FUNDAÇÃO EDSON QUEIROZ
UNIVERSIDADE DE FORTALEZA
ENSINANDO E APRENDENDO

T569 –SISTEMAS DE TEMPO REAL

Aula 7

Prof. Marcelo Sousa

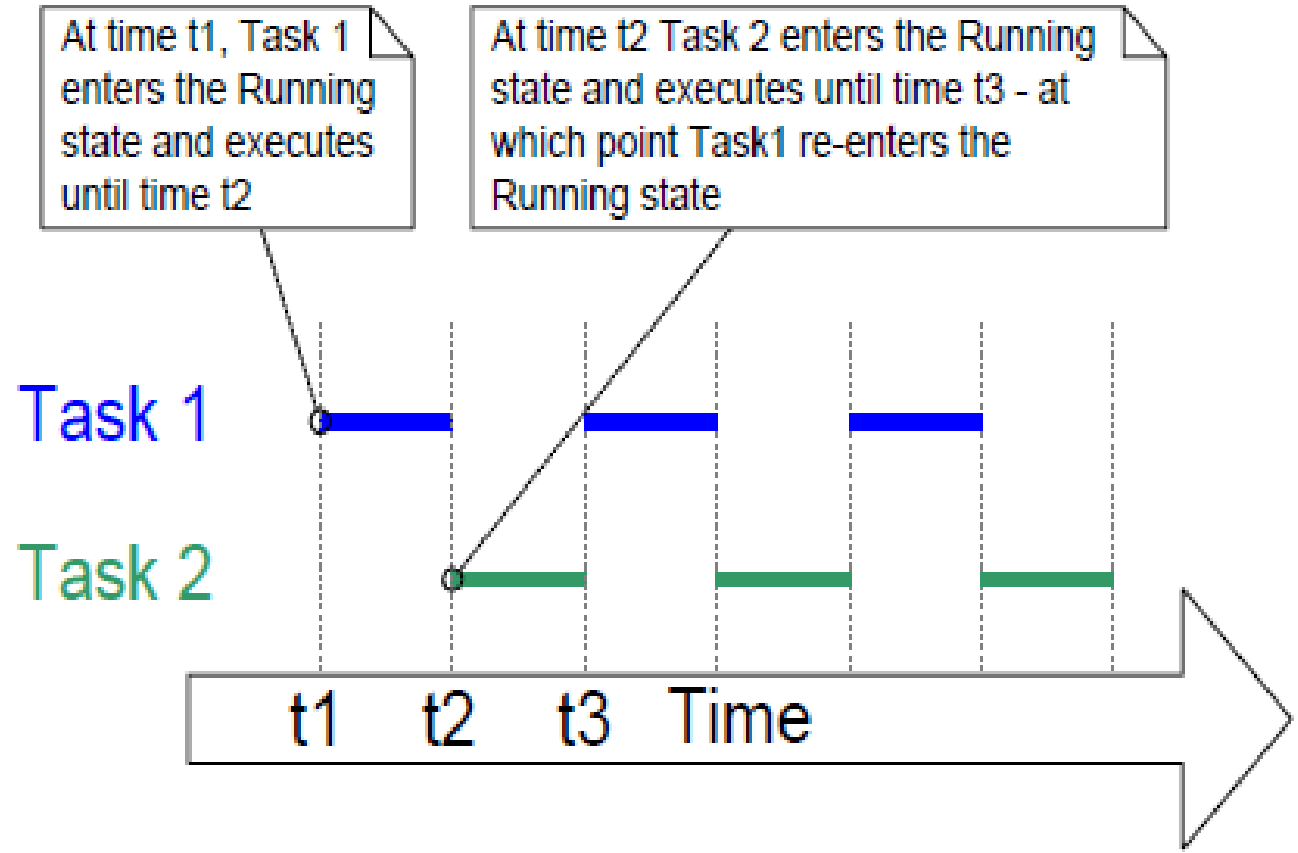
Agenda

- Revisão

- Prioridades de Tasks
  - Prática 3: Escrever Tasks com prioridades

- Estados NOT RUNNING
  - Prática 4:
  - Prática 5:

Revisão

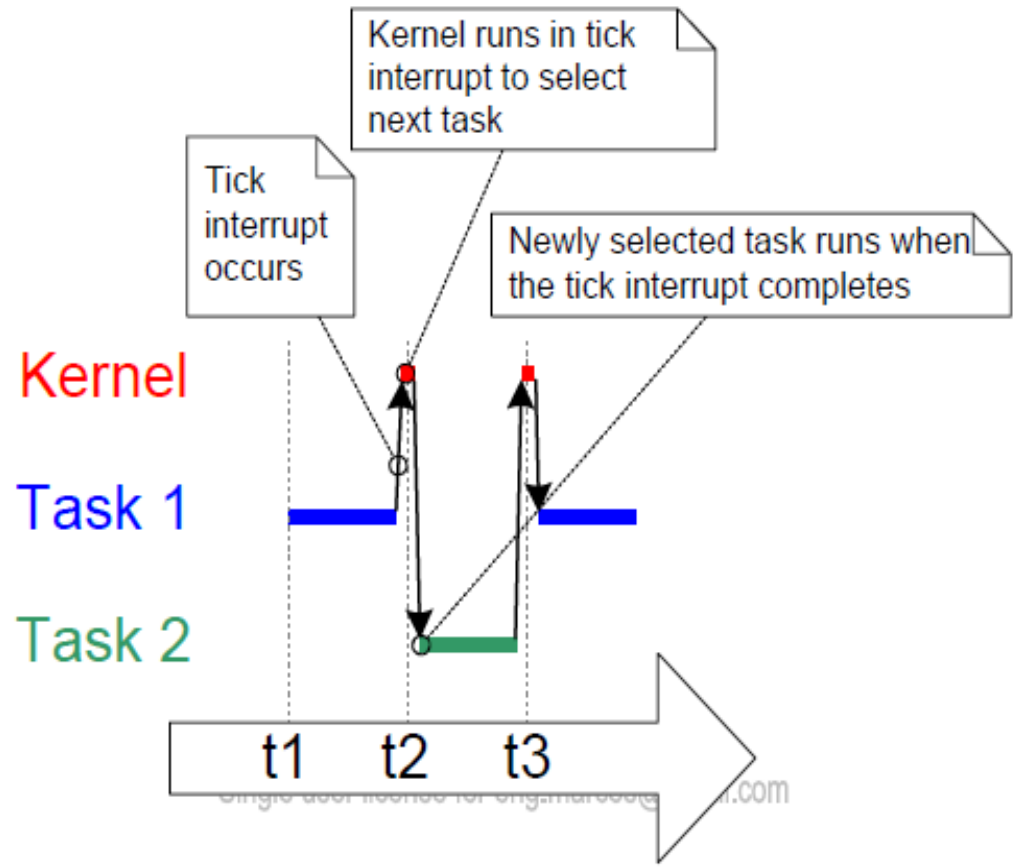

At time t1, Task 1 enters the Running state and executes until time t2
At time t2 Task 2 enters the Running state and executes until time t3 - at which point Task1 re-enters the Running state
Task 1
Task 2
t1
t2
t3
Time

Prioridades de Tasks

- Configuração através do parâmetro:
  - `uxPriority em xTaskCreate()`
- Pode ser modificado através da função
  - `vTaskPrioritySet()`

Prioridades de Tasks

- Tick Interrupt
  - Ocorre periodicamente
  - Análise da próxima *task* em execução a partir da prioridade da *task*



