7 Temporizadores/ Contadores (Timers)

Versão 1.0

Nas aplicações com microcontroladores, é comum a necessidade de se marcar intervalos tempo, gerar sinais com certa periodicidade, medir a duração de eventos e controlar saídas com Modulação por Largura de Pulso (PWM – Pulse Width Modulation). É para atender à essas demandas que os microcontroladores trazem um recurso denominado Temporizador/Contador (TC - Timer/Counter), que é assunto deste capítulo. De agora em diante, para facilitar a leitura, as palavras contador e temporizador ou simplesmente timer serão usadas como sinônimos.

Para atender à necessidade de temporização, o MSP430 F5529, que é motivo deste estudo oferece os seguintes recursos:

- Timer A0, instância 0 do Timer A, com 5 comparadores;
- Timer A1, instância 1 do Timer A, com 3 comparadores;
- Timer A2, instância 2 do Timer A, com 3 comparadores e
- Timer B0, instância 0 do Timer A, com 7 comparadores.

Este capítulo não aborda o Relógio de Tempo Real (Real-time Clock) e nem o Temporizador Cão-de-Guarda (Watchdog Timer). Somente serão tratados os Timers A e B e suas instâncias.

7.0. Quero Usar os Timers e não Pretendo Ler Todo este Capítulo

A ser escrito.

7.1. Introdução aos Timers

Este tópico faz a construção passo a passo de um temporizador simplificado. A intenção é esclarecer e consolidar os conceitos mais importantes. Pode parecer estranho ao leitor os nomes usados neste tópico 7.1, porém a intenção foi a de buscar familiaridade com os nomes usados no MSP. Se o leitor já é familiarizado com o assunto, pode prosseguir a partir do próximo tópico.

Um temporizador (ou timer) nada mais é que um contador digital cuja frequência de contagem é controlada pelo programador. No caso do MSP430 este contador é de 16 bits, como exemplificado na Figura 7.1. Nesta figura temos um contador denominado de TxR, cuja frequência de contagem é especificada pelo sinal TxCLK. Como se pode ver na figura, a qualquer momento, é possível escrever ou ler neste contador TxR.

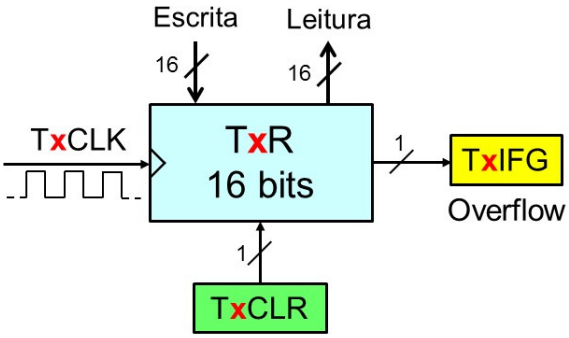

Figura 7.1. Esquema do contador de 16 bits, denominado TxR.

Por ter 16 bits, o contador pode ir de 0x0000 até o máximo que é 0xFFFF (65.535) e na próxima batida do relógio, ele volta a 0x0000 e recomeça a contagem. A transição do valor máximo (0xFFFF) para o valor mínimo (0x0000) é denominada de ultrapassagem ou overflow. Quando isto acontece, o contador avisa fazendo a flag TxIFG = 1. Existe ainda o recurso de zerar o contador TxR com o uso do bit TxCLR. É claro que nesta zeragem, não se ativa a flag de overflow.

Todo este comportamento está representado na Figura 7.2, onde as rampas inclinadas simbolizam a contagem de TxR. As duas linhas horizontais indicam os dois limites da contagem, que são 0x0000 e 0xFFFF. O instante em que o contador volta a zero está marcado com pequenos círculos. É neste instante que a flag TxIFG vai para 1, como indicado.

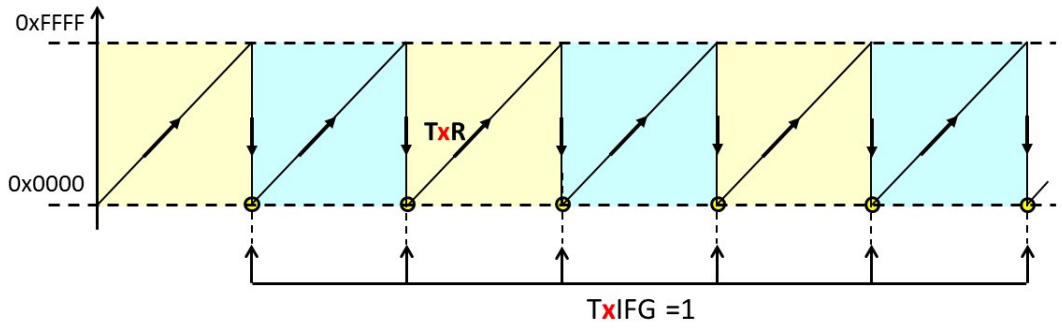

Figura 7.2. Gráfico indicando a contagem de TxR e os instantes em que esse contador volta a zero.

A título de exemplo, vamos considerar TxCLK = 1 MHz, o que significa uma contagem a cada 1 µs. Podemos então dizer que o temporizador ativa a flag TxIFG a cada 65.536 µs. Note que na contagem de 0 até 65.535 existem 65.536 passos, pois o zero é também contado. É importante comentar que o contador apenas ativa a flag TxIFG. É obrigação do programa zerar esta flag para que ele possa perceber uma nova transição de 0 para 1.

Com o que foi apresentado, já podemos fazer alguma temporização. Por exemplo, se precisamos ativar uma determinada saída ou disparar uma rotina após uma espera de 10.000 µs, basta fazer TxR = 55.536 e esperar TxIFG ir para 1. É claro que o hardware oferece condições para escrever no contador TxR e zerar a flag TxIFG. Se, mais adiante, precisarmos repetir essa espera de 10.000 µs, precisaremos reprogramar TxR. Podemos aperfeiçoar um pouco mais a proposta e permitir que a flag TxIFG provoque interrupção. Neste caso fica mais simples, pois preparamos o valor inicial de TxR e damos início à contagem. Depois do tempo especificado, somos interrompidos pela flag TxIFG.

Vamos agora introduzir um comparador, que é recurso muito simples, mas que traz bastante flexibilidade ao contador. A Figura 7.3 apresenta este recurso. Note que, para beneficiar a clareza da figura, os barramentos de escrita e leitura foram omitidos. O comparador é a caixa com o sinal de igualdade (“=”) e para que ele opere, se faz necessário um registrador para armazenar o valor da comparação, denominado TxCCRn. Cada vez que o comparador detecta que o valor do contador TxR ficou igual (coincidiu) com o valor do registrador TxCCRn ele avisa fazendo a flag TxCCIFGn ir para 1.

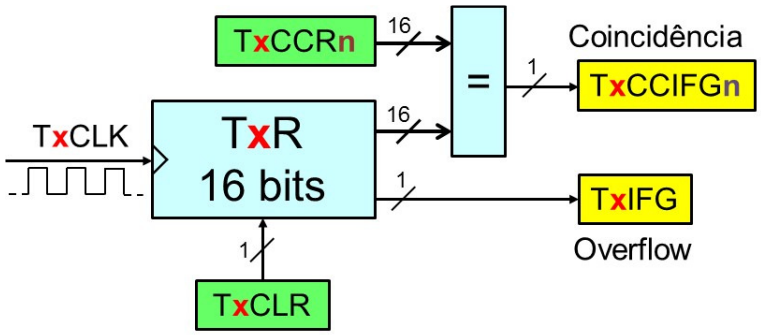

Figura 7.3. Esquema do contador de 16 bits, denominado TxR, com uma unidade de comparação.

A Figura 7.4 apresenta um gráfico que ilustra o contador e os instantes em que seu valor ficou igual ao do registrador de comparação TxCCRn. Essas coincidências estão numeradas e marcadas com um pequeno círculo. A cada coincidência, a flag TxCCIFG vai para 1. Nesta figura, iniciamos com "TxCCR = a", onde "a" é um valor hipotético qualquer (0x000 ≤ a ≤ 0xFFFF). No exemplo, logo após a coincidência de número 4, o programador alterou o valor desse registrador, fazendo "TxCCR = b", onde b > a. Por esse motivo, nesta rampa ascendente tivemos duas coincidências. Note que o circuito do comparador é bem simples: faz TxCCIFG = 1 toda vez que tivermos TxR = TxCCRn. Sempre que o programador alterar TxCCR para um valor acima de TxR, uma coincidência deve acontecer. Entretanto, se ele alterar TxCCR para um valor abaixo de TxR, a coincidência só vai acontecer no próximo ciclo, ou seja, após TxR chegar ao máximo e voltar a zero.

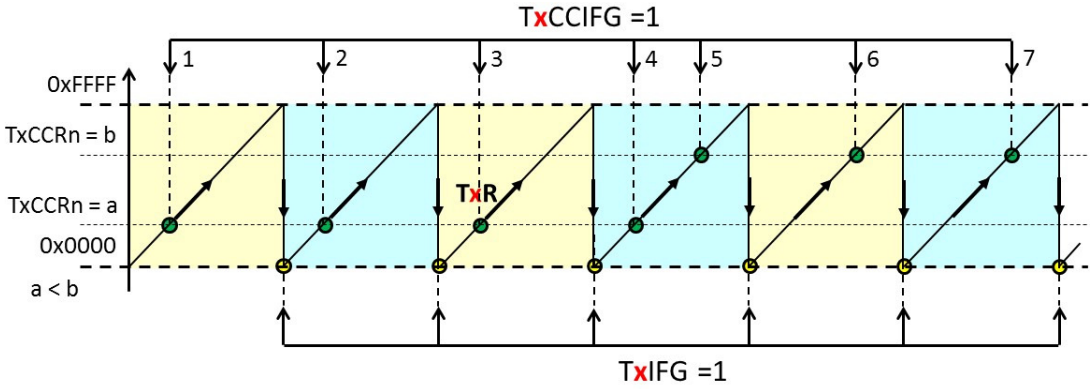

Figura 7.4. Gráfico indicando a contagem de TxR e os instantes em que seu valor coincidiu com o do registrador de comparação TxCCRn.

Vamos agora propor um primeiro pequeno problema. Considerando que o relógio é de 1 MHz, inicializamos: TxR = 20.000 e TxCCRn = 35.000. Uma vez iniciada a contagem, depois de quanto tempo as duas *flags* (TxIFG e TxCCIFGn) serão ativadas?

Resposta: TxCCIFGn = 1 após 15.000 µs e TxIFG = 1 após 45.536 µs.

Sugerimos até agora dois recursos para atender nossas necessidades de temporização. Um inconveniente com o que foi proposto é a necessidade de se recarregar o valor inicial do contador. Pensando em eliminar essa de recarga, podemos adicionar um recurso para limitar o valor máximo de contagem. A ideia é permitir que um comparador force a zeragem de TxR quando seu valor ficar igual ao valor do registrador correspondente. A Figura 7.5 traz duas unidades de comparação (retângulos com o sinal "=") e dois registradores de comparação TxCCRn e TxCCRm. Quando acontece a coincidência de TxR com o valor de TxCCRn, o contador TxR é zerado. Note que somente este comparador TxCCRn tem recursos para zerar o contador. Assim, o contador sempre parte com o valor zero e vai até o limite indicado pelo registrador TxCCRn.

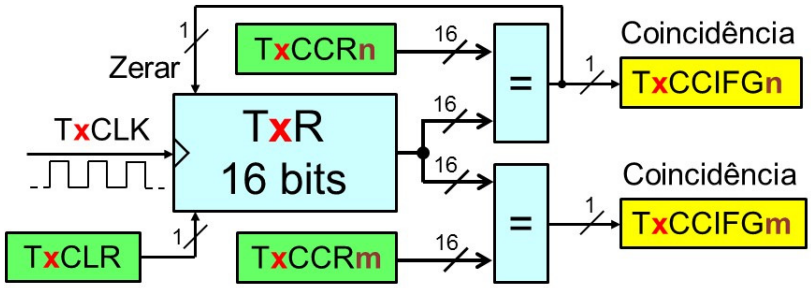

Figura 7.5. Esquema do contador de 16 bits, denominado TxR, com duas unidades de comparação.

Agora, alterando o valor de TxCCRn, podemos controlar a excursão do contador TxR, pois cada vez que TxR ficar igual a TxCCRn, ele volta a zero na próxima batida do relógio. A Figura 7.6 exibe este caso, que inicia com "TxCCRn = c" e depois da terceira coincidência, por ação do programador, teve seu valor alterado para "TxCCRn = b". Note que esses valores "c" e "b" controlam o período da contagem. As flags continuam operando da mesma maneira, como listado abaixo:

- TxIFG = 1: toda vez que TxR voltar a zero;
- TxCCIFGn = 1: toda vez que TxR = TxCCRn e
- TxCCIFGm = 1: toda vez que TxR = TxCCRm.

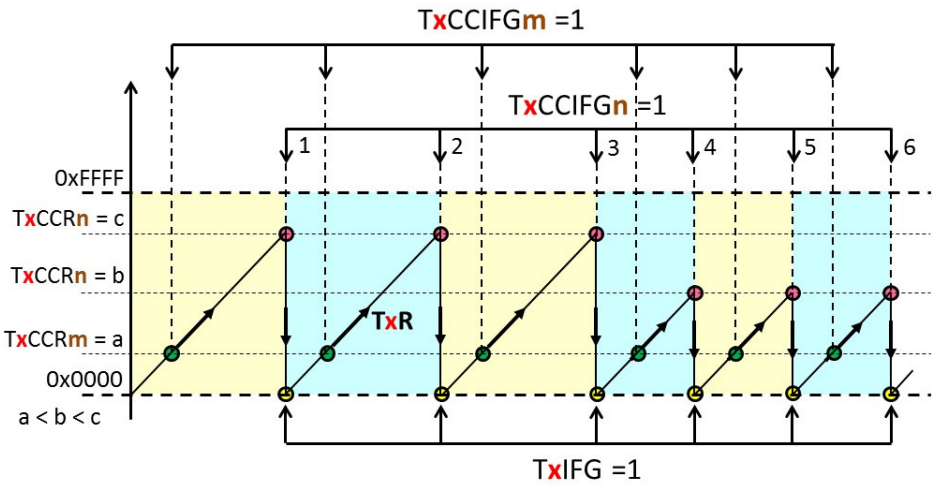

Figura 7.6. Gráfico indicando a contagem de TxR e os instantes de zeragem quando seu valor fica igual ao do registrador TxCCIFGn.

