Objectives

- Reviewing the C language standard
- Putting in evidence the essential C features used in embedded application
- Discovering the Embedded context through several bare-metal labs running on an QEMU emulated board STM32F4-Discovery (Cortex-M4 core based)
- Discovering the Debug features
- Understanding the different steps of a toolchain
- Understanding how to configure a linker script to place code and data in memory
- Understanding the Cortex-M4 Application level programmers' model
- Reviewing the boot sequence
- Analyzing the compiler optimization and how to write optimized code
- Interfacing C and Assembly
- Learning how to handle interrupts

Labs are conducted on a QEMU-emulated ARM-based board

Prerequisites

- Knowledge of binary arithmetic
- Basic knowledge of embedded processors
- The knowledge of embedded processor philosophy is recommended

Course Environment

- Theoretical course
  - PDF course material (in English).
  - Course dispensed using the Teams video-conferencing system.
  - The trainer answers trainees' questions during the training and provide technical and pedagogical assistance through the Teams video-conferencing system.
- Practical activities
  - Practical activities represent from 40% to 50% of course duration.
  - Code examples, exercises and solutions
  - One Online Linux PC per trainee for the practical activities.
  - The trainer has access to trainees' Online PCs for technical and pedagogical assistance.
  - Eclipse environment and GCC compiler.
  - QEMU Emulated board or physical board connected to the online PC (depending on the course).
  - Some Labs may be completed between sessions and are checked by the trainer on the next session.
- Downloadable preconfigured virtual machine for post-course practical activities
- At the start of each session the trainer will interact with the trainees to ensure the course fits their expectations and correct if needed

Duration

- Total: 24 hours
- 4 sessions, 6 hours each (excluding break time)
- From 40% to 50% of training time is devoted to practical activities
- Some Labs may be completed between sessions and are checked by the trainer on the next session

Target Audience

- Any embedded systems engineer or technician with the above prerequisites.

Evaluation modalities

- The prerequisites indicated above are assessed before the training by the technical supervision of the traineein his company, or by the trainee himself in the exceptional case of an individual trainee.
- Trainee progress is assessed in two different ways, depending on the course:
  - For courses lending themselves to practical exercises, the results of the exercises are checked by the trainer while, if necessary, helping trainees to carry them out by providing additional details.
  - Quizzes are offered at the end of sections that do not include practical exercises to verifythat the trainees have assimilated the points presented
- At the end of the training, each trainee receives a certificate attesting that they have successfully completed the course.
  - In the event of a problem, discovered during the course, due to a lack of prerequisites by the trainee a different or additional training is offered to them, generally to reinforce their prerequisites,in agreement with their company manager if applicable.

Plan

First Session

Analyzing the different toolchain elements

- Using cross compilation
- Compiler, Assembler and Linker Purpose
- C source program structure
- Preprocessor
  - Using cross compilation
  - Compiler, Assembler and Linker Purpose
  - C source program structure
  - Preprocessor
- Reviewing the different object file sections
- Library inclusion
- Startup file
- GCC compiler options
- Configuring the linker to place code and data in the memory, executing code from RAM
- Makefile

Exercise: *Following the different build steps of a simple program*
Exercise: *Working with Conditional Compilation*
Exercise: *Working with the linker, placing code and data in the memory*

Lab Environment

- Creating a project from scratch
- Communicating with the Target
- Debugger Windows : Source (C and Disassembly), Memory, Stack, Variables, Registers
- Breakpoints

Types and Operators (1st part)

- Variable storage class (static, automatic, register et extern) with their location and lifetime
- Local and global variable declaration
- Scalar types (char, halfword, int, float and double)
- Constants
- Strings

Second Session

Types and Operators (2nd part)

- Variable storage class (static, automatic, register et extern) with their location and lifetime
- Type conversion, casting
- The volatile attribute
- C operators (logical, arithmetical and relational)
- Operator priority

Exercise: *Working with types and operators*

Control structures

- If/else structure
- Switch/case structure
- While, do/while and for loopf
- Break, continue and go instruction

Exercise: *Working with pointers*

Pointers and Arrays

- Pointer definition
- Pointer Initialization, pointer access, pointer operations
- Constant and volatile pointer
- The restrict attribute
- One- and Multi-dimensional arrays
- Array initialization, array access, array operations
- Pointer array

Exercise: *Working with pointers*
Exercise: *Working with arrays*

Structures and unions

- Structure variable declaration
- Structure variable pointer declaration
- Structure field access
- Padding, #pragma pack compilation directive
- Big and little endian format
- Bit field structure declaration
- Modeling peripheral register
- Structure array
- Typedef type
- Enum type
- Union declaration
- Union intitialization and operation

Third Session

Functions

- Function prototype (arguments, return value)
- Function definition and declaration
- Function visibility
- Function pointer
- Function call
- Passing parameter
- Stack operation

- Stack frame, call stack
- The recursivity and the stack
- Macro vs function
- Pipeline and branch
- Function inlining
- Interfacing C and Assembler

Exercise: *Passing parameter to function*
Exercise: *Analyzing the stack utilization*

Standard library Overview

- Stdio library
- Getchar and putchar functions
- Memcpy function
- Printf and scanf functions
- File access function review

Data structures

- Programming FIFOs
- Programming Linked list (simple and double)

Exercise: *Working with linked list*

Dynamic allocation

- Dynamic allocation functions: malloc, free function
- Sizeof operator
- Dynamic memory allocation vs static memory allocation
- Stack vs Heap
- Memory management algorithms overview
  - Buddy System
  - Best fit / First Fit
  - Pools Management
- Memory management errors

Exercise: *Using dynamic allocation*

Fourth Session

Embedded Context

- Peripheral Programming
- Peripheral register access and Memory access
- Signed vs unsigned
- Memory latency
- Cache
- Synchronization
- Interruption necessity in an embedded context
  - Tail-Chaining
  - Pre-emption (Nesting)
  - NVIC Integrated Interrupt Controller
  - Exception Priority Management
- Level and pulse triggered interrupts
- Interrupt clearance
- Interrupt handler writing
- Vector table
- Vector installation
- System Timer (Systick Timer)
- Clarifying the boot sequence
- Debug Interface Ovevriew

Exercise: *Interrupt Management*

Compiler Hints and Tips for Cortex-M

- Compiler optimizations
- Mixing C and Assembly
- AAPCS
- Function inlining
- Unaligned Accesses, padding
- Local and global data issues, alignment, Structure

Renseignements pratiques

Inquiry : 24 hours