CAPTURE/COMPARE/PWM

O módulo CCP (capture/compare/pwm) é uma funcionalidade do microcontrolador que permite fazer análise detalhada de pulsos externos e também de gerar pulsos.

As funcionalidades deste módulo estão atreladas aos timers:

MODO CCP	TIMER COMPARTILHADO
CAPTURE	Timer1 e Timer3
COMPARE	Timer1 e Timer3
PWM	Timer2

No momento que você aciona o módulo CCP, você perde a funcionalidade do timer correspondente a ele. Ou seja, o selecionado será exclusivo para o funcionamento do módulo CCP. O nosso microcontrolador em específico, PIC18f45k22, possui 5 módulos CCP, e cada um deles pode ser atrelado a algum timer (menos timer0). Essa escolha do timer é feita utilizando os registradores CCPTMRS0 e CCPTMRS1:

REGISTER 14-3: CCPTMRS0: PWM TIMER SELECTION CONTROL REGISTER 0

R/W-0	R/W-0	U-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0
C3TSEL<1:0>		—	C2TSEL<1:0>		—	C1TSEL<1:0>	
bit 7							bit 0

Legend:		
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'
u = Bit is unchanged	x = Bit is unknown	-n/n = Value at POR and BOR/Value at all other Resets
'1' = Bit is set	'0' = Bit is cleared	

bit 7-6 **C3TSEL<1:0>:** CCP3 Timer Selection bits
00 = CCP3 – Capture/Compare modes use Timer1, PWM modes use Timer2
01 = CCP3 – Capture/Compare modes use Timer3, PWM modes use Timer4
10 = CCP3 – Capture/Compare modes use Timer5, PWM modes use Timer6
11 = Reserved

bit 5 **Unused**

bit 4-3 **C2TSEL<1:0>:** CCP2 Timer Selection bits
00 = CCP2 – Capture/Compare modes use Timer1, PWM modes use Timer2
01 = CCP2 – Capture/Compare modes use Timer3, PWM modes use Timer4
10 = CCP2 – Capture/Compare modes use Timer5, PWM modes use Timer6
11 = Reserved

bit 2 **Unused**

bit 1-0 **C1TSEL<1:0>:** CCP1 Timer Selection bits
00 = CCP1 – Capture/Compare modes use Timer1, PWM modes use Timer2
01 = CCP1 – Capture/Compare modes use Timer3, PWM modes use Timer4
10 = CCP1 – Capture/Compare modes use Timer5, PWM modes use Timer6
11 = Reserved

REGISTER 14-4: CCPTMRS1: PWM TIMER SELECTION CONTROL REGISTER 1

U-0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	—	C5TSEL<1:0>		C4TSEL<1:0>	
bit 7							bit 0

Legend:		
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'
u = Bit is unchanged	x = Bit is unknown	-n/n = Value at POR and BOR/Value at all other Resets
'1' = Bit is set	'0' = Bit is cleared	

bit 7-4 **Unimplemented:** Read as '0'

bit 3-2 **C5TSEL<1:0>:** CCP5 Timer Selection bits
00 = CCP5 – Capture/Compare modes use Timer1, PWM modes use Timer2
01 = CCP5 – Capture/Compare modes use Timer3, PWM modes use Timer4
10 = CCP5 – Capture/Compare modes use Timer5, PWM modes use Timer6
11 = Reserved

bit 1-0 **C4TSEL<1:0>:** CCP4 Timer Selection bits
00 = CCP4 – Capture/Compare modes use Timer1, PWM modes use Timer2
01 = CCP4 – Capture/Compare modes use Timer3, PWM modes use Timer4
10 = CCP4 – Capture/Compare modes use Timer5, PWM modes use Timer6
11 = Reserved

Registrador CCPxCON

Esse registrador vale para os 5 módulos CCP do PIC18F45k22. O registrador é extremamente simples, dando somente a opção de escolher o modo de operação. O módulo CCP como um todo deve ser operado com lógica de programação que será explicado em cada funcionalidade.

REGISTER 14-1: CCPxCON: STANDARD CCPx CONTROL REGISTER

U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	DCxB<1:0>		CCPxM<3:0>			
bit 7							bit 0

bit 7-6 **Unused**

bit 5-4 **DCxB<1:0>:** PWM Duty Cycle Least Significant bits

Capture mode:
Unused

Compare mode:
Unused

PWM mode:
These bits are the two LSbs of the PWM duty cycle. The eight MSbs are found in CCPRxL.

