11 - Interrupção e Exceções

Rafael Corsi Ferrão

corsiferrao@gmail.com

20 de abril de 2016

Entregar em formato PDF via github.

1 Exceções

Exceções * são eventos que causam uma mudança no fluxo de execução do programa, que quando ocorridas levam a unidade de processamento a executar uma parte específica do código chamada de : *exception handler*. Depois do término da execução da exceção o programa principal volta a ser executado normalmente. No ARM interrupções são um tipo de exceção, normalmente geradas pelos periféricos do microcontrolador.

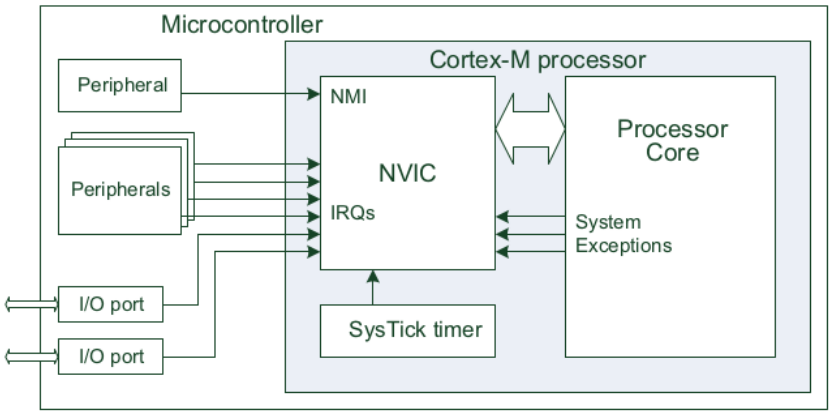

Figura 1: Fontes de exceções
Definitive Guide to ARM Cortex-M3 and Cortex-M4 Processors, The - Yiu, Joseph

O hardware responsável por gerenciar as exceções no ARM é chamado de *Nested vectored interrupt controller* (NVIC). O NVIC pode suportar até 256 diferentes exceções, sendo elas classificadas basicamente em quatro grupos :

*https://www.ece.umd.edu/class/enee447.S2016/ARM-Documentation/ARM-Interrupts-1.pdf

- System Exceptions
- Fault Detection
- Non-Maskable Interrupt (NMI)
- Interrupt Requests (IRQ)

Questão. 1.1: NMI vs IRQ

Qual a diferença entre as exceções NMI e IRQ ?

1.1 Exemplos

- O PIO pode gerar uma interrupção quando acontecer uma mudança de nível em uma entrada;
- O periférico do USB pode gerar uma interrupção quando um dado novo chegar; ou quando a transmissão de um dado finalizar;
- O *timer* pode gerar uma interrupção quando alcançado um determinado valor;

2 Interrupção

Quando um periférico impõem uma interrupção ao NVIC, o seguinte acontece :

1. O processador suspende a execução do código;
2. O processador executa o serviço de rotina da interrupção (*Interrupt Service Routine* - ISR);
3. O processador retoma a execução do código antes da interrupção acontecer.

Questão. 2.1: IRQ vs ISR

Qual a diferença entre as exceções IRQ e ISR ?

Devemos notar que o processador deve salvar os contextos (registradores do core) na primeira passagem ($1 \rightarrow 2$) e após executar o ISR, recarregar os valores na passagem ($2 \rightarrow 3$).

2.1 Prioridades

No ARM, podemos classificar as interrupções por prioridade sendo a de número menor considerada de MAIOR prioridade e de número maior de MENOR prioridade. O

ARM permite que tenhamos uma gama de 256 níveis de prioridades distintas porém fica a cargo do fabricante decidir a quantidade de níveis.

Questão. 2.2: SAM4S

No ARM que utilizamos no curso, quantas são as interrupções suportadas e qual a sua menor prioridade ?

Quando duas interrupções acontecem (não necessariamente simultaneamente) o NVIC verificará qual é a de maior prioridade e a executará primeiro, após sua execução é chamada o ISR da interrupção de menor prioridade.

A Fig. 2 ilustra o que acontece quando uma interrupção é ativada quando um sistema operacional está em uso, nesse caso existem dois tipos distintos de interrupção : IRQ e FIQ (*Fast Interruption Routine*).

