# Detalhando o PWM do Arduino com microcontrolador ATmega328

Por Eng. Roberto Bairros dos Santos

www.bairrospd.com

Data: 09/06/2017

## Sumário

## Prefácio.

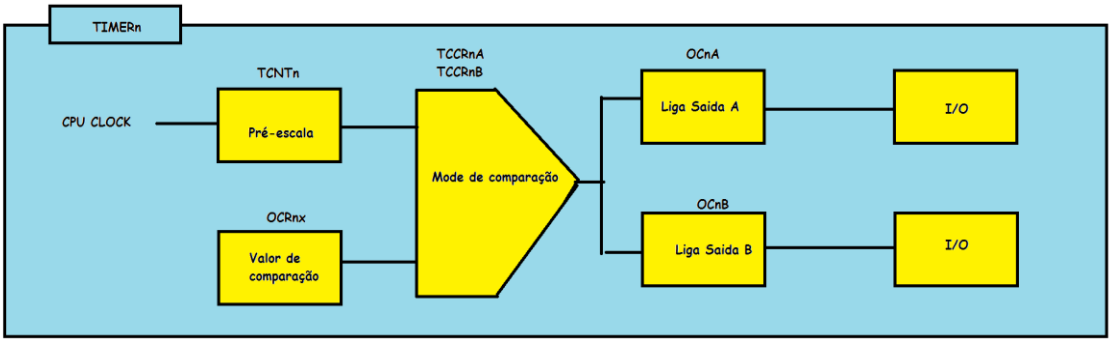
Este tutorial mostra os detalhes do funcionamento do PWM na placa Arduino Uno que usa o microcontrolador ATmega328P.

Conhecendo estes detalhes você será capaz de alterar o funcionamento da função analogwrite() de forma a trabalhar com frequências mais altas!

A aplicação prática deste conhecimento serão abordadas em outros tutoriais onde serão abordados os tópicos: PWM de controle de um Led para evitar cintilações e gerador de senóide com amis precisão.

## Como funciona o Temporizador.

Existem três temporizadores todos seguindo o funcionamento da figura abaixo, onde “n” será o número dos temporizadores, neste caso podendo assumir os valores “0”, “1” ou “2”.




O temporizador é um contador de pulsos de clock, o clock básico é gerado pela CPU, no caso da placa Arduino UNO este clock é de 16MHz.

A contagem dos pulsos é armazenada no registrador de contagem de tempo TCNTn, o tempo do clock pode ser ajustado através da configuração do registrador TCCRnB. O valor máximo do TCNT é definido pela variável TOP e é normalmente configurado em 255.

O valor do contador é comparado com um valor ajustado no programa no registrador OCRn, O valor do OCRn é o tempo do ciclo de trabalho (duty cycle) do PWM na instrução writeAnalog (pino, valor), normalmente este valor pode variar de 0 a 255!

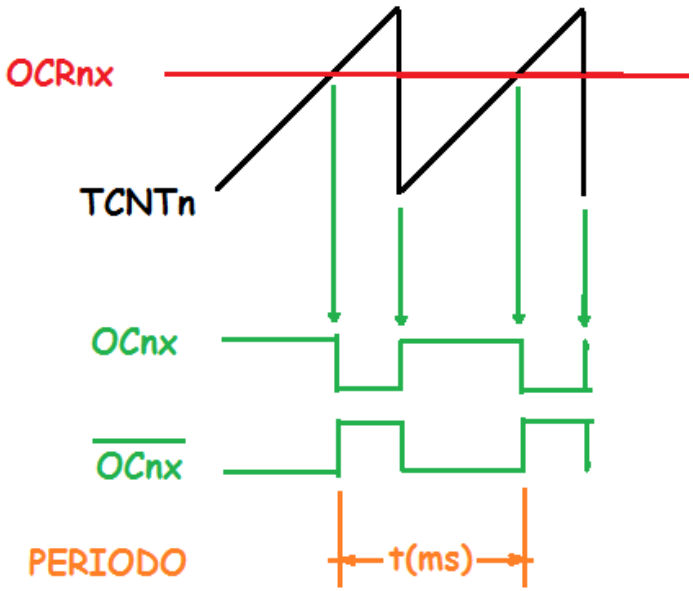
O valor da contagem TCNTn é comparado com o valor de comparação OCRn gerando dois pulsos digitais OCnA e OCnB, estes pulsos é que irão ligar ou desligar os pinos de saída (I/O) do Arduino, por isto você deve configurar estes pinos como OUTPUT no setup.

