**Pesquisa: Interrupção e Exceções**

Por Lucas Scarlato Astur

29 de março de 2017

* (Questão 2.1)

NMI ou Non-Maskable Interruption é um tipo de interrupção de hardware não mascarável, ou seja, não pode ser ignorada pelo sistema. Geralmente Utilizado em situações em que o tempo de resposta da exceção é crítico, ou quando o hardware se encontra numa posição não recuperável, por exemplo por uma falha de corrupção de dados na memória ou do chipset. Possui seus próprios pinos de entrada no hardware

Exemplo de exceção NMI:



*Fig. 1 – tela de interrupção do Windows 8/8.1 ativado pelo erro NMI\_HARDWARE\_FAILURE*

IRQ ou Interruption Request também é um tipo de interrupção de hardware, porém com a opção da CPU ignorá-la, ou executá-la e depois retomar outros processos. Também possui seus próprios pinos de entrada no hardware

Exemplo de exceção IRQ:

- Recebimento de pacotes de dados de Rede

* (Questão 3.1)

O ISR, chamado de Interruption Service Routine é, na verdade, um handler que irá lidar com um IRQ acionado.

* (Questão 3.2)

O NVIC do SAM E70 suporta até 72 serviços de interrupção, cada um com até oito níveis de prioridade.

* (Questão 3.3)

O FIQ é um serviço de interrupção de alta prioridade, capaz de desativar outros serviços de interrupção, como o IRQ, durante sua execução.

* (Questão 3.4)

FIQ.

* (Questão 3.5)

-PIOA: 10

-PIOC: 12

-T0: 23

* (Questão 3.6)

Caso a interrupção não seja “limpa”, os outros processos parados de prioridade inferior nunca serão resumidos, travando o sistema.

* (Questão 3.7)

Interrupt Latency é o tempo passado desde o envio da notificação da interrupção até o começo da execução do serviço relacionado a ela. Durante esta latência, códigos na fila de processamento (de menor prioridade) continuarão a ser executados até a interrupção entrar em processo.

* (Questão 5.1)

De acordo com o próprio código-exemplo da aula, temos a seguinte configuração para o tratamento de interrupções para o botão do sam e70:

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Rafael Corsi - Insper

\* rafael.corsi@insper.edu.br

\*

\* Computação Embarcada

\*

\* 10-PIO-INTERRUPCAO

\*

\* [ref] http://www.atmel.com/Images/Atmel-42142-SAM-AT03258-Using-Low-Power-Mode-in-SAM4E-Microcontroller\_Application-Note.pdf

\* [ref] https://www.eecs.umich.edu/courses/eecs373/labs/refs/M3%20Guide.pdf

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include "asf.h"

#include "conf\_clock.h"

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Defines \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*

\* LEDs

\*/

#define LED\_PIO\_ID ID\_PIOC

#define LED\_PIO PIOC

#define LED\_PIN 8

#define LED\_PIN\_MASK (1<<LED\_PIN)

/\*\*

\* Botão

\*/

#define BUT\_PIO\_ID ID\_PIOA

#define BUT\_PIO PIOA

#define BUT\_PIN 11

#define BUT\_PIN\_MASK (1 << BUT\_PIN)

#define BUT\_DEBOUNCING\_VALUE 79

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* prototype \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void led\_init(int estado);

void but\_init(void);

void but\_Handler();

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Interrupçcões \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void but\_Handler(){

/\*

\* limpa interrupcao do PIO

\*/

*uint32\_t* pioIntStatus;

pioIntStatus = pio\_get\_interrupt\_status(BUT\_PIO);

/\*\*

\* Toggle status led

\*/

if(pio\_get\_output\_data\_status(LED\_PIO, LED\_PIN\_MASK))

pio\_clear(LED\_PIO, LED\_PIN\_MASK);

else

pio\_set(LED\_PIO,LED\_PIN\_MASK);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Funções \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*

\* @Brief Inicializa o pino do LED

\*/

void led\_init(int estado){

pmc\_enable\_periph\_clk(LED\_PIO\_ID);

pio\_set\_output(LED\_PIO, LED\_PIN\_MASK, 1, 0, 0 );

};

/\*\*

\* @Brief Inicializa o pino do BUT

\* config. botao em modo entrada enquanto

\* ativa e configura sua interrupcao.

\*/

void but\_init(void){

/\* config. pino botao em modo de entrada \*/

pmc\_enable\_periph\_clk(BUT\_PIO\_ID);

pio\_set\_input(BUT\_PIO, BUT\_PIN\_MASK, PIO\_PULLUP | PIO\_DEBOUNCE);

/\* config. interrupcao em borda de descida no botao do kit \*/

/\* indica funcao (but\_Handler) a ser chamada quando houver uma interrupção \*/

pio\_enable\_interrupt(BUT\_PIO, BUT\_PIN\_MASK);

pio\_handler\_set(BUT\_PIO, BUT\_PIO\_ID, BUT\_PIN\_MASK, PIO\_IT\_FALL\_EDGE, but\_Handler);

/\* habilita interrupçcão do PIO que controla o botao \*/

/\* e configura sua prioridade \*/

NVIC\_EnableIRQ(BUT\_PIO\_ID);

NVIC\_SetPriority(BUT\_PIO\_ID, 1);

};

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Main \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int main(void)

{

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Inicialização básica do uC \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

sysclk\_init();

WDT->WDT\_MR = WDT\_MR\_WDDIS;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Inicializao I/OS \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

led\_init(1);

but\_init();

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Super loop \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

while(1){

/\* entra em modo sleep \*/

//pmc\_sleep(SLEEPMGR\_SLEEP\_WFI);

};

}

* (Questão 5.2)

Uma interrupção pode ser utilizada pelos PIOs justamente para evitar a checagem constante de um valor em um registrador acionando o evento que vai, por vez, acionar o uC apenas quando necessário. Isso permite manter o Uc em um modo de baixa energia (sleep mode). Por exemplo, ao invés de checar o status de um botão constantemente, um handler pode enviar uma mensagem de mudança de satus diretamente ao Uc.

* (Questão 5.3)

-PIO\_IER / PIO\_IDR controlam o enable e o disable da interrupção de seu PIO.

-PIO\_AIMER / PIO\_AIMDR controlam o enable e o disable de uma interrupção adicional de seu PIO

-PIO\_ELSR irá informar se a interrupção será sensível as bordas ou sensível ao nível

-PIO\_FRLHSR uma vez definido a sensibilidade no PIO\_ELSR, este irá informar se a interrupção será ativada durante a borda de subida ou descida, ou no nível alto ou baixo.