APRESENTAÇÃO DO FreeRTOS

SISTEMAS DE TEMPO REAL 2009

Luís Filipe Terra Ferreira, luis.terra@ua.pt

Tiago Costa Gonçalves , tiagogoncalves@ua.pt

FreeRTOS

Esta apresentação tem como objectivo a caracterização, descrição sintética e exemplificação do sistema operativo de tempo real freeRTOS.

Vamos a isso!

- O que é?
- Organização
- Como funciona?
- Comparativo

CARACTERIZAÇÃO

Caracterização

FreeRTOS é destinado a sistemas embutidos com requisitos de tempo real mas com poucos recursos computacionais.

É distribuído ao longo de 3 ficheiros e conta com aproximadamente 2000 linhas entre código C e ASM.

Actualmente está na versão 5 sendo disponibilizado sobre licença modificada GPL3 suportando assim aplicações proprietárias.

- Microkernel de tempo real
- Open source
- Royalty free
- Aplicação comercial

Mais características!

- É um kernel que ocupa aproximadamente 4KB na ROM
- Suportado em pequenos µC/µP de 8bits a "full feature" 32bits
- Incluído em várias placas de desenvolvimento
- Vasta disponibilidade de exemplos no site oficial.

ESTRUTURA E FUNCIONALIDADES

Descrição

- Suporta tarefas e co-rotinas
- Pode ser criado qualquer número de tarefas
- Tarefas podem ser criadas dinamicamente
- Pode ser atribuído qualquer nível de prioridade
- Existência de Mutexs simples e recursivos
- Disponibilidade de semáforos
- Suporte de filas de mensagens

Escalonador

- Preemptivo: Corre sempre a tarefa mais prioritária. Tarefas de igual prioridade partilham o tempo de CPU por "Round Robin".
- Cooperativo: Só muda de tarefa em locais indicados (quando a tarefa bloqueia ou quando o indica explicitamente através de taskYIELD()

Tick

- É implementado através de um timer
- Interrompe qualquer tarefa
- As tarefas têm uma unidade de execução de um tick

Estados de uma tarefa

- Um pedaço de função ou código que é executado pelo processador é denominada de tarefa
- Uma tarefa pode estar em vários estados

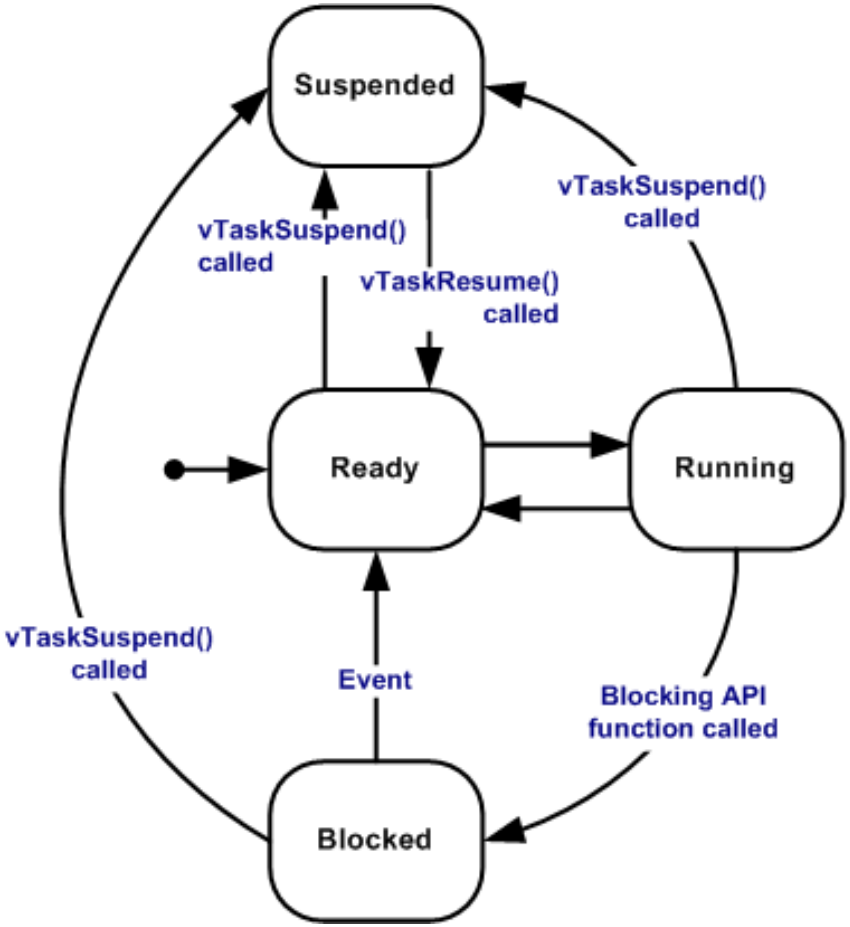
Estados de uma tarefa..

