SISTEMAS MICROPROCESSADOS I

Prof.: João Castelo

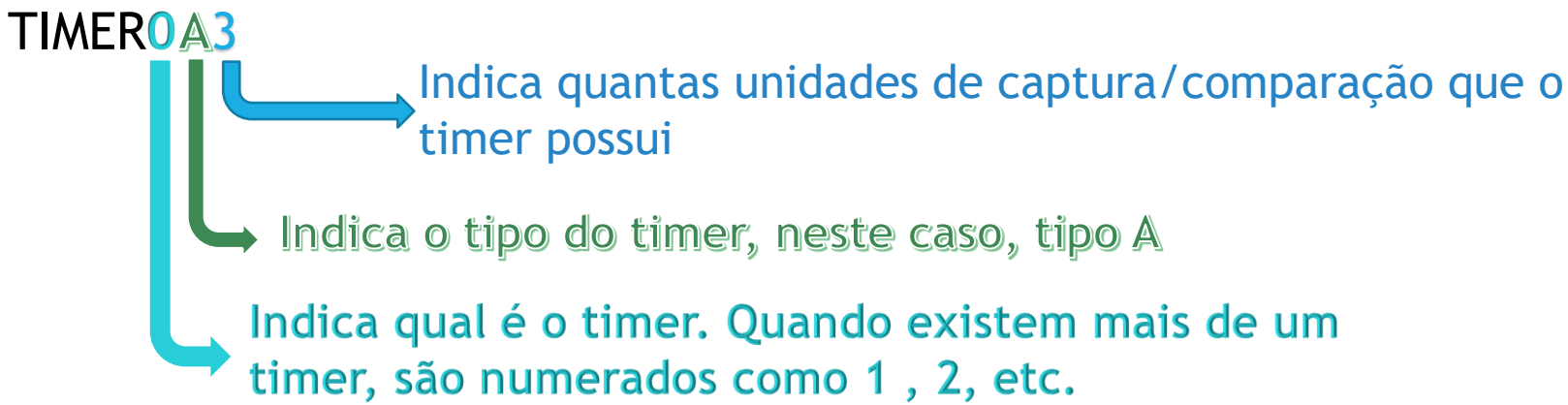
CENTRO UNIVERSITÁRIO
SENAI CIMATEC
SENAI CIMATEC SISTEMA FIEB
Federação das Indústrias do Estado da Bahia

Temporizador/Contador

- É comum nos circuitos que utilizam microcontroladores que se tenha a necessidade de realizar contagens tanto de tempo como de eventos.
- Podemos ver os temporizadores e os contadores como um só elemento, uma vez que o circuito do microcontrolador responsável por ambas as tarefas é o mesmo.
- Isso se deve ao fato de um contador poder contar eventos e assim trabalhar como um contador ou então contar tempo e funcionar como um temporizador.
- Assim sendo um temporizador nada mais é do que um contador que conta o tempo

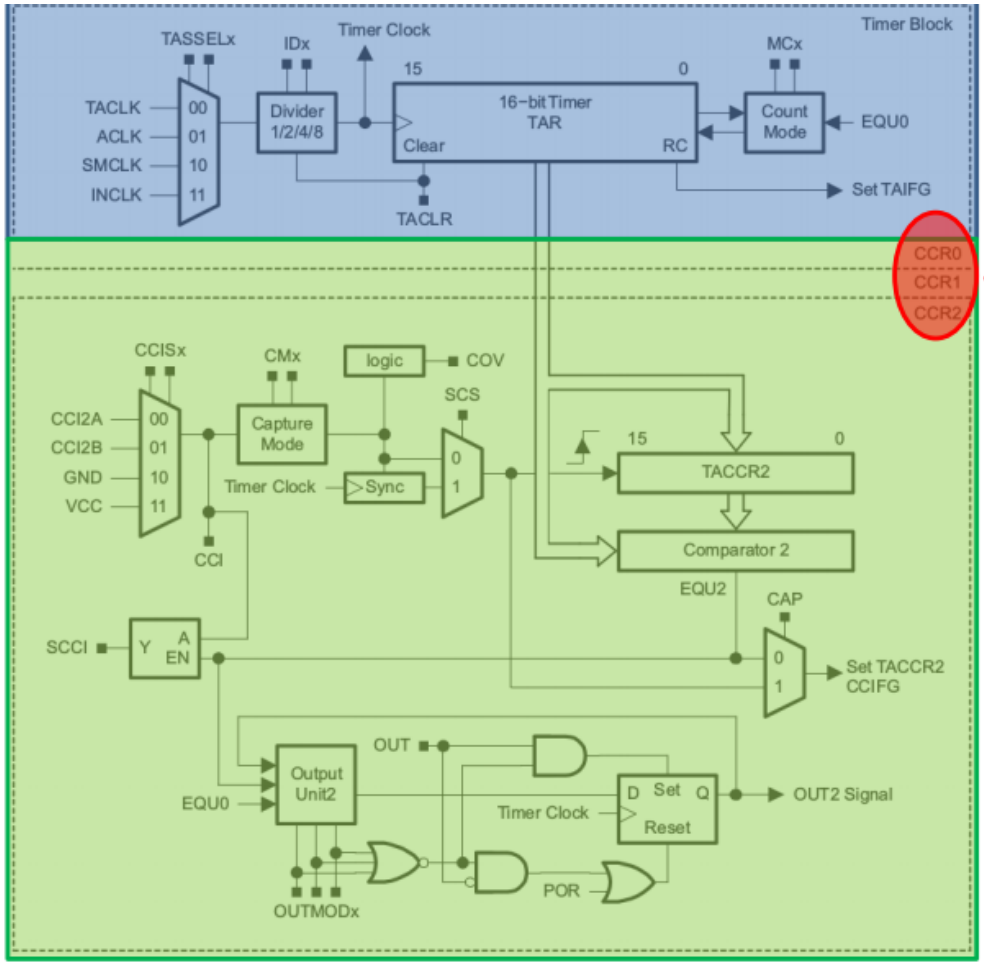
TIMER A

- É um contador/temporizador de 16 bits.
- O TIMER A é composto de dois blocos básicos:
  - Bloco do contador
  - Bloco de captura e comparação;
- A nomenclatura do temporizador/contador indica algumas informações sobre o temporizador:



TIMER A

Bloco Contador



Blocos de Captura e Comparação

São três Blocos de Captura e Comparação

TIMER A

- Um contador assíncrono progressivo/regressivo de 16 bits com módulo programável e capacidade de interrupção.
- Capacidade de operar a partir de diversas fontes de clock interno e externo.
- Até 3 registradores de CCP (ou 5 a depender do modelo) – Captura/Comparação/PWM
  - Captura: medição de período de sinais;
  - Comparação: geração de pulsos de largura programáveis;
  - PWM: geração de sinais de frequência e ciclo ativo programáveis.

TIMER A

Seleção de clock

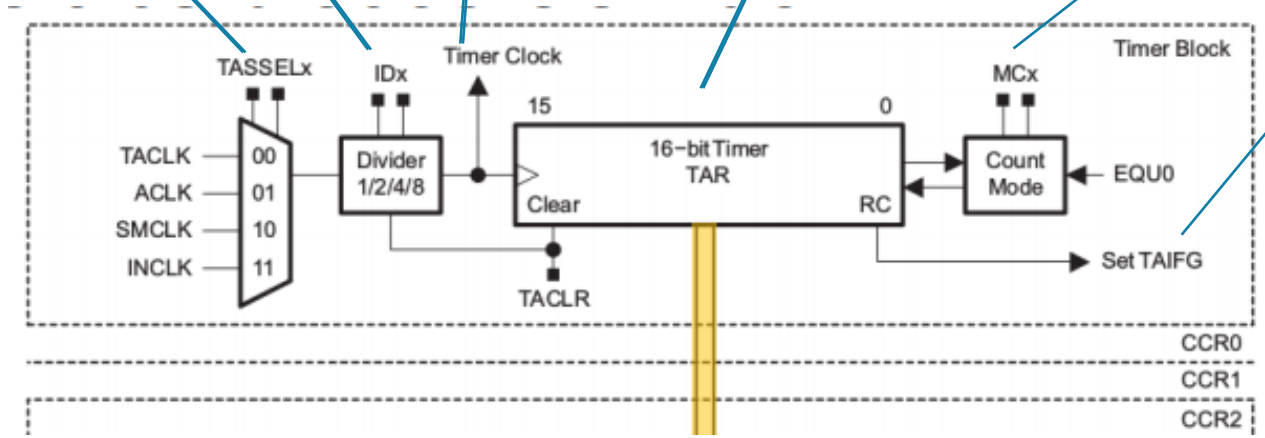
Divisor de clock

Pode usar o clock em outro lugar

Timer A Register

Seleção de modo do contador

Quando há estouro do contador seta o TAIFG



TIMER A

- Os bits TASSSELx, Idx e MCx pertencem ao registrador TACTL.
- O bit MCx seleciona o modo de operação do TIMER A:

MCx	Modo
00	Contagem parada
01	Contagem de módulo (até atingir o módulo de contagem)
10	Contagem contínua (de 0 até 0xFFFF)
11	Contagem progressiva/regressiva

Modos de operação - Bit MCx

- Modo de contagem contínua - MC = 10:

$$T_{INT} = \frac{1}{F_{CLK}/\text{Prescaler}/65536}$$

sendo:

- T_{INT} - Período da interrupção TAIFG em segundos;
- F_{CLK} - Freqüência da fonte de *clock* em Hz;
- *Prescaler* - Fator de divisão do *prescaler* de entrada do *timer*.

