LABORATÓRIO RTOS

UFC

# Prática 05: Sincronizando com Semáforos

Prof. Francisco Helder

30 de maio de 2025

Sistemas de tempo real precisam tomar ações em resposta a eventos originados do ambiente. Por exemplo, um pacote que chega em um periférico Ethernet (o evento) pode precisar passar para uma pilha TCP/IP para processamento (a ação). Sistemas não triviais terão que atender eventos originados de várias fontes, todas com diferentes sobrecargas de processamento e requisitos de tempo de resposta. Em cada caso, um julgamento deve ser feito quanto à melhor estratégia de implementação de processamento de eventos:

1. Como o evento deve ser detectado? Normalmente, interrupções são usadas, mas as entradas também podem ser por polling.

2. Quando interrupções são usadas, quanto processamento deve ser realizado dentro da rotina de interrupção (ISR) e quanto fora? Normalmente, é desejável manter cada ISR o mais curto possível.

3. Como os eventos podem ser comunicados ao código principal (não ISR) e como esse código pode ser estruturado para acomodar melhor o processamento de ocorrências potencialmente assíncronas?

O FreeRTOS não impõe nenhuma estratégia específica de processamento de eventos ao designer do aplicativo, mas fornece recursos que permitirão que a estratégia escolhida seja implementada de forma simples e sustentável. Deve-se notar que apenas funções de API e macros que terminam em 'FromISR' ou 'FROM_ISR' devem ser usadas em uma rotina de interrupção.

# 1 Sincronizando com Semáforos Binários

Um Semáforo Binário pode ser usado para desbloquear uma tarefa sempre que uma interrupção específica ocorrer, sincronizando efetivamente a tarefa com a interrupção. Isso permite que a maioria do processamento de eventos de interrupção seja implementada dentro da tarefa sincronizada, com apenas uma porção muito rápida e curta permanecendo diretamente no ISR. Diz-se que o processamento de interrupção foi "deferred" para uma tarefa "Handler".

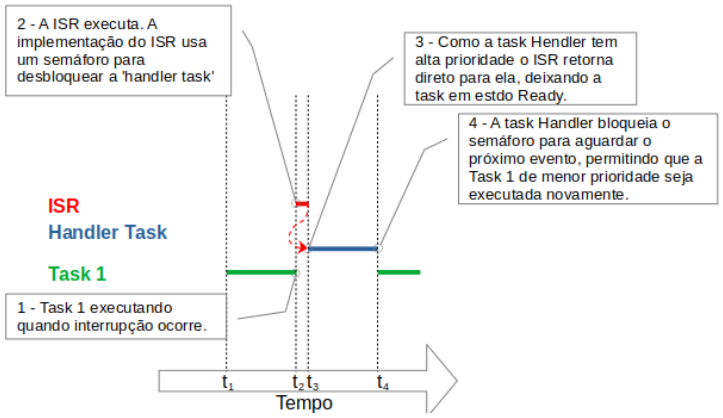


Figura 1: A ISR interrompe uma tarefa, mas retorna para outra.

Se o processamento de interrupção for particularmente crítico em termos de tempo, a prioridade da Task Handler pode ser definida para garantir que ela sempre antecipe a outras tarefas no sistema. Será então a tarefa para a qual o ISR retornará quando o próprio ISR tiver concluído a execução. Isso tem o efeito de garantir que todo o processamento de eventos seja executado

de forma contígua no tempo, como se tudo tivesse sido implementado dentro do próprio ISR, como demonstrado na Figura 1.

A Task Handler usa uma chamada de bloqueio 'take' para um semáforo como um meio de entrar no estado Bloqueado para esperar que o evento ocorra. Quando o evento ocorre, o ISR usa uma operação 'give' no mesmo semáforo para desbloquear a tarefa para que o processamento do evento necessário possa prosseguir.

'Taking' e 'Giving' um semáforo são conceitos que têm vários significados diferentes dependendo do cenário de uso.

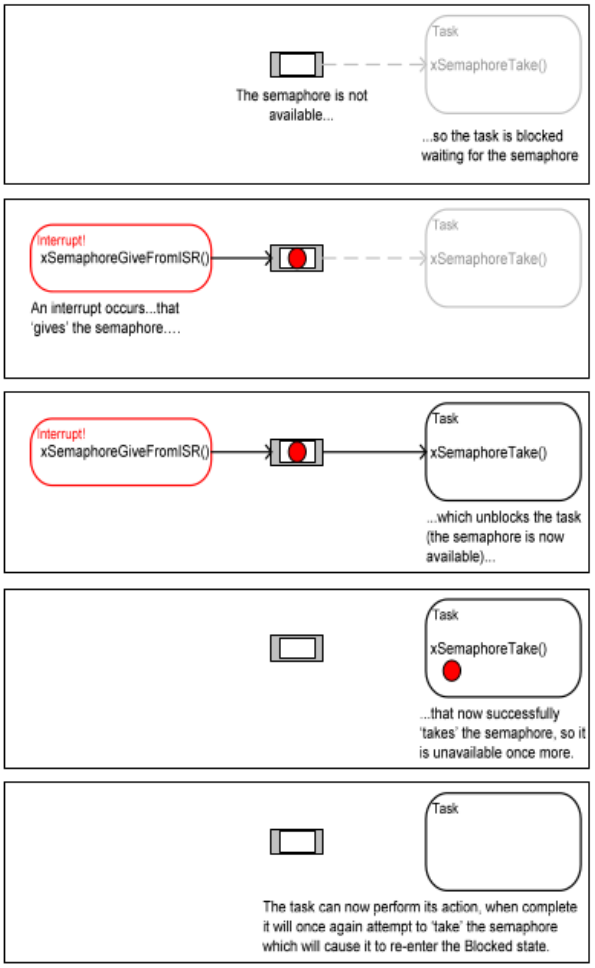


Figura 2: Usando um semáforo binário para sincronizar uma tarefa com uma interrupção.