Prioridades de Tasks

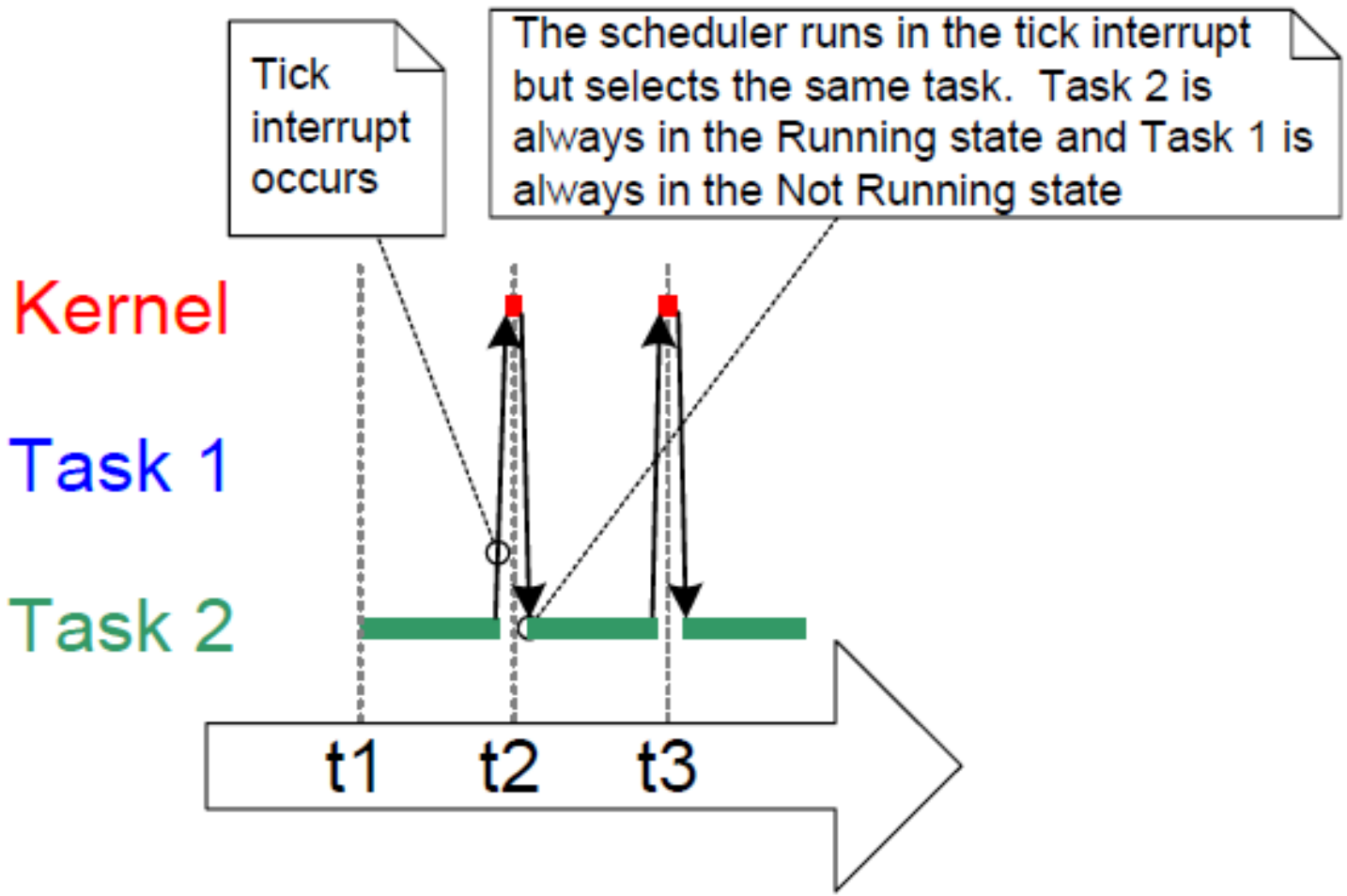
- Diretivas Importantes
  - **`configTICK_RATE_HZ`**: O frequencia do tick speed de interrupção. Utilizado para especificar tamanho do time slice dedicado à execução das tasks. Configurado em `FreeRTOSConfig.h`.
  - **`configMAX_PRIORITIES`**: Número máximo de Prioridades suportadas pelo FreeRTOS. Configurado através do arquivo `FreeRTOSConfig.h`