A saída do comparador também poderá acionar uma interrupção, mas este é um tópico para outro tutorial!

O tipo de comparação é definido pela configuração dos registradores TCCRnA e TCCRnB através dos bits chamados WGM (Waveform Generation Mode) definindo o tipo de PWM a ser usado, existem vários tipos, nós vamos estudar neste trabalho somente os tipos “Fast PWM” e “PWM, phase correct” que são aqueles utilizados nas instruções normais do Arduino.

## Como funciona o modo “Fast PWM”.

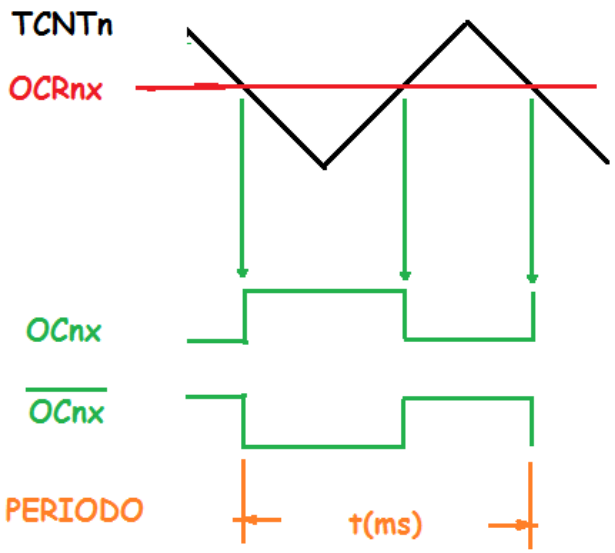
A figura abaixo mostra como funciona este modo.




A comparação é definida quando o contador de tempo TCNTn é comparado com o valor de comparação programado OCRn quando os dois valores forem iguais o registro de saída OCnx muda de valor.

## Como funciona o modo “PWM, phase correct”.

A figura abaixo mostra como funciona este modo.




Neste caso a comparação é definida na descida e subida do  contador de tempo TCNTn. A descida do TCNTn é usado para ligar o registrador de saída OCnx, e a subida é usada para desligar o registrador de saída OCnx, com isto é possível gerar formas de ondas mais simétricas.

Notar que neste modo o tempo do período é o dobro do tempo gerado no modo “Fast PWM”.

## Detalhes do valor máximo de contagem “TOP LImit”.

O valor do limite da contagem é chamado de TOP, este valor é normalmente é 255, mas pode ser alterado através do registrador OCRA.

O valor limite pode ser configurado nos bites de configuração “WGM”!

Nas instruções do Arduino o TOP é 255!

## Como configurar os registradores.

Você configura os registradores para escolher o modo de funcionamento e o período (frequência) do PWM, para isto são usados dois registradores o TCCRnA e o TCCRnB!

O valor de comparação OCRn pode ser  definido no programa, ou indiretamente através da instrução analogWrite(pino, valor).

Usar a instrução analogWrite() é mais simples porque você só vai precisar se preocupara em ajustar o os limites no registradores TCCRnx!

As figuras deste trabalho foram retirados do manual do microcontrolador ATmega328P!

Serão analisado somente os bits relevantes levando em conta a configuração normal da instrução analogWrite().

## TEMPORIZADOR 0.

Veja os registradores do timer 0.

**TCCR0A – Timer/Counter Control Register A**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0x24 (0x44) | COM0A1 | COM0A0 | COM0B1 | COM0B0 | – | – | WGM01 | WGM00 |
| 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 |

Os bits “COM” configura como o resultado da comparação vai atuar nos registradores de saída, o normal é usar o parâmetro “10” que desliga a saída quando ocorrer a comparação.

Os bits “ WGM” deste registrador deverão ser juntados aos bit WGM do registrador “B” para definir o modo de gerar o PWM.

