# EA871 – LAB. DE PROGRAMAÇÃO BÁSICA DE SISTEMAS DIGITAIS

## EXPERIMENTO 10 – Conversor ADC

Profa. Wu Shin-Ting

**OBJETIVO:** Apresentação das funcionalidades do módulo ADC (Conversor Analógico-Digital) e um modo de conversão periódica.

**ASSUNTOS:** Programação do MKL25Z128 para conversão múltipla de sinais analógico-digitais no módulo ADC e geração de interrupções periódicas com uso do módulo LPTMR0.

**O que você deve ser capaz ao final deste experimento?**

Entender o princípio de funcionamento de um conversor ADC por aproximação sucessiva.

Saber programar o conversor ADC para operar no modo de interrupção.

Saber recuperar a grandeza física a partir do valor binário amostrado.

Saber programar o temporizador de baixa potência LPTMR0 para operar no modo interrupção.

## INTRODUÇÃO

A maioria dos sensores e sistemas audio-visuais gera sinais analógicos. Para serem processados pelos processadores digitais, como o nosso MCU, estes sinais precisam ser digitalizados, ou convertidos em sinais digitais. Essencialmente o processo consiste em **quantizar** um intervalo contínuo de valores num conjunto finito de dados através de uma sequência de **amostras** do sinal. Há diferentes técnicas de conversão, envolvendo distintas tecnologias [6].

É integrado no nosso KL25 um conversor analógico-digital de 16 *bits* [1]. A técnica implementada é a de aproximação sucessiva com um registrador de aproximação sucessiva (*successive approximation register SAR*) de até 16 *bits* [2]. Os sinais de entrada $V_{IN}$ são amostrados e segurados (*sample and hold S/H*) para serem comparados com os sinais digitais aproximados e convertidos em analógicos pelo circuito DAC como mostra a Figura 1. E o comparador realimenta o circuito do registrador SAR com a diferença dos dois sinais e atualiza o conteúdo do SAR com esta diferença. E assim, sucessivamente, até completar todos os *bits* do SAR e gerar o resultado EOC (*end of conversion*).

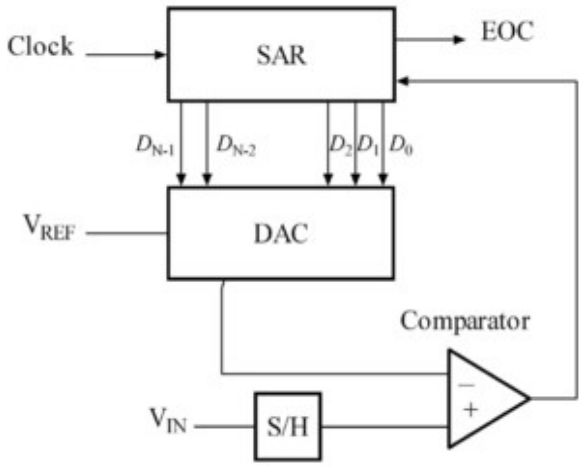

Figura 1: Diagrama de blocos de um ADC por SAR [2].

De acordo com a Tabela 3-32 em [1], o conversor do nosso microcontrolador dispõe de 14 pinos físicos para entrada dos sinais analógicos. Estes pinos podem servir as 24 entradas simples para conversões no modo unipolar (*singular*) ou as 4 entradas diferenciais para conversões no modo bipolar (*differential*). As tensões de referência `VREFH` e `VREFL` utilizadas na conversão [1] são configuráveis pelos *bits* `ADCx_SC2[REFSEL]` [1]. O nosso *kit* é montado para operar no modo `0b00 (VREHH~3V3` e `VREFL=0V)` [1]. Estas tensões podem ser utilizadas na interpretação dos valores binários amostrados conforme explica a Seção 28.6.1.3 em [1]. O modo de operação, uni- ou bipolar, em cada entrada é controlado pelos *bits* de controle `ADCx_SC1n[DIFF]` e as entradas envolvidas numa conversão são definidas pelo 5 *bits* `ADCx_SC1n[ADCH]`. Há um sensor de temperatura [3] integrado no nosso MCU (Seção 28.4.8 em [1]). Ele já se encontra alocado à entrada `0b11010` do conversor. Portanto, para utilizá-lo basta programarmos o conversor de forma que os sinais nesta entrada sejam processados. Observe ainda em [1] que a entrada `0b1111` corresponde à desabilitação do módulo ADC0.

