UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

Universidade Federal do Ceará - Campus Quixadá

Relatório sobre Interrupção GPIO QXD0149 - Técnicas de Programação para Sistemas Embarcados I

Prof. Dr. Francisco Helder Candido
Outubro 2020

Aluno: Samuel Henrique - **Matrícula:** 473360

Quixadá-CE

Sumário

1 Introdução

Interrupção é uma funcionalidade usada para liberar a CPU de certas tarefas, com isso, a interrupção chama o hardware para executar essas tarefas. Essa prática é baseada na utilização de interrupção. O objetivo da interrupção nessa prática é a utilização do botão de forma apropriada, ou seja, no momento em que ele for apertado, o LED será trocado. Anteriormente isso era feito sem interrupção, porém não tinha o funcionamento correto, pois era preciso apertar o botão no momento de verificação do sinal ou não teria o efeito esperado e a CPU deixaria de fazer outras tarefas por estar ocupada.

2 Interrupções

2.1 Descrição Funcional

Basicamente, o controlador de interrupção processa as interrupções de entrada mascarando e classificando por prioridade para produzir o sinais de interrupção para o processador.

2.1.1 Interromper o Processamento

- Seleção de Entrada

  O INTC suporta apenas interrupção de entrada sensível ao sinal. Um periférico confirmando uma interrupção o mantém até que o software manipule a interrupção e instrua o periférico a desabilitá-la. Uma interrupção de software é gerada se o bit correspondente no registrador INTC_ISR_SETn (módulo: n = [0, 1, 2, 3]) estiver setado, 128 linhas de interrupção são suportadas. A interrupção de software é limpa quando o bit correspondendo no registrador INTC_ISR_CLEARn é setado.

- Máscara Individual

  A detecção de interrupções em cada linha de entrada pode ser habilitada ou desabilitada independentemente pelo registrador de máscara de interrupção, INTC_MIRn. Em resposta a uma interrupção não mascarada, o INTC pode gerar um dos dois tipos de interrupção para o processador:

  IRQ: interrupção de baixa prioridade

  FIQ: interrupção rápida

  O tipo de interrupção é determinada pelo bit 0 no registrador INTC_ILR_0 to INTC_ILR_127. O status de interrupção antes do mascaramento pode ser lido pelo registrador INTC_ITRn.

- Mascaramento de Prioridade

  Para permitir o processamento mais rápido de interrupções de alta prioridade, utiliza-se um mascaramento de prioridade, programável no registrador INTC_THRESHOLD[7:0] no campo PriorityThreshold. Isto permite a preempção por interrupções de prioridade mais alta, as prioridade menor ou igual estão mascaradas; prioridade de nível 0 não será mascarada. O campo PriorityThreshold pode ser definido entre os níveis 0x0 (0) e 0x7F (127), alto e baixo respectivamente. Quando não é necessário, 0xFF é usado para desativar o campo.

- Classificação de Prioridade

  Um nível de prioridade é atribuído para cada linha de interrupção, sendo 0 o nível mais alto. O tipo de interrupção e seu nível de prioridade são definidos no mesmo registrador, o INTC_ILR_0 to INTC_ILR_127.

2.1.2 Registrador de Proteção

Se o bit 0 do registrador INTC_PROTECTION estiver setado, então o acesso aos registradores INTC é restrito apenas ao modo privilegiado. Entre-

tanto o INTC_PROTECTION só pode ser acessado pelo modo privilegiado.

2.1.3 Módulo de Economia de Energia

O INTC fornece uma função de ociosidade automática, ou seja, quando não há atividade no barramento o clock da interface é desabilitado dentro do módulo, reduzindo o consumo de energia, uma enorme vantagem. Quando há uma atividade o clock da interface é reiniciado sem penalidade na latência. Este modo vem desabilitado por padrão, é necessário programá-lo, para isso basta acessar o registrador INTC_SYSCONFIG e configurar o bit 0 como setado para habilitar a função de Autoidle.

