CONCEPT

Embedded Systems and Microcontroller Programming Course: Embedded C, ARM Cortex-M and RTOS

6 – 10 June 2027
Jeddah - Corniche four-star



Embedded Systems and Microcontroller Programming Course: Embedded C, ARM Cortex-M and RTOS

Ref: 1406_23461 **Date:** 6 – 10 June 2027 **Location:** Jeddah - Corniche four-star **Fees:** 19500 SAR

Introduction:

Embedded systems and microcontroller programming with embedded C, ARM Cortex-M and RTOS is a 1-week course for electronics, instrumentation and product engineering staff, ending with a Sensor Data Logger Firmware Package. Organisations that build or maintain microcontroller-based devices lose time and reputation to firmware that hangs in the field, drains batteries, misses interrupts or cannot be updated safely after release. Nominees already write or modify C firmware at work, and the course is taught as a hands-on lab on Cortex-M development boards with debug probes. CoreConcept Training Center delivers this embedded systems course.

Course Objectives:

- Map device firmware onto the Cortex-M memory map, core profile and a super loop, interrupt-driven or RTOS architecture
- Write register-level embedded C drivers for GPIO, timers, PWM, ADC and DMA and check them against MISRA C guidelines
- Configure NVIC priorities and interrupt handlers that share data with application code without corruption
- Implement UART, SPI, I2C and CAN communication and diagnose bus errors with a logic analyser and the debug probe
- Design RTOS tasks, queues and mutexes that avoid priority inversion and deadlock, and set sleep modes to a power budget
- Build updatable firmware with watchdog and brown-out handling, a bootloader update path and host-based unit tests

Target Audience:

- Electronics design engineers responsible for microcontroller firmware in new products
- Instrumentation engineers responsible for smart sensors, transmitters and data acquisition devices
- Product engineers responsible for maintaining and updating firmware on devices already in service
- Test and validation engineers responsible for verifying firmware behaviour on hardware benches
- Hardware engineers taking on firmware responsibility within mixed hardware and software teams

Course Outline:

Day 1: Embedded System Architecture, Microcontroller Internals and Memory Map

- Embedded System Block Model From Sensor Input to Communication Output
- ARM Cortex-M Core Profiles From ARMv6-M to ARMv8.1-M
- Microcontroller Memory Map Covering Flash, SRAM and Peripheral Registers
- Super Loop Versus Interrupt-Driven Versus RTOS Firmware Architectures
- Current-State Review of an Existing Device Firmware Codebase

Day 2: Embedded C Essentials, Toolchains and On-Chip Debugging

- Embedded C Volatile, Const and Fixed-Width Integer Types
- Bit Manipulation Masks and Register Access Through Structure Overlays
- Cross-Compiler Toolchain, Linker Script and Startup Code Walkthrough
- MISRA C Guideline Categories Checked With Static Analysis
- SWD and JTAG Debug Probes With Breakpoints and Watchpoints

Day 3: GPIO, Timers, Interrupts, ADC and Serial Bus Drivers

- GPIO Configuration for Push-Pull, Open-Drain and Debounced Inputs
- Hardware Timers, SysTick and PWM Output Generation
- NVIC Interrupt Priorities, Handlers and Shared Data Protection
- ADC Sampling, DMA Transfers and Sensor Signal Scaling
- UART, SPI and I2C Driver Writing With Logic Analyser Checks

Day 4: CAN Networking, Low-Power Design, RTOS Tasks and Firmware Reliability

- CAN Frame Arbitration, Error Counters and Bus-Off Recovery
- Low-Power Sleep Modes and Wake-Up Source Budgeting
- RTOS Task Design With Priorities, Queues and Semaphores
- Priority Inversion, Deadlock and Mutex Priority Inheritance Cases
- Watchdog Timer, Brown-Out Detection and Bootloader Firmware Update

Day 5: Lab Building a Sensor Data Logger With CAN Reporting

- Data Logger Requirements, Pin Allocation and Peripheral Plan
- Sensor Acquisition Task and Circular Buffer Implementation Lab
- CAN Message Reporting and Fault Injection Test Lab
- Host-Based Unit Tests and Sleep Current Measurement Lab
- Sensor Data Logger Firmware Package Completion and Peer Review

Skills You Will Gain:

- Register-Level Driver Development
- Interrupt Priority Design
- Serial Bus Diagnostics
- RTOS Task Partitioning
- Low-Power Firmware Budgeting
- On-Chip Debugging
- Static Code Analysis
- Firmware Update Design

Why Attend This Course:

- Hand a Sensor Data Logger Firmware Package, with drivers, task design, test results and power figures, to the engineering lead who owns the next device release
- Choose between a super loop, an interrupt-driven design or an RTOS for a new product, and justify task priorities and stack sizes
- Avoid field failures, battery drain and unrecoverable devices caused by unprotected shared data, missed watchdog service and unsafe firmware updates
- Pass on driver templates, a MISRA C review checklist and a bus debugging procedure to colleagues who write firmware

Conclusion:

Back at work, the participant hands the engineering lead a Sensor Data Logger Firmware Package containing register-level drivers, an RTOS task map, CAN message definitions, unit test results and a measured power budget. The unit can reuse it as a reference design when choosing a firmware architecture for the next microcontroller product and when setting coding and review rules for the team. After its first use on a live project, the unit should review static analysis findings, watchdog reset counts during testing and measured sleep current against the budget.

Frequently Asked Questions FAQ:

What should participants know before the embedded systems and microcontroller programming course?

Participants should already read schematics and write C, including functions, pointers and structures. Experience of flashing and debugging a microcontroller helps. No RTOS knowledge is assumed. A short description of a device they support helps them relate each lab to their own firmware.

How does the embedded systems and microcontroller programming course differ from a PLC programming or electronics repair course?

It teaches writing C firmware directly against microcontroller registers, peripherals and an RTOS. PLC courses program industrial controllers in ladder and function block languages, and electronics repair courses fault-find boards at component level without writing code.

When does embedded systems firmware need an RTOS instead of a super loop?

An RTOS earns its place when several activities with different deadlines must run together, such as sampling sensors, serving a bus and logging data. A super loop suits simple devices; an RTOS adds scheduling, queues and mutexes, but also memory use and priority inversion risks.

What does a participant take back from the embedded systems and microcontroller programming course?

Each participant takes back a Sensor Data Logger Firmware Package: Cortex-M drivers for timers, ADC and serial buses, an RTOS task design, CAN reporting code, host-based unit tests, a MISRA C review record and a measured power budget, ready to adapt to a device in their unit.