Modos de operação - Bit MCx

- Modo de contagem de módulo - MCx = 01:

$$T_{INT} = \frac{1}{F_{CLK}/\text{Prescaler}/(\text{TACCR0}+1)}$$

sendo:

- T_{INT} - Período da interrupção TAIFG em segundos;
- F_{CLK} - Freqüência da fonte de *clock* em Hz;
- *Prescaler* - Fator de divisão do *prescaler* de entrada do *timer*;
- TACCR0 - Módulo da contagem armazenado no registrador TACCR0.

Modos de operação - Bit MCx

- Modo de contagem progressiva/regressiva - MCx = 11 :

$$T_{INT} = \frac{1}{F_{CLK}/\text{Prescaler}/(\text{TACCR0} * 2)}$$

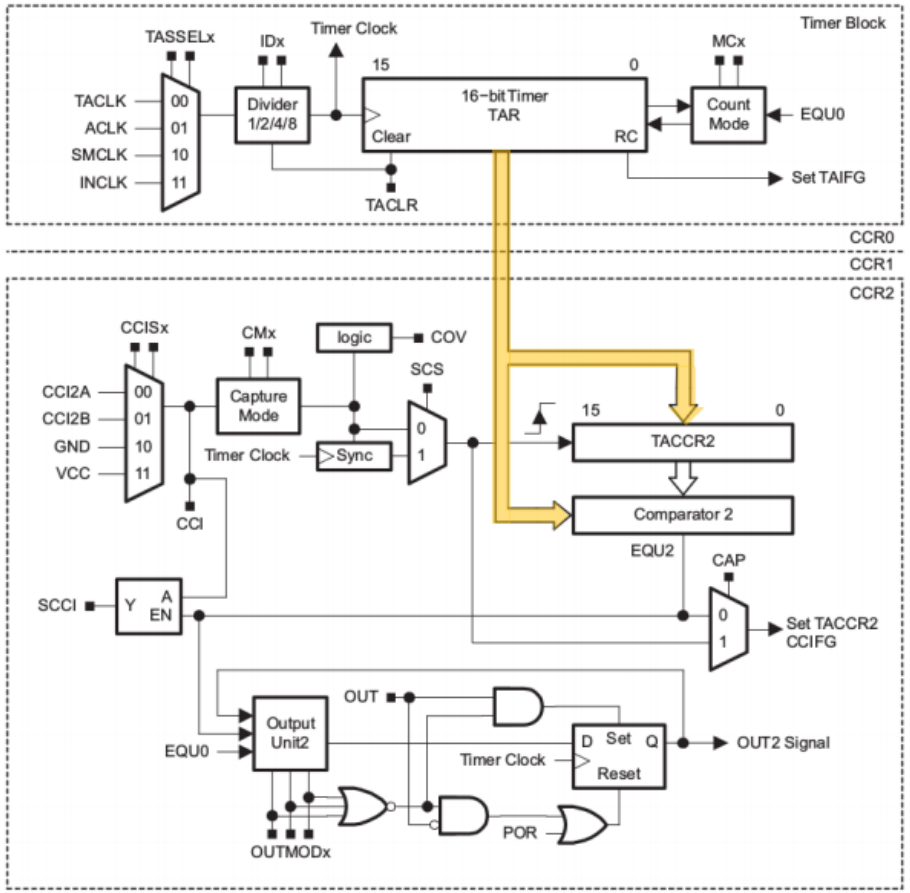
sendo:

- T_{INT} - Período da interrupção TAIFG em segundos;
- F_{CLK} - Freqüência da fonte de *clock* em Hz;
- *Prescaler* - Fator de divisão do *prescaler* de entrada do *timer*;
- TACCR0 - Módulo da contagem armazenado no registrador TACCR0.

- Nesse modo o contador TAR realiza a contagem progressiva desde 0 até o valor programado no registrador TACCR0, depois de atingir o contador passa a contador de forma regressiva até 0. Ao atingir esse valor, a direção da contagem é novamente invertida e todo o processo tem início novamente.

Blocos CCR

- Existe um barramento de dados de 16 bits interligando o contador ao três blocos CCR

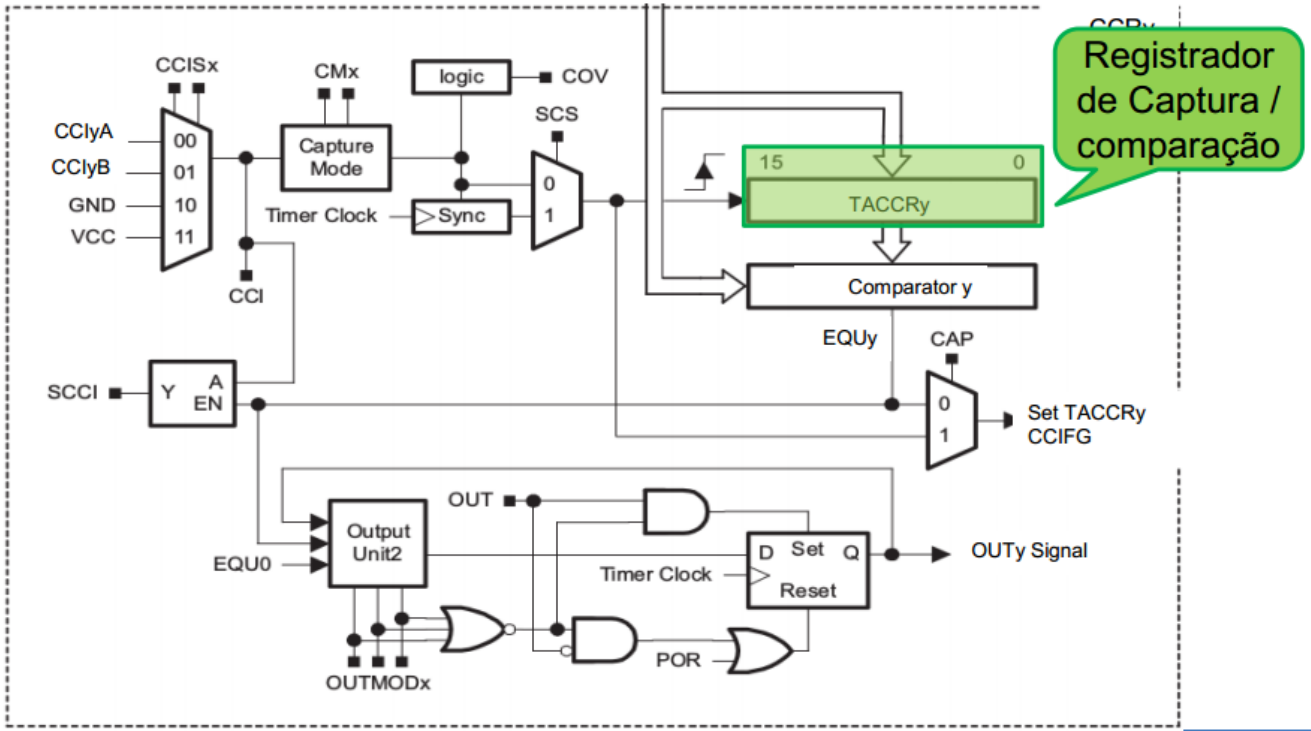


Blocos CCR

- São três blocos (ou mais) iguais na maioria dos dispositivos;
- Alguns dispositivos possuem apenas um ou dois blocos de captura e comparação;
- A quantidade destes blocos é especificada pelo sufixo no nome do timer (TIMERxA**3**)