Em execução
Em execução, tem o controlo do processador

Pronta
Pode executar (não se encontra bloqueada nem suspensa) mas não está em execução

Bloqueada
- Encontra-se neste estado se estiver à espera de um evento temporal ou externo para ser executada
- Bloqueio temporal – a tarefa chama uma função de espera
- Evento – espera num semáforo ou numa fila
- Presença de um "timeout"

Suspensa
- A entrada e saída deste estado é feito apenas através de comandos específicos
- Não é possível definir nenhum "timeout"
- Não podem ser escalonadas



Tarefas

- Simples
- Sem restrições no uso
- Suportam total preempção
- Totalmente prioritarizavel
- Cada tarefa mantém a sua própria stack que resulta numa maior utilização de RAM
- Reentrância tem de ser cuidadosamente considerada se for usada preempção

Co-rotinas


Partilham a stack entre sí reduzindo o uso de RAM


Operação cooperativa faz a reentrância menos 1 prob

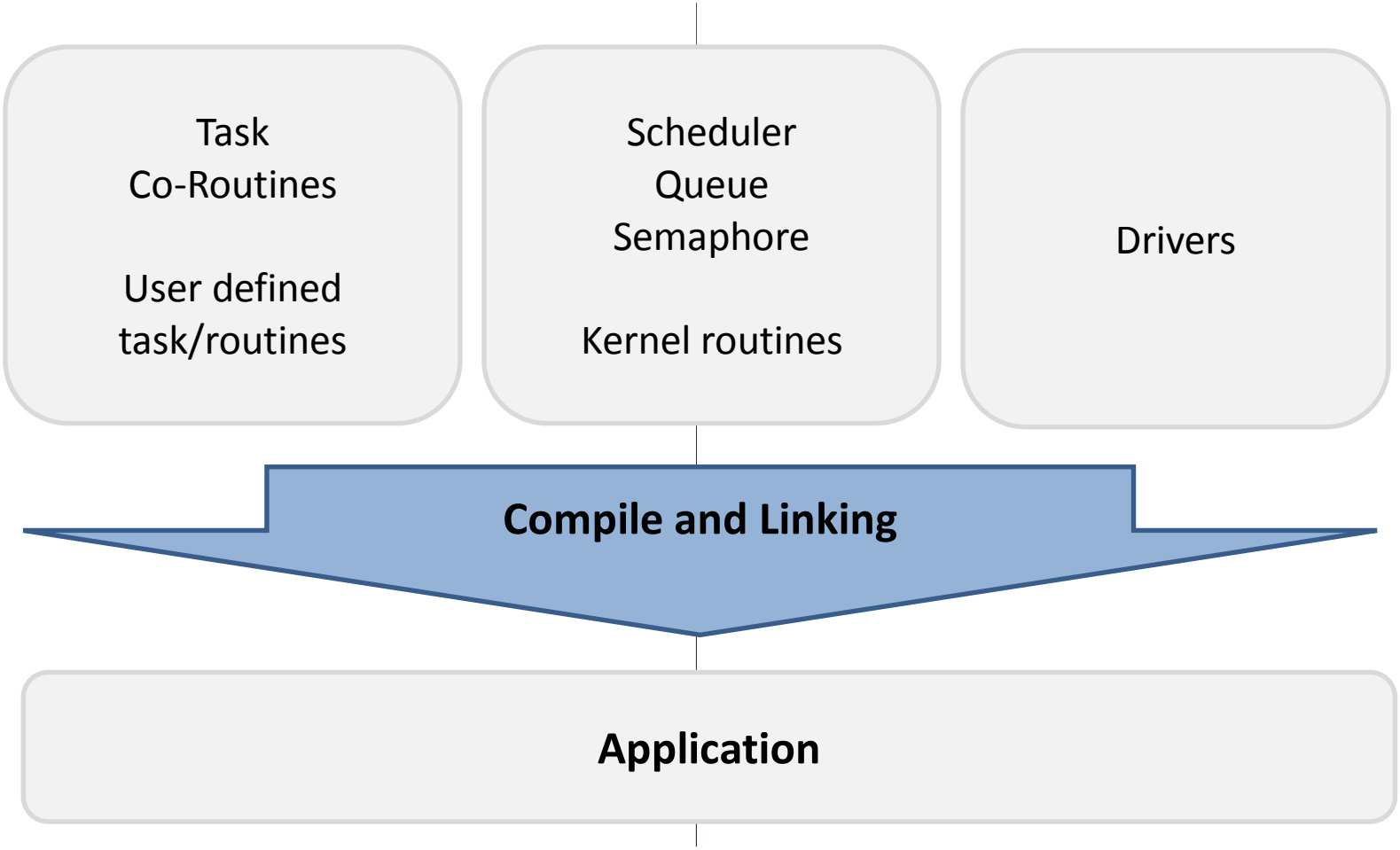

Muito portáveis entre arquitecturas


Totalmente prioritarizavel relativamente a outras co-rotina, mas podem ser sempre interrompidas por tarefas se ambas estiverem misturadas

