# Training Course on Firmware Development and Debugging Techniques

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 intensive training course provides a comprehensive deep dive into Firmware Development and Debugging Techniques, equipping participants with the essential skills to write robust, efficient, and reliable low-level software for embedded systems. The curriculum moves beyond basic programming, focusing on bare-metal programming, real-time considerations, memory management, and optimizing code for constrained environments. Attendees will gain hands-on expertise with industry-standard toolchains, mastering crucial concepts such as bootloader design, interrupt service routines (ISRs), peripheral drivers, and advanced debugging methodologies like JTAG/SWD and real-time trace. Training Course on Firmware Development and Debugging Techniques is meticulously crafted to empower engineers to tackle the complexities of modern embedded development, ensuring the creation of high-quality firmware that forms the backbone of countless devices, from IoT sensors to sophisticated control systems.

The program emphasizes practical application and problem-solving, exploring trending topics like secure firmware updates (OTA), performance profiling, fault injection testing, and advanced unit testing for embedded code. Participants will delve into intricate hardware-software interactions, learning to diagnose elusive bugs, optimize execution speed, and ensure deterministic behavior. By the end of this course, attendees will possess the expertise to architect, implement, and rigorously debug complex firmware solutions, leveraging both traditional and cutting-edge techniques to deliver reliable performance, optimized resource utilization, and enhanced system stability. This training is indispensable for professionals seeking to elevate their firmware development prowess and master the art of bringing embedded hardware to life.

## Programme Curriculum

### Training Course on Firmware Development and Debugging Techniques

### Introduction

This intensive training course provides a comprehensive deep dive into Firmware Development and Debugging Techniques, equipping participants with the essential skills to write robust, efficient, and reliable low-level software for embedded systems. The curriculum moves beyond basic programming, focusing on bare-metal programming, real-time considerations, memory management, and optimizing code for constrained environments. Attendees will gain hands-on expertise with industry-standard toolchains, mastering crucial concepts such as bootloader design, interrupt service routines (ISRs), peripheral drivers, and advanced debugging methodologies like JTAG/SWD and real-time trace. Training Course on Firmware Development and Debugging Techniques is meticulously crafted to empower engineers to tackle the complexities of modern embedded development, ensuring the creation of high-quality firmware that forms the backbone of countless devices, from IoT sensors to sophisticated control systems.

The program emphasizes practical application and problem-solving, exploring trending topics like secure firmware updates (OTA), performance profiling, fault injection testing, and advanced unit testing for embedded code. Participants will delve into intricate hardware-software interactions, learning to diagnose elusive bugs, optimize execution speed, and ensure deterministic behavior. By the end of this course, attendees will possess the expertise to architect, implement, and rigorously debug complex firmware solutions, leveraging both traditional and cutting-edge techniques to deliver reliable performance, optimized resource utilization, and enhanced system stability. This training is indispensable for professionals seeking to elevate their firmware development prowess and master the art of bringing embedded hardware to life.

**Course duration**

10 Days

## Course Objectives

1. Master bare-metal firmware development for microcontrollers.
2. Design and implement robust bootloaders for secure and reliable system startup.
3. Develop efficient and reliable peripheral drivers for various hardware components.
4. Understand and manage memory organization in constrained embedded environments.
5. Implement advanced interrupt service routines (ISRs) for time-critical responses.
6. Utilize Direct Memory Access (DMA) for high-throughput data transfers.
7. Apply advanced debugging techniques using JTAG/SWD, trace, and logic analyzers.
8. Perform firmware performance profiling and optimization.
9. Implement secure firmware update (OTA) mechanisms.
10. Develop robust error handling and fault tolerance strategies in firmware.
11. Conduct unit testing and integration testing for embedded software.
12. Diagnose and resolve complex hardware-software interaction issues.
13. Leverage source code analysis and static analysis tools for quality assurance.

