EA871 – LAB. DE PROGRAMAÇÃO BÁSICA DE SISTEMAS DIGITAIS

EXPERIMENTO 7 – Interrupção

Profa. Wu Shin-Ting

OBJETIVO: Apresentação de um projeto de interface entre um microcontrolador e uma chave simples por interrupção.

ASSUNTOS: Técnica de interrupção.

O que você deve ser capaz ao final deste experimento?

Entender a diferença entre a técnica *polling* e a técnica de interrupção no processamento de eventos de entrada assíncronos.

Saber caracterizar o mecanismo de interrupção suportado pelo microcontrolador KL25Z.

Conhecer a técnica de programação em C adotada no IDE CodeWarrior para declaração do vetor de interrupções.

Saber programar o processamento de interrupções no microcontrolador KL25Z.

INTRODUÇÃO

No experimento 6, para saber se as botoeiras PTA4, PTA5 ou PTA12 foram pressionadas, tivemos que testar periodicamente os seus estados. Quando o nível lógico nos respectivos pinos for 0, consideramos que se fechou o circuito [1]. Essa prática de espera por um evento acontecer, ocupando o processador com a leitura de um registrador de estado, é conhecida por ***polling***.

Na maioria dos sistemas digitais, espera-se que o núcleo de processamento seja capaz de responder apropriadamente aos diferentes estados de um dispositivo do mundo exterior (teclado, sensores, contadores de tempo, etc.). No entanto, para monitorar os estados de um dispositivo, a técnica *polling* tem várias desvantagens, sendo as principais o desperdício de tempo de processamento na varredura das entradas e o tempo de resposta a uma entrada crítica e o consumo de energia para manter o processador em atividade de amostragem. Porisso, a estratégia por **interrupção** é a mais recomendada para um microcontrolador responder a um evento externo. E, o nosso microcontrolador dispõe de um *hardware* específico, o NVIC (*Nested Vectored Interrupt Controller*), para monitorar e processar as entradas assíncronas externas **por módulo**, desviando a execução do programa para uma função chamada *handler* ou **rotina de serviço** (ISR - *Interrupt Service Routine*) no processador (Cap.3 de [2]). Quando se conclui a execução da sequência de instruções da rotina de serviço, o processador retorna ao fluxo original. Operando de forma cooperativa com o núcleo de processamento, o NVIC proporciona uma forma eficiente com uma interface de programação simples para tratar as múltiplas e aninhadas (*nested*) solicitações de interrupção. Figura 1 mostra a relação do NVIC com outros módulos do nosso microcontrolador no processamento de um acionamento momentâneo de uma botoeira, por interrupção. Note que uma parte do processamento ocorre automaticamente em *hardware* desde que seja (1) configurado o estado do sinal de entrada (bordas ou níveis) que deve ser reconhecido como uma requisição de interrupção pelo registrador PORTx_PCRn, e (2) habilitado o controlador NVIC para processar tal requisição pelo registrador NVIC_ISR.

