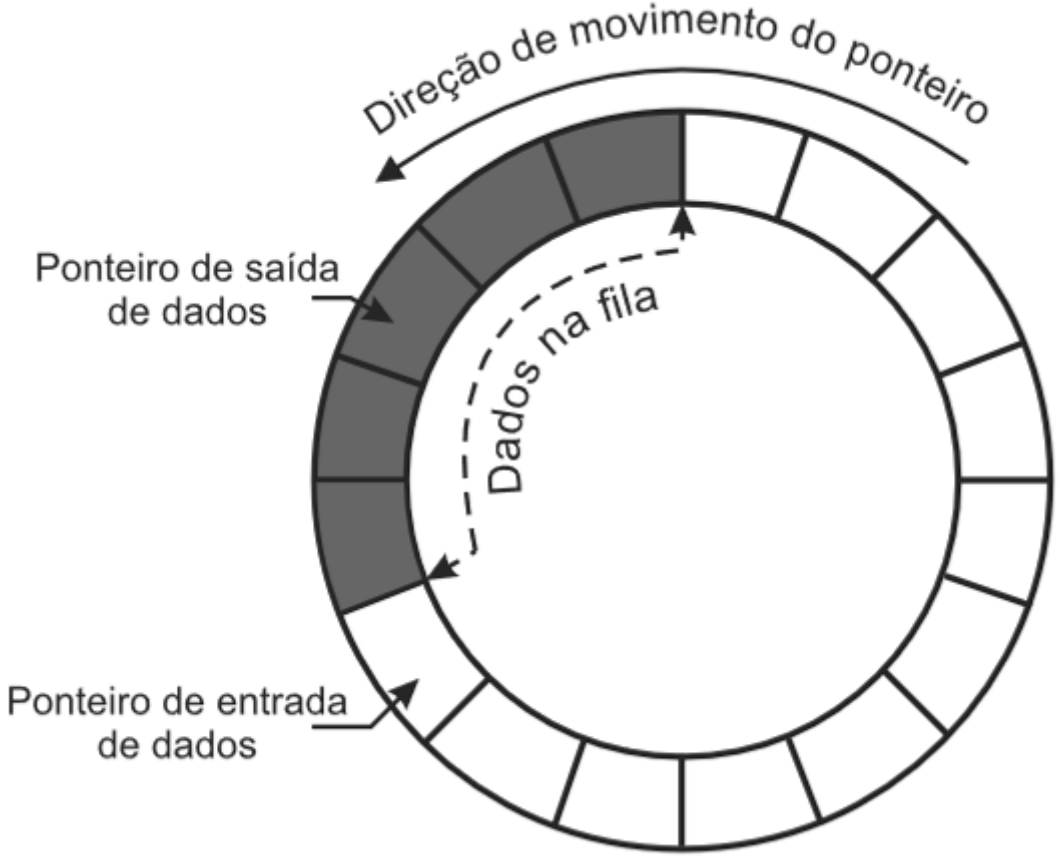
FreeRTOS

Lab 2 – Objetos de Comunicação entre tarefas

Filas e Notificações

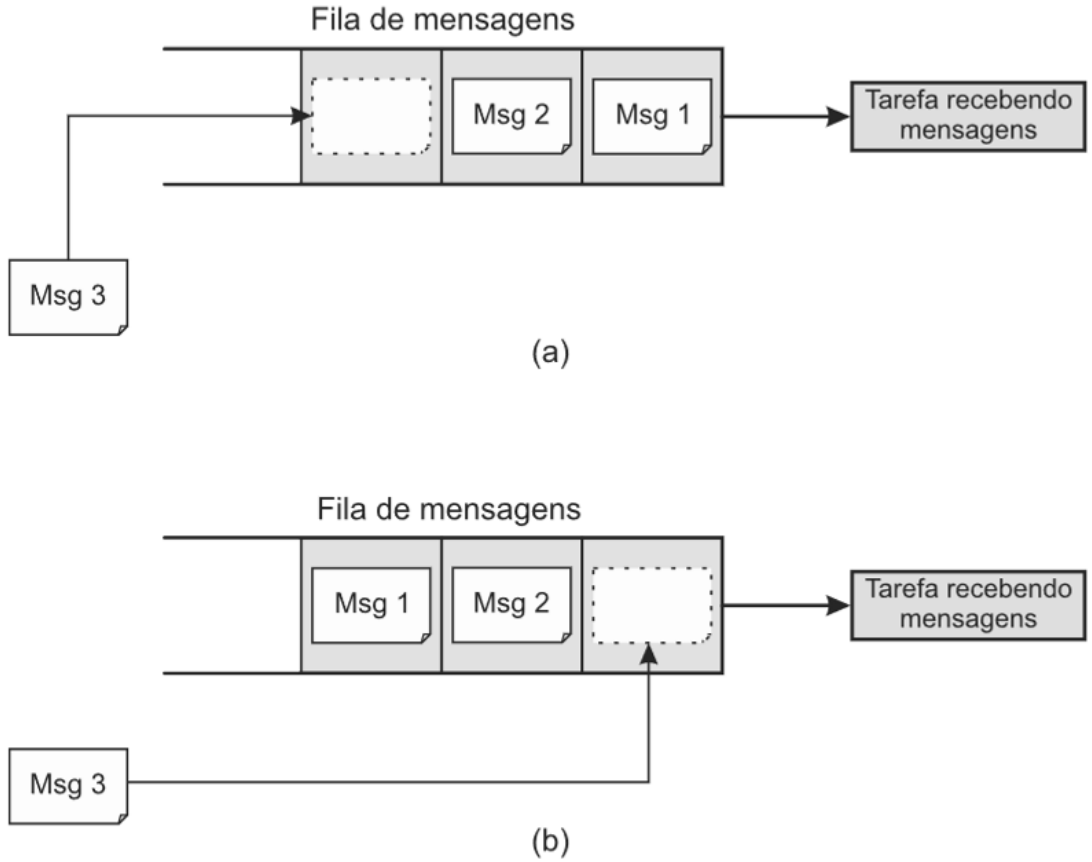
Filas de mensagens

- Uma **fila de mensagem** é um recurso dos núcleos de sistemas operacionais por meio do qual RSI (Rotinas de serviços de interrupção) e tarefas podem enviar e receber mensagens, com o intuito de se comunicarem ou sincronizar dados



Filas de mensagens

- Para o gerenciamento da entrada e retirada de dados de uma fila existem ponteiros de entrada e saída de dados



Criando uma Fila no FREERTOS

- As filas no FreeRTOS têm uma particularidade, que é a possibilidade de enviar uma mensagem para o final ou início da fila simplesmente usando os serviços com final “ToBack” ou “ToFront”, respectivamente

```
// Declara o ponteiro para uma estrutura de fila
xQueueHandle TestQueue;

void Example_Task (void *param){
   /* Configuração da tarefa */
   char mensagem;
   (void)param;
   // Cria uma fila com 128 posições e mensagens do tamanho de uma varável char
   TestQueue = xQueueCreate(128, sizeof(char));
   if( TestQueue == NULL ){
      // Falha na alocação da fila. Trate este erro aqui !!!
   }else{
      // Inicialize aqui o hardware que controlará a interrupção
   }

   /* Laço da tarefa */
   for (;;){
      // Espera indefinidamente por uma interrupção ocorrer
      if(xQueueReceive(TestQueue, &mensagem, portMAX_DELAY) == pdTRUE){
         // Trata mensagem recebida pela interrupção aqui!
      }else{
         // Falha na recepção da mensagem. Trate esta exceção aqui!
      }
   }
}

void Example_interrupt (void){
   // Declara variável utilizada para determinar se haverá preempção
   signed portBASE_TYPE pxHigherPriorityTaskWoken = pdFALSE;

   // Limpa flags da interrupção e recebe dado da porta serial
   char dado = recebe_serial();

   // Copia o dado recebido pela porta serial para o final da fila