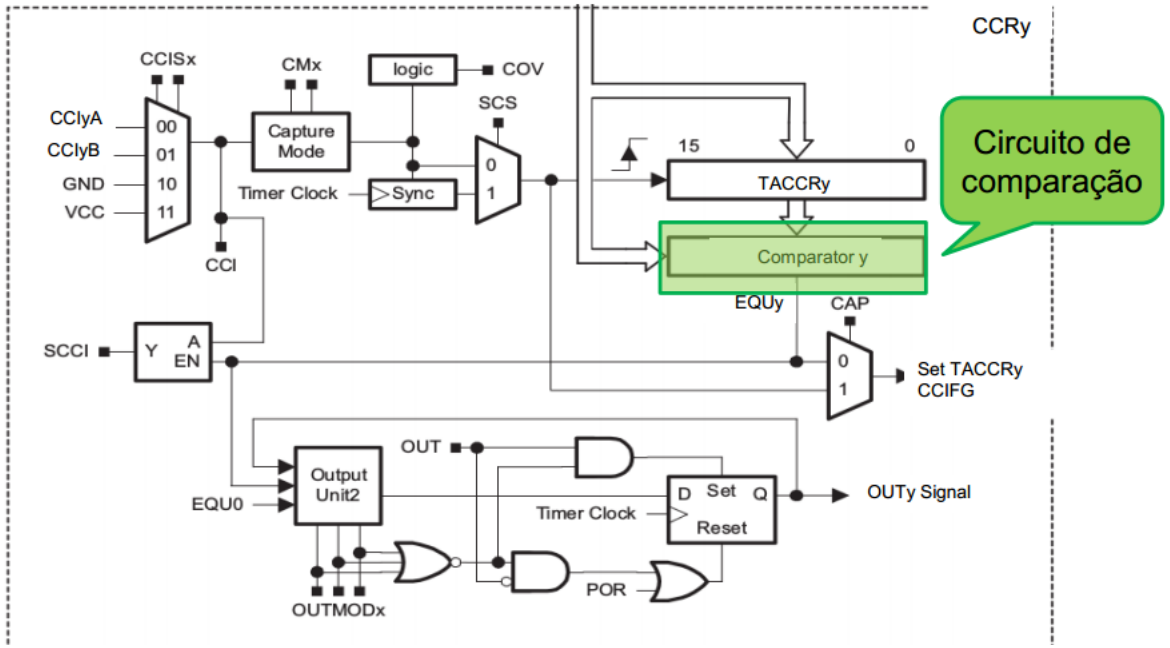
Registrador TAxCCRy

- O registrador TAxCCRy guarda o valor capturado no modo de captura ou contém o valor a ser comparado com o TAR no modo de comparação.



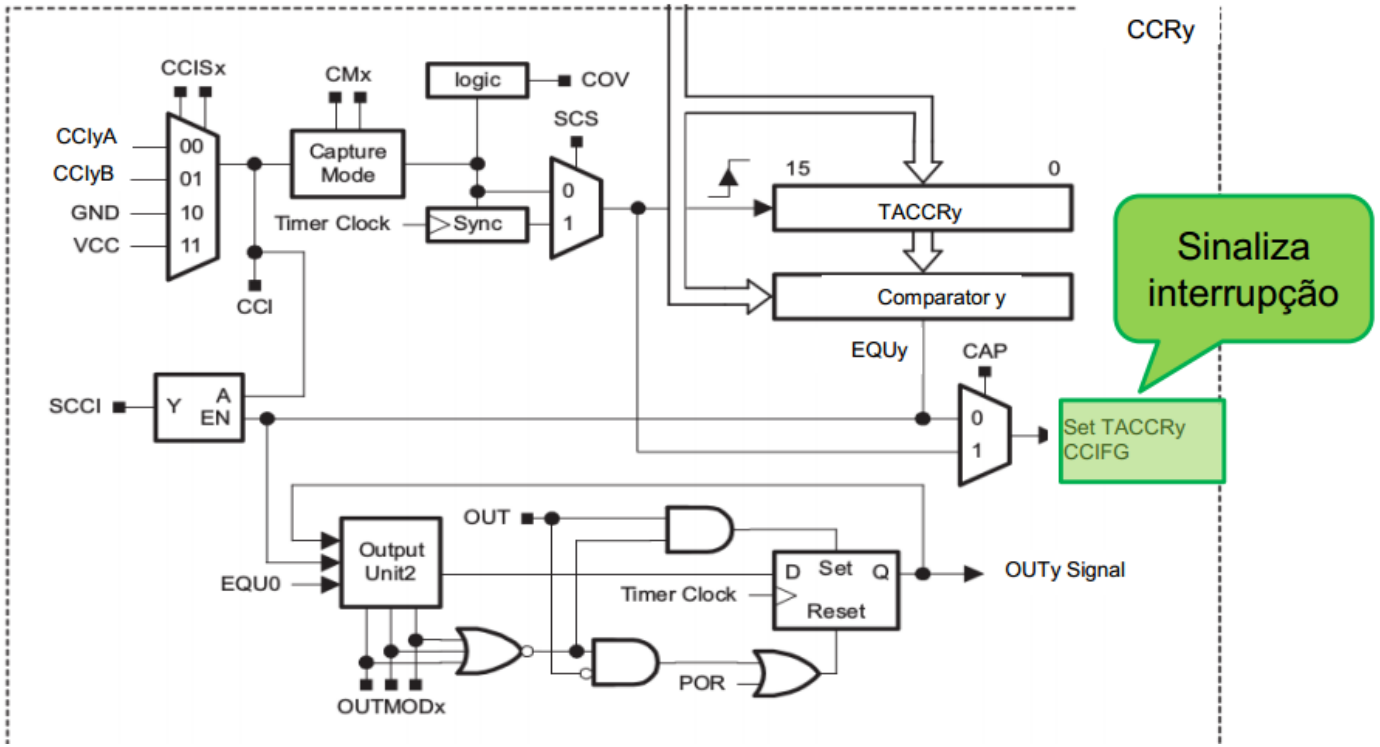
Circuito de Comparação

- Este circuito compara o valor do TAR com o valor em TAxCCRy no modo de comparação.



Flag CCIFG – Registrador TACCRy

- Cada bloco tem a capacidade de gerar interrupção independentemente um do outro.



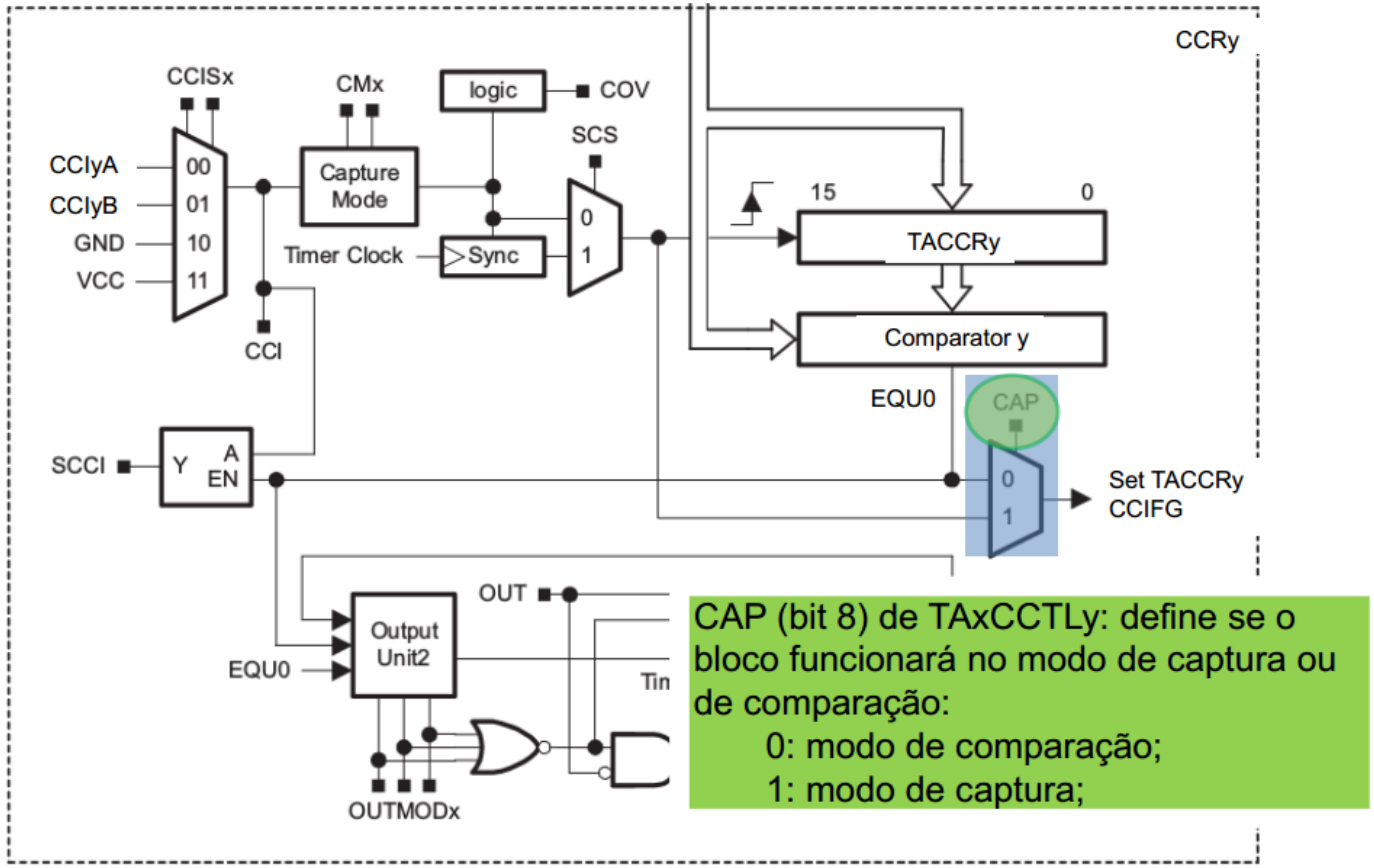
Blocos de Captura/Comparação CCRo

- O bloco CCR0 tem prioridade sobre os demais:
  - O registrador TACCR0 é usado para determinar o período de contagem do temporizador nos modos UP e UP-DOWN;
  - Sua interrupção tem prioridade maior que os demais blocos, inclusive sobre a interrupção do bloco contador (TAR);

