Embedded Programming

Mathias Balling & Mads Thede

Last updated: June 19, 2024

Contents

1 ARM architecture

Acorn RISC Machine. Reduced Instruction Set Computer, later Advanced RISC Machine. Key feature is that ARM processors have good performance per Watt.

1.1 RISC

RISC is a small set of simple and general instructions. Advantages compared to CISC:

- Instructions take one clock cycle
- Performance is better due to simplified instruction sets
- Less chip space is used due to reduced instruction set
- Can easily be designed as compared to CISC
- Reduced per chip cost, as it uses smaller chips
- Performance of the processor will vary according to the code being executed
- RISC processors require very fast memory systems to feed various instructions

Risc is used in Qualcomm snapdragon which is in most smartphones. In addition to this apple silicon is also based on ARM.

1.2 ARM Cortex-M4 peripherals

SysTick timer is a 24-bit timer that counts down to zero.

Registers: A processor register is one of a small set of data holding places that are part of the computer processor. A register may hold an instruction, a storage address, or any kind of data (such as a bit sequence or individual chracters). Some instructions specify registers as part of the instruction.

Data types: 32-bit words, 16-bit half words, 8-bit bytes.

Interrupt vector table: Contains the addresses of the interrupt service routines (ISR). Using the addresses you can jump to the ISR when an interrupt is triggered.

2 Notes

Static variables

Global variables

Volatile

extern

Build process

2.1 Interrupts

An interrupt is the automatic transfer of software execution in response to a hardware event that is asynchronous with the current software execution.

The hardware event is called a trigger. When this happens, an ISR is called. This pauses the execution of the regular code, until the ISR returns or a higher priority interrupt is triggered.

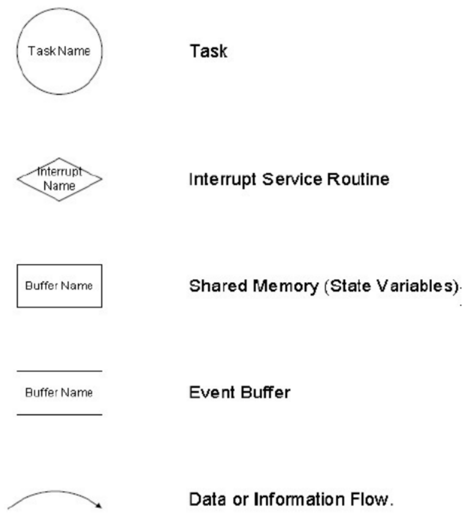
2.2 Task model

A task is an independent thread of execution that can compete with other concurrent tasks for processor execution time. It is the same as a process in a multitasking operating system.

Division of applications into tasks

Criteria for task creation:

- Parallelism: Simultaneous and independent functionality
- Timing: Different timing constraints
- Priority: Divide tasks with different priority
- Structure: Each task handles one well defined problem
- Coupling: Divide problem into loosely coupled tasks.
- Periodicity: A task that must execute with a fixed period is a task by itself.



A shared memory element keeps its value after it has been read. An event from the event buffer gets destroyed after reading and processing.

3 Real Time Operating System

Responds to events within a strictly defined time. The scheduler is deterministic, meaning that the time it takes to switch between tasks is predictable. Priority based scheduling is used to determine which task to run next. Thread=task. Data is processed as it comes in, typically without buffer delays.

Application of RTOS could be airbag deployment, anti-lock braking system, etc.

