Training Course on Real-Time Embedded Systems Design

Professional Development Training Course Outline

Category	Engineering	**Duration**	10 Days
Base Fee	$3,000 USD	**Delivery Mode**	Online / On-site Hybrid

Course Introduction

Introduction

This comprehensive training course offers an in-depth exploration of Real-Time Embedded Systems Design, equipping participants with the critical skills to develop deterministic, high-performance, and ultra-reliable embedded solutions. Training Course on Real-Time Embedded Systems Design focuses on the fundamental principles and advanced techniques required to manage time-critical operations, covering real-time operating systems (RTOS), task scheduling, inter-process communication (IPC), and hardware-software co-design. Attendees will gain hands-on experience with popular real-time microcontrollers and development environments, mastering crucial concepts such as latency analysis, jitter reduction, priority inversion avoidance, and fault-tolerant design. This course is meticulously crafted to empower engineers to architect and implement robust real-time systems that meet the stringent demands of industries like aerospace, automotive, medical devices, and industrial automation.

The program emphasizes practical application and industry best practices, delving into cutting-edge topics like real-time Linux extensions (PREEMPT_RT), functional safety standards (e.g., ISO 26262), hardware acceleration, and advanced debugging for time-critical paths. Participants will learn to optimize code for predictable execution, manage complex resource contention, and ensure system responsiveness under all conditions. By the end of this course, attendees will possess the expertise to design, verify, and validate sophisticated real-time embedded systems, leveraging the full potential of both hardware and software to deliver guaranteed response times, high availability, and intrinsic safety. This training is essential for professionals seeking to enhance their proficiency in a demanding and highly specialized field, enabling them to contribute to the next generation of mission-critical embedded applications.

Programme Curriculum

Training Course on Real-Time Embedded Systems Design

Introduction

This comprehensive training course offers an in-depth exploration of Real-Time Embedded Systems Design, equipping participants with the critical skills to develop deterministic, high-performance, and ultra-reliable embedded solutions. Training Course on Real-Time Embedded Systems Design focuses on the fundamental principles and advanced techniques required to manage time-critical operations, covering real-time operating systems (RTOS), task scheduling, inter-process communication (IPC), and hardware-software co-design. Attendees will gain hands-on experience with popular real-time microcontrollers and development environments, mastering crucial concepts such as latency analysis, jitter reduction, priority inversion avoidance, and fault-tolerant design. This course is meticulously crafted to empower engineers to architect and implement robust real-time systems that meet the stringent demands of industries like aerospace, automotive, medical devices, and industrial automation.

The program emphasizes practical application and industry best practices, delving into cutting-edge topics like real-time Linux extensions (PREEMPT_RT), functional safety standards (e.g., ISO 26262), hardware acceleration, and advanced debugging for time-critical paths. Participants will learn to optimize code for predictable execution, manage complex resource contention, and ensure system responsiveness under all conditions. By the end of this course, attendees will possess the expertise to design, verify, and validate sophisticated real-time embedded systems, leveraging the full potential of both hardware and software to deliver guaranteed response times, high availability, and intrinsic safety. This training is essential for professionals seeking to enhance their proficiency in a demanding and highly specialized field, enabling them to contribute to the next generation of mission-critical embedded applications.

Course duration

10 Days

Course Objectives

1. Understand core concepts of real-time systems, including determinism, deadlines, and predictability.
2. Master real-time operating system (RTOS) architectures and their selection criteria.
3. Design and implement advanced real-time task scheduling algorithms (e.g., Rate Monotonic, EDF).
4. Effectively manage inter-process communication (IPC) and synchronization in real-time contexts.
5. Analyze and mitigate latency, jitter, and priority inversion in embedded systems.
6. Develop robust interrupt service routines (ISRs) for time-critical events.
7. Implement memory management strategies optimized for real-time performance.
8. Apply functional safety principles and standards (e.g., IEC 61508, ISO 26262) to real-time design.
9. Utilize real-time Linux extensions (PREEMPT_RT) for soft real-time applications.
10. Employ advanced debugging and profiling tools for real-time performance analysis.
11. Design and validate fault-tolerant and high-availability architectures for embedded systems.
12. Integrate hardware acceleration (FPGA, DSPs) for critical real-time tasks.
13. Develop test and verification strategies specific to real-time embedded systems.