Registrador TAxCCTLy

- Configura o bloco de captura e comparação do temporizador/contador;
- É um registrador de 16 bits acessado por software;
- Existe um registrador específico para cada unidade de captura e comparação:
  TAxCCTL0: registrador de configuração do CCR0;
  TAxCCTL1 : registrador de configuração do CCR1;
  TAxCCTL2: registrador de configuração do CCR2;

Modo de captura ou comparação



CAP (bit 8) de TAxCCTLy: define se o bloco funcionará no modo de captura ou de comparação:

0: modo de comparação;

1: modo de captura;

Modo de Captura

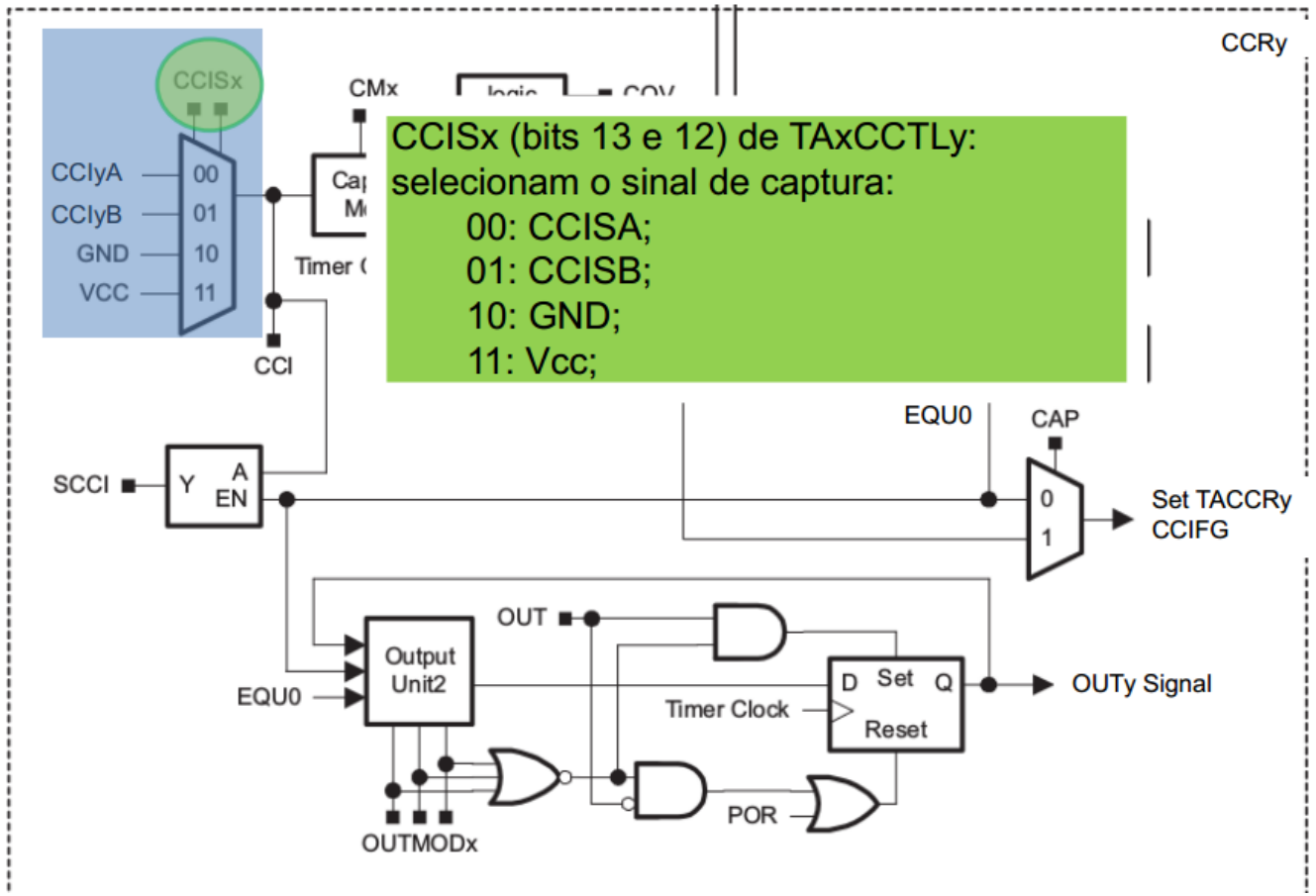
- Pode ser utilizado para medição de período de sinais ou outras medições de tempo em que se requeiram máxima precisão e mínima intervenção da CPU.

- Exemplo: sensor numa linha de produção para contar passagem de peças:
  - Cada vez que a peça passa pela linha de produção o sensor gera um pulso;
  - Compara o valor atual do TAR com o valor do TACCRx (x corresponde ao módulo que está o sensor), então tem-se uma base de tempo entre quando passa pelo primeiro sensor e quando passa pelo segundo sensor;
  - Logo, consegue medir tempo de eventos externos ou frequência.

Modo de Captura

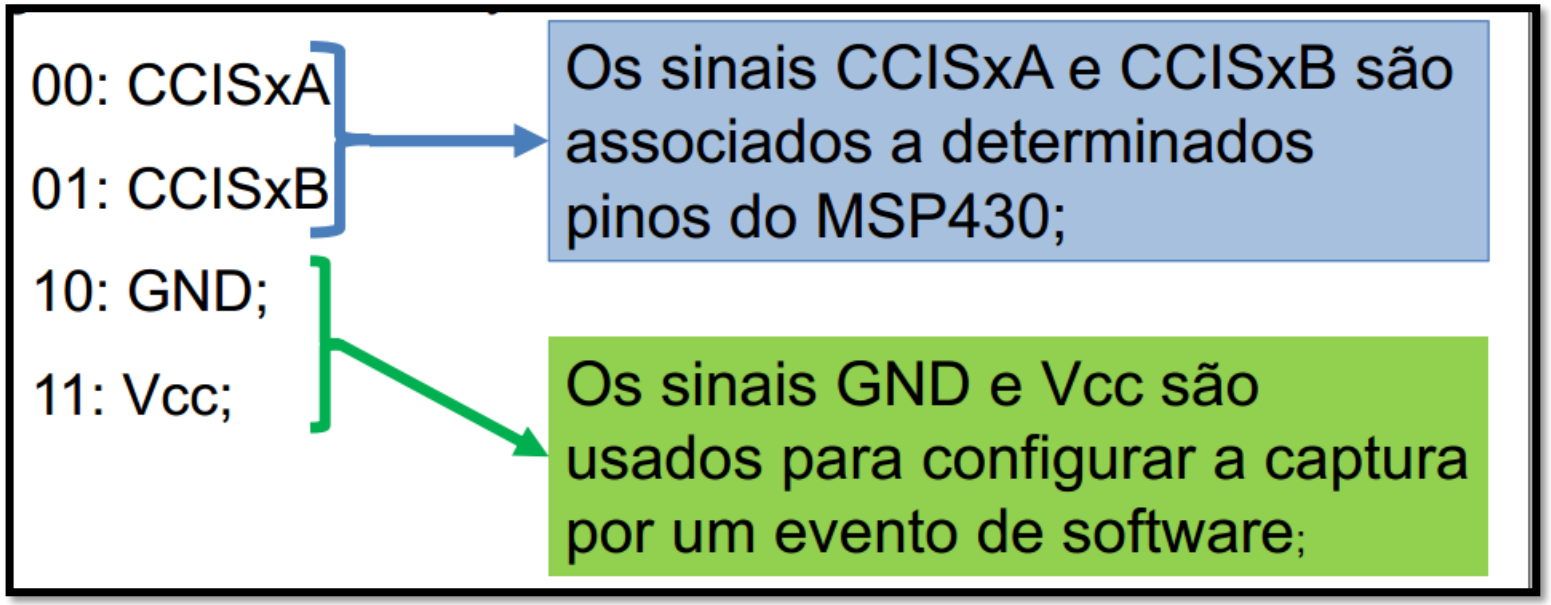
- No modo de captura, o valor do TAR é copiado para o registrador TAxCCRy quando um sinal for aplicado na entrada de captura;
- O TAR continua a contagem enquanto o processo de captura ocorre;
- É possível configurar uma interrupção quando for realizada a captura.