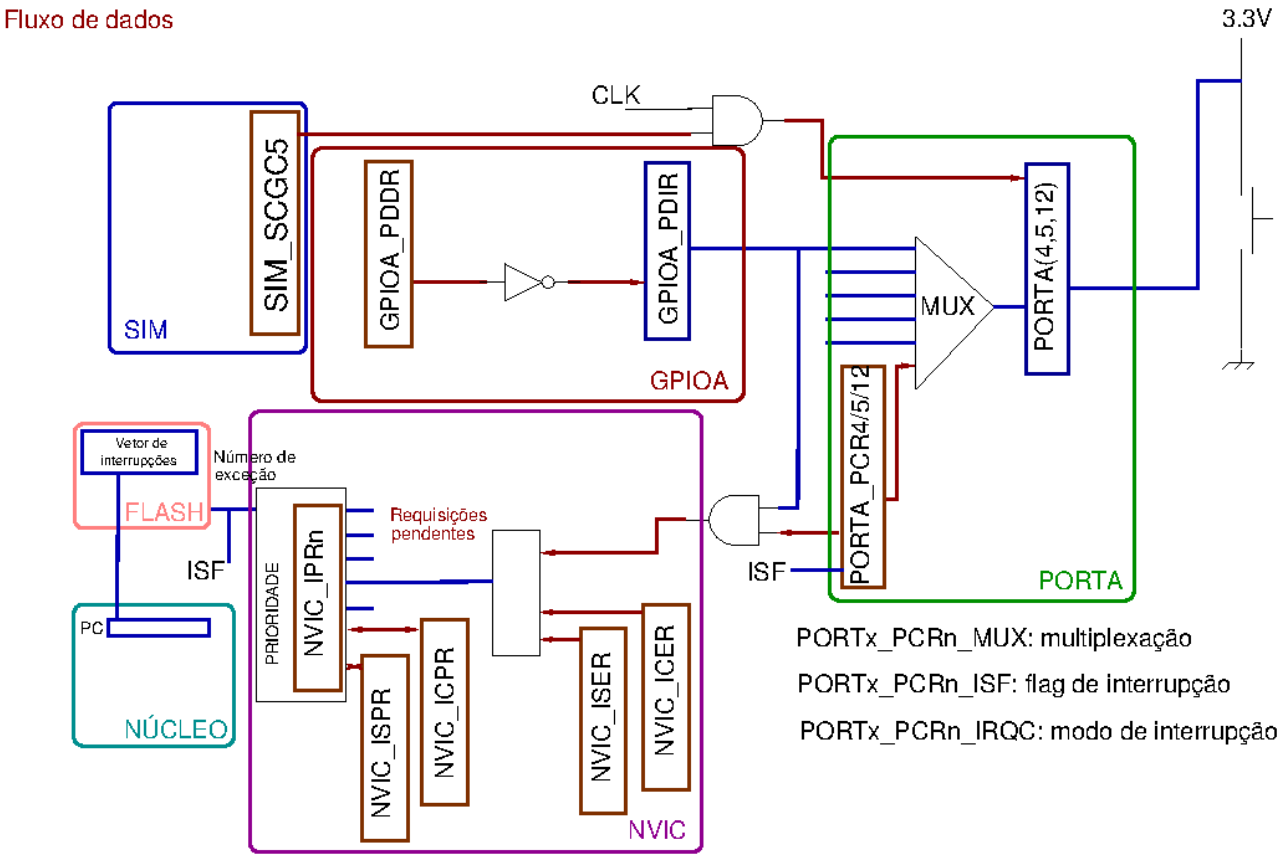

Figura 1: Fluxo de dados no processamento de uma interrupção.No momento de criação de um novo projeto, o ambiente CodeWarrior gera automaticamente os arquivos `Project_Settings/Startup_Code/kinetis_sysinit.c` e `Project_Settings/Linker_Files/MKL25Z128_flash.ld`. Estes contém, respectivamente, o conteúdo do segmento `.vectortable` (tabela de vetor de interrupções) e a posição da memória onde este segmento será relocado. Esta relocação ocorre automaticamente na carga do *firmware* no microcontrolador. Cada entrada i da tabela é o endereço (inicial) da rotina de serviço que processa uma requisição de número de exceção i. Como as **interrupções** do nosso microcontrolador são **vetorizadas**, quando o controlador NVIC torna ativa uma requisição, passa para o processador a sua identificação (número de exceção) através da qual o processador consegue obter o endereço da rotina de serviço pela tabela de vetores e carregá-lo no contador de programa (PC), de forma que a próxima instrução a ser executada seja a da rotina de serviço/tratamento da interrupção requisitada. Quando há mais de uma solicitação de interrupção, o NVIC arbitra automaticamente a exceção de maior prioridade de atendimento com base no nível de prioridade das solicitações, com base na política de arbitragem adotada pelo microcontrolador e o nível de prioridade em cada grupo configurado no registrador NVIC_IPRn (Seção B.1.5 em [4]).

Com tantas funções relativas à interrupção já implementadas no nosso microcontrolador, **o trabalho do programador se reduz a configurar os módulos do nosso MCU para que eles operem no modo de interrupção e a customizar a forma de tratamento de cada exceção através da reprogramação das rotinas de serviço**. E, quando se trata de um evento externo, é necessário configurar o NVIC para ele arbitrar a prioridade de atendimento de múltiplas e aninhadas interrupções externas.

EXPERIMENTO

Neste experimento vamos criar uma nova versão para o projeto "escolha_cor" do Experimento 6, em que a escolha de cor seja feita pela botoeira IRQ5 e que a captura dos acionamentos desta botoeira seja feita por interrupção.

1. Em quais endereços os 12 registradores do NVIC são mapeados? Qual é a função de cada *bit* deste registrador?

2. Além do *bit* 24 do registrador PORTx_PCRn que mostra o estado de interrupção em cada pino individualmente, há um registrador de estado de interrupção de 32 *bits* para cada porta que mostra o estado de interrupção de todos os pinos. Ou seja, é possível consultar o estado de interrupção pelo *bit* 24 do registrador PORTx_PCRn de cada pino individualmente ou o estado de interrupção de todos os pinos de uma porta por um único registrador no microcontrolador KL25Z128M4. Qual é este registrador? Em qual endereço ele está mapeado?

3. Como é chamada a função PORTA_IRQHandler no projeto **interrupt**? Qual é a ordem de prioridade de atendimento das chaves NMI, IRQ5 e IRQ12? Justifique.

4. Implemente o aplicativo **escolha_cor_interrupt** em C por técnica de interrupção.

5. Documente todas as funções que não foram geradas pelo IDE CodeWarrior.

RELATÓRIO

Para este experimento, responda as questões 1 a 3 num arquivo em **pdf**, implemente e documente o projeto **escolha_cor_interrupt.** 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] Esquemático do *shield* EA871
ftp://ftp.dca.fee.unicamp.br/pub/docs/ea871/complementos/Esquematico_EA871-Rev3.pdf
[2] KL25 Sub-Family Reference Manual, Rev. 3, September 2012 (pg 29)
ftp://ftp.dca.fee.unicamp.br/pub/docs/ea871/ARM/KL25P80M48SF0RM.pdf#page=29
[3] KL25 Sub-Family Reference Manual, Rev. 3, September 2012 (pg. 51)
ftp://ftp.dca.fee.unicamp.br/pub/docs/ea871/ARM/KL25P80M48SF0RM.pdf#page=51
[4] ARMv6-M Architecture Reference manual (pg. 218)
ftp://ftp.dca.fee.unicamp.br/pub/docs/ea871/ARM/ARMv6-M.pdf#page=218

Outubro de 2020