Organizational Benefits

1. Development of highly reliable and deterministic embedded products, reducing field failures.
2. Improved system performance and responsiveness in time-critical applications.

3. Faster time-to-market for complex real-time solutions due to enhanced design proficiency.
4. Reduced development costs through optimized resource utilization and fewer design iterations.
5. Compliance with industry-specific safety and reliability standards, opening new market opportunities.
6. Enhanced ability to troubleshoot and resolve complex real-time issues in-house.
7. Increased capacity for developing sophisticated multi-threaded applications with RTOS.
8. Greater innovation in areas requiring precise timing and control.
9. Competitive advantage in markets where real-time performance is paramount.
10. Development of more robust and maintainable real-time embedded software.

Target Participants

- Embedded Software Engineers
- Real-Time Systems Architects
- Firmware Developers
- Hardware Engineers involved in embedded system design
- Test and Validation Engineers for embedded systems
- Technical Leads managing real-time projects
- Engineers working in safety-critical domains (automotive, aerospace, medical)

Course Outline

Module 1: Fundamentals of Real-Time Systems

- **Defining Real-Time:** Hard, soft, and firm real-time systems, deadlines, response time.
- **Characteristics of RTOS:** Determinism, predictability, concurrency.
- **Real-Time Metrics:** Latency, jitter, throughput.
- **System Components:** Tasks, interrupts, resources, schedulers.
- **Case Study:** Analyzing the real-time requirements of an automotive engine control unit (ECU).

Module 2: Real-Time Operating System (RTOS) Architecture

- **Kernel Services:** Task management, inter-task communication, memory management.
- **Context Switching:** Mechanism and overhead.
- **RTOS Selection Criteria:** Footprint, features, licensing, community support (FreeRTOS, Zephyr, VxWorks).
- **Porting an RTOS:** Adapting RTOS to specific hardware platforms.
- **Case Study:** Comparing FreeRTOS and Zephyr for a small-scale IoT real-time application.

Module 3: Advanced Real-Time Task Scheduling

- **Preemptive vs. Non-Preemptive Scheduling:** Advantages and disadvantages.
- **Fixed-Priority Scheduling:** Rate Monotonic Scheduling (RMS), Deadline Monotonic Scheduling (DMS).
- **Dynamic-Priority Scheduling:** Earliest Deadline First (EDF), Least Laxity First (LLF).
- **Schedulability Analysis:** Utilization bounds, response time analysis.
- **Case Study:** Designing and analyzing a task set for a robotic arm control system using RMS.

Module 4: Inter-Task Communication and Synchronization

- **Semaphores:** Binary, Counting, protecting shared resources.
- **Mutexes and Priority Inversion:** Solutions like Priority Inheritance, Priority Ceiling.
- **Message Queues:** Asynchronous communication, message passing, producer-consumer.
- **Event Flags and Mailboxes:** Signaling task completion and data transfer.
- **Case Study:** Implementing a data pipeline in an RTOS with sensors, processing tasks, and display using message queues and mutexes.

Module 5: Interrupt Handling and Determinism

- **Interrupt Service Routines (ISRs):** Structure, constraints, reentrancy.
- **Interrupt Latency and Jitter:** Minimizing non-determinism.
- **Deferred Interrupt Processing:** Task-level vs. ISR-level processing (e.g., FreeRTOS Direct-to-Task Notifications).
- **Critical Sections and Atomic Operations:** Protecting shared data from race conditions.
- **Case Study:** Optimizing an ISR for low-latency data acquisition from a high-speed sensor.

Module 6: Real-Time Memory Management

- **Heap Management in RTOS:** Dynamic memory allocation challenges, fragmentation.
- **Memory Pools:** Pre-allocated memory blocks for deterministic allocation.
- **Stack Management:** Stack overflow detection, optimizing stack sizes.
- **Memory Protection Units (MPU):** Hardware-assisted memory access control for robustness.
- **Case Study:** Designing memory pools for a real-time audio processing application to avoid dynamic allocation issues.

Module 7: Real-Time Linux (PREEMPT_RT)

- **Soft vs. Hard Real-Time on Linux:** Understanding the limitations of standard Linux.
- **PREEMPT_RT Patch:** Kernel preemption, high-resolution timers, priority inheritance.
- **Configuring a Real-Time Linux Kernel:** Building and deploying PREEMPT_RT.
- **Real-Time Programming APIs:** POSIX real-time extensions (sched_setscheduler, pthread_attr_setschedpolicy).
- **Case Study:** Porting a real-time control algorithm from an RTOS to a PREEMPT_RT Linux environment.

