Insper

Computação Embarcada - RTOS

Rafael Corsi - rafael.corsi@insper.edu.br

Março - 2018

Introdução

Real Time Operation System (RTOS) é uma classe de sistemas operacionais desenvolvida para aplicações que demandam a execução de múltiplas tarefas no mesmo processador (multitarefas) com uma criticidade no tempo.

1

- Defina tarefas o que é uma tarefa sistema operacional.

RTOS diferenciam-se de outros sistemas operacionais (Linux, Windows, . . .) por ser implementado de tal forma que é possível prever a ordem de execução de cada tarefa e quando a mesma será executada (preemptividade).

Existem duas maneiras de se classificar um RTOS: Soft e Hard a diferença entre ambos se da pela maneira com a não execução de uma tarefa no tempo correta e tratada. No caso do soft RTOS a não execução de uma tarefa no tempo certo não causa um dano irreversível para o sistema em que ela está atuando, no caso do Hard RTOS, a execução incorreta de uma tarefa causa um dano irreversível ao sistema.

Um exemplo claro da utilização do Hard RTOS é a aterrizagem da sonda Curiosity no planeta Marte, caso um dos módulos do controle da descida não funcionasse corretamente devido ao não atendimento de uma tarefa, a sonda se chocaria com o planeta fincando inoperante. Nesse caso, não existira uma segunda vez.

! https://www.youtube.com/watch?v=gwinFP8_qIM

2

De exemplos reais da utilização de RTOS HARD e SOFT.

3

Liste exemplos de RTOS opensource e de outros proprietários.

RTOS vs baremetal

Baremetal é a codificação de um projeto embarcado sem a utilização de um sistema operacional. O código é implementado a través de interrupções e polling de dados do sistema. Já com um RTOS o programa não possui acesso direto aos recursos do uC, esse acesso se da via uma camada de software implementada pelo sistema operacional.

Diferente sistemas tais como Linux, Windows, MAC, . . . sistemas operacionais de tempo real são projetados para serem determinísticos, ou seja, a ordem de execução das tarefas pode ser determinada a partir do estado anterior do programa. Isso não é verdade em sistemas operacionais "convencionais", essa variação se da em grande parte pelo escalonador do sistema operacional que é projetado com finalidade distintas. Em um sistema multitarefas o processamento é compartilhado entre as diversas tarefas que estão em execução, sendo o escalonador responsável por esse chaveamento. Outra grande diferença entre um RTOS e um OS está na maneira como as prioridades de cada tarefa são respeitas, em um RTOS, a prioridade define realmente o quanto de recurso de processamento que cada tarefa irá utilizar. Uma tarefa de uma determinada prioridade em um RTOS irá bloquear o escalonamento até que uma das duas opções sejam satisfeitas :

- outra tarefa de mais alta prioridade requisita o processador
- a tarefa termina de realizar o processamento e entra em estado de dormência.

Outra grande diferença em relação um RTOS é a latência de execução de interrupção, essa latência é mínima no caso de um RTOS e não tão crítica em um OS convencional.

4

Por que uma interrupção em um RTOS tem que ser resolvida o quanto antes e em um OS tipo windows não ?

RTOS conceitos básicos

Quando executado em um RTOS, as tarefas são como pequenos trechos de códigos executados no microcontrolador. Cada tarefa pode ter uma função específica

como: ler dado A/D, calcular malha de controle, atualizar interface do usuário. As tarefas são implementadas como funções que executam infinitamente.

A tarefa a seguir é responsável por piscar um LED, no exemplo 22-RTOS. Repare que essa tarefa executa infinitamente (while(1)). A função vTaskDelay() diferente da utilizada anteriormente durante o curso delay_ms() não é um delay executado por software mas sim um delay gerenciado pelo escalonado do RTOS. Quando a função atinge esse ponto, o sistema operacional libera a utilização do processador para outra tarefa, e só irá retornar a executar-la novamente quando o tempo definido na função (1000) for atingido.

```
static void task_led ( void * pvParameters )
{
  while (1) {
    LED_Toggle ( LED0 ) ;
    vTaskDelay ( 1000 ) ;
  }
}
```

Prioridade

Para toda tarefa criada em um RTOS é necessário a definição de sua prioridade, a prioridade irá definir qual tarefa ira executar quando mais de uma tarefa desejar utilizar o processamento.