FreeRTOS

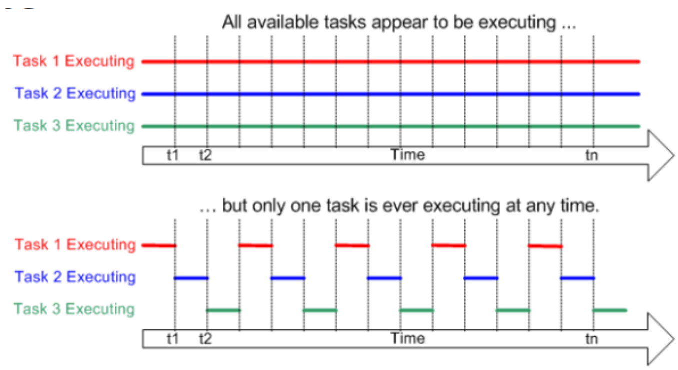
RTOS small enough to run on a microcontroller. Embedded controllers don't warrant full RTOS because of their hardware limitation. FreeROTS provides only:

- Real-time scheduling functionality
- Inter-task communication
- Timing and synchronization primitives

Multi-tasking

Kernel - core of OS.

Multi-tasking in OS is if the OS can run multiple tasks at the same time. This allows for complex applications to be subdivided into small tasks.



Scheuler

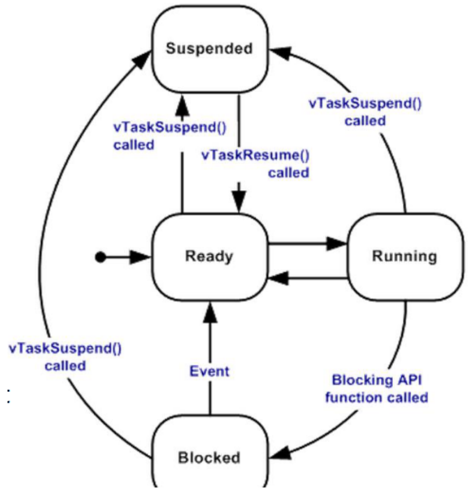
The scheduler is a part of the kernel that decides which task will be executed when. The scheduler can suspend and resume tasks, but tasks can suspend themselves too.

FreeRTOS choice of scheduling policy

Two choices: Preemptive and Cooperative. Pre-emptive always runs the highest priority task that is ready to run. Having multiple tasks with the same priority, they will run using round robin scheduling. Co-operative scheduling is where context switches only occur if a task blocks or explicitly calls task YIELD().

FreeRTOS Task States

There are four states in FreeRTOS: Running, Ready, Blocked, Suspended. Running is the task that is currently executing. Ready is where the task is able to execute but is not currently executing due to another task with equal or higher priority running. Blocked is where the task waits for a temporal or external event. It can block waiting for a queue or semaphore event. Suspended is where the task is not available for scheduling. Tasks only enter or exit suspend by vTaskSuspend() and xTaskResume().



Sempahores

Semaphores are used to synchronize tasks and protect shared resources. A semaphore can be a signaling semaphore or a mutex. Meaning that a semaphore can be used to signal between tasks or to lock a resource. Semaphores can be binary or counting.

Resource Management

There is a potential risk for a conflict in a multitasking system. If one task starts to access a resource but does not complete its access before being transitioned out of the running state. If the task left the resource in an inconsistent state, then access to the same resource by any other task or interrupt could result in data corruption.

Mutual exclusion

Mutual exclusion can be implemented in FreeRTOS by: Disabling interrupts, using a mutex or disabling the scheduler.

Mutex

A mutex is a special type of binary semaphore that is used to control access to a resource. Mutex vs. Semaphore: Mutexes can be released only by the task which took them, while binary semaphores can be released by any task. Mutexes are for protecting resources - binary semaphores are for serialization and signaling.

4 Learning objectives

- Explain the special demands that embedded systems place on software, and how to meet those demands by using the programming language: C.
- Analyze specifications of I/O-devices, and design high performance, hardware-near programs for these
- Evaluate real-time conditions in an embedded system
- Explain principles and algorithms for central parts of operating systems
- Design functions to a real-time operating system

What is embedded programming?

Programming of embedded systems. An embedded system is a computer system - a combination of a computer processor, computer memory, and input/output peripheral devices - that has a dedicated function within a larger mechanical or electrical system.

Why is embedded programming important?

It is important because it is used in many devices that we use in our everyday life. For instance in cars, airplanes, household appliances, and medical equipment.