Vamos agora propor um segundo pequeno problema. Considerando que o relógio do contador TxR é de 1 MHz, e que TxCCRn = 4.999 e TxCCRm = 3.000, qual a frequência de ativação de cada uma das flags?

Resposta: o item mais importante é o valor de TxCCRn, já que ele limita a contagem. Assim, o contador TxR sai de 0 e vai até 4.999 e, depois, volta a zero. São então 5.000 contagens (lembre-se de que o zero foi contado), o que resulta em um período de 5.000 µs. Assim, a frequência de ativação da flag TxCCIFGn é de 200 Hz. Note que as demais flags (TxIFG e TxCCIFGm) também vão ter a mesma taxa de ativação que é de 200 Hz.

Até agora, com o limitado exemplo aqui apresentado, parece que há redundância entre as flags TxIFG, TxCCIFGn e TxCCIFGm. Mais adiante esta impressão será removida.

Vamos adicionar mais um recurso ao hardware proposto. Agora será uma Unidade de Saída, cuja finalidade principal é a de gerar uma saída PWM e que está mostrada na Figura 7.7. Note que a Unidade de Saída tem a informação de quando o contador TxR voltou a zero (TxR = TxCCRn) e de quando ele ficou igual ao valor de TxCCRm. O funcionamento desta unidade, cuja saída é OUTm, é bem simples.

- OUTm vai para 1 → sempre que TxR = 0 e
- OUTm vai para 0 → sempre que TxR = TxCCRm.

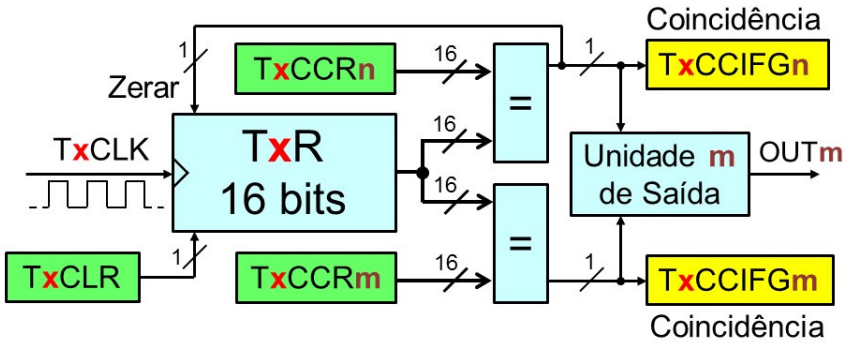

Figura 7.7. Esquema do contador de 16 bits, denominado TxR, com uma Unidade de Saída (OUTm).

Com essa Unidade de Saída, fica muito fácil a geração de PWM. A Figura 7.8 apresenta a a saída OUTm para dois valores de TxCCRm. Note que, à medida que se aumenta TxCCRm, aumenta-se o ciclo de carga. Resumindo: o valor de TxCCRn define o período do PWM e TxCCRm define seu ciclo de carga, de acordo com a expressão abaixo.

$$Ciclo\ de\ Carga\ (\%) = \frac{TxCCRm}{TxCCRn} \times 100\%$$

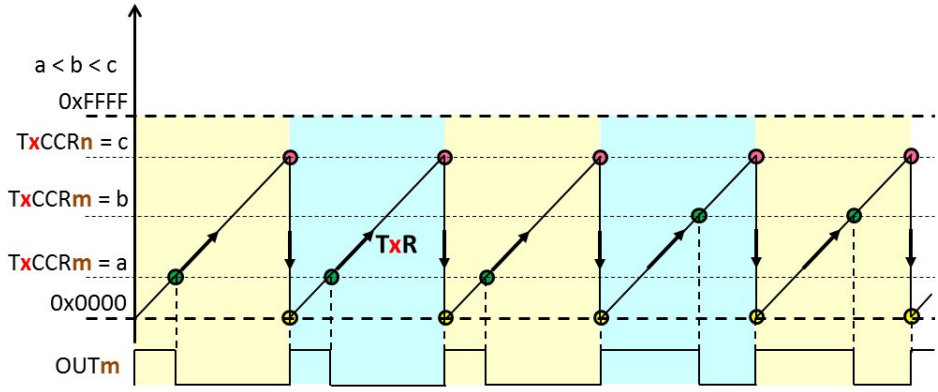

Figura 7.8. Gráfico indicando a geração de uma saída PWM com os registradores TxCCRn e TXCCRm.

Vamos agora propor um terceiro pequeno problema. Considerando que o relógio do contador TxR é de 1 MHz, indique os valores de TxCCRn e TxCCRm para gerar uma saída PWM com período de 10 ms e ciclos de carga de 30% e 70%.

Resposta: o período é controlado pelo valor de TxCCRn e o ciclo de carga pelo valor de TxCCRm. As contas são simples:

- Para o período de 10 ms, precisamos de 10.000 contagens (1.000.000 x 0,01) e, com isso, temos TxCCRn = 9.999, pois o zero é também contado.
- Para um ciclo de carga de 30%, temos TxCCRm = 0,3 x 9.999 = 2.999,7, que aproximamos para 3.000 contagens.

- Para um ciclo de carga de 70%, temos TxCCRm = 0,7 x 9.999 = 6.999,3, que aproximamos para 7.000 contagens.

Finalmente, outro recurso interessante a ser adicionado ao temporizador proposto é a possibilidade de captura de eventos. Por capturar um evento, simplesmente se quer dizer marcar o instante de tempo em que aconteceu uma alteração num determinado pino, que pode ser um flanco de subida (↑) ou um flanco de descida (↓), ou ainda, qualquer um deles. A Figura 7.9 apresenta um diagrama com esse novo recurso. Na parte inferior da figura, está a Lógica de Captura, onde o usuário indica qual flanco deseja capturar. Quando acontece o evento selecionado, a Lógica de Captura copia o valor do contador TxR no registrador TxCCRm e, para marcar esse fato, faz a flag TxCCIFGm = 1.

É de se notar que o registrador TxCCRm, que nas figuras anteriores era usado pelo comparador, passou a ser dedicado à captura de eventos. Não está mostrado nesta proposta, mas é claro que o programador deve poder indicar se quer operar no modo captura ou no modo comparação.

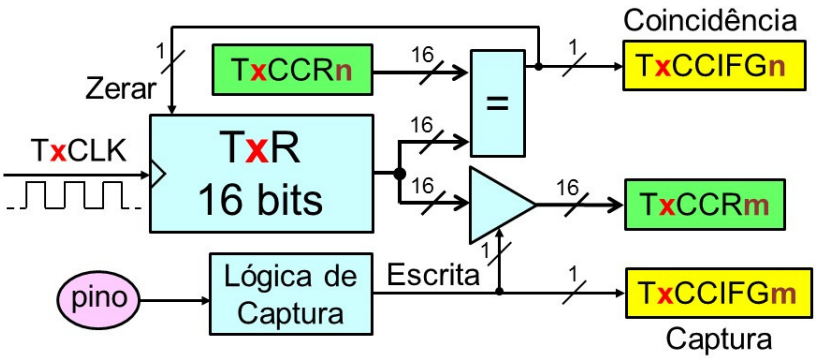

Figura 7.9. Esquema do contador de 16 bits, com uma lógica de captura.

Para explicar ainda mais, vamos supor que se deseja medir quanto tempo um determinado sinal digital fica em nível alto. Para resolver esse problema, ligamos este sinal ao "pino" indicado na figura, selecionamos o modo de captura e preparamos a Lógica de Captura para o evento flanco de subida. Quando ele acontecer, o valor de TxR é copiado para o registrador TxCCRm. O programa percebe que aconteceu uma captura porque a *flag* TxCCIFGm vai para 1. Então, o valor de TxCCRm é copiado para uma variável qualquer, a *flag* é zerada e a Lógica de Captura é agora programada para o evento flanco de descida. Quando acontecer esse segundo evento, a diferença entre o valor atual TxCCRm e o valor que foi guardado anteriormente vai indicar (em contagens de TxR) quanto tempo o sinal ficou em nível alto.

Uma dúvida comum sobre a captura é: não posso escrever um programa para fazer tudo isso? Sim, mas um programa vai introduzir erro, pois vai consumir tempo para detectar

alteração no pino e copiar o valor de TxR para uma variável. Será que não poderia monitorar a alteração do pino com o uso de uma interrupção? Esta solução é pior ainda, porque as interrupções possuem uma latência grande. Além disso, se o evento de captura acontecer enquanto se atende a uma outra interrupção, o atraso será maior ainda. Por outro lado, o hardware que foi esquematizado é capaz de copiar o valor de TxR para o registrador TxCCRm com a precisão de um período de relógio do contador.

Temos agora uma boa ideia do que esperar de um Temporizador/Contador. Ele deve oferecer ao programador recursos para:

- Ativar *flags* com a periodicidade desejada;
- Provocar interrupções periódicas;
- Gerar atrasos de tempo;
- Controlar saída PWM e
- Medir intervalo entre dois eventos.

Passemos então ao estudo da arquitetura dos Temporizadores/Contadores disponíveis na família MSP430F5529.

7.1. Introdução aos Timers do MSP430F559

Iniciamos explicando a forma como o fabricante identificou esses recursos. A arquitetura MSP430 oferece três diferentes tipos de timers:

- Timer A;
- Timer B e
- Timer D.

Cada versão da família MSP430 pode ter diferentes quantidades de um determinado tipo de timer, que são chamadas de instâncias. No nosso caso, o MSP430F5529 tem 3 instâncias do timer A, denominadas de TA0, TA1 e TA2, apenas uma instância do timer B, denominada de TB0 e nenhuma instância do Timer D. Cada instância possui um contador e vários registradores de comparação e captura, formando a sigla “CC”, que aparece no nome dos registradores.

Tabela 7.1. Instâncias dos timers disponibilizados pelo MSP430F5529

-	**Timer A0**	**Timer A1**	**Timer A2**	**Timer B0**
Contador	TA0R	TA1R	TA0R	TB0R
Registrador Compara/Captura	TA0CCR0	TA1CCR0	TA2CCR0	TB0CCR0
Registrador Compara/Captura	TA0CCR1	TA1CCR1	TA2CCR1	TB0CCR1
Registrador Compara/Captura	TA0CCR2	TA1CCR2	TA2CCR2	TB0CCR2
Registrador Compara/Captura	TA0CCR3	-	-	TB0CCR3
Registrador Compara/Captura	TA0CCR4	-	-	TB0CCR4
Registrador Compara/Captura	-	-	-	TB0CCR5
Registrador Compara/Captura	-	-	-	TB0CCR6

Para tornar a ideia bem clara, apresentamos a Figura 7.10 onde se vê a instância zero do timer A (TA0), composto por seu contador (TA0R) e pelos 5 registradores de comparação e captura (TA0CCR0, TA0CCR1, ..., TA0CCR4). O leitor deve começar a prestar atenção aos nomes dos registradores e ir se acostumando com sua lógica de formação.

Existe um registrador para o controle do contador e um registrador para o controle de cada unidade de comparação. Por exemplo, o controle do contador TA0R é feito pelo registrador TA0CTL. O controle de cada uma das unidades de comparação de TA0 é feito pelos registradores TA0CCTL0, TA0CCTL1, ..., TA0CCTL4.

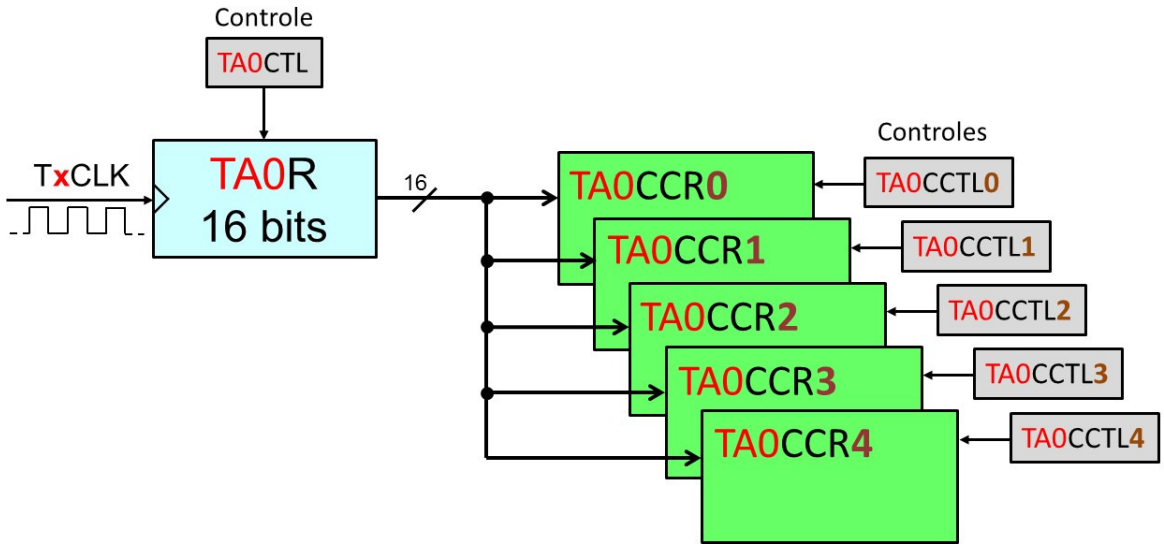

Figura 7.10. Instância zero do timer A (TA0R), com seus 5 registradores de comparação e captura (TA0CCR0, TA0CCR1, ..., TA0CCR5) e os registradores de controle.