Os bits 7 e 6 não são utilizados, enquanto o 5-4 são explicados na parte de PWM. Se não for utilizar PWM, eles são ignorados também

bit 3-0 **CCPxM<3:0>:** ECCPx Mode Select bits

0000 = Capture/Compare/PWM off (resets the module)
0001 = Reserved
0010 = Compare mode: toggle output on match
0011 = Reserved

0100 = Capture mode: every falling edge
0101 = Capture mode: every rising edge
0110 = Capture mode: every 4th rising edge
0111 = Capture mode: every 16th rising edge

1000 = Compare mode: set output on compare match (CCPx pin is set, CCPxIF is set)
1001 = Compare mode: clear output on compare match (CCPx pin is cleared, CCPxIF is set)
1010 = Compare mode: generate software interrupt on compare match (CCPx pin is unaffected, CCPxIF is set)
1011 = Compare mode: Special Event Trigger (CCPx pin is unaffected, CCPxIF is set)
TimerX (selected by CxTSEL bits) is reset
ADON is set, starting A/D conversion if A/D module is enabled(1)
11xx =: PWM mode

Aqui você pode selecionar o modo de operação do CCP, além de definir o prescaler do Capture e o modo de operação do Compare.

Modo Capture

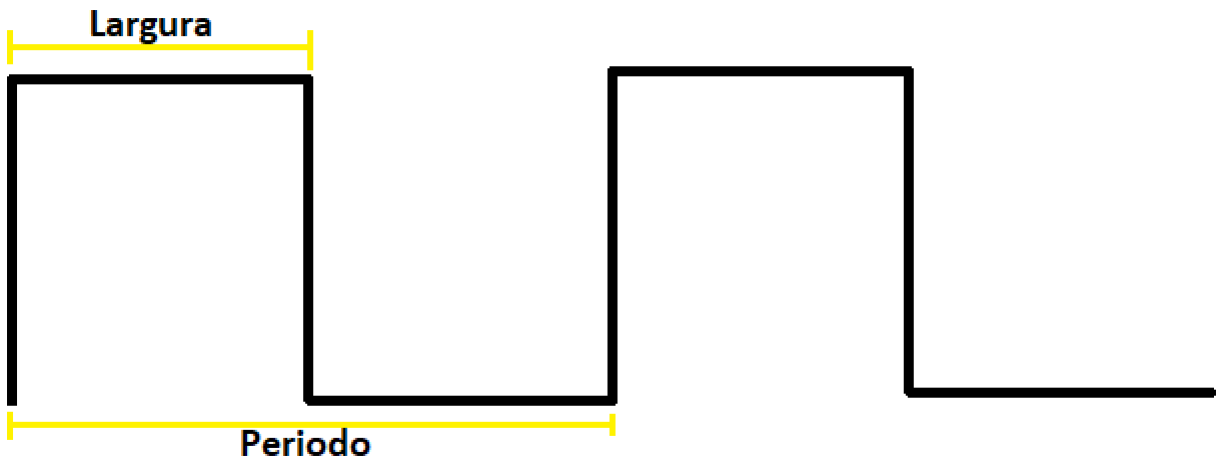
Com essa funcionalidade, você consegue ler o período e a largura de um pulso externo. Isso pode ser utilizado para construir um frequencímetro, por exemplo.

Você deve programar o timer1 pois ele será a base de medição da largura de pulso de um sinal. Imagine o timer1 como um cronômetro que irá iniciar quando um sinal chegar e finalizar quando esse sinal terminar. Você pode acessar o resultado deste cronômetro através dos registradores TMR1. Você pode programar o incremento do TMR1 em intervalos de 1 microssegundo. Assim, saberá que para cada unidade incrementada no registrador durante o pulso, passou 1 microssegundo.

Outra coisa muito importante a respeito do timer para o módulo CCP, é que é recomendado que você habilite o sincronismo no registrador TxCON.

Outra medida muito importante que pode ser adquirida com este periférico é o Duty Cycle, peça chave de um pulso. O duty cycle consiste na razão entre a largura do pulso e seu período. Em outras palavras, o tempo que o sinal permaneceu ligado divido pelo tempo total do sinal. Essa razão afeta diretamente a tensão resultante gerada pelo sinal.

