RTOS là gì – Nó hoạt động như thế nào?

Trung Thành July 2, 2015 Điện tử, Hệ Thống, Hệ thống nhúng, Lập Trình No Comments

Khi chúng ta nghe thấy ai đó nhắc tới hệ điều hành (Operating System – OS) thì chúng ta sẽ nghĩ ngay tới chiếc máy tính chạy Windows XP, Windows 7, Windows 8 hay chạy một distro Linux và Android hoặc iOS cho điện thoại. Chúng ta biết đến hệ điều hành chủ yếu là dành cho máy tính. Tuy nhiên trong thực tế, có rất nhiều thiết bị điện tử có chạy một dạng hệ điều hành rút gọn bên trong nó. Bên cạnh đó cũng có rất nhiều loại hệ điều hành được thiết kế cho vi điều khiển. Trong số chúng có một số thuộc họ Hệ điều hành thời gian thực (Real Time Operating System – RTOS), cụm từ ***thời gian thực*** ở đây chỉ ra rằng thời gian phản hồi của hệ thống là rất nhanh.

Bạn có thể hiểu như sau:

- ***Hệ điều hành thông thường*** hiện diện trong máy tính của bạn, khi bạn mở ứng dụng lên thì sẽ có nhiều lúc bạn phải chờ khá lâu. Việc chờ như vậy hầu như không ảnh hưởng

Search the site

Search

ĐƯỢC XEM NHIỀU

Fix lỗi ST-Link – No target connected với STM32CubeF1 v1.4
July 8, 2016 • 2 Comments

[Dev-Fun] Một số comment "bá đạo" của các coder, programmer
July 28, 2016 • No Comment

BÀI LIÊN QUAN

Một vài điều cần nhớ về Linux

gì nhiều lắm, và bạn có thể pha cho mình 1 tách cafe trong khi chờ ứng dụng khởi chạy. Đôi khi ứng dụng lỗi thì chúng ta chỉ cần đóng process rồi chạy lại, gần như chẳng ảnh hưởng gì đến ai, có chăng thì khó chịu một chút thôi.

- ***Hệ điều hành thời gian thực*** được thiết kế ra cho các nhiệm vụ đặc biệt. Các ứng dụng cần được thực thi với thời gian thật chính xác, các lỗi phát sinh cần được cô lập và xử lý nhanh chóng. Mọi sự chậm trễ, lỗi phát sinh không lường trước có thể khiến hệ thống bị đổ vỡ.

Vì vậy, RTOS sử dụng trong những ứng dụng yêu cầu thời gian đáp ứng nhanh, chính xác về thời gian. Vi điều khiển có tài nguyên rất giới hạn. Do đó, hệ điều hành này chỉ tập trung vào một số ít các tính năng. Chúng cố gắng tối ưu tối đa số luồng, bộ lập lịch và các tác vụ (task) trên một hệ thống cỡ nhỏ.

Thông thường, RTOS là một phân đoạn hoặc một phần của chương trình, trong đó nó giải quyết việc điều phối các task, lập lịch và phân mức ưu tiên cho task, nắm bắt các thông điệp gửi đi từ task. RTOS khá phức tạp, nói một cách dễ hiểu hơn là nó thực hiện việc xử lý các trạng thái máy (State Machine). Dưới đây là 1 minh hoạ cho bạn dễ hiểu hơn về thứ gọi là trạng thái máy.

```
while(1)
{
 switch(state)
 {
 case 1: //Code for Task 1;
 state= 2;
 case 2: //Code for Task 2;
 state= 3;
 case 3: //Code for Task 3;
 state= 4;
 case 4: //Code for Task 4;
 state=1;
 }
}
```

Bạn có thể thấy trong đoạn mã trên, có sự điều chuyển giữa các mệnh lệnh thực thi một cách liên tục. Nó có thể được phát triển và thiết kế phức tạp hơn nữa. Người lập trình có thể thay đổi và sử dụng các lệnh điều kiện (if-else, switch-case…) để chuyển qua lại giữa các task. Do vậy, các luồng chương trình có thể được xác định rõ ràng.

