EA871 – LAB. DE PROGRAMAÇÃO BÁSICA DE SISTEMAS DIGITAIS

EXPERIMENTO 12 – Módulo ADC

Profa. Wu Shin-Ting

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

ASSUNTOS: Programação do módulo ADC do MKL25Z128 para conversão de sinais analógico-digitais com iniciação controlada por *software* ou por *hardware*.

O que você deve ser capaz ao final deste experimento?

Entender o princípio de funcionamento de um conversor ADC por aproximações sucessivas (SAR).

Entender os erros envolvidos no processo de conversão e o uso de função de auto-calibração para mitigá-los.

Saber configurar o conversor ADC para operar em diferentes modos de operação.

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

Saber programar um módulo para fazer iniciar de forma controlada e automática as conversões periódicas.

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 **amostrar** o sinal analógico e **quantizá**-lo num código binário de N *bits*. Há diferentes técnicas de conversão, envolvendo distintas tecnologias [4].

É integrado no nosso KL25 um conversor analógico-digital de 16 *bits* [1]. A técnica implementada é a de **aproximações sucessivas** com um registrador de aproximações sucessivas (*successive approximation register SAR*) de 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 base nesta diferença. E assim, sucessivamente, até completar todos os *bits* do SAR, do *bit* mais significativo para o menos significativo, e o estado EOC (*end of conversion*)/COCO (*conversion complete*) fique em 1.

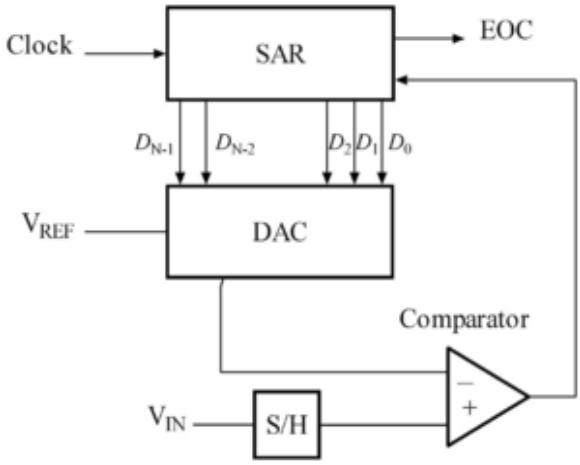

Figura 1: Diagrama de blocos de um ADC por SAR [2].

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* foi implementado para operar no modo `0b00 (VREFH~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]`. Como só há um circuito conversor, o sinal analógico processado em cada instante é o que está configurado nos 5 *bits* `ADCx_SC1n[ADCH]` (`ADCx_SC1A[ADCH]` para modo unipolar e `ADCx_SC1A[ADCH]` e `ADCx_SC1B[ADCH]` para bipolar). Há um sensor de temperatura [3] integrado no nosso MCU (Seção 28.4.8 em [1]). Ele já se encontra alocado ao canal `0b11010` do conversor. Portanto, para utilizá-lo basta setarmos este canal com campo `ADCx_SC1A[ADCH]`. Observe ainda em [1] que o código `0b1111` neste campo 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_SC1A` ou por *hardware* via a ativação de um estado do sinal externo ao módulo ADC, selecionado pelos *bits* `SIM_SOPT7[ADCxTRGSEL]` [1]. Enquanto uma conversão estiver em progresso, o *bit* `ADCx_SC2[ADACT]` será setado. O modo de conversão pode ser único (uma só vez) ou contínuo (sucessivamente após um único disparo inicial), 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 que satisfaz a função de comparação configurada. 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]. A média gerada pela conversão é automaticamente comparada 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[ACREN].

Para aumentar a acurácia dos valores convertidos, há uma função de auto-calibração para disparos por *software*, implementada no microcontrolador. Os *bits* de estado ADC0_SC3[CAL] e ADC0_SC3[CALF] mostram, respectivamente, o progresso e o resultado do processo da calibração. O fim de um processo de calibração é também indicado pelo *bit* ADCx_SC1n[COCO]. Com esta função geram-se os valores de compensação para os erros de offset e de ganho, aplicados automaticamente pelo circuito de conversão. 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]. Neste curso usaremos BUS CLOCK. 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 conversão, que envolve a amostragam e a quantização – conversão propriamente dita -- são medidos em termos de número de ciclos de ADCK e de BUS CLOCK. Eles dependem do modo de amostragem setado nos campos ADC0_CFG1[ADLSMP] e ADC0_CFG2[ADLSTS], a velocidade de conversão configurada no campo ADC0_CFG2[ADHSC] e da resolução do resultado digital (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 um canal. 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 uma calibração tenha sido concluída ou 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 declarado pelo *CodeWarrior* é ADC0_IRQHandler.