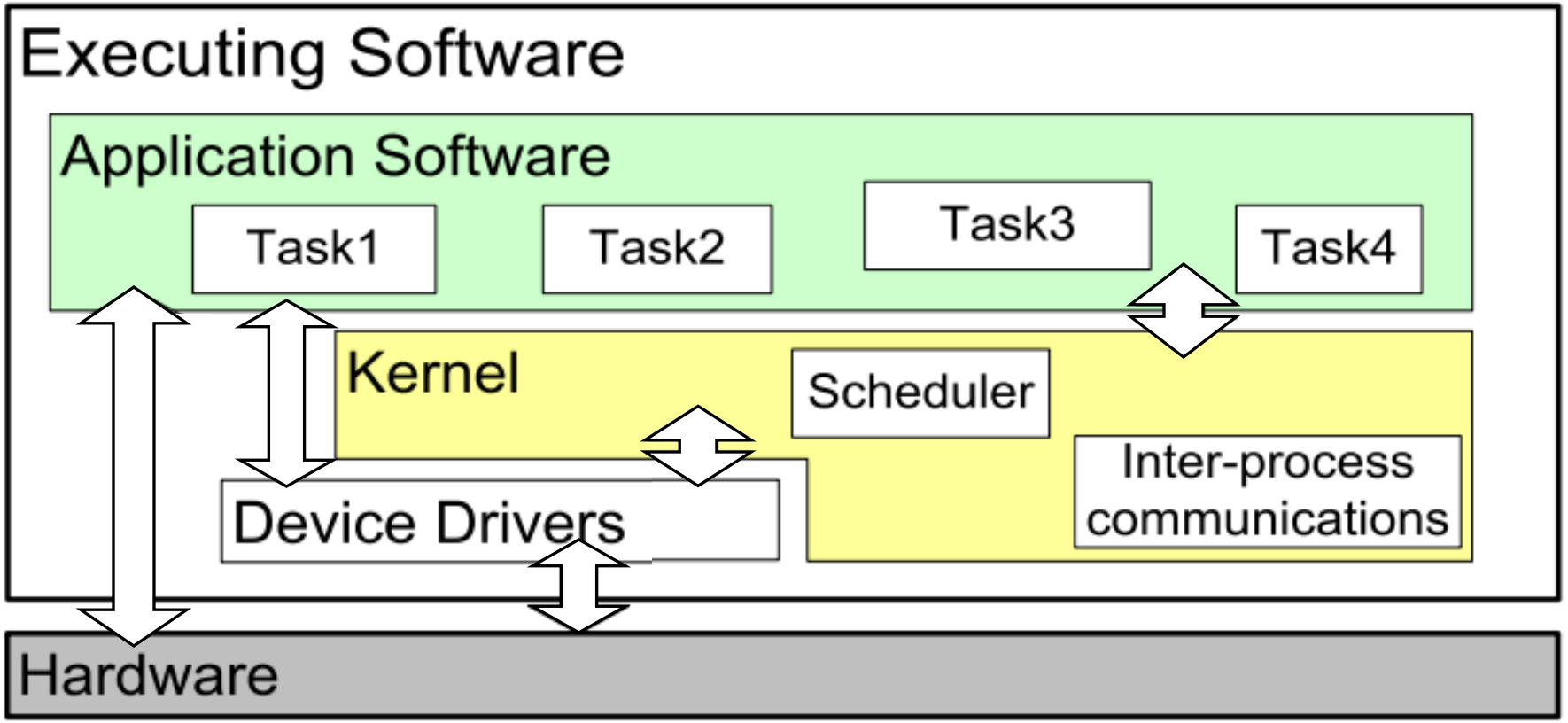

Restrições onde as chamas às API podem ser feitas


Operação cooperativa apenas entre co-rotinas

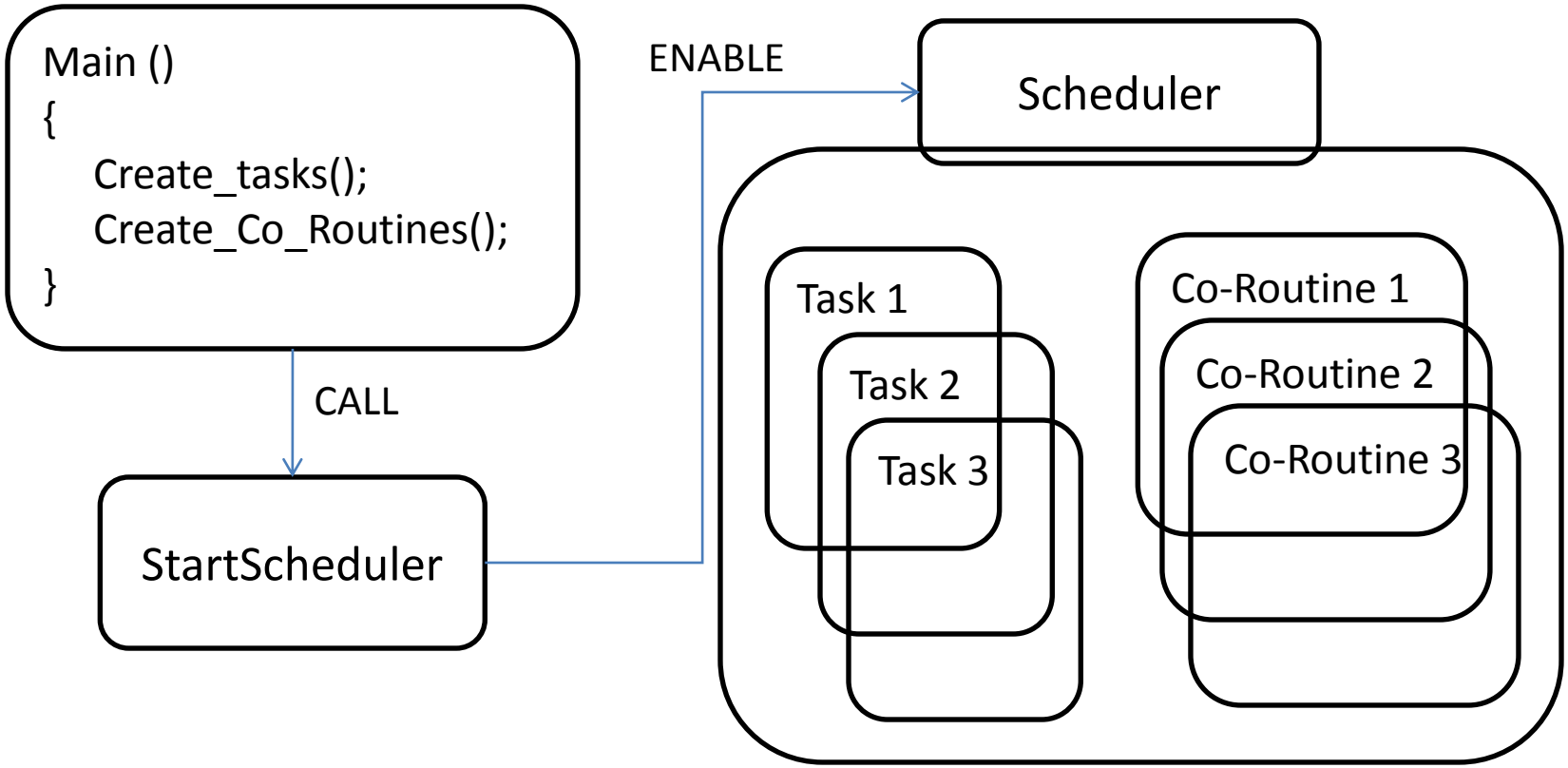