O instante em que uma conversão se inicia é configurável através do campo `ADCx_SC2[ADTRG]` [1]. Este disparo pode ser por *software* através de um acesso de escrita ao registrador `ADCx_SC1n` ou por *hardware* via um dos sinais de tempo selecionado pelos *bits* `SIM_SOPT7[ADCxTRGSEL]` [1]. Enquanto uma conversão estiver em progresso, o *bit* `ADCx_SC2[ADACT]` será setado. O modo de amostragem do conversor pode ser uma vez ou múltiplas vezes (contínuo) conforme a configuração no campo `ADCx_SC3[ADCO]`. Quando se completa uma conversão a bandeira de estado `ADCx_SC1n[COCO]` é levantada e o resultado da conversão é guardado no registrador de dados `ADCx_Rn`, do canal selecionado pelo *bit* `ADCx_CFG2[MUXSEL]` [1]. Este resultado é, de fato, uma média de um conjunto de amostras. O número de amostras por resultado é configurável através dos campos `ADCx_SC3[AVGE]` e `ADCx_SC3[AVGS]` e a resolução do resultado pode ser de 8, 10, 12 ou 16 *bits*., configurável peloos *bits* `ADCx_CFG1[MODE]` [1].

O conversor ADC dispõe ainda de um comparador que compara o resultado com os valores pré-setados nos registradores de dados ADC0_CVn quando o *bit* de controle ADC0_SC2[ACFE] estiver setado. O tipo de comparação a ser feito é configurável pelos *bits* de controle ADC0_SC2[ACFGT] e ADC0_SC2[ACRE]. Para aumentar a precisão dos valores convertidos, há uma função de calibração implementada no microcontrolador. Os *bits* de estado ADC0_SC3[CAL] e ADC0_SC3[CALF] mostram, respectivamente, o progresso e o resultado de uma calibração. Esta função gera correção de erros no ganho do comparador. Na seção 28.4.6 em [1] encontra-se um procedimento de calibração recomendado pelo fabricante.

A fonte dos sinais de relógio ADCK para o circuito de conversão é configurável pelos *bits* de controle ADC0_CFG1[ADICLK] [1]. Há ainda um divisor de frequência ADC0_CFG1[ADIV] através do qual podemos reduzir a frequência da fonte (Seção 28.4.1 em [1]). Os tempos gastos numa amostragem são medidos em termos de número de ciclos de ADCK e eles variam com o tempo de amostragem setado nos campos ADC0_CFG1[ADLSMP] e ADC0_CFG2[ADLSTS] e a velocidade de amostragem configurada no campo ADC0_CFG2[ADHSC] (Seção 28.4.4.5 em [1]). Para operar, a fonte dos sinais de relógio deve ser habilitada através do *bit* de controle SIM_SCGC6[ADC0] [1] e os pinos alocados ao módulo ADC devem assumir o papel de "entradas do conversor analógico-digital". Cada pino serve apenas uma entrada. Por exemplo, de acordo com a tabela na Seção 10.3.1 em [1], o pino PTB3 poderia servir o canal 13 do ADC se PORTB_PCR3[MUX]=0x00.

Finalmente, o conversor ADC é servido pelo controlador NVIC, ou seja, quando o seu *bit* de controle ADCx_SC1n[AIEN] estiver setado, assim que o a bandeira ADCx_SC1n[COCO] levantar, indicando que o resultado está disponível no registrador de dados ADCx_Rn, gera-se uma interrupção IRQ=15/Número de vetor=31 (Tabela 3-7 em [1]). E pela tabela InterruptVector do arquivo Project_Settings/Startup_Code/kinetis_sysinit.c o nome da rotina de serviço pré-definido pelo *CodeWarrior* é ADC0_IRQHandler.

É possível combinar as funcionalidades do módulo ADC0 com um módulo de temporizador para fazer amostragens periódicas, como ilustra o projeto adc.zip [4]. Este projeto mostra uma forma de calibrar o módulo ADC0, utilizar o sensor de temperatura integrado nele para amostrar a temperatura e utilizar o pino PTB3 para amostrar as tensões de um potenciômetro conectado nele. No projeto utilizamos um quarto módulo de temporizador, LPTMR, do nosso micro-controlador. As bases de tempo disponíveis para este temporizador são MCGIRCLK, LPO, ERCLK32K e OSCERCLK. A seleção é feita através do campo LPTMR0_PSR[PCS] (Figura 5-4 em [1]). A frequência selecionada pode ser ainda reduzida através do divisor LPTMR0_PSR[PRESCALE] se o *bit* LPTMR0_PSR[PBYP] estiver em 0.