Seleção do sinal de captura

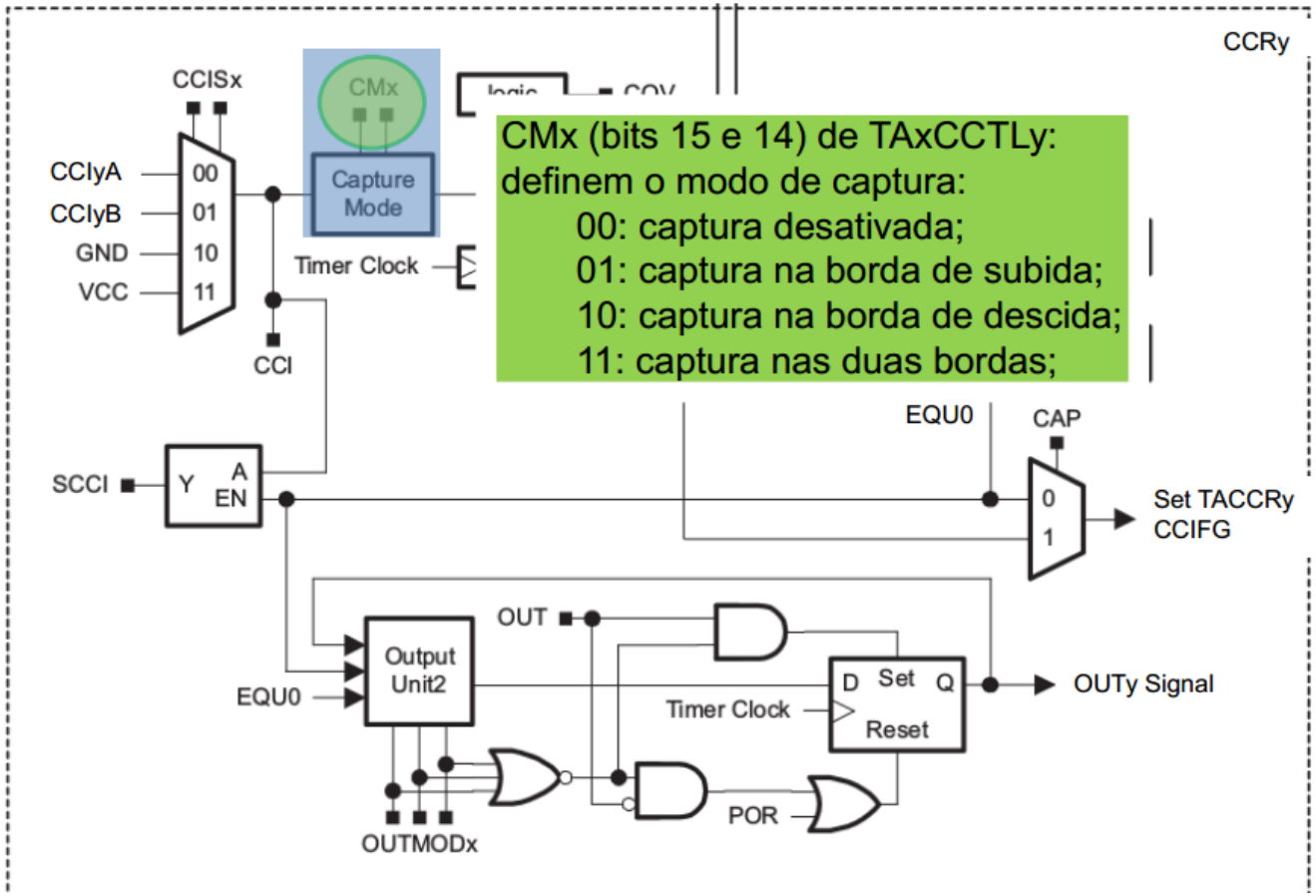


Seleção do sinal de captura

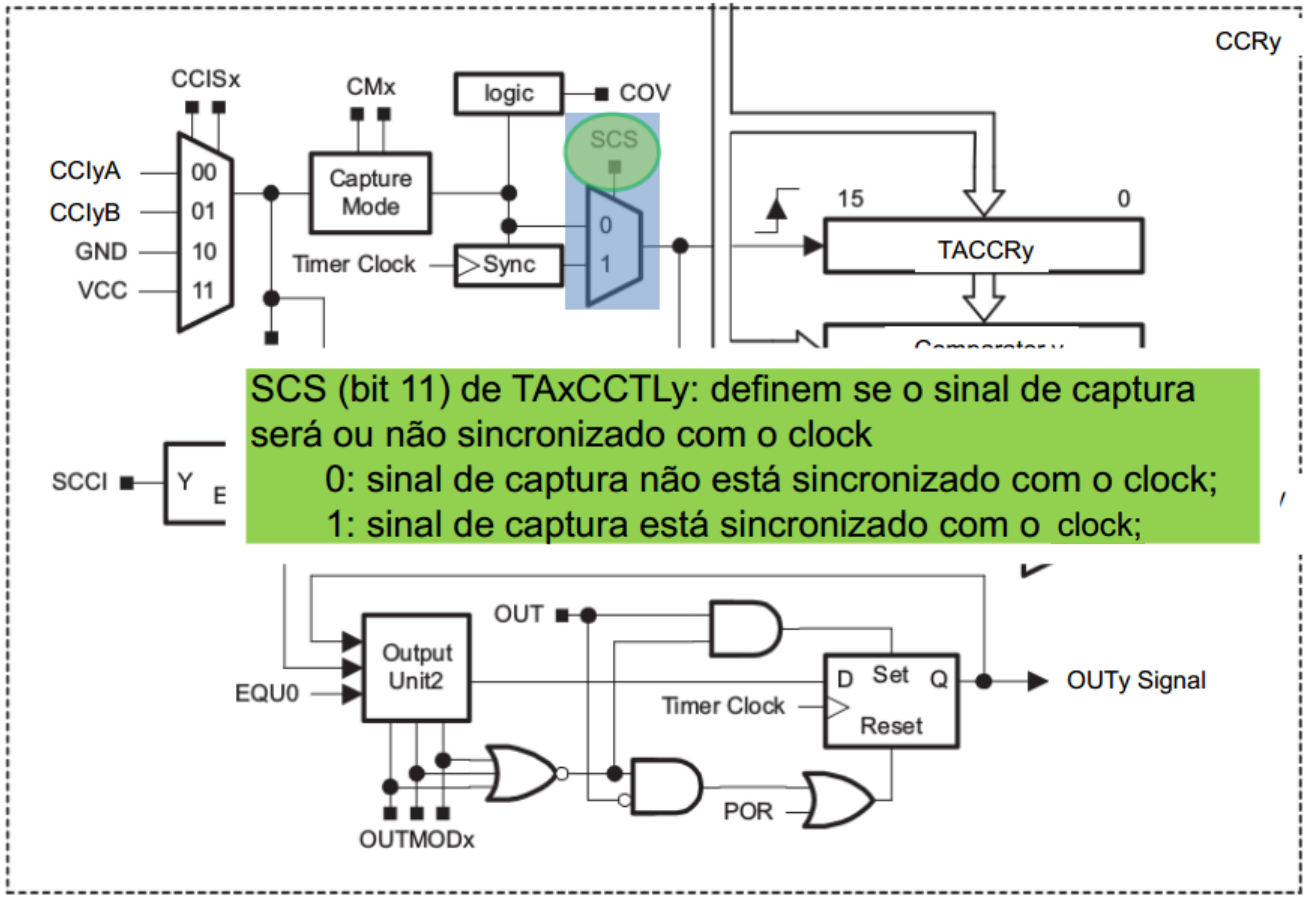
- O sinal de captura é selecionado pelos bits CCISx do registrador TAxCCTLy:



Seleção do modo de captura



Sincronização do sinal de captura



Características – Modo de Captura

- O bit COV do registrador TAxCCTLy indica se ocorreu uma sobrescrita do valor capturado.
  - Caso após uma comparação o conteúdo do TACCRx não tenha sido lido pelo software e ocorra uma nova comparação, o novo período será armazenado no TACCRx – o que levará a perda do valor anterior – logo, o bit COV é setado.
- Para o caso de interrupção, o bit CCIFG deve ser resetado por software para os blocos CCR1 e CCR2. Para o bloco CCR0 este bit é resetado automaticamente pelo hardware;
- Uma aplicação típica para o modo de captura é a medida de largura de pulso;

Modo de Captura –medição da largura de pulso

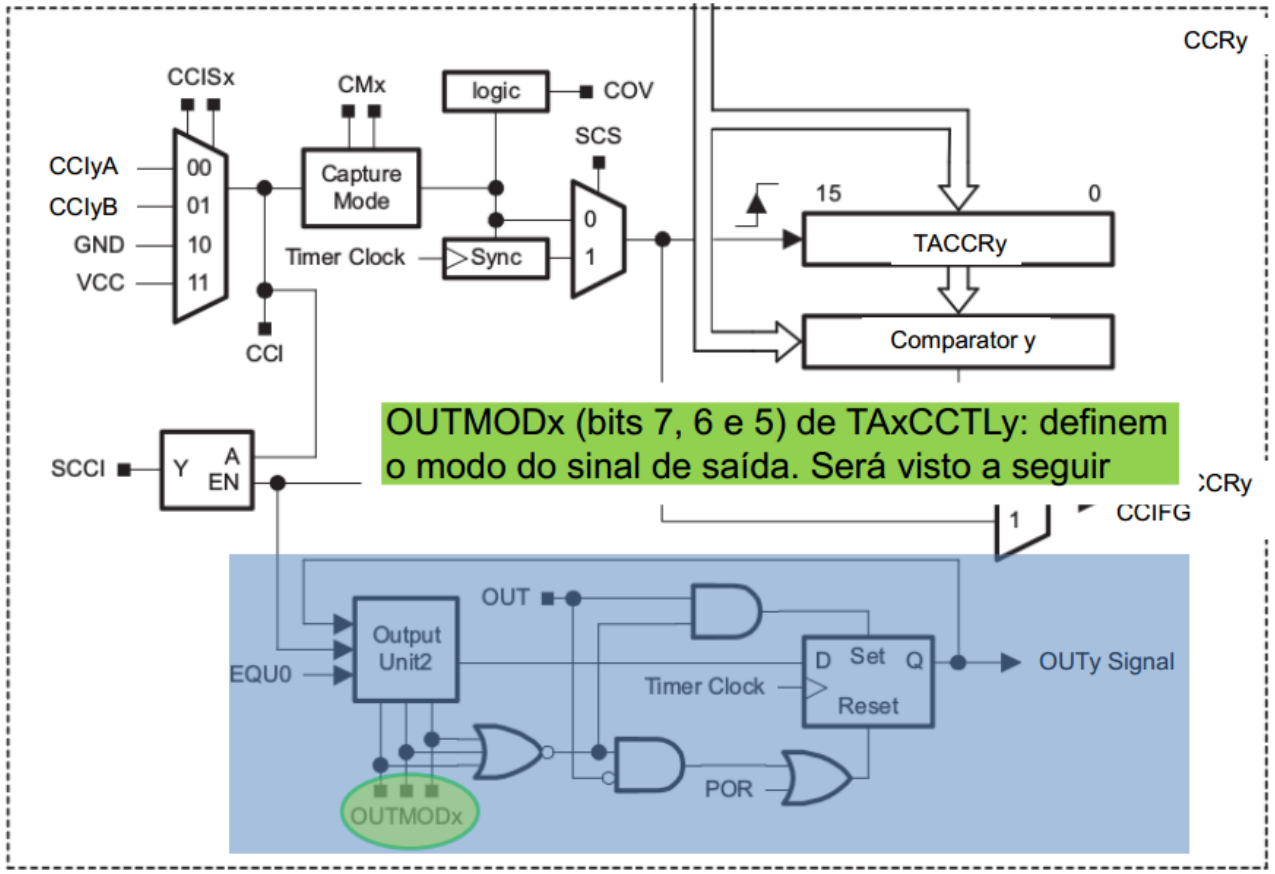
- Configura-se o bloco CCRx para o modo de captura;
- Seleciona-se por qual borda do sinal a ser medido será realizada a captura;
- Quando a primeira captura for realizada, na rotina de interrupção, armazena-se o valor capturado;
- Quando a segunda captura for realizada, subtrai-se o valor da captura atual com o valor armazenado da primeira captura;
- O período do sinal será a diferença entre duas medições (capturas) consecutivas multiplicado pelo tempo do período do clock do TIMER;
- É possível medir até três períodos simultaneamente – para dispositivos com 3 módulos de comparação (CCR);

Modo de Comparação/PWM

- Pode ser utilizado principalmente para geração de pulsos ou interrupções com intervalos de tempo precisos ou também para a geração de sinais PWM.
- Esse modo é selecionado apagando o bit CAP (registrador TACCTLx).
- O seu funcionamento está relacionado diretamente às unidades de saída (Outputs Units) existentes no hardware do timer.
- Basicamente, uma unidade de saída consiste em um comparador digital que compara a contagem do TAR com o valor programado num dos registradores TACCRx.
- Quando as duas contagens são iguais é gerado um sinal EQUx.
- O sinal EQUx é utilizado para gerar o sinal de comparação OUTx, conforme definido pelos bits OUTMODx.

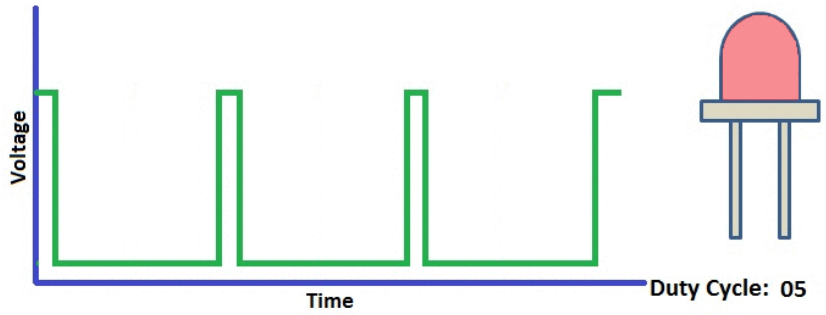
Modo de Comparação/PWM

- Modo do sinal de saída



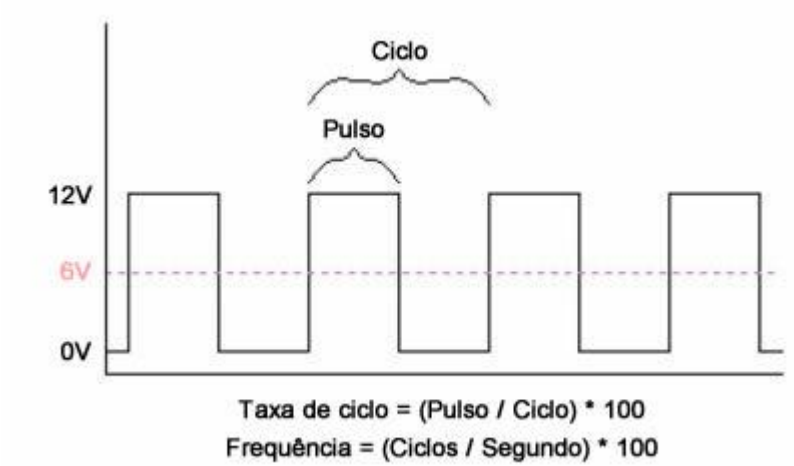
Revisando PWM

- **PWM** (Pulse Width Modulation) refere-se ao conceito de pulsar rapidamente um sinal digital em um condutor.
- Às vezes você precisa de mais do que apenas "ligar" e "desligar" no controle de dispositivos. Caso você deseje controlar o brilho de um LED (ou qualquer lâmpada) ou a velocidade de um motor elétrico CA, simplesmente não será possível aplicando somente o controle (ligar/desligar). **Para contornar esta situação, habilmente foi desenvolvida a técnica chamada PWM ou Pulse Width Modulation.**



