

11

Conversor Analógico/Digital 12 bits (ADC12)

Versão 1.0

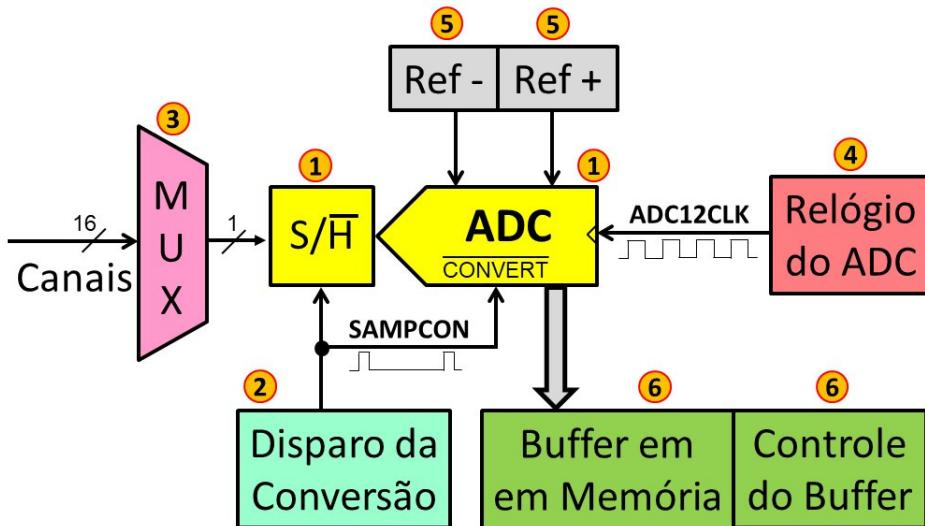
Nas aplicações com microcontroladores, é comum necessitarmos medir grandezas físicas tais como temperatura, pressão, umidade, quantidade de luz, PH, etc. Para tanto, precisamos dos *transdutores* que convertem essas grandezas para sinais elétricos. A saída de tais transdutores costuma ser um sinal elétrico analógico, linearmente proporcional ou, de alguma forma, relacionado com grandeza que se está medindo. Para que o microcontrolador possa trabalhar com esses sinais analógicos, é necessário convertê-los em digitais. Esta é a tarefa que está a cargo do Conversor Analógico Digital, também chamado de Conversor A/D ou, simplesmente, ADC.

Sendo um pouco mais formal, o ADC converte um sinal analógico contínuo no tempo em um sinal amostrado, em tempo discreto e quantizado dentro de sua faixa de resolução. Recomenda-se aos interessados, um estudo mais profundo do tema Sinais em Tempo Discreto.

Antes de abordarmos o conversor ADC12 da família MSP430, vamos apresentar um pequeno estudo sobre Conversores Digital-Analógico (DAC) e Conversores Analógico-Digital (ADC). Este estudo não tem a pretensão de esgotar assunto, mas é importante para a compreensão dos recursos e características que estão disponibilizados neste processador. Se o leitor não se interessar ou já for conhedor do assunto, pode avançar para o próximo tópico (item 11.2).

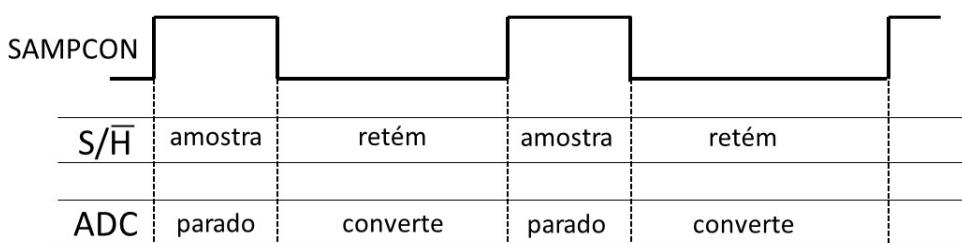
**11.0. Quero Usar os ADC12 e
não Pretendo Ler Todo este Capítulo**

O diagrama abaixo resume o conversor AD do MSP430 F5529 que é denominado ADC12. A seguir é feita uma rápida descrição de seus diversos componentes, de acordo com os números indicados.



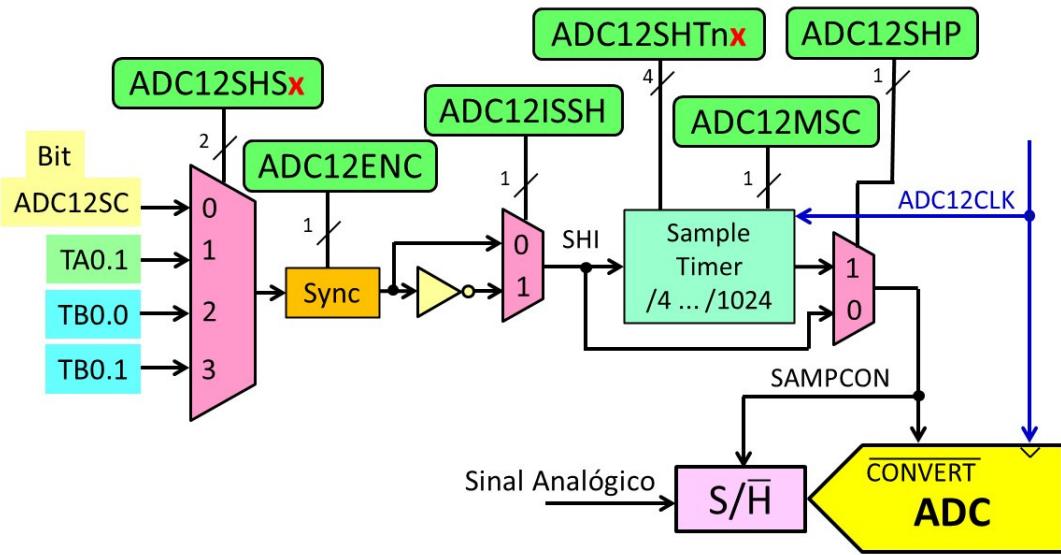
Componente 1

É o conversor (ADC) conectado a uma unidade para amostragem e retenção (S/H = sample and holder). O sinal SAMPON em nível alto coloca S/H no modo amostragem e mantém o ADC parado. SAMPON em nível baixo coloca S/H no modo retenção e inicia uma conversão.



Componente 2

É responsável por gerar o sinal SAMPON para amostrar e disparar as conversões. Devido a sua importância, apresentamos o diagrama abaixo para facilitar sua configuração. Veja que SAMPON pode ser gerado diretamente (via SHI) a partir de TA0 ou TB0 ou por software (ADC12SC). Neste caso o programador é o responsável por garantir que o sinal de disparo fique em nível alto tempo necessário para a amostragem (S/H). Por outro lado, o bloco Sample Timer permite que se programe o tempo de amostragem e assim, torna-se necessário apenas um flanco de subida em SHI.



Componente 3

Aqui temos um multiplexador analógico com 16 entradas (A_0, A_1, \dots, A_{15}) por onde o usuário seleciona o sinal a ser convertido. A tabela abaixo indica a disposição dessas 16 entradas, sendo que as marcadas em cinza claro não estão acessíveis no barramento de pinos da LaunchPad.

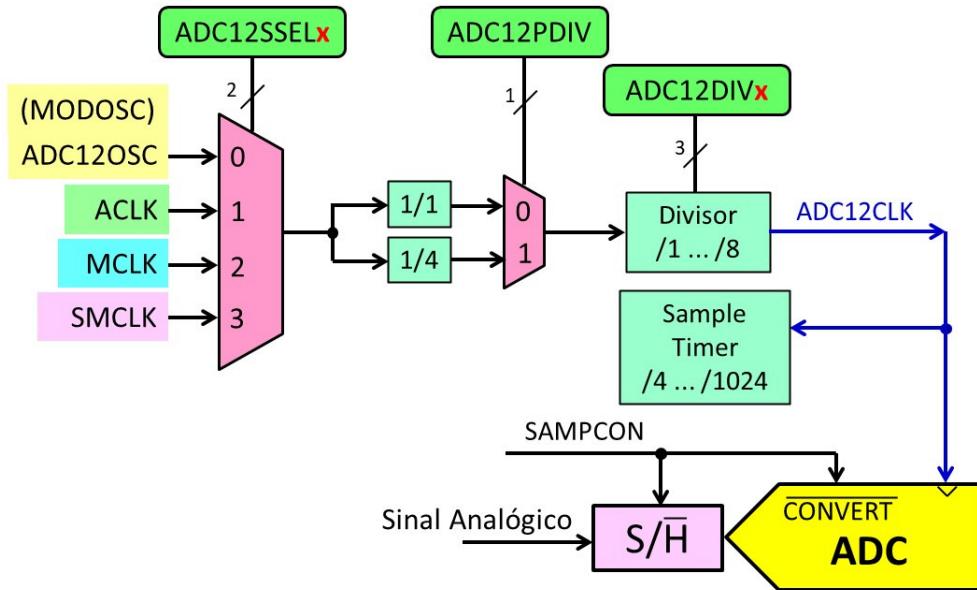
Entrada	Pino	Finalidade
A_0	P6.0	Geral
A_1	P6.1	Geral
A_2	P6.2	Geral
A_3	P6.3	Geral
A_4	P6.4	Geral
A_5	P6.5	Geral
A_6	P6.6	Geral
A_7	P6.7	Geral

Entrada	Pino	Finalidade
A_8	-	V_{eREF+}
A_9	-	V_{eREF-}
A_{10}	-	Temperatura
A_{11}	-	$AVcc/2$
A_{12}	P7.0	Geral
A_{13}	P7.1	Geral
A_{14}	P7.2	Geral
A_{15}	P7.3	Geral

Componente 4

Seu diagrama de blocos está logo abaixo e permite ao usuário selecionar o relógio ADC12CLK para a máquina de estados do ADC12. É claro que este relógio deve ter uma frequência superior à frequência de conversão (f_s). Como regra simples sugerimos usar a expressão abaixo, onde N é a quantidade de bits (8, 10 ou 12) selecionada. À exceção do consumo de energia, não há problema em usar um ADC12CLK bem alto.

$$ADC12CLK > 2 \times N \times f_s$$



Componente 5

Um diagrama completo deste componente está na Figura 11.33. Ele permite selecionar as referências de tensão positiva (V_{R+}) e negativa (V_{R-}) para o conversor. Em geral usamos o AVcc (3,3 V) como referência positiva e o AGND (0,0 V) como referência negativa. Internamente estão disponíveis referências de 1,5V, 2,0V e 2,5V. O código entregue pelo ADC12 quando operando em 12 bits é dado pela equação abaixo, onde V_{in} é a tensão analógica na entrada. O número 4.095 deve ser substituído por 1.023 quando se opera em 10 bits e por 255 para o caso de 8 bits. O campo ADC12RES permite ao usuário selecionar a resolução. O reset configura a resolução para 12 bits.

$$N = 4.095 \times \frac{V_{in} - V_{R-}}{V_{R+} - V_{R-}}$$

Componente 6

Este componente contém as 16 posições de memória usadas para receber o resultado das conversões e seu diagrama está na Figura 11.34. São 16 posições de 16 bits denominadas ADC12MEM0, ..., ADC12MEM15. Para cada uma dessas posições existe um registrador de controle de 8 bits denominado ADC12MCTL0, ..., ADC12MCTL15. Cada registrador de controle especifica (para sua memória) o canal analógico a ser usado, as referências de tensão e ainda um bit, denominado EOS, para indicar se essa posição é a última de uma sequência. Com o uso inteligente desta memória, se pode operar em 4 modos.

- (CONSEQ = 0) Conversão de um canal;
- (CONSEQ = 1) Conversão de uma sequência de canais;
- (CONSEQ = 2) Conversão de um canal com repetição e

- (CONSEQ = 3) Conversão de uma sequência de canais com repetição.

Na configuração o programador indica no campo ADC12CSTARTADD um número de 0 a 15 para especificar a primeira posição de memória a ser usada. Se o ADC12 estiver no modo de conversão de um canal, o disparo realiza a conversão de acordo com o programado na memória de controle. Entretanto, se for o modo sequência de canais, as conversões continuam pelas próximas posições de memória até encontrar uma com seu registrador de controle marcado com o bit EOS = 1. Em qualquer um dos dois casos anteriores, ao terminar o ADC12 é desabilitado (ADC12ENC = 0). Porém, ser for em um dos modos com repetição, ao término do que foi programado, o ADC12 permanece habilitado (ADC12ENC = 1) e tudo se repete.

Existe uma grande quantidade de detalhes que são omitidos nesta pequena explicação. Em especial é recomendado o tópico 11.14 que detalha os diversos modos de operação do ADC12. É fortemente recomendado o estudo de todo este capítulo.

Para finalizar, indicamos as principais perguntas que devem ser feitas ao se usar um conversor AD.

- Quais serão as entradas analógicas?
- Qual a faixa de excursão de cada sinal (valores máximo e mínimo)?
- Que tensões de referência usar?
- Qual a frequência de amostragem?
- Quanto tempo para amostrar (S/H) e quanto tempo para conversão?
- Quantos bits deve ter a conversão?
- Qual será o valor do relógio ADC12CLK?
- Que modo usar (um canal ou sequência de canais, com ou sem repetição)?

11.1. Introdução aos Conversores Analógico/Digital

Uma boa forma de se entender a conversão analógica/digital, é analisando a evolução de nossos equipamentos eletrônicos. Nas últimas décadas vimos uma grande migração dos sinais analógicos para os digitais. Há algum tempo usávamos sinais analógicos no disco de vinil, no gravador de som, na TV e no videocassete. Isto significa que estes equipamentos trabalhavam ou gravavam a forma de onda do sinal analógico, que era reconstruída no instante da reprodução. Hoje em dia, temos todos esses equipamentos trabalhando de forma digital e, de alguma maneira, a partir dos sinais digitais somos capazes de reconstruir o sinal analógico original. As maiores vantagens dessa migração para o mundo digital são a facilidade de armazenamento e a reprodução. Os conversores ADC e DAC têm uma contribuição importante neste ambiente digital.

Para mostrarmos o papel dos conversores ADC e DAC, começamos com um sinal analógico qualquer, como o mostrado na Figura 11.1.a. Partindo deste sinal, vamos

abordar todas as etapas necessárias para representá-lo sob a forma digital e depois estudar sua reconstrução. Inicialmente, será necessário tomar valores em momentos (ou instantes de tempo) muito bem definidos. Para se marcar esses momentos, usa-se uma grande periódica no tempo, como mostrado na Figura 11.1.b. Essa grade periódica pode ser, por exemplo, de 1 kHz. Isto significa que tomaremos 1.000 valores por segundo, em outras palavras, um valor a cada 0,001 segundo (1 ms).

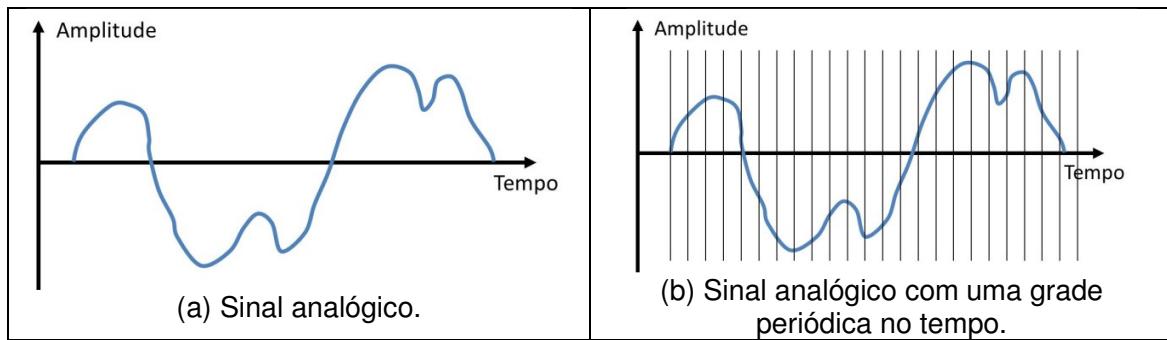


Figura 11.1. Sinal analógico e sinal analógico com grade periódica.

Na Figura 11.2.a vemos marcadas as intersecções do sinal analógico com a grade periódica. Os valores dessas intersecções é que serão considerados. Na Figura 11.2.b esses valores são marcados por barras verticais com um pequeno círculo na extremidade. É uma forma simples de indicar o sinal que foi discretizado no tempo.

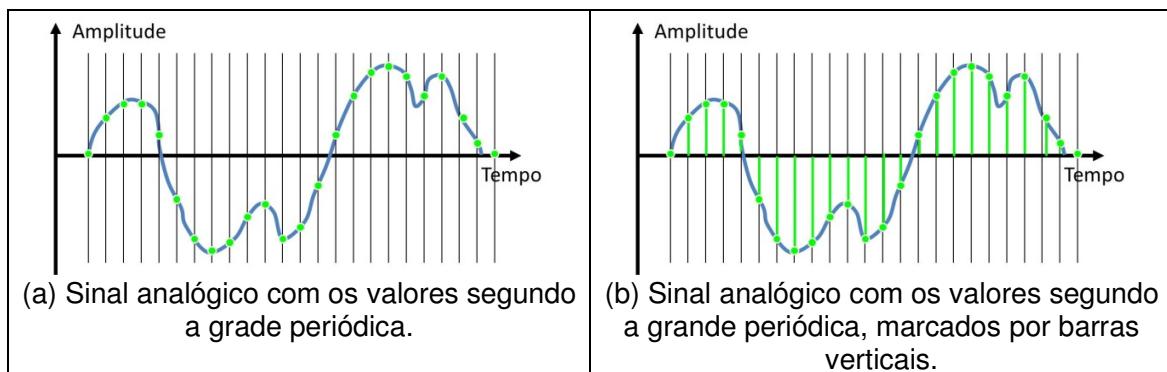


Figura 11.2. Sinal analógico discretizado no tempo.

A Figura 11.3 apresenta apenas o sinal que é o resultado da intersecção da grade periódica com o sinal analógico. Em outras palavras, em cada instante, ditado pela grade temporal, fez-se uma amostragem do sinal analógico original. Este sinal recebe o nome de **Sinal em Tempo Discreto**. Temos agora um sinal que só existe em instantes de tempo bem definidos. Porém, a amplitude pode ter qualquer valor, pois foi originada a

partir de um sinal analógico. Precisamos então comparar as amplitudes com um padrão de medição. Este processo recebe o nome de quantização.

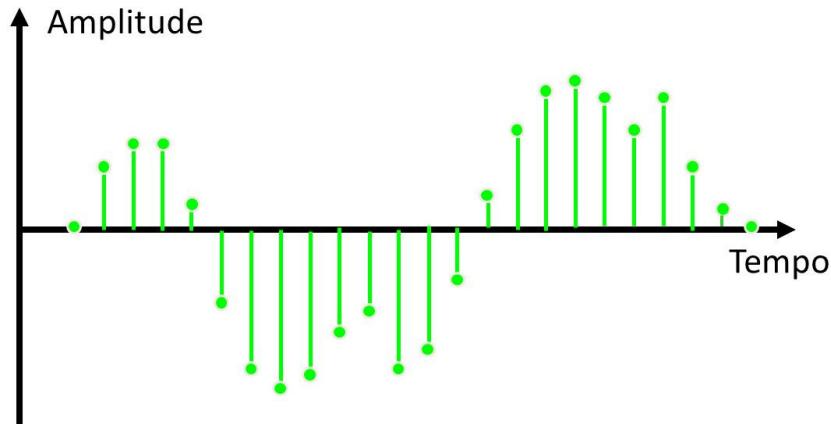


Figura 11.3. Sinal em tempo discreto, resultado da “amostragem” do sinal analógico em momentos definidos por uma grade periódica no tempo.

Por simplicidade, mesmo parecendo estranho (facilita a explicação), vamos trabalhar com uma quantização de 9 níveis: 4 positivos, 4 negativos e o zero. A Figura 11.4 apresenta o processo de quantização do sinal em tempo discreto. A amplitude de cada amostra será representada pelo valor quantizado mais próximo, como mostrado nesta figura. O leitor vai notar que se cometem erros nessa aproximação. É claro que quanto maior for a quantidade de níveis de quantização, mais fiel será a representação dos níveis analógicos, porém os números ficarão maiores, ou seja, aumenta-se a quantidade de bits necessária para a representação.

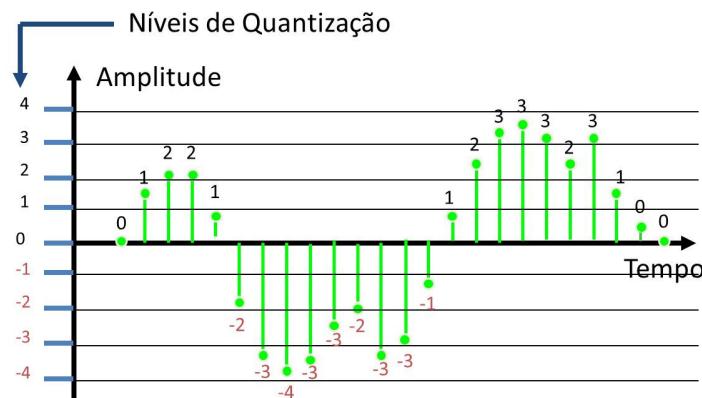


Figura 11.4. Processo de quantização de um sinal em tempo discreto. O sinal analógico original é agora representado pela sequência numérica: 0, 1, 2, 2, 1, -2, -3, -4, -3, -3, -2, -3, -1, 0, 1, 2, 3, 3, 3, 2, 3, 1, 0, 0.

O mais importante agora é que o sinal analógico original ficou representado pela sequência de números: 0, 1, 2, 2, 1, -2, -3, -4, -3, -2, -3, -3, -1, 0, 1, 2, 3, 3, 3, 2, 3, 1, 0, 0. Sua duplicação, armazenamento ou transmissão, não introduz perdas, pois basta com reproduzir, guardar ou transmitir essa sequência de números. É claro que também se precisa informar qual foi a grade temporal e a quantidade de níveis de quantização. O nome dessa grade temporal é taxa de amostragem e a quantidade de níveis de quantização é ditada pelo número de bits usados. Um bom exemplo é o sinal de voz sendo amostrado com uma taxa de 8 kHz, ou seja, 8.000 amostras por segundo e quantizado em 256 níveis (8 bits). Neste caso, cada segundo de voz precisa de 64 kbits. Este é o padrão da telefonia móvel. Outro exemplo é o CD de áudio, que usa amostragem de 16 bits em 44,1 kHz.

O Conversor Analógico-Digital (ADC) desempenha esta função recém-descrita, ou seja, amostra e quantiza sinais analógicos. O instante de cada conversão é ditado pela taxa de amostragem e a quantização pela quantidade de bits.

A reconstrução do sinal analógico segue a ordem inversa. Conhecendo-se a taxa de amostragem (grade periódica), os níveis de quantização e os valores numéricos, basta marcarem-nos na grade, como mostrado na Figura 11.5. É o processo inverso ao da Figura 11.4.

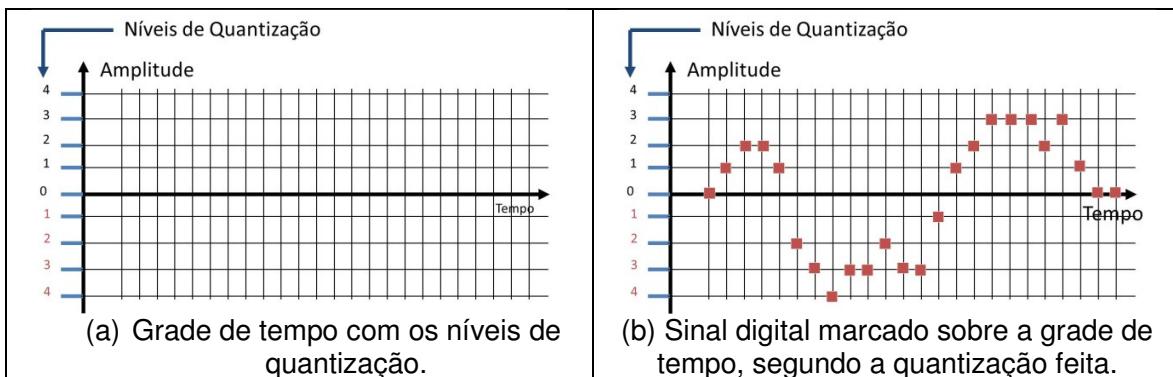


Figura 11.5. Grade periódica com os níveis de quantização usados para “marcar” o sinal digital.

A Figura 11.6 apresenta o sinal reconstruído a partir das informações digitais. O Conversor Digital Analógico (DAC) é responsável por receber o valor digital e gerar a tensão analógica correspondente (sinal analógico em tempo discreto). Um filtro, chamado

de filtro de reconstrução, usualmente um filtro do tipo passa-baixa, é usado para fazer a interpolação entre os diversos pontos do sinal em tempo discreto, resultando assim no sinal analógico reconstruído.

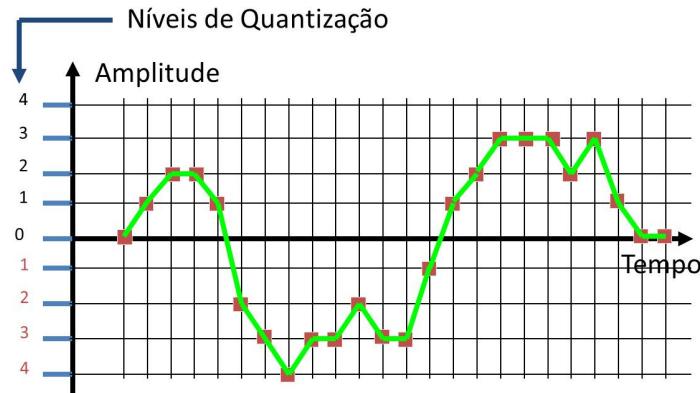


Figura 11.6. Sinal analógico reconstruído a partir do sinal digital.

É claro que o sinal reconstruído não é idêntico ao sinal analógico original, mas espera-se que as diferenças sejam irrelevantes para a finalidade a que se destina. A Figura 11.7 apresenta o sinal analógico original e o sinal reconstruído a partir da representação digital adotada. De forma grosseira, podemos afirmar que quanto maior for a taxa de amostragem e quanto maior for a quantidade de níveis de quantização, mais fiel será a representação do sinal analógico. Por outro lado, o preço a ser pago é a quantidade de dados gerados. Assim, trabalha-se com uma relação de compromisso entre qualidade e quantidade de bits.

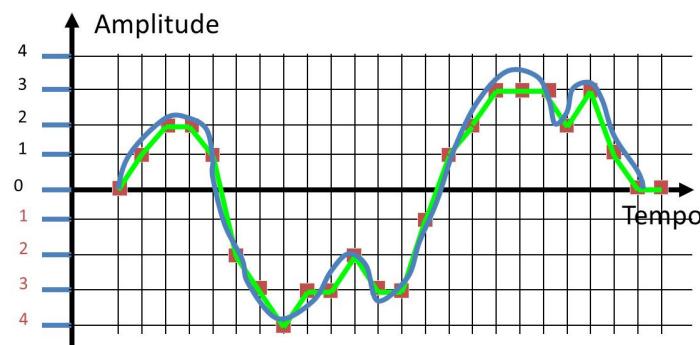


Figura 11.7. Comparação do sinal analógico original com o sinal reconstruído a partir do sinal digital.

Sem querer fugir muito ao tema, mas para deixar um pouco mais completa esta abordagem, vamos falar do Teorema da Amostragem (Teorema de Nyquist). Este teorema diz que a taxa de amostragem de um sinal analógico deve ser maior ou igual ao dobro da componente de maior frequência presente no sinal. Como exemplo, vamos tomar sinais de música, que têm frequências na faixa de 20 Hz até 20 kHz. Segundo esse teorema, para digitalizarmos sinal de música, devemos amostrar numa taxa igual ou superior a 40 kHz.