Permission
April 26, 2015 • No Comment

[CSharp] Tăng hiệu suất Insert dữ liệu khi làm việc với SQLite
November 22, 2015 • No Comment

C/C++ Preprocessor – Constant và Conditional Inclusion
April 22, 2016 • No Comment

Add thêm thư viện 3D trong Altium
March 10, 2015 • 2 Comments

[Trick] Make Borderless Form Moveable
April 3, 2015 • No Comment

Từ khóa virtual trong C++
April 18, 2016 • No Comment

Tự học lập trình ARM – Phần 2: Cài đặt IDE và thiết lập cấu hình
March 4, 2015 • 6 Comments

[C/C++] Include guard trong C và

Nhân (kernel) của hệ điều hành giao cho CPU thực hiện việc xử lý các task theo những khoảng thời gian. Nó cũng liên tục kiểm tra mức ưu tiên của các task, xắp xếp các thông điệp từ task và tiến hành lập lịch.
Trong một hệ thống chạy RTOS, cũng có các tài nguyên dùng chung cùng với phần được cấp phát riêng cho mỗi task.

Các chức năng cơ bản của một RTOS:

- Bộ lập lịch (Scheduler).
- Các dịch vụ thời gian thực (Realtime Services).
- Đồng bộ và xử lý thông điệp (Synchronization and Messaging).

SCHEDULER

Mỗi task có thể có 3 trạng thái.

- ***Ready to run:*** Là trạng thái mà task đã có đủ các tài nguyên để khởi chạy nhưng chưa chạy. Đây là trạng thái chuẩn bị của task.
- ***Running:*** Là trạng thái mà task đang được thực thi.
- ***Blocked:*** Khi task không có đủ các tài nguyên cần thiết để chạy thì nó sẽ được đưa về trạng thái blocked.

Để lập lịch cho Task, có 3 kỹ thuật được áp dụng:

- ***Co-operative scheduling:*** Mỗi task được thực thi đến khi kết thúc quá trình thì task tiếp theo mới được thực thi.
- ***Round Robin Scheduling:*** Mỗi task được chia cho một khe thời gian cố định, nếu trong khoảng thời gian được chia đó mà task chưa thực hiện xong thì sẽ bị tạm dừng, chờ đến lượt tiếp theo để thực hiện tiếp công việc sau khi hệ thống xử lý hết một lượt các task.
- ***Preemptive Scheduling:*** Phương pháp này ưu tiên phân bổ thời gian cho các task có mức ưu tiên cao hơn. Mỗi task được gán 1 mức ưu tiên duy nhất. Có thể có 256 mức ưu tiên trong hệ thống, và có thể có nhiều task có cùng mức ưu tiên.
  — ***Preemptive:*** Các task có mức ưu tiên cao nhất luôn được kiểm soát bởi CPU, khi phát sinh ISR thì hệ thống sẽ

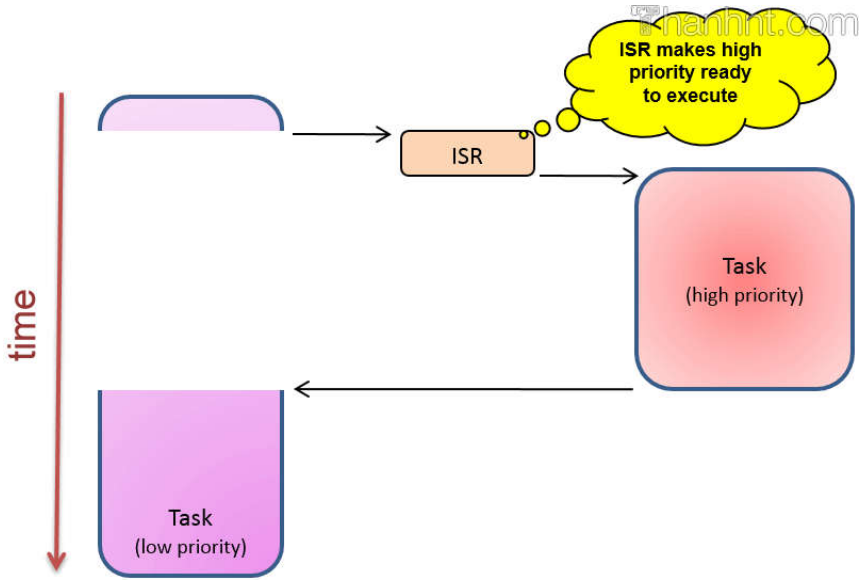
C++
February 22, 2016 • No Comment

BẠN BÈ

Nhým Xù's Blog

tạm dừng task đang thực thi, hoàn thành ISR sau đó hệ thống thực thi task có mức ưu tiên cao nhất tại thời điểm đó. Sau đó hệ thống mới tiến hành nối lại các task đang bị gián đoạn.