Um sistema baseado em freeRTOS



Interligação entre o hardware e software

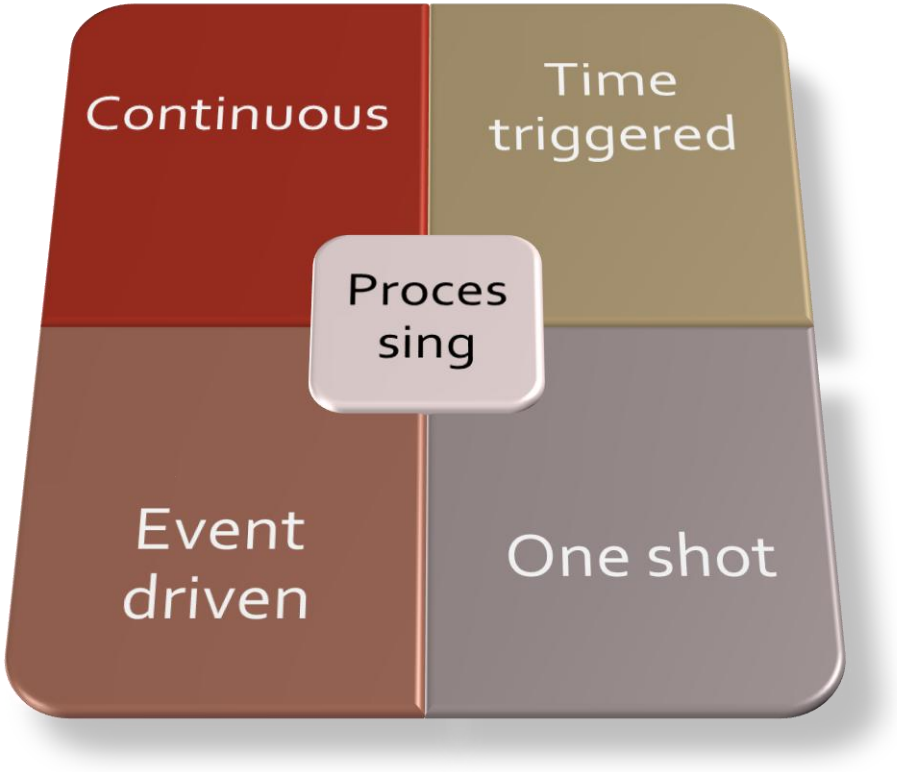


Sequência de funcionamento