Uma representação do sinal de voz na faixa de 20 até 4 kHz é bastante inteligível, tanto que é usada por diversos sistemas de telefonia. Neste caso, deveremos digitalizar tal sinal a uma taxa igual ou superior a 8 kHz. Entretanto, fica a pergunta: como garantir que no sinal a ser digitalizado não existam componentes com frequências superiores a 4 kHz? Para garantir isso, todo sistema de digitalização faz uso, na entrada, de um filtro passabaxa para eliminar as altas frequências. Esse filtro recebe o nome de “Filtro Anti-aliasing”. Finalmente, a Figura 11.8 apresenta um sistema típico para processamento digital de sinais analógicos.



Figura 11.8. Típico sistema para processamento digital de sinais analógicos.

Vamos agora estudar um pouco a codificação realizada por um ADC. Consideremos, a título de exemplo, um sinal analógico que excursiona entre 0 V e +5 V e uma quantização de um byte, ou seja, 8 bits. Com 8 bits temos $2^8 = 256$ níveis de quantização. Isto significa que a faixa de 0 V → 5 V será varrida em 256 degraus, sendo a distância entre dois degraus contíguos igual a $5 \text{ V} / 256 = 19,53 \text{ mV}$. À tensão 0 V associamos o binário 0000 0000 (0x00) e ao 5 V o binário 1111 1111 (0xFF), como mostrado na Figura 11.9.a. No caso de usarmos um conversor de 12 bits, serão 2048 ($2^{12} = 2048$) degraus, e o passo entre dois degraus consecutivos é de $5 \text{ V} / 2048 = 2,44 \text{ mV}$, como mostrado na Figura 11.9.b.

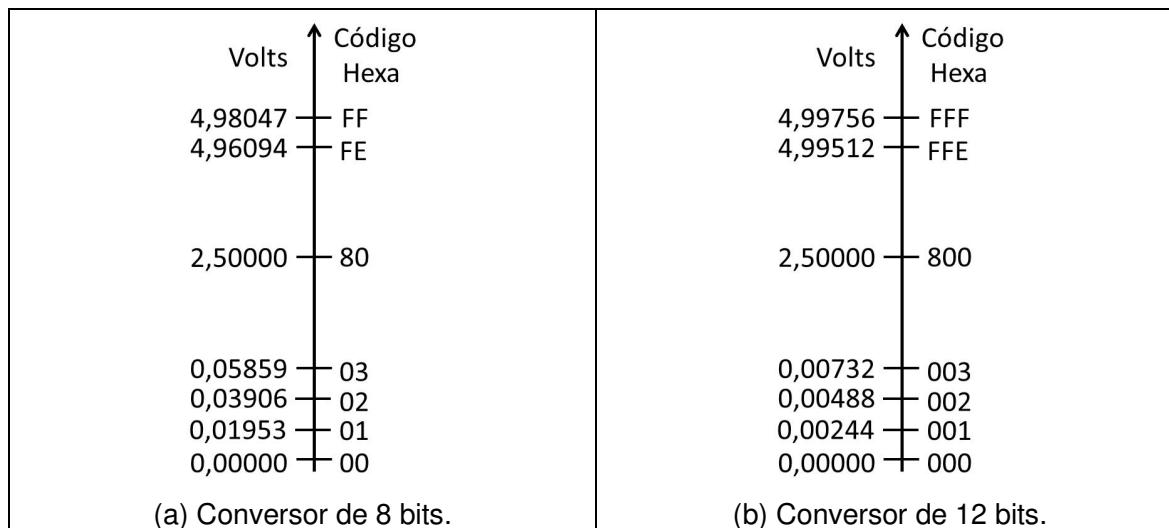


Figura 11.9. Dois casos de conversor operando na faixa de 0 a 5 V, à esquerda com resolução de 8 bits e à direita com resolução de 12 bits. Os passos estão fora de escala.

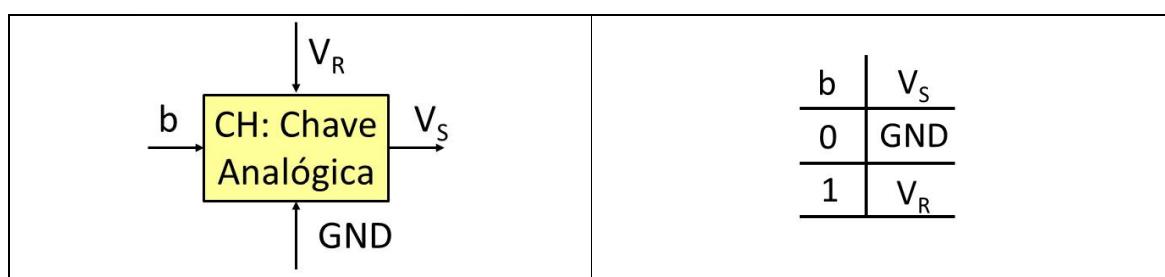
Em resumo, um ADC de n bits converte a faixa entre 0 V (GND) e a tensão de referência (V_{REF}) em 2^n passos. No caso da Figura 11.9, tínhamos $V_{REF} = 5$ V. O código mais baixo é o zero e o código mais alto é o $2^n - 1$.

Após essa pequena noção, vamos abordar o funcionamento do ADC e do DAC. Iniciamos com o DAC (conversor digital-analógico), por ser mais simples. Para explicarmos o funcionamento do DAC, vamos precisar de uma chave analógica, como a mostrada na Figura 11.10. Esta chave disponibiliza na sua saída uma das duas tensões de entradas (V_R ou GND), em função do estado da entrada de controle, aqui denominada de “ b ”.

- Se $b = 0 \rightarrow V_S = \text{GND}$ (0 V).
- Se $b = 1 \rightarrow V_S = V_R$ (Tensão de referência).

De forma simples, podemos escrever a saída da chave CH em função de b :

$$V_S = b \cdot V_R$$



(a) CH: chave analógica.

(b) Tabela da chave analógica.

Figura 11.10. Chave analógica, com duas entradas (V_R e GND) e uma saída (V_S), controlada pela entrada “b”.

Um conversor DA muito simples é o de malha resistiva ponderado, mostrado na Figura 11.11. Temos aí um DAC de 4 bits, sendo esses bits numerados como b3, b2, b1 e b0. Note o uso das chaves “CH”, mostradas na Figura 11.10, cada uma controlada por um bit. Por simplicidade, ao desenhar as chaves omitimos as conexões V_R e GND. Neste conversor, cada resistor tem o dobro do valor do anterior, por isso ele é chamado de malha resistiva ponderada. O amplificador operacional está ligado na configuração de somador de correntes, com realimentação igual a $R/2$.

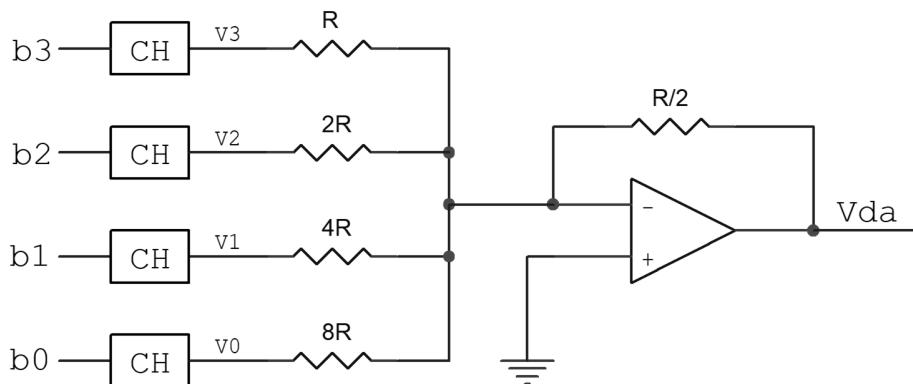


Figura 11.11. Conversor Digital Analógico (DAC), construído com malha resistiva ponderada.

A equação do circuito da Figura 11.11 é dada por:

$$\frac{V_{da}}{R/2} = - \left(\frac{V_3}{R} + \frac{V_2}{2R} + \frac{V_1}{4R} + \frac{V_0}{8R} \right)$$

Explicitando a saída V_{da} chegamos à seguinte equação, onde se conclui que o valor de R não é relevante para esta equação.

$$2V_{da} = - \left(V_3 + \frac{V_2}{2} + \frac{V_1}{4} + \frac{V_0}{8} \right)$$

É de se notar que foi feito um somador ponderado. Se colocarmos cada tensão em função de seu bit (b_3 , b_2 , b_1 ou b_0), ficamos com as seguintes equações:

$$2V_{da} = -\left(b_3 \cdot V_R + \frac{b_2 \cdot V_R}{2} + \frac{b_1 \cdot V_R}{4} + \frac{b_0 \cdot V_R}{8}\right)$$

$$2V_{da} = -V_R \left(b_3 + \frac{b_2}{2} + \frac{b_1}{4} + \frac{b_0}{8}\right)$$

$$V_{da} = -\frac{V_R}{16} (8b_3 + 4b_2 + 2b_1 + b_0)$$

A equação logo acima mostra de forma clara como é feita a ponderação da tensão V_R em função de sua posição na palavra binária. Apresentamos a seguir as tensões de saída (V_{da}) correspondentes a alguns valores binários:

- $b_3b_2b_1b_0 = 0000 \rightarrow V_{da} = 0$
- $b_3b_2b_1b_0 = 0001 \rightarrow V_{da} = -V_R/16$
- $b_3b_2b_1b_0 = 0010 \rightarrow V_{da} = -2V_R/16$
- $b_3b_2b_1b_0 = 0011 \rightarrow V_{da} = -3V_R/16$
- ...
- $b_3b_2b_1b_0 = 1111 \rightarrow V_{da} = -15V_R/16$

Esta forma de se construir um DAC não é prática, pois demanda por resistores com valores difíceis de serem encontrados no mercado. É possível construir esta malha ponderada usando apenas resistores de dois valores. É a chamada rede R/2R. Mas isto foge ao objetivo deste capítulo. A verdade é que existe uma grande variedade de técnicas para se construir um DAC. O leitor está convidado a fazer uma busca na Internet.

Passemos agora ao nosso principal objetivo, que é o estudo dos conversores analógico/digital. Vamos propor a seguir duas soluções simplificadas para se construir um ADC. Elas serão feitas a partir de um DAC, um contador e um comparador. Antes de iniciarmos, vamos apresentar um comparador de tensões elementar, mostrado na figura abaixo. Ele será necessário na construção do ADC. Note que sua saída é 0 ou 1, em função do resultado da comparação entre $V+$ e $V-$.

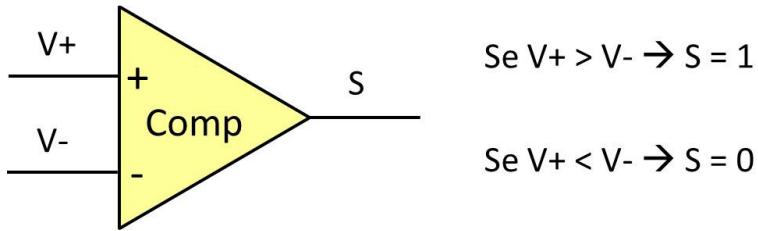
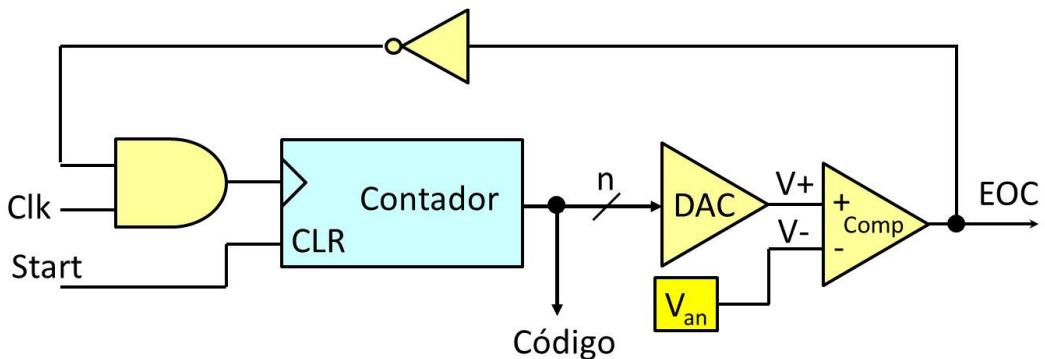


Figura 11.12. Comparador de tensões.

A Figura 11.13 apresenta um esquema muito simples para um ADC. Note que a tensão analógica a ser convertida (V_{an}) é apresentada na entrada V_- do comparador. Temos ainda um DAC cuja entrada digital vem de um contador de n bits. Quando este contador é zerado, a tensão na saída do DAC é zero. O início da conversão é dado pela ativação do sinal *Start* que zera o contador através do seu CLR (clear). Nesse momento, a saída do DAC deve ser zero. Se considerarmos que $V_{an} > 0$, vamos ter na entrada do comparador $V_+ < V_-$, o que significa que sua saída vai estar em 0. Sob essa condição a porta “E” permite a passagem do sinal de relógio (CLK) e o contador vai contando; assim a saída do DAC vai aumentando de valor. O momento interessante é quando a tensão na saída do DAC (que é resultante da contagem) ultrapassa o valor de V_{an} . Quando isto acontece, a saída do comparador vai para 1 e através do inversor e da porta “E”, bloqueia a chegada do relógio ao contador, ou seja, o contador para de contar. Neste momento terminou a conversão e o valor do contador é o resultado da conversão da tensão V_{an} . Note que a saída do comparador foi denominada EOC (em inglês, *End Of Conversion*).

Figura 11.13. Conversor AD de n bits, usando um contador, um DAC e um comparador.

O emprego é muito simples. Apresenta-se a tensão analógica a ser convertida (V_{an}) e aciona-se o *Start*. Com isso, temos $EOC = 0$ e o contador vai contando. Quando EOC for para 1, pode-se ler no contador o valor da conversão. O caso de $V_{an} = 0$ não é problemático, pois, o contador não chega a contar e a saída EOC vai imediatamente para o valor 1.

O erro na conversão só será próximo de zero se a tensão de saída do DAC for próxima, mas superior à tensão V_{an} . Isto deve acontecer poucas vezes, por isso, a conversão do ADC tem erro e ele é menor que 1 bit. Uma solução é deslocar a tensão na saída do DAC de um valor de tensão correspondente a $\frac{1}{2}$ bit. Assim, o erro máximo do ADC passa a ser de $+/- \frac{1}{2}$ bit.

A resolução deste tipo de ADC é ditada pelo número de bits de seu contador. Quanto maior o número de bits, maior a resolução. A grande desvantagem é o tempo necessário para a contagem. No pior caso, um ADC de 8 bits vai precisar de 256 contagens. Já um ADC de 12 bits vai precisar, no pior caso, de 2048 contagens. Uma forma de diminuir essa grande quantidade de contagem é trabalhar com um Registrador de Aproximações Sucessivas (SAR = *Successive Approximation Register*), que está mostrado na Figura 11.14. Um ADC construído com um SAR precisa de uma quantidade de ciclos bem menor para completar a conversão.

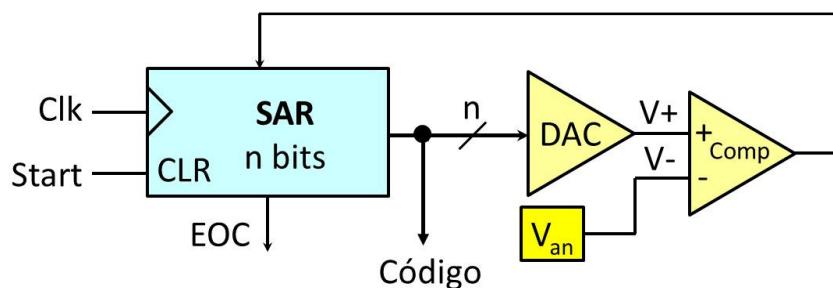


Figura 11.14. ADC com registrador de aproximações sucessivas.

Para entender o funcionamento do SAR e vamos adotar aqui a explicação apresentada por Taub & Schilling no seu livro “Sistemas Digitais”. Eles compararam o SAR ao processo de pesagem com uma balança de dois pratos, como mostrado na Figura 11.15. No prato da esquerda temos um objeto a ser pesado, de peso igual ou menor a 1 kg e no prato da direita vamos colocar os pesos padrão. Temos pesos padrão de diversos valores:

$$\begin{array}{llll}
 P7 = 1/2 \text{ kg}, & P6 = 1/4 \text{ kg}, & P5 = 1/8 \text{ kg}, & P4 = 1/16 \text{ kg} \\
 P3 = 1/32 \text{ kg}, & P2 = 1/64 \text{ kg}, & P1 = 1/128 \text{ kg} & P0 = 1/256 \text{ kg}
 \end{array}$$

O algoritmo para efetuarmos a pesagem é simples. Começamos colocando P7 no prato. Se a balança pender para o lado dos pesos, é porque ele é grande e deve ser removido. Se a balança não se mover, ele deve ser mantido. Em seguida, repetimos o processo

para P6 e assim por diante até testarmos todos os pesos. Toda vez que um peso se mostrar excessivo, ele é removido. Ao final da pesagem, ao somar os pesos que ficaram no prato, teremos uma aproximação do peso do objeto. Por exemplo, digamos que os pesos que ficaram no prato foram: P7, P4 e P3, como mostrado na Figura 11.15. Podemos representar isso com a seguinte sequência: 1001 1000, onde cada bit indica se o peso correspondente permaneceu na balança. Neste caso, o peso aproximado do objeto é dado por:

$$\text{Peso} = 1\frac{1}{2} + 0\frac{1}{4} + 0\frac{1}{8} + 1\frac{1}{16} + 1\frac{1}{32} + 0\frac{1}{64} + 0\frac{1}{128} + 0\frac{1}{256} = 0,59375 \text{ kg}$$

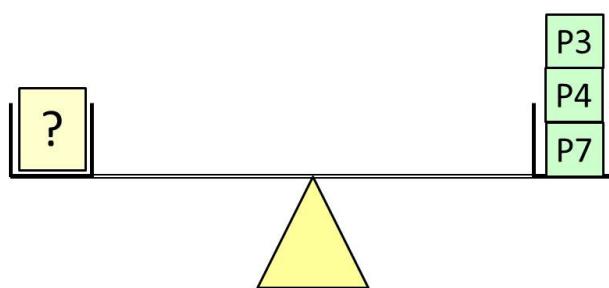


Figura 11.15. Balança com dois pratos para pesagem de um objeto colocado no prato da esquerda.

O SAR trabalha de forma semelhante, só que a grandeza agora é a tensão. Ele começa como os bits do SAR zerados. Isto significa que a saída do DAC é zero. Para iniciar o processo, o bit mais à esquerda do SAR é colocado em 1. Se a saída do DAC ficar maior que a tensão V_{an} , este bit volta a ser zerado, caso contrário, ele é mantido em 1. O processo é repetido, da esquerda para a direita, com todos os bits do SAR. Ao final teremos no SAR o código correspondente à tensão V_{an} . Assim, um conversor de 8 bits vai precisar apenas de 8 ciclos. Um de 12 bits precisará de 12 ciclos. Em suma, ficaram bem mais rápidos.

Em seguida, precisamos abordar o circuito amostrador/retentor (em inglês, *sample/holder*). É fácil entender sua finalidade. Já vimos que o ADC gasta algum tempo para fazer sua conversão. É óbvio que o sinal analógico na entrada do ADC deve permanecer estático (constante) enquanto acontece essa conversão. Se a entrada analógica ficar mudando de valor, o ADC dará um resultado errado. Assim, precisamos de um circuito que sustente o valor da entrada enquanto acontece a conversão. A Figura 11.16 apresenta um circuito amostrador, cujo funcionamento é muito simples.

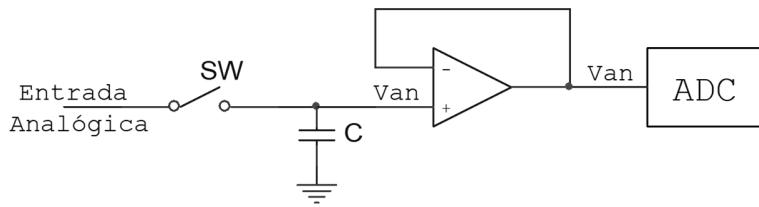


Figura 11.16. Esquema de um amostrador/sustentador (sample/holder) para estabilizar o sinal a ser convertido pelo ADC.

O amplificado operacional está no modo seguidor, ou seja, a saída é igual à entrada. Note a presença da chave da SW (comandada eletronicamente) e do capacitor C. Enquanto a chave está fechada, a tensão no capacitor acompanha a entrada analógica. Diz-se então que o circuito está no modo amostrador (*sample*). Quando a chave é aberta, o capacitor armazena e sustenta o último valor analógico. Nesta condição, o circuito está no modo sustentação (*hold*). Como a entrada do amplificador operacional consome pouca corrente, o capacitor sustenta a tensão durante o tempo necessário para a conversão do ADC. O valor do capacitor C está sujeito a uma relação de compromisso. Quanto maior for o valor de C, por mais tempo ele vai sustentar a tensão analógica, entretanto, mais tempo vai demorar em se carregar com a tensão da entrada. Um capacitor pequeno amostra mais rápido, porém, sustenta a tensão por um tempo menor.

MELHORAR A EXPLICAÇÃO DOS PRÓXIMOS PARÁGRAFOS!

Para finalizar este pequeno estudo, vamos agora abordar alguns detalhes importantes. Por simplicidade, vamos considerar um conversor ADC de 3 bits, trabalhando na faixa de 0,0 V até 3 V. Com 3 bits, temos 8 faixas, cada uma com um valor de 0,375 V (3,0/8 V), como ilustrado na Tabela 11.1. A última linha desta tabela, apresenta a faixa a 7/8 a 8/8 e que corresponde à faixa de tensões de 3,000 V a 3,375. Esse valor de 3,375 V é estranho por que o valor máximo deveria ser de 3,000 V. Parece-nos que essa faixa deveria ficar limitada em 3,000 V. Isso acontece porque o código binário corresponde ao limite esquerdo da faixa, ou seja, 011 corresponde à faixa de 3/8 a 4/8.

Tabela 11.1. Resumo de um ADC de 3 bits com Fundo de Escala (FS) igual a 3V.

Faixa	Volts (FS = 3,0 V)	Binário
0/8 a 1/8	0,000 a 0,375	000
1/8 a 2/8	0,375 a 1,125	001
2/8 a 3/8	1,125 a 1,500	010
3/8 a 4/8	1,500 a 1,875	011
4/8 a 5/8	1,875 a 2,250	100
5/8 a 6/8	2,250 a 2,625	101
6/8 a 7/8	2,625 a 3,000	110

7/8 a 8/8	3,000 a 3,375	111
-----------	---------------	-----

A Figura 11.17.a apresenta de forma gráfica o conteúdo da Tabela 11.1. Cada pequeno traço horizontal indica a faixa de tensão que resulta num mesmo código. Temos, então, uma pequena escada com 8 degraus. Note que ao binário 111 corresponde a faixa de 3,000 V a 3,375 V. Uma outra deficiência deste esquema é que os passos de tensão ($1/8$, $2/8$, $3/8$, ...) estão nas extremidades de cada faixa binária. Explicando melhor: a tensão de $3/8$ V ($1,500$ V) pertence às faixas binárias de 010 e 011. Neste caso, pode-se dizer que a precisão é de ± 1 bit.

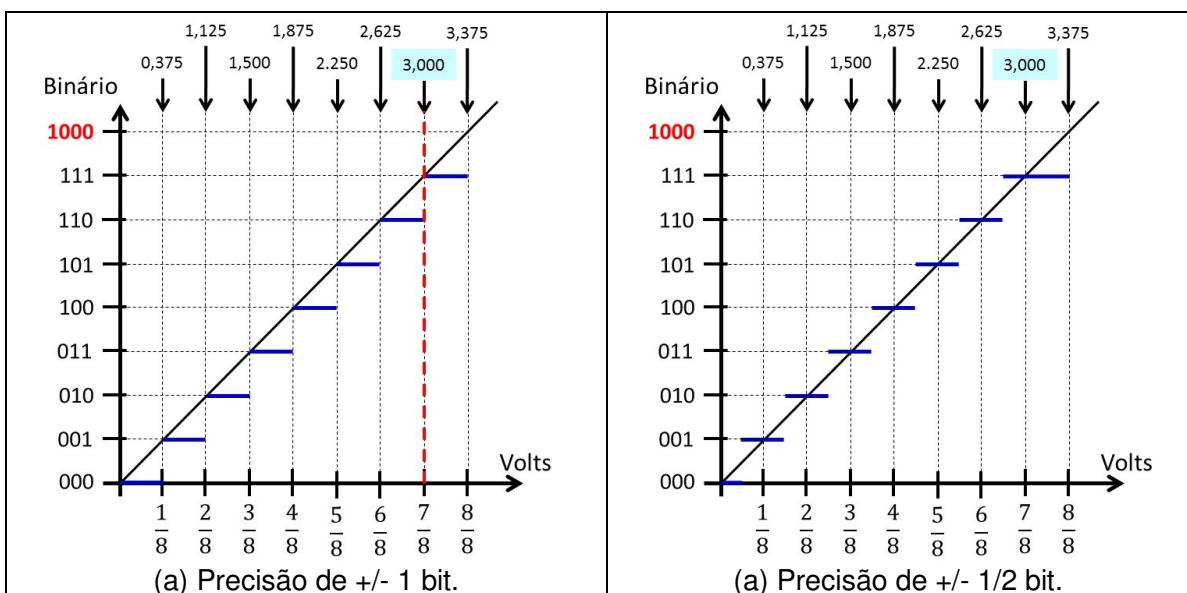


Figura 11.17. Gráfico correspondente ao AC de 3 bits apresentado na Tabela 11.1.

O ideal é que cada passo de tensão estivesse no centro de cada patamar binário. Isso é conseguido ao se deslocar a escada para a esquerda, como mostrado na Figura 11.17.b. Ao se fazer isso, o primeiro degrau da escada (binário 000) ficou com metade do tamanho e o último degrau (binário 111) está com tamanho aumentado em 50%. O interessante é que agora cada código binário oferece precisão de $\pm 1/2$ bit (metade da faixa de tensão equivalente a um bit). Por exemplo, qualquer valor de tensão na faixa de $(3/8 - 3/16)$ V até $(3/8 + 3/16)$ Volts corresponde ao valor binário 011.

Abordamos agora as imperfeições mais comum dos ADC. O termo Não-Linearidade Diferencial faz referência ao tamanho dos degraus da escada, como mostrado na Figura 11.18.a. Note que os degraus com os valores binários 010, 011 e 100 não têm o mesmo tamanho, ou seja, os passos não têm o mesmo valor em Volts. Já o Código Faltando

(Missing Code) representa a situação onde o ADC “pula” um determinado degrau. Na Figura 11.18.b não existe o código binário 011.