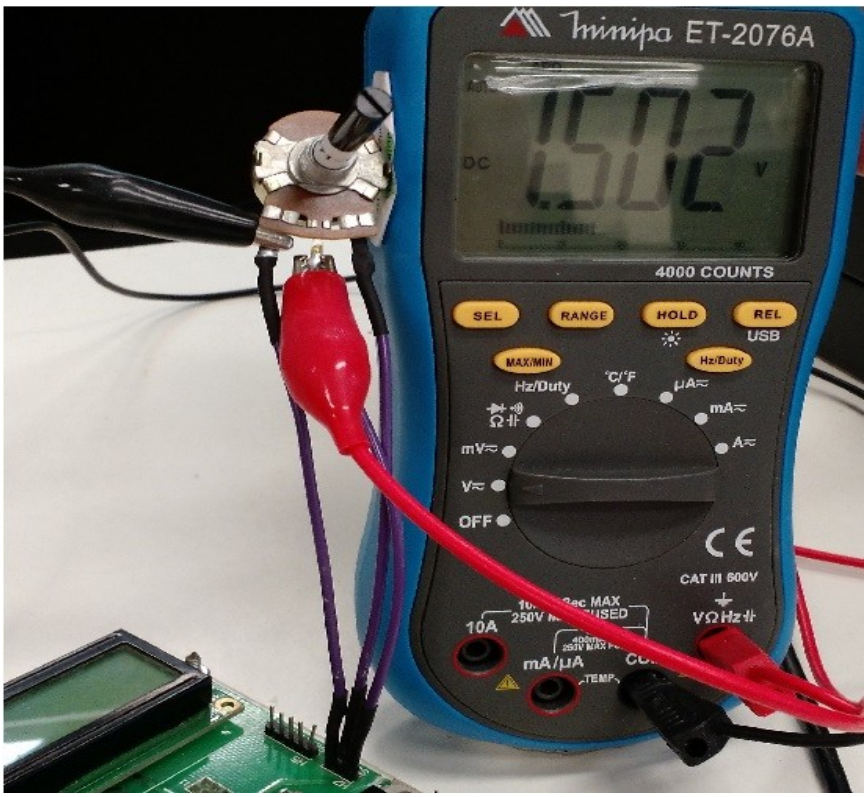


Figura 2: Montagem do potenciômetro [5].

## EXPERIMENTO

1. ***Vamos entender como programar os módulos ADC0 e LPTMR0 no nosso microcontrolador?*** Leia atentamente um exemplo de configuração do módulo ADC0 em [1] e um exemplo de aplicação de amostragem periódica em [7]. Observe que o procedimento de calibração descrito em [1] é aplicado em ambos os exemplos.

2. ***Vamos ver como os conceitos são traduzidos na prática através do programa adc.zip [4]?*** Este programa faz amostragem periódica dos sinais analógicos do sensor de temperatura integrado ao micro-controlador e do potenciômetro conectado aos pinos do *header* H7 conforme a montagem da Figura 2. A periodicidade de amostragem é garantida pelo temporizador `LPTMR0` enquanto o processo de conversão do sinal analógico amostrado é de responsabilidade do conversor `ADC0`. Diferentemente do exemplo dado em [7], temos duas entradas analógicas, ao invés de uma única, para serem monitoradas. Portanto, ao invés de utilizar gatilhos de conversão por *hardware*, foi optado gatilho por *software* periodicamente.

   a. Compare a função `calibraADC` implementada em [4] com o procedimento de calibração recomendado na Seção 28.4.6 em [1].
   b. Veja na Seção 28.5 em [1] um procedimento de inicialização recomendado pelo fabricante para o módulo ADC. Compare-o com o procedimento implementado em [4]. Quais registradores devem ser utilizados para configurar classe dos canais a serem utilizados, modo de conversão (uni- ou bipolar), modo de consumo de energia, fonte dos sinais de *clock*, divisor de frequência, tensões de referência, gatilhos de conversão (único ou contínuo), resolução dos resultados de conversão, quatidade de amostras por conversão?