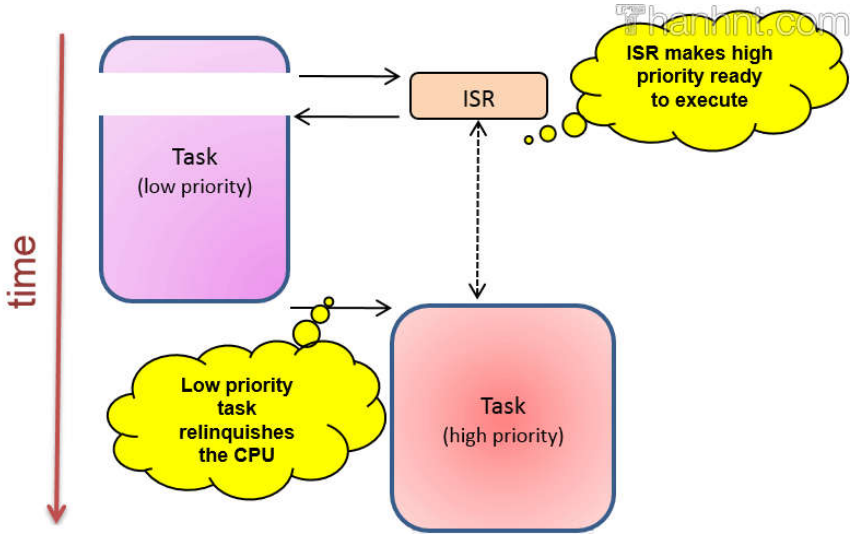
Ở chế độ preemptive, hệ thống có thể đáp ứng các công việc khẩn cấp một cách nhanh chóng. Đa số các hệ thống thực tế đang chạy ở chế độ này.



— ***Non-preemptive:*** Ở chế độ non-preemptive thì các task được chạy cho đến khi nó hoàn tất. Khi phát sinh ISR thì hệ thống sẽ tạm dừng task đang thực thi và hoàn thành ISR , sau khi hoàn thành ISR thì hệ thống sẽ quay lại thực thi nốt phần việc còn lại của task bị gián đoạn. Task có mức ưu tiên cao hơn sẽ giành quyền kiểm soát CPU sau khi task bị gián đoạn thực thi xong.

Ưu điểm của non-preemptive: độ trễ gián đoạn thấp.

Nhược điểm của non-preemptive: do phải chờ task thực thi xong thì task có mức ưu tiên cao mới được thực thi, do đó mức đáp ứng của hệ thống thấp. Vì vậy rất ít hệ thống có sử dụng non-preemptive.



Kernel tiến hành quản lý task ở nhiều giai đoạn. Chúng bao gồm:

- Tạo task.
- Huỷ task.
- Thay đổi mức ưu tiên của task.
- Thay đổi trạng thái của task.

RTOS SERVICES (DỊCH VỤ THỜI GIAN THỰC)

Kernel là trái tim của mỗi hệ điều hành. Nó thực hiện chức năng quản lý và lập lịch các quá trình sử dụng CPU và điều phối tài nguyên. Mỗi task chỉ được thực thi bởi CPU trong một khoảng thời gian, trong các khoảng thời gian còn lại thì task được quản lý bởi các dịch vụ quản lý của hệ điều hành.

Các dịch vụ của RTOS bao gồm:

- Xử lý ngắt (Interrupt handling services).
- Dịch vụ quản lý thời gian (Time services).
- Dịch vụ quản lý thiết bị (Device management services).
- Dịch vụ quản lý bộ nhớ (Memory management services).
- Dịch vụ quản lý các kết nối Vào – Ra (IO services).

MESSAGING (CÁC THÔNG ĐIỆP)

Các thông điệp sử dụng để trao đổi thông tin giữa các hệ thống khác nhau hoặc giữa các task. Các dịch vụ quản lý thông điệp bao gồm:

- Semaphores
- Event flags
- Mailboxes
- Pipes
- Message queues

Semaphores: Dùng để đồng bộ hóa quyền truy cập vào các tài nguyên dùng chung.

Event Flags: Dùng để đồng bộ hóa các hoạt động cần có sự phối hợp của nhiều task.