Faremos a seguir o estudo do Timer A e de suas instâncias. Depois estudaremos o Timer B vendo suas diferenças e particularidades em relação ao Timer A.

7.2. Estudo do Timer A

Como já vimos, o MSP430F5529 oferece 3 instâncias do Timer A, cada uma com certa quantidade de comparadores (vide Tabela 7.1). Para facilitar a explicação e as referências adotaremos as seguintes convenções para denominar os registradores pertinentes.

- A letra “x” indica uma instância do timer A (x = 0, 1, ...) e

- A letra “n” indica uma instância da unidade captura e comparação (n = 0, 1, ...).

Como já foi afirmado e está ilustrado na Figura 7.10, existem registradores para controlar a operação do contador e o funcionamento das unidades de captura e comparação. A tabela abaixo resume esses registradores. Cada instância de um timer possui um registrador de controle denominado TAxCTL e os diversos registradores de controle das unidades de captura e compara deste timer são denominados TAxCCTLn.

Tabela 7.2. Resumo dos registradores das instâncias do Timer A e de seus registradores de Captura e Compara

	Registrador	Controle	Flag IFG
Contador	TAxR	TAxCTL	TAIFG
Captura/Compara	TAxCCRn	TAxCCTLn	CCIFG

A lógica usada na nomenclatura dos registradores é importante e sua compreensão facilita muita o uso dos timers. Assim, apresentamos insistimos no tema e apresentamos um resumo:

- TAxR → contador da instância x do timer A;
- TAxCTL → registrador para controle do contador da instância x do timer A;
- TAxCCRn → registrador da instância n do módulo de captura/compara de TAx e
- TAxCCTLn → registrador de controle da instância n do módulo de captura/compara de TAx.

Cada registrador de controle tem uma flag muito importante para indicar o progresso de seu contador. Como muitas dessas flags têm o mesmo nome, a distinção entre elas é feita pelo registrador de controle onde ela se encontra. Na listra abaixo usamos a notação “registrador.flag”.

- TAxCTL.TAIFG → indica que o contador TAxR voltou a zero e
- TAxCCTLn.CCIFG → indica que o valor de TAxR coincidiu com o de TAxCCRn.

7.2.1. Contagem com o Timer A

O uso mais simples de um timer é para a contagem. O Timer A possui 4 modos de contagem:
Modo 0 → Parado (stop);
Modo 1 → Ascendente (up);
Modo 2 → Contínuo (continuous) e
Modo 3 → Ascendente/Descendente (up/down).

7.2.1.1. Modo 0 – Parado (Stop)

Quando no Modo 0 (Parado ou Stop), o contador fica parado. Então este modo serve apenas para parar a contagem de TAxR.

7.2.1.2. Modo 1 – Ascendente (Up)

Quando no Modo 1 - Ascendente (Up), o contador TAxR conta até seu valor ficar igual ao do registrador TAxCCR0, quando então ele recomeça a contagem a partir de zero, como mostrado na Figura 7.11. Este modo permite controlar a excursão do contador. O período será igual a TAxCCR0+1 contagens, porque o zero é também contado. Se este modo é selecionado com TAxR maior que TAxCCR0, o contador é zerado imediatamente. Note que somente o comparador 0 (TAxCCR0) pode ser usado para zerar o TAxR.

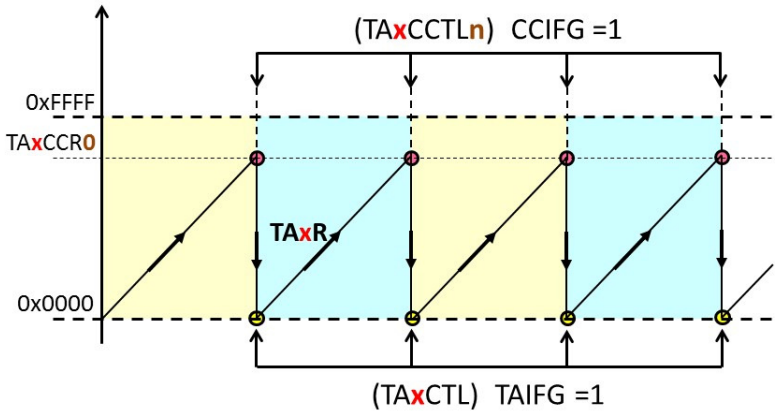

Figura 7.11. Modo 1 - Ascendente (Up), o contador TAxR é zerado quando seu valor fica igual ao do registrador TAxCCR0.

A Figura 7.11 ainda mostra o instante de ativação das flags CCIFG e TAIFG. Quando o contador TAxR fica igual ao registrador de comparação TAxCCR0 a flag CCIFG é ativada (além de zerar TAxR) e quando TAxR fica igual a zero, a flag TAIFG é ativada. Essas flags estão nos registradores de controle, como indicado na Figura 7.11 (note o nome entre os parêntesis). Para melhor visualização dessas flags, consulte as figuras 7.24 e 7.26.b. O resumo abaixo repete o conceito, usando a notação “registrador.flag”:

- TAxCTL.**TAIFG** → indica que TAxR voltou a zero e
- TAxCCTL0.**CCIFG** → indica que o valor de TAxR coincidiu com o de TAxCCR0.

O instante exato em que as flags são ativadas é importante e está ilustrado na Figura 7.12. Note que na batida do relógio que faz TAxR igual a TAxCCR0, a flag CCIFG vai para 1. Já na batida que leva TAxR para zero, a flag TAIFG vai para 1.

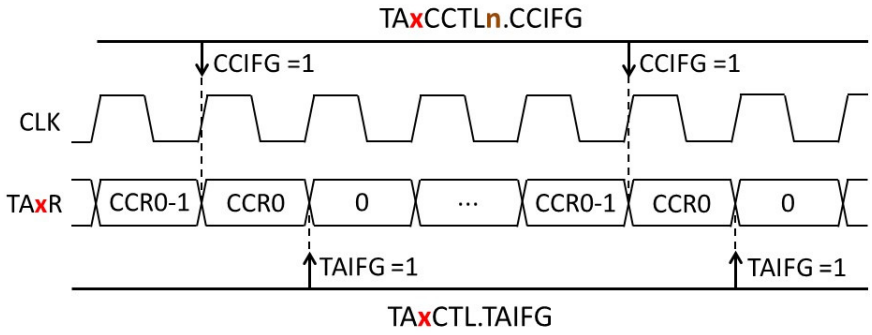

Figura 7.12. Modo 1 (Up ou Ascendente), mostrando o instante em que cada flag é ativada.

Para tornar o conceito ainda mais claro, a Figura 7.13 apresenta o exemplo de se empregar o Modo 1 com TAxCCR0 = 999. Serão 1.000 contagens. Cada vez que o contador TAxR ficar igual a TAxCCR0 (que é 999), a flag CCIFG vai para 1. Na próxima batida do relógio o contador TAxR volta a zero e a flag TAIFG vai para a 1.

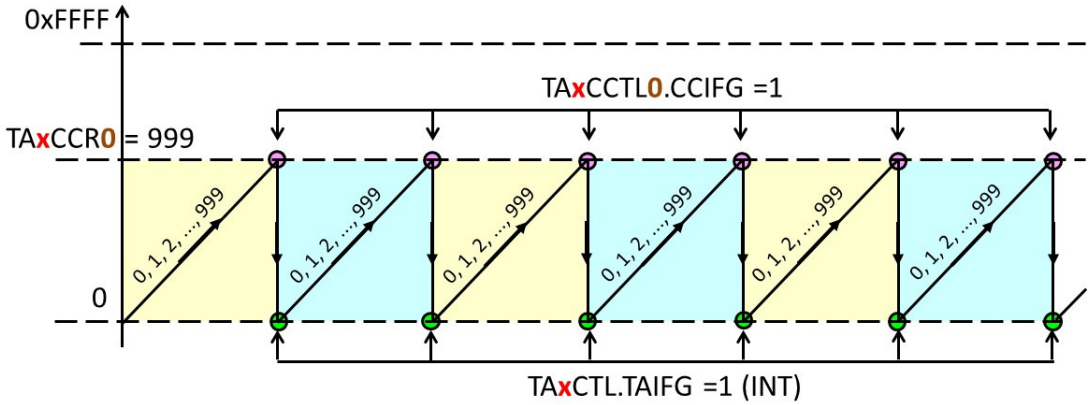

Figura 7.13. Modo 1 - Ascendente (Up), usando exemplo de TAxCCR0 = 999 e mostrando o instante em que cada flag é ativada.

O valor de TAxR pode ser lido ou escrito a qualquer momento. Se o novo valor de TAxR for maior que o valor de TAxCCR0, ele é zerado automaticamente, ou seja, a contagem segue a partir do zero (verificar). Se o valor de TAxCCR0 for alterado para um valor idêntico ou superior ao atual, tudo se passa como o esperado. Porém, se TAxCCR0 for alterado para um valor abaixo do valor atual de TAxR, este registrador TAxR é zerado. Entretanto, o manual alerta que uma contagem extra pode ocorrer.

7.2.1.3. Modo 2 – Contínuo (Continuous)

Quando no Modo 2 – Contínuo (Continuous), o contador TAxR conta até seu limite de 16 bits, que é o valor 0xFFFF. Quando chega a este limite, o contador volta a zero e a flag TAIFG vai para 1. Note que nenhum registrador é usado para limitar a contagem. A Figura 7.11 ilustra este modo de operação, que é semelhante ao modo anterior.

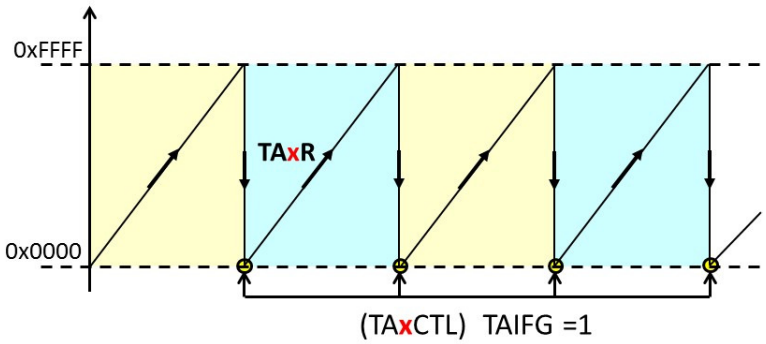

Figura 7.14. Modo 2 - Contínuo (Continuous), o contador conta até seu limite de 16 bits (0xFFFF) e em seguida volta a zero. Cada vez que volta a zero, a flag TAIFG vai para 1.

7.2.1.4. Modo 3 – Sobe/Desce (Up/Down)

Quando no Modo 3 – Sobe/Desce (Up/Down), o contador TAxR inicia contando de forma ascendente e conta até ficar igual ao valor deTAxCCR0, quando então passa a contar de forma descendente até voltar a zero e reiniciar o ciclo. O contador TAxR fica um período com valor igual ao de TAxCCR0 e um período em zero. A Figura 7.15 ilustra este modo de operação. É importante notar o instante de ativação das flags TAIFG e TAxCCIFG0. A direção da contagem é armazenada em um latch, isso permite que o contador continue no mesmo sentido, caso ele seja parado momentaneamente. Caso o contador seja zerado usando o TACLR, a contagem inicia a partir do zero e em modo ascendente.

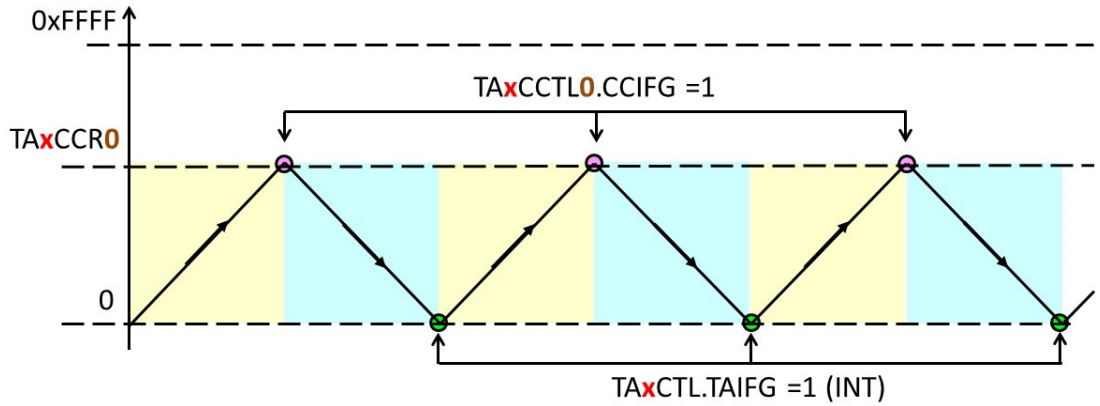

Figura 7.15. Modo 3 - Sobre/Desce (Up/Down), o contador conta até seu limite de 16 bits (0xFFFF) e em seguida volta a zero. Cada vez que volta a zero, a flag TAIFG vai para 1.