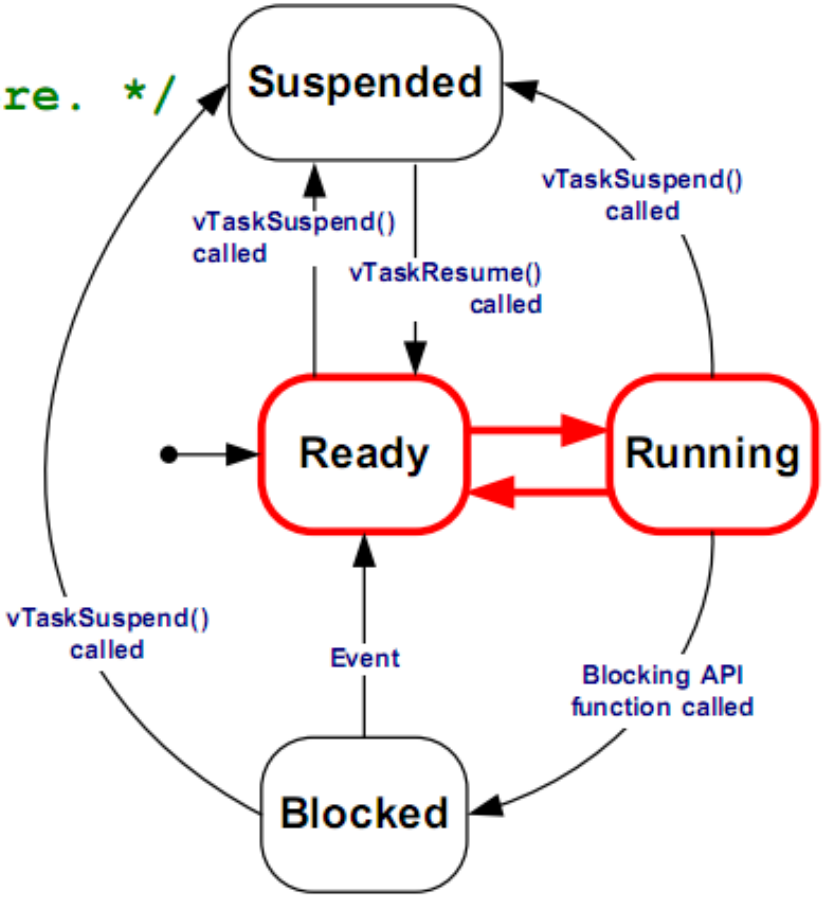
Using tasks to implement

- ✓ Continuous processing
- ✓ Time triggered
- ✓ Event driven processing
- ✓ "One shot" processing