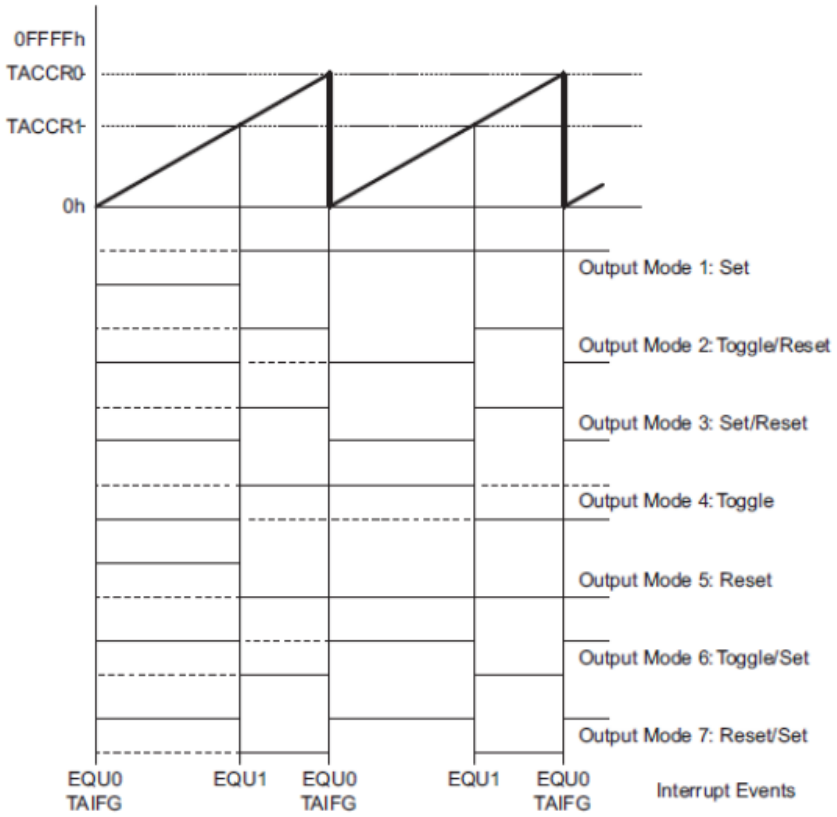
Definições - PWM

- **Ciclo ou Período** – o intervalo de tempo entre a subida de um pulso (dado em segundos);
- **Frequência** – a taxa de bordas de subida de um pulso (dado em Hz ou ciclos por segundo). É simplesmente o inverso do período;
- **Taxa de Ciclo** – tempo no período em que o pulso está ativo ou alto, dividido pelo tempo de ciclo (é dado em porcentagem do período completo)



Bits OUTMODx

- Exemplo de saída para o modo UP



Modo	OUTMODx	Modo	Descrição
0	000	Saída direta	Neste modo, o estado do sinal OUTx é definido diretamente pelo estado do *bit* OUT (registrador TACCTLx).
1	001	Setar	A saída é setada quando a contagem do TAR atinge o valor programado no TACCRx. A saída permanece setada até a seleção de outro modo de saída ou um *reset* do *timer*.
2	010	Inverter/ apagar	A saída tem seu estado lógico invertido quando a contagem do TAR atinge o valor do TACCRx e apagada quando a contagem do TAR atinge o valor do TACCR0. Este modo não é útil no canal 0, tendo em vista que ele é utilizado para o controle do período do sinal.
3	011	Setar/ apagar	A saída é setada quando a contagem do TAR atinge o valor do TACCRx e apagada quando a contagem do TAR atinge o valor do TACCR0. Este modo não é útil no canal 0, tendo em vista que é utilizado para o controle do período do sinal.
4	100	Inverter	A saída tem seu estado lógico invertido cada vez que a contagem do TAR atinge o valor programado no TACCRx. O período do sinal de saída é igual ao dobro do período de contagem total do *timer*.
5	101	Apagar	A saída é apagada quando a contagem do TAR atinge o valor programado no TACCRx. A saída permanece apagada até a seleção de outro modo de saída ou um *reset* do *timer*.
6	110	Inverter/ setar	A saída tem o seu estado lógico invertido cada vez que a contagem do TAR atinge o valor programado no TACCRx e setada quando a contagem do TAR atinge o valor programado no TACCR0. Este modo não é útil no canal 0, tendo em vista que ele é utilizado para o controle do período do sinal.
7	111	Apagar/ setar	A saída é apagada quando a contagem do TAR atinge o valor programado no TACCRx e setada quando a contagem do TAR atinge o valor programado no TACCR0. Este modo não é útil no canal 0, tendo em vista que é utilizado para o controle do período do sinal.

Vamos ajustar o sinal PWM

PWM

- Os modos 3 e 7 são os mais indicados para a geração de sinais PWM.
- O registrador TACCR0 é utilizado para determinar o período do sinal.
- O registrador TACCRx determinao duty cycle.
- No modo 3 temos um sinal PWM ativo em nível 0.
- No modo 7 temos um sinal PWM ativo em nível 1.
- Os modos 2 e 6 são indicados para a geração de sinais PWM complementares.

REGISTRADOR TACTL

Endereço	Nome		BIT 15	BIT 14	BIT 13	BIT 12	BIT 11	BIT 10	BIT 9	BIT 8
0x0160 0x0180	TACTL TA1CTL	Leitura Escrita	Não utilizados						TASSELx	
		Reset	0	0	0	0	0	0	0	0
			BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
		Leitura Escrita	IDx		MCx		Não utilizado	TACLR	TAIE	TAIFG
		Reset	0	0	0	0	0	0	0	0

TASSELx - seleção da fonte de *clock* do *timer* A:
00 - TACLK (*clock* externo) (símbolo **TASSEL_0**);
01 - *clock* auxiliar (ACLK) (símbolo **TASSEL_1**);
10 - *clock* secundário (SMCLK) (símbolo **TASSEL_2**);
11 - INCLK (nos *chips* 4xx é o sinal TACLK invertido) (símbolo **TASSEL_3**).

IDx - divisor do *clock* do *timer* A:

IDx	Fator de divisão	Símbolo
00	1	**ID_0**
01	2	**ID_1**
10	4	**ID_2**
11	8	**ID_3**

MCx - seleção do modo de funcionamento do *timer* A:
00 - parado (símbolo **MC_0**);
01 - contagem progressiva com módulo (0 até o valor de TACCR0) **(símbolo MC_1)**;
10 - contínuo (0 até 65.535 e depois reinicia do zero) **(símbolo MC_2)**;
11 - crescente/decrescente (contagem de 0 até TACCR0 e depois até 0 novamente) **(símbolo MC_3)**.

TACLR - *reset* do *timer* A: setando esse *bit*, a contagem do TAR, do divisor de *clock* e a direção de contagem são apagadas. O *bit* retorna automaticamente a zero após o *reset* do *timer* (símbolo **TACLR**);

TAIE - habilitação da interrupção de estouro de contagem do TAR:
0 - interrupção desabilitada;
1 - interrupção habilitada (símbolo **TAIE**).

TAIFG - sinalizador de interrupção do TAR:
0 - nenhuma interrupção pendente;
1 - o TAR estourou a contagem e há uma interrupção pendente (símbolo **TAIFG**).

REGISTRADOR TAR

Endereço	Nome		*BIT* 15	*BIT* 14	*BIT* 13	*BIT* 12	*BIT* 11	*BIT* 10	*BIT* 9	*BIT* 8
0x0170 0x0190	TAR TA1R	Leitura Escrita	TARx							
		Reset	0	0	0	0	0	0	0	0
			BIT 7	*BIT* 6	*BIT* 5	*BIT* 4	*BIT* 3	*BIT* 2	*BIT* 1	*BIT* 0
		Leitura Escrita	TARx							
		Reset	0	0	0	0	0	0	0	0

TARx - 16 *bits* do registrador de contagem do *timer* A.

REGISTRADOR TACCTLx

Endereço	Nome		BIT 15	BIT 14	BIT 13	BIT 12	BIT 11	BIT 10	BIT 9	BIT 8
0x0162	TACCTL0	Leitura	CMx		CCISx		SCS	SCCI	Não utilizado	CAP
0x0164	TACCTL1	Escrita						-		
0x0166	TACCTL2	*Reset*	0	0	0	0	0	0	0	0
0x0182	TAICCTL0		BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
0x0184	TAICCTL1	Leitura	OUTMODx			CCIE	CCI	OUT	COV	CCIFG
0x0186	TAICCTL2	Escrita					-			
0x0188	TAICCTL3	*Reset*	0	0	0	0	0	0	0	0
0x018A	TAICCTL4									

CMx - seleção do modo de captura:
00 - não realiza capturas (símbolo **CM_0**);
01 - captura na borda de subida (símbolo **CM_1**);
10 - captura na borda de descida (símbolo **CM_2**);
11 - captura em ambas as bordas (símbolo **CM_3**).

CCISx - seleção da entrada de captura/comparação:
00 - CCIxA (símbolo **CCIS_0**);
01 - CCIxB (símbolo **CCIS_1**);
10 - GND (símbolo **CCIS_2**);
11 - V_{CC} (símbolo **CCIS_3**).

SCS - sincronização da entrada de captura com o *clock* interno:
0 - captura assíncrona;
1 - captura síncrona (símbolo **SCS**).

