EA871 – LAB. DE PROGRAMAÇÃO BÁSICA DE SISTEMAS DIGITAIS

EXPERIMENTO 4 – Primeiro Programa em C

Profa. Wu Shin-Ting

OBJETIVO: Proporcionar as noções básicas sobre a programação em C do microcontrolador Kinetis KL25Z.

ASSUNTOS: Programação de sinais digitais nos pinos do microcontolador; forma de acesso aos registradores envolvidos com o controle destes sinais; módulos SIM, PORT e GPIO do KL2x.

O que você deve ser capaz ao final deste experimento?

Entender o princípio básico de operação do microcontrolador KL2x.

Saber consultar nos manuais os endereços dos registradores de cada módulo e as funções dos seus *bits*.

Saber acessar os registradores do microcontrolador em C.

Ter uma noção como representar a execução de uma tarefa em termos de uma **máquina de estados** (**fluxo de controle**) e **fluxo de dados**.

Saber descrever o fluxo de controle de um programa por um pseudo-código.

Transcrever um pseudocódigo para um programa em C.

INTRODUÇÃO

No experimento 3 apresentamos o projeto asm_FRDMKL25 que controla o sinal digital do pino 23 da porta E do microcontrolador KL25Z, ao qual está conectado um *led* azul [1]. Como o objetivo era ilustrar a estimativa de tempo de execução na base de ciclos de instrução das operações envolvidas, instruções referentes ao controle dos sinais de alimentação do *led* azul foram apresentadas como uma caixa preta. Neste experimento vamos "dissecar" esta caixa preta através de uma versão do código em C.

Uma visão do Kinetis KL2x, em diagrama de blocos, é mostrada na Figura 1. Além do núcleo que inclui o processador Cortex-M0+, temos blocos de controle do sistema, um banco de memórias, uma série de temporizadores, interfaces com sinais analógicos e sinais digitais, e diferentes interfaces de comunicação serial. Como uma estratégia para reduzir o consumo de energia, somente os módulos essenciais para operação tem os sinais de relógio habilitados na inicialização. Os sinais de relógio dos outros módulos precisam ser habilitados por meio dos *bits* nos registradores de controle de *System Integration Module* (SIM). Como uma medida para reduzir a quantidade de pinos, e portanto as dimensões do microcontrolador, a outra estratégia é a multiplexação dos sinais em cada pino. Há um módulo *Port control and interrupts* (PORT) específico para controlar tal configuração em cada pino da porta. Assim , para programar qualquer bloco dentro das classes *Analog*, *Timers*, *Communication Interfaces* e *HMI* (*Human Machine Interface*), é necessário habilitar o seu sinal de relógio via registradores de controle do módulo SIM e, se há transferência de sintais entre o bloco com o mundo externo, é preciso alocar e configurar, via o módulo PORT, um

pino. Além de SIM e PORT, o microcontrolador provê blocos dedicados para processamento de sinais de características específicas, como sinais analógicos, sinais digitais de propósito geral, e sinais de comunicação serial, tais como UART (*Universal Asynchronous Receiver/Transmitter*), I2C (*Inter-Integrated Circuit*) e SPI (*Serial Peripheral Interface*).