   xQueueSendToBackFromISR(TestQueue, &dado, &pxHigherPriorityTaskWoken);

   // Somente troca o contexto se uma tarefa de maior prioridade acordou pelo semáforo
   if (pxHigherPriorityTaskWoken == pdTRUE) portYIELD();
}
```

Exemplo

- Execute o código “SeparaParesDeImpares”

- Execute o código “FilaTextoEnvioChave”

Notificação de tarefas

- A notificação de tarefa é um evento enviado diretamente para a tarefa, podendo adicionar à tarefa notificada a lista de prontos e, opcionalmente, atualizar o valor da variável de notificação dessa tarefa

```
// Declara variável que conterá o manipulador da tarefa a ser notificada
static TaskHandle_t xTaskToNotify = NULL;

void Example_Task (void *param){
   (void)param;

   /* Armazena o manipulador da tarefa a ser notificada. */
   xTaskToNotify = xTaskGetCurrentTaskHandle();

   // Inicialize aqui o hardware que controlará a interrupção

   for (;;){
      // Espera por uma interrupção ocorrer em até 100 ticks do sistema
      if (ulTaskNotifyTake(pdTRUE, 100) == 1){
            // Trata evento gerado pela interrupção aqui!
         }else{
         // Ocorreu timeout. Trate esta exceção aqui!
      }
   }
}

void Example_interrupt (void){
   // Declara variável utilizada para determinar se haverá preempção
   signed portBASE_TYPE pxHigherPriorityTaskWoken = pdFALSE;

   // Limpa flags da interrupção

   // Informa a tarefa indicada por xTaskToNotify que a interrupção ocorreu
   vTaskNotifyGiveFromISR(xTaskToNotify, &pxHigherPriorityTaskWoken);

   // Somente troca o contexto se uma tarefa de maior prioridade acordou pelo semáforo
   if (pxHigherPriorityTaskWoken == pdTRUE){
      portYIELD();
   }
}
```

Referências

- DENARDIN, Gustavo Weber; BARRIQUELLO, Carlos Henrique. Sistemas operacionais de tempo real e sua aplicaçao em sistemas embarcados. Editora Blucher, 2019.
- Programando Multitarefa na prática: Utilizando a linguagem C/C++, freeRTOS e Arduino. Max Back. 2ed.