Prioridades de Tasks

- Prática 3:
  - Implementar a criação de duas *tasks* com prioridades diferentes que imprimam na tela "*Task 1 Running*" e "*Task 2 Running*"
    - TASK 1 com prioridade 1
    - TASK 2 com prioridade 2
  - Observar os comportamento das *tasks* e comentar no código suas conclusões.
  - PS.: As *tasks* devem conter a mesma implementação utilizada no exercício da prática 2

Prioridades de Tasks

- Processo de *Starvation* ou Inanição



Estados ***NOT RUNNING***

- Necessidade de gerenciamento das tarefas com prioridades diferentes.

- *Event-Driven Tasks:* Tarefas que são executadas apenas depois da ocorrência de algum evento e não estão aptas a entrar em modo RUNNING antes da ocorrência do evento.

-

Estados *NOT RUNNING*

- Blocked State
  - Neste estado, estão as tasks que estão a espera de um evento
  - As tasks podem entrar neste estágio e esperar por dois tipos de eventos:
    - Eventos Temporais: Pode ser um tempo específico.
      - Ex.: Delay, Tempo absoluto, …
    - Eventos Síncronos: Gerados a partir de outras tasks
      - Ex: Semáforos, Mutexes, Filas, ...
    - Eventos Mistos: Combinação dos dois eventos anteriores

Estados *NOT RUNNING*

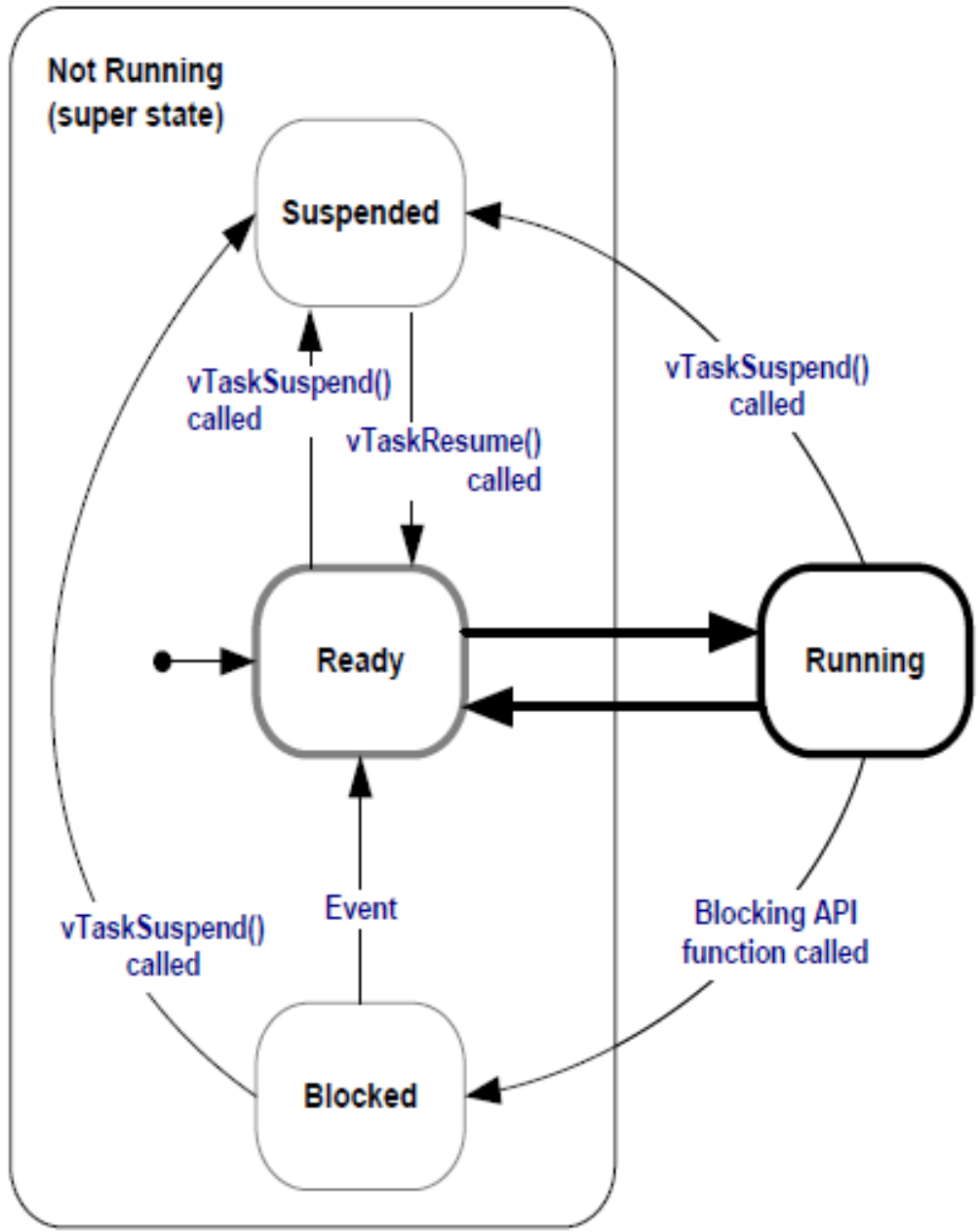
- Suspend State
  - Tasks que não estão aptas a serem gerenciadas pelo escalonador
  - Tasks podem ser colocadas em modo suspenso através da chamada `vTaskSuspend()`
  - Podem retornar através da função `vTaskResume()` ou `xTaskResumeFromISR()`