O valor de TAxR pode ser lido ou escrito a qualquer momento. Se o novo valor de TAxR for maior que o valor de TAxCCR0, este registrador TAxR é zerado, ou seja, a contagem segue a partir do zero (verificar). Se o valor de TAxCCR0 for alterado durante a fase descendente, o contador TAxR segue normalmente até chegar ao zero e somente na fase ascendente é que o novo valor de TAxCCR0 será efetivo. Se TAxCCR0 for alterado durante a fase ascendente para um valor igual o superior ao atual, a contagem segue da forma esperada. Se TAxCCR0 for alterado durante a fase ascendente para um valor inferior ao valor atual de TAxR, a contagem recomeça de forma ascendente a partir do zero.

7.2.1.5. Diagrama de Blocos do Contador

A Figura 7.16 apresenta um diagrama de blocos do contador. De acordo com a convenção adotada, todos os retângulos de cantos arredondados indicam bits configurados (controlados) pelo programador. Os retângulos preenchidos com amarelo vivo indicam flags que o programador pode consultar ou usar para provocar interrupção. Essas flags têm os nomes de acordo com o padrão "xxxIFG". Os pinos do processador são indicados por nomes dentro de formas circulares.

Nesta figura é muito importante a forma de se selecionar o relógio (CLK) do contador TAxR. Como se vê no extremo esquerdo, graças a um multiplexador, o programador pode escolher uma dentre 4 fontes distintas. É possível usar um dos relógios internos, ACLK ou SMCLK, ou o usuário pode fornecer um relógio (TAxCLK) por um pino específico do processador. A entrada de número 3 (INCLK) não está disponível na versão F5529. Depois de selecionada a entrada desejada, ela pode ser submetida a dois divisores cascateados até se transformar em CLK, que é o relógio do contador TAxR.

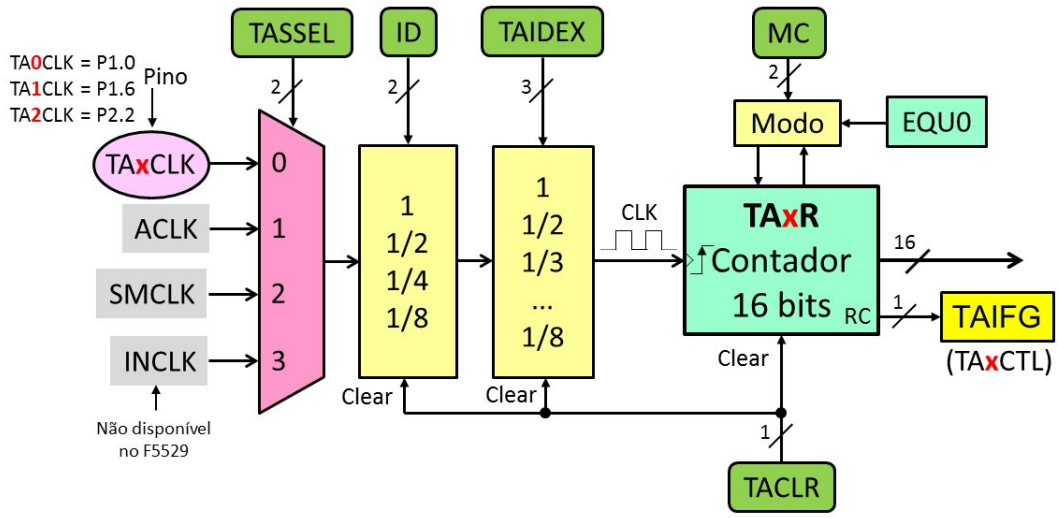

Figura 7.16. Diagrama de blocos para os modos de contagem do Timer A (TAxR). Nota-se de importante a seleção do relógio (CLK) para a contagem.

De acordo com o modo selecionado (MC), o contador TAxR conta na frequência do sinal CLK selecionado. Cada vez que esse contador volta a zero, a flag TAIFG é ativada (pela saída RC, que em inglês significa Ripple Carry Out). Essa flag TAIFG não tem nome próprio, ela é individualizada pelo registrador de controle ao qual pertence (TAxCTL). A caixa denominada "EQU0", indica o instante em que o contador TAxR ficou igual ao registrador TAxCCR0, pois como já foi visto, isto é importante para os modos 1 e 3. Para finalizar, o bit "TACLR" faz a zeragem do contador TAxR e dos estados internos dos dois divisores. Um ponto interessante é que o usuário escreve 1 neste bit e, após a zeragem do contador, ele volta automaticamente para 0.

A Figura 7.17 apresenta apenas a parte ligada à comparação. O comparador é a caixa marcada com o sinal de igualdade (=). Ele, continuamente, compara o valor do contador TAxR com o valor que o programador armazenou no registrador de comparação TAxCCRn. A flag CCIFG vai a 1 cada vez que esses dois registradores têm o mesmo valor, fato designado como "coincidência". Note que essa flag CCIFG não tem nome próprio, ela é individualizada pelo registrador de controle do comparador ao qual pertence (TAxCCTLn). A Tabela 7.1, já apresentada, lista a quantidade de comparadores para cada instância do timer A.

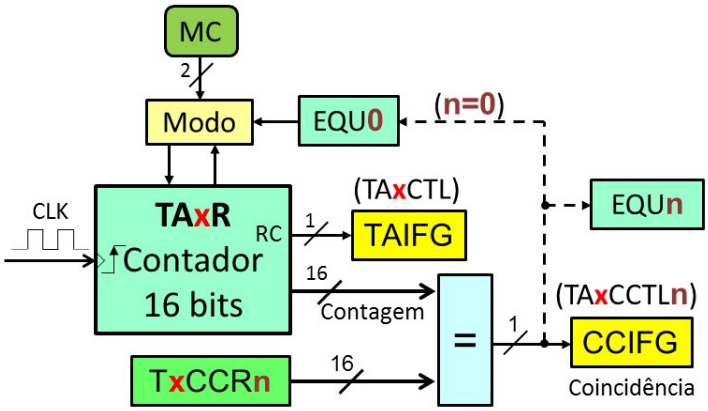

Figura 7.17. Diagrama de blocos para os modos de contagem, apresentando o registrador de comparação

Quando ocorre a coincidência entre o valor do contador TAxR e do comparador TxCCRn, um sinal interno EQUn é ativado. Esse sinal será importante na geração de saídas, como será visto mais adiante. Por enquanto, o que nos interessa é o EQU0, pois ele é o responsável por zerar o TAxR quando este opera nos modos 1 ou 3.

Exemplo 7.1: Usando TA0 e ACLK = 32.768 Hz, construa um relógio de 24 horas com precisão de décimos de segundos.

Solução: Vamos usar o Modo 1 (Ascendente), onde o limite de contagem é indicado pelo valor de TA0CCR0. O ACLK faz 32.768 contagens em 1 segundo, portanto, em 0,1 s (um décimo) são 3.276,8 contagens. Como precisamos trabalhar com inteiros, arredondamos para 3.277 contagens e comentemos um pequeno erro (0,610 µs). O relógio vai atrasar um pouco. Lembrando que nessa contagem, o zero é também contado, subtraímos 1 desse valor, que passa a ser 3.276. Serão usados alguns registradores, que estão listados mais adiante.

```
# define TRUE 1
void relógio (void){
unsigned char hora=0, min=0, seg=0, dseg=0;
TA0CTL = TASSEL_1 | MC_1 | TACLR;   //Selecionar ACLK e Modo 1
TA0CCR0 = 3277;                     //Limite da contagem
while(TRUE){
      while ( (TA0CTL0&TAIFG) ==  0);     //Esperar TAIFG=1;
      TA0CTL &= ~TAIFG;                   //Fazer TAIFG=0
      if (++deseg == 10){                 //Incr. decimos de segundos
            dseg=0;
            if (++seg == 60){             //Incr. segundos
                  seg=0;
                  if (++min == 60){       //Incr. minutos
                        min=0;
                        if(++hora == 24)  //Incr. horas
                              hora=0;
                  }
            }
      }
}
```

7.2.1.6. Geração de Saída com Timers

Como já vimos, são várias instâncias dos timers, sendo que cada instância tem vários comparadores. Cada comparador possui uma Unidade Geradora de Saída. Essa unidade, usando os sinais (EQUn) fornecidos pelos comparadores, pode gerar diversos tipos de saída, para cada modo de contagem.

A Figura 7.18 apresenta a um diagrama de blocos para o estudo da Unidade Geradora de Saída do comparador n, que na figura foi rotulada com “Lógica OUTn”. Essa Lógica OUTn usa as informações (EQU0 e EQUn) fornecidas pelas Unidades de Comparação. De acordo com o modo de saída programado (OUTMOD) um determinado sinal é gerado no

pino do processador. É claro que o modo de operação do contador TAxR vai influenciar diretamente na forma do sinal gerado.

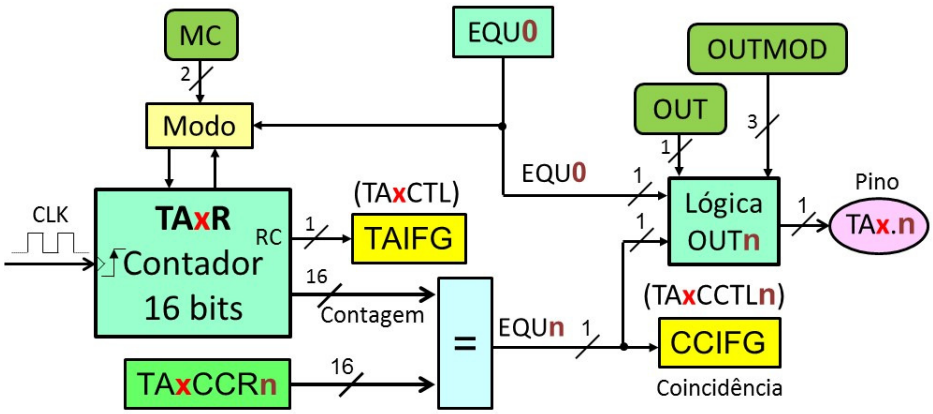

Figura 7.18. Diagrama de blocos para a Unidade Geradora de Saídas (Lógica OUTn) do comparador n.

A Tabela 7.3 relaciona o comportamento da saída em função do Modo de Operação da Unidade de Saída (OUTMOD) e das coincidências (EQU0 e EQUn) sinalizadas pelas unidades de comparação. A palavra inglesa “Toggle”, usada nesta tabela, significa inversão do estado da saída. O Modo de Saída 0, permite que a saída seja controlada diretamente pelo bit OUT do registrador de controle (TAxCCRn). Em outras palavras, neste modo (OUTMOD = 0), a saída acompanha o bit OUT, que é controlado pelo programador. Os modos 2, 3, 6 e 7 usam sinais de duas unidades de comparação sendo que, forçosamente, a unidade 0 é uma delas. Assim, esses modos não podem ser utilizados com a unidade de saída do comparador 0. As Figuras 7.19, 7.20 e 7.21 ilustram esses diversos modos de operação da Unidade de Saída

Tabela 7.3. Tabela apresentando os 8 modos de operação da Unidade Geradora de Saída.

OUTMOD	Modo	EQUn	EQU0
0	Output	Output	Output
1	Set	1	-
2	Toggle/Reset	T	0
3	Set/Reset	1	0
4	Toggle	T	-
5	Reset	0	-
6	Toggle/Set	T	1
7	Reset/Set	0	1

T = Inverte estado da saída

A Figura 7.19 apresenta os Modos de Saída do Comparador n quando o contador está operando no Modo Ascendente (Up, MC = 1). Neste Modo Ascendente, o registrador TAxCCR0 é usado para limitar a excursão da contagem, ou seja, ele especifica o período das formas de onda geradas na saída. A saída n (OUTn) é controlada pelo Comparador n. Note que alguma coisa acontece com a saída quando o contador TAxR fica igual ao valor de TAxCCRn. Este Modo Ascendente é o mais indicado para gerar PWM de alta frequência, como é o caso de regulação de potência, de retificação e de construção de conversores Digital/Analógico (DAC).

Como se pode ver na figura, os modos 2, 3, 6 e 7 são os mais indicados para a geração de PWM. Neste caso, temos:

- TAxCCR0 → especifica o período do PWM e
- TAxCCRn → especifica o ciclo de carga do PWM.

Não está indicado na Figura 7.19, mas as flags CCIFG continuam sendo ativadas cada vez que o valor do contador TAxR coincide com o valor de algum registrador de comparação (TAxCCRn).

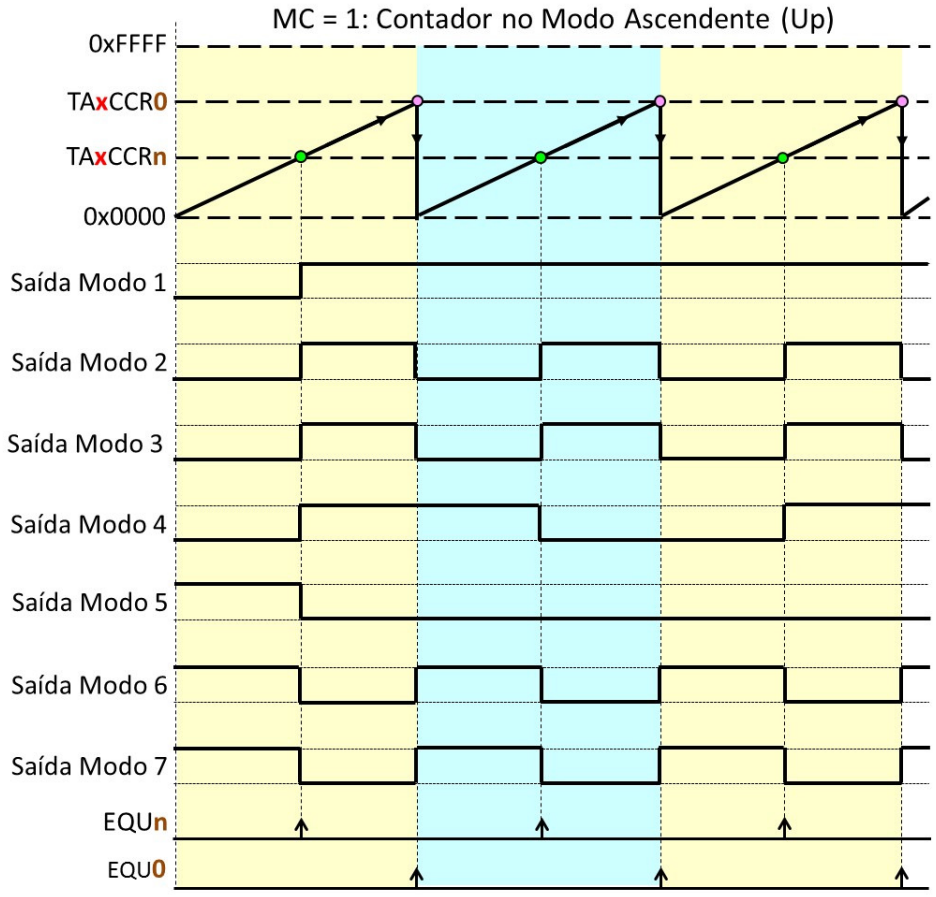

Figura 7.19. Opções de Modos de Saída para quando o contador opera no Modo Ascendente.