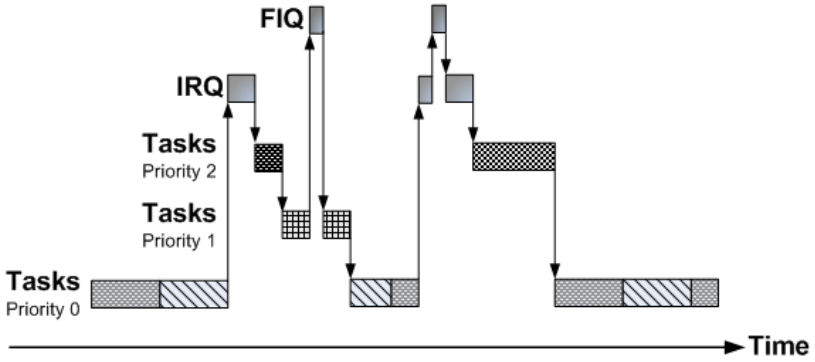

Figura 2: Ilustração do fluxo de interrupção
`http://www.keil.com/support/man/docs/rlarm/rlarm_ar_inter_funct.htm`

Questão. 2.3: FIQ

Descreva o uso do FIQ.

Questão. 2.4: IRQ vs FIQ

No diagrama anterior, quem possui maior prioridade IRQ ou FIQ ?

2.2 Interrupt Requests - IRQ

As interrupções do tipo IRQs, geradas pelos periféricos são "mascaradas", ou seja, devemos ativar em em um registrador ($\boxed{NVIC\_ISERx}$) quais serão as interrupções que estarão ativas.

Questão. 2.5: SAM4S número da interrupção dos periféricos

No datasheet, secção 11.1 informa o ID do periférico que está associado com a sua interrupção. Busque a informação e liste o ID dos seguintes periféricos :

- PIOA
- PIOB
- TC0

Além de ativarmos a interrupção do periférico específico, precisamos definir sua prioridade. Após o reset do uC o ARM configura todas as prioridades para o nível 0 (mais alto). Isso é feito pelos registradores $\boxed{NVIC\_IPRx}$.

Alguns periféricos, após ativarem a interrupção necessitam que o processador limpe um registrador específico para que a interrupção seja desativada.

Questão. 2.6: Limpando interrupção

O que aconteceria caso não limpemos a interrupção ?

2.3 Interrupt Service Routine - ISR

Após uma interrupção ser detectada pelo NVIC, o CORE salva os contextos e aponta a execução do código para uma região específica. Utilizaremos uma função nessa região específica para lidar com a interrupção.

Questão. 2.7: Latência da interrupção.

O que é latência na resolução de uma interrupção, o que é feito nesse tempo ? (*Interrupt latency*).

Questão. 2.8: LatÊncia Cortex M4.

De quantos ciclos é a latência do ARM Cortex M4 ?

Uma interrupção no uC pode estar nos seguintes estados :

- Cada interrupção pode estar desativada (padrão) ou ativada;
- Cada interrupção pode estar pendente (esperando para ser executada) ou não pendente;
- Cada interrupção pode estar ativada (em execução) ou inativada.

Podemos fazer diferentes combinações dos atributos listados anteriormente, por

exemplo, enquanto estivermos lidando com uma interrupção (ativada) podemos desativar lá para que a mesma interrupção não seja chamada novamente quando a interrupção acabar de ser executada.

Para uma interrupção ser aceita, devemos ter o seguinte cenário :

- A interrupção está em pendência,
- A interrupção está ativada, e,
- A prioridade da interrupção é maior (menor valor) do que o nível atual.

3 Software - CMSIS

Utilizaremos as funções definidas no *Cortex Microcontroller Software Interface Standard* (CMSIS) * para configurar o NVIC e o CORE, essas funções são de uso geral do ARM Cortex e são independentes do fabricante (Atmel, Texas, ...).

As funções utilizadas serão :

```
//Set the priority grouping
void NVIC_SetPriorityGrouping(uint32_t priority_grouping)

//Enable IRQn
void NVIC_EnableIRQ(IRQn_t IRQn)

// Disable IRQn
void NVIC_DisableIRQ(IRQn_t IRQn)

// Return true (IRQ-Number) if IRQn is pending
uint32_t NVIC_GetPendingIRQ (IRQn_t IRQn)

// Set IRQn pending
void NVIC_SetPendingIRQ (IRQn_t IRQn)

// Clear IRQn pending status
void NVIC_ClearPendingIRQ (IRQn_t IRQn)

// Return the IRQ number of the active interrupt
uint32_t NVIC_GetActive (IRQn_t IRQn)

// Set priority for IRQn
void NVIC_SetPriority (IRQn_t IRQn, uint32_t priority)

// Read priority of IRQn
uint32_t NVIC_GetPriority (IRQn_t IRQn)

// Reset the system
```

*http://www.arm.com/products/processors/cortex-m/cortex-microcontroller-software-interface-standar
php

```
void NVIC_SystemReset (void)
```

Essas funções apenas configura o NVIC + CORE, devemos também configurar o periférico que será responsável por gerar a interrupção.