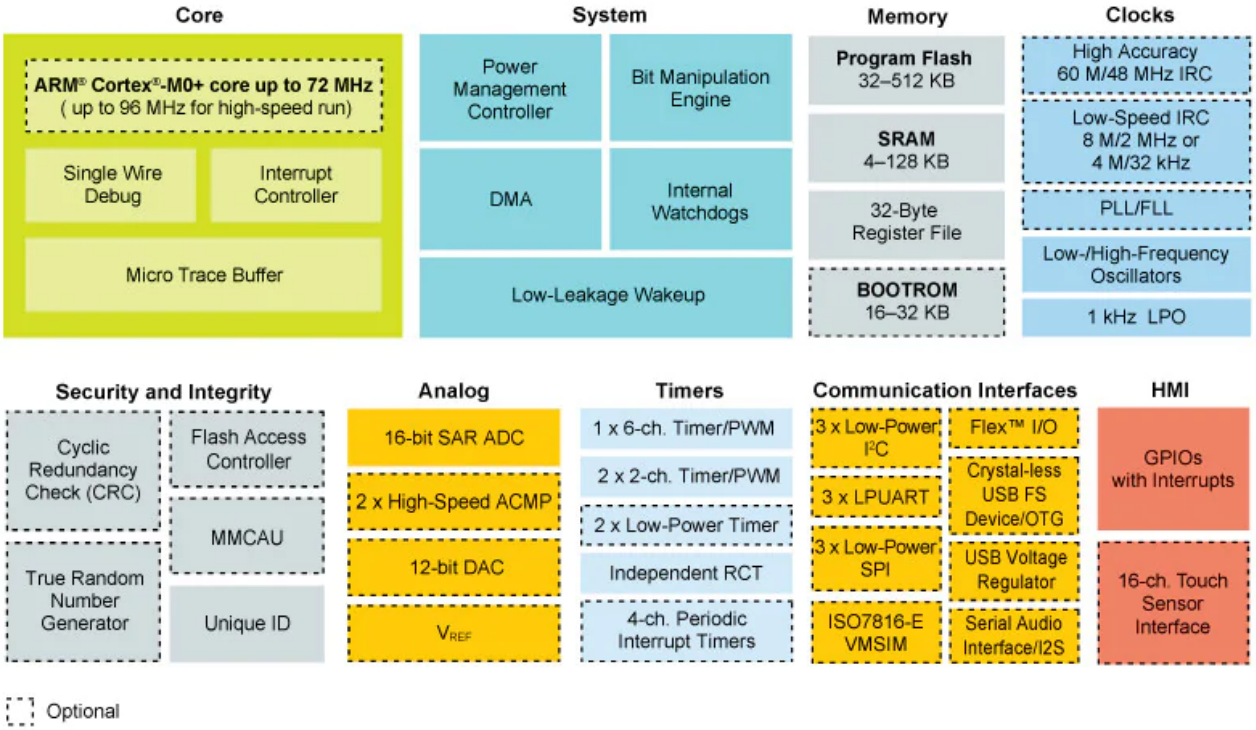

Figura 1: Diagrama de blocos de Kinetis KL2x (Fonte: [2])

Nos projetos asm_FRDMKL25 e hello_FRDMKL25 foi usado o módulo GPIOx para processar o sinal digital que alimenta o *led* azul da placa emergencial [4], conectado ao pino PTE23 do *shield* FEEC871 [5]. Figura 2 apresenta uma descrição da terefa executada com uso de uma máquina de estados finitos e um fluxo de dados controlado por ela. São 6 estados controlados por distintos registradores de controle programáveis:

1. ativação e inicialização do pino 23 de PORTE como um pino para sinal digital de propósito geral;
2. configuração do pino 23 como um pino de saída;
3. *led* aceso;
4. contagem de uma variável aré atingir o limite com *led* aceso;
5. *led* apagado; e
6. contagem de uma variável até atingir o limite com *led* apagado.