A Figura 7.20 apresenta os Modos de Saída do Comparador n quando o contador está operando no Modo Contínuo (Continuous, MC = 2). Neste modo, o limite de contagem é fixo e igual a 0xFFFF e, portanto, o mesmo acontece com o período dos sinais gerados na saída. A geração da saída n (OUTn) envolve o registrador de comparação TAxCCRn e, forçosamente, o registrador TAxCCR0. Note que alguma coisa acontece com a saída quando o contador TAxR fica igual ao valor de um dos registradores TAxCCR0 e TAxCCRn. Este modo de contagem permite que selecione o instante exato dos flancos de subida e de descida da saída gerada, ou seja, permite que se posicione a carga do PWM em qualquer lugar dentro do seu período. Este modo já é mais indicado para o controle de motores. Quais outras aplicações?

Não está indicado na Figura 7.20, mas as flags CCIFG continuam sendo ativadas cada vez que o valor do contador TAxR coincide com o valor de algum registrador de comparação (TAxCCRn).

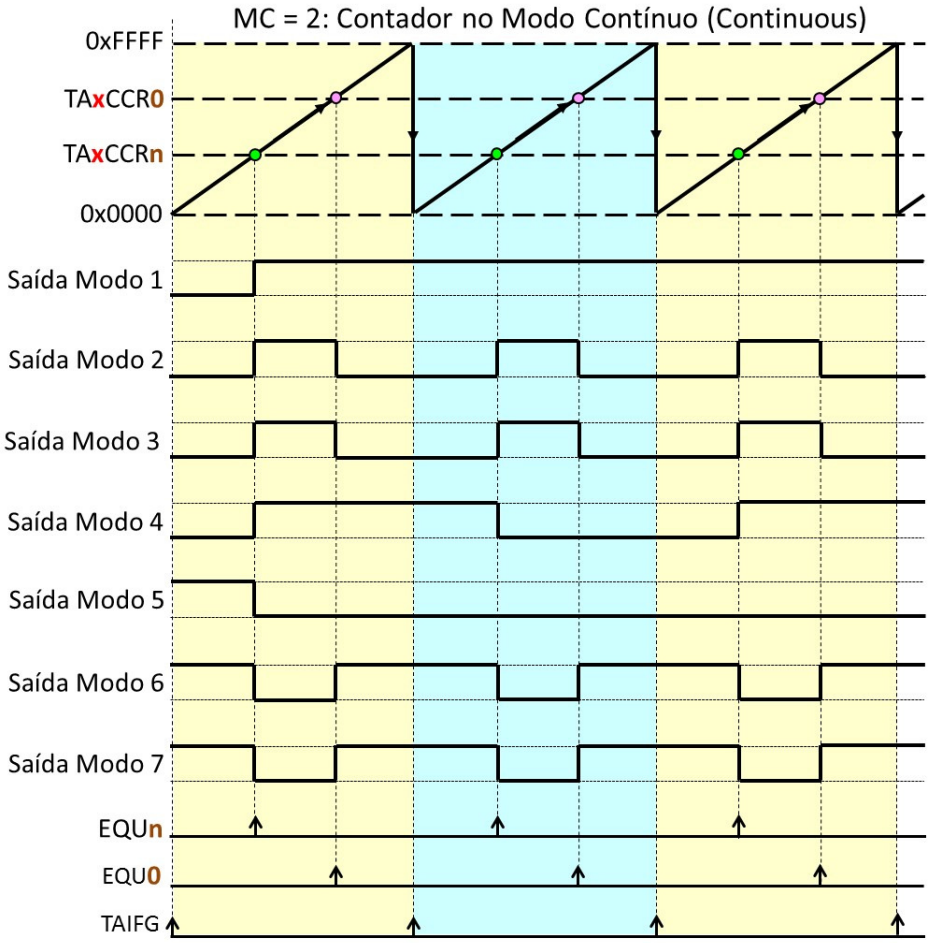

Figura 7.20. Opções de Modos de Saída para quando o contador opera no Modo Contínuo.

A Figura 7.21 apresenta os Modos de Saída do Comparador n quando o contador está operando no Modo Ascendente/Descendente (Up/Down, MC = 3). Neste modo, o limite de contagem é especificado pelo valor de TAxCCR0 e o contador TAxR vai de forma ascendente desde o zero até valor de TAxCCR0, quando então ele passa a contar de forma descendente até chegar a zero e recomeçar tudo de novo.

Como se pode ver na figura, os modos 2, 3, 4, 6 e 7 são os mais indicados para a geração de PWM. Neste caso, temos:

- TAxCCR0 → especifica (metade de) o período do PWM e
- TAxCCRn → especifica o ciclo de carga do PWM.

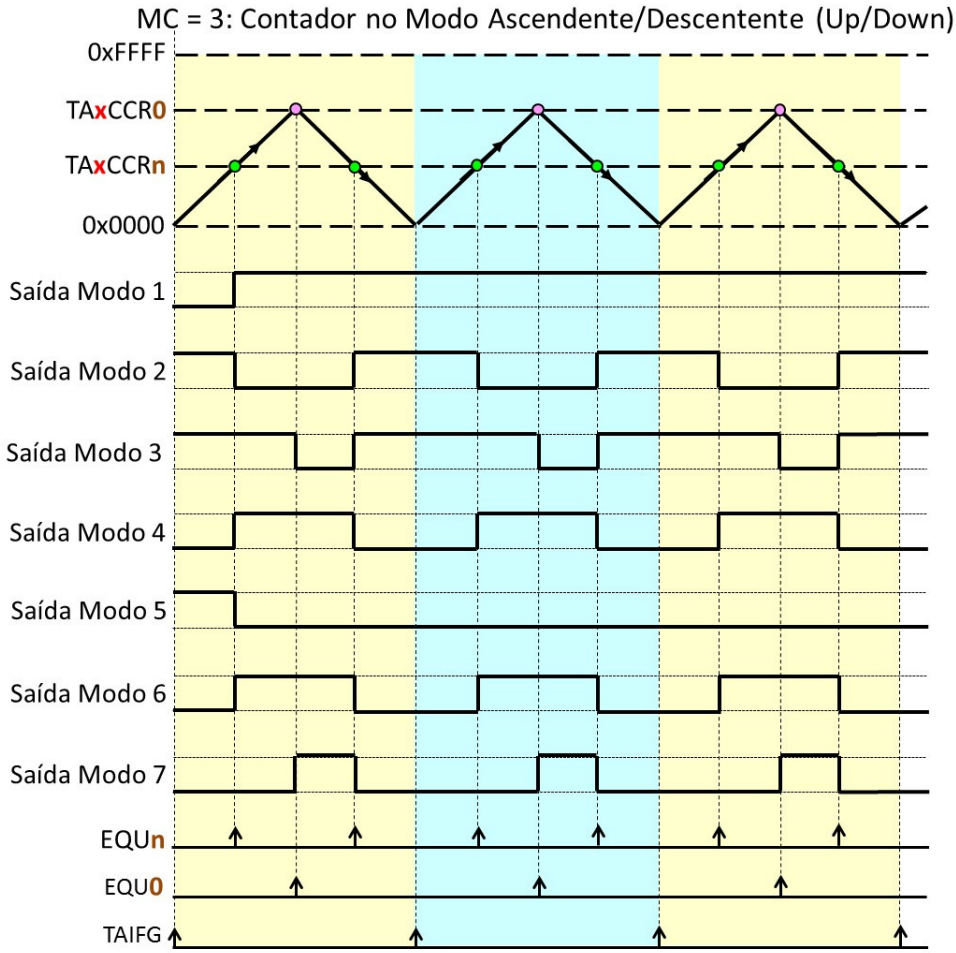

Figura 7.21. Opções de Modos de Saída para quando o contador opera no Modo Ascendente/Descendente (MC=3).

Este modo é o mais recomendado para a geração de PWM para o controle de motores, especialmente quando se precisa alterar o ciclo de carga com grande frequência. Note que nos modos 2, 4 e 6 o centro do pulso gerado coincide com o pico da contagem. Isto garante que os pulsos sempre tenham o mesmo espaçamento temporal, independente do ciclo de carga. Em outros modos de geração de saída, a alteração da carga do PWM pode alterar o intervalo entre dois pulsos consecutivos.

7.2.1.7. Captura de Eventos com Timers

Como já vimos no início deste capítulo, a captura nada mais é que a cópia do contador TAxR num registrador de comparação TxCCRn quando o evento especificado acontece. O evento pode ser, por exemplo, um determinado pino indo para nível alto. Isto permite que se meça o intervalo de tempo entre dois eventos. Serve para medir largura de pulsos, periodicidade de um determinado sinal, etc.

A Figura 7.22 apresenta o diagrama de blocos da unidade de captura “n”. O multiplexador mais à esquerda, denominado de MUX, controlado pelos bits CCIS seleciona a entrada que será usada para gerar o evento de captura. São 4 entradas possíveis. Entretanto, nesta versão F5529, somente a entrada CCInA está conectada a um pino do processador. A Tabela 7.5 apresenta uma lista dos pinos que nesta versão do MSP podem disparar a captura. A entrada CCInB não está disponível e as outras duas entradas estão conectadas à terra, GND, à VCC. O leitor deve estar estranhando o porquê de ligar entradas à GND e VCC. Isto será explicado no próximo parágrafo.

A saída do MUX1 é entregue a um circuito, denominado de Modo de Captura, que verifica se o flanco especificado aconteceu. São 4 modos de captura, controlados pelos bits CM, como especificados na Tabela 7.4. Em seu programa, ao alternar entre as entradas 2 e 3 do MUX1, ele pode provocar um flanco (de subida ou descida) e assim disparar uma captura por software.

Continuando com a Figura 7.22, a entrada 0 do MUX2 permite que o programador use diretamente o sinal gerado pela Unidade de Modo de Captura e a entrada 1 do MUX2, faz a sincronização deste sinal com o relógio do contador TAxR. Como a detecção dos flancos é assíncrona, há o risco de metaestabilidade. Por isso, o manual recomenda, sempre que possível, usar a sincronização (entrada 1 do MUX2).

A saída do MUX1 provoca a escrita do valor atual do contador TAxR no registrador de comparação TAxCCRn. Esta mesma saída, através da entrada 1 do MUX3 ativa a flag CCIFG. O MUX3, como se nota na figura é controlado pelo bit CAP. Quando CAP = 1, a ativação de CCIFG é feita pela captura e, quando CAP = 0, a ativação de CCIFG é feita pela Unidade de Comparação. Assim, quando no Modo Captura (CAP = 1), a flag CCIFG alerta ao programador que aconteceu uma captura e que ele pode ler o resultado no registrador de comparação TAxCCRn. Se estiver operando por polling, o programador é

responsável por zerar esta flag, para que ele a perceba novamente indo para 1 na próxima captura.

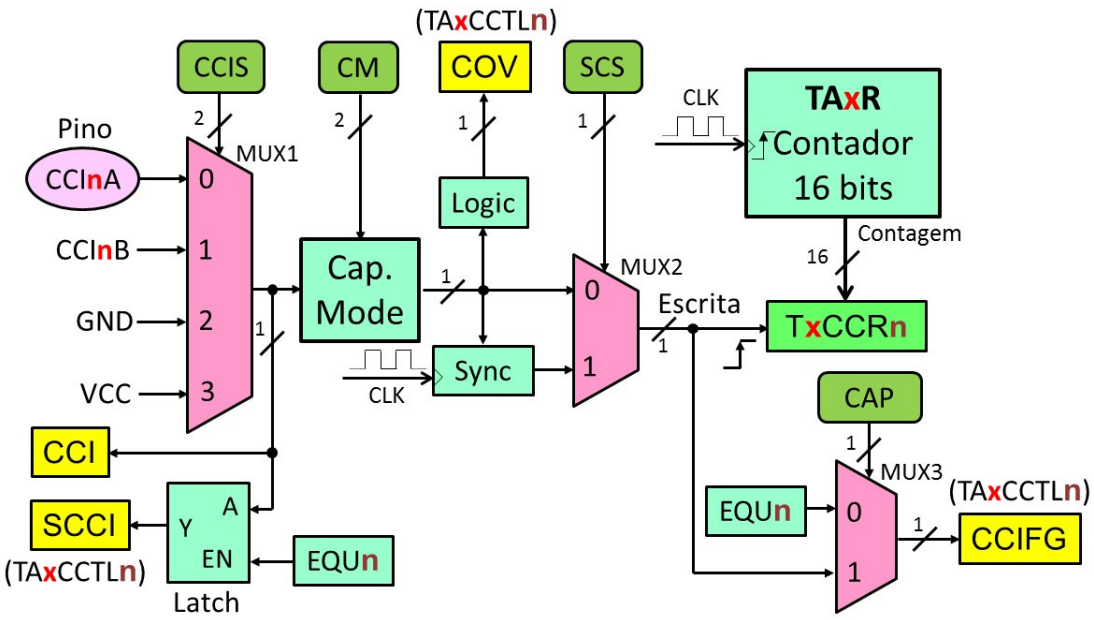

Figura 7.22. Diagrama de blocos para a operação no Modo Captura de Eventos.