4 Software - ASF

Atmel Software Framework (ASF) * é uma ferramenta disponibilizada pela Atmel para programação dos seus uC, nela encontramos diversas funções portáveis que possibilitam ao desenvolvedor em utilizar seus periféricos via uma camada de abstração de software.

Podemos encontrar Drivers tais como :

- Acesso aos PIO
- Controle dos Timers
- USB, TFT, ...

No link a seguir, podemos encontrar a lista completa dos drivers disponíveis :

- `http://asf.atmel.com/docs/latest/get_started.html`

Questão. 4.1: ASF - PIO

Via documentação disponível no ASF, verifique as funções disponíveis para controlar o PIO. Qual a semelhança com as funções desenvolvidas em sala ?

Questão. 4.2: ASF - Timer Counter (TC)

Via documentação disponível no ASF, descreva o uso das seguintes funções do Timer Counter.

- tc_init()
- tc_start()
- tc_enable_interrupt()

5 PIO - Interrupção

A interrupção no PIO é gerenciada por meio de registradores e pode ser configurada para detectar :

*`http://asf.atmel.com/docs/latest/architecture.html`

- Rising edge detection
- Falling edge detection
- Low-level detection
- High-level detection

Questão. 5.1: PIO - Interrupção Botão

Qual deve ser a configuração para operarmos com interrupção no botão do kit SAM4S-EK2 ?

Uma visão geral do hardware responsável por gerenciar as interrupções é demonstrado na Fig. 3.

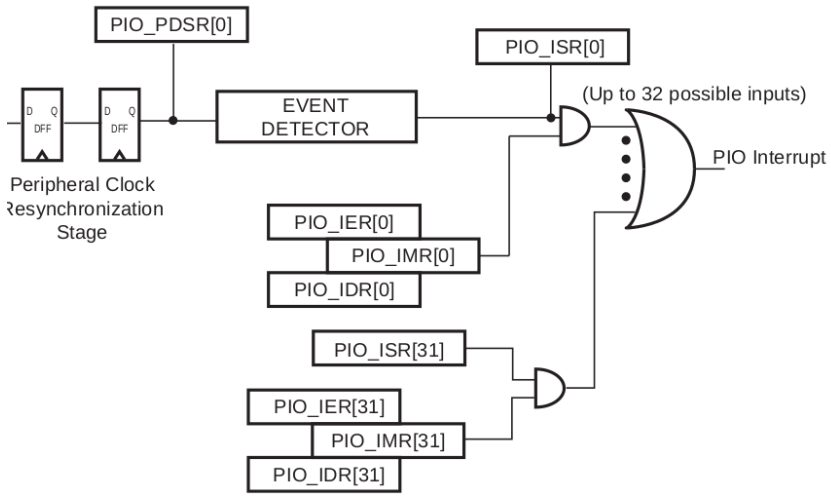

Figura 3: PIO interrupção
SAM4S Datasheet

O texto a seguir foi extraído do datasheet do SAM4S e descreve a operação dessa parte do PIO :

31.5.10 Input Edge/Level Interrupt

The PIO Controller can be programmed to generate an interrupt when it detects an edge or a level on an I/O line. The Input Edge/Level interrupt is controlled by writing the Interrupt Enable Register (PIO_IER) and the Interrupt Disable Register (PIO_IDR), which enable and disable the input change interrupt respectively by setting and clearing the corresponding bit in the Interrupt Mask Register (PIO_IMR). As input change detection is possible only by comparing two successive samplings of the input of the I/O line, the peripheral clock must be enabled. The Input Change interrupt is available regardless of the configuration of the I/O line, i.e., configured as an input only, controlled by the PIO Controller or assigned to a peripheral function.

By default, the interrupt can be generated at any time an edge is detected on the input.

Some additional interrupt modes can be enabled/disabled by writing in the Additional Interrupt Modes Enable Register (PIO_AIMER) and Additional Interrupt Modes Disable Register (PIO_AIMDR). The current state of this selection can be read through the Additional Interrupt Modes Mask Register (PIO_AIMMR).

