oL2 - Langage C pour les MCUs embarqués

Objectifs

- Aperçu du standard du langage C
- Mettre en évidence les fonctionnalités essentielles du langage C utilisées dans les applications embarquées
- Découvrir le contexte de l'embarqué au travers de plusieurs tests bare-metal
- Découvrir les fonctionnalités de débogage
- Comprendre les différentes étapes d'une chaîne de compilation
- Comprendre comment configurer un script de l'éditeur de lien pour placer le code et les données en mémoire
- Comprendre le modèle des programmeurs du Cortex-M
- Passer en revue la séquence de démarrage
- Analyser l'optimisation du compilateur et comment écrire un code optimisé
- Interfaçage et Assemblage du C
- Apprendre à gérer les interruptions

Les travaux pratiques seront effectués sur une carte basée sur ARM Cortex-M et émulée par QEMU

Pré-requis

- Connaissance de l'arithmétique binaire
- Connaissance de base des processeurs embarqués
- La connaissance de la philosophie des processeurs embarqués est recommandée

Environnement du cours

- Cours théorique
  - Support de cours au format PDF (en anglais).
  - Cours dispensé via le système de visioconférence Teams.
  - 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 en ligne par stagiaire pour les activités pratiques.
  - Le formateur a accès aux PC en ligne des stagiaires pour l'assistance technique et pédagogique.
  - Certains travaux pratiques peuvent être réalisés entre les sessions et sont vérifiés par le formateur lors de la session suivante.
- Une machine virtuelle préconfigurée téléchargeable pour refaire les activités pratiques après le cours
- Au début de chaque session 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.

Durée

- Total : 24 heures
- 4 sessions de 6 heures chacune (hors temps de pause)
- De 40% à 50% du temps de formation est consacré aux activités pratiques

- Certains travaux pratiques peuvent être réalisés entre les sessions et sont vérifiés par le formateur lors de la session suivante

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

Première 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

Deuxième 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

Troisième 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*

Quatrième 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

Durée : **24 heures**
Prix : **2360 € HT**
Prochaines sessions : **du 28 au 31 mai 2024 - Online EurAsia (9h-16h CET)**