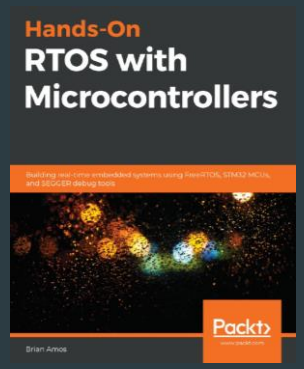
FreeRTOS Simulation on Linux

Hossein Assaran

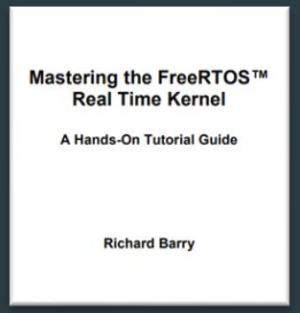
About Me

- B.S. in Electrical Engineering: Shahed University
- M.A. in Philosophy
- Over 10 Years in Embedded System Design and Development
  - AVR, PIC ,ARM9, ARM7, CORTEX-M0, CORTEX-M3, CORTEX-M4, CORTEX-A5, CORTEX-A8
- Startup Owner: Fibook, Khanebazia, Cpuban, Filink, Lernino, …
- Linux
- FreeRTOS
- Kotlin
- GitHub: https://github.com/HosseinAssaran

Resources



Hands-On RTOS With Microcontrollers
Brian Amos



Mastering the FreeRTOS™ Real Time Kernel
Richard Barry

FreeRTOS.org

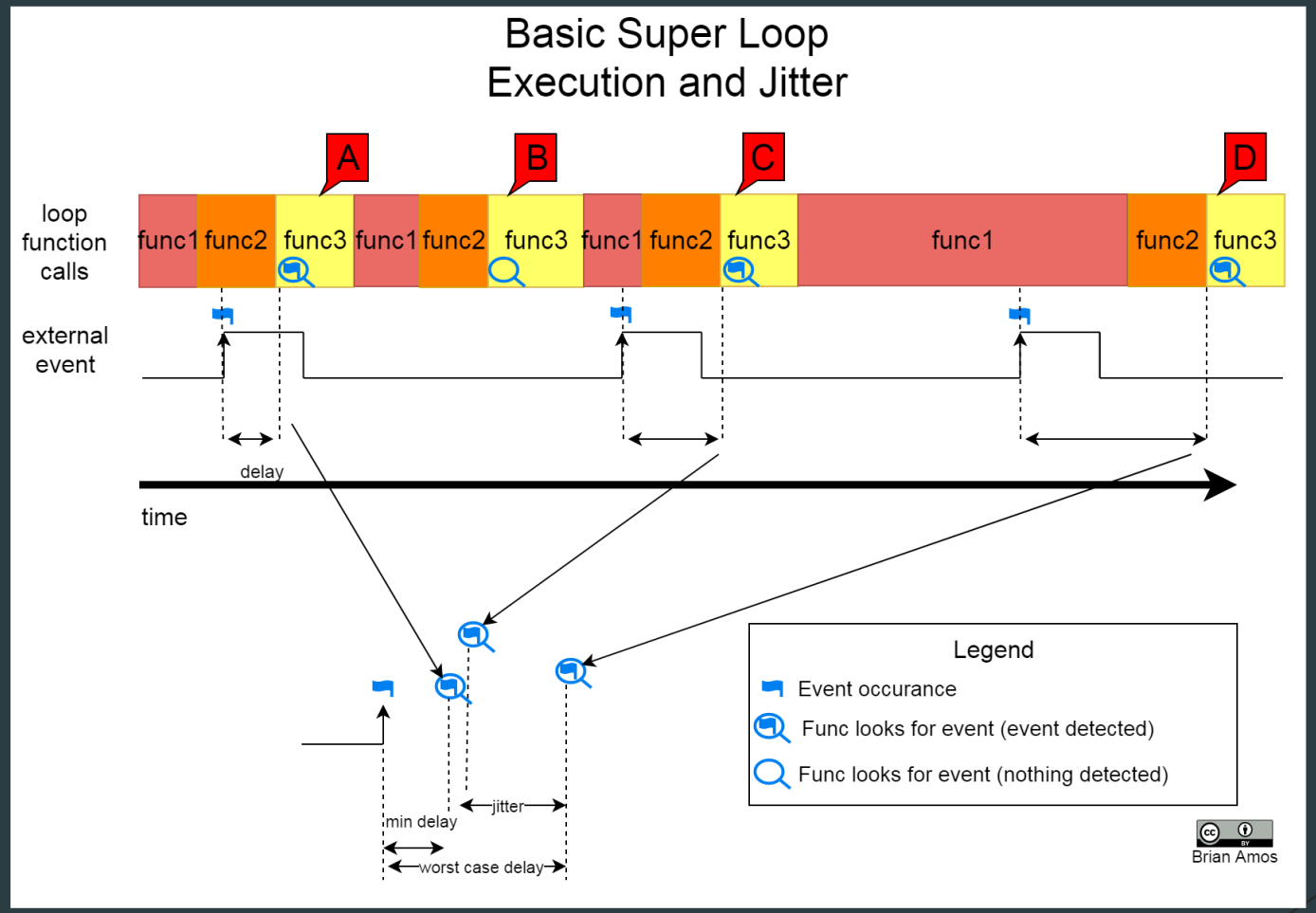
What is Real-Time?