**Table 15-3.** Compare Output Mode, Fast PWM Mode[1]

| COM0A1 | COM0A0 | Description |
|---|---|---|
| 0 | 0 | Normal port operation, OC0A disconnected. |
| 0 | 1 | WGM02 = 0: Normal Port Operation, OC0A Disconnected.<br>WGM02 = 1: Toggle OC0A on Compare Match. |
| 1 | 0 | Clear OC0A on Compare Match, set OC0A at BOTTOM, (non-inverting mode). |
| 1 | 1 | Set OC0A on Compare Match, clear OC0A at BOTTOM, (inverting mode). |

**Table 15-6.** Compare Output Mode, Fast PWM Mode[1]

| COM0B1 | COM0B0 | Description |
|---|---|---|
| 0 | 0 | Normal port operation, OC0B disconnected. |
| 0 | 1 | Reserved |
| 1 | 0 | Clear OC0B on Compare Match, set OC0B at BOTTOM, (non-inverting mode) |
| 1 | 1 | Set OC0B on Compare Match, clear OC0B at BOTTOM, (inverting mode). |

**TCCR0B – Timer/Counter Control Register B**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| 0x25 (0x45) | FOC0A | FOC0B | – | – | WGM02 | CS02 | CS01 | CS00 | TCCR0B |
| Read/Write | W | W | R | R | R/W | R/W | R/W | R/W | |
| Initial Value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |

A principal função deste registrador é definir a frequência de trabalho através dos bits ” CS” e juntamente com os bits “WGM do temporizador “A” definir o modo de gerar o PWM!

Os modos mais usados são Modo 1 e Modo 3!

**Table 15-8.** Waveform Generation Mode Bit Description

| Mode | WGM02 | WGM01 | WGM00 | 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 |

A frequencai do PWM é definido pela tabela abaixo e o modo do PWM.

**Table 15-9.** Clock Select Bit Description

| CS02 | CS01 | CS00 | Description |
|---|---|---|---|
| 0 | 0 | 0 | No clock source (Timer/Counter stopped) |
| 0 | 0 | 1 | $clk_{I/O}$/(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 T0 pin. Clock on falling edge. |
| 1 | 1 | 1 | External clock source on T0 pin. Clock on rising edge. |

Tabela mostrando o valor inteiro de CS0, o valor de N, o valor deda frequência em Hz e o valor do período em ms!

O período é calculador por

$t=1/f_{OCnxPCPWM}$!

**TIMER 0 Fast PWM**

| CS | N | fOCnxPWM | t(ms) |
|---|---|---|---|
| 1 | 1 | 62500 | 0,016 |
| 2 | 8 | 7812,5 | 0,128 |
| 3 | 64 | 976,5625 | 1,024 |
| 4 | 256 | 244,140625 | 4,096 |
| 5 | 1024 | 61,03515625 | 16,384 |

$$f_{OCnxPWM} = \frac{f_{clk\_I/O}}{N \cdot 256}$$

**TIMER 0 PWM, phase correct**

| CS | N | fOCnxPCPWM | t(ms) |
|---|---|---|---|
| 1 | 1 | 31372,55 | 0,031875 |
| 2 | 8 | 3921,57 | 0,255 |
| 3 | 64 | 490,20 | 2,040 |
| 4 | 256 | 122,55 | 8,160 |
| 5 | 1024 | 30,64 | 32,640 |

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

## TEMPORIZADOR 1.

Veja os registradores do Timer 1

**TCCR1A – Timer/Counter1 Control Register A**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| (0x80) | COM1A1 | COM1A0 | COM1B1 | COM1B0 | – | – | WGM11 | WGM10 |
| 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 |

Os bits “COM” configura como o resultado da comparação vai atuar nos registradores de saída, o normal é usar o parâmetro “10” que desliga a saída quando ocorrer a comparação.

Os bits “ WGM” deste registrador deverão ser juntados aos bit WGM do registrador “B” para definir o modo de gerar o PWM.

**Table 16-1.** Compare Output Mode, non-PWM

| COM1A1/COM1B1 | COM1A0/COM1B0 | Description |
|---|---|---|
| 0 | 0 | Normal port operation, OC1A/OC1B disconnected. |
| 0 | 1 | Toggle OC1A/OC1B on Compare Match. |
| 1 | 0 | Clear OC1A/OC1B on Compare Match (Set output to low level). |
| 1 | 1 | Set OC1A/OC1B on Compare Match (Set output to high level). |