Processamento continuo

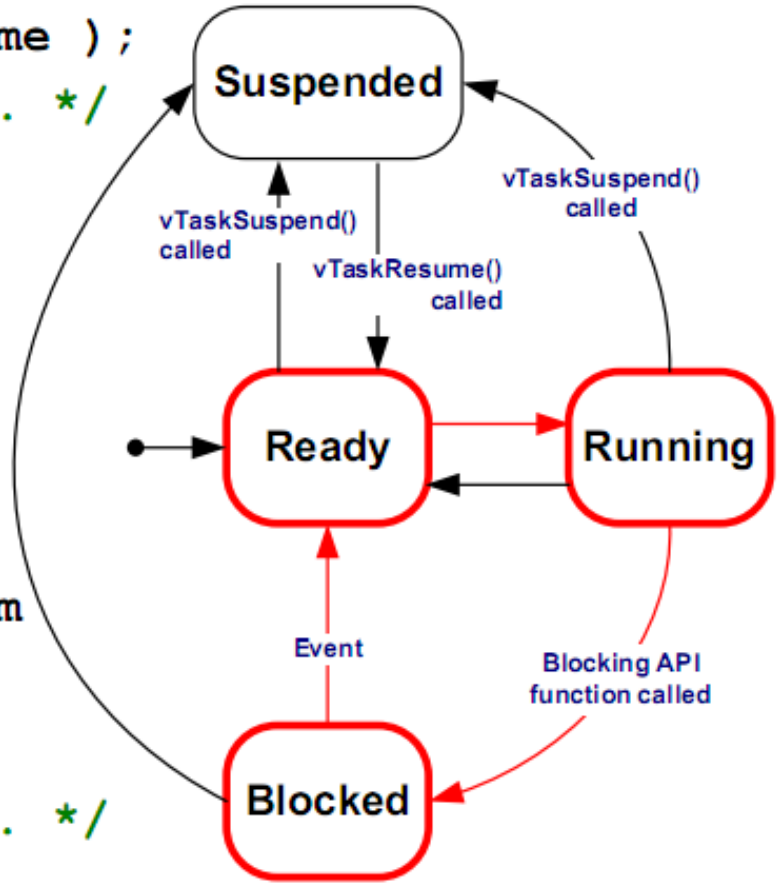
```
void aTask( void * pvParameters )
{
  for( ;; )
  {
    /* Task processing goes here. */
  }

  vTaskDelete( NULL );
}
```



Time Triggered

```
void aFixedFrequencyPeriodicTask( void * pvParameters)
{
   for( ;; )
   {
      vTaskDelayUntil( aRelativeTime );
      /* Task processing goes here. */
   }
}
```

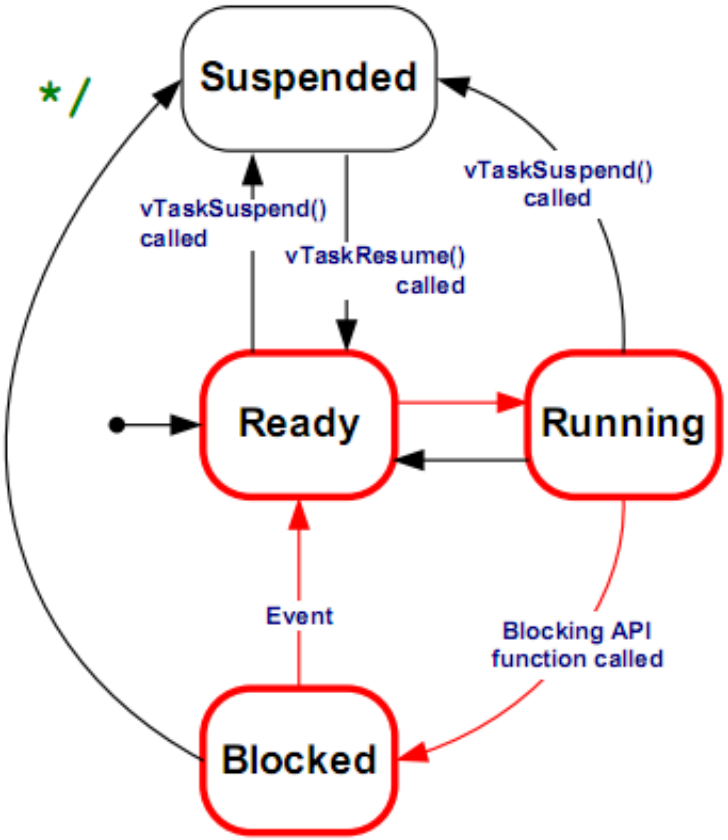
Time is specified in 'ticks'




```
Void aPeriodicTask( void *pvParam
{
   for( ;; )
   {
      vTaskDelay( aTime );
      /* Task processing goes here. */
   }
}
```

