FEABHAS

C for Real-Time Embedded Developers

Course category	C Training Courses
Training area	Programming Languages
Course code	C-501
Duration	5 days
Course date	19th February 2024
Price exc VAT	£3000.00
Additional information	Public courses delivered remotely over Zoom, using QEMU to emulate the hardware

Learning the C programming language is one thing, but learning how to use it effectively and appropriately in a real-time embedded environment is another. For many programmers, these skills are learnt the hard way, through trial and error on the job. This course addresses this issue by not only teaching the C programming language, but through emphasising the pragmatic use of C in a real-time environment. This is achieved through both lecture and direct hands-on experience.

This is an intensive five day course covering C in the context of real-time embedded application development. Approximately 50% of the course consists of writing code for a real target. During the week, attendees will build up a complete solution to a case study that exercises all significant parts of the course.

Course objectives:

- To provide an understanding of the essentials of the C programming language.
- To give you practical experience of writing C for real-time and embedded systems.
- To demonstrate the traps and pitfalls of the language when used in a real-time system.
- To give you the confidence to apply these new concepts to your next real-time project.

Delegates will learn:

- The core C syntax and semantics
- How to access hardware and program interrupts in the language
- About memory and performance issues associated with C
- Best practices in writing C for robust systems

Pre-requisites:

- Experience of programming, e.g. another high-level language

- Prior knowledge of C is useful but not essential

Who should attend:

This course is designed for engineers who have little or no experience of using C in a real-world production environment. It brings real-world knowledge to those with an academic understanding or who are self-taught. It is also suitable for those needing to support a customer using C, and those requiring to be brought up to date since initially being taught at University.

Duration:

- Five days

Course materials:

- Delegate handbook

Course workshop:

This course makes use of target hardware during the real-time practical exercises. The board targeted is an ARM Cortex-M based MCU which gives attendees a real sense of embedded application development.

Program structure

- Program structure
- Basic processor architecture
- The build process
- Loading code on the target

Formatted output

- Formatted output
- Multiple format specifiers and arguments

Integer types

- Integer types
- Two's-complement representations
- Placement of definition specifies an object's lifetime
- Object visibility – scope

Floating point types

- Floating-point types
- IEEE-754 encoding

Constants

- Constants
- The const qualifier
- Enumerated types

Expressions

- Statements
- Implicit type conversion
- Operator precedence and associativity
- Arithmetic conversions

Control flow

- Equality and Relational Operators
- If statements
- Conditional expression
- switch

Iteration

- while loop
- do-while
- for loop

Pointers

- Finding the address of an object
- Indirect object access via a pointer
- Pointers-to-const and modifiable objects

System startup

- The system power-up state
- Uninitialised statics are set to zero
- Static object initial values

Hardware Manipulation

- Hardware Manipulation
- Memory-mapped registers are accessed via pointers
- volatile objects
- Bitwise Operators

Function basics

- Functions vs Procedures
- Passing parameters
- Calling a function in another source file
- Function prototypes

Arrays

- Array basics
- Named member initialisation
- 'Multi-dimensional' arrays
- Pointers and arrays
- Pointer arithmetic

Structures

- Organising data
- Member initialisation
- The struct in memory
- Pointers to structs

Function parameters

- How function arguments are passed
- ARM, register-based calling convention (AAPCS)
- The overheads of pass-by-value
- Pass-by-reference

Function inlining

- When function calls are expensive
- Macros with parameters
- The do-while(0) idiom
- Function inlining

Modularisation

- Coupling
- Encapsulation
- Cohesion
- Abstraction

Structuring code

- Separating Interface and Implementation
- Compilation Dependencies
- Include guards
- #pragma once

Conditional compilation

- Preprocessor conditionals
- Weak linkage

bit-fields and unions

- Bit-field padding
- Unions bit and byte access
- Hardware overlay of a union

Dynamic memory

- Standard library dynamic allocation
- Defined-Used paths for objects
- Memory leak
- Killed-used problem
- Mismatched-free problem

Interrupts

- Interrupt controllers
- Priorities and masks
- Communicating with the rest of the system
- Atomic operations