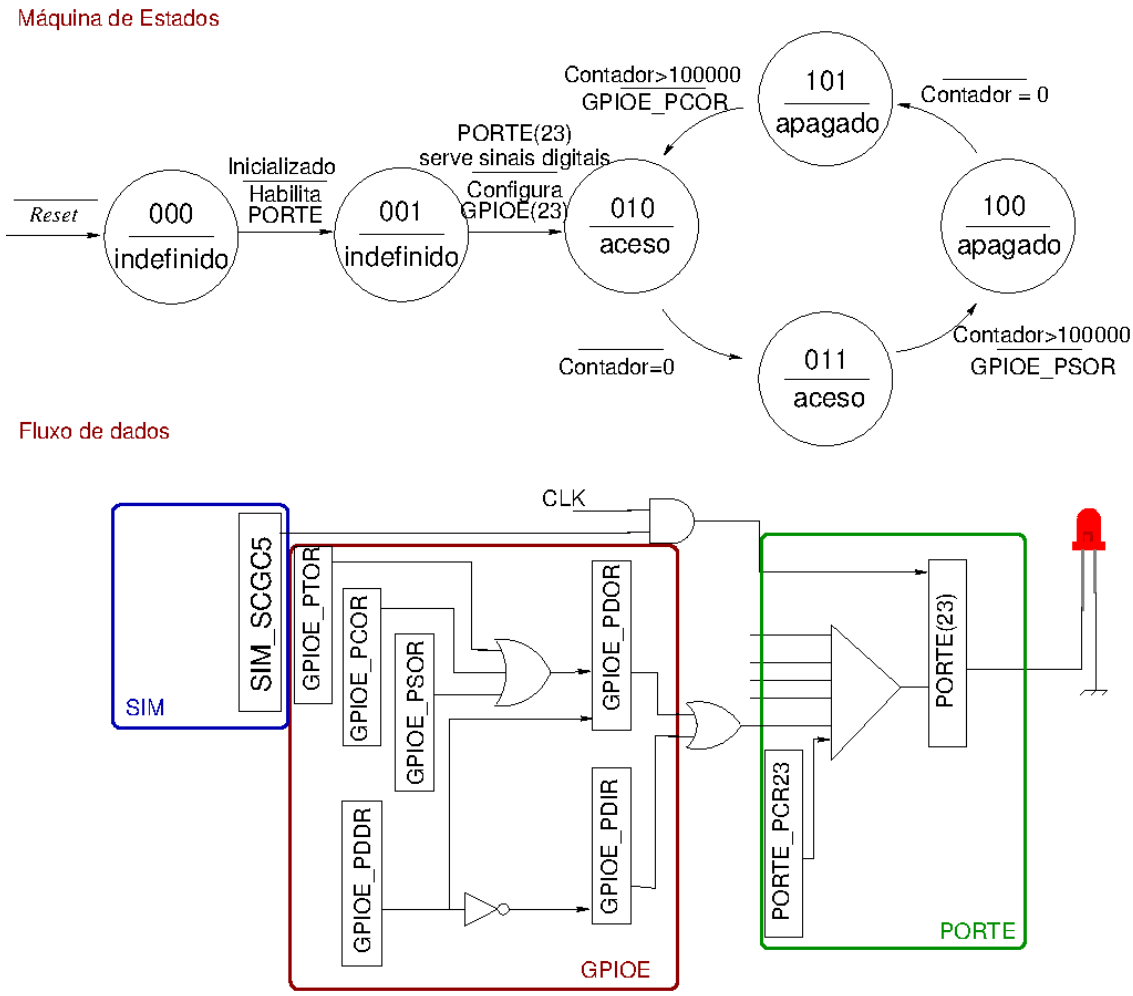

Figura 2: Diagrama de estados e fluxo de dados na execução da tarefa de luz piscante.Todos os registradores que aparecem no fluxo de dados são mapeados no espaço de endereços de memória do microcontrolado e estes endereços podem ser consultados em [3] ou na aba `Registers` da perspectiva Debug do IDE CodeWarrior. Figura 3, extraída de [3], mostra nas colunas 1 e 4 os endereços de alguns registradores do módulo SIM e os tipos de acesso permitidos, respectivamente.

O fluxo de dados é simples. O sinal do *bit* 23 do registrador de dados de saída digital, GPIOE_PDOR, é multiplexado para o pino 23 da porta E onde está conectado o *led*, se o sinal de relógio da porta E (PORTE) é habilitado, ou seja, SIM_SCGC5[13]=1 (Seção 12.2.9 em [3]), e o pino 23 da PORTE é configurado para receber o sinal digital do módulo GPIO (*General Purpose I/O*). No Kinetis KL2x, até 8 sinais são multiplexáveis por pino físico. A seleção de um sinal específico se dá pelos *bits* 8, 9 e 10 (MUX) do registrador de controle PORTE_PCR23 (Seção 11.5.1 em [3]). O código binário correspondente a cada alternativa de sinal para cada pino pode ser consultado na tabela da Seção 10.3.1 em [3]. Em C, para setar os 3 *bits* MUX sem afetar os outros *bits* de controle do registrador PORTE_PCR23, usamos **operadores lógicos**. O módulo GPIOE controla o fluxo de sinais digitais em duas direções e dispõe um registrador de entrada (GPIOE_PDIR) (Seção 41.2.5 .em [3]) e um de saída (GPIOE_PDOR) (Seção 41.2.6 .em [3])). Por *default*, o fluxo é de entrada (0). Caso o programador queira alterar o sentido do fluxo, basta setar o *bit* correspondente ao pino no registrador GPIOE_PDDR em 1 ((Seção 41.2.6 .em [3]). A

fim de facilitar a atribuição de um valor lógico num pino, o módulo GPIOE tem ainda 3 registradores de controle, GPIOE_PSOR (Seção 41.2.2 .em [3]), GPIOE_PCOR (Seção 41.2.3 .em [3]) e GPIOE_PTOR (Seção 41.2.4 .em [3]), que permitem a modificação "segura" do valor de um *bit* do registrador GPIOE_PDOR para o valor desejado sem operações de mascaramento.

SIM memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4004_7000	System Options Register 1 (SIM_SOPT1)	32	R/W	See section	12.2.1/193
4004_7004	SOPT1 Configuration Register (SIM_SOPT1CFG)	32	R/W	0000_0000h	12.2.2/194
4004_8004	System Options Register 2 (SIM_SOPT2)	32	R/W	0000_0000h	12.2.3/195
4004_800C	System Options Register 4 (SIM_SOPT4)	32	R/W	0000_0000h	12.2.4/197
4004_8010	System Options Register 5 (SIM_SOPT5)	32	R/W	0000_0000h	12.2.5/199
4004_8018	System Options Register 7 (SIM_SOPT7)	32	R/W	0000_0000h	12.2.6/200
4004_8024	System Device Identification Register (SIM_SDID)	32	R	See section	12.2.7/202
4004_8034	System Clock Gating Control Register 4 (SIM_SCGC4)	32	R/W	F000_0030h	12.2.8/204
4004_8038	System Clock Gating Control Register 5 (SIM_SCGC5)	32	R/W	0000_0180h	12.2.9/206
4004_803C	System Clock Gating Control Register 6 (SIM_SCGC6)	32	R/W	0000_0001h	12.2.10/207
4004_8040	System Clock Gating Control Register 7 (SIM_SCGC7)	32	R/W	0000_0100h	12.2.11/209