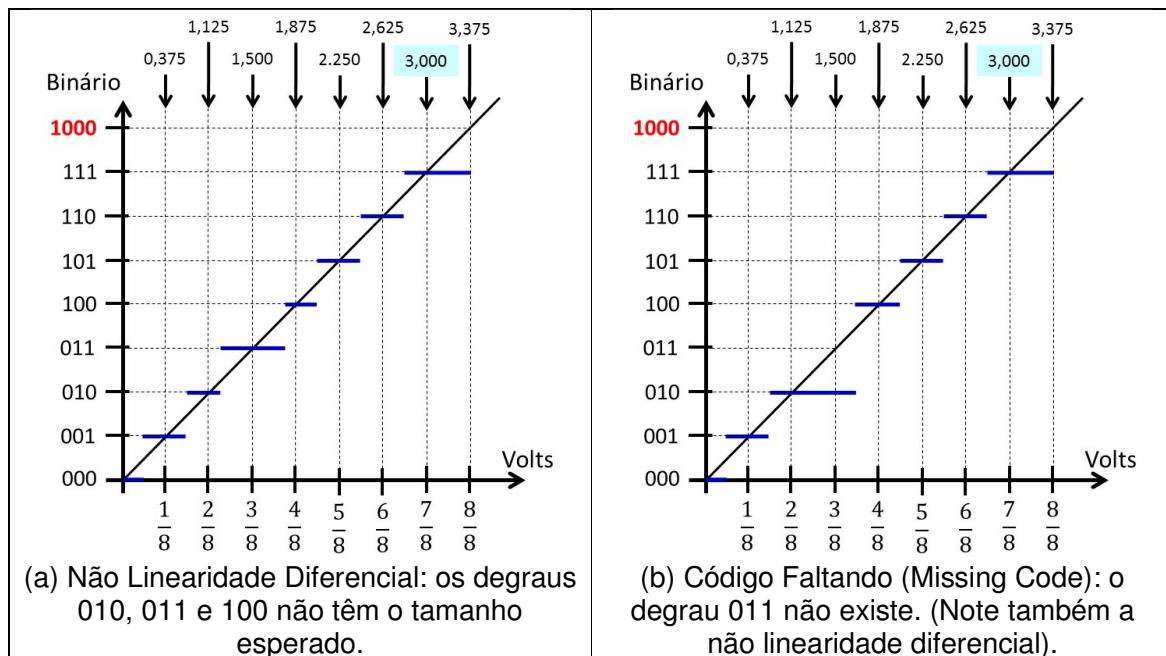


Figura 11.18. Duas possíveis imperfeições dos ADCs.

Para explicar as duas próximas imperfeições, vamos aproximar a escada do ADC por uma reta. Na Figura 11.19.a temos ilustrado o erro de *Offset*. Note que neste exemplo, a escada do ADC ficou deslocada para a direita, ou seja, código zero não corresponde mais ao 0 Volts. Já na Figura 11.19.b nota-se que a escada tem uma ascendente mais rápida do que serial o ideal.

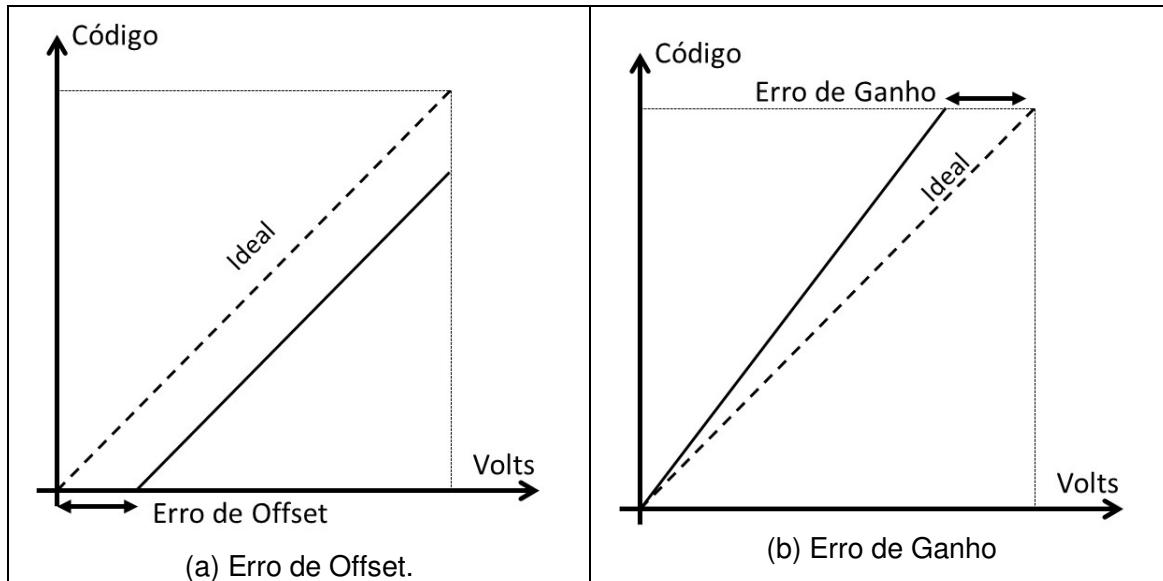


Figura 11.19. Duas possíveis imperfeições dos ADCs: Erro de Offset e Erro de Ganho.

Em geral, não é difícil corrigir os erros de offset e ganho de um ADC. Entretanto, é comum que após essas correções a escala do ADC não fique idêntica à ideal. De forma simples, após essas correções, ao maior desvio vertical em relação ao caso ideal dá-se o nome de Não-Linearidade Integral, ou INL (do inglês *Integral Non-Linearity*). Isto está ilustrado na Figura 11.20.

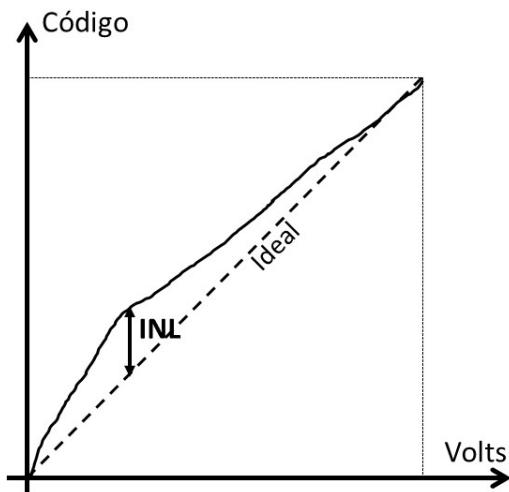


Figura 11.20. Ilustração da INL (Integral Non-Linearity). Duas possíveis imperfeições dos ADCs: Erro de Offset e Erro de Ganho.

Finalizamos aqui este breve estudo de conversores DAC e ADC. Lembramos que existem diversos tipos de conversores A/D. Citamos alguns: rampa, Wilkinson, integrador, delta, sigma-delta etc. O leitor está convidado a fazer uma rápida pesquisa sobre esses diversos tipos e suas típicas aplicações. Passemos agora ao estudo dos recursos do ADC12 do MSP430.

11.2. Características do ADC12

Vamos começar o estudo específico do ADC12 do MSP430 F5529 e listamos a seguir suas principais características.

- Taxa de amostragem de até 200 ksps (200.000 amostras por segundo);
- Monotônico (linear) e sem códigos faltantes (*no missing codes*);
- Amostrador/Retentor (*sample/holder*) programáveis;
- Conversão disparada por *timers* ou programa;
- Referência de tensão programada por *software*;
- Disponibilidade de até 12 canais externos;
- Relógio do conversor programável;
- 4 modos de operação: simples (repetido), sequencial (repetido);
- Recursos para economia de energia (*power down*);
- Disponibilidade de 18 interrupções e
- 16 registradores para armazenar as conversões.

Ao longo deste capítulo estudaremos com detalhes todos esses recursos. Agora vamos iniciar um estudo do diagrama de blocos deste conversor.

11.3. Diagrama de Blocos do ADC12

O ADC12 é bastante complexo e por isso, iniciamos com um diagrama de blocos bem simples, apenas 6 blocos numerados, como apresentado na Figura 11.21 e listado abaixo:

- 1) Conversor ADC junto com o Amostrador/Retentor (S/H);
- 2) Disparo da conversão, responsável por gerar o sinal SAMPCON;
- 3) Multiplexador para selecionar uma das 16 possíveis entradas analógicas;
- 4) Relógio para operação do ADC;
- 5) Referências de tensão e
- 6) Memórias para armazenar os resultados das conversões.

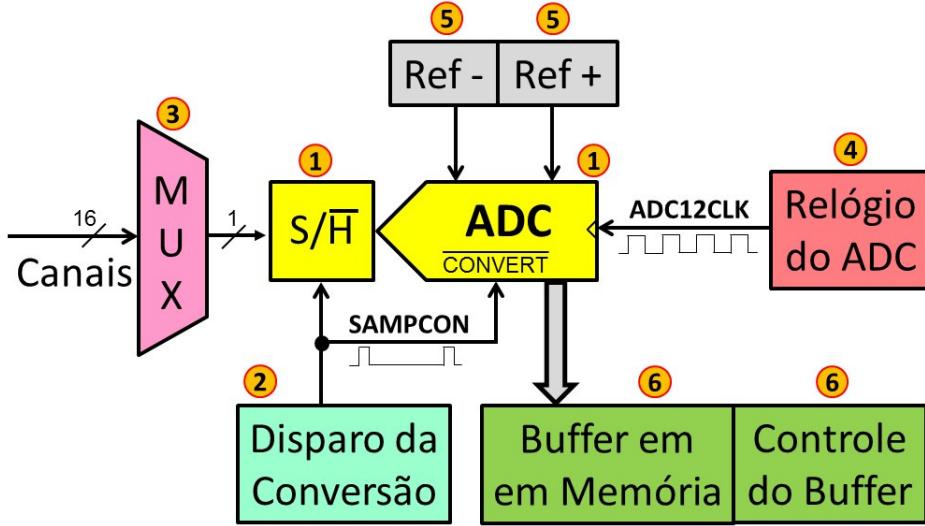


Figura 11.21. Diagrama de blocos simplificado do ADC12.

Antes de entrarmos em detalhes, vamos fazer um estudo simples, usando o diagrama da recém apresentado. A parte mais importante é a operação conjunta do ADC e do S/H, sincronizados pelo sinal SAMPCON (do inglês *Sample/Convert*). Note que a entrada CONVERT do ADC é habilitada em nível baixo, isto significa que a conversão é iniciada com SAMPCON = 0. Por sua vez, o bloco S/H está no modo amostragem, enquanto SAMPCON = 1 e entra no modo retenção SAMPCON = 0. Este esquema garante que o ADC faça as conversões enquanto o S/H está no modo retenção. Em outras palavras, comandados pelo SAMPCON, o ADC fica parado enquanto o S/H está no modo amostragem. O ADC inicia a conversão quando o S/H entra no modo de retenção. A Figura 11.22 ilustra esta coordenação entre o ADC e o S/H feita pelo sinal SAMPCON, onde se nota que o período de amostragem é determinado pelo tempo em que SAMPCON fica em nível alto.

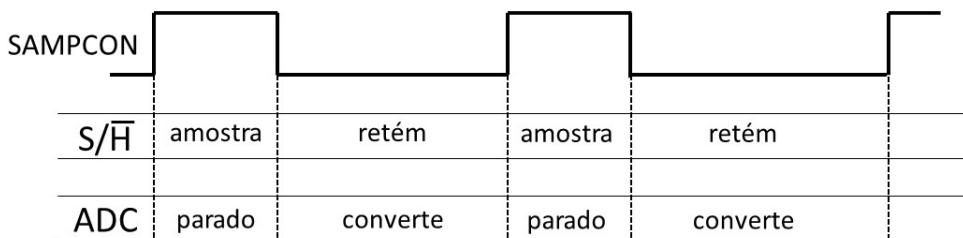


Figura 11.22. Instantes de amostragem e conversão, comandados pelo sinal SAMPCON.

Voltando à Figura 11.21, o bloco de número 2, denominado de Disparo da Conversão, é o responsável por gerar o sinal SAMPCON. O MSP oferece diversos recursos para a

configuração deste bloco. É interessante comentar que a frequência de amostragem é ditada pela frequência do sinal SAMPCON. O período de amostragem é ditado pelo intervalo de tempo em que SAMPCON fica em nível alto e o período de conversão pelo intervalo de tempo em que SAMPCON fica em nível baixo. O programador especifica esses dois parâmetros em função de sua necessidade.

O recurso de conversão AD é um só e está compartilhado entre 16 entradas com o uso de um multiplexador analógico (bloco 3). Desses 16 entradas, 12 estão disponíveis externamente.

Como o ADC é uma máquina de estados digital, para sua operação é preciso definir um relógio, o que é feito pelo bloco 4, que gera o sinal ADC12CLK. Tomando como base o ACLK ou SMCLK, o usuário programa esse relógio, que deve ser superior à frequência de amostragem. Porém, quanto maior for a frequência deste sinal, maior o consumo de energia.

O bloco 5 representa as duas referências para o ADC. Note que o usuário tem a liberdade de definir referências positivas e negativas, um recurso não muito comum nos ADCs comerciais. Para terminar, o bloco 6 representa as 16 posições de memória que funcionam como *buffer* para armazenar o resultado das conversões. A forma como acontece as conversões e o armazenamento é programada no bloco de controle.

Finalizamos este tópico indicando as principais perguntas a ser fazer quando se usa um ADC. Note que cada pergunta diz respeito a um bloco da Figura 11.21.

- 1) Qual modo de operação do ADC?
- 2) Qual a taxa de conversão?
- 3) Qual a entrada analógica?
- 4) Que relógio programar no ADC?
- 5) Quais referências usar?
- 6) Como armazenar os resultados?

11.4. Seleção da Entrada Analógica do ADC12

Como já foi afirmado, o MSP oferece um ADC compartilhado entre 16 entradas analógicas. Desses 16 entradas, 12 estão disponíveis externamente ao usuário e 4 estão dedicadas a monitorar parâmetros internos. A Figura 11.23 apresenta um diagrama deste multiplexador. Lembramos da convenção usada neste texto, onde os recursos que o usuário pode programar estão indicados por retângulos verdes, com cantos arredondados. As *flags* que o usuário monitora estão marcadas em amarelo canário.

Na Figura citada, vê-se claramente as 16 entradas, denominadas de A0, A1, ..., A15. A seleção da entrada a ser usada é feita pelos 4 bits denominados de INCHx. As 4 entradas destinadas a parâmetros internos têm as seguintes finalidades:

- A8 → Tensão da referência positiva externa (V_{eREF+});
 - A9 → Tensão da referência negativa interna ou externa (V_{REF-}/V_{eREF-});
 - A10 → Tensão fornecida pelo sensor de temperatura e
 - A11 → Metade da tensão de alimentação analógica (AV_{CC}).

O sensor de temperatura informa a temperatura interna do MSP430 e pode ser usado para monitorar superaquecimento do *chip* ([está correto?](#)). Por motivos de economia de energia, o sensor de temperatura pode ser desabilitado com o bit ADC12TCOFF. Quando o usuário seleciona a entrada A11 (INCHx = 11), é fechada a chave que alimenta o divisor resistivo. Note que esse divisor é formado por dois resistores de mesmo valor, assim a tensão na entrada A11 deve ser $AV_{CC}/2$. Esta entrada serve para verificar o correto funcionamento do ADC12 ([é isso mesmo?](#)).

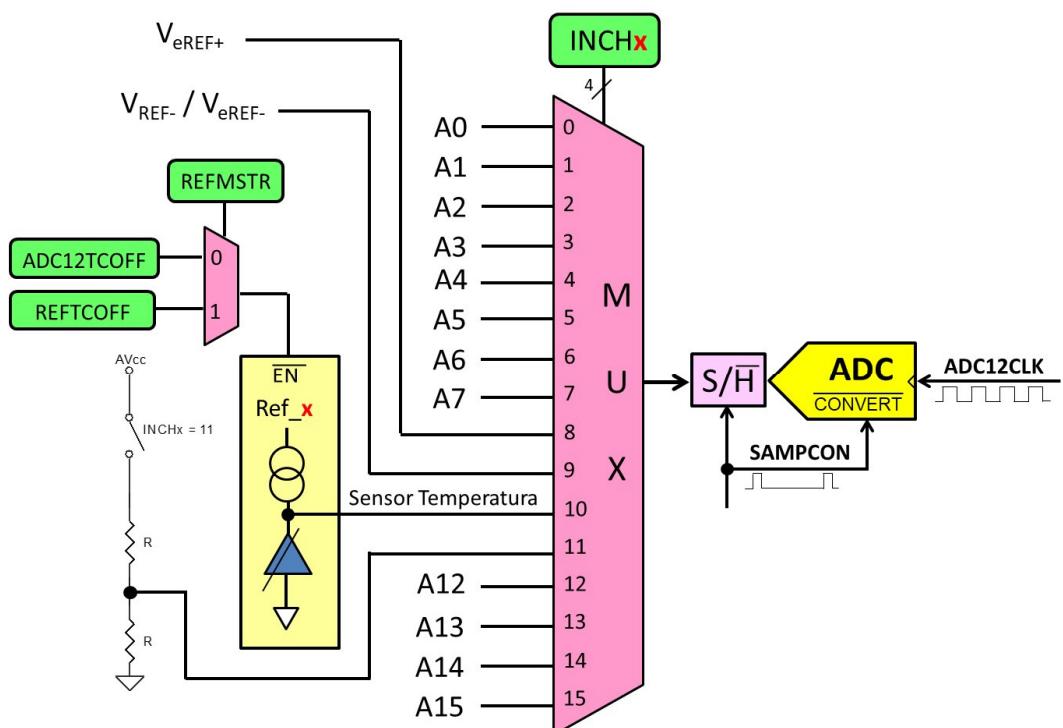


Figura 11.23. Diagrama da seleção das entradas do ADC.

A Tabela 11.2 relaciona as entradas analógicas e os pinos correspondentes. As entradas de A8 até A11 estão dedicadas a recursos internos e por isso não têm correspondência com as portas. As entradas que não estão disponíveis na LaunchPad estão marcadas em cinza-claro.

As entradas analógicas estão multiplexadas com os pinos digitais. Um sinal analógico aplicado num determinado pino pode levar a um desperdício de energia, especialmente se esse sinal analógico estiver próximo do nível de transição da porta digital, que fica desnecessariamente chaveando. Por isso, é interessante desabilitar essa porção digital da porta, o que é conseguido com o registrador PxSEL. Por exemplo, para se usar a entrada A0 (P6.0) é interessante colocar em nível alto o bit 0 do registrador P6SEL (`P6SEL |= BIT0;`).

Tabela 11.2. Correspondência entre as entradas analógicas (A_0, A_1, \dots, A_{15}) e as portas do MSP430F5529 (em cinza-claro estão os pinos não disponíveis no Launch Pad)

Correspondência entre Entradas Analógicas e Portas do MSP430							
A0	P6.0	A4	P6.4	A8	-	A12	P7.0
A1	P6.1	A5	P6.5	A9	-	A13	P7.1
A2	P6.2	A6	P6.6	A10	-	A14	P7.2
A3	P6.3	A7	P6.7	A11	-	A15	P7.3

11.5. Disparo de uma Conversão do ADC12

É grande a sofisticação para o disparo de conversões no ADC12, como se pode ver na Figura 11.24. No lado esquerdo desta figura e listadas abaixo, estão as 4 fontes de disparo que são selecionadas por um multiplexador controlado pelos bits ADC12SHS.

- ADC12SC → bit controlado pelo programa do usuário;
 - TA0.1 → Timer A0, comparador 1;
 - TB0.0 → Timer B0, comparador 0 e
 - TB0.1 → Timer B0, comparador 1;

O bit ADC12SC (onde SC = *Start Conversion*) oferece a forma mais simples de se disparar uma conversão. Cada vez que o programa o coloca em nível alto, uma conversão é iniciada. Este bit é zerado automaticamente. É claro que a operação deste bit deve prever tempo suficiente para que a conversão possa terminar. **Verificar como fica o período de amostragem se o Sample Timer não for usado (ADC12SHP = 0). O que acontece se for habilitado o inversor (ADC12ISSH = 1)?**

Os sinais TA0.1, TB0.0 e TB0.1 são gerados pelos *timers* TA0 ou TB0, vide Capítulo 7. A forma mais simples é usar o recurso do PWM. Neste caso, a frequência do PWM vai corresponder à frequência de amostragem e o ciclo de carga do PWM (período em nível alto) vai ditar o tempo de amostragem do S/H.

Na figura citada, o bit ADC12ENC é a habilitação do ADC, em outras palavras, só é possível iniciar uma conversão com este bit em nível alto. Seguindo da esquerda para a direita, o bit ADC12ISSH permite inverter o sinal que vai disparar a conversão. Em função de sua necessidade, o programador decide se vai ou não inverter o sinal de disparo. Chegamos agora ao local onde o sinal de disparo está rotulado como SHI.

Este sinal SHI pode se transformar no SAMPCON (ADC12SHP = 0), ou ele pode acionar o bloco *Sample Timer*, que passa a ser o responsável por gerar o sinal SAMPCON (ADC12SHP = 1).

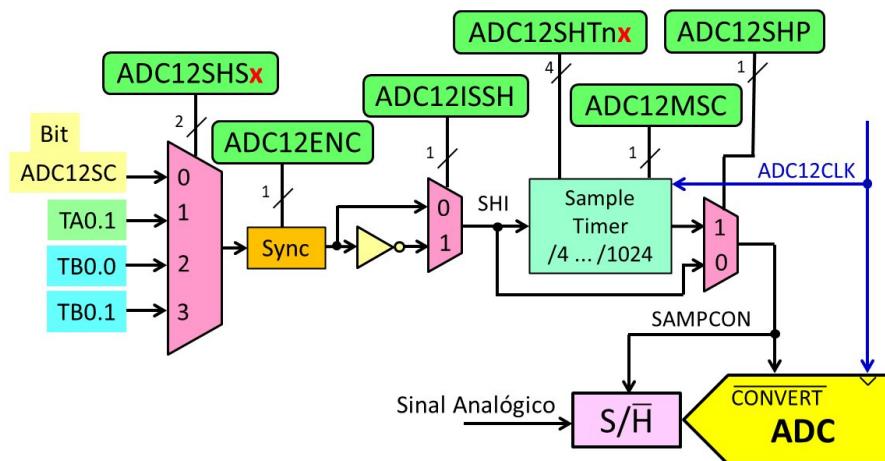


Figura 11.24. Diagrama para disparo das conversões do ADC.

Com o bit ADC12SHP = 0, o sinal SHI se transforma no SAMPCON. Neste caso, o programador deve se preocupar com os intervalos de amostragem e conversão. Como pode ser visto na Figura 11.25, o período em que o sinal SHI (SAMPCON) fica em nível alto, corresponde ao período de amostragem. O período de retenção (S/H) e conversão (ADC12) é ditado pelo tempo em que este sinal permanece em nível baixo.

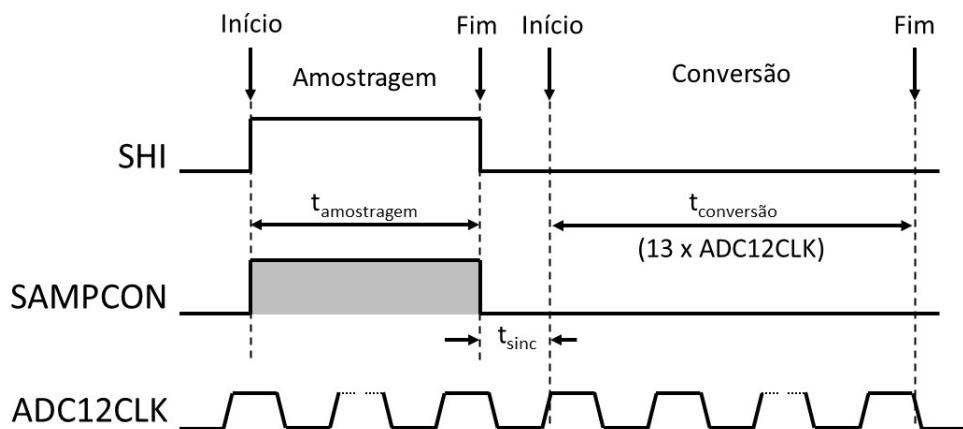


Figura 11.25. Tempo de amostragem especificado pelo sinal SHI. Nesta figura foi considerada a operação em 12 bits, por isso o tempo de conversão equivale a $13 \times ADC12CLK$.

Para facilitar a tarefa do programador, está disponível o circuito *Sample Timer*, onde é possível programar um determinado período de amostragem. Para usá-lo, basta colocar o bit $ADC12SHP = 1$. Neste caso, como pode ser visto na Figura 11.24, o sinal SHI ativa esta unidade *Sample Timer* que gera o sinal SAMPCON. O usuário programa nesta unidade o período de amostragem que necessita. Assim, cada vez que SHI vai para nível alto, a unidade *Sample Timer* gera o sinal SAMPCON de acordo com o especificado. Isto pode ser observado na Figura 11.26.

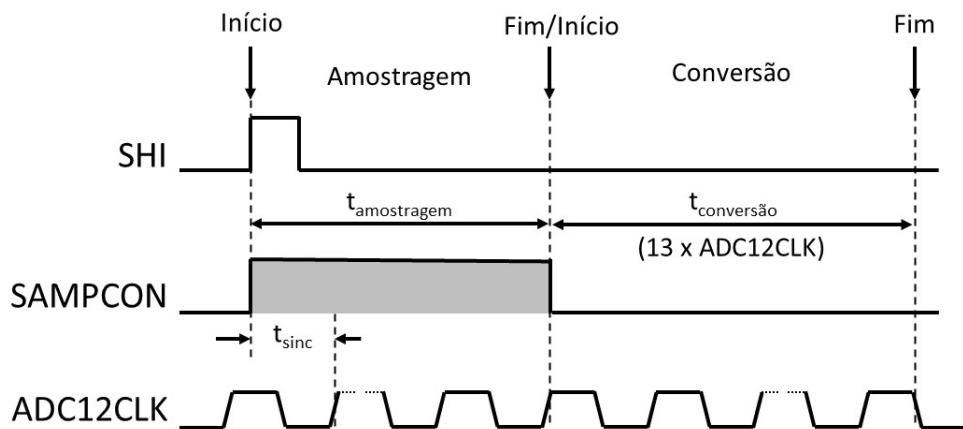


Figura 11.26. Tempo de amostragem programado no bloco Sample Timer, assim o sinal SHI apenas indica o início da amostragem. Nesta figura foi considerada a operação em 12 bits, por isso o tempo de conversão equivale a $13 \times ADC12CLK$.

O cálculo de período de amostragem necessário é ditado pela impedância da fonte do sinal a ser digitalizado. A Figura 11.27 apresenta o circuito equivalente da entrada analógica do MSP. Ele é modelado por um filtro passa-baixa RC. Note que é entregue ao ADC12 a tensão sobre o capacitor (V_c). Então, o que se precisa é que a tensão V_c esteja bastante próxima da tensão V_i . Como nosso ADC tem um erro de $+/- \frac{1}{2}$ bit, o ideal é que essa diferença esteja abaixo desse erro. A expressão abaixo resume este conceito, onde n representa a quantidade de bits do conversor.

$$V_s - V_c < \frac{1}{2} \text{"bit de } V_s \text{"} \quad \text{ou} \quad V_s - V_c < \frac{V_s}{2^{n+1}}$$

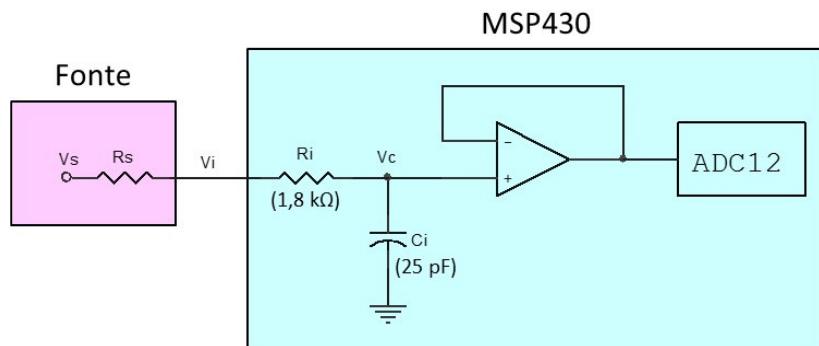


Figura 11.27. Circuito equivalente da entrada analógica do MSP, quando no modo amostragem, mostrando os valores típicos.