A forma do pulso (ou sinal) que estamos dizendo é a quadrada:

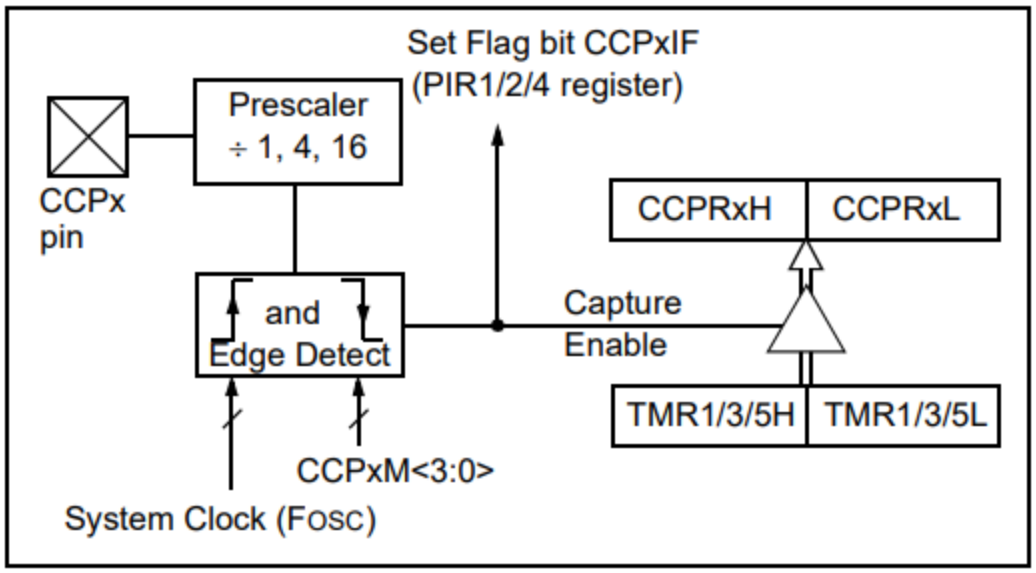


$$DutyCycle = Largura/Periodo$$

A parte “ligada” do sinal seria o nível lógico 1, ou +5v, enquanto a parte baixa é o nível lógico 0. Tanto a largura quanto o período são medidos em segundos. Ainda, a frequência do sinal pode ser obtida calculando $1/periodo$, com resultado em Hertz.

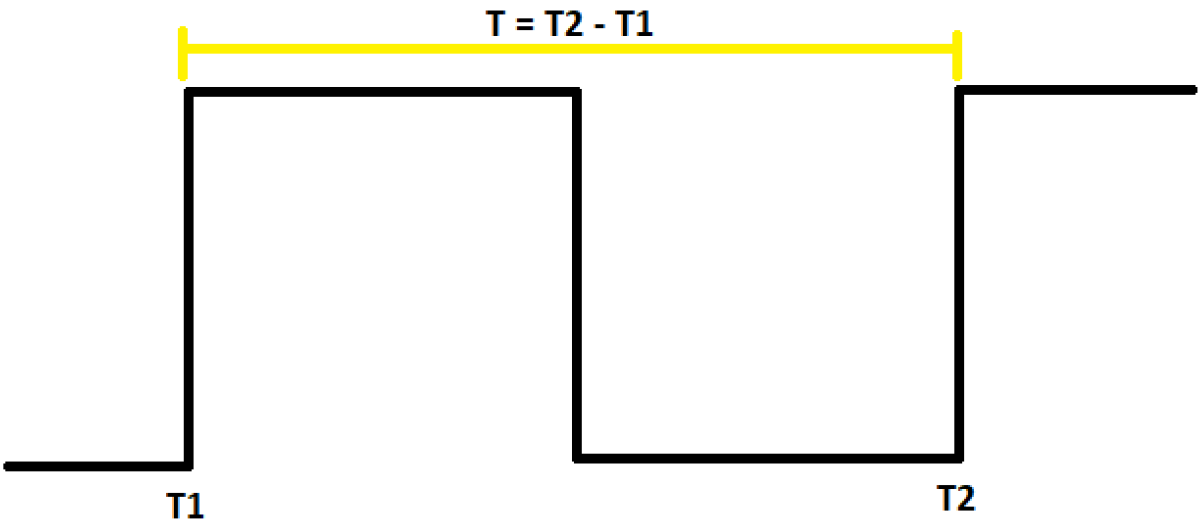
Funcionamento

O módulo Compare identifica automaticamente uma borda no sinal. Pode ser uma borda de decida, subida ou um conjunto de bordas (definido no registrador CCPxCON). Quando essa borda chega, os valores de TMR1L e TMR1H são salvos automaticamente nos registradores CCPR1L e CCPR1H, e assim você tem disponível o tempo exato que corresponde à largura do pulso.



