Training Course on Advanced Microcontroller Programming and Interfacing

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 advanced microcontroller programming and interfacing, equipping participants with the sophisticated skills needed to develop complex embedded systems. The curriculum moves beyond basic concepts, focusing on 32-bit ARM Cortex-M microcontrollers, advanced peripheral utilization, and efficient code optimization. Attendees will gain hands-on expertise in interrupt-driven design, direct memory access (DMA), low-power modes, and real-time operating system (RTOS) integration for bare-metal applications. Training Course on Advanced Microcontroller Programming and Interfacing is meticulously designed to empower engineers to create high-performance, ultra-reliable, and energy-efficient embedded solutions for a wide range of applications, from industrial control to cutting-edge IoT devices.

The program emphasizes practical application, industry best practices, and the integration of trending technologies such as sensor fusion, communication protocols (CAN, Ethernet), and robust debugging methodologies. Participants will explore advanced memory management techniques, advanced timer applications for precision control, and strategies for ensuring system reliability and safety. By the end of this course, attendees will possess the expertise to architect and implement sophisticated microcontroller-based systems, leveraging the full capabilities of modern hardware to deliver deterministic behavior, optimized resource utilization, and enhanced functionality. This training is indispensable for professionals seeking to elevate their embedded programming skills and tackle the complexities of next-generation microcontroller applications.

Programme Curriculum

Training Course on Advanced Microcontroller Programming and Interfacing

Introduction

This intensive training course provides a comprehensive deep dive into advanced microcontroller programming and interfacing, equipping participants with the sophisticated skills needed to develop complex embedded systems. The curriculum moves beyond basic concepts, focusing on 32-bit ARM Cortex-M microcontrollers, advanced peripheral utilization, and efficient code optimization. Attendees will gain hands-on expertise in interrupt-driven design, direct memory access (DMA), low-power modes, and real-time operating system (RTOS) integration for bare-metal applications. Training Course on Advanced Microcontroller Programming and Interfacing is meticulously designed to empower engineers to create high-performance, ultra-reliable, and energy-efficient embedded solutions for a wide range of applications, from industrial control to cutting-edge IoT devices.

The program emphasizes practical application, industry best practices, and the integration of trending technologies such as sensor fusion, communication protocols (CAN, Ethernet), and robust debugging methodologies. Participants will explore advanced memory management techniques, advanced timer applications for precision control, and strategies for ensuring system reliability and safety. By the end of this course, attendees will possess the expertise to architect and implement sophisticated microcontroller-based systems, leveraging the full capabilities of modern hardware to deliver deterministic behavior, optimized resource utilization, and enhanced functionality. This training is indispensable for professionals seeking to elevate their embedded programming skills and tackle the complexities of next-generation microcontroller applications.

Course duration

10 Days

Course Objectives

1. Master advanced ARM Cortex-M architecture features and programming models.
2. Design and implement complex interrupt service routines (ISRs) for high-performance systems.
3. Utilize Direct Memory Access (DMA) for efficient data transfers and reduced CPU load.
4. Develop power-optimized microcontroller applications for extended battery life.
5. Integrate and configure Real-Time Operating Systems (RTOS) for multi-tasking embedded solutions.
6. Implement advanced timer and pulse-width modulation (PWM) techniques for precision control.
7. Interface with diverse sensors and actuators using advanced communication protocols (SPI, I2C, UART).
8. Develop robust communication solutions using CAN bus and Ethernet (LwIP).
9. Apply sensor fusion algorithms for accurate environmental perception.
10. Implement secure coding practices and basic cryptographic operations on microcontrollers.
11. Troubleshoot and debug complex hardware-software interactions using advanced tools.
12. Design and validate fault-tolerant microcontroller systems for enhanced reliability.
13. Optimize code for performance and memory footprint in resource-constrained environments.

Organizational Benefits

1. Reduced development cycles for complex microcontroller-based products.
2. Improved product performance and efficiency through optimized code and resource utilization.

3. Enhanced reliability and robustness of embedded systems, leading to fewer field failures.
4. Lower power consumption in products, translating to longer battery life and reduced energy costs.
5. Increased capacity for in-house development of sophisticated embedded solutions.
6. Faster integration of new sensors and communication standards into products.
7. Stronger defense against embedded system vulnerabilities through secure coding.
8. Greater ability to troubleshoot and diagnose complex hardware-software issues.
9. Competitive advantage by adopting cutting-edge microcontroller programming techniques.
10. Development of more innovative and feature-rich embedded products.

Target Participants

- Embedded Software Engineers
- Hardware Engineers
- System Architects
- Firmware Developers
- IoT Device Developers
- Electrical Engineers interested in embedded programming
- Technical Leads in embedded systems development

Course Outline

Module 1: Advanced ARM Cortex-M Architecture

- **Cortex-M Core Internals:** Pipeline, instruction sets (Thumb-2), register set, FPU.
- **Memory Map and Access:** Flash, SRAM, Peripherals, Memory Protection Unit (MPU).
- **Interrupt and Exception Handling:** NVIC, interrupt priority, preemption, vector table.
- **Reset and Clock Control:** Power-on reset, clock tree configuration, system states.
- **Case Study:** Configuring a microcontroller's clock system for optimal performance and power.

Module 2: Advanced General Purpose Input/Output (GPIO)

- **GPIO Modes and Configurations:** Input, output, alternate functions, analog, open-drain, push-pull.
- **External Interrupts (EXTI):** Edge/level triggering, interrupt callback functions.
- **Bit-Banding and Atomic Operations:** Efficient and safe register manipulation.
- **Power Consumption with GPIO:** Managing pull-ups/downs, output drive strength.
- **Case Study:** Implementing a multi-button interface with debouncing and advanced interrupt handling.