A tensão sobre o capacitor está equacionada abaixo, onde $R = R_s + R_i$.

$$V_c = V_s(1 - e^{-t/RCi})$$

Substituindo a expressão de V_c na primeira equação, temos:

$$V_s - V_s(1 - e^{-t/RCi}) < \frac{V_s}{2^{n+1}}$$

Manipulando a equação e substituindo R por $R_s + R_i$, e adicionando o tempo de comutação da chave (800 ns) -verificar tempo da chave- chegamos à equação final:

$$t_{amost} > (R_s + R_i) \cdot C_i \cdot \ln(2^{n+1}) + 800 \text{ ns}$$

Problema: Calcule o valor mínimo do tempo de amostragem considerando que se trabalha com 12 bits e que $R_s = 50 \text{ k}\Omega$. Resposta: 12,47 μs (conferir).

Veja na Figura 11.24 que o Sample Timer toma como referência o ADC12CLK, assim, esse tempo precisa ser especificado com base neste relógio. O exercício resolvido ER 11.x apresenta um exemplo.

É interessante comentar que o tempo de amostragem não pode ser programado livremente para cada canal. Os canais que usam as posições de memória de ADC12MEM0 até ADC12MEM7, compartilham o mesmo tempo de amostragem, da mesma forma que os canais que usam as posições ADC12MEM8 até ADC12MEM15. Isto acontece pela forma como se programa o registrador ADC12CTL0_H, vide [Figura 11.x](#).

11.6. Seleção do Relógio do ADC12 (ADC12CLK)

O ADC12 é uma máquina de estados que por aproximações sucessivas (SAR), fornece o código correspondente à tensão analógica na entrada. Para que possa funcionar, essa máquina de estados precisa de um relógio, que aqui recebe o nome de ADC12CLK. As possibilidades para a seleção deste relógio estão apresentadas na Figura 11.28. Note que o programador pode selecionar a partir de 4 fontes: ADC12OSC, ACLK, MCLK ou SMCLK. O sinal selecionado pode ser dividido por até 32 ($1/4 \times 1/8$), para depois se tornar o ADC12CLK.

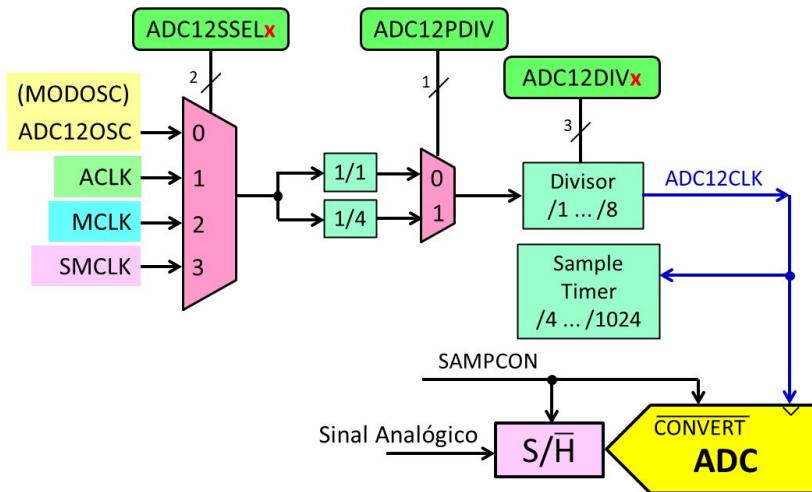


Figura 11.28. Diagrama para a seleção do relógio ADC12CLK.

O parâmetro mais importante para a escolha da frequência do ADC12CLK é o tempo gasto para o ADC realizar a conversão e a taxa de amostragem. Se n (8, 10 ou 12) é a quantidade de bits programada, o ADC12 gasta $(n+1) \cdot T_{ADC12CLK}$. Por exemplo, se for usado $ADC12CLK = 100$ kHz (período de 10 μ s) e 12 bits, serão necessários 130 μ s para cada conversão.

Problema exemplo. Vamos supor que seja necessário digitalizar um sinal com resolução de 12 bits usando uma taxa de 20.000 amostras por segundo (20 ksp) e há a imposição

de se usar o SMCLK (1.048.576 Hz). A impedância de saída do dispositivo que gera esse sinal demandou por um tempo de amostragem superior a 10 μs . Que valor devemos selecionar para o ADC12CLK?

Solução: Se a taxa é de 20 ksps, então o período é de 50 μs . Dentro deste período devem estar os 10 μs de amostragem e mais o tempo gasto para o ADC12 realizar a conversão, como mostrado na Figura 11.29, sem esquecer o período extra do ADC12CLK para a sincronização (t_{sinc}). São então 13 períodos para o ADC12 converter em 12 bits mais um extra período para a sincronização. É fácil de ver que o período do ADC12CLK deve ser menor que 40/14 μs ($40/14 = 3,077 \mu\text{s}$). Isto resulta na frequência mínima de 350 kHz. Com o SMCLK especificado, o valor mais próximo (aproximação por valores superiores) que podemos chegar desta frequência é usando o divisor por 2, o que resulta em 524.288 Hz (período de 1,91 μs , aproximadamente).

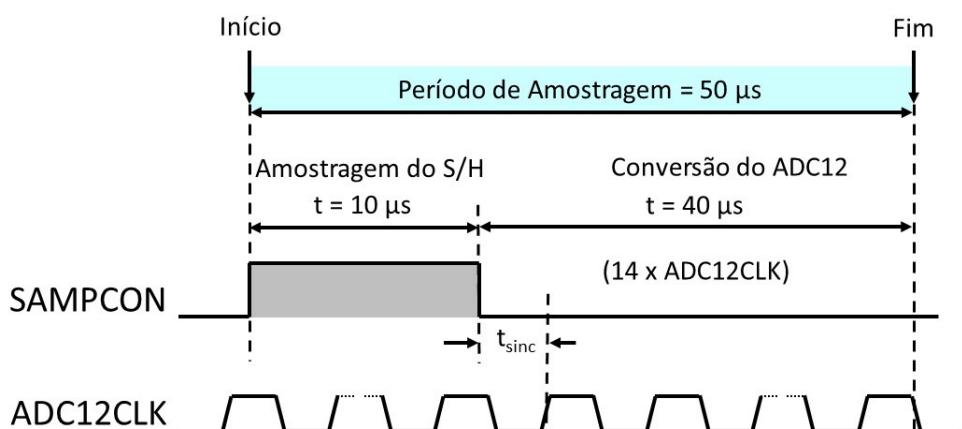


Figura 11.29. Diagrama de tempo para a escolha do relógio ADC12CLK, de acordo com o problema proposto.

Vamos agora fazer uma conferência para ver se este relógio é realmente adequado. Para gerar o mínimo de 10 μs de amostragem, e obedecendo as restrições do *Sample Timer*, vamos programar para amostrar durante 8 períodos (vide [Figura 11.x](#)), o que resulta em 15,26 μs ($8 * 1,91 \mu\text{s} = 15,26 \mu\text{s}$) de amostragem, acima dos 10 μs exigidos. A Figura 11.30 resume, com alguma aproximação, uma etapa de amostragem e conversão. Em 50 μs , temos a disponibilidade de 26 períodos de ADC12CLK (50 μs / 524.288 Hz) e fizemos o seguinte uso:

- 8 períodos para amostragem;
- 1 período para a sincronização;
- 13 períodos para o ADC12 converter e
- 4 períodos ocioso.

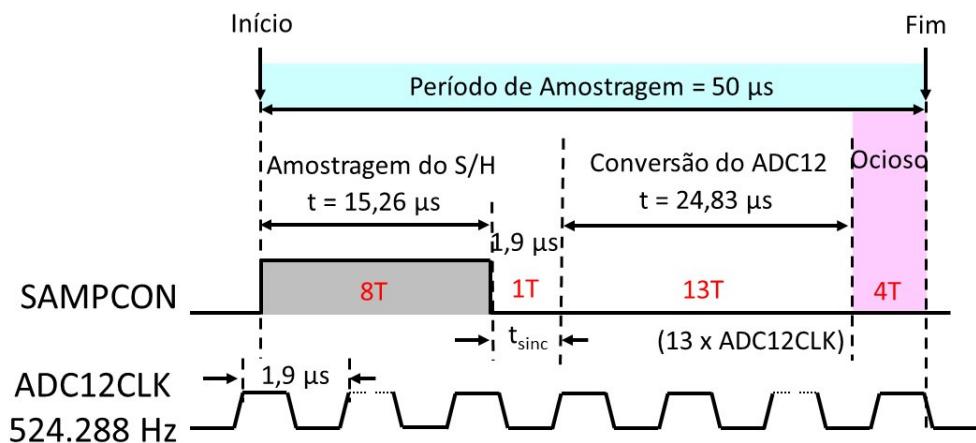


Figura 11.30. Diagrama de tempo comprovando a adequação do relógio ADC12CLK em 524.288 Hz (a figura não está em escala).

Sugestão de exercício para um gerador aleatório: inicializar o gerador aleatório do ER 2 do Apêndice C usando uma entrada em aberto do ADC, ou então, selecionar uma referência que não existe. Fazer várias vezes, selecionando 2 ou 3 menos significativos de forma a montar um número.

11.7. Seleção das Tensões de Referência

Esse assunto de tensão de referência demanda por uma pequena explicação de desacoplamento entre circuitos analógicos e digitais. Quando se trabalha com um projeto misto, envolvendo componentes analógicos e digitais, é importante separar os caminhos de alimentação (V_{cc}) e os caminhos de retorno (GND), como mostrado na figura abaixo. Isto porque o chaveamento dos circuitos digitais gera muito ruído sobre as linhas de alimentação e terra que perturbam o funcionamento da porção analógica. O desacoplamento é tão mais importante quanto maior for resolução do ADC. Assim surgem os termos alimentação analógica (AV_{cc}) e terra analógica ($AGND$). A figura não mostra, mas junto aos componentes devem ser usados capacitores de desacoplamento.

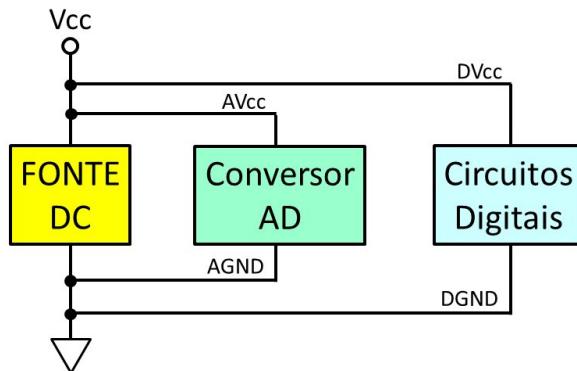


Figura 11.31. Ilustração do desacoplamento dos circuitos analógicos dos digitais.

O ADC12 oferece bastante flexibilidade para a seleção das tensões de referência. Na Figura 11.9 vimos o conceito de tensão de referência para típico ADC. A novidade no MSP é a possibilidade de duas tensões de referência: a superior é denominada V_{R+} e inferior é denominada V_{R-} . O leitor deve ter claro que a denominação V_{R-} não é uma tensão abaixo de 0 Volts, mas sim o limite inferior da faixa de conversão do ADC. A Figura 11.32 deixa clara essa ideia.

A equação abaixo indica o código gerado pelo ADC em função da tensão analógica (V_{in}) apresentada na entrada, para o caso de configuração em 12 bits. O valor 4095 (12 bits) deve ser alterado para 1023 (10 bits) ou 255 (8 bits), em função da resolução que se emprega. Note que após o Reset o ADC é sempre colocado no modo 12 bits (ADC12RES = 2).

$$N = 4.095 \times \frac{V_{in} - V_{R-}}{V_{R+} - V_{R-}}$$

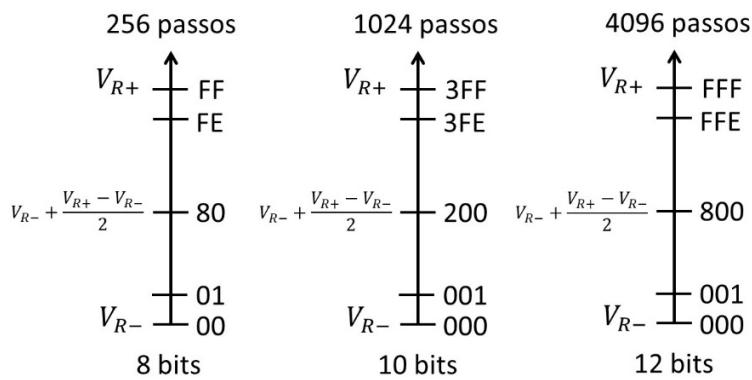


Figura 11.32. Passos de conversão em função das tensões de referência e resolução do ADC12.

Essas referências de tensão, V_{R+} e V_{R-} , podem ser geradas internamente no MSP ou definidas externamente pelo usuário, ao apresentar tensões em dois pinos do microcontrolador. Neste caso, elas são denominadas de V_{eREF+} e V_{eREF-} . A tabela abaixo indica os limites para essas tensões de referência e seu acesso pelos pinos P5.0 e P5.1. No caso do LaunchPad, esses pinos não estão disponíveis nas barras de terminais e por isso seu emprego é difícil. Seria necessário acessar os pinos 9 e 10 do chip do microcontrolador.

*Tabela 11.3. Limites para as tensões externas de referência usadas no ADC12
(Manual MSP430F5529, pag 73)*

Parâmetro	Mínimo	Máximo	Pino
V_{eREF+}	1,4 V	AVcc	P5.0
V_{eREF-}	0 V	1,2 V	P5.1
$V_{eREF+} - V_{eREF-}$	1,4 V	AVcc	-

Vamos agora abordar as flexibilidades para se especificar as referências de tensão (V_{R+} e V_{R-}) do ADC12. A Figura 11.33 apresenta o diagrama de blocos, que é específico para o F5529. O leitor pode até se assustar com sua complexidade. Na verdade, não é assim tão complexo. Note que os multiplexadores estão numerados. Na parte inferior direita desta figura é o ADC com suas duas entradas de referência (V_{R+} e V_{R-}).

Iniciamos a especificação de V_{R-} (referência “negativa”) que é feita pelo multiplexador 1. Esse multiplexador tem apenas duas entradas: a terra analógica (AV_{SS}) e referência externa (V_{eREF-}) que corresponde ao pino P5.1 do MSP. Pelo bit ADC12SREF2 o usuário indica qual referência deseja usar. Em nosso caso, este bit estará em zero, ou seja, sempre vamos usar AGND como referência “negativa”.

A indicação de V_{R+} (referência “positiva”) oferece bastante flexibilidade. O multiplexador 2 mostra que são três possibilidades, selecionadas pelos bits de ADC12SREF.

- MUX2.0: VCC analógico (AVCC);
- MUX2.1: Referência interna (V_{REF+}) ou
- MUX2.2 e 3: Referência externa (V_{eREF+})

A entrada 0 deste multiplexador está conectada ao AVCC que é igual ao VCC que alimenta o chip CPU, em geral, 3.3 V. Ainda neste multiplexador 1, as entradas 2 e 3 estão conectadas à referência externa (V_{eREF+}) que corresponde ao pino P5.0 do MSP.

Vamos agora ao estudo da referência interna, que corresponde à entrada 1 do multiplexador 2. Essa parte é um pouco mais complexa pois visa atender a duas versões de microcontrolador:

- Microcontroladores com o Módulo de Referência (REF) em separado que pode fornecer tensões de 1,5 V, 2,0 V e 2,5 V, como é o caso do MSP430F5529, objeto deste estudo e
- Microcontroladores sem o Módulo de Referência (REF) e só podem trabalhar com referência de 1,5 V, e 2,5 V, como é o caso dos MSP430F54xx.

São usados então duas classes de sinais de controle, mas com funções equivalentes. Isto é feito por questões de compatibilidade. O bit de controle mais importante é o REFMSTR que habilita o módulo de referência. A tabela abaixo indica a correspondência entre os sinais de controle e os multiplexadores (ver Figura 11.33) que os selecionam. Note que os sinais de controle são exclusivos. Por exemplo, se REFMSTR = 0, os sinais REFON, REFVSEL, REFOUT e REFTCOFF são *don't care* e vice-versa.

Tabela 11.4. Resultados da seleção entre os dois módulos de referência de tensão

MUX	REFMSTR =1 (seleciona Módulo de Referência)		REFMSTR = 0 (seleciona Módulo ADC12_A)	
3	REFON	Habilita módulo	ADC12REFON	Habilita módulo
4	REFVSEL	1,5 V / 2,0 V / 2,5 V	ADC12REF12_5V	1,5 V / 2,5 V
5	REFOUT	Hab. saída externa	ADC12REFOUT	Hab. saída externa
-	REFTCOFF	Temperatura	ADC12TCOFF	Temperatura

Ao selecionar o módulo de referência do ADC12_A (REFMSTR = 0), só é possível trabalhar com referências de 1,5 V (ADC12REF2_5V = 0) ou 2,5V (ADC12REF2_5V = 1). Se o bit ADC12REFOUT for colocado em nível alto, a tensão de referência fica disponível no pino P5.0, que é o de referência externa positiva (V_{eREF+}). Isto é feito para permitir que o usuário adicione um capacitor de desacoplamento externo (usual é um eletrolítico de 10 μ F em paralelo com cerâmico de 100 n F) para garantir a estabilidade da referência interna (eliminar ruídos).

Esse multiplexador numerado de 6 merece uma explicação à parte. Mesmo quando se usa referência interna, é preciso garantir sua estabilidade e eliminar ruídos. Isso é muito importante quando se trabalha em 12 bits. Por exemplo, vamos considerar a referência de 2,0 V sendo usada na conversão em 12 bits. A resolução (passo) neste caso é de, aproximadamente, 500 μ V (2 / 4096). Isto significa que qualquer ruído acima de 500 μ V nos “rouba” 1 bit. Um ruído com amplitude de 1 mV nos “rouba” 2 bits, o que significa, que configuramos para 12 bits, mas só temos informação em 10 bits, os outros dois são apenas ruído.

Por isso, o fabricante precisava de criar um caminho elétrico que permitisse ao usuário, se fosse o caso, adicionar um capacitor externo para desacoplamento. Esse é o papel do multiplexador 6. Se a sua entrada 1 está selecionada, a referência interna é conectada ao pino P5.0 (V_{eREF+}). Por outro lado, este pino P5.0 é também destinado a que o usuário

forneca sua tensão de referência. Neste caso, o multiplexador 6 deve estar com sua entrada 0 selecionada. Note que nada está conectado à essa entrada. Sim, para se usar uma referência externa, é preciso estar com saída da referência interna desligada ($\text{REFOUT} = 0$ ou $\text{ADC12REFOUT} = 0$).

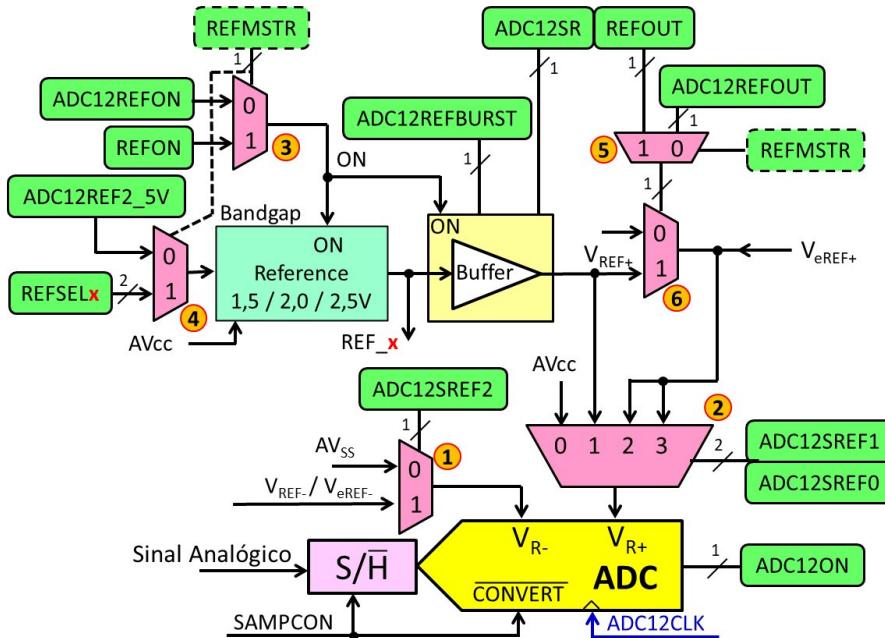


Figura 11.33. Diagrama para a seleção das referências de tensão para o ADC12. Note que o bloco REFMSTR está repetido para facilitar o desenho.

Vamos agora tratar do consumo de energia na geração das referências internas. Em primeiro lugar, como mostrado na figura, a unidade de referência possui um *buffer* com um controle em separado. Se a referência for desabilitada com $\text{ADC12REFON} = 0$ ou $\text{REFON} = 0$, conforme o caso, o *buffer* também é desabilitado.

Se a referência interna for habilitada $\text{ADC12REFON} = 1$ ou $\text{REFON} = 1$, o *buffer* então é controlado pelo bit ADC12REFBURST . Se este bit estiver em 0, o *buffer* sempre está ativo. Esta é a situação para quando se habilita a saída externa da referência interna, com $\text{ADC12REFOUT} = 1$ ou $\text{REFOUT} = 1$. Por outro lado, com $\text{ADC12REFBURST} = 1$, o *buffer* só é ativado quando se inicia uma conversão no ADC. Enquanto está desabilitado, o *buffer* não consome corrente. Além disso, há uma relação de compromisso entre energia consumida pelo *buffer* e taxa de conversão. Se a taxa de conversão for menor que 50 ksps, então a ativação do bit ADC12SR reduz em 50% seu consumo de corrente.

Um detalhe merece comentário. O acesso aos bits ADC12REF0 , 1 e 2 é feito pelo registrador ADC12MCTLx , que é estudado adiante. A tabela abaixo indica como esses três bits são usados.

Tabela 11.5. Seleção das referências de tensão no campo ADC12SREFx

ADC12SREFx			Nr	V _{R+}	V _{R-}
ADC12 REF0	ADC12 REF1	ADC12 REF2			
0	0	0	0	AVcc	AVss
0	0	1	1	V _{REF+}	AVss
0	1	0	2	V _{eREF+}	AVss
0	1	1	3	V _{eREF+}	AVss
1	0	0	4	AVcc	V _{REF- / V_{eREF-}}
1	0	1	5	V _{REF+}	V _{REF- / V_{eREF-}}
1	1	0	6	V _{eREF+}	V _{REF- / V_{eREF-}}
1	1	1	7	V _{eREF+}	V _{REF- / V_{eREF-}}

11.9. Memória para a Conversão

O ADC12 pode trabalhar convertendo um único canal ou uma sequência de canais, e ainda oferece a opção de repetição. Isto ainda será estudado com mais detalhes, por enquanto é importante saber que a memória de conversão tem um papel muito importante nos modos de conversão. São 4 modos, definidos pelos bits de CONSEQ:

- (CONSEQ = 0) Um único canal;
- (CONSEQ = 1) Um único canal repetido;
- (CONSEQ = 2) Sequência de canais e
- (CONSEQ = 3) Sequência de canais repetida.

Os resultados das conversões do ADC12 são armazenados em 16 posições de memória, denominadas de ADC12MEM0, ADC12MEM1, ..., ADC12MEM15, como mostrado na Figura 11.34. Para cada uma dessas posições de memória existe um registrador de controle. Eles são denominados ADC12MCTL0, ADC12MCTL1, ..., ADC12MCTL15. Cada registrador de controle especifica o canal de entrada, as referências de tensão e ainda tem um bit para indicar se a posição em questão é a última quando se trabalha com sequência de canais. No registrador ADC12CTL1 existem 4 bits denominados CSTARTADD que permitem ao programador indicar a primeira posição de memória a ser usada.

A Tabela 11.6, apresentada logo adiante traz uma ilustração das posições de memória que recebem os resultados das conversões do ADC12 e dos registradores de controle

para cada uma dessas posições de memória. O controle possui 3 campos, com os seguintes significados:

- EOS → indica se é o fim de uma sequência de conversões
- ADC12SREFx → seleciona uma dentre as 8 possíveis combinações de tensão de referência
- ADC12INCHx → indica qual das 16 possíveis entradas será convertida.

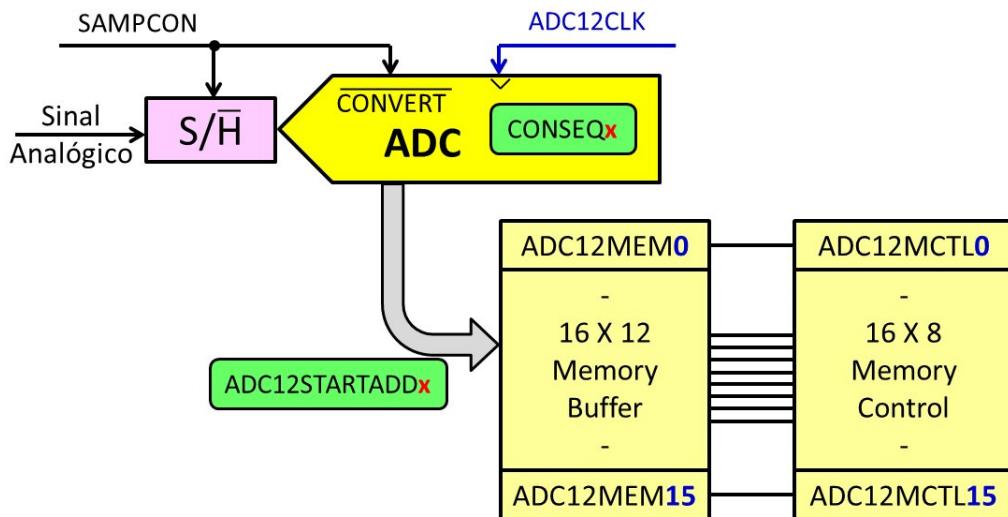


Figura 11.34. Diagrama indicando o acesso do ADC12 à memória de conversão.

Tabela 11.6. Ilustração das memórias para receber os resultados das conversões do ADC12 e de seus registradores de controle

Memória	Controle da Memória			
16 bits	1 bit	3 bits	4 bits	
ADC12MEM0	EOS	ADC12SREFx	ADC12INCHx	
ADC12MEM1	EOS	ADC12SREFx	ADC12INCHx	
ADC12MEM2	EOS	ADC12SREFx	ADC12INCHx	
...	
ADC12MEM14	EOS	ADC12SREFx	ADC12INCHx	
ADC12MEM15	EOS	ADC12SREFx	ADC12INCHx	

Esta forma de se organizar as posições de memória que vão receber os resultados e seus respectivos controles oferece bastante flexibilidade. A Tabela 11.7 apresenta um exemplo. Vamos considerar que esteja programada a conversão de uma sequência de canais sem repetição (CONSEQ = 3) e que a posição de memória 2 é selecionada como sendo a de

partida (ADC12CSTARTADD = 2). O leitor deve notar que o controle da posição ADC12MEM6 tem o bit EOS em 1, o que significa que esta é a última posição a ser convertida. A cada disparo do conversor, temos então os seguintes resultados (a Figura 11.x apresenta as possibilidades para as tensões de referência):

- **Disparo 1:** ADC12MEM2 recebe o resultado a conversão do canal 3, usando como tensões de referência AVCC e AVSS;
- **Disparo 2:** ADC12MEM3 recebe o resultado a conversão do canal 7, usando como tensões de referência V_{REF+} e AVSS;
- **Disparo 3:** ADC12MEM4 recebe o resultado a conversão do canal 5, usando como tensões de referência AVCC e AVSS;
- **Disparo 4:** ADC12MEM5 recebe o resultado a conversão do canal 0, usando como tensões de referência V_{REF+} e AVSS;
- **Disparo 5:** ADC12MEM6 recebe o resultado a conversão do canal 4, usando como tensões de referência AVCC e AVSS e esta é a última conversão porque está marcada com EOS = 1.