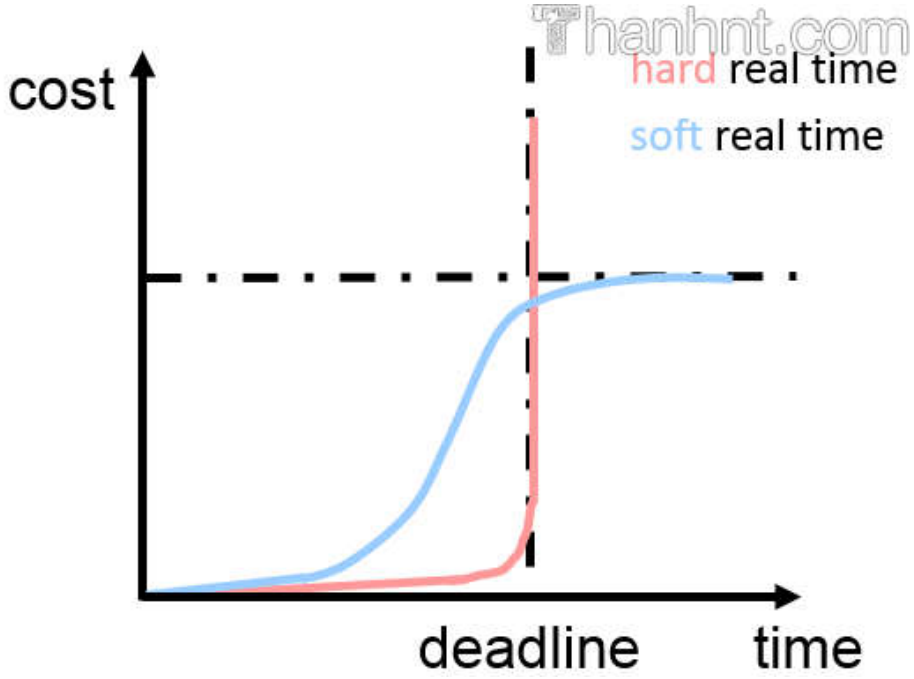
Mailboxes, Pipes, Message queues: Dùng để quản lý các thông điệp gửi đi – đến giữa các task.

CÁC DẠNG THỜI GIAN THỰC

Một hệ thống có thể chia làm 2 loại thời gian thực dựa vào mức độ trễ – được gọi là "*thời hạn lập danh mục*" (*scheduling deadline*) :

- ***Hard – Realtime:*** hệ thống phải tiếp nhận và nắm bắt được *scheduling deadline* của nó tại mỗi và mọi thời điểm. Sự sai sót trong việc tiếp nhận deadline có thể dẫn đến hậu quả nghiêm trọng.
- ***Soft – Realtime:*** *scheduling deadline* có dễ thở hơn chút ít, với *soft-realtime* thì không có gì quá nghiêm trọng xảy ra nếu hệ thống thỉnh thoảng bị trễ. Lỗi về mặt thời gian có thể chỉ đơn giản là dẫn đến hậu quả giảm độ tin cậy của đối tượng đối với hệ thống mà không dẫn đến hậu quả nghiêm trọng nào khác xảy ra.

Có hệ thống bao gồm cả 2 loại này.



CÁC LOẠI RTOS

Hiện tại có vô vàn hệ điều hành thời gian thực, bạn có thể Google với từ khóa RTOS để tìm hiểu nhiều hơn với từng loại.

Tham khảo Voer, Circuitstoday

Related Posts:

1. **Một số khái niệm khi làm việc với RTOS**
2. **RTOS – Lắp thêm cánh cho vi điều khiển**
3. **Kinh nghiệm debug lỗi khi làm lập trình nhúng**
4. **[Hệ thống] Tràn bộ nhớ Stack**
5. **5 Lý Do Bạn Nên Chọn Embedded Software**
6. **Lập trình STM32 – Tạo Project mới với STM32CubeMX**
7. **Tự học lập trình ARM – Phần 2: Cài đặt IDE và thiết lập cấu hình**
8. **Tự học lập trình ARM – Phần 1: Bắt đầu với ARM**
9. **Xử lý lỗi tự thoát Debug của Keil uVision**
10. **Hệ thống nhúng là gì?**

Tags: RTOS

About The Author

Trung Thành

Hiện tại tôi đang sống tại Hà Nội, công việc của tôi chủ yếu liên quan đến điện tử, cụ thể là Embedded Systems, PCB Design. Thỉnh thoảng có thể chuyển sang làm Windows Application Developer hoặc Server Administrator.

Leave a Reply

Be the First to Comment!

Notify of new follow-up comments Email ›

Start the discussion