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:

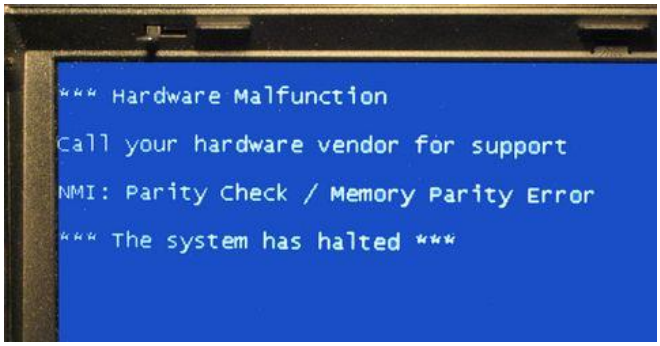

Fig. 1 – tela de interrupção do Windows 8/8.1 ativado pelo erro NMI_HARDWARE_FAILURE

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
```

```c
#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.