Tabela 11.7. Exemplo de uma configuração dos Registradores de Controle da memória de conversão para uma sequência de conversões

Memória	Controle da Memória		
	EOS	ADC12SREFx	ADC12INCHx
ADC12MEM0	0	0	2
ADC12MEM1	0	0	4
ADC12MEM2	0	0	3
ADC12MEM3	0	1	7
ADC12MEM4	0	0	5
ADC12MEM5	0	2	0
ADC12MEM6	1	0	4
...

Vemos então que esta arquitetura de memória permite uma grande flexibilidade na especificação dos canais e das referências de tensão. No caso do exemplo, tivemos 6 conversões. Se o modo Sequência de Canais com Repetição estivesse selecionado, tudo seria repetido. Se nenhuma das posições de controle tiver o bit EOS ativado, após usar ADC12MEM15 a sequência volta para ADC12MEM0. A cada posição de memória preenchida, é ativada a *flag* respectiva no registrador ADC12IFG.

Estão disponíveis dois formatos para se gravar o resultado na memória. Eles são especificados pelo bit ADC12DF. Se ADC12DF = 0, então a conversão é sem sinal e os bits do resultado são alinhados pela direita, isto significa que as posições à esquerda são

preenchidas com zeros. Se $ADC12DF = 1$, então os bits da conversão são alinhados pela esquerda e gravados como complemento a 2. (Café, acho que a figura não está correta. Não entendi direito. Primeiro se calcula o complemento a 2 e depois se alinha pela esquerda ou é ao contrário? Qual a vantagem disso? Estudar melhor!)

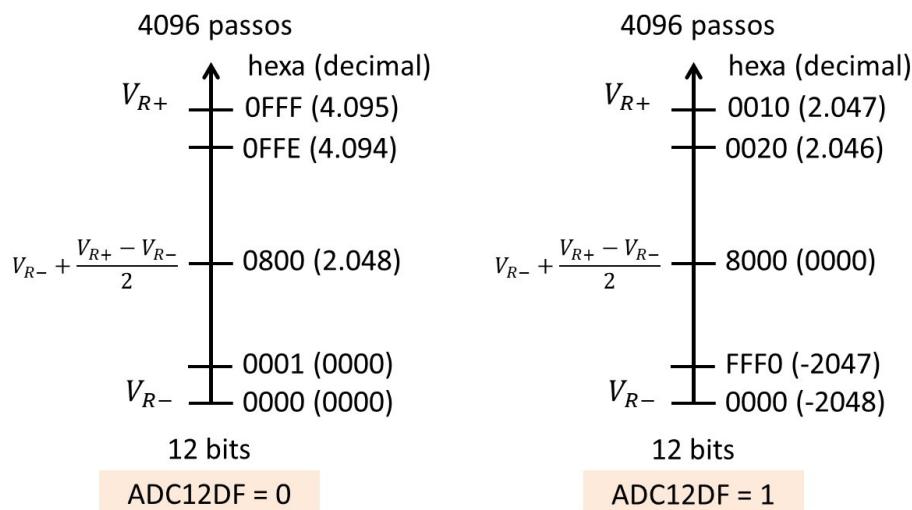


Figura 11.35. Resultados do ADC12 para os dois possíveis valores de $ADC12DF$: justificado pela direita ($ADC12DF = 0$) e justificado pela esquerda em complemento a 2 ($ADC12DF = 1$). Acho que a figura está errada.

11.10. Modos de Conversão

Como já foi afirmado, o SDC12_A oferece 4 modos de operação, como mostrado na Tabela 11.8. O modo é selecionado no campo CONSEQ (2 bits) do registrador ADC12CTL1.

Tabela 11.8. Modos de operação do ADC12_A

ADC12CONSEQ (2 bits)	Modo	Operação
00	Um canal, uma conversão	Um canal é convertido uma única vez.

01	Sequência de canais (<i>autoscan</i>)	Uma sequência de canais é convertida uma única vez.
10	Um canal repetido	Um canal é convertido repetidamente.
11	Sequência de canais repetida (<i>repeat autoscan</i>)	Uma sequência de canais é convertida repetidamente.

A diferença entre os modos repetido ou não-repetido é simples e se resume no controle do bit ADC12ENC, que é o de habilitação do ADC12_A. No modo não repetido, ao final da operação programada, este bit é colocado em zero. Assim nova conversão demanda por uma ação do programador para ativar a habilitação do ADC12_A (ADC12ENC = 1). No modo repetido, ao final da operação programada, a habilitação permanece ativada, assim, quando ocorrer o novo disparo de conversão, tudo se repete.

Os modos de operação serão estudados mais adiante, com o uso de exercícios resolvidos.

Modo de Conversão Rápida

Este é um modo especial e permite sucessivas conversões, tão rápidas quanto possível. Assim que termina uma conversão, a próxima é iniciada, independente de disparo. Em inglês, o fabricante o denominou de *Multiple Sample and Convert* e sua habilitação é conseguida com a ativação do bit ADC12MSC.

Ao se colocar em 1 o bit ADC12MSC com CONSEQ = 1, 2, ou 3 e tendo selecionado o sample timer (vide Figuras 11.24, 25 e 26) o primeiro flanco de subida de SHI dispara a primeira conversão. ([Manual 28.2.7.5, pag 743, o que se quer dizer com usando o sample timer, ADC12SHP = 1? Note que com ADC12SHP = 0, também gera o sinal SHI. Provar diversas combinações para esclarecer a dúvida](#)). A partir daí, sucessivas conversões são disparadas, tão logo a anterior termine.

Se o modo Sequência de Canais (sem repetição) estiver selecionado, os próximos flancos de SHI serão ignorados até completar a sequência. Para repetir, é preciso colocar em 1 o bit ADC12ENC, que é o bit que controla a habitação do ADC.

Nos modos com repetição, os próximos flancos de SHI são ignorados e as conversões continuam até o programador desabilitar o ADC desativando o bit ADC12ENC.

Como Parar as Conversões

O bit de habilitação ADC12ENC é o indicado para parar as conversões, entretanto, o comportamento do ADC12 é diferente, de acordo com o modo selecionado,

No modo um canal sem repetição (CONSEQ = 0), ao zerar ADC12ENC, a conversão para imediatamente e o resultado é imprevisível. Para poder aproveitar o resultado, é recomendado que, se faça o polling do bit de ocupado (ADC12BUSY) esperando primeiro que ele vá para zero para só depois zerar ADC12ENC.

No modo um canal repetido (CONSEQ = 1), ao zerar ADC12ENC a operação para após o fim da atual conversão.

No modo sequência de canais, com ou sem repetição (CONSEQ = 2 ou 3), ao zerar ADC12ENC a operação para após o fim da sequência que estiver em curso.

De qualquer forma, qualquer conversão pode ser parada ao se fazer CONSEQ = 0 e ADC12ENC = 0. O conversor para imediatamente e o resultado da conversão que estava em curso é imprevisível.

11.11. Sensor de Temperatura Integrado

Ficou para o futuro.

11.12. Interrupções com o ADC12_A

O ADC12_A oferece 18 fontes de interrupção, elas estão descritas logo a seguir.

ADC12IFG0, ..., ADC12IFG15 são as flags indicam quando uma conversão terminou e sua respectiva posição de memória foi preenchida com o resultado desta conversão. Para facilitar o trabalho com essas 16 interrupções existe o registrador ADC12IV que trabalha da forma já conhecida. A única diferença é o acesso ao ADC12IV não apaga nenhuma das flags. Uma determinada flag é apagada quando sua respectiva posição de memória é lida. Elas também podem ser apagadas por software.

ADC12OV é a flag que indica o atropelamento, ou seja, indica se o resultado de uma conversão foi escrito na respectiva posição de memória antes do resultado da conversão anterior ter sido lido. Em outras palavras, o hardware do ADC12_A ativa esta flag toda vez que escrever numa posição de memória que esteja com sua flag ADC12IFG em 1.

ADC12TOV é a flag que indica o atropelamento no tempo, ou seja, quando se pede para iniciar uma nova conversão antes da conversão anterior terminar.

Qualquer acesso ao registrador ADC12IV zera as flags ADC12OV e ADC12TOV se forem da interrupção pendente de maior prioridade. Não faz muito sentido. Veja transcrição do manual (tópico 28.2.10.1, pag 746) que me parece confusa.

Any access, read or write, of the ADC12IV register automatically resets the ADC12OV condition or the ADC12TOV condition if either was the highest-pending interrupt. Neither interrupt condition has an accessible interrupt flag.

Usando a linguagem C, apresentamos abaixo uma sugestão para o manipulador de interrupções para o ADC12_A. A lógica é bem simples e já estudada por ocasião da abordagem de diversos outros recursos do MSP430.

```
#pragma vector = ADC12_VECTOR //Interrupção de número 54
__interrupt void adc12_isr(void){
    switch(__even_in_range(ADC12IV, 0x24)) {
        case 0x0: break; //Nada pendente
        case 0x2: adc_ov(); break; //Atropelamento em memória
        case 0x4: adc_tov(); break; //Atropelamento no tempo
        case 0x6: adc_ifg0(); break; //ADC12MEM0 preenchida
        case 0x8: adc_ifg1(); break; //ADC12MEM1 preenchida
        ...
        case 0x22: adc_ifg14(); break; //ADC12MEM14 preenchida
        case 0x24: adc_ifg15(); break; //ADC12MEM15 preenchida
    }
}
```

11.13. Registradores do ADC12_A

Nesta seção são descritos os registradores disponíveis para a operação do ADC12_A. A Tabela 11.9 apresenta uma listagem completa de todos esses registradores. Alguns podem ser acessados como um registrador de 16 bits ou em duas porções de 8 bits. Outros possuem acesso exclusivo em 8 bits. Para o programador em C, isso não faz grande diferença, já que o compilador cuida dos detalhes.

Tabela 11.9. Registradores do ADC12_A

16 bits	8 bits	Acesso	Reset	Ordem
ADC12CTL0	ADC12CTL0_H	R/W	0	MSB
	ADC12CTL0_L	R/W	0	LSB
ADC12CTL1	ADC12CTL1_H	R/W	0	MSB
	ADC12CTL1_L	R/W	0	LSB
ADC12CTL1	ADC12CTL1_H	R/W	0	MSB
	ADC12CTL1_L	R/W	0	LSB

ADC12CTL2	ADC12CTL2_H	R/W	0	MSB
	ADC12CTL2_L	R/W	0x20	LSB
ADC12IFG	ADC12IFG_H	R/W	0	MSB
	ADC12IFG_L	R/W	0	LSB
ADC12IE	ADC12IE_H	R/W	0	MSB
	ADC12IE_L	R/W	0	LSB
ADC12IV	ADC12IV_H	R/W	0	MSB
	ADC12IV_L	R/W	0	LSB
ADC12MEM0	ADC12MEM0_H	R/W	?	MSB
	ADC12MEM0_L	R/W	?	LSB
ADC12MEM1	ADC12MEM1_H	R/W	?	MSB
	ADC12MEM1_L	R/W	?	LSB
...
ADC12MEM15	ADC12MEM15_H	R/W	?	MSB
	ADC12MEM15_L	R/W	?	LSB
-	ADC12MCTL0	R/W	?	-
-	ADC12MCTL1	R/W	?	-
...
-	ADC12MCTL15	R/W	?	-

? = valor indefinido

11.13.1. ADC12CTL0 – Registrador 0 de Controle do ADC12 (*ADC12_A Control Register 0*)

Este registrador permite ligar ou desligar o conversor, habilitá-lo, disparar conversão por software, especificar os períodos de amostragem (com ASDC12SHP = 1), controlar tensão de referência e interrupções por atropelamento.

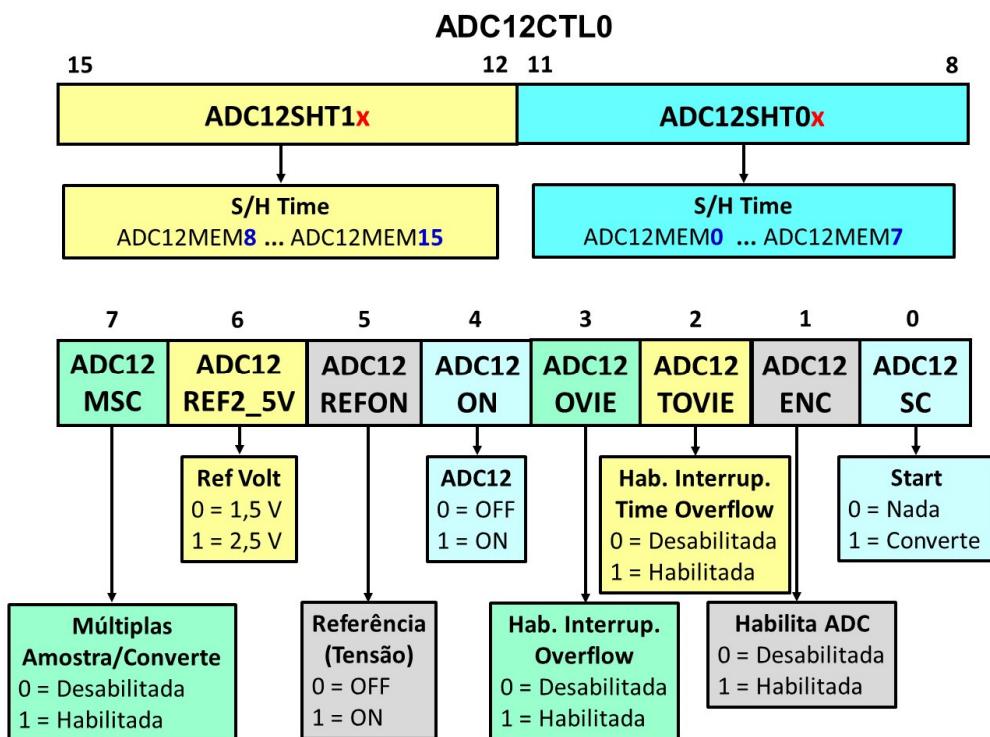


Figura 11.36. Descrição dos bits do registrador ADC12CTL0.

(R/W) Bits 15, ..., 12: ADC12SHT1x – Temporização 1 para Amostra/Retém (ADC12_A Sample-and-Hold Time)

Com esses 4 bits o programador especifica a quantidade de ciclos do ADC12CLK que deve durar o período de amostragem, para os registradores ADC12MEM8, ..., ADC12MEM15. A tabela abaixo indica as possibilidades.

Tabela 11.10. Lista das possibilidades para o período de amostragem

ADC12_SHT1x ADC12_SHT0x	Períodos de ADC12CLK
0	4
1	8
2	16
3	32
4	64
5	96
6	128

ADC12_SHT1x ADC12_SHT0x	Períodos de ADC12CLK
8	256
9	384
10	512
11	768
12	1024
13	1024
14	1024

7	192	15	1024
---	-----	----	------

**(R/W) Bits 11, ..., 8: ADC12SHT0x – Temporização 0 para Amostra/Retém
(Sample-and-hold time)**

Com esses 4 bits o programador especifica a quantidade de ciclos do ADC12CLK que deve durar o período de amostragem, para os registradores ADC12MEM0, ..., ADC12MEM7. A tabela abaixo indica as possibilidades. A Tabela 11.10, apresentada acima lista as possibilidades.

**(R/W) Bit 7: ADC12MSC – Múltiplas Amostragem e Conversão
(Multiple sample and conversion)**

Este bit só deve ser usado quando se usa algum modo sequencial ou repetição, em outras palavras, ele não deve ser usado quando o modo é conversão única (ADC12CONSEQ = 0).

ADC12MSC = 0 → é necessário um flanco ascendente de SHI para disparar cada ciclo de amostra-e-converte.

ADC12MSC = 1 → o primeiro flanco de SHI dispara o temporizador de amostragem, e os próximos ciclos de amostra-e-converte são realizadas automaticamente, assim, que a conversão anterior termina.

**(R/W) Bit 6: ADC12REF2_5V – Tensão do Gerador de Referência do ADC12_A
(Reference generator voltage)**

Para que este bit tenha efeito, é preciso que o gerador de referência esteja ligado (ADC12REFON = 1). No caso do MSP430F5529, que também possui um módulo de referência de tensão (Módulo REF), este bit só é válido com REFMSTR = 0. Ver registrador de controle do da unidade REF.

ADC12REF2_5V = 0 → referência em 1,5 V (V_{REF+}).

ADC12REF2_5V = 1 → referência em 2,5 V (V_{REF+}).

**(R/W) Bit 5: ADC12REFON – Ligar Gerador de Referência do ADC12_A
(Reference generator voltage)**

Ligar o gerador de referência do ADC12_A. Para que este bit tenha efeito, é preciso que o gerador de referência esteja ligado (ADC12REFON = 1). No caso do MSP430F5529, que também possui um módulo de referência de tensão (Módulo REF), este bit só é válido com REFMSTR = 0. Ver registrador de controle do da unidade REF.

ADC12REFON = 0 → gerador de referência desligado.

ADC12REFON = 1 → gerador de referência ligado.

**(R/W) Bit 4: ADC12ON – Ligar Unidade ADC12_A
(ADC12_A on)**

Ligar a unidade ADC12_A.

ADC12ON = 0 → ADC12_A desligada.

ADC12ON = 1 → ADC12_A ligada.

(R/W) Bit 3: ADC12OVIE – Habilita Interrupção por Atropelamento**(*ADC12MEMx overflow-interrupt enable*)**

Habilita que ocorra interrupção por atropelamento, que quanto o ADC12 escrever num registrador de memória (ADC12MEMx) que continha uma conversão que ainda não foi lida.

ADC12OVIE = 0 → Interrupção por atropelamento desabilitada.

ADC12OVIE = 1 → Interrupção por atropelamento habilitada.

(R/W) Bit 2: ADC12TOVIE – Habilita Interrupção por Atropelamento no Tempo**(*ADC12MEMx conversion-time-overflow interrupt enable*)**

Habilita que ocorra interrupção por atropelamento no tempo, que é quando se dispara uma nova conversão antes da anterior ter terminado.

ADC12TOVIE = 0 → Interrupção por atropelamento no tempo desabilitada.

ADC12TOVIE = 1 → Interrupção por atropelamento no tempo habilitada.

(R/W) Bit 1: ADC12ENC – Habilita o Conversor**(*ADC12_A enable conversion*)**

Habilita o uso do ADC12_A.

ADC12ENC = 0 → Conversor ADC12_A desabilitado.

ADC12ENC = 1 → Conversor ADC12_A habilitado.

(R/W) Bit 0: ADC12SC – Dispara Conversão**(*ADC12_A start conversion*)**

Este bit serve para disparar uma conversão por software. Após iniciada a conversão, este bit é automaticamente zerado.

ADC12SC = 0 → Nenhum disparo.

ADC12SC = 1 → Dispara um ciclo de amostra-e-converte.

11.13.2. ADC12CTL1 – Registrador 1 de Controle do ADC12**(*ADC12_A Control Register 2*)**

Este registrador permite ligar ou desligar o conversor, habilitá-lo, disparar conversão por software, especificar os períodos de amostragem (com ASDC12SHP = 1), controlar tensão de referência e interrupções por atropelamento.

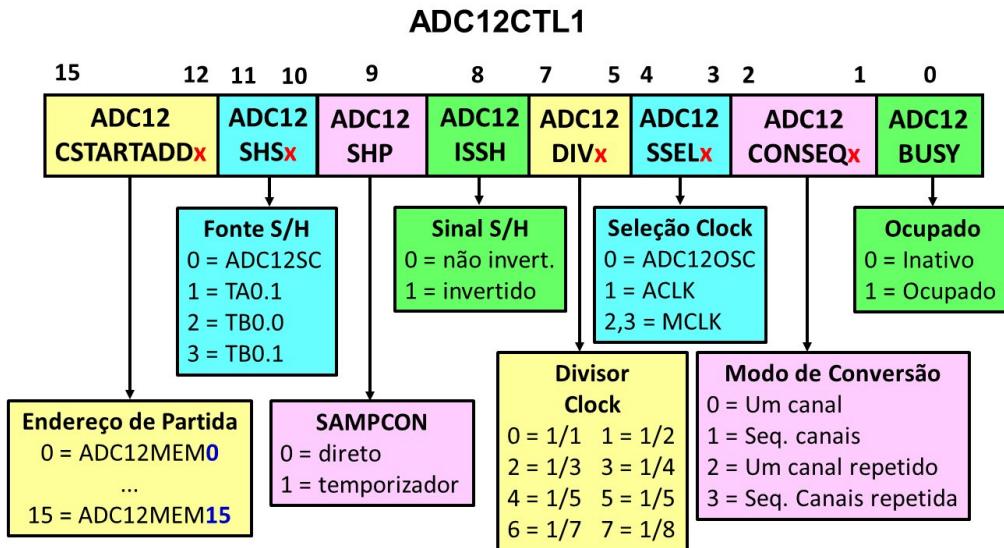


Figura 11.37. Descrição dos bits do registrador ADC12CTL1.

(R/W) Bits 15, ..., 12: ADC12STARTADD_x – Endereço de Partida**(ADC12_A conversion start address)**

Com esses 4 bits o programador indica que o primeiro registrador memória de conversão a ser usado.

(R/W) Bits 11, ..., 10: ADC12SHS_x – Origem do Sinal de Disparo da Conversão**(ADC12_A sample-and-hold source select)**

Com esses 2 bits o programador especifica a origem do sinal que vai dar origem à sequência de amostra-e-converte. No caso do MSP430 F5529 são as seguintes origens.

ADC12SHS = 0 → bit ADC12ST (disparar conversão por software).

ADC12SHS = 1 → TA0.1 dispara a conversão.

ADC12SHS = 2 → TB0.0 dispara a conversão.

ADC12SHS = 3 → TB0.1 dispara a conversão.

(R/W) Bit 9: ADC12SHP – Seleção da Origem do Sinal SAMPCOM**(ADC12_A sample-and-hold pulse-mode select)**

Com este bit, o programador indica que ser usar a saída do temporizador de amostragem ou usar o sinal SHI diretamente.

ADC12SHP = 0 → SAMPCOM gerado diretamente a partir da entrada (SHI)

ADC12SHP = 1 → SAMPCOM gerado pelo temporizador de amostragem.

(R/W) Bit 8: ADC12ISSH – Inverter sinal de disparo**(ADC12_A invert signal sample-and-hold)**

Este bit permite usar o sinal de disparo diretamente ou invertê-lo.

ADC12ISSH = 0 → Sinal de disparo sem inversão.

ADC12ISSH = 1 → Sinal de disparo é invertido.

**(R/W) Bits 7, ..., 5: ADC12DIV – Divisor do Relógio do ADC12_A
(*ADC12_A clock divider*)**

Com esses 3 bits o programador seleciona o divisor a ser usado para gerar o sinal ADC12CLK.

Tabela 11.11. Possibilidades de divisão na geração do ADC12CLK

ADC12DIV	Divisão
0	1
1	2
2	3
3	4

ADC12DIV	Divisão
4	5
5	6
6	7
7	8

**(R/W) Bits 4, 3: ADC12SSEL – Seleção da Fonte para Gerar o Relógio ADC12CLK
(*ADC12_A clock source select*)**

Com esses 2 bits o programador especifica a origem do sinal que vai dar origem ao relógio do ADC12_A que é o ADC12CLK.

ADC12SSEL = 0 → ADC12OSC, corresponde ao MODCLK da unidade UCS.

ADC12SSEL = 1 → ACLK.

ADC12SSEL = 2 → MCLK.

ADC12SSEL = 3 → SMCLK.

**(R/W) Bits 2, 1: ADC12CONSEQ – Seleção da Modo de Conversão
(*ADC12_A conversion sequence mode select*)**

Com esses 2 bits o programador especifica o modo a ser usado nas conversões.

ADC12CONSEQ = 0 → Um canal, uma conversão (sem repetição).

ADC12CONSEQ = 1 → Sequência de canais, sem repetição.

ADC12CONSEQ = 2 → Um canal, com repetição.

ADC12CONSEQ = 3 → Sequência de canais, com repetição.

**(R/W) Bit 0: ADC12BUSY – Indicação de ADC12_A Ocupado
(*ADC12_A busy*)**

Este bit indica que uma operação de amostra-e-converte está em curso.

ADC12BUSY = 0 → Nenhuma operação sendo executada.

ADC12BUSY = 1 → Sequência de amostra-e-converte está em curso.

**11.13.3. ADC12CTL1 – Registrador 2 de Controle do ADC12
(*USCI_Bx Control Register 1*)**

Este registrador permite controlar a resolução (quantidade de bits), a justificação do resultado (direita ou esquerda) da conversão, desligar o sensor de temperatura e configurar alguns outros recursos.

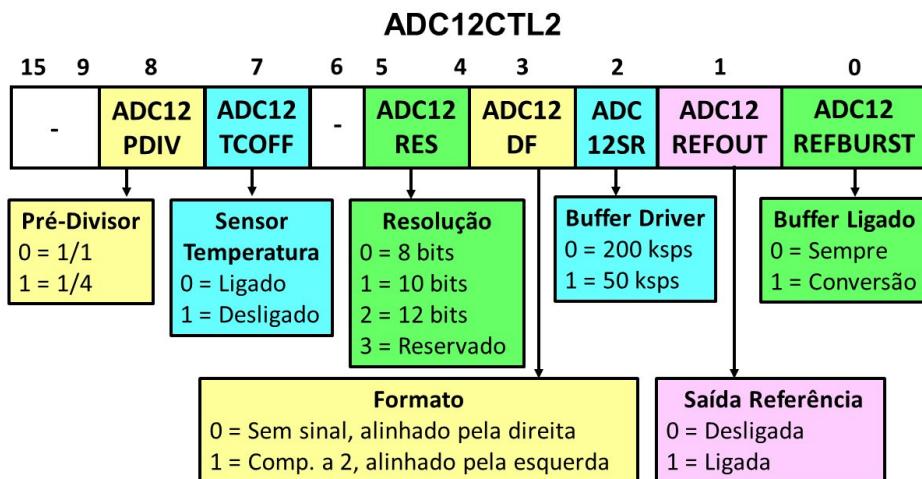


Figura 11.38. Descrição dos bits do registrador ADC12CTL2.

(R) Bits 15, ..., 9: Reservados

Esses bits são reservados e sua leitura sempre retorna zero.

(R/W) Bit 8: ADC12PDIV – Pré-divisor para Gerar ADC12CLK

(ADC12_A predivider)

Este bit permite ativar um divisor por 4 na geração do ADC12CLK.

ADC12PDIV = 0 → Nenhuma divisão é executada.

ADC12PDIV = 1 → É executada a divisão por 4.

(R/W) Bit 7: ADC12TCOFF – Desligar Sensor de Temperatura

(ADC12 A temperature sensor off)

Este bit permite desativar o sensor de temperatura para economizar energia. Este bit é válido se o bit REFMSTR do Módulo REF estiver em zero.

ADC12TCOFF = 0 → Sensor de temperatura ligado.

ADC12TCOFF = 1 → Sensor de temperatura desligado.

(R) *Bit 6: Reservado*

(4) Bit de Reserva
Este bit é reservado e sua leitura sempre retorna zero.

(R/W) Bits 5, 4: ADC12RES – Seleção da Resolução

(ADC12 A resolution)

Esses 2 bits permitem ao programador especificar quantidade de bits (resolução) da conversão. O Reset coloca o ADC no modo 12 bits (ADC12REFS = 2).

$\text{ADC12BES} = 0 \rightarrow$ Conversão em 8 bits (consome 9 períodos de ADC12CI K)

ADC12BES = 1 → Conversão em 10 bits (consome 11 períodos de ADC12CLK).

ADC12RES = 2 → Conversão em 12 bits (consome 12 períodos de ADC12CLK).
ADC12RES = 3 → reservado.

**(R/W) Bit 3: ADC12DF – Formato do Resultado da Conversão
(ADC12_A data read-back format)**

Formato e alinhamento (justificação) do resultado da conversão, vide Figura 11.35.

ADC12DF = 0 → Sem sinal e justificado pela direita.