A maioria do RTOS suportam a implementação de mais de uma tarefa com a mesma prioridade, nesse caso, o escalonador irá fazer um timming slice e permitir que cada tarefa de mesma prioridade execute um pouco.

Uma tarefa no RTOS pode estar em 4 estados distintos: Blocked, Ready, Suspended e Running, conforme Fig. 1.

Texto extraído de : http://www.freertos.org/RTOS-task-states.html

- Running : When a task is actually executing it is said to be in the Running state. It is currently utilising the processor. If the processor on which the RTOS is running only has a single core then there can only be one task in the Running state at any given time.
- Ready : Ready tasks are those that are able to execute (they are not in the Blocked or Suspended state) but are not currently executing because a different task of equal or higher priority is already in the Running state.
- Blocked : A task is said to be in the Blocked state if it is currently waiting for either a temporal or external event. For example, if a task calls vTaskDelay() it will block (be placed into the Blocked state) until the delay period has expired - a temporal event. Tasks can also block to wait for queue, semaphore, event group, notification or semaphore event. Tasks in the Blocked state normally have a 'timeout' period, after which the task

will be timeout, and be unblocked, even if the event the task was waiting for has not occurred.

Tasks in the Blocked state do not use any processing time and cannot be selected to enter the Running state.

- Suspended : Like tasks that are in the Blocked state, tasks in the Suspended state cannot be selected to enter the Running state, but tasks in the Suspended state do not have a time out. Instead, tasks only enter or exit the Suspended state when explicitly commanded to do so through the vTaskSuspend() and xTaskResume() API calls respectively.

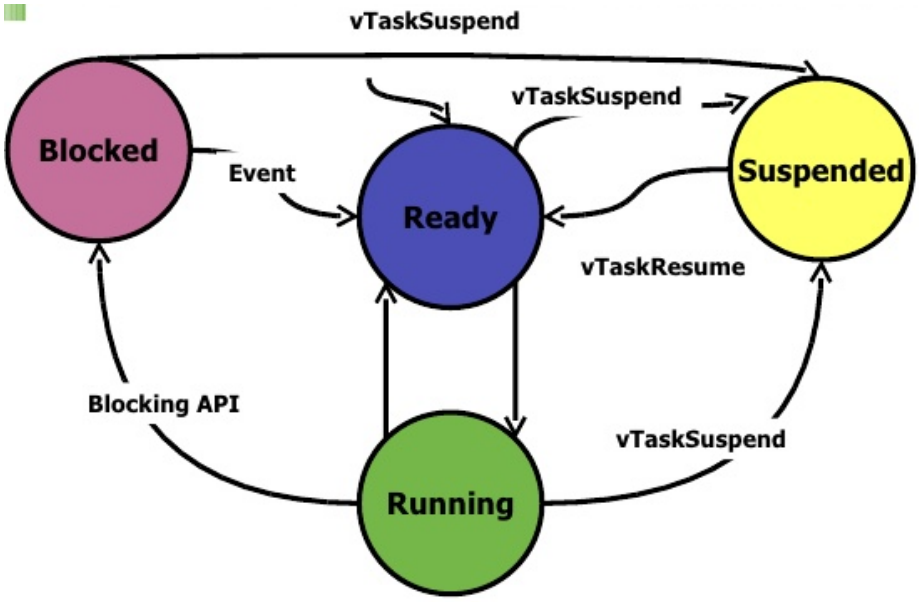

Figure 1: Estados

5

Qual a função do estado blocked ?

Escalonador

O escalonador é a parte do RTOS responsável por decidir qual tarefa irá executar e por quanto tempo ela terá acesso ao processador. O escalonador trabalha com uma base de tempo chamado de TICK a cada novo TICK o sistema decide quais tarefas estarão ativas ou não.

A base de tempo do TICK é normalmente gerada por um dos TCs do uC, e não deve ser alterada a nível de programa.

6

Pesquise como o escalonador do FreeRtos funciona e escreva um pequeno texto explicando.