Medindo o período de um sinal com Capture

Você deverá setar interrupções para o Capture. Sempre que uma borda de subida (ou a borda que você selecionou no prescaler) for lida, a interrupção será chamada. Então para ler um período de um sinal, basta salvar o valor nos registradores CCPR (aqueles que copiam o valor do TMR1) em duas interrupções e depois subtrair um pelo outro, assim você terá o período do sinal.



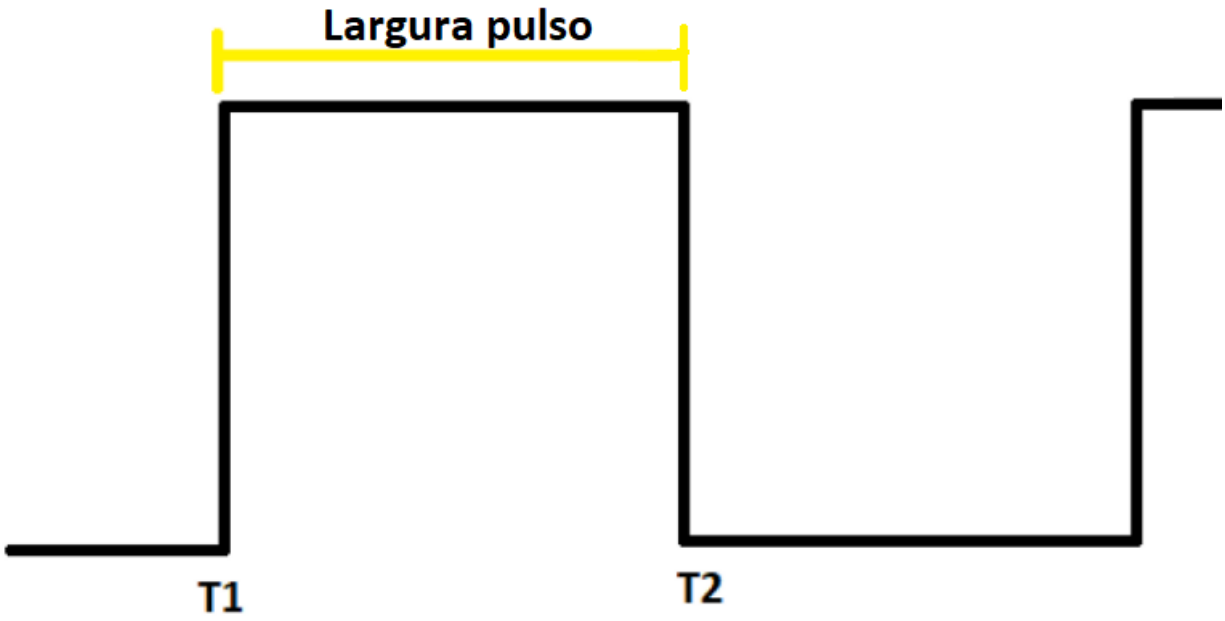
Para realizar este processo, você deve

- Habilitar as interrupções globais
- Habilitar interrupção pelo módulo CCP
- Configurar o timer1 da forma que achar melhor, lembrando que o tempo de incremento do timer influencia diretamente no resultado final do período do sinal
- Configurar a função de interrupção para o módulo CCP de forma que você consiga salvar o valor de CCPR de 2 interrupções e retornar para a main
- Com esses valores você pode encontrar o período

Medindo a largura de pulso com Capture

A única diferença do método anterior é configurar a borda de ativação como sendo de descida logo após a primeira captura. Então, no meio da leitura, você alterará a configuração da borda no registrador CCP1CON.

Desta forma, na primeira subida ocorrerá uma interrupção e na primeira descida ocorrerá outra interrupção. Assim, você terá a largura do pulso.



Para todos os casos do modo capture, você deve se preocupar com a largura mínima de pulso. Cada instrução em nosso sistema leva 0,5 microssegundo para ser executada, portanto, em frequências muito altas a leitura pode não acontecer apropriadamente porque a largura de pulso é menor que o tempo mínimo que o microcontrolador precisa para executar as instruções.