ADC12DF = 1 → Em complemento a 2 e justificado pela esquerda.

**(R/W) Bit 2: ADC12SR – Taxa de Amostragem (Consumo na Referência de Tensão)
(ADC12_A sampling rate)**

Este bit controla o consumo do buffer da referência de tensão. Se a taxa de conversão é baixa, o buffer pode operar no modo de baixo consumo.

ADC12SR = 0 → Buffer pode operar com taxas de até 200 ksps (maior consumo).

ADC12SR = 1 → Buffer pode operar com taxas de até 50 ksps (menor consumo).

**(R/W) Bit 1: ADC12REFOUT – Saída da Tensão de Referência
(Reference output)**

A tensão de referência é disponibilizada num pino do MSP430 (P5.0 no caso do MSP430 F5529). A principal finalidade é permitir que o usuário adicione capacitores externos de desacoplamento, para assim tornar a referência mais estável. Este bit só é válido se o bit REFMSTR do Módulo de REF estiver em zero.

ADC12REFOUT = 0 → Saída da tensão de referência desligada

ADC12REFOUT = 1 → Saída da tensão de referência ligada

**(R/W) Bit 0: ADC12REFBURST – Controle do Buffer da Tensão de Referência
(Reference burst)**

O buffer da tensão de referência pode ficar ligado continuamente (maior consumo de energia) ou apenas ser ligado durante um ciclo de amostra-e-converte (menor consumo).

ADC12REFBURST = 0 → Buffer da tensão de referência ligado continuamente.

ADC12REFBURST = 1 → Buffer da tensão de referência só é ligado durante um ciclo de amostra-e-converte.

11.13.4. ADC12MEMx – Registrador Memória de Conversão (ADC12_A Conversion Memory Register)

São 16 posições nomeadas de ADC12MEM0, até ADC12MEM15. Elas são chamadas de Registradores Memória de Conversão e recebem os resultados das conversões.

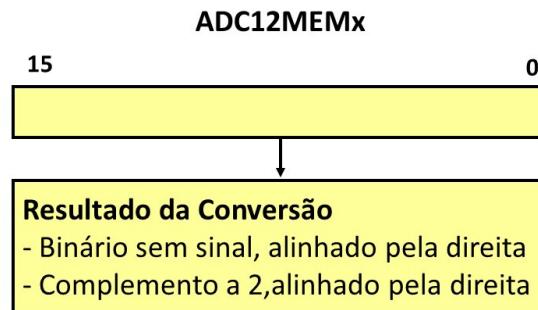


Figura 11.39. Registrador Memória de Conversão ($x = 0, 1, \dots, 15$).

11.13.5. ADC12MCTL_x – Registrador de Controle da Memória de Conversão (ADC12_A Conversion Memory Control Register)

São 16 registradores nomeados de ADC12MCTL0, até ADC12MCTL15. Eles controlam os parâmetros da conversão para cada posição de memória. Eles só podem ser alterados com o ADC12_A desabilitado, ou seja, ADC12EN = 0.

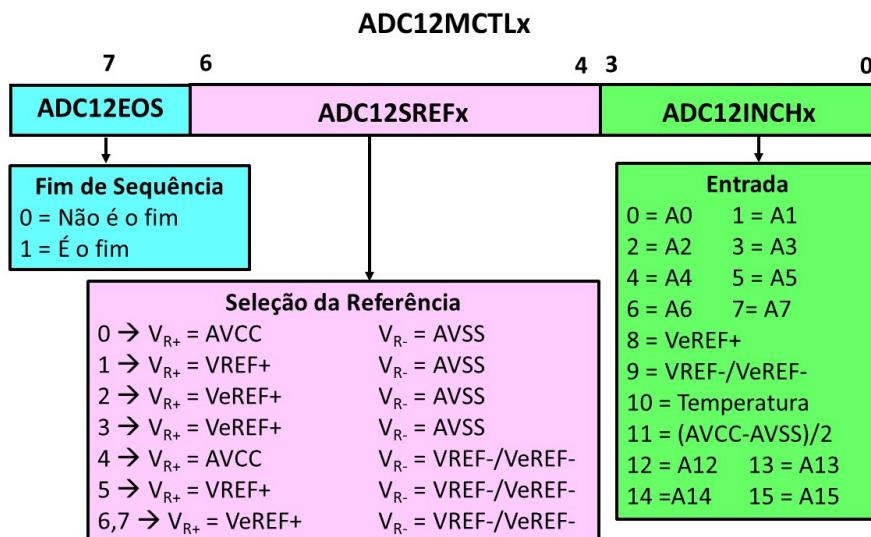


Figura 11.40. Registrador de Controle da Memória de Conversão ($x = 0, 1, \dots, 15$).

(R/W) Bit 7: ADC12EOS – Fim de Sequência (End of Sequence)

Este bit indica que a atual posição de memória é a última da sequência de conversões.
 ADC12EOS = 0 → Não é o fim da sequência.
 ADC12EOS = 1 → É o fim da sequência.

(R/W) Bits 6, ..., 4: ADC12REF_x – Seleção da Referência (Select reference)

Esses 3 bits permitem selecionar uma das 8 combinações de tensão de referência listadas na tabela abaixo.

Tabela 11.12. Possibilidades para as tensões de referência

ADC12REFx	V _{R+}	V _{R-}
0	AVcc	AVss
1	V _{REF+}	AVss
2	V _{eREF+}	AVss
3	V _{eREF+}	AVss
4	AVcc	V _{REF- / V_{eREF-}}
5	V _{REF+}	V _{REF- / V_{eREF-}}
6	V _{eREF+}	V _{REF- / V_{eREF-}}
7	V _{eREF+}	V _{REF- / V_{eREF-}}

**(R/W) Bits 3, ..., 0: ADC12INCHx – Seleção do Canal de Entrada Analógica
(Input channel select)**

Esses 4 bits permitem selecionar uma das 16 entradas analógicas, listadas na tabela abaixo. A entrada A12 pode ser usada para monitorar a tensão VBAT em dispositivos com bateria de backup.

*Tabela 11.13. Possibilidades para as entradas analógicas
(pinos indicados para o caso do MSP430 F5529)*

ADC12INCHx	Canal	Pino	ADC12INCHx	Canal	Pino
0	A0	P6.0	8	V _{eREF+}	-
1	A1	P6.1	9	V _{REF- / V_{eREF-}}	-
2	A2	P6.2	10	Temperatura	-
3	A3	P6.3	11	(AVcc - AVss) / 2	-
4	A4	P6.4	12	A12	P7.0
5	A5	P6.5	13	A13	P7.1
6	A6	P6.6	14	A14	P7.2
7	A7	P6.7	15	A15	P7.3

Observação: os pinos em cinza claro não estão acessíveis na LaunchPad.

11.13.6. ADC12IE – Registrador de Habilitação de Interrupções

(ADC12_A Interrupt Enable Register)

Este registrador permite habilitar a interrupção por ocasião de uma conversão. São 16 bits, um para cada posição de memória.



Figura 11.41. Registrador para habilitar interrupção para cada posição de memória que recebe o resultado da conversão ($x = 0, 1, \dots, 15$).

(R/W) Bits 15, ..., 0: ADC12IEx – Habilitação de Interrupção (Interrupt Enable)

Para cada posição de memória existe um bit denominado ADC12IFGx ($x = 0, \dots, 15$) que é ativado quando a respectiva memória recebe o resultado de uma conversão. Este registrador permite que cada um desses bits provoque interrupção.

ADC12IEx = 0 → Interrupção desabilitada.

ADC12IEx = 1 → Interrupção habilitada.

11.13.7. ADC12IFG – Registrador de Flags de Interrupções (ADC12_A Interrupt Flag Register)

Este registrador contém as flags que são ativadas por ocasião de uma conversão. São 16 flags, uma para cada posição de memória.

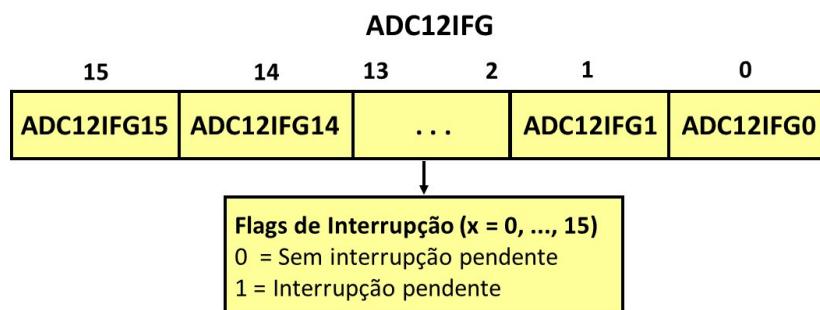


Figura 11.42. Registrador com as flags de interrupção. Uma flag é ativada quando a respectiva posição de memória que recebe o resultado da conversão ($x = 0, 1, \dots, 15$).

**(R/W) Bits 15, ..., 0: ADC12IFGx – Flag de Interrupção
(ADC12MEMx interrupt flag)**

A flag ADC12IFGx ($x = 0, \dots, 15$) é ativada quando a respectiva memória recebe o resultado de uma conversão. Esta flag é apagada quando a posição de memória é lida. Ela também pode ser apagada por software.

ADC12IFG = 0 → Sem interrupção pendente.

ADC12IFG = 1 → Interrupção pendente.

**11.13.8. ADC12IV – Registrador Vetor de Interrupção
(ADC12_A Interrupt Vector Register)**

Como existe somente uma posição na Tabela de Vetores de Interrupção para o ADC12_A, este registrador é usado para gerar vetores (números) que permitem identificar qual das 18 interrupções aconteceu. Ele faz ainda algo mais, no caso de várias interrupções pendentes, ele indica o número correspondente à de maior prioridade.

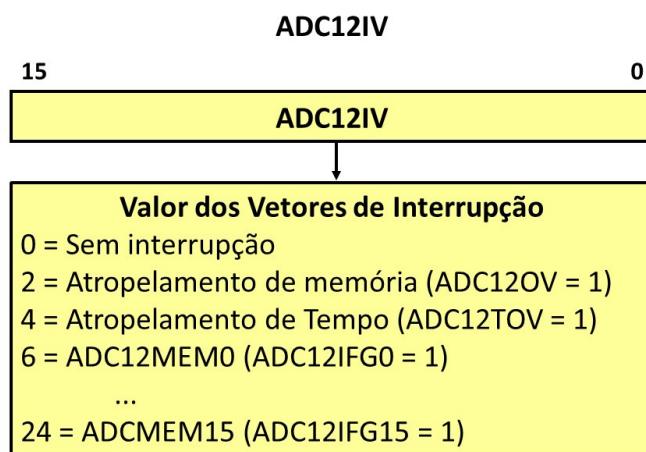


Figura 11.43. Registrador gerador de vetores de interrupção.

*Tabela 11.14. Valores do registrador de Vetor de Interrupção
(ADC12OV = Maior prioridade e ADC12IFG15 = menor prioridade)*

Valor	Interrupção	Valor	Interrupção
0x00	Nenhuma	0x14	AC12IFG7
0x02	ADC12OV	0x16	AC12IFG8

0x04	ADC12TOV
0x06	AC12IFG0
0x08	AC12IFG1
0x0A	AC12IFG2
0x0C	AC12IFG3
0x0E	AC12IFG4
0x10	AC12IFG5
0x12	AC12IFG6

0x18	AC12IFG9
0x1A	AC12IFG10
0x1C	AC12IFG11
0x1E	AC12IFG12
0x20	AC12IFG13
0x22	AC12IFG14
0x24	AC12IFG15

11.14. Ensaios dos Modos de Operação do ADC12_A

O manual do MSP430 apresenta, a partir da página 739, diagramas de estado para cada um dos 4 modos de operação (figuras 28-7, 28-8, 28-9 e 29-10). Esses diagramas abordam todas as opções para cada modo de operação e, algumas vezes, terminam por confundir o usuário. A experiência demonstrou que muitas vezes se perde bastante tempo tentando entender as nuances de cada modo.

Por isso, a proposta deste tópico é ilustrar com um pequeno programa e um diagrama de estado os modos de operação. Os diagramas estão simplificados e ressaltam apenas os detalhes importantes para cada modo. Os programas ilustrativos serão resumidos para poupar espaço. Como exemplo, será usado o canal A1 (P6.1) ou a sequência de canais A0, A1, A2 e A3 (P6.0, 1, 2 e 3).

11.14.1. Ensaio 1

Modo	Repetição	Disparo	CONSEQ	ADC12 SHS	ADC12 SHP	ADC12 MSC
Simples	Não	ADC12SC	0	0	0	0

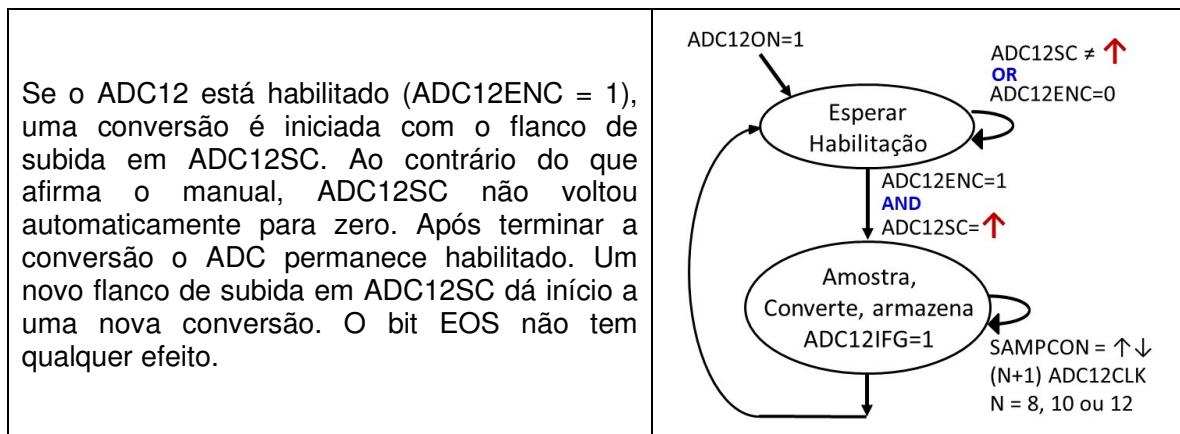


Figura 11.44. Explicação e diagrama de estados para o ADC12 operando no modo um canal sem repetição e conversão disparada por software (ADC12SC).

Listagem do Ensaio 1

```

// EN 11.1 - Ensaio
...
ADC_config();
while(TRUE) {
    ADC12CTL0 |= ADC12SC; //Disparar conversão
    ADC12CTL0 &= ~ADC12SC; //Zerar para a próxima
    while( (ADC12IFG&ADC12IFG0) == 0); //Esperar IFG0
    adc=ADC12MEM0;
}
...
void ADC_config(void) {
    ADC12CTL0 &= ~ADC12ENC; //Desabilitar para configurar
    ADC12CTL0 |= ADC12ON; //Ligar ADC
    ADC12CTL1 |= ADC12CONSEQ_0 | //Modo único canal
                  ADC12SHS_0 | //Disparo com ADC12SC
                  ADC12CSTARTADD_0 | //Resultado em ADC12MEM0
                  ADC12SSEL_3; //ADC12CLK = SMCLK
    ADC12MCTL0 = ADC12SREF_0 | ADC12INCH_1; //Config MEM0
    P6SEL |= BIT1; // Desligar digital de P6.1
    ADC12CTL0 |= ADC12ENC; //Habilitar ADC12
}
  
```

11.14.2. Ensaio 2

Modo	Repetição	Disparo	CONSEQ	ADC12 SHS	ADC12 SHP	ADC12 MSC
Simples	Não	TA0.1	0	1	0	0

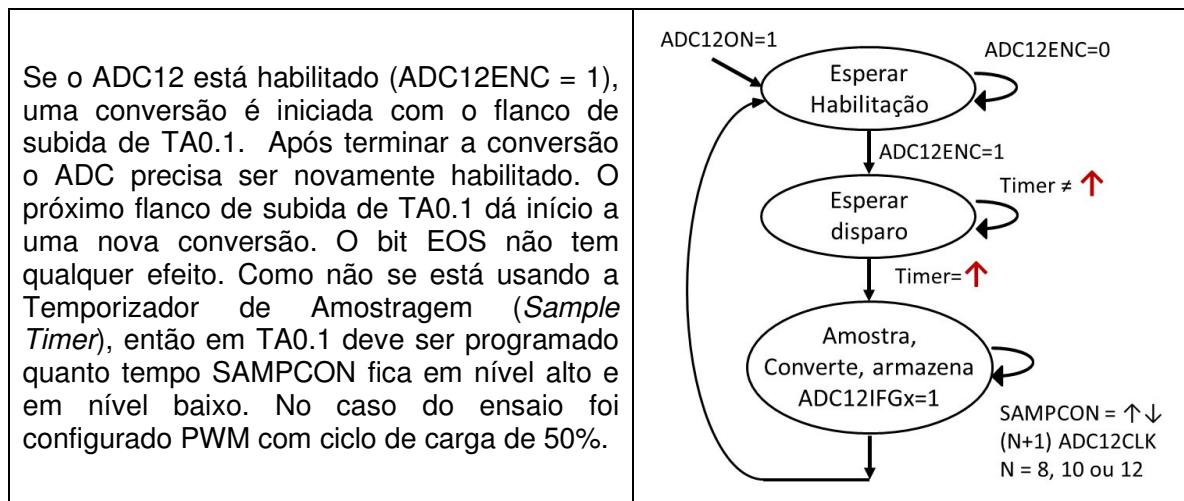


Figura 11.45. Explicação e diagrama de estados para o ADC12 operando no modo um canal sem repetição e conversão disparada pelo timer (TA0.1).

Listagem do Ensaio 2

```

// EN 11.2 - Ensaio
...
TA0_config();
ADC_config();
while(TRUE) {
    ADC12CTL0 |= ADC12ENC;           //Habilitar a cada conversão
    while( (ADC12IFG&ADC12IFG0) == 0); //IFG0 ?
    adc=ADC12MEM0;
}
...
void ADC_config(void){
    ADC12CTL0 &= ~ADC12ENC;          //Desabilitar para configurar
    ADC12CTL0 = ADC12SHT0_3 | ADC12ON; //Ligar ADC
    ADC12CTL1 = ADC12CONSEQ_0      | //Modo único canal
                ADC12SHS_1        | //Selecionar TA0.1
                ADC12CSTARTADD_0   | //Resultado em ADC12MEM0
                ADC12SSEL_3;        //ADC12CLK = SMCLK
    ADC12MCTL0 = ADC12SREF_0 | ADC12INCH_1; //Config MEMO
    P6SEL |= BIT1;                  // Desligar digital de P6.1
    ADC12CTL0 |= ADC12ENC;          //Habilitar ADC12
}
void TA0_config(void){
    TA0CTL = TASSEL_1 | MC_1;
    TA0CCTL1 = OUTMOD_6;           //Out = modo 6
    TA0CCR0 = 3276;                //100 Hz
    TA0CCR1 = TA0CCR0/2;           //Carga 50%
}

```

11.14.3. Ensaio 3

Modo	Repetição	Disparo	CONSEQ	ADC12 SHS	ADC12 SHP	ADC12 MSC
Sequência de canais	Não	ADC12SC	1	0	0	0

Se o ADC12 está habilitado ($\text{ADC12ENC} = 1$), uma conversão é iniciada com o flanco de subida em ADC12SC . Ao contrário do que afirma o manual, ADC12SC não voltou automaticamente para zero. Isto se repete enquanto a marca EOS não for atingido. Após a última conversão (EOS), o ADC12 é desabilitado e uma nova sequência só pode ser iniciada com $\text{ADC12ENC} = 1$. Se a marca EOS = 1 nunca for encontrada, a sequência continua a partir da posição 0 (ADC12MEM0). Durante uma sequência, se o ADC12 for desabilitado e novamente habilitado, a sequência recomeça da primeira posição. Um novo flanco de ADC12SC só devem acontecer após a conversão anterior terminar.

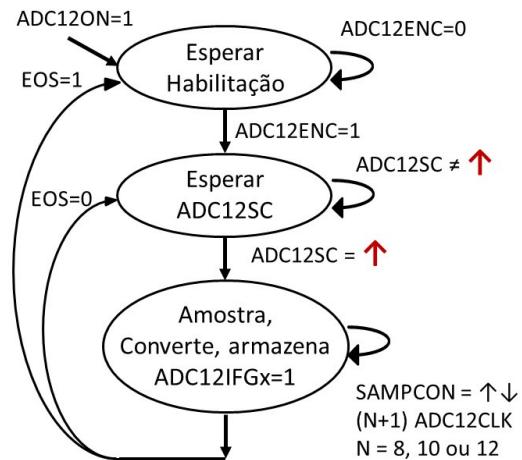


Figura 11.46. Explicação e diagrama de estados para o ADC12 operando no modo sequência de canais sem repetição e conversão disparada por software (ADC12SC).

Listagem do Ensaio 3

```

// EN 11.3 - Ensaio
...
ADC_config();
while(TRUE) {
    ADC12CTL0 |= ADC12ENC; //Habilitar a cada sequência
    ;
    ADC12CTL0 |= ADC12SC; //Disparar conversão
    ADC12CTL0 &= ~ADC12SC; //Zerar para a próxima
    while( (ADC12IFG&ADC12IFG0) == 0); //IFG0 ?
    adc[0]=ADC12MEM0;
    ;
    ADC12CTL0 |= ADC12SC; //Disparar conversão
    ADC12CTL0 &= ~ADC12SC; //Zerar para a próxima
    while( (ADC12IFG&ADC12IFG1) == 0); //IFG1 ?
    adc[1]=ADC12MEM1;
    ;
}
  
```

```

    ADC12CTL0 |= ADC12SC; //Disparar conversão
    ADC12CTL0 &= ~ADC12SC; //Zerar para a próxima
    while( (ADC12IFG&ADC12IFG2) == 0); //IFG2 ?
    adc[2]=ADC12MEM2;
    ;
    ADC12CTL0 |= ADC12SC; //Disparar conversão
    ADC12CTL0 &= ~ADC12SC; //Zerar para a próxima
    while( (ADC12IFG&ADC12IFG3) == 0); //IFG3 ?
    adc[3]=ADC12MEM3;
}

...
void ADC_config(void){
    ADC12CTL0 &= ~ADC12ENC; //Desabilitar para configurar
    ADC12CTL0 = ADC12ON; //Ligar ADC
    ADC12CTL1 = ADC12CONSEQ_1 | //Modo sequência de canais
        ADC12SHS_0 | //Selecionar ADC12SC
        ADC12CSTARTADD_0 | //Resultado em ADC12MEM0
        ADC12SSEL_3; //ADC12CLK = SMCLK
    ADC12MCTL0 = ADC12SREF_0 | ADC12INCH_0; //Config MEM0
    ADC12MCTL1 = ADC12SREF_0 | ADC12INCH_1; //Config MEM1
    ADC12MCTL2 = ADC12SREF_0 | ADC12INCH_2; //Config MEM2
    ADC12MCTL3 = ADC12EOS| ADC12SREF_0 | ADC12INCH_3; //MEM3 = última
    P6SEL |= BIT3|BIT2|BIT1|BIT0; // Desligar digital de P6.3,2,1,0
    ADC12CTL0 |= ADC12ENC; //Habilitar ADC12
}

```

11.14.4. Ensaio 4

Modo	Repetição	Disparo	CONSEQ	ADC12 SHS	ADC12 SHP	ADC12 MSC
Sequência de canais	Não	TA0.1	1	1	0	0

Se o ADC12 está habilitado ($\text{ADC12ENC} = 1$), uma conversão é iniciada com o flanco de subida de TA0.1. Isto se repete enquanto a marca EOS não for atingido. Após a última conversão (EOS), o ADC12 é desabilitado e uma nova sequência só pode ser iniciada com $\text{ADC12ENC} = 1$. Se a marca $\text{EOS} = 1$ nunca for encontrada, a sequência continua a partir da posição 0 (ADC12MEM0). Durante uma sequência, se o ADC12 for desabilitado e novamente habilitado, a sequência recomeça da primeira posição. Os disparos pelo timer devem garantir o tempo necessário para cada conversão.

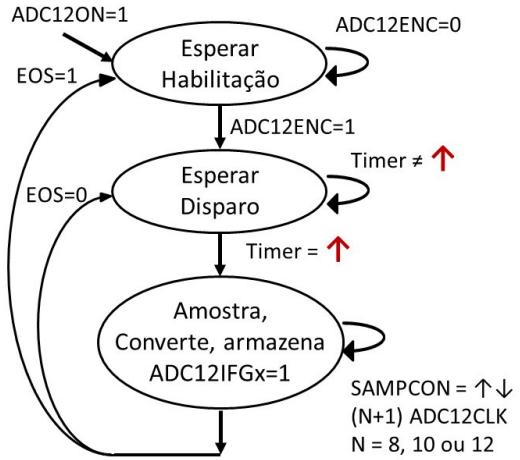


Figura 11.47. Explicação e diagrama de estados para o ADC12 operando no modo sequência de canais sem repetição e conversão disparada pelo timer (TA0.1).

Listagem do Ensaio 4

```

// EN 11.4 - Ensaio
...
TA0_config();
ADC_config();
while(TRUE) {
    ADC12CTL0 |= ADC12ENC; //Habilitar a cada sequência
    while( (ADC12IFG&ADC12IFG3) == 0); //Esperar apenas IFG3
    adc[0]=ADC12MEM0;
    adc[1]=ADC12MEM1;
    adc[2]=ADC12MEM2;
    adc[3]=ADC12MEM3;
}
...
void ADC_config(void){
    ADC12CTL0  &= ~ADC12ENC;           //Desabilitar para configurar
    ADC12CTL0 = ADC12ON;              //Ligar ADC
    ADC12CTL1 = ADC12CONSEQ_1        | //Modo sequência de canais
                    ADC12SHS_1          | //Selecionar TA0.1
                    ADC12CSTARTADD_0     | //Resultado em ADC12MEM0
                    ADC12SSEL_3;          //ADC12CLK = SMCLK
    ADC12MCTL0 = ADC12SREF_0 | ADC12INCH_0; //Config MEM0
    ADC12MCTL1 = ADC12SREF_0 | ADC12INCH_1; //Config MEM1
    ADC12MCTL2 = ADC12SREF_0 | ADC12INCH_2; //Config MEM2
    ADC12MCTL3 = ADC12EOS| ADC12SREF_0 | ADC12INCH_3; //MEM3 = última
    P6SEL |= BIT3|BIT2|BIT1|BIT0;      // Desligar digital de P6.3,2,1,0
    ADC12CTL0 |= ADC12ENC;            //Habilitar ADC12
}
void TA0_config(void){
    TA0CTL = TASSEL_1 | MC_1;
    TA0CCTL1 = OUTMOD_6;             //Out = modo 6
}
  
```

```

TA0CCR0 = 3276;           //100 Hz
TA0CCR1 = TA0CCR0/2;      //Carga 50%
}

```

11.14.5. Ensaio 5

Modo	Repetição	Disparo	CONSEQ	ADC12 SHS	ADC12 SHP	ADC12 MSC
Simples	Sim	ADC12SC	2	0	0	0

No caso de um canal com repetição disparado por software (ADC12SC), fica tudo igual ao Ensaio 1. Em outras palavras, quando se dispara por software (ADC12SC), CONSEQ = 0 e CONSEQ = 2 têm o mesmo resultado.