SCCI - entrada sincronizada de captura/comparação. Esse sinal é sincronizado com o sinal EQUx e permite ler o estado da entrada CCI (símbolo **SCCI**).

CAP - seleção do modo de captura/comparação:
0 - modo de comparação;
1 - modo de captura (símbolo **CAP**).

REGISTRADOR TACCTLx

OUTMODx - seleção do modo de comparação:

000 - a saída recebe o valor do *bit* OUT (símbolo **OUTMOD_0**);

001 - a saída é setada na comparação (símbolo **OUTMOD_1**);

010 - a saída é invertida na comparação e apagada ao atingir o valor do TACCR0 (símbolo **OUTMOD_2**);

011 - a saída é setada na comparação e apagada ao atingir o valor do TACCR0 (símbolo **OUTMOD_3**);

100 - a saída é invertida a cada comparação (símbolo **OUTMOD_4**);

101 - a saída é apagada na comparação (símbolo **OUTMOD_5**);

110 - a saída é invertida na comparação e setada ao atingir o valor do TACCR0 (símbolo **OUTMOD_6**);

111 - a saída é apagada na comparação e setada ao atingir o valor do TACCR0 (símbolo **OUTMOD_7**).

CCIE - habilitação da interrupção de captura/comparação do canal:

0 - interrupção desabilitada;

1 - interrupção habilitada (símbolo **CCIE**).

CCI - entrada de captura/comparação: permite ler o estado do sinal da entrada de captura/comparação (símbolo **CCI**).

OUT - estado do sinal de saída (OUTx) do canal. No modo 0, esse *bit* controla diretamente o estado da saída de comparação:

0 - saída em nível lógico "0";

1 - saída em nível lógico "1" (símbolo **OUT**).

COV - indicador de *overflow* de captura, ou seja, uma segunda captura foi feita antes da leitura da primeira:

0 - não ocorreu *overflow* de captura;

1 - ocorreu um *overflow* de captura (símbolo **COV**).

CCIFG - sinalizador de interrupção de captura/comparação:

0 - nenhuma interrupção pendente;

1 - uma interrupção de captura/comparação está pendente (símbolo **CCIFG**).

REGISTRADOR TACCRx

Endereço	Nome		BIT 15	BIT 14	BIT 13	BIT 12	BIT 11	BIT 10	BIT 9	BIT 8
0x0172	TACCR0	Leitura	TACCRx							
0x0174	TACCR1	Escrita								
0x0176	TACCR2	*Reset*	0	0	0	0	0	0	0	0
0x0192	TA1CCR0		BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
0x0194	TA1CCR1	Leitura	TACCRx							
0x0196	TA1CCR2	Escrita								
0x0198	TA1CCR3	*Reset*	0	0	0	0	0	0	0	0
0x019A	TA1CCR4									

TACCRx - 16 *bits* do registrador de captura/comparação do canal x. Lembrando que nos modos 1 e 3 do *timer*, o registrador TACCR0 funciona como registrador de módulo de contagem do TAR.

REGISTRADOR TAIV

- Caso ocorra uma interrupção do canal 0 do timer e ela esteja habilitada, o fluxo do programa será desviado para o vetor indicado na tabela de vetores de interrupções do chip.
- Para os demais canais as interrupções compartilham o mesmo vetor de interrupções, logo é necessário um mecanismo que permita identificar qual delas provocou o desvio do programa.
- Esse mecanismo existe sob a forma do gerador de vetor de interrupções – TAIV.

REGISTRADOR TAIV

Endereço	Nome		*BIT* 15	*BIT* 14	*BIT* 13	*BIT* 12	*BIT* 11	*BIT* 10	*BIT* 9	*BIT* 8
0x012E 0x011E	TAIV TA1IV	Leitura	0	0	0	0	0	0	0	0
		Escrita	-	-	-	-	-	-	-	-
		Reset	0	0	0	0	0	0	0	0
			BIT 7	*BIT* 6	*BIT* 5	*BIT* 4	*BIT* 3	*BIT* 2	*BIT* 1	*BIT* 0
		Leitura	0	0	0	0	TAIVx			0
		Escrita	-	-	-	-				-
		Reset	0	0	0	0	0	0	0	0

TAIV	Fonte de interrupção	*Flag* de interrupção	Prioridade
0x00	Nenhuma interrupção	-	
0x02	CCP canal 1	TACCR1:CCIFG TA1CCR1:CCIFG *	Maior
0x04	CCP canal 2	TACCR2:CCIFG TA1CCR2:CCIFG *	
0x06	CCP canal 3 *	TA1CCR3:CCIFG *	
0x08	CCP canal 4 *	TA1CCR4:CCIFG *	
0x0A	Estouro do *timer*	TACTL:TAIFG	
0x0C	Reservado	-	
0x0E	Reservado	-	Menor

* nos *chips* com cinco canais CCP

Exemplo

- Gerar um sinal de onda quadrada com período configurável utilizando o modo de comparação do Timer A.
- O timer A está configurado para utilizar como fonte de clock o sinal SMCLK – aproximadamente 800 KHz.
- Ao passar pelo divisor de entrada – programado para 8 – o sinal de clock que chega ao TAR é de aproximadamente 100 KHz.
- O que implica que cada unidade de contagem do TAR vale aproximadamente 10us.

Exemplo

```
#include <io430x14x.h>
#include <intrinsics.h>
unsigned int intervalo;
#pragma vector = TIMERA1_VECTOR
__interrupt void trata_timer(void)
{
  switch (TAIV)
  {
    case 0x02 : // interrupção do canal 1
                TACCR1 += intervalo;
                break;
    case 0x04 : // interrupção do canal 2
                break;
    case 0x06 : // interrupção do canal 3
                break;
    case 0x08 : // interrupção do canal 4
                break;
    case 0x0A : // interrupção de estouro do timer
                break;
  }
}

int main( void )
{
  WDTCTL = WDTPW + WDTHOLD;      // desativa o watchdog
  P1DIR_bit.P1DIR_2 = 1; // configura o pino P1.2 como saída
  // configura o timer A:
  // TASSEL_2 -> fonte de clock = SMCLK
  // ID_3 -> divisão por 8 do clock de entrada
  // MC_2 -> modo contínuo (contagem de 0 a 65535)
  TACTL = TASSEL_2+ID_3+MC_2;
  intervalo = 25000;
  TACCR1 = intervalo;            // ponto de disparo do CCP1
  // configura o canal CCP1:
  // OUTMOD_4 -> modo de saída 4 (inversão da saída a cada comparação)
  // CCIE -> interrupção de comparação do CCP1 habilitada
  TACCTL1 = OUTMOD_4+CCIE;
  P1SEL_bit.P1SEL_2 = 1;         // função alternativa para P1.2
  __bis_SR_register(GIE);        // habilita as interrupções
  while(1);                      // loop infinito
}
```

Exercícios

- Implemente os exemplos 5-6 e 5-7 (páginas 155 e 156) do livro de Fábio Pereira no IAR.

Trabalhos Práticos

- As equipes devem apresentar os seguintes temas:
  - Equipe 1: Prática 2 – Luca Torres 22/05
  - Equipe 2: Basic Timer (Timer 1) – Pedro Menezes – 22/05
  - Equipe 3: Prática 2 – Eduardo Fiscina – 24/05
  - Equipe 4: Timer B – Eduardo Nery – 24/05
  - Equipe 5: Prática 2 – Carlos Henrique – 29/05
  - Equipe 6: Prática 2 –Arley Matos – 29/05
- As equipes que ficaram com o Timer 1 ou Timer B devem apresentar um exemplo prático simples de configuração do Timer A.
- As apresentações ocorrerão a partir do dia **22/05/2023**
- Obrigatória a apresentação de todos os integrantes.

PRÁTICA 2

- Escreva um programa para controlar o brilho do LED verde usando um sinal PWM gerado pelo bloco CCR1 do TIMER0;
- Use um sinal PWM com frequência de 100Hz;
- O controle do *duty-cycle* do sinal PWM deve ser feito pelo botão no pino P1.3, **usando interrupção**:
- Cada vez que o botão for pressionado, o *duty-cycle* aumenta em 10% até atingir o máximo;
- Quando atinge o máximo, cada vez que o botão for pressionado, o *duty-cycle* diminui em 10% até atingir o mínimo;
- Na função *main*, coloque o MSP430 no modo LPM0;

Trabalho Prático

- A prática deve ser feita em ambiente de programação para microcontroladores
- O circuito pode ser simulado no Proteus.
- A apresentação deverá ter:
  - Contextualização;
  - Fundamentação Teórica;
  - Materiais e Métodos;
  - Descritivo da solução;
  - Resultados;
  - Conclusão.

Referências

- O que é PWM e Para que Serve?. Disponível em: <https://www.citisystems.com.br/pwm/>