Neste cenário de sincronização de interrupção, o semáforo pode ser conceitualmente pensado como uma fila que tem um comprimento de um. A fila pode conter no máximo um item a qualquer momento, então está sempre vazia ou cheia (portanto, binária). Ao chamar xSemaphoreTake(), a Task Handler tenta efetivamente ler da fila com um tempo de bloqueio, fazendo com que a tarefa entre no estado Bloqueado se a fila estiver vazia. Quando o evento ocorre, o ISR simplesmente usa a função xSemaphoreGiveFromISR() para colocar um token (o semáforo) na fila, deixando-a cheia. Isso faz com que a Task Handler saia do estado Bloqueado e remova o token, deixando a fila vazia mais uma vez. Depois que a Task Handler conclui seu processamento, ela tenta ler da fila mais uma vez e, encontrando a fila vazia, entra novamente no estado Bloqueado para esperar pelo próximo evento, como vista na Figura 2.

# 2 Contando Eventos com Semáforos

Os semáforos de contagem são normalmente usados para duas coisas:

1 - Contagem de eventos.

Neste cenário uma interrupção “give” um semáforo cada vez que um evento ocorrer, fazendo com que o valor da contagem do semáforo seja incrementado em cada give. Uma Task Handler “take” um semáforo cada vez que processar um evento, fazendo com que o valor da contagem do semáforo seja decrementado em cada tomada. O valor da contagem é a diferença entre o número de eventos que ocorreram e o número que foi processado. Este mecanismo é mostrado na Figura 3. A contagem do semáforo é usado para contar os eventos que são criados, tendo como valor inicial o zero.

2 - Gerenciamento de recursos.

Neste cenário de uso, o valor da contagem indica o número de recursos disponíveis. Para obter o controle de um recurso, uma tarefa deve primeiro obter um semáforo, diminuindo o valor da contagem do semáforo. Quando o valor da contagem chega a zero, não há recursos livres. Quando uma tarefa termina com o recurso, ela “give” o semáforo, incrementando o valor da contagem de semáforos.

A contagem do semáforo é usado para gerenciar os recursos que são criados, desde modo o seu valor de contagem inicial deve ser igual ao número de recursos que estão disponíveis.

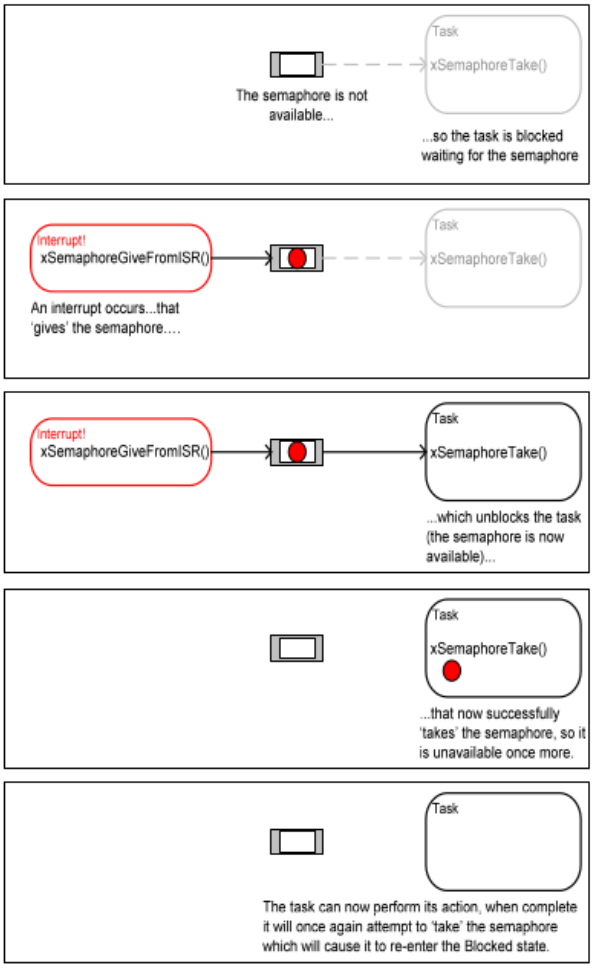


Figura 3: Usando semáforo para ’contar’ eventos.

# 3 Atividade Prática

**pratica 1:**

Nesta atividade, vamos trabalhar com interrupções. A interrupção do botão deve notificar a tarefa do botão via semáforo binário quando o mesmo for pressionado, e quando receber o semáforo, a tarefa do botão irá enviar via queue para a tarefa de led um comando para mudar o status do led.

**pratica 2:**

Nesta atividade iremos sincronizar a interrupção do botão e a tarefa de acionamento dos leds usando semáforo contador. Altere o semáforo usado na tarefa do botão para um semáforo contador.