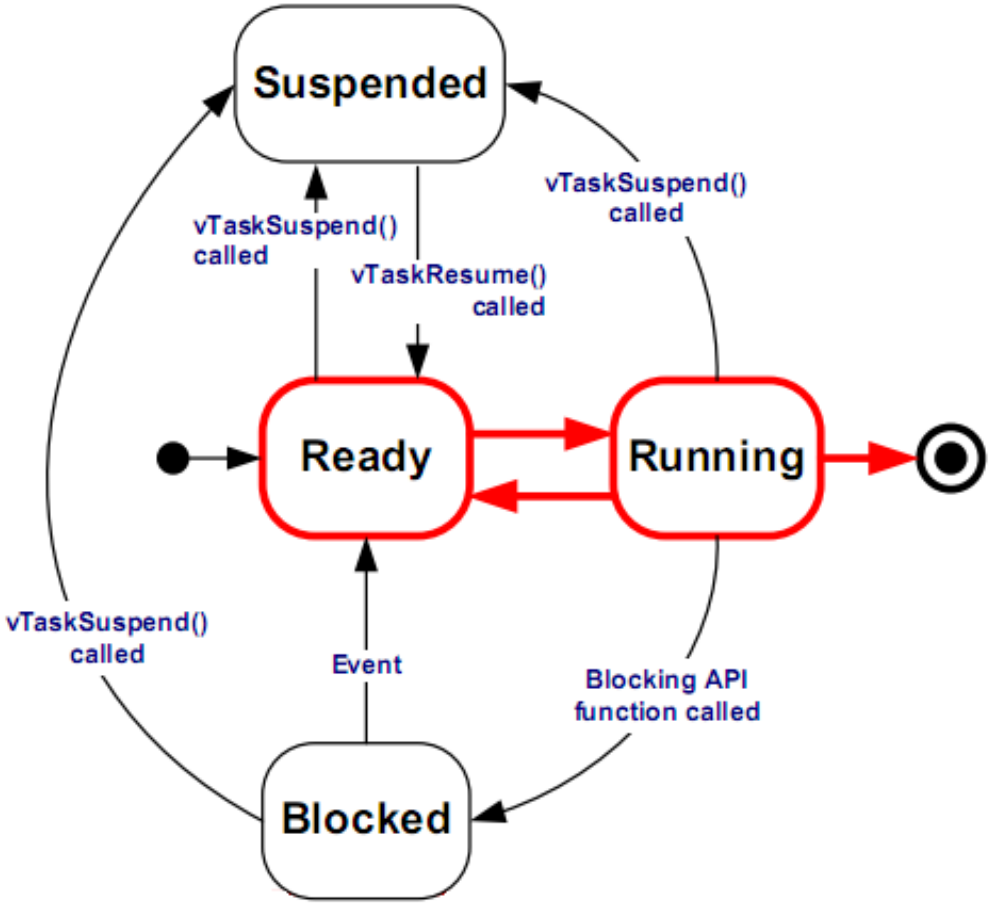
Event Driven Tasks

```
void aPeriodicTask( void * pvParameters)
{
   for( ;; )
   {
      xSemaphoreTake( aSemaphore,
                      aTimeOut );

      /* Task processing goes here. */
   }
}
```



One Shot Tasks

```
void aOneShotTask( void * pvParameters )
{
   /* Task processing goes here. */
   vTaskDelete( NULL );
}
```



COMPARATIVO

freeRTOS vs eCOS

freeRTOS

- Scheduler: - cooperativo
- Número ilimitado de tarefas
- Com muitas tarefas fica cada vez menos determinístico
- Vasto suporte

eCOS

- Bitmap e MQL, + rápido
- Número limitado de tarefas
- É mais determinístico
- Suporte a hardware mais "potente", mínimo de 2MB de RAM

Bibliografia:

- **FreeRTOS Reference Manual - API Functions and Configuration Options**
- www.freertos.org
- FreeRTOS.org on Microchip's, University of Microchip, Masters 2007
- Wikipedia.org
- Entre muitos outros..

Dúvidas e comentários