11.14.6. Ensaio 6

Modo	Repetição	Disparo	CONSEQ	ADC12 SHS	ADC12 SHP	ADC12 MSC
Simples	Sim	TA0.1	2	1	0	0

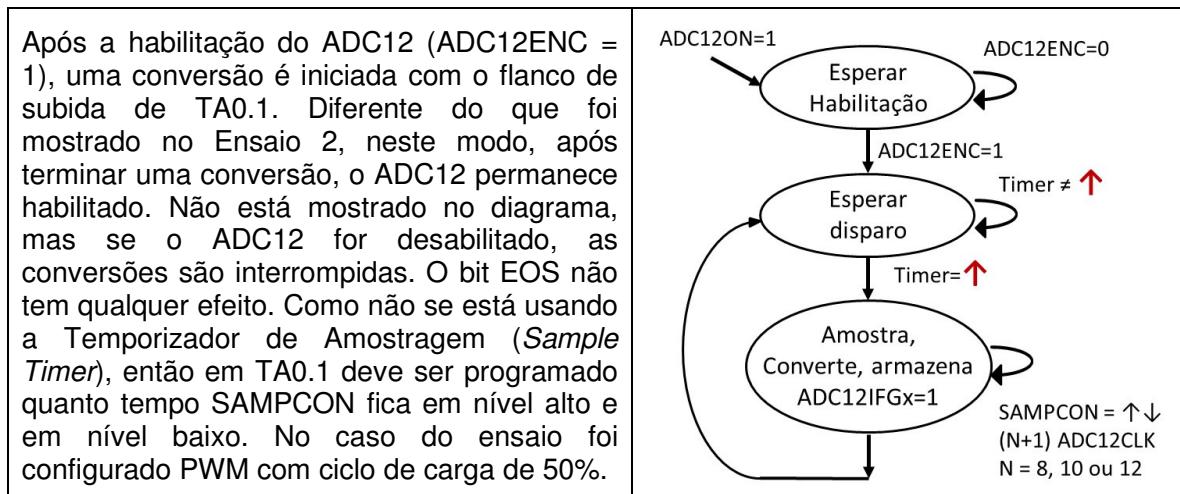


Figura 11.48. Explicação e diagrama de estados para o ADC12 operando no modo um canal com repetição e conversão disparada pelo timer (TA0.1).

Listagem do Ensaio 4

```

// EN 11.6 - Ensaio
...

```

```

TA0_config();
ADC_config();
while(TRUE) {
    while( (ADC12IFG&ADC12IFG0) == 0); //IFG0 ?
    adc=ADC12MEM0;
}
...
void ADC_config(void){
    ADC12CTL0  &= ~ADC12ENC;           //Desabilitar para configurar
    ADC12CTL0 = ADC12ON;             //Ligar ADC
    ADC12CTL1 = ADC12CONSEQ_2      | //Modo único canal repetido
                ADC12SHS_1        | //Selecionar TA0.1
                ADC12CSTARTADD_0 | //Resultado em ADC12MEM0
                ADC12SSEL_3;       //ADC12CLK = SMCLK
    ADC12MCTL0 = ADC12SREF_0 | ADC12INCH_1; //Config MEMO
    P6SEL |= BIT1;                  // Desligar digital de P6.1
    ADC12CTL0 |= ADC12ENC;         //Habilitar ADC12
}

void TA0_config(void){
    TA0CTL = TASSEL_1 | MC_1;
    TA0CCTL1 = OUTMOD_6;          //Out = modo 6
    TA0CCR0 = 3276;              //100 Hz
    TA0CCR1 = TA0CCR0/2;         //Carga 50%
}

```

11.14.7. Ensaio 7

Modo	Repetição	Disparo	CONSEQ	ADC12 SHS	ADC12 SHP	ADC12 MSC
Sequência de canais	Sim	ADC12SC	3	0	0	0

Se o ADC12 está habilitado ($\text{ADC12ENC} = 1$), uma conversão é iniciada com o flanco de subida em ADC12SC . Ao contrário do que afirma o manual, ADC12SC não voltou automaticamente para zero. Isto se repete enquanto a marca EOS não for atingido. Após a última conversão (EOS), a sequência é reiniciada automaticamente. Se a marca EOS = 1 nunca for encontrada, a sequência continua a partir da posição 0 (ADC12MEM0). Durante uma sequência, se o ADC12 for desabilitado e novamente habilitado, a sequência recomeça da primeira posição. Um novo flanco de ADC12SC só devem acontecer após a conversão anterior terminar.

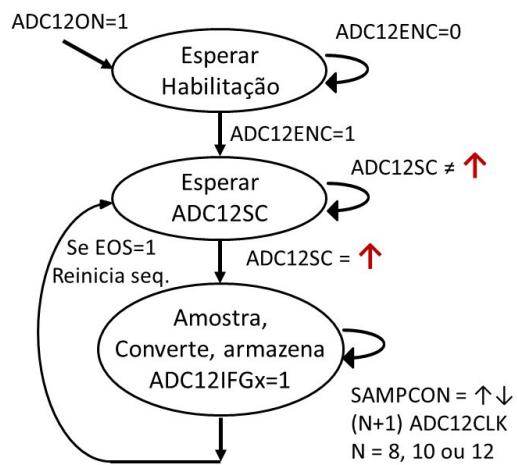


Figura 11.49. Explicação e diagrama de estados para o ADC12 operando no modo sequência de canais sem repetição e conversão disparada por software (ADC12SC).

Listagem do Ensaio 7

```

// EN 11.7 - Ensaio
...
while(TRUE) {
    ADC12CTL0 |= ADC12SC; //Disparar conversão
    ADC12CTL0 &= ~ADC12SC; //Zerar para a próxima
    while( (ADC12IFG&ADC12IFG0) == 0); //IFG0 ?
    adc[0]=ADC12MEM0;
    ;
    ADC12CTL0 |= ADC12SC; //Disparar conversão
    ADC12CTL0 &= ~ADC12SC; //Zerar para a próxima
    while( (ADC12IFG&ADC12IFG1) == 0); //IFG1 ?
    adc[1]=ADC12MEM1;
    ;
    ADC12CTL0 |= ADC12SC; //Disparar conversão
    ADC12CTL0 &= ~ADC12SC; //Zerar para a próxima
    while( (ADC12IFG&ADC12IFG2) == 0); //IFG2 ?
    adc[2]=ADC12MEM2;
    ;
    ADC12CTL0 |= ADC12SC; //Disparar conversão
    ADC12CTL0 &= ~ADC12SC; //Zerar para a próxima
    while( (ADC12IFG&ADC12IFG3) == 0); //IFG3 ?
    adc[3]=ADC12MEM3;
}
...
void ADC_config(void){
    ADC12CTL0  &= ~ADC12ENC;           //Desabilitar para configurar
    ADC12CTL0 = ADC12ON;               //Ligar ADC
    ADC12CTL1 = ADC12SEQ_3 |          //Modo sequência de canais
}

```

```

        ADC12SHS_0           | //Selecionar ADC12SC
        ADC12CSTARTADD_0    | //Resultado em ADC12MEM0
        ADC12SSEL_3;          //ADC12CLK = SMCLK
        ADC12MCTL0 = ADC12SREF_0 | ADC12INCH_0; //Config MEM0
        ADC12MCTL1 = ADC12SREF_0 | ADC12INCH_1; //Config MEM1
        ADC12MCTL2 = ADC12SREF_0 | ADC12INCH_2; //Config MEM2
        ADC12MCTL3 = ADC12EOS| ADC12SREF_0 | ADC12INCH_3; //MEM3=última
        P6SEL |= BIT3|BIT2|BIT1|BIT0; // Desligar digital de P6.3,2,1,0
        ADC12CTL0 |= ADC12ENC; //Habilitar ADC12
    }
}

```

11.14.8. Ensaio 8

Modo	Repetição	Disparo	CONSEQ	ADC12 SHS	ADC12 SHP	ADC12 MSC
Sequência de canais	Sim	TA0.1	3	1	0	0

Se o ADC12 está habilitado (ADC12ENC = 1), uma conversão é iniciada a cada flanco de subida de TA0.1. Quando a marca EOS for atingida, sequência é reiniciada. Se a marca EOS = 1 nunca for encontrada, a sequência continua a partir da posição 0 (ADC12MEM0). Durante uma sequência, se o ADC12 for desabilitado e novamente habilitado, a sequência recomeça da primeira posição. Os disparos pelo timer devem garantir o tempo necessário para cada conversão.

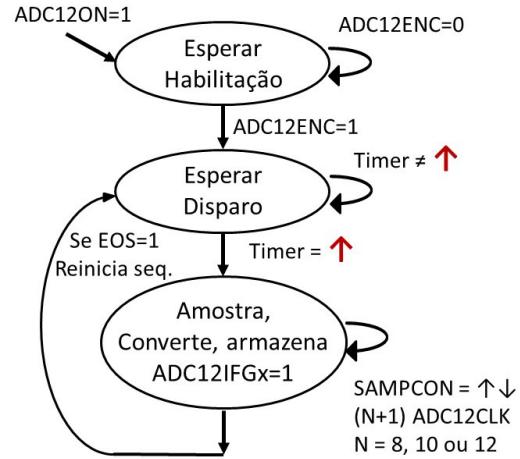


Figura 11.50. Explicação e diagrama de estados para o ADC12 operando no modo sequência de canais sem repetição e conversão disparada pelo timer (TA0.1).

Listagem do Ensaio 8

```

// EN 11.8 - Ensaio
...
while(TRUE) {
    while( (ADC12IFG&ADC12IFG3) == 0); //IFG3? (esperar a última)
    adc[0]=ADC12MEM0;
    adc[1]=ADC12MEM1;
    adc[2]=ADC12MEM2;
}

```

```

        adc[3]=ADC12MEM3;
    }
    ...
void ADC_config(void){
    ADC12CTL0  &= ~ADC12ENC;           //Desabilitar para configurar
    ADC12CTL0 = ADC12ON;              //Ligar ADC
    ADC12CTL1 = ADC12CONSEQ_3        | //Modo sequência de canais
                ADC12SHS_1          | //Selecionar TA0.1
                ADC12CSTARTADD_0     | //Resultado em ADC12MEM0
                ADC12SSEL_3;          //ADC12CLK = SMCLK
    ADC12MCTL0 = ADC12SREF_0 | ADC12INCH_0; //Config MEM0
    ADC12MCTL1 = ADC12SREF_0 | ADC12INCH_1; //Config MEM1
    ADC12MCTL2 = ADC12SREF_0 | ADC12INCH_2; //Config MEM2
    ADC12MCTL3 = ADC12EOS| ADC12SREF_0 | ADC12INCH_3; //MEM3=última
    P6SEL |= BIT3|BIT2|BIT1|BIT0;      // Desligar digital de P6.3,2,1,0
    ADC12CTL0 |= ADC12ENC;            //Habilitar ADC12
}

void TA0_config(void){
    TA0CTL = TASSEL_1 | MC_1;
    TA0CCTL1 = OUTMOD_6;             //Out = modo 6
    TA0CCR0 = 3276;                 //100 Hz
    TA0CCR1 = TA0CCR0/2;            //Carga 50%
}

```

11.14.9. Ensaio 9

Modo	Repetição	Disparo	CONSEQ	ADC12 SHS	ADC12 SHP	ADC12 MSC
Simples	Sim	ADC12SC	2	1	1	1

Para que o modo amostra e converte ($\text{ADC12MSC} = 1$) funcione, é preciso que a unidade de temporização de amostragem esteja selecionada ($\text{ADC12SHP} = 1$), por isso, o programador deve selecionar o período de amostragem adequado. O primeiro ciclo de amostra e converte acontece com o flanco de subida de ADC12SC . Os próximos acontecem automaticamente, assim que o anterior termina. A interrupção dos ciclos de conversão acontece quando o usuário desabilita o ADC ($\text{ADC12ENC} = 0$). O comportamento é semelhante quando se usa um timer. O primeiro flanco inicia os ciclos.

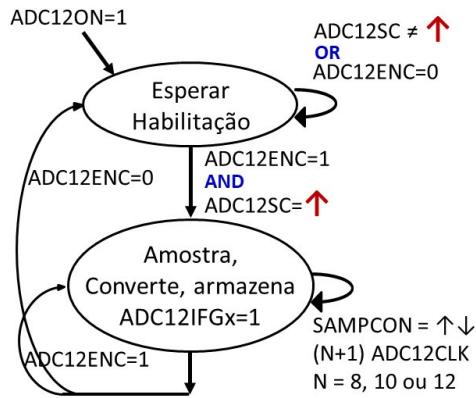


Figura 11.51. Explicação e diagrama de estados para o ADC12 operando com canal simples com repetição e usando o recurso de amostra e converte ($\text{ADC12MSC} = 1$), sendo que a primeira conversão disparada por software (ADC12SC).

Listagem do Ensaio 9

```
// EN 11.9 - Ensaio
...
ADC_config();
ADC12CTL0 |= ADC12SC; //Disparar conversão
ADC12CTL0 &= ~ADC12SC; //Zerar
while(TRUE) {
    while( (ADC12IFG&ADC12IFG0) == 0); //IFG0 ?
    adc=ADC12MEM0;
    cont++; //Contador dos ciclos de conversões
}
...
void ADC_config(void){
    ADC12CTL0  &= ~ADC12ENC; //Desabilitar para configurar
    ADC12CTL0 = ADC12SHT0_4 | //Config temporiz de amostragem
                ADC12MSC | //Múltiplos amostra e converte
                ADC12ON; //Ligar ADC12
    ADC12CTL1 = ADC12CONSEQ_2 | //Modo único canal repetido
                ADC12SHS_0 | //Selecionar ADC12SC
                ADC12SHP | //Usar temporiz de amostragem
                ADC12CSTARTADD_0 | //Resultado em ADC12MEM0
                ADC12SSEL_3; //ADC12CLK = SMCLK
    ADC12MCTL0 = ADC12SREF_0 | ADC12INCH_1; //Config MEMO
    P6SEL |= BIT1; // Desligar digital de P6.1
    ADC12CTL0 |= ADC12ENC; //Habilitar ADC12
}
```

11.14.9. Ensaio 10

Modo	Repetição	Disparo	CONSEQ	ADC12 SHS	ADC12 SHP	ADC12 MSC
Sequência de canais	Não	ADC12SC	1	1	1	1

Para que o modo amostra e converte (ADC12MSC = 1) funcione, é preciso que a unidade de temporização de amostragem esteja selecionada (ADC12SHP = 1), por isso, o programador deve selecionar o período de amostragem adequado. O primeiro ciclo de amostra e converte acontece com o flanco de subida de ADC12SC. Os próximos acontecem automaticamente, até encontrar EOS = 1, quando então o ADC é desabilitado. O comportamento é semelhante quando se usa um timer. O primeiro flanco do timer inicia os ciclos. Se a marca EOS = 1 nunca for encontrada, a sequência continua a partir da posição 0 (ADC12MEM0).

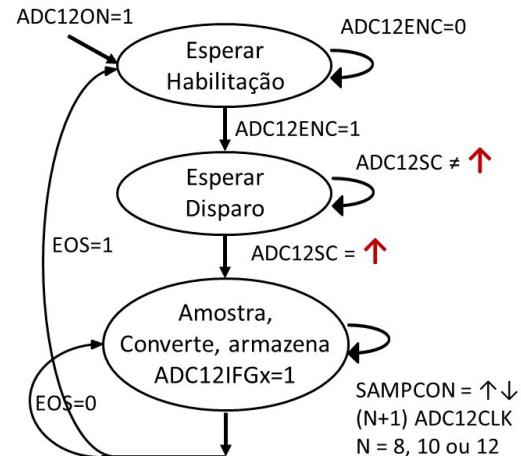


Figura 11.52. Explicação e diagrama de estados para o ADC12 operando com sequência de canais sem repetição e usando o recurso de amostra e converte (ADC12MSC = 1), sendo que a primeira conversão disparada por software (ADC12SC).

Listagem do Ensaio 10

```

// EN 11.10 - Ensaio
...
ADC_config();
while(TRUE) {
    ADC12CTL0 |= ADC12ENC; //Habilitar a cada sequência
    ADC12CTL0 |= ADC12SC; //Disparar conversão
    ADC12CTL0 &= ~ADC12SC; //Zerar para a próxima
    while( (ADC12IFG&ADC12IFG3) == 0); //Última ?
    adc[0]=ADC12MEM0;
    adc[1]=ADC12MEM1;
    adc[2]=ADC12MEM2;
    adc[3]=ADC12MEM3;
    cont++; //Contador indica ciclos de conversão
}
...
void ADC_config(void) {
    ADC12CTL0 &= ~ADC12ENC; //Desabilitar para configurar
    ADC12CTL0 = ADC12SHT0_4 | //Config temporiz de amostragem
                ADC12MSC | //Múltiplos amostra e converte

```

```

        ADC12ON;           //Ligar ADC12
ADC12CTL1 = ADC12CONSEQ_1 | //Modo sequência de canais
              ADC12SHS_0 | //Selecionar ADC12SC
              ADC12SHP | //Usar temporiz de amostragem
              ADC12CSTARTADD_0 | //Resultado em ADC12MEM0
              ADC12SSEL_3;    //ADC12CLK = SMCLK
ADC12MCTL0 = ADC12SREF_0 | ADC12INCH_0; //Config MEM0
ADC12MCTL1 = ADC12SREF_0 | ADC12INCH_1; //Config MEM1
ADC12MCTL2 = ADC12SREF_0 | ADC12INCH_2; //Config MEM2
ADC12MCTL3 = ADC12EOS| ADC12SREF_0 | ADC12INCH_3; //Config MEM3
última
P6SEL |= BIT3|BIT2|BIT1|BIT0; // Desligar digital de P6.3,2,1,0
ADC12CTL0 |= ADC12ENC;       //Habilitar ADC12
}

```

11.15. Exercícios Resolvidos

A seguir apresentamos uma série de exercícios para ilustrar a operação do ADC12_A.

ER 11.1. Escreva um programa que fique em um laço infinito, disparando a conversão por software. Use o modo um canal sem repetição, converta a tensão analógica da entrada A1 (P6.1) e use a memória 5 para receber o resultado da conversão.

Solução:

Este primeiro exercício vai usar o ADC12_A no seu modo mais simples. Uma forma segura de realizar a configuração é preenchendo os três registradores de controle, como mostrado abaixo, junto com as Figuras 11.23, 11.24, 11.28 e 11.34.

ADC12INCH = 1 → Converter o canal A1 (P6.1)

ADC12SHS = 1 → Usar ADC12SC para disparar (disparo por software)

ADC12SHP = 0 → Usar SHI para gerar SAMPCOM

ADC12SSEL = 3 → Usar o SMCLK como origem do ADC12CLK direto

ADC12 CTL0	15	14	13	12	11	10	9	8
	ADC12SHT1				ADC12SHT0			
	0				0			
	7	6	5	4	3	2	1	0
	ADC12 MSC	ADC12 REF2_5V	ADC12 REFON	ADC12 ON	ADC12 OVIE	ADC12 TOVIE	ADC12 ENC	ADC12 SC
	0	0	0	1	0	0	0/1	0

ADC12 CTL1	15	14	13	12	11	10	9	8
	ADC12 CSTARTADD				ADC12 SHS		ADC12 SHP	ADC12 ISSH
	5				0		0	0
	7	6	5	4	3	2	1	0
	ADC12 DIV				ADC12 SSEL		ADC12 CONSEQ	ADC12 BUSY
	0				3		0	0
ADC12 CTL2	15	14	13	12	11	10	9	8
	-	-	-	-	-	-	-	ADC12 PDIV
	-	-	-	-	-	-	-	0
	7	6	5	4	3	2	1	0
	ADC12 TCOFF	-	ADC12 RES		ADC12 DF	ADC12 SR	ADC12 REFOUT	ADC12 REFBURST
	0	-	2		0	0	0	0
ADC1 MCTLx	7	6	5	4	3	2	1	0
	ADC12 EOS	ADC12 SREFx			ADC12 INCHx			
	0	0			1			

Figura 11.44. Configuração dos registradores de controle do ADC12.

Sabendo os valores a serem programados nos diversos registradores, todo o resto fica simples, como mostrado na listagem abaixo. Note que após a configuração, o programa fica num laço infinito, disparando as conversões e lendo o resultado na variável `adc`, que foi declarada com `volatile` para evitar a otimização do CCS. O disparo da conversão está marcado em vermelho. Foi notada uma disparidade em relação ao manual que afirma que o bit ADC12SC é automaticamente ressetado. Bem, isto não aconteceu. Foi necessário adicionar uma linha só para zerá-lo. O leitor pode fazer ensaios com essas duas linhas em vermelho para comprovar a afirmação.

Com a entrada A1 em aberto, o resultado varia bastante, mas deve ser um valor próximo de 2.000 (4.096 / 2). Ao conectar a entrada A1 à terra (GND) o resultado da conversão deve ficar muito próximo de zero. Ao conectar essa entrada ao 3,3 V, o resultado da conversão deve ficar próximo de 4.095 (0xFFFF). Cuidado, nunca ligue a entrada aos 5 V. Pode queimar o diodo de proteção da entrada. Para finalizar, note que foi ativado o bit 1

do registrador P6SEL (`P6SEL |= BIT1`), com isso foi desabilitada a parte digital deste pino, favorecendo a economia de energia.

Listagem da solução do ER 11.1

```
// ER 11.1
// Converter entrada A1 em ADC12MEM5
// Disparar conversão por software (ADC12SC)

#include <msp430.h>

#define TRUE 1
void ADC_config(void);

int main(void){

    volatile int adc;
    WDTCTL = WDTPW | WDTHOLD;      // stop watchdog timer
    ADC_config();
    while(TRUE){
        ADC12CTL0 |= ADC12SC;      //Disparar conversão
        ADC12CTL0 &= ~ADC12SC;    //Disparar conversão
        while( (ADC12IFG&ADC12IFG5) == 0 );
        adc=ADC12MEM5;
    }
    return 0;
}

void ADC_config(void){
    ADC12CTL0  &= ~ADC12ENC;           //Desabilitar para configurar

    ADC12CTL0 = ADC12ON;              //Ligar ADC

    ADC12CTL1 = ADC12CSTARTADD_5 |   //Resultado em ADC12MEM5
                ADC12SHS_0       |   // Selecionar ADC12SC bit
                ADC12DIV_0        |   //Clock ADC Divisor = 1
                ADC12SSEL_3        |   //Clock ADC = SMCLK
                ADC12CONSEQ_0;       //Modo Single

    ADC12CTL2  = ADC12TCOFF |        // Desligar sensor temperatura
                ADC12RES_2;          // Resolução 12-bit

    ADC12MCTL5 = ADC12EOS     |    //Fim de sequência
                ADC12SREF_0    |    //VR+ = AVCC e VR- = AVSS
                ADC12INCH_1;      ////(P6.1) A1 = Canal 1
    P6SEL |= BIT1;                  // Desligar parte digital de P6.1

    ADC12CTL0 |= ADC12ENC;          //Habilitar ADC12
```

{}

Curiosidade: O leitor pode estar se perguntando o porquê desse exercício usar a entrada analógica A1 e não a A0. A explicação é simples: danificamos essa entrada A0! Num momento de descuido, ao invés do pino de 3,3 V a ligamos ao pino de 5 V. Esses dois pinos são vizinhos. No nosso caso, a entrada A0 (P6.0) está em curto com a terra.

ER 11.2. Escreva um programa que fique em um laço infinito, disparando a conversão por software. Use o modo sequência de canais sem repetição, convertendo as entradas A0, A1, A2 e A3 (P6.0, 1, 2, 3) e armazene os resultados a partir da memória 0.

Solução:

Este exercício pretende ilustrar o uso da sequência de canais sem repetição. A sequência de canais é conseguida com a indicação, no campo ADC12CSTARTADD, do primeiro endereço de memória a ser usada. O ADC12 vai convertendo e percorrendo a memória em sequência até encontrar uma posição de memória marca com sendo a última (EOS = 1). Se nenhuma memória estiver com EOS = 1, a sequência continua a partir do endereço 0.

Como é sem repetição, cada vez que completa um ciclo, o bit de habilitação do ADC12 (ADC12ENC) é zerado. Isto significa que para repetir a mesma sequência, o programa precisa colocar este bit em 1. Para cada conversão, o bit ADC12SC precisa ser colocado em 1 e depois em zero. (**isto conflita com o manual que afirma que ADC12SC é zerado automaticamente**).

A listagem abaixo apresenta a solução. O leitor deve usar os recursos do CCS para verificar os resultados das conversões. Use um cabo conectado à terra, para levar a zero o resultado da conversão de um canal. Isto irá confirmar o funcionamento do ADC12. Não conecte os pinos à tensão de 5 V.

Listagem da solução do ER 11.2

```
// ER 11.2
// Usar sequência de canais sem repetição
// Converter entradas A0, A1, A2 e A3
// Armazenar em ADC12MEM0, 1, 2 e 3
// Disparar conversão por software (ADC12SC)

#include <msp430.h>

#define TRUE 1
void ADC_config(void);

int main(void){
```

```

volatile int adc[4];
WDTCTL = WDTPW | WDTHOLD; // stop watchdog timer
ADC_config();
while(TRUE){

    // A cada repetição é preciso habilitar o ADC12
    ADC12CTL0 |= ADC12ENC; //Habilitar ADC12

    // 0) Disparra conversão
    ADC12CTL0 |= ADC12SC; //Disparar conversão
    ADC12CTL0 &= ~ADC12SC; //Zerar ADC12SC para a próxima
    while( (ADC12IFG&ADC12IFG0) == 0); //Esperar IFG0

    // 1) Disparar conversão
    ADC12CTL0 |= ADC12SC; //Disparar conversão
    ADC12CTL0 &= ~ADC12SC; //Zerar ADC12SC para a próxima
    while( (ADC12IFG&ADC12IFG1) == 0); //Esperar IFG1

    // 2) Disparar conversão
    ADC12CTL0 |= ADC12SC; //Disparar conversão
    ADC12CTL0 &= ~ADC12SC; //Zerar ADC12SC para a próxima
    while( (ADC12IFG&ADC12IFG2) == 0); //Esperar IFG2

    // 3) Disparar conversão
    ADC12CTL0 |= ADC12SC; //Disparar conversão
    ADC12CTL0 &= ~ADC12SC; //Zerar ADC12SC para a próxima
    while( (ADC12IFG&ADC12IFG3) == 0); //Esperar IFG3 (última)

    // Ler o resultados, apaga IFG0,1,2,3
    adc[0]=ADC12MEM0;
    adc[1]=ADC12MEM1;
    adc[2]=ADC12MEM2;
    adc[3]=ADC12MEM3;
}

return 0;
}

void ADC_config(void){
    ADC12CTL0 &= ~ADC12ENC; //Desabilitar para configurar

    ADC12CTL0 = ADC12ON; //Ligar ADC

    ADC12CTL1 = ADC12CSTARTADD_0 | //Resultado em ADC12MEM0
                ADC12SHS_0 | //Selecionar ADC12SC bit
                ADC12DIV_0 | //Clock ADC Divisor = 1
                ADC12SSEL_3 | //Clock ADC = SMCLK
                ADC12CONSEQ_1; //Sequência de canais

    ADC12CTL2 = ADC12TCOFF | // Desligar sensor temperatura
                ADC12RES_2; // Resolução 12-bit
}

```

```

ADC12MCTL0 = ADC12SREF_0 | ADC12INCH_0;      //MEM0 = A0 (P6.0)
ADC12MCTL1 = ADC12SREF_0 | ADC12INCH_1;      //MEM1 = A1 (P6.1)
ADC12MCTL2 = ADC12SREF_0 | ADC12INCH_2;      //MEM2 = A2 (P6.2)
ADC12MCTL3 = ADC12EOS | ADC12SREF_0 | ADC12INCH_3; //MEM3=A3 (P6.3)
//Notar que MEM3 é a última posição pois tem EOS=1

P6SEL |= (BIT3|BIT2|BIT1|BIT0); //Desligar digital de P6.0,1,2,3

ADC12CTL0 |= ADC12ENC;           //Habilitar ADC12
}

```

ER 11.3. Neste exercício vamos empregar um joystick, como mostrado na figura abaixo. É pedido para fazer 500 conversões de 8 bits, na taxa de 100 Hz por canal e mostrar o resultado usando do recurso gráfico do CCS.