Quando não há nenhum tipo de interrupção sendo processada ou pendente é ideal que o clock funcional seja desabilitado, para economizar energia também. Para isso, basta setar o bit 0 do registrador INTC_IDLE, ativando a função FuncIdle. Quando uma nova interrupção de entrada não mascarada é detectada, o clock é reiniciado e o INTC processa a interrupção. Se esse modo for desabilitado a latência de interrupção é reduzida em um ciclo.

2.1.4 Erros de Manipulação

Causarão um erro:

Violação do privilégio, tentativa de acessar os registradores que estão em modo de proteção pelo modo usuário.

Comandos não suportados

Não causarão nenhuma resposta de erro:

Acesso um endereço não codificado

Gravar em um registrador de apenas leitura

2.2 Modelo Básico de Programação

2.2.1 Sequência de Inicialização

1. Se necessário, habilite o clock da interface definindo o bit AutoIdle através do registrador INTC_SYSCONFIG.

2. Programe o registrador INTC_IDLE, se precisar usar o módulo de economia de energia.

3. Programe o registrador INTC_ILRm para definir o tipo de interrupção, FIQ ou IRQ.

4. Programe o registrador INTC_IMRn para habilitar interrupções mascaradas, lembrando que existem os registradores INTC_IMR_SETn e INTC_IMR_CLEARn para facilitar esse processo, mesmo sendo possível programar diretamente o INTC_IMRn.

2.3 Interrupções do ARM Cortex-A8

De acordo com o manual, existem 127 tipos de interrupções diferentes. Entretanto, nessa prática apenas a de GPIO1 foi utilizada, interrupção de número 98.

3 Código Base

Para configurar uma interrupção é necessário habilitar o IRQ através da função **IntMasterIRQEnable**. Após isso, é necessário iniciar o módulo de GPIO1 através da função **gpioInitModule**, depois é preciso definir os pinos como entrada ou saída, os pinos 21, 22, 23 e 24 foram todos configurados como saída, já o pino 28 foi configurado como entrada, através das funções **gpioPinMuxSetup** e **gpioSetDirection**.

O pino 28 é justamente o pino onde será gerada a interrupção, pois o botão estará conectado nele e ela deve ser gerada no momento em que ele for apertado. Para isso é necessário configurar o GPIO1, através da função **gpioAintcConfigure** que tem como parâmetro o número da interrupção, 98 nesse caso. Primeiramente a função **gpioAintcConfigure** limpa os registradores de interrupção, após isso vai ser atribuído a rotina de interrupção a um vetor de interrupções pela sub-rotina **IntRegister**. Em seguida a prioridade de interrupção é definida através de **IntPrioritySet**, salvando no registrador INTC_ILR. O último passo é habilitar a interrupção do sistema, para isso é necessário gravar no registrador INTC_MIR_SET para habilitar uma interrupção mascarada, através da função **IntSystemEnable**.

Para habilitar o pino 28 como interrupção de GPIO é preciso utilizar a função **gpioPinIntConfig**. Essa função tem o objetivo de gravar no registrador GPIO_IRQSTATUS_SET no bit correspondente ao pino que a interrupção IRQ será gerada.

O próximo passo é definir o tipo de interrupção que vai ser utilizada, essa última parte varia muito, pois depende do projeto que será feito. É preciso analisar que tipo de sinal entra no pino, também chamado de tipo de evento, a partir deles é gerada a interrupção. Existem diferentes tipos de sinal: sinal alto, sinal baixo, ambos os sinais, borda de subida, borda de descida, ambas as bordas, sem bordas e sem sinal de baixo ou alto. Para configurar qual tipo de sinal presente no projeto, basta pegar o registrador GPIO correspondente ao tipo de sinal e setá-lo no bit correspondente ao número do pino.

4 Conclusão

Com a prática foi perceptível a imensa importância do uso da interrupção, pois com ela foi possível ter uma precisão maior na troca de LEDs, já que independente de onde o código esteja executando, a interrupção ocorrerá ao apertar o botão, e consequentemente a troca dos LEDs. Além disso, a

interrupção permite uma liberdade maior para CPU, pois quem executa a rotina de interrupção é o hardware, então a CPU fica livre para fazer outra ação, dando performance e velocidade para o programa embarcado.