  - PS.: Poucas Aplicações utilizam este estado

Estados *NOT RUNNING*

- The Ready Stage
  - Tarefas que não estão em *Blocked State* ou *Suspend State*.
  - São tarefas que estão aptas a serem executadas, mas não se encontram em estado RUNNING

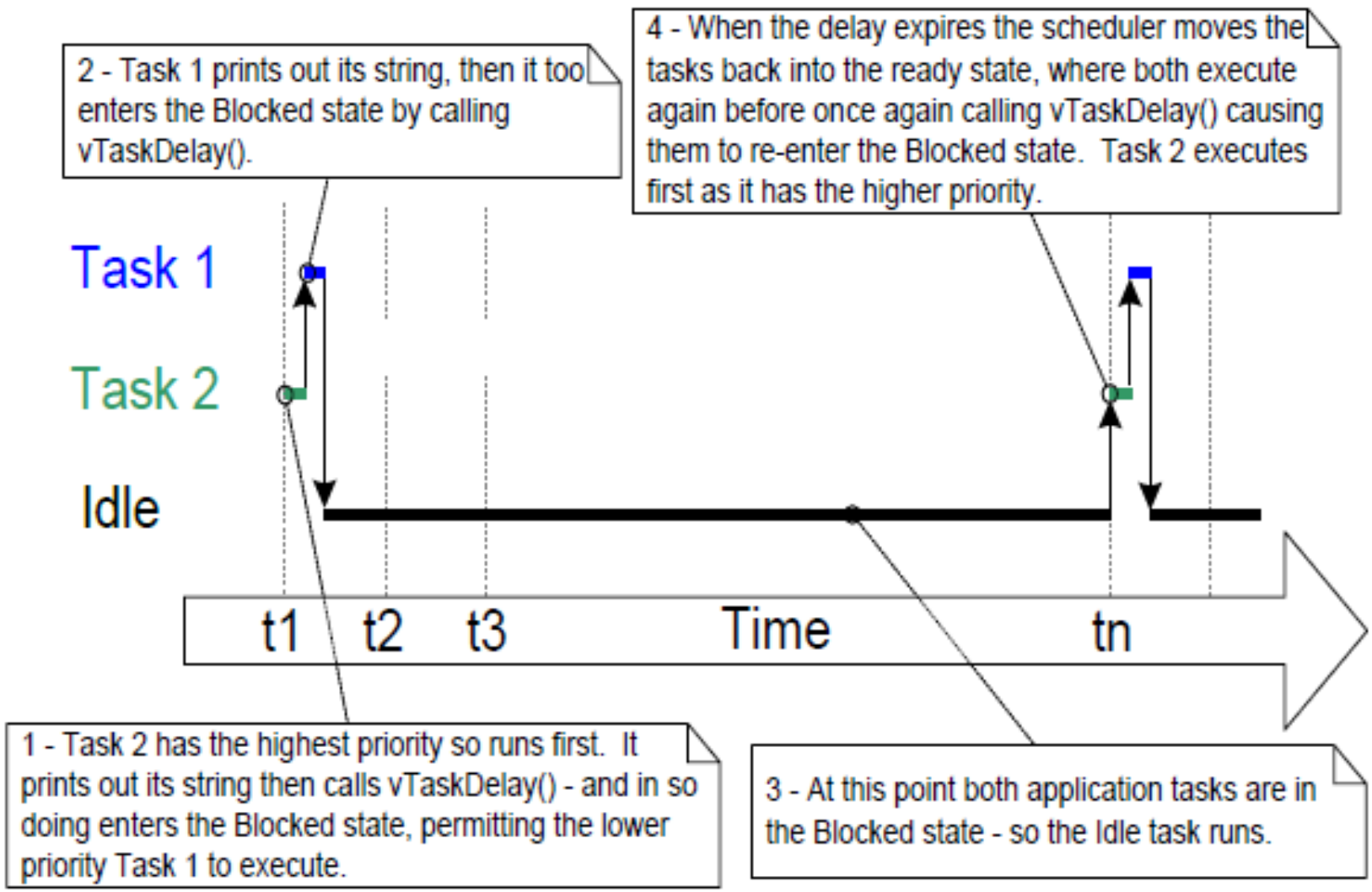
Máquina de Estados

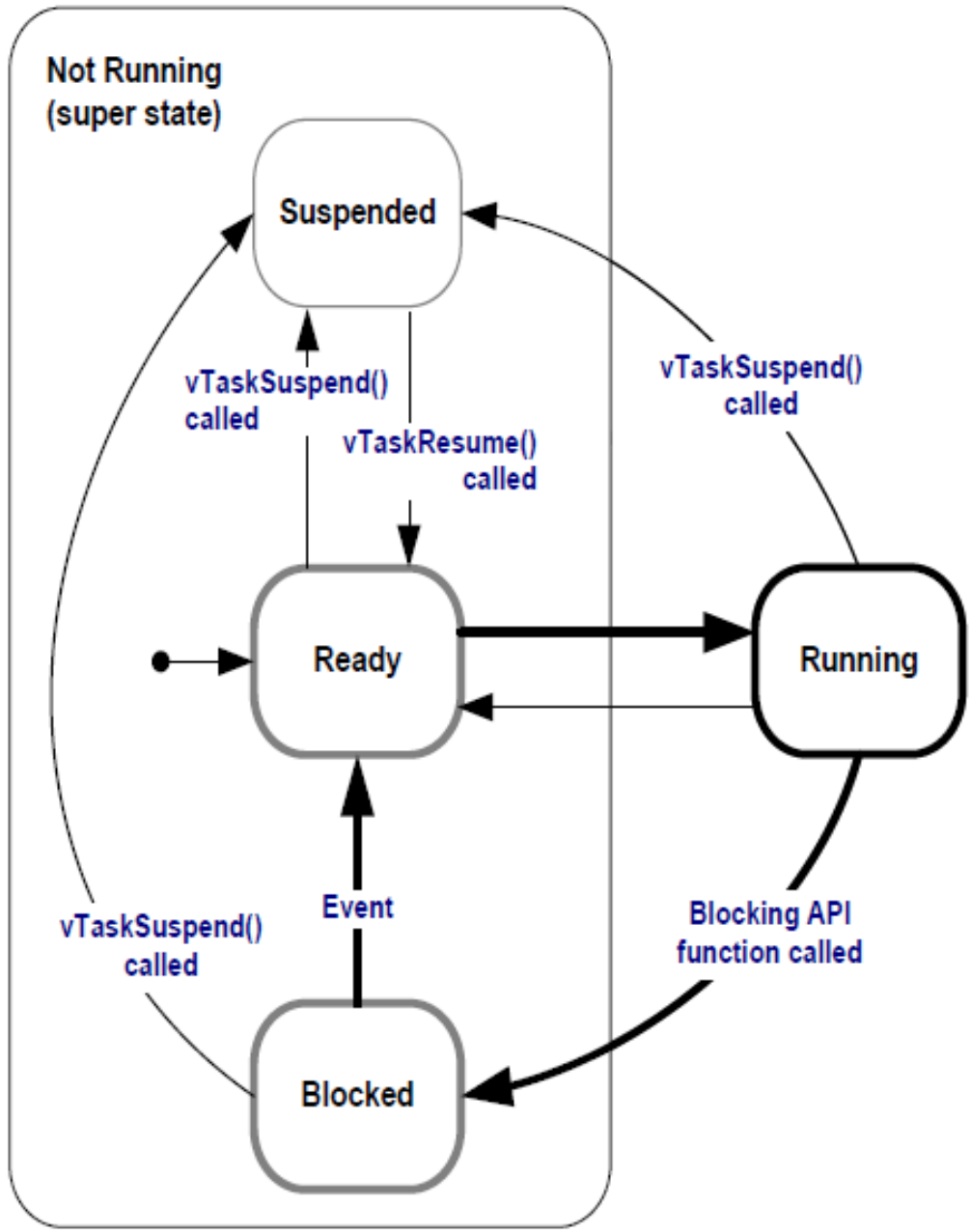


Prática 4

- Utilizar o estado Blocked para produzir um delay e ajustar o exemplo da prática 3 através da função `vTaskDelay()`.

Idle Task


2 - Task 1 prints out its string, then it too enters the Blocked state by calling vTaskDelay().
4 - When the delay expires the scheduler moves the tasks back into the ready state, where both execute again before once again calling vTaskDelay() causing them to re-enter the Blocked state. Task 2 executes first as it has the higher priority.