Figura 3: Mapa de memória de alguns registradores do módulo SIM (Fonte: [3])

Diferente do repertório de instruções ARM-Thumb que tem uma classe de instruções dedicadas para transferência de dados entre a memória e o processador, a linguagem C não faz distinção entre acessos aos registradores e acessos à memória. Para diferenciar o endereço de memória representado por um valor inteiro de um valor inteiro ordinário, deve-se usar o **operador de conversão explícita** entre tipos de dados (*casting*).

Sintetizando, o fluxo de controle do programa pode ser representado pelo seguinte pseudo-código. Com base deste pseudo-código foram implementados os projetos asm_FRDMKL25 e hello_FRDMKL25:

```
subrotina delay
INICIO
        Entrada: valor
        Saída: nenhuma
        Enquanto valor  é diferente de zero
                decrementa valor de 1
        Fim enquanto
FIM

programa main
INÍCIO
        Entrada: nenhuma
        Saída: sinais digitais alternados no pino 23 do GPIOE
        Inicailização:
          SIM_SCGC5[13] (0x4004_8038)← 1;
```

```
PORTE_PCR23[8:10] (0x4004_D05C) ← 0b001;
GPIOE_PDDR[23] (0x400F_F114) ← 1;
GPIOE_PCOR (0x400F_F108) ← 2^23;
Enquanto (verdadeiro)
        GPIOE_PSOR (0x400F_F104) ← 2^23;
        chama delay com 100000 de entrada
        GPIOE_PCOR (0x400F_F108) ← 2^23;
        chama delay com 100000 de entrada
    Fim enquanto
FIM
```

EXPERIMENTO

Incluir no projeto hello_FRDMKL25 mais duas cores, a vermelha e a verde do *led* RGB que estão conectadas nos pinos 21 e 22 da porta E.

1. Ambos os programas hello_FRDMKL25 e asm_FRDMKL25 tem a mesma função de controlara as piscadas do *led* azul. Compare o tamanho dos dois programas em termo de número de instruções (espaço de memória ocupado).
2. Utilize o analisador lógico para medir a frequência média das piscadas do *led* azul do projeto hello_FRDMKL25 (os pinos dos canais 0, 1, 2 e 3 do analisador lógico estão conectados nos pinos PTE20, PTE21, PTE22 e PTE23, respectivamente).
3. Quais são os endereços dos registradores que controlam a operação dos pinos 21 e 22 da porta E para transferência de sinais digitais necessários às duas cores?
4. Escreva um pseudo-código que gere uma sequência cíclica de cores vermelha – verde – azul – branca numa frequência de 0.5Hz.
5. Implemente em C o pseudo-código. Reuse o código de atraso em *assembly* implementado no experimento 2 para aumentar a precisão da frequência das piscadas.

RELATÓRIO

O relatório deve ser devidamente identificado, contendo a identificação do instituto e da disciplina, o experimento realizado, o nome e RA do aluno. Para este experimento, responda os itens 1-4 do roteiro e suba-o, acompanhado da nova versão hello_FRDMKL25 exportada no ambiente IDE CodeWarrior, no sistema *Moodle*.

REFERÊNCIAS

[1] Experimento 3
http://www.dca.fee.unicamp.br/cursos/EA871/2s2020/ST/roteiros/roteiro3.pdf

[2] KL2x: Kinetis® KL2x-72/96 MHz, USB Ultra-Low-Power Microcontrollers (MCUs) based on Arm® Cortex®-M0+ Core
https://www.nxp.com/products/processors-and-microcontrollers/arm-microcontrollers/general-purpose-mcus/kl-series-cortex-m0-plus/kinetis-kl2x-72-96-mhz-usb-ultra-low-power-microcontrollers-mcus-based-on-arm-cortex-m0-plus-core:KL2x

[3]KL25 Sub-Family Reference Manual, Rev. 3, September 2012
ftp://ftp.dca.fee.unicamp.br/pub/docs/ea871/ARM/KL25P80M48SF0RM.pdf
[4] Esquemático de conexão do *led* RGB para acesso remoto
ftp://ftp.dca.fee.unicamp.br/pub/docs/ea871/ensino_emergencial/LEDRGB.pdf
[5] Esquemático do shield FEEC871
ftp://ftp.dca.fee.unicamp.br/pub/docs/ea871/complementos/Esquematico_EA871-Rev3.pdf

Agosto de 2016.
Revisado em Fevereiro de 2017.
Revisado em Julho de 2017.
Revisado em Fevereiro de 2018.
Revisado em Setembro de 2020.