Ainda na Figura 7.22, a flag COV indica atropelamento na captura, ou seja, aconteceu uma nova captura sendo que o resultado a captura anterior ainda não foi lido. Ou seja, perdeu-se uma captura. O programador é responsável por zerar esta flag. Para terminar, o leitor deve notar o bit CCI que permite ao programador, a qualquer momento ler o estado da entrada selecionada para gerar o evento de captura. Já o bit SCCI faz a sincronização deste sinal com a saída do comparador. Para que isso?

Tabela 7.4. Modos de Captura disponíveis

CM	Modo de Captura
0	Sem captura
1	Flanco de subida (↑)
2	Flanco de descida (↓)
3	Ambos os flancos (↕)

Tabela 7.5. Distribuição das entradas de captura nas Portas GPIO para a arquitetura F5529

Timer A0	
TA0.0	P1.1

Timer A1	
TA1.0	P1.7

Timer A2	
TA2.0	P2.3

TA0.1	P1.2	TA1.1	P2.0	TA2.1	P2.4
TA0.2	P1.3	TA1.2	P2.1	TA2.2	P2.5
TA0.3	P1.4	-	-	-	-
TA0.4	P1.5	-	-	-	-

7.2.2. Interrupções com o Timer A

Cada instância do Timer A, tem dedicada a ela duas posições da tabela de vetores de interrupção. Uma posição (vetor) está dedicada exclusivamente ao comparador 0 e a outra (vetor), aos demais comparadores e ao TAIFG. A Figura 7.23 apresenta uma ideia mais clara deste tópico, onde se podem ver as flags que provocam interrupções (CCIFG e TAIFG) e as respectivas habilitações (CCIE e TAIE). É claro que se a habilitação está em 0, a interrupção correspondente não acontece. Não se pode esquecer da habilitação geral feita com o bit GIE do registrador de estado.

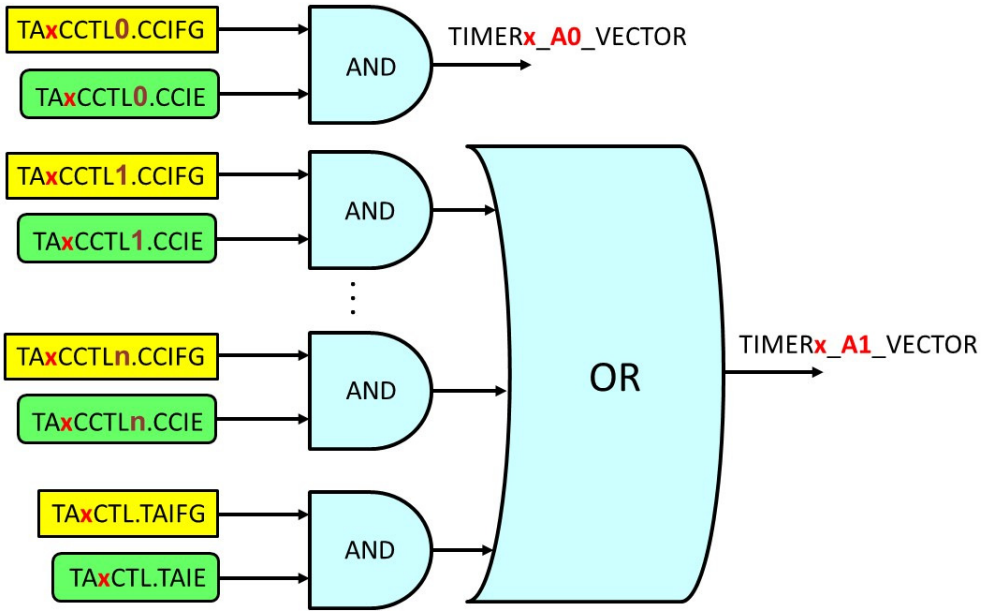

Figura 7.23. Esquema de distribuição dos dois vetores de interrupção para cada instância do Timer A.

A Tabela 7.6 apresenta a lista completa de todas as interrupções dedicadas às diversas instâncias do Timer A. Nesta tabela o vetor de maior número tem maior prioridade. O CCS traz um label (uma constante) para cada um desses vetores. É importante dar atenção à essa denominação, já que os nomes não foram felizes e podem induzir a erros. Muitas vezes, é mais seguro usar o número do vetor.

Relembrando o comportamento das flags:

- Em modo comparação, TAxCCIFGn = 1 quando TAxCCRn = TAxR;

- Em modo captura, TAxCCIFGn = 1 quando ocorre a captura programada e
- Em qualquer modo, TAIFG = 1 sempre que TAxR voltar a zero.

Essas flags também podem ser alteradas por software.

Tabela 7.6. Distribuição das entradas do vetor de interrupções para as diversas instâncias do Timer A

Fonte	Flags	Vetor	Label para o Vetor
Timer A0	TA0CCTL0.CCIFG	53	TIMER0_A0_VECTOR
	TA0CCTL1.CCIFG ... TA0CCTL4.CCIFG TA0CTL.TAIFG	52	TIMER0_A1_VECTOR
Timer A1	TA1CCTL0.CCIFG	49	TIMER1_A0_VECTOR
	TA1CCTL1.CCIFG ... TA1CCTL2.CCIFG TA1CTL.TAIFG	48	TIMER1_A1_VECTOR
Timer A2	TA2CCTL0.CCIFG	44	TIMER2_A0_VECTOR
	TA2CCTL1.CCIFG ... TA2CCTLn.CCIFG TA2CTL.TAIFG	43	TIMER2_A1_VECTOR

Como existe um vetor exclusivo para a interrupção do comparador 0 (TAxCCTL0.CCIFG), sua manipulação é simples, como mostrado na listagem abaixo, que considera o uso de TA0. Se o leitor tiver dúvidas sobre interrupções, é recomendado a leitura do Capítulo 6. Neste caso, quando processador desvia para a rotina de interrupção, a flag CCIFG é zerada automaticamente.

```
// Tratar TA0CCTL0.CCIFG
#pragma vector = 53  //#pragma vector = TIMER0_A0_VECTOR
__interrupt void isr_ta0_ccifg0(void){
      ;
      Rotina para tratar a interrupção;
      ;
    }
}
```

Porém, existe um único vetor para as demais flags de interrupções (CCIFGs e TAIFG), assim, a rotina que trata a interrupção precisa verificar, dentre os possíveis candidatos, quem provocou a interrupção. O leitor pode imaginar que esta verificação vá tomar tempo, pois são vários candidatos. Para facilitar este trabalho o MSP traz o registrador TAxIV. Ele retorna um número (valor) correspondente à interrupção de maior prioridade dentre as flags que foram ativadas, como apresentado na Tabela 7.7. Quanto menor o número, maior a prioridade. É de se notar que os valores são números pares. No caso de assembly, eles são somados diretamente ao contador de programa (PC). A cada leitura de TAxIV, a flag de maior prioridade é automaticamente zerada. Ao retornar da rotina de interrupção, caso haja algum pedido pendente, uma nova interrupção é gerada e assim prossegue até que não haja mais pedidos pendentes.

Tabela 7.7. Possíveis valores do registrador TAxIV

Prioridade	Valor	Descrição
-	0	Nenhuma interrupção pendente
Maior	2	TAxCCTL1.CCIFG
	4	TAxCCTL2.CCIFG
	6	TAxCCTL3.CCIFG
	8	TAxCCTL4.CCIFG
	...	...
Menor	-	TAxCTL.TAIFG

A listagem abaixo apresenta uma sugestão para a rotina de interrupção para tratar os pedidos do TA0. Relembrando, a função `__even_in_range(TA0IV,0xA)` garante que o valor lido do registrador TA0IV seja par e esteja dentro da faixa de 0x0 até 0xA. De acordo com o valor retornado, uma determinada função é executada.

```
// Tratar TA0CCTL1.CCIFG, TA0CCTL2.CCIFG, ..., TA0CTL.TAIFG

#pragma vector = 53  //#pragma vector = TIMER0_A1_VECTOR
__interrupt void isr_ta0(void){
    int n;
    n = __even_in_range(TA0IV,0xA);
    switch(n){
        case 0x0: break;
        case 0x0: ta0_ccifg1();     break;
        case 0x2: ta0_ccifg2();     break;
        case 0x4: ta0_ccifg3();     break;
        case 0x6: ta0_ccifg4();     break;
        case 0xA: ta0_taifg();      break;
    }
}

// Tratar TA0CCTL1.CCIFG
void ta0_ccifg1 (void){ ...}  // Tratar TA0CCTL1.CCIFG
void ta0_ccifg2(void){ ...}   // Tratar TA0CCTL1.CCIFG
void ta0_ccifg3(void){ ...}   // Tratar TA0CCTL1.CCIFG
```

```
void ta0_ccifg(void){ ...}     // Tratar TA0CCTL1.CCIFG
void ta0_taifg (void){ ...}    // Tratar TA0CCTL1.CCIFG
```

7.2.3. Registradores do Timer A

A seguir, na Tabela 7.8, estão listados os registradores para o controle e operação do Timer A. Logo a seguir se faz a descrição de cada um deles.

Tabela 7.8. Registradores do Timer A

16 bits	Acesso	Reset
TAxCTL	R/W	0
TAxR	R/W	0
TAxCCTLn	R/W	0
TAxCCRn	R/W	0
TAxIV	R/W	0
TAxEX0	R/W	0

7.2.3.1. TAxCTL – Registrador de Controle do Timer Ax (Timer Ax Control *Register*)

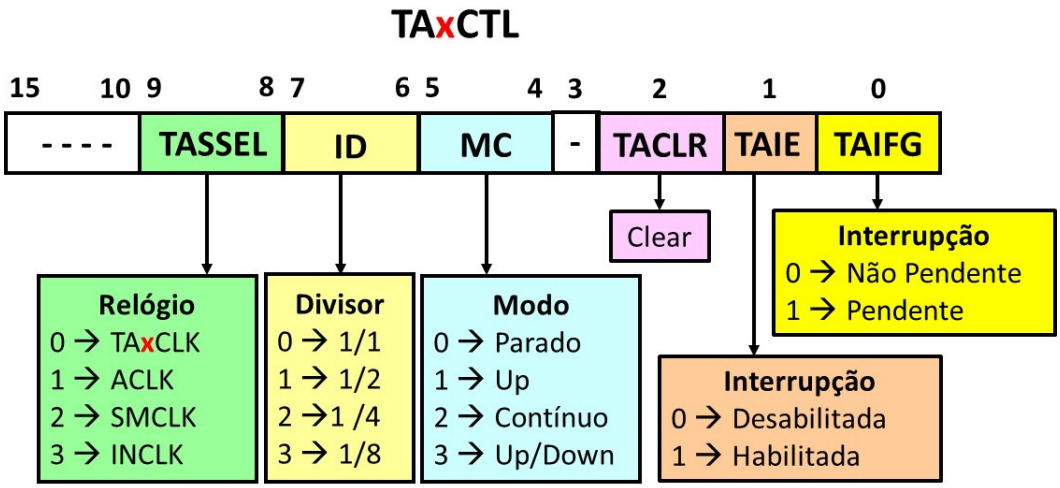

Figura 7.24. Descrição dos bits do registrador TAxCTL (Reset faz TAxCTL=0).

(R/W) *Bits* 15:10: Reservados

(R/W) *Bits* 9:8: TASSEL – Selecionar relógio para TAx
(Timer Ax clock source select)
O usuário seleciona o relógio que será usado na contagem realizadas pelo TAxR.

- TASSEL = 0 → Usar relógio fornecido no pino TAxCLK (ver Figura 7.16), TA0CLK = P1.0, TA1CLK =P1.6 e TA2CLK = P2.2
- TASSEL = 1 → ACLK é selecionado
- TASSEL = 2 → SMCLK é selecionado
- TASSEL = 3 → Não disponível no F5529

(R/W) *Bits* 7:6: ID – Divisor do relógio para TAx
(Input divider)
O usuário indica um divisor para o relógio selecionado, que deve ser usado junto com o registrador TAxIDEX, que oferece mais divisores.

- ID = 0 → CLK/1
- ID = 1 → CLK/2
- ID = 2 → CLK/3
- ID = 3 → CLK/4

(R/W) *Bits* 5:4: MC – Controle do modo de TAx
(Mode control)
O usuário seleciona um dos 4 modos de contagem disponíveis.

- MC = 0 → Contador parado (economiza energia)
- MC = 1 → Modo ascendente (Up)
- MC = 2 → Modo Contínuo (Continuous)
- MC = 3 → Modo Ascendente/Descendente (Up/Down)

(R/W) *Bit* 3: Reservado

(R/W) *Bit* 2: TACLR – Zeragem do contador TAxR
(Timer A clear)
Ao ser ativado, este bit provoca a zeragem do contador TAxR e do estado interno dos divisores do relógio. Este bit volta a zero automaticamente, ou seja, ele é sempre lido como zero.

(R/W) *Bit* 1: TAIE – Habilitação da interrupção por ultrapassagem (TAIFG)
(Timer A interrupt enable)
Quando este bit está em 1, a flag TAIFG provoca interrupção quando ativada.