Task 1
Task 2
Idle
t1
t2
t3
Time
tn
1 - Task 2 has the highest priority so runs first. It prints out its string then calls vTaskDelay() - and in so doing enters the Blocked state, permitting the lower priority Task 1 to execute.
3 - At this point both application tasks are in the Blocked state - so the Idle task runs.


Not Running
(super state)
Suspended
vTaskSuspend()
called
vTaskResume()
called
vTaskSuspend()
called
Ready
Running
Event
vTaskSuspend()
called
Blocking API
function called
Blocked

Tarefas Exatamente Períodicas

- Problemas na utilização do vTaskDelay.
  - O tempo para a próxima execução só é computado a partir do momento da chamada da função, não sendo considerado o tempo computacional da lógica da função.
- vTaskDelayUntil()
  - Utilizada para fixar a periodicidade de uma task levando em consideração seu tempo computacional.
- xTaskGetTickCount()
  - Número do *Tick* Atual do Sistema

Prática 5

- Converter a implementação da prática 4 utilizando a função vTaskDelayUntil.

Prática 6

- Utilizar os conceitos estudados e implementar duas tarefas que devem ser executadas em ***modo contínuo*** imprimindo na tela as expressões:
  - Tarefa Contínua 1 em Execução.
  - Tarefa Contínua 2 em Execução.
- Uma terceira tarefa deve ser criada e funcionar de modo exatamente periódico com tempo de 500ms e imprimir na tela a expressão:
  - Tarefa Períodica 3 em Execução.
- PS.: Utilize a(s) função(ões) de delay mais apropriada(s)

T569 –SISTEMAS DE TEMPO REAL