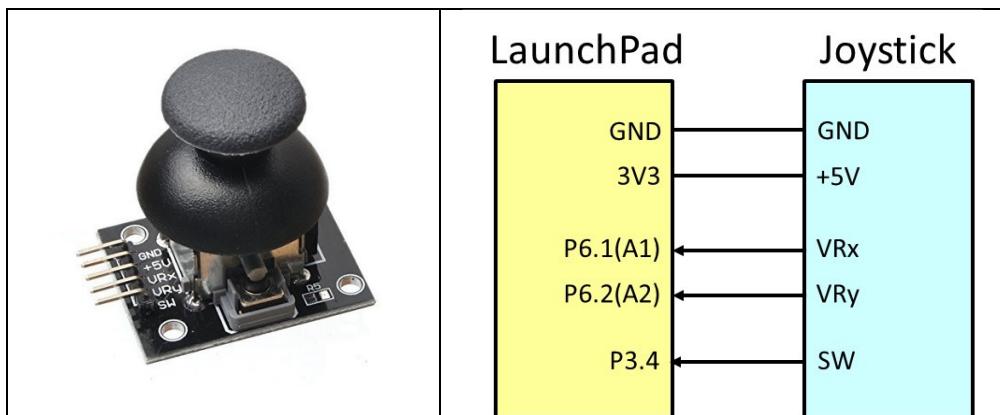


Figura 11.52. Foto do joystick a ser usado e indicação da conexão com a LaunchPad. Note que o pino do joystick marcada com “+5V” deve ser conectado aos 3,3 V. Não o ligue aos 5 V, pois pode queimar sua entrada analógica.

Solução:

Neste exercício vamos usar os potenciômetros de um joystick para experimentar os recursos do ADC12. A figura abaixo apresenta se circuito, que é muito simples.

Ao lado está o esquema elétrico do joystick. São dois potenciômetros de $10\text{ k}\Omega$, um para cada eixo. A tomada central varia de acordo com a posição da alavanca. Ele deve ser alimentado com 3,3 V, para que as tensões fiquem dentro da faixa do MSP430. O dispositivo ainda traz uma pequena chave que é acionada ao se pressionar a alavanca e faz um curto para a terra. Com este experimento, o leitor poderá constatar que esses joysticks têm baixa qualidade e que uma pequena variação na alavanca leva a valores extremos.

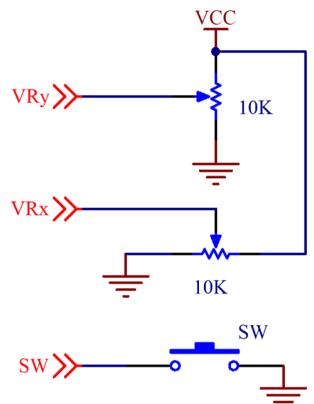


Figura 11.53. Esquema do joystick e uma pequena explicação de seu funcionamento.

O programa é relativamente simples. Vamos usar o modo sequência de canais com repetição para converter as entradas A1 (P6.1) e A2 (P6.2). Para que fique fácil de lembrar, vamos usar as memórias 1 e 2, sendo que a memória 2 vai estar com seu bit EOC ativado. Assim, depois de escrever na memória 2, o ADC12 volta a usar a memória 1 e assim vai, alternadamente.

- A1 (P6.1) é convertido na memória ADC12MEM1 e
- A2 (P6.2) é convertido na memória ADC12MEM2, que está com EOC = 1.

O pedido é para digitalizar em 100 Hz cada canal. Como são dois canais, vamos usar o TA0.1 para gerar disparos em 200 Hz. Para tornar a solução mais interessante, usaremos a interrupção. Como são apenas dois canais, será habilitada apenas a interrupção da escrita na memória 2. Quando esta interrupção acontecer, faremos a leitura das duas posições usadas.

A listagem a seguir apresenta a solução. Logo após o ressete, o ADC12 está configurado para trabalhar em 12 bits (ADC12RES = 2). O pedido é para converter em 8 bits, foi necessário zerar o registrador de controle 2 (ADC12CTL2) e com isso, alteramos para 8 bits (ADC12RES = 0). Como estamos usando interrupção, foram criados dois vetores globais ($\text{VRx}[]$ e $\text{VRy}[]$) e dois indexadores (ix e iy). A quantidade de aquisições está definida pela constante QTD.

O leitor pode notar que o programa fica preso num laço principal aguardando o usuário acionar a chave SW do joystick. Após este acionamento, o led vermelho é acesso e assim permanece durante todo o período de aquisição. O programa principal apenas fica monitorando o indexador iy para saber o instante correto de interromper a conversão, o que é feito com a desabilitação do ADC (ADC12ENC = 0) e apagar o led vermelho. Assim, quando o led vermelho apaga, o leitor pode usar os recursos do CCS para examinar os dois vetores, ou ainda, clicar com o botão direito do mouse sobre o vetor e pedir para apresentar o conteúdo do vetor de forma gráfica (opção *Graph*).

Listagem da solução do ER 11.3

```
// ER 11.3

// Ler sinais de um joystick
// Armazenar em ADC12MEM1 e 2
// Conversão em 100 Hz por canal
// Usar TA0.1

#include <msp430.h>

#define TRUE 1
#define QTD 500 //Nr de amostras por canal

void ADC_config(void);
void TA0_config(void);
void GPIO_config(void);
void led_VM(void);
void led_vm(void);

// Variáveis globais
volatile char vrz[QTD]; //Vetor x
volatile char vry[QTD]; //Vetor y
volatile int ix, iy; //Indexadores

int main(void){
    WDTCTL = WDTPW | WDTHOLD; // stop watchdog timer
    GPIO_config();
    TA0_config();
    ADC_config();
    __enable_interrupt();
    while(TRUE){
        while ((P3IN&BIT4) == BIT4); //Esperar SW
        led_VM(); //Acender vermelho
        ix=iy=0; //Zerar indexadores
        ADC12CTL0 |= ADC12ENC; //Habilitar ADC
        while (iy<QTD); //Esperar QTD conversões
        ADC12CTL0 &= ~ADC12ENC; //Parar as conversões
        led_vm(); //Apagar vermelho
    }
    return 0;
}

//#pragma vector = 54
#pragma vector = ADC12_VECTOR
__interrupt void adc_int(void){
    vrz[ix++]=ADC12MEM1;
    vry[iy++]=ADC12MEM2;
}
```

```

void ADC_config(void) {
    ADC12CTL0  &= ~ADC12ENC;           //Desabilitar para configurar
    ADC12CTL0 = ADC12ON;             //Ligar ADC
    ADC12CTL1 = ADC12CONSEQ_3      | //Modo sequência de canais
                ADC12SHS_1        | //Selecionar TA0.1
                ADC12CSTARTADD_1 | //Resultado em ADC12MEM1
                ADC12SSEL_3;       //ADC12CLK = SMCLK
    ADC12CTL2 = 0;                  //ADC12RES=0, Modo 8 bits
    ADC12MCTL1 = ADC12SREF_0 | ADC12INCH_1; //Config MEM1
    ADC12MCTL2 = ADC12EOS| ADC12SREF_0 | ADC12INCH_2; //MEM2 = última
    P6SEL |= BIT2|BIT1;            // Desligar digital de P6.2,1
    ADC12CTL0 |= ADC12ENC;         //Habilitar ADC12
    ADC12IE |= ADC12IE2;          //Hab interrupção MEM2
}

void TA0_config(void) {
    TA0CTL = TASSEL_1 | MC_1;
    TA0CCTL1 = OUTMOD_6;          //Out = modo 6
    TA0CCR0 = 32767/200;          //200 Hz (100 Hz por canal)
    TA0CCR1 = TA0CCR0/2;          //Carga 50%
}

void GPIO_config(void) {
    P1DIR |= BIT0;   //Led vermelho
    P1OUT &= ~BIT0;
    P3DIR &= ~BIT4; //P3.4 = SW
    P3REN |= BIT4;
    P3OUT |= BIT4;
}

void led_VM(void){ P1OUT |= BIT0; } //Acender Verm
void led_vm(void){ P1OUT &= ~BIT0; } //Apagar Verm

```

ER 11.4. Vamos aproveitar as conexões feitas no ER 11.3 para avaliar a relação entre a posição da alavanca do joystick e o resultado da conversão. Para tanto, vamos controlar os leds da LaunchPad de acordo com a tabela abaixo, onde 1 e 0 indicam led aceso e apagado, respectivamente. Para evitar ruído, vamos atualizar os leds após a média de 8 conversões por canal. Continuamos a converter em 100 Hz por canal e em 8 bits. A chave SW do joystick alterna entre VRx e VRy. O programa deve iniciar mostrando o resultado para VRx.

Tabela 11.15. Comportamento dos leds para o ER 11.4

Resultado da Conversão	Led 1 (P1.0) Vermelho	Led 2 (P4.7) Verde
0x00 → 0x3F	0	0

0x40 → 0x7F	0	1
0x80 → 0xBF	1	0
0xC0 → 0xFF	1	1

Solução:

Este exercício ilustra muito bem a vantagem do modo sequência de canais com repetição. Vamos programar o ADC para fazer 8 conversões sequenciais do canal A1 e 8 conversões sequenciais do canal A2, usando as memórias ADC12MEM0 até ADC12MEM15. Após terminar as 16 conversões, basta somar as 8 posições de cada canal e calcular a média. Vamos habilitar somente a interrupção da memória 15. Assim, quando ela acontece, as demais conversões já foram realizadas e basta então calcular a média. O programa principal, apenas monitora a chave SW e chama a função que atualiza os leds de acordo com o eixo do joystick que está selecionado. As variáveis media_x e media_y são atualizadas pela interrupção.

É importante explicar que na configuração do ADC, realizada pela função ADC_config(), para evitar as 16 aborrecidas escritas nos registradores de controle de memória, como mostrado na listagem abaixo, elas foram compactadas com o uso de ponteiro. O mesmo foi feito para ler o resultado e calcular a média. Note ainda que no cálculo desta média, para dividir por 8, usamos o deslocamento de 3 bits para a direita (media = soma >> 3). Veja os trechos marcados em abóbora. Note que os ponteiros devem seguir o tamanho do registrador.

volatile unsigned char *pt → para os ADC12MCTLx, que são de 8 bits e
 volatile unsigned int *pt → para os ADC12MEMx, que são de 16 bits e

```

ADC12MCTL0 = ADC12SREF_0 | ADC12INCH_1;
ADC12MCTL1 = ADC12SREF_0 | ADC12INCH_1;
ADC12MCTL2 = ADC12SREF_0 | ADC12INCH_1;
ADC12MCTL3 = ADC12SREF_0 | ADC12INCH_1;
ADC12MCTL4 = ADC12SREF_0 | ADC12INCH_1;
ADC12MCTL5 = ADC12SREF_0 | ADC12INCH_1;
ADC12MCTL6 = ADC12SREF_0 | ADC12INCH_1;
ADC12MCTL7 = ADC12SREF_0 | ADC12INCH_1;
ADC12MCTL8 = ADC12SREF_0 | ADC12INCH_2;
ADC12MCTL9 = ADC12SREF_0 | ADC12INCH_2;
ADC12MCTL10= ADC12SREF_0 | ADC12INCH_2;
ADC12MCTL11= ADC12SREF_0 | ADC12INCH_2;
ADC12MCTL12= ADC12SREF_0 | ADC12INCH_2;
ADC12MCTL13= ADC12SREF_0 | ADC12INCH_2;
ADC12MCTL14= ADC12SREF_0 | ADC12INCH_2;
ADC12MCTL15= ADC12EOS | ADC12SREF_0 | ADC12INCH_2;
  
```

A listagem abaixo apresenta a solução. O programa principal fica preso num laço infinito, apenas chamando a função `leds (media_x >> 6) ou leds (media_y >> 6)`. Para o caso do presente problema, só temos interesse nos 2 bits mais significativos da média, por isso, o argumento é deslocado 6 posições para a direita. A função que monitora a chave SW que está ligada em P3.4 é a mesma que criamos no ER 5.3. Um ponto a ser comentado é que o ADC12, logo após o reset, é configurado para 12 bits. Para colocá-lo no modo 8 bits, zeramos o registrador ADC12CTL2.

Agora o leitor pode checar seu joystick e ver que ele não é linear. Um pequeno deslocamento a alavanco o leva para valores extremos. É sugerido que o leitor insira um break point em algum lugar do laço principal e habilite o modo “refresh all Windows”. Assim, será possível observar instantaneamente as médias calculadas.

Listagem da solução do ER 11.4

```
// ER 11.4
// Acender leds de acordo com conversão
// 0x00 ==> 0x3F: led1 led2
// 0x3F ==> 0x7F: led1 LED2
// 0x80 ==> 0xBF: LED1 led2
// 0xC0 ==> 0xFF: LED1 LED2
// SW alterna entre VRx e VRy
// Usar TA0.1 para disparar a conversão em 100 Hz

#include <msp430.h>

#define FALSE    0
#define TRUE     1
#define VRX      0 //Indicar canal VRx
#define VRY      1 //Indicar canal VRy
#define FECHADA  0 //SW fechada
#define ABERTA   1 //SW aberta
#define DBC      1000 //Debounce

// Funções
void leds(char x);
int sw_mon(void);
void ADC_config(void);
void TA0_config(void);
void GPIO_config(void);
void led_VM(void);
void led_vm(void);
void led_VD(void);
void led_vd(void);
void debounce(int valor);

// Variáveis globais
```

```

volatile char media_x, media_y;

int main(void){
    char canal=VRX;
    WDTCTL = WDTPW | WDTHOLD;      // stop watchdog timer
    GPIO_config();
    TA0_config();
    ADC_config();
    __enable_interrupt();
    while(TRUE){
        if (canal == VRX)    leds(media_x >> 6);
        else                  leds(media_y >> 6);
        if (sw_mon() == TRUE)  canal ^= 1; //Inverter
    }
    return 0;
}

// Controlar os leds de acordo com x
void leds(char x){
    switch(x){
        case 0: led_vM();   led_vd();   break;      //0
        case 1: led_vM();   led_VD();   break;      //1
        case 2: led_VM();   led_vd();   break;      //2
        case 3: led_VM();   led_VD();   break;      //3
    }
}

//#pragma vector = 54
#pragma vector = ADC12_VECTOR
__interrupt void adc_int(void){
    volatile unsigned int *pt;
    unsigned int i,soma;
    pt = &ADC12MEM0;

    soma = 0;
    for (i=0; i<8; i++) soma +=pt[i];
    media_x = soma >>3;

    soma = 0;
    for (i=8; i<16; i++) soma +=pt[i];
    media_y = soma >>3;
}

// Monitorar SW (P3.4), retorna TRUE se foi acionada
int sw_mon(void){
    static int psw=ABERTA;           //Guardar passado de Sw
    if ( (P3IN&BIT4) == 0){         //Qual estado atual de Sw?
        if (psw==ABERTA){          //Qual o passado de Sw?
            debounce(DBC);
            psw=FECHADA;
        }
    }
}

```

```

        return TRUE;
    }
}
else{
    if (psw==FECHADA) {           //Qual o passado de Sw?
        debounce(DBC);
        psw=ABERTA;
        return FALSE;
    }
}
return FALSE;
}

void ADC_config(void){
    volatile unsigned char *pt;
    unsigned char i;
    ADC12CTL0  &= ~ADC12ENC;          //Desabilitar para configurar
    ADC12CTL0 = ADC12ON;             //Ligar ADC
    ADC12CTL1 = ADC12CONSEQ_3 |     //Modo sequência de canais repetido
                ADC12SHS_1 |         //Selecionar TA0.1
                ADC12CSTARTADD_0 |   //Resultado a partir de ADC12MEM0
                ADC12SSEL_3;         //ADC12CLK = SMCLK
    ADC12CTL2 = 0;                  //ADC12RES=0, Modo 8 bits

    pt=&ADC12MCTL0;
    for (i=0; i<8; i++)
        pt[i]=ADC12SREF_0 | ADC12INCH_1; //ADC12MCTL0 até ADC12MCTL7
    for (i=8; i<16; i++)
        pt[i]=ADC12SREF_0 | ADC12INCH_2; //ADC12MCTL8 até ADC12MCTL15
    pt[15] |= ADC12EOS;              //EOS em ADC12MCTL15

    P6SEL |= BIT2|BIT1;            // Desligar digital de P6.2,1
    ADC12CTL0 |= ADC12ENC;         //Habilitar ADC12
    ADC12IE |= ADC12IE15;         //Hab interrupção MEM2
}

void TA0_config(void){
    TA0CTL = TASSEL_1 | MC_1;
    TA0CCTL1 = OUTMOD_6;           //Out = modo 6
    TA0CCR0 = 32767/200;           //200 Hz (100 Hz por canal)
    TA0CCR1 = TA0CCR0/2;           //Carga 50%
}

void GPIO_config(void){
    P1DIR |= BIT0; P1OUT &= ~BIT0; //Led Vermelho
    P4DIR |= BIT7; P4OUT &= ~BIT7; //Led Verde
    P3DIR &= ~BIT4; //P3.4 = SW
    P3REN |= BIT4;
    P3OUT |= BIT4;
}
}

```

```

void led_VM(void){ P1OUT |= BIT0; } //Acender Verm
void led_vm(void){ P1OUT &= ~BIT0; } //Apagar Verm
void led_VD(void){ P4OUT |= BIT7; } //Acender Verde
void led_vd(void){ P4OUT &= ~BIT7; } //Apagar Verde

// Debounce
void debounce(int valor){
    volatile int x; //volatile evita optimizador
    for (x=0; x<valor; x++); //Apenas gasta tempo
}

```

ER 11.5. Teclado com divisor resistivo Teclas com divisor resistivo.

ER 11.6. Este exercício é bastante audacioso e vai fornecer soluções para diversos casos. Digitalizar um sinal de voz a 8 kHz, com resolução de 8 bits e passar o resultado para o Matlab via canal Bluetooth. Para tanto use um microfone de eletreto e um pré-amplificador.

Solução:

Este exercício ilustra muito bem a vantagem do modo sequência de canais com repetição. Vamos programar o ADC para fazer 8 conversões sequenciais do canal A1 e 8

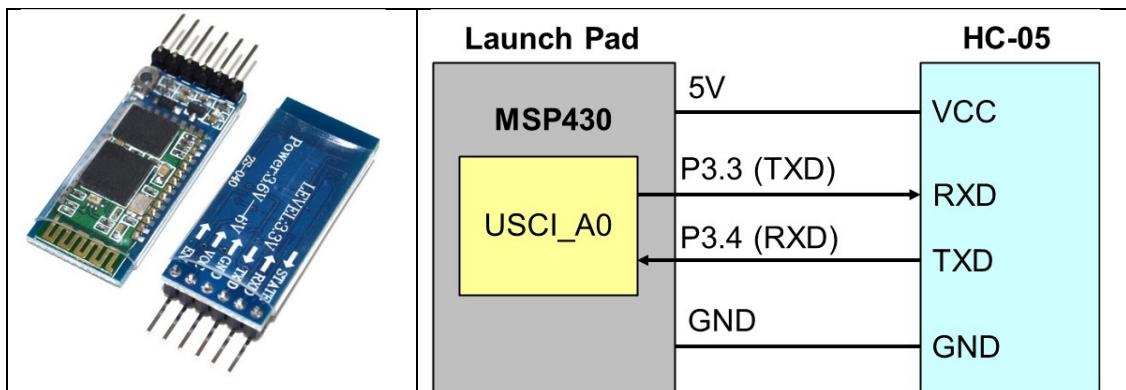


Figura 8.40. Foto do Módulo Bluetooth HC-05 e sua conexão com a LaunchPad.

ER 11.x. Vamos repetir ER 11.4, mas agora, além de acender os leds, vamos também apresentar no LCD o valor hexadecimal da leitura de canal. Na linha superior a média de VRx e na linha inferior a média de VRy.

ER 11.x Adquirir sinal de voz.

ER 11.7. Adquirir sinal de voz e passar para o Matlab.

11.16. Exercícios Propostos

Apresentamos a seguir uma lista com diversos exercícios para que o leitor pratique o que foi estudado sobre o conversor ADC. A lista é grande. Sempre que possível, verifique sua solução no LaunchPad. É recomendado cuidado com os limites de tensão que cada pino, ou seja, apresenta sinais na faixa de 0,0 V até 3,3 V.

EP 11.1. Escreva um programa que fique em um laço infinito, disparando a conversão por software (ADC12SC). Use o modo um canal sem repetição para converter a entrada A0 (P6.0). Use a memória 0 para receber o resultado da conversão. Com um cabo você pode ligar alternadamente o pino P6.0 à terra ou ao 3,3 V para verificar os resultados das conversões. Cuidado para não ligar esse pino aos 5V.

EP 11.2. Este exercício é semelhante ao ER 11.2. Escreva um programa que fique em um laço infinito, disparando a conversão por software (ADC12SC). Use o modo sequência de canais sem repetição para converter as entradas A0, A1, A2 e A3 (P6.0, 1, 2, 3). As memórias devem ser usadas da forma listada abaixo. Com um cabo você pode ligar os pinos P6.0, 1, 2 e 3 à terra ou ao 3,3 V para verificar os resultados das conversões. Cuidado para não ligar esse pino aos 5V.

- ADC12MEM0 = conversão da entrada A3 (P6.3);
- ADC12MEM1 = conversão da entrada A2 (P6.2);
- ADC12MEM2 = conversão da entrada A1 (P6.1);
- ADC12MEM3 = conversão da entrada A0 (P6.0);

EP 11.3. Escreva um programa que fique em um laço infinito, esperando as conversões que são disparadas pelo TA0.1 programado em 10 Hz. Programe o modo um canal sem repetição para converter a entrada A0 (P6.0). Use a memória 0 para receber o resultado da conversão. Com um cabo você pode ligar alternadamente o pino P6.0 à terra ou ao 3,3 V para verificar os resultados das conversões. Cuidado para não ligar esse pino aos 5V.

EP 11.4. Escreva um programa que fique em um laço infinito, esperando as conversões que são disparadas pelo TA0.1. Programe o modo sequência de canais sem repetição para converter as entradas A0, A1, A2 e A3 (P6.0, 1, 2, 3). As memórias devem ser usadas da forma indicada abaixo. Com um cabo você pode ligar os pinos P6.0, 1, 2 e 3 à terra ou ao 3,3 V para verificar os resultados das conversões. Cuidado para não ligar esse pino aos 5V.

- ADC12MEM0 = conversão da entrada A3 (P6.3);
- ADC12MEM1 = conversão da entrada A2 (P6.2);
- ADC12MEM2 = conversão da entrada A1 (P6.1);
- ADC12MEM3 = conversão da entrada A0 (P6.0);

EP 11.5. Escreva um programa que fique em um laço infinito, esperando as conversões que são disparadas pelo TA0.1 programado em 10 Hz. Programe o modo um canal com repetição para converter a entrada A0 (P6.0). Use a memória 0 para receber o resultado da conversão. Compare com a solução do EP 11.3. Verifique os resultados das conversões.

EP 11.6. Escreva um programa que fique em um laço infinito, esperando as conversões que são disparadas pelo TA0.1. Programe o modo sequência de canais com repetição para converter as entradas A0, A1, A2 e A3 (P6.0, 1, 2, 3). As memórias devem ser usadas da forma indicada abaixo. Compare com a solução do EP 11.3. Verifique os resultados das conversões.

- ADC12MEM0 = conversão da entrada A3 (P6.3);
- ADC12MEM1 = conversão da entrada A2 (P6.2);
- ADC12MEM2 = conversão da entrada A1 (P6.1);
- ADC12MEM3 = conversão da entrada A0 (P6.0);

EP 11.7. Neste exercício vamos usar um joystick, como mostrado na Figura 11.52, para controlar o brilho dos leds vermelho e verde. O eixo X controla o brilho do led vermelho e o eixo Y o do led verde. Programe o ADC12 para disparar as conversões pelo TA0.1 programado em 100 Hz. Para controlar o brilho dos leds use também o TA0 para gerar PWM com período de 10 ms e 100 passos de ciclo de carga.

EP 11.8. Neste exercício vamos substituir o joystick por um potenciômetro de 10 kΩ. Vamos ligar o terminal central à entrada P6.0 (A0) e usar um voltímetro para conferir as tensões. Crie uma variável float para mostrar o resultado da conversão em Volts e confira com o indicado pelo multímetro. Faça ensaios usando resolução de 8, 10 e 12 bits.

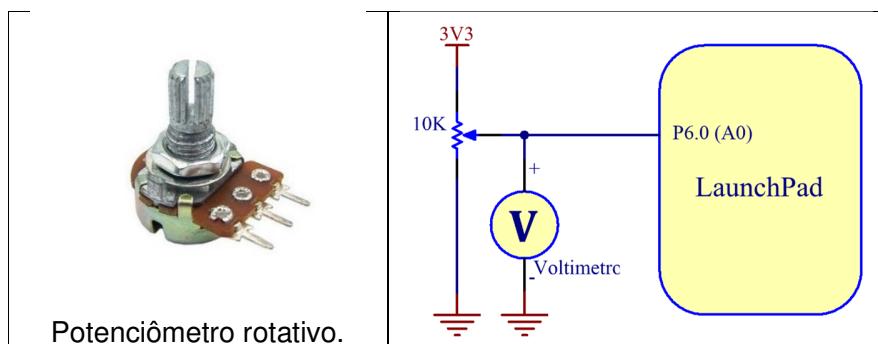


Figura 8.xx. Uso de um potenciômetro para gerar tensão analógica a ser convertida pelo ADC12 e de um voltímetro para conferir os resultados.

EP 11.9. Iluminar led para gerar tensão.

EP 11.10. Usar LDR

EP 11.11. Usar um fototransistor

Usar LDR

Neste exercício vamos substituir o joystick por um potenciômetro de $10\text{ k}\Omega$.

Colocar com um apêndice
Gabarito para configurar os registradores do ADC12_A

ADC12 CTL0	15	14	13	12	11	10	9	8
	ADC12SHT1					ADC12SHT0		
	7	6	5	4	3	2	1	0
	ADC12 MSC	ADC12 REF2_5V	ADC12 REFON	ADC12 ON	ADC12 OVIE	ADC12 TOVIE	ADC12 ENC	ADC12 SC

ADC12 CTL1	15	14	13	12	11	10	9	8
	ADC12 CSTARTADD				ADC12 SHS		ADC12 SHP	ADC12 ISSH
	7	6	5	4	3	2	1	0
	ADC12 DIV			ADC12 SSEL		ADC12 CONSEQ		ADC12 BUSY

ADC12 CTL2	15	14	13	12	11	10	9	8
	-	-	-	-	-	-	-	ADC12 PDIV
	-	-	-	-	-	-	-	
	7	6	5	4	3	2	1	0
	ADC12 TCOFF	-	ADC12 RES		ADC12 DF	ADC12 SR	ADC12 REFOUT	ADC12 REFBURST
		-						

ADC1 MCTLx	7	6	5	4	3	2	1	0
	ADC12 EOS	ADC12 SREFx			ADC12 INCHx			

	7	6	5	4	3	2	1	0
UCBxCTL 0	UCA 10	UCSLA1 0	UCMM	-	UCMST	UCMODE		UCSYN C
				0				
UCBxCTL 1	UCSSEL		-	UCTR	UCTXNA CK	UCTXS TP	UCTXS TT	UCWR ST
			0					
UCBxSTA T	-	UCSLL OW	UCGC	UCBBU SY	-	-	-	-
	0				0	0	0	0
UCBxIE	-	-	UCNACKI E	UCALIE	UCSTPIE	UCSTTI E	UCTXIE	UCRXI E
	0	0						
UCBxIFG	-	-	UCNACKI FG	UCALIF G	UCSTPIF G	UCSTTI FG	UCTXIF G	UCRXI FG
	0	0						
UCBxRXB UF								
UCBxTXB UF								
UCBxIV								