## Organizational Benefits

1. Reduced development cycles due to enhanced firmware development proficiency.
2. Improved product reliability and stability through robust firmware design.
3. Faster identification and resolution of complex bugs, leading to reduced debugging time.
4. Optimized product performance and resource utilization (memory, CPU).
5. Reduced product recall rates due to fewer firmware defects.
6. Enhanced ability to implement secure firmware updates for deployed devices.
7. Greater confidence in embedded product quality through systematic testing.

8. Increased capacity for in-house development of complex embedded systems.
9. Competitive advantage by delivering highly reliable and efficient embedded products.
10. Lower post-production support costs due to more stable firmware.

## Target Participants

- Embedded Software Engineers
- Firmware Developers
- Hardware Engineers
- System Architects
- IoT Device Developers
- Test and Validation Engineers for embedded systems
- Technical Leads in embedded product development

## Course Outline

### Module 1: Introduction to Firmware Development

- **Firmware vs. Software:** Key differences, bare-metal vs. OS-based firmware.
- **Embedded System Architecture Overview:** Microcontrollers, memory types, peripherals.
- **Development Toolchain:** Compilers (GCC, Keil, IAR), linkers, debuggers, IDEs.
- **Programming Languages:** C, C++ for embedded systems.
- **Case Study:** Setting up a new project in an embedded IDE for a basic blinking LED application.

### Module 2: Microcontroller Architecture Deep Dive

- **Processor Cores:** ARM Cortex-M series (M0/M3/M4/M7), RISC-V basics.
- **Memory Map and Organization:** Flash, SRAM, Registers, Memory Protection Unit (MPU).
- **Clock Systems:** Oscillators, PLLs, clock tree configuration.
- **Reset Management:** Power-on reset, external resets, watchdog resets.
- **Case Study:** Analyzing the memory map and clock configuration of a target microcontroller.

### Module 3: Bootloader Development

- **Boot Process Fundamentals:** Power-on reset to application execution.
- **Types of Bootloaders:** Initial Program Loader (IPL), Secondary Bootloaders, DFU.
- **Bootloader Design Principles:** Small footprint, robustness, update mechanisms.
- **Security Considerations:** Authenticated boot, rollback prevention.
- **Case Study:** Developing a simple UART-based bootloader for firmware updates.

### Module 4: Peripheral Driver Development

- **General Purpose Input/Output (GPIO):** Advanced configurations, interrupts (EXTI).
- **Serial Communication:** UART, SPI (Master/Slave), I2C (Master/Slave).
- **Analog-to-Digital Converters (ADC) and Digital-to-Analog Converters (DAC):** Advanced modes, calibration.
- **Timers and PWM:** Advanced modes (Input Capture, Output Compare, complex PWM).
- **Case Study:** Writing a custom SPI driver to interface with an external flash memory chip.

### Module 5: Interrupt Service Routines (ISRs) and Exception Handling

- **Interrupt Controller (NVIC):** Priority, preemption, vector table.
- **ISR Design Best Practices:** Short execution time, reentrancy, avoiding blocking calls.
- **Deferred Interrupt Processing:** Using flags, semaphores, or queues to move work to tasks.
- **Exception Handling:** Hard Faults, Bus Faults, debugging and resolving them.
- **Case Study:** Optimizing an ISR for a high-frequency input to minimize latency and jitter.

**Module 6: Direct Memory Access (DMA)**

- **DMA Controller Architecture:** Channels, streams, circular mode, buffer management.
- **DMA Transfer Modes:** Peripheral-to-memory, memory-to-peripheral, memory-to-memory.
- **DMA with Peripherals:** UART, SPI, ADC for high-speed data transfer.
- **DMA Interrupts and Error Handling:** Ensuring reliable data transfers.
- **Case Study:** Implementing DMA-driven UART communication for efficient data logging.

**Module 7: Memory Management in Firmware**

- **Memory Sections:** .text, .data, .bss, .rodata, stack, heap.
- **Linker Scripts:** Customizing memory placement and organization.
- **Heap Management:** Dynamic memory allocation considerations, fragmentation.
- **Stack Analysis:** Estimating stack usage, overflow detection.
- **Case Study:** Optimizing memory usage for a firmware application by customizing the linker script and implementing a memory pool.