Quando se opta por disparos de inicialização de uma conversão por *hardware*, é necessário configurar e habilitar o módulo que gera sinais de disparo. Na Tabela 3-1 em [1] é apresentada uma lista de módulos interconectados no nosso microcontrolador, incluindo os módulos cujos eventos gerados possam servir de disparos para o módulo ADC (*ADC Triggering*). Na sexta coluna desta tabela são mostrados os campos do registrador SIM_SOPT7 que configuram as fontes de disparo para as entradas A e B do ADC0. No capítulo 11 em [5] há um exemplo de uso do módulo LPTMR0 para disparos de conversões por *hardware*.

Quando devidamente calibrado, podemos considerar que a função de transferência do módulo ADC seja linear, de forma que o resultado em código binário de N *bits* de resolução possa ser obtido por uma regra de três simples:

(Tensão amostrada-VREFL) → Código Binário

(VREFH – VREFL) → 2^N

Quando as tensões amostradas estiverem fora do intervalo `[VREFL,VREFH]`, o resultado é 0 para tensões menores que `VREFL` e 2^{N-1} para tensões maiores que `VREFH`.

Se a aplicação demandar valores em grandezas físicas originais dos sinais amostrados, é necessário pós-processar os códigos binários das amostras, convertendo-os para valores em grandezas físicas. Isso pode ser feito em dois passos: (1) Converter o código binário para a tensão correspondentes, e (2) Converter a tensão para o valor em grandeza física original. No primeiro passo podemos aplicar a função linear inversa do módulo ADC e para o segundo passo precisamos recorrer aos *datasheets* dos fabricantes dos sensores.

EXPERIMENTO

Neste exeprimento vamos desenvolver o projeto **termometro** que amostra periodicamente (em cada 1s) a temperatura de uma fonte de calor, amostrada pelo sensor de temperatura LM61 [6] conectado no pino PTB1, e exibí-lo no canto direito superior do visor do LCD no formato "XX C". O *led* RGB acenderá em 6 diferentes cores correspondentes às 6 faixas de temperaturas menor que 22, [22,23), [23,24), [24,24), [25,26),maior ou igual a 26. No mínimo, 3 cores não são primárias nem secudárias. A fonte de calor é emulada por uma lâmpada torpedo cuja intensidade luminosa é controlável remotamente pelo *slider* da interface remota.

O modo de operação do módulo ADC especificado é

disparo por *hardware* usando o módulo PIT
frequência ADCK: 5242880Hz
resolução de 12 *bits*
tempo de amostragem longo habilitado
alta velocidade de conversão habilitada
média habilitada para 16 amostras por conversão

1. Qual é o modo de conversão, único ou contínuo, a ser configurado para que o módulo ADC opere corretamente? Justifique.
2. Como deve ser configurado PIT para que sejam gerados periodicamente os eventos de *overflow* capazes de disparar as conversões periódicas? Qual é o menor período para o qual o conversor opere corretamente? Justifique com base no tempo de amostragem dada na tabela da página 488 em [1] e no tempo de conversão dado pela equação da Fig. 28-62 em [1].

3. Quais foram os valores salvos nos registradores ADC0_OFS, ADC0_PG e ADC0_MG após a calibração?
4. Qual é a função de conversão entre o código binário armazenado no registrador ADC0_Rn e a temperatura amostrada? E como você pode aplicá-la para o controle da cor do led RGB? Justifique.
5. Escreva o pseudocódigo do fluxo de controle principal do seu projeto, onde se encontram as instruções de pós-processamento e o controle só estado do led RGB.
6. Implemente o aplicativo **termometro** em C.
7. Documente todas as funções que não foram geradas pelo IDE CodeWarrior.

RELATÓRIO

Para este experimento, responda as questões 1 a 4 num arquivo em **pdf**, implemente e documente o projeto **termometro.** Exporte o projeto no ambiente IDE CodeWarrior para um arquivo em formato zip. Suba **os dois arquivos, em separado**, no sistema *Moodle*. Não se esqueça de identificar todos os seus arquivos de códigos com a palavra reservada "@author" de Doxygen.

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] 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)
[5] Freescale. *Kinetis L Peripheral Module Quick Reference (Rev. 0.09/2012)*.
ftp://ftp.dca.fee.unicamp.br/pub/docs/ea871/ARM/KLQRUG.pdf
[6] *Datasheet* de LM61
ftp://ftp.dca.fee.unicamp.br/pub/docs/ea871/datasheet/LM61.pdf

Revisado em Novembro de 2020
Revisado em Agosto de 2017

Elaborado com base no roteiro do Experimento 13 no Segundo Semestre de 2015.