COM1A1)|(1 << COM1B1);

    // set none-inverting mode

    TCCR1A |= (1 << WGM11);

    TCCR1B |= (1 << WGM12)|(1 << WGM13);

    // set Fast PWM mode using ICR1 as TOP

    TCCR1B |= (1 << CS10);

    // START the timer with no prescaler

    while (1);
    {
        // we have a working Fast PWM

    }
}

**TIMER2 (8BIT PWM):**



*Figure 2: ATmega328 Timer2 (8bit)*

    Timer/Counter0 is pretty cool because it has 2 outputs, OC2A and OC2B.  Since both of these outputs run off the same timer and waveform generators both OC2A and OC2B are synchronized, this make the timer perfect for making tank robots (I love tank robots).

    As stated under Timer/Counter0, Timer/Counter2 has all of its prescaler values, this makes it capable of producing a lot more frequencies then its 8 bit brother.

    Timer/Counter0 is capable of running on 4 modes the Fast PWM with a max TOP (0xFF), a Fast PWM mode with a variable TOP (OCR2A), a Phase Corrected PWM mode with a max TOP (0xFF) and a Phase Corrected PWM mode with a variable TOP (OCR2A).  Each of these modes can be inverted or none-inverted.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|   | **7 bit** | **6 bit** | **5 bit** | **4 bit** | **3 bit** | **2 bit** | **1 bit** | **0 bit**  |
| **TCCR2A** | COM2A1 | COM2A0  | COM2B1 | COM2B0 | - |  - |  WGM21 | WGM20  |

*Timer/Counter Control Register 2 A*

|  |  |  |
| --- | --- | --- |
| **COM2A1** | **COM2A0** | **DESCRIPTION** |
| 0 | 0  |  OC2A disabled |
| 0  | 1  |  WGM22 = 0: Normal Port Operation, OC2A Disconnected WGM22 = 1: Toggle OC2A on Compare Match |
| 1 | 0  |  None-inverted mode (HIGH at bottom, LOW on Match) |
| 1 | 1  |  Inverted mode (LOW at bottom, HIGH on Match) |

*Applies only to PWM modes*

|  |  |  |
| --- | --- | --- |
| **COM2B1** | **COM2B0** | **DESCRIPTION** |
| 0 | 0  |  OC2B disabled |
| 0 | 1  |  Reserved |
| 1 | 0  |  None-inverted mode (HIGH at bottom, LOW on Match) |
| 1 | 1  |  Inverted mode (LOW at bottom, HIGH on Match) |

*Applies only to PWM modes*

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|   | **7 bit** | **6 bit** | **5 bit** | **4 bit** | **3 bit** | **2 bit** | **1 bit** | **0 bit**  |
| **TCCR2B** | FOC2A | FOC2B | - | - | WGM22 |  CS22 |  CS21 | CS20  |

*Timer/Counter Control Register 2 A*

|  |  |  |  |
| --- | --- | --- | --- |
| **CS22** | **CS21** | **CS20** | **DESCRIPTION** |
| 0 | 0  | 0  |  Timer/Counter2 Disabled  |
| 0 | 0  | 1  |  No Prescaling |
| 0 | 1  | 0  |  Clock / 8 |
| 0 | 1  | 1  |  Clock / 32 |
| 1 | 0  | 0  |  Clock / 64 |
| 1 | 0  | 1  |  Clock / 128 |
| 1 | 1  | 0  |  Clock / 256 |
| 1 | 1  | 1  |  Clock / 1024 |

*CS bits*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **MODE** | **WGM22** | **WGM21** | **WGM20** |  **TOP** | **DESCRIPTION** |
| 0 | 0 | 0 | 0 |  0xFF |  Normal  |
| 1 | 0 | 0 | 1 |  0xFF |  PWM Phase Corrected |
| 2 | 0 | 1 | 0 |  OCRA |  CTC |
| 3 | 0 | 1 | 1 |  0xFF |  Fast PWM  |
| 4 | 1 | 0 | 0 |  - |  Reserved  |
| 5 |  1  | 0 | 1 |  OCR0A  |  PWM Phase Corrected  |
| 6 |  1  | 1 | 0 |  -  |  Reserved |
| 7 |  1  | 1 | 1 |  OCR0A  |  Fast PWM  |

*Waveform Generator Mode bits*

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|   | **7 bit** | **6 bit** | **5 bit** | **4 bit** | **3 bit** | **2 bit** | **1 bit** | **0 bit**  |
|  **TIMSK2** |  - | -  | -  | -  | -  |  OCIE2B |  OIE2A | TOIE2  |

*Timer/Counter Interrupt Mask Register*

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|   | **7 bit** | **6 bit** | **5 bit** | **4 bit** | **3 bit** | **2 bit** | **1 bit** | **0 bit**  |
|  **TIFR2** |  - | -  | -  | -  | -  |  OCF2B |  OCF2A | TOV2  |

*Timer/Counter Interrupt Flag Register*

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|   | **7 bit** | **6 bit** | **5 bit** | **4 bit** | **3 bit** | **2 bit** | **1 bit** | **0 bit**  |
|  **TCNT2** |   |  |  |  |  |  |  |   |

*Timer/Counter Register (stores the counter value)*

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|   | **7 bit** | **6 bit** | **5 bit** | **4 bit** | **3 bit** | **2 bit** | **1 bit** | **0 bit**  |
|  **OCR2A** |   |  |  |  |  |  |  |   |

*Output Compare Register A (stores the compare value)*

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|   | **7 bit** | **6 bit** | **5 bit** | **4 bit** | **3 bit** | **2 bit** | **1 bit** | **0 bit**  |
|  **OCR2B** |   |  |  |  |  |  |  |   |

*Output Compare Register B (stores the compare value)*

*ATmega168/328 Code:*

// this code sets up counter2 for an 8kHz Fast PWM wave @ 16Mhz Clock

#include <avr/io.h>

int main(void)
{
    DDRD |= (1 << DDD6);
    // PD6 is now an output

    OCR2A = 128;

    // set PWM for 50% duty cycle

    TCCR2A |= (1 << COM2A1);

    // set none-inverting mode

    TCCR2A |= (1 << WGM21) | (1 << WGM20);

    // set fast PWM Mode

    TCCR2B |= (1 << CS21);
    // set prescaler to 8 and starts PWM

    while (1);
    {
        // we have a working Fast PWM

    }
}

<http://www.atmel.com/Images/doc2542.pdf>