c. No projeto [4] o *bus clock* (relógio do barramento) é usado como fonte de sinais de *clock* do módulo ADC0. Vimos que este sinal é **compartilhado** por diversos módulos, como UARTx e TPMx. Veja na Figura 5.1 em [1] que a frequência do relógio do barramento é `(freq. do núcleo)/(SIM_CLKDIV1[OUTDIV1]*SIM_CLKDIV1[OUTDIV4])`. Sabendo que a frequência do relógio do núcleo MCGOUTCLK é `20.971520 MHz`, qual é o período do sinal ADCK do projeto [4]?
d. As tensões de referência utilizadas pelo conversor são selecionáveis através dos *bits* `ADCx_SC2[REFSEL]` (Seção 28.4.2 em [1]). Para uma conversão correta a tensão de entrada analógica deve estar no intervalo definido pelas tensões de referência selecionadas. Qual é o valor assumido pelo conversor quando as tensões de entrada estiverem fora deste intevalo ? Como se obtém a tensão medida a partir do valor armazenado no registrador de dados `ADC0_RA`? (Seção 28.6.1.3 em [1])
e. Em quais entradas do conversor `ADC0` estão conectadas o sensor de temperatura e o potenciômetro? A quais pinos físicos elas estão associadas para adquirir sinais analógicos?
f. A função de temperatura em relação à tensão do sensor de temperatura é linear por parte. O ponto de descontintuidade é 25ºC. Na Figura 2 em [3] mostra que a inclinação é 1.646 para tensões menores que 25ºC e 1.769 acima de 25ºC. E a tensão típica para 25ºC com alimentação de 3V é 0.703125V. Como você determina a temperatura medida a partir do valor lido do
g. O módulo LPTMR0 é responsável pelo período de amostragem. Como podemos configurar o período de amostragem dos sinais analógicos?
h. Analise o código do projeto [4] e descreva sucintamente, no formato de um pseudo-código, a lógica implementada para amostrar periodicamente (como se inicia cada amostragem e como os resultados convertidos são armazenados na memória pelo mecanismo de interrupção) os dois sinais analógicos, o de temperatura e o de potenciômetro. Como foi configurado o mecanismo de interrupção dos dois módulos, `LPTMR0` e `ADC0`?
i. Em relação à técnica de programação, foram adicionados arquivos relacionados aos dois módulos, `adc.*` e `lptmr.*`. Para reduzir a quantidade de argumentos que serão necessários na chamada da rotina `initADC0`, criamos uma estrutura `ADCConfig` contendo os valores dos parâmetros a serem utilizados na configuração do módulo `ADC`. Você saberia dizer por quê?

3. **Vamos praticar o que apredemos com o programa adc.zip *[4]?*** Complete o projeto, substituindo xxxx nos códigos. Veja os valores mostrados no visor quando o potenciômetro e a temperatura do ambiente são alterados.

4. ***Vamos ver se você entendeu?*** Crie um novo projeto com o seu programa do experimento 9 [8] e adapte-o para exibir a temperatura do ambiente no canto superior direito do visor LCD e para ter uma entrada alternativa do intervalo do temporizador pelo potenciômetro, dividindo o range do potenciômetro em quantidade de intervalos contemplados pela botoeira PTA5. Submeta um relatório contendo um diagrama de estados da nova versão do projeto e os códigos do novo projeto junto com a biblioteca com os arquivos atualizados no sistema Moodle. Limpe os dois projetos (*Project* > *Clean* …) antes de exportá-los.

## REFERÊNCIAS

[1] Freescale. *KL25 Sub-Family Reference Manual*.
ftp://ftp.dca.fee.unicamp.br/pub/docs/ea871/ARM/KL25P80M48SF0RM.pdf
[2] *Wikipedia*. *Successive Approximation ADC*.
https://en.wikipedia.org/wiki/Successive_approximation_ADC
[3] Temperature Sensor for the HCS08 Microcontroller Family
ftp://ftp.dca.fee.unicamp.br/pub/docs/ea871/complementos/AN3031.pdf
[4] Wu Shin-Ting. adc.zip
http://www.dca.fee.unicamp.br/cursos/EA871/2s2017/codes/adc.zip
[5] Roteiros dos experimentos do EA871 – segundo semestre de 2015
http://faraj7.github.io/
[6] Instituto Newton C. Braga. Como funcionam os conversores A/D?
http://www.newtoncbraga.com.br/index.php/como-funciona/1508-conversores-ad
(parte 1) e http://www.newtoncbraga.com.br/index.php/como-funciona/1509-conversores-ad-2
(parte 2)
[7] Freescale. *Kinetis L Peripheral Module Quick Reference (Rev. 0.09/2012)*.
ftp://ftp.dca.fee.unicamp.br/pub/docs/ea871/ARM/KLQRUG.pdf
[8] Wu S.-T.. EA871 - Roteiro 9 – 2s2017
http://www.dca.fee.unicamp.br/cursos/EA871/2s2017/EM/roteiros/exp9.pdf

*Elaborado com base no roteiro do Experimento 13 no Segundo Semestre de 2015 [5].*
Revisado em Agosto de 2017