(R/W) *Bit* 0: TAIFG – Flag de ultrapassagem
(Timer A interrupt flag)
Esta flag vai para 1 de acordo com o modo de contagem selecionado. Ela pode ser consultado (polling) pelo programa ou provocar interrupção se o bit TAIE estiver em 1. Esta flag pode ser apagada por software.

7.2.3.2. TAxR – Contador do Timer Ax

(Timer Ax *Register*)

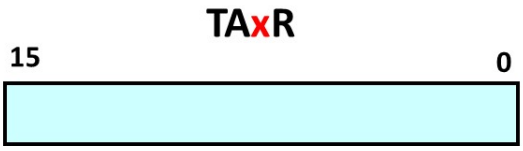

Figura 7.25. Contador de 16 bits do Timer A (Reset faz TAxR = 0).

(R/W) *Bits* 15:0: TAxR – Contador de TAx
(Timer A register)
Este é um registrador de 16 bits, responsável por fazer as contagens. Ele pode ser lido ou escrito a qualquer momento.

7.2.3.3. TAxCCTLn – Registrador de Controle da instância n da unidade de captura e comparação
(Timer Ax Capture/Compare Control *n Register*)

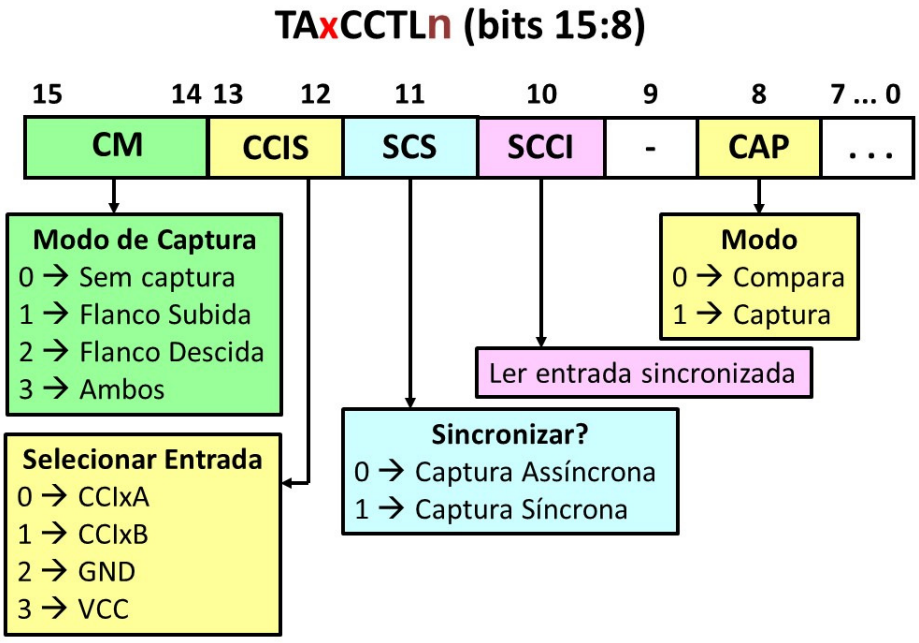

Figura 7.26.a. Descrição dos bits 15:8 do registrador de controle da instância n de captura e comparação do Timer A (Reset faz TAxCCTLn= 0).

(R/W) *Bits* 15:14: CM – Modo de captura
(Capture mode)
O usuário indica que tipo de flanco caracteriza o evento que ele deseja capturar.

- CM = 0 → Nenhuma captura;
- CM = 1 → (↑) Captura no flanco de subida
- CM = 2 → (↓) Captura no flanco de descida
- CM = 3 → (↕) Captura em ambos os flancos

(R/W) *Bits* 13:12: CCIS – Seleção da entrada de captura
(Capture/Compare input select)
O usuário seleciona por qual entrada ele vai apresentar o evento a ser capturado. Na verdade, apenas a entrada CCIxA está disponível externamente. Na versão F5529, a entrada CCIxB não está disponível. Veja Tabela 7.5 para identificar os pinos que disponibilizam as entradas CCIxA para cada instância do Timer A. A alternância entre as opções 2 (GND) e 3 (VCC) permite que, por software, se provoque um flanco de subida ou de descida.

- CM = 0 → CCIxA
- CM = 1 → CCIxB
- CM = 2 → GND (terra)
- CM = 3 → VCC (alimentação)

(R/W) *Bit* 11: SCS – Sincronizar entrada de captura
(Synchronize Capture source)
Quando este bit está em 1, o sinal a ser capturado é sincronizado com o relógio de TAx. Com este bit em 0, o sinal a ser capturado é assíncrono com o contador TAxR, o que pode resultar em metaestabilidade. É recomendado sempre habilitar esta sincronização.

(R/W) *Bit* 10: SCCI – Ler entrada de captura sincronizada
(Synchronized capture/compare input)
Por este bit se pode ler a entrada de captura, amostrada no instante em que o sinal interno EQUn é ativado. Para que serve isso? Quando opera no modo Up, permite saber se aconteceu captura ?

(R/W) *Bit* 9: Reservado

(R/W) *Bit 8*: CAP – Seleção do modo captura
(Capture mode)
Por este bit o usuário escolhe se quer usar o Modo de Comparação (CAP = 0) ou o Modo de Captura (CAP = 1).

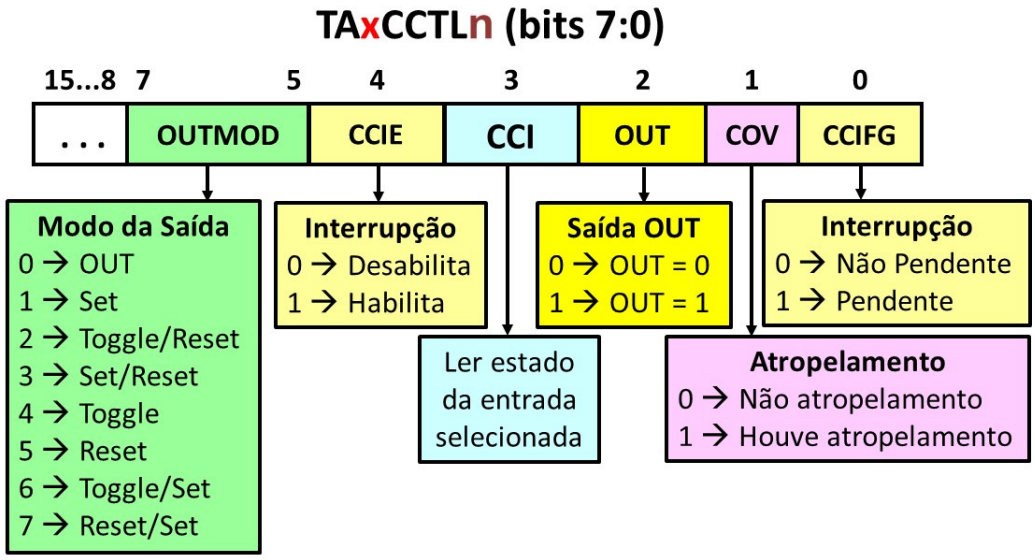

Figura 7.26.b. Descrição dos bits 7:0 do registrador de controle da instância n de captura e comparação do Timer A (Reset faz TAxCCTLn= 0).

(R/W) *Bits* 7:5: OUTMOD – Seleção do modo de saída
(Output mode)
O usuário seleciona como deve operar a unidade geradora de saída. Ver as Figura 7.19, 7.20 e 7.21. Com OUTMOD = 0, a saída acompanha o estado do bit 2 deste registrador, denominado OUT.

- OUTMOD = 0 → Acompanha bit OUT
- OUTMOD = 1 → Set
- OUTMOD = 2 → Toggle/Reset
- OUTMOD = 3 → Set/Reset
- OUTMOD = 4 → Toggle
- OUTMOD = 5 → Reset
- OUTMOD = 6 → Toggle/Set
- OUTMOD = 7 → Reset/Set

(R/W) *Bit* 4: CCIE – Habilitação da interrupção por captura/comparação (CCIFG)
(Capture/Compare interrupt enable)
Quando este bit está em 1, a flag CCIFG provoca interrupção quando ativada.

(R/W) *Bit* 3: CCI – Entrada de captura
(Capture/compare input)
Por este bit se pode ler o estado instantâneo da entrada de captura.

(R/W) *Bit* 2: OUT – Saída
(Output)
Com OUTMOD = 0, a saída acompanha o estado deste bit.

(R/W) *Bit* 1: COV – Atropelamento na captura
(Capture overflow)
Esta flag vai para 1 quando um novo evento de captura acontece antes do resultado da captura anterior ser lido. Esta flag precisa ser zerada por software.

(R/W) *Bit* 0: CCIFG – Flag de captura/comparação
(Capture/compara interrupt flag)
Quando no Modo Comparação, esta flag vai para 1 para indicar que o valor do contador TAxR ficou igual ao do registrador de comparação TAxCCRn. Quando no Modo Captura, esta flag vai para 1 para indicar que o evento de captura selecionado ocorreu e que o resultado está disponível no registrador TAxCCRn. Esta flag pode provocar interrupção quando o bit CCIE está em 1. Ela pode ser apagada por software.

7.2.3.4. TAxR – Reg. da Unidade n de Captura/Comparação do Timer Ax (*Timer Ax Capture/Compare n Register*)

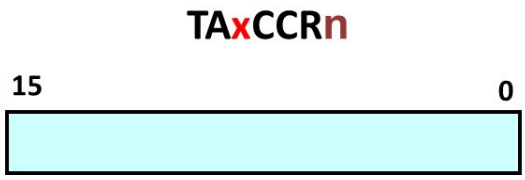

Figura 7.27. Registrador da instância n da Unidade de Captura/Comparação do timer TAx. (Reset faz TAxCCRn= 0).

(R/W) *Bits* 15:0: TAxCCRn – Registrador da Unidade n de Captura/Comparação (*Capture/Compare n Register*)
Se no modo comparação (CAP = 0), este registrador armazena o valor a ser comparado com o contador TAxR. Se no modo captura (CAP = 1), conteúdo do contador TAxR é copiado para este registrador, quando ocorre o evento de captura especificado.

7.2.3.5. TAxIV – Registrador de Vetor de Interrupção do Timer Ax (*Timer Ax Interrupt Vector Register*)

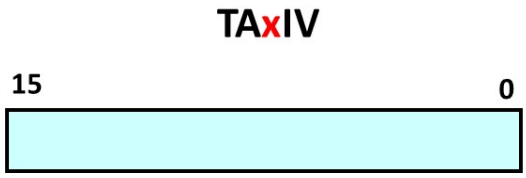

Figura 7.28. Registrador de Vetor de Interrupção do timer TAx. (Reset faz TAxIV= 0).

(R) *Bits* 15:0: TAxCCRn – Valor do Vetor de Interrupção
(Interrupt Value *Register*)
A leitura deste registrador retorna um número que indica, dentre as interrupções pendentes, a de maior prioridade e, ao mesmo tempo, apaga este pedido pendente (o de maior prioridade). A tabela abaixo indica as opções para as instâncias do Timer A disponíveis na versão F5529.

Tabela 7.9. Possíveis valores do Registrador TAxIV, de acordo com as instâncias do Timer A disponíveis na versão F5529.

Valor	Timer A0	Timer A1	Timer A2
0x0	Nada pendente	Nada pendente	Nada pendente
0x2	TA0CCR1.CCIFG	TA1CCR1.CCIFG	TA2CCR2.CCIFG
0x4	TA0CCR2.CCIFG	TA1CCR2.CCIFG	TA2CCR3.CCIFG
0x6	TA0CCR3.CCIFG	TA1CTL.TAIFG	TA2CTL.TAIFG
0x8	TA0CCR4.CCIFG	-	-
0xA	TA0CTL.TAIFG	-	-

7.2.3.6. TAxEX0 – Registrador 0 para Expansão de Timer Ax (*Timer Ax Expansion 0 Register*)

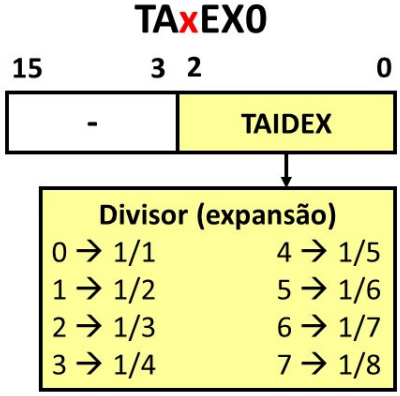

Figura 7.29. Registrador 0 para expansão do timer TAx. (Reset faz TAxEX0 = 0).

(R) *Bits* 15:3: Reservados

(R/W) *Bits 2:0*: TAIDEX – Expansão do divisor do relógio
(Input divider expansion)
Esses 3 bits oferecem mais opções para a divisão do relógio que vai acionar o contador TAxR. O valor máximo de divisão que se pode obter é igual a 64 (ID = 3 → 1/8 e TAIDEX = 7 → 1/8). Vide Figura 7.16.

7.3. Estudo do Timer B

O MSP430F5529 oferece apenas a instância zero do Timer B, com 7 instâncias de Unidades de Captura/Comparação. O timer B tem muita semelhança com o timer A, razão pela qual se recomenda o estudo do tópico anterior, que aborda com detalhes o timer A. Aqui neste tópico serão apresentadas apenas as diferenças entre estes dois timers.

O timer B apresenta as seguintes diferenças com relação ao timer A:

- O tamanho do contador (TB0R) é configurável para 8, 10, 12 ou 16 bits;
- Os registradores TB0CCRn têm dupla buferização e podem ser agrupados;
- Todas as saídas do timer B0 podem entrar no estado de alta-impedância e
- O bit SCCI não está disponível.

