10 Pesquisa – Interrupção e Exceções

Pedro Cunial Campos

March 29, 2017

Contents

1 Introdução

2 Exceções

2.1 Qual a diferença entre as exceções NMI e IRQ?

As exceções NMI (Non-maskable Interrupt) que não podem ser ignoradas por técnicas comuns de mascaramento de interrupções, ou seja, são interrupções de maior prioridade, comumente utilizadas em caso de erros críticos que podem gerar estados não recuperáveis do software ou hardware caso não sejam tratadas com urgência.

Enquanto isso, as IRQ (Interrupt Request) são interrupções de menor prioridade que, como o nome já sugere, adiciona um pedido de interrupções na stack de um manipulador de interrupções do sistema. IRQ costumam ser utilizadas para a manipulação de eventos, sejam eles de recebimentos de dados, ou de teclas precionadas.

3 Interrupção

3.1 Qual a diferença entre exceções IRQ e ISR?

Enquanto as IRQ inserem requesições de interrupções direto na stack do manipulador de eventos, enquanto as ISR são tratadas pelo seu respectivo interruption handler.

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

O NVIC do ARM utilizado no curso suporta 72 interrupções diferentes, sendo o menor nível de prioridade possível 7 (8 níveis diferentes de prioridade).

3.3 Descreva o uso do FIQ.

A FIQ é uma interrupção de altíssima prioridade que desabilita qualquer outro handler (seja o handler comum de eventos e interrupções ou o handler específico de uma ISR ou outra FIQ).

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

Pelo diagrama, o FIQ tem maior prioridade em comparação ao IRQ, o que pode ser explicado pelo fato das FIQs desabilitarem qualquer outro handler ao enquanto acionadas.

3.5 No datasheet, secção 13.1 informa o ID do periférico que está associado com a sua interrupção. Busque a informação e liste o ID dos seguintes periféricos:

- PIOA

  10

- PIOC

  12

- TC0

  23

3.6 O que aconteceria caso não limpemos a interrupção?

Caso não limpemos a interrupção, o NVIC não saberá que a interrupção fora resolvida e, portanto, ficará "preso" no seu processo, podendo apenas receber interrupções de maior prioridade, e sem voltar para a thread principal da sua aplicação.

3.7 O que é é latência na resolução de uma interrupção, o que é feito nesse tempo? (Interrupt Latency)

Em suma, a latência na resolução de uma interrupção é o tempo entre o envio da interrupção e a execução do código associado a mesma. Dependendo do tipo de interrupção, existem diferentes processos que podem ocorrer entre a mesma e sua execução, por exemplo, no caso de interrupções de menor prioridade, o intervalo entre seu sinal e o início de sua execução pode ser utilizado pela execução de uma interrupção de maior prioridade. Em casos de IRQ, por exemplo, sua latência costuma também estar associada à eventos "à sua frente" na stack do manipulador de eventos.

4 Software - CMSIS

5 PIO - Interrupção

5.1 Qual deve ser a configuração para operarmos no botão do kit SAME70-EK2?

Para operarmos um botão pela sua interrupção, devemos utilizar um handler associado ao mesmo, como feito no exemplo da aula 10. O trecho de código abaixo exemplifica seu uso. Repare na função $\text{pio}_{\text{handlerset}}$, é nela que associamos ao botão o seu handler.

```
/**
* @Brief Inicializa o pino do BUT
*  config. botao em modo entrada enquanto
*  ativa e configura sua interrupcao.
*/
void but_init(
    Pio *p_but_pio,
    const u_int32_t pio_id,
    const u_int32_t but_pin_mask)
{
  /* config. pino botao em modo de entrada */
  pmc_enable_periph_clk(pio_id);
  pio_set_input(p_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(p_but_pio, but_pin_mask);
  switch (but_pin_mask) {
    case BUT_PIN_MASK:
      pio_handler_set(p_but_pio, pio_id, but_pin_mask, PIO_IT_FALL_EDGE, but_Handler);
      break;
    case BUT1_PIN_MASK:
      pio_handler_set(p_but_pio, pio_id, but_pin_mask, PIO_IT_RISE_EDGE, but1_Handler)
      break;
    case BUT2_PIN_MASK:
      pio_handler_set(p_but_pio, pio_id, but_pin_mask, PIO_IT_FALL_EDGE, but2_Handler)
      break;
    case BUT3_PIN_MASK:
```

```
      pio_handler_set(p_but_pio, pio_id, but_pin_mask, PIO_IT_RE_OR_HL, but3_Handler);
      break;
  }

  /* habilita interrupçcão do PIO que controla o botao */
  /* e configura sua prioridade                        */
  NVIC_EnableIRQ(pio_id);
  NVIC_SetPriority(pio_id, 1);
}

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);
}
```

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

Um PIO Controller pode gerar uma interrupção em dadas variações de seu respectivo valor, por exemplo, podemos associar uma interrupção ao pressionar um dado botão, fazendo com que o uC só seja de fato requisitado no evento do pressionar deste dado botão.

Da mesma forma, podemos associar eventos ao pressionar e ao soltar de um dado botão, podendo reproduzir o código de acender o LED enquanto o botão estiver pressionado utilizando apenas eventos de pressionar e soltar o botão, mantendo o uC em sleep mode (reduzido gasto energético).

5.3 Descreva as funções dos registradores:

- PIO_{IER} / PIO_{IDR}

  Como o próprio nome já sugere, o PIO Interrupt Enable/Disable Register, respectivamente, habilitam ou desabilitam o uso de interrupções associadas ao dado PIO.

- PIO_{AIMER} / PIO_{AIMDR}

  O PIO Additional Interrupt Modes Enable/Disable Register são utilizados para definir opções adicionais possíveis para as interrupções associadas ao dado PIO.

- PIO_{ELSR}

  O PIO Edge/Level Select Register é utilizado para definir se a interrupção será enviada em uma borda (subida ou descida), ou de acordo com o seu nível (alto ou baixo).

- PIO_{FRLHSR}

  O PIO Fall/Rise Low/High Status Register serve para definir se o PIO deve agir na borda de subida ou descida (no caso de estar definido como trabalhando nas bordas) ou no sinal alto ou baixo (no caso de ser definido pelo nível).