These additional modes are:

- *Rising edge detection*
- *Falling edge detection*
- *Low-level detection*
- *High-level detection*

In order to select an additional interrupt mode:

- *The type of event detection (edge or level) must be selected by writing in the Edge Select Register (PIO_ESR) and Level Select Register (PIO_LSR) which select, respectively, the edge and level detection.*
- *The current status of this selection is accessible through the Edge/Level Status Register (PIO_ELSR).*

The polarity of the event detection (rising/falling edge or high/low-level) must be selected by writing in the Falling Edge/Low-Level Select Register (PIO_FELLSR) and Rising Edge/High-Level Select Register (PIO_REHLSR) which allow to select falling or rising edge (if edge is selected in PIO_ELSR) edge or high- or low-level detection (if level is selected in PIO_ELSR). The current status of this selection is accessible through the Fall/Rise - Low/High Status Register (PIO_FRLHSR).

When an input edge or level is detected on an I/O line, the corresponding bit in the Interrupt Status Register (PIO_ISR) is set. If the corresponding bit in PIO_IMR is set, the PIO Controller interrupt line is asserted. The interrupt signals of the 32 channels are ORed-wired together to generate a single interrupt signal to the interrupt controller.

When the software reads PIO_ISR, all the interrupts are automatically cleared. This signifies that all the interrupts that are pending when PIO_ISR is read must be handled. When an Interrupt is enabled on a "level", the interrupt is generated as long as the interrupt source is not cleared, even if some read accesses in PIO_ISR are performed.

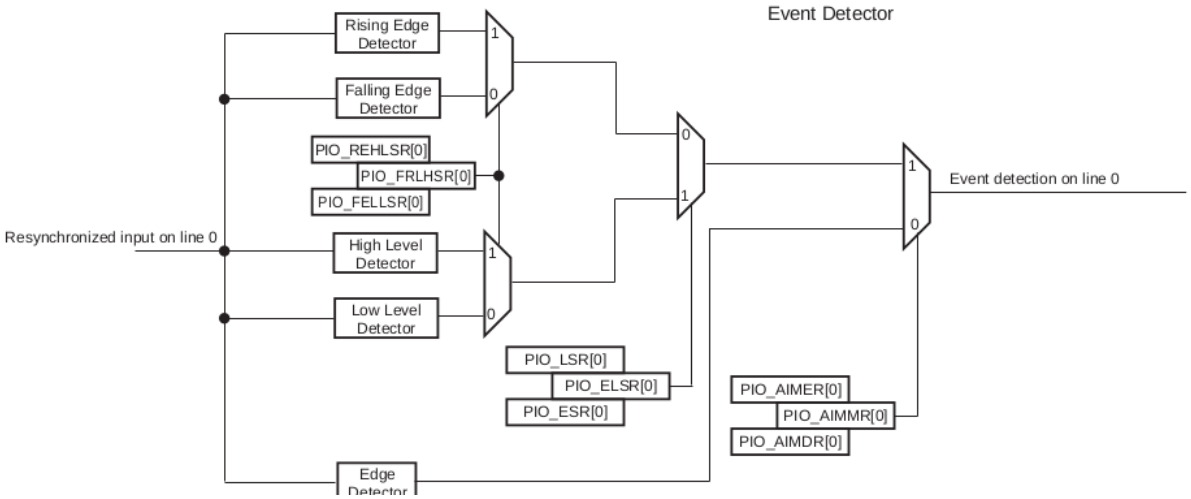

Figura 4: Detecção de eventos
SAM4S Datasheet

Questão. 5.2: PIO - Interrupção

Com base no texto anterior e nos diagramas de blocos descreva o uso da interrupção e suas opções.

Questão. 5.3: Registradores Interrupção

Descreva as funções dos registradores :

- $\boxed{PIO\_IER}$ / $\boxed{PIO\_IDR}$
- $\boxed{PIO\_AIMER}$ / $\boxed{PIO\_AIMDR}$
- $\boxed{PIO\_ELSR}$
- $\boxed{PIO\_FRLHSR}$

6 Programa

O Software disponibilizado em *"Codigos/11 - INTERRUPCAO_PIO"* possui um atalho de quais funções utilizar para ativarmos uma interrupção no botão.

Utilize esse projeto para desenvolver um software que :

6.1 Interrupção + blink

- Pisque o led Azul a uma frequência definida
- Mude o estado do led verde quando um botão for pressionado (utilizando interrupção).

6.2 Interrupção + sleep mode

Coloque o uC em modo sleep e mude o estado do LED quando um botão for pressionado.