Já foi dito que existem registradores para controlar a operação do contador e o funcionamento das unidades de captura e comparação. Como na versão F5529 está disponível apenas a instância zero, os nomes dos registradores sempre serão iniciados com TB0. A tabela abaixo resume os registradores deste timer. O controle do contador é feito pelo registrador TB0CTL. Os diversos registradores de controle das unidades de captura e compara são denominados TB0CCTLn, onde n = 0, 1, ...6.

Tabela 7.10. Resumo dos registradores das instâncias do Timer B e de seus registradores de Captura e Compara

	Registrador	Controle	Flag IFG
Contador	TB0R	TB0CTL	TAIFG
Captura/Compara	TB0CCRn	TB0CCTLn	CCIFG

7.3.1. Excursão da Contagem do Timer B0

O contador TB0R pode ser configurado para operar com 8, 10, 12 ou 16 bits. Estas opções são selecionadas com os bits CNTL do registrador TB0CTL. Os valores escritos

no registrador TB0R quando este opera com menos de 16 bits, são alinhados pela direita e a porção da esquerda é preenchida com zeros, como seria esperado. Abaixo estão listados os valores máximos de contagem, para cada excursão disponível.

- 8 bits → máximo = 0x00FF;
- 10 bits → máximo = 0x03FF;
- 12 bits → máximo = 0x0FFF;
- 16 bits → máximo = 0xFFFF;

7.3.2. Dupla buferização do Timer B0

A necessidade da dupla buferização se explica pela possibilidade de que, durante a geração de PWM, uma alteração no registrador de comparação gere pulsos de largura não desejada. Para explicar melhor vamos considerar a situação apresentada na Figura 7.30. Nesta figura o contador está operando no Modo Ascendente (Modo Up) e o registrador TB0CCRn é usado para controlar o ciclo de carga do PWM. Podemos ver que o valor de TB0CCRn era "q" e foi alterado para "p", sendo p < q. Porém, essa troca de valor pode acontecer num instante ruim, como ilustrado nesta figura. Note que o contador TB0R estava entre "p" e "q" quando a troca aconteceu. Como TB0CCRn foi alterado para um valor abaixo do valor atual de TB0R, este vai contar até o limite (valor de TB0CCR0) e durante este período de contagem nunca teremos TB0R = TB0CCRn. Isto significa que a saída permanece em 1, gerando um pulso largo, diferente do que se desejava.

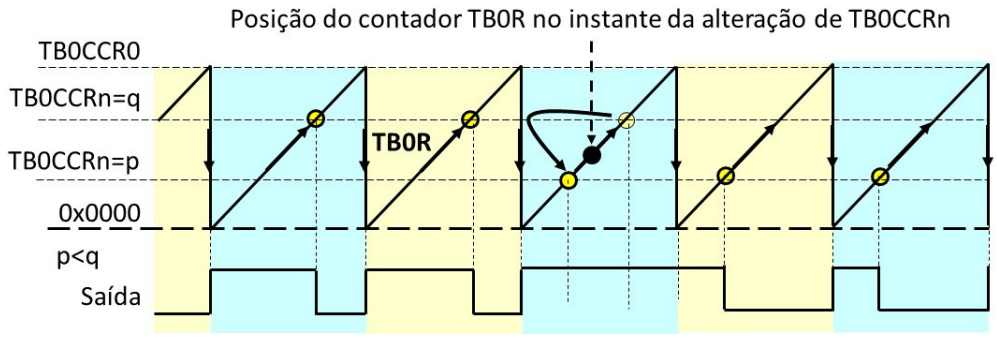

Figura 7.30. Exemplo de uma saída inesperada que pode acontecer quando se altera do registrador de comparação (TB0CCRn), sem o recurso de dupla buferização.

Na Figura 7.31 é possível ver que o programador não tem acesso direto ao registrador TB0CLn, que é usado na comparação. Para mudar o valor de comparação, ele deve escrever no registrador TB0CCRn e, de acordo com o que foi programado nos bits CLLD, o valor é transferido para TB0CLn num instante de tempo muito bem determinado. Com o uso deste recurso, o problema indicado na Figura 7.30 não aconteceria. A Tabela 7.11 apresenta as opções para a seleção do instante desta transferência.

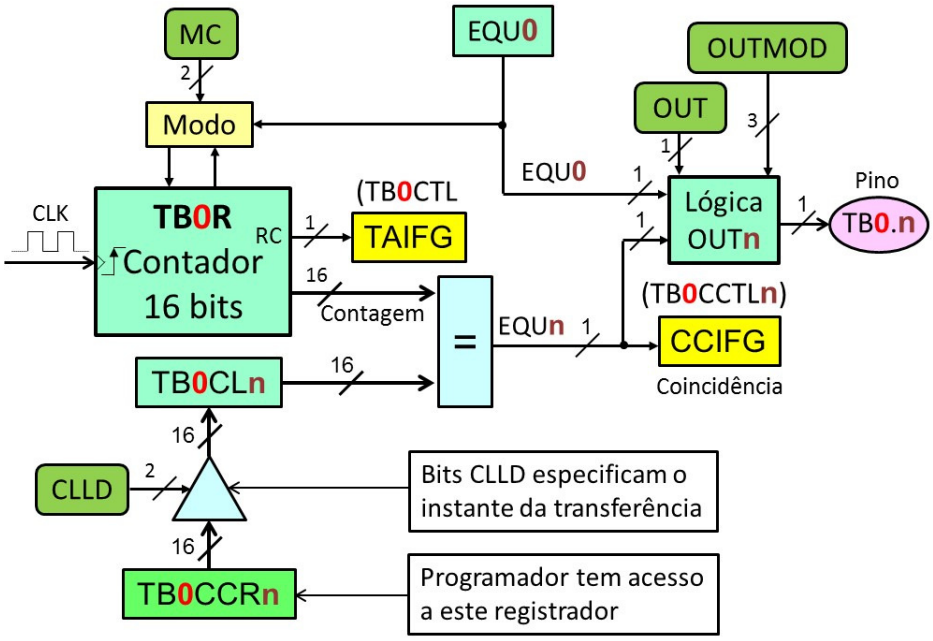

Figura 7.31. Diagrama de blocos para a Unidade Geradora de Saídas (Lógica OUTn) do comparador n, indicando a lógica de dupla buferização.

Tabela 7.11. Definição do instante de atualização do registrador TB0CTLn.

CLLD	Transferência	Descrição
0	TB0CCRn → TB0CLn	Imediata
1		TB0R = 0
2		(MC = 1 ou 2) TB0R = 0 (MC = 3) TB0R = 0 ou TB0R = TB0CL0 antigo
3		TB0R = TB0CLn antigo

7.3.3. Agrupamento de Registradores de Comparação do Timer B0

Algumas aplicações podem exigir que o usuário altere, simultaneamente, dois registradores de comparação. Por exemplo, imaginemos o caso de se alterar o ciclo de carga de duas saídas PWM que precisam estar sincronizadas. Mesmo usando a dupla buferização, pode ser que essas duas alterações fiquem separadas por um período de contagem. Com o que foi visto até agora, é impossível garantir que dois registradores sempre sejam alterados ao mesmo tempo, já que se faz necessário o uso de duas instruções de escrita.

Para solucionar esta necessidade, o hardware do Timer B faz uso da dupla buferização, mas agora o instante da transferência dos valores é dado pela escrita num determinado registrador. A isso se denomina agrupamento. Por exemplo, os registradores de comparação TB0CCR1 e TB0CCR2 podem ser agrupados, assim o usuário escreve em

TB0CCR2 e depois em TB0CCR1. Quando acontecer a segunda escrita, os dois registradores (TB0CL1 e TB0CL2) são atualizados, simultaneamente.

Quando se trabalha com agrupamento de registradores, a escrita no registrador de menor número define o momento da atualização de todo o agrupamento. A Tabela 7.12 apresenta as opções de agrupamento. Há uma particularidade. Note que na última opção, todos os registradores de comparação estão agrupados, incluindo TB0CTL0, e que o controle é feito pela escrita em TB0CCR1.

Tabela 7.12. Opções de agrupamento de registradores

TBCLGRP0	Agrupamento	Controle
0	Nenhum	Individual
1	TB0CL1 + TB0CL2 TB0CL3 + TB0CL4 TB0CL5 + TB0CL6	TB0CCR1 TB0CCR3 TB0CCR5
2	TB0CL1 + TB0CL2 + TB0CL3 TB0CL4 + TB0CL5 + TB0CL6	TB0CCR1 TB0CCR4
3	TB0CL0 + TB0CL1 + … + TB0CL6	TB0CCR1

7.3.4. Registradores do Timer B

A seguir, Tabela 7.14, estão listados os registradores para o controle e operação do Timer B. Como a versão F5529 oferece apenas a instância zero, todos esses registradores iniciam com as letras TB0.

Tabela 7.14. Registradores do Timer A

16 bits	Acesso	Reset
TB0CTL	R/W	0
TB0R	R/W	0
TB0CCTLn	R/W	0
TB0CCRn	R/W	0
TB0IV	R/W	0
TB0EX0	R/W	0

A seguir são os bits dos diversos registradores. Como é grande a semelhança com o Timer A, apenas pontos divergentes serão detalhados.

7.3.4.1. TB0CTL – Registrador de Controle do Timer B0 (Timer B0 Control *Register*)

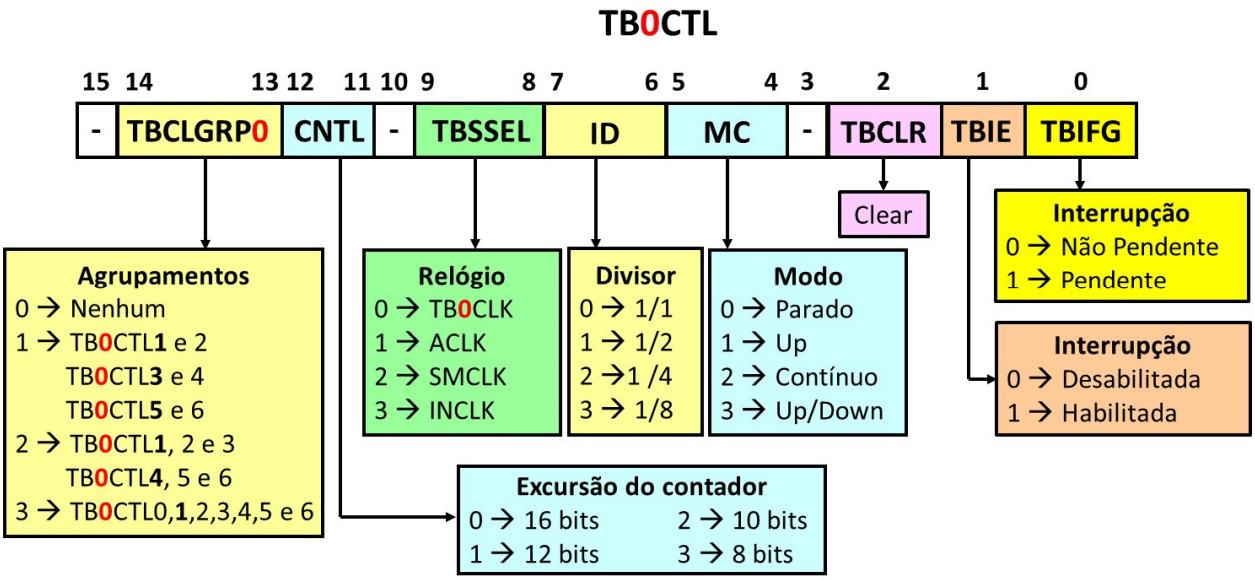

Figura 7.32. Descrição dos bits do registrador TAxCTL (Reset faz TAxCTL=0).

(R/W) *Bits* 14:13: TBCLGRP0 – Controle de agrupamento (TB0CLn Group)

O usuário seleciona o agrupamento que deseja para os registradores de captura/contagem. Nos modos de agrupamento 1 e 2, a escrita no registrador de menor número dispara a escrita em todo o grupo. Note que na opção 3 estão agrupados todos os registradores de captura/contagem.

- TBCLGRP0 = 0 → Nenhum agrupamento;
- TBCLGRP0 = 1 → (TB0CCR1) TB0CTL1 + TB0CTL2, (TB0CCR3) TB0CTL3 + TB0CTL4, (TB0CCR5) TB0CTL5 + TB0CTL6. A escrita no registrador entre parêntesis dispara a escrita em todo o grupo.
- TBCLGRP0 = 2 → (TB0CCR1) TB0CTL1 + TB0CTL2 + TB0CTL3, (TB0CCR4) TB0CTL4 + TB0CTL5 + TB0CTL6. A escrita no registrador entre parêntesis dispara a escrita em todo o grupo.
- TBCLGRP0 = 3 → TB0CTL0 + (TB0CCR1) TB0CTL1 + TB0CTL2 + TB0CTL3 + TB0CTL4 + TB0CTL5 + TB0CTL6. A escrita no registrador entre parêntesis dispara a escrita em todo o grupo.

(R/W) *Bits* 12:11: CNTL – Controle da excursão da contagem (Counter lenght)

O usuário seleciona o tamanho (número de bits) do contador TB0R.

- CNTL = 0 → 16 bits, máximo com TB0R = 0xFFFF
- CNTL = 1 → 12 bits, máximo com TB0R = 0xFFF
- CNTL = 2 → 10 bits, máximo com TB0R = 0x3FF
- CNTL = 3 → 8 bits, máximo com TB0R = 0xFF

7.4.4.2. TB0R – Contador do Timer B, instância 0 (Timer B0 *Register*)

Figura 7.33. Contador de 16 bits do Timer B, instância 0 (Reset faz TB0R= 0).

7.4.4.3. TB0CCTLn – Registrador de Controle da instância 0 da unidade de captura e comparação (Timer B0 Capture/Compare Control *n Register*)