**Table 16-2.** Compare Output Mode, Fast PWM(1)

| COM1A1/COM1B1 | COM1A0/COM1B0 | Description |
|---|---|---|
| 0 | 0 | Normal port operation, OC1A/OC1B disconnected. |
| 0 | 1 | WGM13:0 = 14 or 15: Toggle OC1A on Compare Match, OC1B disconnected (normal port operation). For all other WGM1 settings, normal port operation, OC1A/OC1B disconnected. |
| 1 | 0 | Clear OC1A/OC1B on Compare Match, set OC1A/OC1B at BOTTOM (non-inverting mode) |
| 1 | 1 | Set OC1A/OC1B on Compare Match, clear OC1A/OC1B at BOTTOM (inverting mode) |

**TCCR1B – Timer/Counter1 Control Register B**

| 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 | |

A principal função deste registrador é definir a frequência de trabalho através dos bits ” CS” e juntamente com os bits “WGM do temporizador “A” definir o modo de gerar o PWM!

Os modos mais usados são Modo 1  e Modo 5!

**Table 16-4.** Waveform Generation Mode Bit Description[1]

| 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 |

A frequencai do PWM é definido pela tabela abaixo e o modo do PWM.

**Table 16-5.** 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. |

Tabela mostrando o valor inteiro de CS0, o valor de N, o valor deda frequência em Hz e o valor do período em ms!

O período é calculador por

$$t=1/f_{OCnxPCPWM}!$$

Na tabela abaixo o valor do TOP foi considerado 255, que o valor comum nas funções do Arduino.

**TIMFR 1 Fast PWM**

| CS | N | fOCnxPWM | t(ms) |
|---|---|---|---|
| 1 | 1 | 62500 | 0,016 |
| 2 | 8 | 7812,5 | 0,128 |
| 3 | 64 | 976,5625 | 1,024 |
| 4 | 256 | 244,140625 | 4,096 |
| 5 | 1024 | 61,03515625 | 16,384 |

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

**TIMER 1 PWM, phase correct**

| CS | N | fOCnxPCPWM | t(ms) |
|---|---|---|---|
| 1 | 1 | 31372,55 | 0,031875 |
| 2 | 8 | 3921,57 | 0,255 |
| 3 | 64 | 490,20 | 2,040 |
| 4 | 256 | 122,55 | 8,160 |
| 5 | 1024 | 30,64 | 32,640 |

$$f_{OCnxPCPWM} = \frac{f_{clk\_I/O}}{2 \cdot N \cdot TOP}$$

## TEMPORIZADOR 2.

Veja os registradores do Timer 2

**TCCR2A – Timer/Counter Control Register A**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| (0xB0) | COM2A1 | COM2A0 | COM2B1 | COM2B0 | – | – | WGM21 | WGM20 |
| 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 |

Os bits “COM” configura como o resultado da comparação vai atuar nos registradores de saída, o normal é usar o parâmetro “10” que desliga a saída quando ocorrer a comparação.

Os bits “ WGM” deste registrador deverão ser juntados aos bit WGM do registrador “B” para definir o modo de gerar o PWM.

**Table 18-3.** Compare Output Mode, Fast PWM Mode[1]

| COM2A1 | COM2A0 | Description |
|---|---|---|
| 0 | 0 | Normal port operation, OC2A disconnected. |
| 0 | 1 | WGM22 = 0: Normal Port Operation, OC0A Disconnected.<br>WGM22 = 1: Toggle OC2A on Compare Match. |
| 1 | 0 | Clear OC2A on Compare Match, set OC2A at BOTTOM, (non-inverting mode). |
| 1 | 1 | Set OC2A on Compare Match, clear OC2A at BOTTOM, (inverting mode). |

**Table 18-6.** Compare Output Mode, Fast PWM Mode[1]

| COM2B1 | COM2B0 | Description |
|---|---|---|
| 0 | 0 | Normal port operation, OC2B disconnected. |
| 0 | 1 | Reserved |
| 1 | 0 | Clear OC2B on Compare Match, set OC2B at BOTTOM, (non-inverting mode). |
| 1 | 1 | Set OC2B on Compare Match, clear OC2B at BOTTOM, (inverting mode). |

**TCCR2B – Timer/Counter Control Register B**

| 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 | |

A principal função deste registrador é definir a frequência de trabalho através dos bits ” CS” e juntamente com os bits “WGM do temporizador “A” definir o modo de gerar o PWM!

Os modos mais usados são Modo 1 e Modo 3!

**Table 18-8.** 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 |

A frequencai do PWM é definido pela tabela abaixo e o modo do PWM.

