EA871 – LAB. DE PROGRAMAÇÃO BÁSICA DE SISTEMAS DIGITAIS

EXPERIMENTO 1 – Introdução ao *Hardware* (*FRDM KL25* e *shield* EA871)

Profa. Wu Shin-Ting

OBJETIVO: Apresentação do *kit* de desenvolvimento FRDM-KL25 e do *shield* EA871

ASSUNTOS: Microcontrolador e periféricos integrados no kit FRDM-KL25. Revisão dos conceitos relacionados com o microcontrolador. Arquitetura ARM.

O que você deve ser capaz ao final deste experimento?

Relacionar projetos digitais baseados em componentes eletrônicos e tecnologia FGPA da disciplina EA773, ou equivalente, com projetos baseado em sistemas embarcados.

Recordar os conceitos vistos na disciplina EA869 ou equivalente.

Ter uma noção do *kit* FRDM-KL25 e dos principais manuais a serem utilizados em EA871.

Ter uma noção do *shield* EA871.

Gerar um programa em C para MKL25Z128 com uso do ambiente *IDE CodeWarrior*.

INTRODUÇÃO

Os conceitos da disciplina EA869 serão revistos e ampliados numa perspectiva prática. Para tanto, será utilizada uma plataforma de desenvolvimento denominada FRDM-KL25. Esta placa de desenvolvimento é fabricada pela *Freescale Semiconductors*, e permite desenvolver e depurar programas em várias linguagens, possuindo diversas conexões para circuitos digitais, em vários padrões de interface.

O "comandante" da placa é o microcontrolador MKL25Z128VLK4 de 32 *bits*, da série Kinetis L da *Freescale* [1]. Este dispositivo é um *System-on-a-Chip* (SoC), isto é, num *chip* são agregados um núcleo de processamento Cortex-M0+ de arquitetura *ARM*® e vários módulos, como temporizadores, interfaces de comunicação, conversores DA/AD, interrupções e unidades de armazenamento. Aliados o baixo custo e o baixo consumo de energia ao tamanho reduzido e à flexibilidade no desenho de um novo projeto, ele constitui uma alternativa para desenvolver aplicativos portáteis e de alto desempenho.

São integrados na placa FRDM-KL25 um circuito de alimentação e de *clock*, um *touchpad* capacitivo, um acelerômetro, *leds* RGB [2]. Além disso, há na placa um adaptador serial e de depuração, *OpenSDA*, que permite que sejam transferidos programas de um computador-hospedeiro para a memória interna do microcontrolador-alvo e que os mesmos sejam depurados. Isso é feito com um *software* específico rodando no computador-hospedeiro, conectado à placa através de uma interface USB. É interessante observar que o circuito do *OpenSDA* é baseado num outro microcontrolador da família Kinetis K20 da *Freescale*, K20DX128VFM5.

EXPERIMENTO

1. Vamos recordar um projeto digital baseado em componentes eletrônicos, analisando o esquemático de um circuito de pisca de um *led* vermelho na freqüência de 1Hz da Figura 1. Descreva o funcionamento do circuito, explicitando a função do *flip-flop* JK 7411, a função do resistor 330Ohms e a função e a freqüência do sinal CLK.

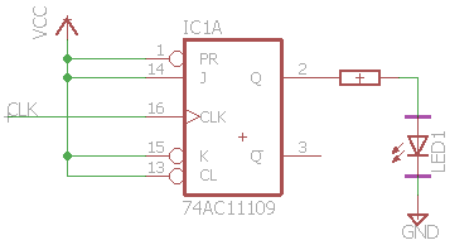

Figura 1: Esquemático do circuito de pisca do *led*.

Ao invés de componentes eletrônicos, utilizaremos microcontroladores nesta disciplina para controlar os periféricos como o *led* vermelho. Muito diferente das **funções fixas** implementadas no circuito apresentado, programamos a sequência de execução das funcionalidades disponíveis no microcontrolador.

2. *Vamos entender o hardware a ser utilizado nos experimentos da disciplina EA871?* **Leia** atentamente o documento [3] para conhecer alguns detalhes, sob o ponto de vista de programação, da placa de desenvolvimento FRDM-KL25 e do *shield* EA871. Podemos configurar, via *software*, o nosso microcontrolador para realizar uma tarefa específica. Vimos que, além de módulos dedicados, como temporizadores e interfaces de comunicação, ele possui um sistema complexo e configurável de geração de sinais de relógio no módulo MCG (*Multipurpose Clock Generator*) (veja Capítulo 24 de [1]), controlado pelo módulo SIM (*System Integration Module*) (veja Capítulo 12 de [1]), e de multiplexação dos sinais nos seus pinos físicos, PORT (*Port Control and Interrupts*) (veja Capítulo 11 de [1]). E todos os elementos de *hardware* são mapeados num mesmo espaço de endereços de memória de forma que possamos manipulá-los através dos seus endereços dentro de um programa (*software*). Estes endereços são usualmente conhecidos por endereços dos **registradores** de controle, de dados ou de estado. Ao inicializar/resetar o microcontrolador, todos os registradores são inicializados com valores especificados pelo fabricante. **Tanto os endereços quanto os valores de "*Reset*" acompanham a descrição de cada registrador em** [1].

3. *Vamos ver como os conceitos são traduzidos na prática através de um programa* [4]? O programa dado realiza a tarefa específica de piscar o *led* vermelho do *kit* FRDM-KL25Z, de forma equivalente ao comportamento do circuito baseado em componentes eletrônicos no item 1. O *led* vermelho está ligado no pino 18 da porta B do nosso microcontrolador com o nível lógico ativo baixo (Figura 13 em [3]). Portanto, para piscá-lo, precisamos gerar alternadamente no pino 18 da porta B **sinais digitais** 0 e 1 de largura correspondente à metade do período correspondente à frequência desejada. Qual é o estado do *led* vermelho,

aceso ou apagado, logo após um *reset* do microcontrolador? Como configurar/programar o microcontrolador para gerar este sinal? Justifique.