Encontrado Duty Cycle com módulo compare

O Duty Cycle une os dois conhecimentos que trabalhamos até aqui. Ele consiste na relação entre o período e a largura do pulso, dado em uma porcentagem. Em outras palavras, é a porção ligada do sinal. Precisaremos criar uma função interrupção combinada para extrair tanto o período quanto a largura do sinal.

Mexer com leituras sincronizadas pode gerar todo o tipo de problema, desde imprecisão até quebra total do resultado final. Isso ocorre devido a forma que o microcontrolador executa instruções.

Nós temos que lembrar que a execução das instruções levam tempo (0,5 us em nosso caso) e o próprio timer tem seu tempo de overflow que pode ocorrer no meio da leitura. Por isso, tenha em mente que é importante ajustar seu código de acordo com a frequência que irá trabalhar. Para frequências muito altas, talvez seja necessário utilizar um cristal oscilador mais rápido. Vai depender da aplicação.

Adicionando a frequência à leitura

Como sabemos, $frequencia = 1/periodo$, mas nosso caso, não basta executar este cálculo diretamente.

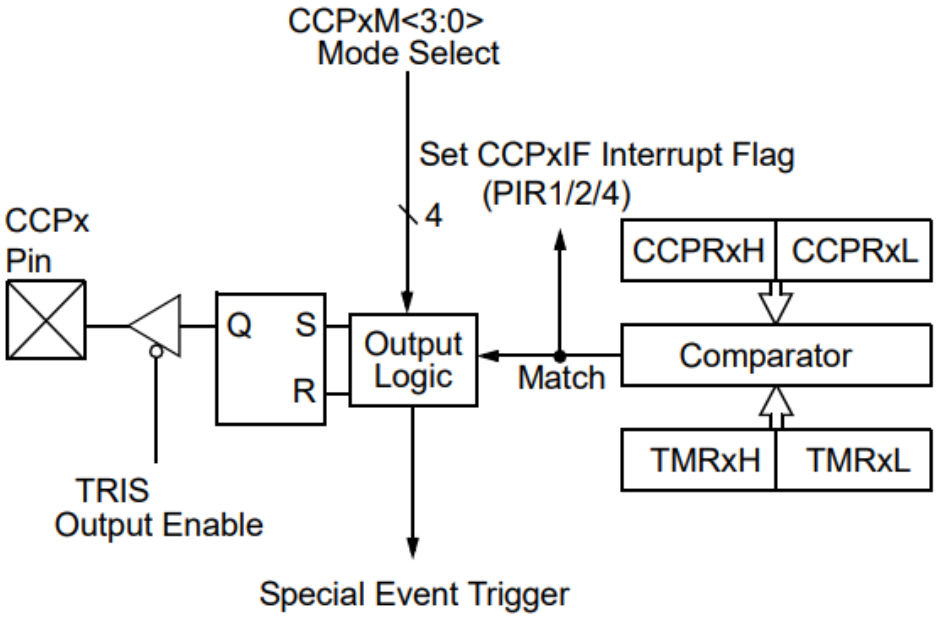
O período deve estar na unidade correta, que é em microssegundos. Até esse momento, trabalhamos com ele como se fosse segundos, algo que não teve problema devido ao tipo de cálculo até agora. Para converter na unidade correta, dividimos período por 1 milhão e depois convertemos para frequência.

```
float f_periodo;
f_periodo = (float) periodo / 1000000;
frequencia = 1. / f_periodo;
```

Modo Compare

É o modo mais simples. Com ele, você pode gerar pulsos no pino CCP e gerar interrupções.

A configuração não é muito diferente do Capture. Se antes você precisava ler o CCPR1H e CCPR1L para capturar um pulso, agora você precisará escrever nestes registradores para gerar um pulso.



O comparator compara este valor ao valor do timer a todo o momento. Quando os valores são iguais, a interrupção é sinalizada (CCP1IF = 1) e uma entre 5 ações podem ser definidas no registrador CCPCON para acontecer:

- 0010: Inverte o pino CCP
- 1000: Seta o pino CCP
- 1001: Limpa o pino CCP
- 1010: Somente gera interrupção
- 1011: Inicia conversão AD (evento especial disponível para este chip)

Gerando um sinal de saída de 100Hz

- Para facilitar, configure o timer para incrementar a cada 1us
- Encontre o período através da frequência (10000us)
- Programe o CCP1CON para togglar na metade deste tempo, assim você terá 5000us de tempo ligado e 5000us de tempo desligado
- Programe a interrupção por CCP e limpe os registradores do timer

É somente isso, cuide para não salvar um valor maior que o timer pode atingir. Com incremento de 1us, o tempo máximo que nosso timer atinge é de 65535us, que é bem acima dos 5000us que programamos, então está correto.