**Table 18-9.** 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) |

Tabela mostrando o valor inteiro de CS0, o valor de N, o valor deda frequência em Hz e o valor do período em ms!

O período é calculador por

$$t=1/f_{OCnxPCPWM}!$$

Note que este temporizador possui duas pré-escalas que os outros não tem.

**TIMER 2 Fast PWM**

| CS | N | fOCnxPWM | t(ms) |
|---|---|---|---|
| 1 | 1 | 62500 | 0,016 |
| 2 | 8 | 7812,5 | 0,128 |
| 3 | 32 | 1953,125 | 0,512 |
| 4 | 64 | 976,5625 | 1,024 |
| 5 | 128 | 488,28125 | 2,048 |
| 6 | 256 | 244,140625 | 4,096 |
| 7 | 1024 | 61,03515625 | 16,384 |

**TIMER 2 PWM, phase correct**

| CS | N | fOCnxPWM | t(ms) |
|---|---|---|---|
| 1 | 1 | 31372,54902 | 0,031875 |
| 2 | 8 | 3921,568627 | 0,255 |
| 3 | 32 | 980,3921569 | 1,02 |
| 4 | 64 | 490,1960784 | 2,04 |
| 5 | 128 | 245,0980392 | 4,08 |
| 6 | 256 | 122,5490196 | 8,16 |
| 7 | 1024 | 30,6372549 | 32,64 |

$$f_{OCnxPWM} = \frac{f_{clk\_I/O}}{N \cdot 256}$$

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

## Nomes dos registradores na linguagem do Arduino.

Você pode escrever diretamente no registrador de 8Bits como faz com qualquer variável do tipo char no Arduino pois todos os registradores estão definidos na linguagem do Arduino.

Você também pode alterar os bits diretamente pois todos os bits são definidos na linguagem do Arduino. Para ligar um bit (colocar um no bit) você pode usar o macro _BV(nome do bit) ou ainda ~(_BV(nome do bit) ) para desligar o bit (colocar zero no bit)!

A lista com os nomes dos registradores usados no temporizador é mostrada abaixo.

| Registradores de 8 bits | bits | | | | | |
|---|---|---|---|---|---|---|
| TCCR0A | WGM01 | WGM00 | COM0A1 | COM0A0 | COM0B1 | COMOB0 |
| TCCR0B | WGM02 | CS02 | CS01 | CS00 | | |
| OCR0A | | | | | | |
| OCR0B | | | | | | |
| TCNT0 | | | | | | |
| TCCR1A | WGM11 | WGM10 | COM1A1 | COM1A0 | COM1B1 | COM1B0 |
| TCCR1B | WGM13 | WGM12 | CS12 | CS11 | CS10 | |
| OCR1A | | | | | | |
| OCR1B | | | | | | |
| TCNT1 | | | | | | |
| TCCR2A | WGM21 | WGM20 | COM2A1 | COM2A0 | COM2B1 | COM2B0 |
| TCCR2B | WGM22 | CS22 | CS21 | CS20 | | |
| OCR2A | | | | | | |
| OCR2B | | | | | | |
| TCNT2 | | | | | | |

Exemplo:

```
TCCR2A = _BV(COM2A0) ;//liga só o bit COM2A  do registrador TCCR2A
TCCR2A = _BV(COM2A0) | _BV(COM2B1) | _BV(WGM20);//liga mais de um bit  OCR2A = 180;
OCR2B = 50;//altera o valor do duty cycle da saída B do temporizador 2
TCCR0A=163;//0b10100011 altera direto todos os bits usando número inteiro
TCCR0A=0b10100011;//163 altera direto usando número binário
```

## Conclusão.

Neste tutorial você viu como funcionam os temporizadores do microcontrolador ATmega328 usado na placa Arduino UNO.

Nos próximos tutoriais você verá como eles são configurados para executar as instruções de PWM do Arduino  e como alterar estas configurações para aumentar a frequência do PWM.

No tutorial final você verá como aplicar estes conhecimentos para alterar o PWM de controle de um Led para evitar cintilações e gerador de senóide com mais precisão!

## Referências.

Bibliografia.

Manual da Atmel para o microcontrolador ATmega328P.

Sites: www.bairrospd.com

SEO: www.bairrospd.com, Arduino, PWM, alterando a frequência do PWM, senóide, LED, eletrônica, tutorial