Vimos que a função de cada pino é controlado pelo módulo PORTx e o relógio de cada módulo PORTx controlado pelo módulo SIM. Os circuitos que estão por baixo são mapeados, respectivamente, nos endereços 0x4004A048 (Seção 11.5 em [1]) e 0x40048038 (Seção 12.2.9 em [1]). Neste caso específico, a função do pino é gerar sinal digital de propósito geral, portanto foi atribuído o valor 0b001 no campo MUX do endereço 0x4004A048 correspondente ao pino 18 da porta B (Seção 11.5.1 em [1]). O valor 0b001 é o código binário da função GPIO (*General Purpose Input/Output*) que o pino desempenhará. Uma vez configurado como GPIO, as características específicas dos sinais digitais genéricas passam a ser controladas pelo módulo GPIO, como setar a direção dos sinais, o nível do sinal no pino configurado como saída e alternar o nível do sinal através dos endereços 0x400FF054, 0x400FF040 e 0x400FF04C, respectivamente (Seção 41.2 em [1]). Observe que cada *bit* destes registradores controla um pino da porta B. Portanto, as operações de atribuição devem ser por *bit* utilizando os operadores lógicos & (*and*), | (*or*), << (deslocamento para esquerda) e >> (deslocamento para direita) da linguagem C [5]. Há, porém, três "registradores auxiliares" que facilitam esta atribuição. Eles modificam o valor binário dos *bits* somente quando o *bit* correspondente estiver em "1", GPIOx_PSOR (Seção 41.2.2 em [1]), GPIOx_PCOR (Seção 41.2.3 em [1]) e GPIOx_PTOR (Seção 41.2.4 em [1]). Note ainda que no programa todos os endereços foram renomeados pelos nomes utilizados em [1] via `typedef` para tornar o programa mais inteligível [6]. Finalmente, veja que todos os endereços são declarados com o qualificador `volatile` para evitar que o compilador otimize trechos de códigos contendo registradores cujo conteúdo seja passível de ser alterado por eventos externos ao processador [6].

4. *Como editar um programa e instalá-lo no microcontrolador?* **Leia** atentamente o documento [7] para ver como se cria, edita, gera um executável para microcontrolador e depura um projeto no ambiente *IDE CodeWarrior* instalado em todos os computadores *desktop* do laboratório LE-30. Vale a pena ter sempre em mãos a tabela de atalhos dos comandos do ambiente [9]. Siga os passos nas Seções 2.1 – 2.4 para instalar o programa [4] no microcontrolador e meça a freqüência aproximada das piscadas do *led* vermelho.

5. *Vamos praticar mais o que acabamos de ver?* Quais alterações você faria no *programa* [4] (1) para adicionar a cor verde e a cor azul do *led* RGB na composição da cor "branca", (2) para alternar o estado dos *leds* com uso de GPIOx_PTOR e (3) para evitar uso desnecessário do qualificador `volatile`? Os três *leds*, vermelho, verde e azul, devem ser iniciados com o estado "apagado".

RELATÓRIO

O relatório deste experimento deve ser devidamente identificado seguindo o modelo de relatório [8]. Suba o relatório em pdf e o seu programa em C, `main.c`, no sistema *Moodle*.

REFERÊNCIAS

Todas as referências podem ser encontradas nos *links* abaixo ou ainda na página do curso.

[1] *KL25 Sub-Family Reference Manual – Freescale Semiconductors* (*doc. Number KL25P80M48SF0RM*, Setembro 2012).
ftp://ftp.dca.fee.unicamp.br/pub/docs/ea871/ARM/KL25P80M48SF0RM.pdf
[2] *FRDM-KL25Z User's Manual – Freescale Semiconductors*, Setembro 2012.
ftp://ftp.dca.fee.unicamp.br/pub/docs/ea871/ARM/FRDMKL25Z.pdf
[3] Wu Shin-Ting e A.A.F. Quevedo. Ambiente de Desenvolvimento – *Hardware*
ftp://ftp.dca.fee.unicamp.br/pub/docs/ea871/apostila_C/AmbienteDesenvolvimentoHardware.pdf
[4] apostila.c
http://www.dca.fee.unicamp.br/cursos/EA871/1s2017/ST/codes/apostila.c
[5] Wu Shin-Ting e D. S. Oliveira. Linguagem C: Operações sobre os Dados.
ftp://ftp.dca.fee.unicamp.br/pub/docs/ea871/apostila_C/Operadores.pdf
[6] D. S. Oliveira e Wu Shin-Ting. Linguagem C: Representação de Dados.
ftp://ftp.dca.fee.unicamp.br/pub/docs/ea871/apostila_C/RepresentacaoDados.pdf
[7] Wu Shin-Ting e A.A.F. Quevedo. Ambiente de Desenvolvimento – *Software*
ftp://ftp.dca.fee.unicamp.br/pub/docs/ea871/apostila_C/AmbienteDesenvolvimentoSoftware.pdf
[8] Modelo de Relatório
http://www.dca.fee.unicamp.br/cursos/EA871/2s2016/UW/roteiros/relatorio_template.txt
[9] -. Eclipse CDT 8.0 Cheat Sheet.
ftp://ftp.dca.fee.unicamp.br/pub/docs/ea871/complementos/eclipseCDT8.0-cheatsheet.pdf

Agosto de 2016
Revisado em Fevereiro de 2017
Revisado em Julho de 2017