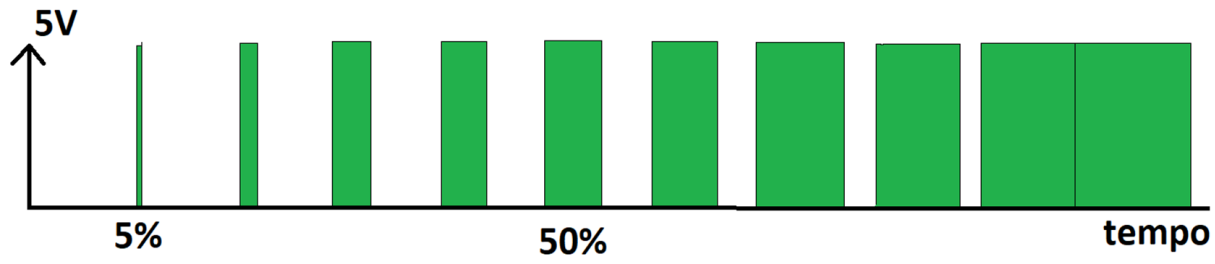
Também é importante limpar os registradores do timer porque você deverá iniciar a contagem de zero sempre que o compare ocorrer.

Gerando sinal de 1Khz

O período de 1Khz é 1 milissegundo. Portanto, nas mesmas configurações do exemplo anterior, basta carregar o valor 500 (500 microssegundos, ou 0,5 milissegundo) nos registradores CCPR e já está feito.

Modo PWM

Modulação por largura de sinal. É um tipo de sinal de alta frequência onde você pode fazer o controle do Duty Cycle. Ele possui várias aplicações, dentre eles, controlar a tensão de saída de uma fonte chaveada



É importante deixar claro que durante um PWM a frequência do sinal deve permanecer a mesma, a única coisa que é alterada é a largura do sinal, ou seja, o duty cycle. Essa variação da largura do sinal resulta em aumento ou diminuição da tensão final.

Acionamento do PWM

- Configure o pino atrelado ao PWM como saída
- Defina o modo PWM no registrador CCPCON
- Definir o Duty Cycle com o registrador CCPR1L
- Configurar o T2CON
- Carregar o PR2 do timer2

Utilizar o PWM compromete a utilização do timer2.

Quem determina o **período** é o PR2 do TMR2

Quem determina o **Duty Cycle** é o CCPRL a

Não é necessário configurar interrupção para zerar o timer2 do PWM. Como vimos no módulo de timer

Na segunda e última fase, o incremento do TMR2 agora vai ser comparado com o PR2, no momento que esses valores coincidirem, o sinal do CCP1 vai para 1 novamente e o ciclo reinicia.

Portanto, o **duty cicle** é definido por CCPR1L/PR2. Já o **período** é definido por CCPR1L + PR2

Encontrando constantes para programar um PWM

As equações estão todas esquematizadas na página 187 do datasheet.

PR2 (Período e frequência)

$$PR2 = \frac{Fosc}{4 * Fpwm * TMR2pre} - 1$$

- PR2 → Registrador
- Fosc → Frequência do oscilador
- Fpwm → Frequência desejada para o PWM
- TMR2pre → Valor do prescaler TMR2

Encontrando o perído a partir de um PR2 definido

$$Tpwm = (PR2 + 1) * 4 * Tosc * TMR2pre$$

- Tpwm → Período do PWM (1/Fpwm)
- PR2 → Registrador do Timer2
- Tosc → Período do oscilador primário (1/Fosc)
- TMR2pre → Valor do prescaler TMR2

CCPRL (duty cycle)

$$CCPRL = \frac{\%DCpwm * Fosc}{100 * Fpwm * TMR2pre}$$

- CCPRL deve ficar entre 0 e 1023 !!!
- DCpwm → Duty Dycle desejado
- Fosc → Frequência do oscilador
- Fpwm → Frequência desejada para o PWM
- TMR2pre → Valor do prescaler TMR2

Encontrando Duty Cycle a partir de um CCPRL defindo

$$DC = \frac{CCPRL}{4 * (PR2 + 1)}$$

- DC → Valor do duty cycle (decimal)
- CCPRL → Registrador
- PR2 → Registrador