Learning how to program a Microcontroller (especially the Cortex-M based ones)

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 STM32F4 MCU (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
- Getting an Overview on STM32F4 Architecture
- 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
- Programming a serial communication
- Configuring DMA transfers
- Working with an ADC

Course environment

- Convenient course material with space for taking notes
- Example code, labs and solutions
- A STM32F2 (Cortex-M3) Board for Labs
- A PC for two trainees with the free System Workbench for STM32 IDE to flash and debug applications

Prerequisites

- Basic knowledge of C language (or another low-level language)
- Knowledge of binary arithmetic
- The knowledge of embedded processor philosophy is recommended

Environnement du cours

- Cours théorique
  - Support de cours imprimé et au format PDF (en anglais).
  - Le formateur répond aux questions des stagiaires en direct pendant la formation et fournit une assistance technique et pédagogique.
- Activités pratiques
  - Les activités pratiques représentent de 40% à 50% de la durée du cours.
  - Elles permettent de valider ou compléter les connaissances acquises pendant le cours théorique.
  - Exemples de code, exercices et solutions
  - Un PC (Linux ou Windows) par binôme de stagiaires (si plus de 6 stagiaires) pour les activités pratiques avec, si approprié, une carte cible embarquée.
  - Le formateur accède aux PC des stagiaires pour l'assistance technique et pédagogique.

- Une machine virtuelle préconfigurée téléchargeable pour refaire les activités pratiques après le cours
- Au début de chaque demi-journée une période est réservée à une interaction avec les stagiaires pour s'assurer que le cours répond à leurs attentes et l'adapter si nécessaire

Audience visée

- Tout ingénieur ou technicien en systèmes embarqués possédant les prérequis ci-dessus.

Modalités d'évaluation

- Les prérequis indiqués ci-dessus sont évalués avant la formation par l'encadrement technique du stagiaire dans son entreprise, ou par le stagiaire lui-même dans le cas exceptionnel d'un stagiaire individuel.
- Les progrès des stagiaires sont évalués de deux façons différentes, suivant le cours:
  - Pour les cours se prêtant à des exercices pratiques, les résultats des exercices sont vérifiés par le formateur, qui aide si nécessaire les stagiaires à les réaliser en apportant des précisions supplémentaires.
  - Des quizz sont proposés en fin des sections ne comportant pas d'exercices pratiques pour vérifier que les stagiaires ont assimilé les points présentés
- En fin de formation, chaque stagiaire reçoit une attestation et un certificat attestant qu'il a suivi le cours avec succès.
  - En cas de problème dû à un manque de prérequis de la part du stagiaire, constaté lors de la formation, une formation différente ou complémentaire lui est proposée, en général pour conforter ses prérequis, en accord avec son responsable en entreprise le cas échéant.

Plan

First Day

Analyzing the different Toolchain elements

- Using cross compilation
- Compiler, Assembler and Linker Purpose
- C source program structure
- Preprocessor
  - #define, #include instructions
  - Writing Macros with precaution
  - Writing Header files with precaution
- 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
- Debug probe
- Communicating with the Target
- Debugger Windows : Source (C and Disassembly), Memory, Stack, Variables, Registers
- Breakpoints

Types and Operators

- 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
- Type conversion, casting
- The volatile attribute
- C operators (logical, arithmetical and relational)
- Operator priority

Exercise : *Working with types and operators*

Second Day

Control structures

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

Exercise : *Working with control structures*

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

Exercise : *Modeling a STM32F2 timer to program it*

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

Third Day

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*

Embedded Context

- Peripheral Programming
- Peripheral register access and Memory access
- Signed vs unsigned
- Memory latency
- Cache
- Synchronization
- Interruption necessity in an embedded context
- Level and pulse triggered interrupts
- Interrupt clearance
- Interrupt handler writing
- Vector table
- Vector installation

Cortex-M Architecture Overview

- V7-M Architecture Overview
- Harvard Architecture, I-Code, D-Code and System Bus
- Registers (Two stacks pointers)
- States
- Different Running-modes and Privileged Levels
- System Control Block
- Systick Timer

- Alignment and Endianness
- CMSIS Library
- Exception / Interrupt Mechanism Overview
- Vector Table
- Interrut entry and return Overview
  - Tail-Chaining
  - Pre-emption (Nesting)
  - NVIC Integrated Interrupt Controller
  - Exception Priority Management
- Debug Interface Ovevriew
- Clarifying the boot sequence

Exercise : *Timer Interrupt Management*

Fourth Day

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

Exercise : *Interfacing C and Assembler*
Exercise : *Demo: Long Branch, Function inlining, padding*

STM32F4 MCUs Architecture Overview

- ARM core based architecture
- Description of STM32F407X SoC architecture
- Clarifying the internal data and instruction paths: Bus Matrix, AHB-lite interconnect, peripheral buses, AHB-to-APB bridges, DMAs
- Memory Organization
  - Flash memory read interface
  - Adaptive Real-Time memory accelerator, instruction prefetch queue and branch cache
  - Sector and mass erase
  - Internal SRAMs
  - Concurrent access to 112 KB and 16 KB blocks
- SoC mapping
- Flash Programming methods
- Boot Configuration
- Power, Reset and Clocking Overview
- DMA Overview
- UART Overview

Exercise : *Get Metrics for a data copy using the DMA or not*
Exercise : *Programming the UART to redirect the printf on the serial port*
Exercise : *Displaying ADC value on LCD*

Renseignements pratiques

Renseignements : 4 jours