Module 3: Timers for Precision Control

- **Advanced Timer Modes:** Input Capture, Output Compare, PWM generation (center-aligned, complementary).
- **Encoder Interfaces:** Reading quadrature encoders for motor position feedback.
- **One-Pulse Mode and Gated Timers:** Specific event generation and synchronization.
- **Timer Synchronization and Chaining:** Coordinating multiple timers for complex sequences.

- **Case Study:** Developing a precise motor speed control system using advanced PWM and encoder input.

Module 4: 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:** ADC, DAC, SPI, I2C, UART for high-speed data transfer.
- **DMA Interrupts and Error Handling:** Ensuring reliable data transfers.
- **Case Study:** Implementing high-throughput data logging from an ADC to a memory buffer using DMA.

Module 5: Real-Time Operating Systems (RTOS) Integration

- **RTOS Concepts Review:** Tasks, scheduling, inter-task communication, synchronization.
- **Porting an RTOS:** FreeRTOS/Zephyr on ARM Cortex-M, kernel configuration.
- **Task Management and Priorities:** Dynamic task creation, priority assignment.
- **RTOS Objects:** Semaphores, mutexes, message queues, event groups.
- **Case Study:** Migrating a bare-metal multi-threaded application to a FreeRTOS environment.

Module 6: Advanced Serial Communication (UART, SPI, I2C)

- **UART with DMA:** Efficient reception and transmission without CPU overhead.
- **SPI Master/Slave:** Multi-slave communication, advanced modes (CPOL, CPHA).
- **I2C Master/Slave:** Multi-master arbitration, clock stretching.
- **Error Handling and Robustness:** Checksums, retries, timeout mechanisms.
- **Case Study:** Interfacing a high-speed sensor (e.g., IMU) via SPI with DMA.

Module 7: Analog-to-Digital (ADC) and Digital-to-Analog (DAC) Converters

- **ADC Advanced Modes:** Scan mode, continuous conversion, injected channels.
- **DAC Waveform Generation:** Arbitrary waveform generation, DMA-driven output.
- **Sampling Techniques:** Oversampling, averaging for noise reduction.
- **Calibration and Linearity Correction:** Improving measurement accuracy.
- **Case Study:** Building a data acquisition system for multiple analog inputs with high precision.

Module 8: Low-Power Design and Energy Management

- **Microcontroller Power Modes:** Sleep, Stop, Standby, Vbat modes.
- **Wake-up Sources:** Interrupts, RTC, external pins.
- **Power Consumption Analysis:** Using current meters, optimization techniques.
- **Energy Harvesting Integration:** Interfacing with solar, thermoelectric generators.
- **Case Study:** Designing a battery-powered sensor node with multi-year battery life.

Module 9: Controller Area Network (CAN Bus)

- **CAN Protocol Deep Dive:** Arbitration, message frames, error handling.
- **CAN Transceivers and Physical Layer:** ISO 11898 standard.
- **CAN Controller Configuration:** Mailboxes, filters, acceptance masks.
- **CANopen and J1939 Overview:** Higher-layer protocols for industrial and automotive.
- **Case Study:** Implementing CAN communication between two microcontrollers for a distributed control system.

Module 10: Ethernet Connectivity (LwIP)

- **Ethernet MAC/PHY Interfacing:** MII, RMII, SGMII.
- **TCP/IP Stack (LwIP):** Concepts, configuration, memory pools.
- **Sockets Programming:** TCP client/server, UDP communication.
- **Web Server Implementation:** HTTP server on a microcontroller.
- **Case Study:** Developing an embedded web server to remotely control a device over Ethernet.

Module 11: Memory Management and External Memory Interfaces

- **External Memory Controllers (EMC/FMC):** Connecting to external SDRAM, SRAM, NOR/NAND Flash.
- **Memory Protection Unit (MPU) Applications:** Isolating tasks, preventing memory corruption.
- **Bootloader Design:** Secure boot, firmware update mechanisms.
- **File Systems on External Flash:** FatFs, LittleFS for data storage.
- **Case Study:** Interfacing an external SDRAM module and using it for large data buffers.

Module 12: Sensor Fusion and Data Processing

- **Introduction to Sensor Fusion:** Combining data from multiple sensors for improved accuracy.
- **Filtering Techniques:** Kalman filter, Complementary filter for IMU data.
- **Digital Signal Processing (DSP) Basics:** FIR, IIR filters, FFT for microcontroller.
- **Using CMSIS-DSP Library:** Optimized DSP functions for ARM Cortex-M.
- **Case Study:** Implementing a sensor fusion algorithm for an Inertial Measurement Unit (IMU) to estimate orientation.

Module 13: Advanced Debugging and Profiling

- **SWD/JTAG Debugging:** Breakpoints, watchpoints, trace capabilities.
- **Real-Time Monitoring:** Segger SystemView, Percepio Tracealyzer.
- **Fault Analysis:** Hard Faults, Bus Faults, debugging exception handlers.
- **Performance Profiling:** Identifying CPU hotspots and optimization opportunities.
- **Case Study:** Diagnosing a hard fault exception in an RTOS application and profiling task execution times.

Module 14: System Reliability and Safety Features

- **Watchdog Timers:** Independent and Windowed Watchdogs for system recovery.
- **CRC (Cyclic Redundancy Check):** Data integrity verification.
- **ECC (Error Correction Code):** Memory error detection and correction.
- **Self-Test and Diagnostics:** Built-in self-test (BIST) routines.
- **Case Study:** Implementing redundant watchdog timers and CRC checks for critical data storage.

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

Start Date	End Date	Location	Price
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

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

Register Online Now

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