**Module 8: Introduction to Embedded Debugging**

- **Debugging Interfaces:** JTAG, SWD, UART, SWV (Serial Wire Viewer).
- **Debugger Features:** Breakpoints, watchpoints, step-by-step execution, memory inspection.
- **Real-Time Data Views:** Peripheral registers, RTOS awareness.
- **Non-Intrusive Debugging:** Trace capabilities (ETM, ITM).
- **Case Study:** Using a hardware debugger (e.g., SEGGER J-Link) to step through code and inspect variables.

**Module 9: Advanced Debugging Techniques**

- **Debugging Hard Faults and Exceptions:** Identifying root causes.
- **Debugging Race Conditions and Deadlocks:** Using real-time trace and logic analyzers.
- **Debugging Power-Related Issues:** Correlating code execution with current consumption.
- **Watchdog Timers and Resets:** Debugging unexpected resets.
- **Case Study:** Diagnosing a complex intermittent bug involving a race condition between an ISR and a main loop task.

**Module 10: Firmware Performance Profiling and Optimization**

- **Performance Metrics:** CPU utilization, execution time, code size, memory footprint.
- **Profiling Tools:** CPU cycle counters, trace tools, built-in profilers.
- **Optimization Techniques:** Compiler flags, algorithm optimization, loop unrolling.
- **Assembly Language Optimization (Basics):** When and how to use it.
- **Case Study:** Profiling a critical algorithm in firmware and optimizing its execution time.

**Module 11: Firmware Testing and Validation**

- **Unit Testing for Embedded Code:** Challenges and frameworks (Unity, Google Test for embedded).
- **Integration Testing:** Testing interactions between modules and hardware.
- **Hardware-in-the-Loop (HIL) Testing:** Testing firmware with simulated environments.

- **Test Automation:** Automating firmware build and test processes.
- **Case Study:** Writing unit tests for a peripheral driver and integrating them into a build system.

**Module 12: Secure Firmware Updates (OTA)**

- **OTA Architectures:** Dual-bank (A/B) updates, delta updates.
- **Cryptographic Signatures:** Authenticating firmware images.
- **Rollback Prevention:** Protecting against downgrade attacks.
- **Robustness and Error Recovery:** Handling power loss during updates.
- **Case Study:** Implementing a secure and robust OTA update mechanism for a remote IoT device.

**Module 13: Fault Tolerance and Error Handling**

- **Watchdog Timers:** Independent and Windowed Watchdogs for system recovery.
- **Cyclic Redundancy Check (CRC):** Data integrity verification.
- **Error Correction Codes (ECC):** Memory error detection and correction.
- **Fail-Safe and Fail-Operational Design:** Strategies for critical systems.
- **Case Study:** Designing an error detection and recovery mechanism for critical data storage using CRC.

**Module 14: Source Code Analysis and Quality Assurance**

- **Static Analysis Tools:** Lint, PC-Lint, SonarQube for code quality and security.
- **Code Reviews:** Best practices for peer review.
- **Coding Standards:** MISRA C/C++ for safety and reliability.
- **Upcoming Scheduled Sessions**

| Start Date | End Date | Location | Price |
|---|---|---|---|
| **21 Sep 2026** | 02 Oct 2026 | Online | $2,000 |
| **21 Sep 2026** | 02 Oct 2026 | Physical | $3,000 |
| **21 Sep 2026** | 02 Oct 2026 | Physical | $0 |
| **21 Sep 2026** | 02 Oct 2026 | Physical | $0 |
| **21 Sep 2026** | 02 Oct 2026 | Physical | $0 |
| **21 Sep 2026** | 02 Oct 2026 | Physical | $0 |
| **21 Sep 2026** | 02 Oct 2026 | Physical | $0 |
| **21 Sep 2026** | 02 Oct 2026 | Physical | $0 |

Register Online Now

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