- Any system that has a deterministic response to a given event can be considered "real-time."

Types Of Real Time Systems

- Hardware
- Bare-metal firmware
- RTOS-based firmware
- RTOS-based software
- Carefully crafted OS software

What is Jitter?



What is FreeRTOS?

- A lightweight, open-source real-time operating system for microcontrollers
- Designed for embedded systems with limited resources
- Compiled and linked directly with the application code

Bare Metal, Linux RT, and FreeRTOS – Ease of Use

Bare Metal	Linux RT	FreeRTOS
Low-Level Control	Familiar Environment	Modular Design
High Complexity	More Complex Setup and Configuration	Simpler Learning Carve And Easier To Setup Compared To Linux
More Development Time	Rich Ecosystem	Comprehensive Documentation

Bare Metal, Linux RT, and FreeRTOS – Performance and Resource Management

Bare Metal	Linux RT	FreeRTOS
Maximum Performance	Low Performance	Balanced Performance
Minimal Overhead	Higher Resource Demands - Requiring A Microprocessor With At Least 300mhz Speed	Efficient Resource Management
Smallest Footprint	A Kernel Size That Cannot Be Reduced Below Several Hundred Kilobytes	Minimal Footprint - RTOS Kernel Itself Required About 5 To 10 Kbytes Of ROM Space

Bare Metal, Linux RT, and FreeRTOS – Flexibility and Scalability

Bare Metal	Linux RT	FreeRTOS
High Flexibility	Highly Configurable	Modular And Portable
Limited Scalability, Typically Used For Very Specific Applications	Highly Scalable From Embedded Systems To Large Servers	Scalable For Small To Medium Embedded Systems

Bare Metal, Linux RT, and FreeRTOS – Use Case

Bare Metal	Linux RT	FreeRTOS
Suitable For Low-level System Development, Firmware, And Performance-critical Applications - Microcontroller-based applications, high-performance computing tasks, and real-time signal processing	Ideal For Complex Systems Requiring Both Real-time Performance And The Rich Features Of A General-purpose OS. Automotive systems, industrial automation, telecommunications, and mission-critical infrastructure.	Best For Embedded Systems With Real-time Requirements And Constrained Resources. IoT devices, consumer electronics, medical devices, and industrial control systems.

Why Simulate FreeRTOS on Linux?

- Faster development cycles without the need for physical hardware.
- Easier debugging and testing
- Cost-effective and convenient
- Ability to quickly iterate and test changes
- Speeding Up Development Cycles

Setting Up FreeRTOS Simulation on Linux

- Installing necessary software and tools
  - QEMU For Virtualization
  - GCC For Compiling
  - GDB For Debugging
- Complete Guide on:
  - https://github.com/HosseinAssaran/FreeRTOS_Examples

Let’s Do It