Module 8: Functional Safety and Reliability

- **Introduction to Functional Safety:** IEC 61508, ISO 26262, DO-178C.
- **Safety Life Cycle:** Hazard analysis, risk assessment, safety integrity levels (SIL, ASIL).
- **Fault Tolerance Techniques:** Redundancy, watchdog timers, error detection and correction.
- **Safety Mechanisms:** Self-tests, diagnostics, fail-safe states.
- **Case Study:** Designing a watchdog timer system for a safety-critical medical device.

Module 9: Hardware-Software Co-Design for Real-Time

- **Processor Selection:** DSPs, FPGAs, Microcontrollers, Microprocessors for real-time tasks.
- **Hardware Acceleration:** Offloading computationally intensive tasks.
- **DMA and Memory Architectures:** Optimizing data flow for real-time.
- **Custom Hardware Interfaces:** Designing peripherals for specific real-time needs.
- **Case Study:** Deciding between a microcontroller and an FPGA for a high-speed image processing task.

Module 10: Advanced Real-Time Debugging and Profiling

- **Real-Time Trace Tools:** Segger SystemView, Percepio Tracealyzer for event logging.
- **Hardware Debuggers:** JTAG, SWD, trace capabilities for non-intrusive monitoring.
- **Performance Counters:** Using hardware counters to identify bottlenecks.
- **Debugging Multithreaded Issues:** Race conditions, deadlocks in real-time.
- **Case Study:** Using a real-time trace tool to diagnose a subtle priority inversion issue in an RTOS application.

Module 11: Time Synchronization and Network Protocols

- **Network Time Protocol (NTP):** Synchronizing device clocks.
- **Precision Time Protocol (PTP - IEEE 1588):** High-accuracy time synchronization for industrial networks.
- **Real-Time Ethernet:** EtherCAT, Profinet IRT, TSN (Time-Sensitive Networking).
- **Determinism in Networked Systems:** Managing network latency and jitter.
- **Case Study:** Implementing PTP for time synchronization across multiple sensors in an industrial automation system.

Module 12: Real-Time Data Acquisition and Signal Processing

- **High-Speed ADC/DAC Interfacing:** Nyquist theorem, anti-aliasing filters.
- **Digital Filtering:** FIR, IIR filters for real-time signal processing.
- **Fast Fourier Transform (FFT) on Microcontrollers:** Real-time spectrum analysis.
- **Sensor Fusion for Real-Time:** Combining data for robust perception (e.g., Kalman filter).
- **Case Study:** Designing a real-time vibration monitoring system with FFT analysis on an embedded processor.

Module 13: System Integration and Testing for Real-Time

- **Hardware-in-the-Loop (HIL) Testing:** Testing software with real hardware components.
- **Software-in-the-Loop (SIL) Testing:** Simulating embedded software execution.
- **Test Automation for Real-Time:** Automated regression testing, test frameworks.
- **Stress Testing and Robustness Testing:** Validating system behavior under extreme conditions.
- **Case Study:** Developing a HIL test bench for an automotive real-time braking system.

Module 14: Safety-Critical Real-Time Software Development

- **Coding Standards:** MISRA C/C++ for safety and reliability.
- **Static and Dynamic Analysis:** Tools for code quality and defect detection.
- **Formal Methods:** Introduction to formal verification for critical systems.
- Upcoming Scheduled Sessions

Start Date	End Date	Location	Price
07 Sep 2026	18 Sep 2026	Online	$2,200
14 Sep 2026	25 Sep 2026	Online	$2,200
21 Sep 2026	02 Oct 2026	Online	$2,200
28 Sep 2026	09 Oct 2026	Online	$2,200

Start Date	End Date	Location	Price
05 Oct 2026	16 Oct 2026	Online	$2,200
12 Oct 2026	23 Oct 2026	Online	$2,200
19 Oct 2026	30 Oct 2026	Online	$2,200
26 Oct 2026	06 Nov 2026	Online	$2,200

Ready to enroll or request a customized program? Book online or contact us:

Register Online Now

Email: info@fineskilltrainingcenter.com | Website: www.fineskilltrainingcenter.com