L2 - Le langage C pour les microcontrôleurs embarqués

Apprendre à programmer un microcontrôleur (en particulier ceux à base de Cortex-M)

Objectifs

- Réviser le standard du langage C
- Mettre en évidence les caractéristiques essentielles du C utilisées dans les applications embarquées
- Découvrir le contexte embarqué au travers de plusieurs exercices bare-metal tournant sur un MCU STM32F4 (à base de cœur Cortex-M4)
- Découvrir les fonctions de débogage
- Comprendre les différentes étapes d'une toolchain
- Comprendre comment configurer un script de linker pour placer le code et les données en mémoire
- Obtenir une vue d'ensemble de l'architecture du STM32F4
- Comprendre le modèle de programmation applicatif du Cortex-M4
- Passer en revue la séquence de démarrage
- Analyser les optimisations du compilateur et apprendre à écrire du code optimisé
- Interfacer le C et l'assembleur
- Apprendre à traiter les interruptions
- Programmer une communication série
- Configurer des transferts DMA
- Travailler avec un ADC

Environnement du cours

- Un support de cours pratique avec de la place pour prendre des notes
- Du code d'exemple, des exercices et leurs corrigés
- Une carte STM32F2 (Cortex-M3) pour les exercices
- Un PC pour deux stagiaires avec l'IDE gratuit System Workbench for STM32 pour flasher et déboguer les applications

Prérequis

- Connaissance de base du langage C (ou d'un autre langage de bas niveau)
- Connaissance de l'arithmétique binaire
- La connaissance de la philosophie des processeurs embarqués est recommandée

Audience visée

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

Plan du cours

Premier jour

Analyser les différents éléments de la toolchain

- Utiliser la compilation croisée
- Rôle du compilateur, de l'assembleur et du linker
- Structure d'un programme source C
- Le préprocesseur
  - Instructions #define, #include

    - Écrire des macros avec précaution
    - Écrire des fichiers d'en-tête avec précaution
- Passer en revue les différentes sections d'un fichier objet
- Inclusion de bibliothèques
- Fichier de démarrage
- Options du compilateur GCC
- Configurer le linker pour placer le code et les données en mémoire, exécuter du code depuis la RAM
- Makefile

Environnement des exercices

- Créer un projet de zéro
- Sonde de débogage
- Communiquer avec la cible
- Fenêtres du débogueur : source (C et désassemblage), mémoire, pile, variables, registres
- Points d'arrêt

Types et opérateurs

- Classe de stockage des variables (static, automatic, register et extern) avec leur emplacement et leur durée de vie
- Déclaration de variables locales et globales
- Types scalaires (char, halfword, int, float et double)
- Constantes
- Chaînes de caractères
- Conversion de type, transtypage
- L'attribut volatile
- Les opérateurs du C (logiques, arithmétiques et relationnels)
- Priorité des opérateurs

Deuxième jour

Structures de contrôle

- La structure if/else
- La structure switch/case
- Les boucles while, do/while et for
- Les instructions break, continue et goto

Pointeurs et tableaux

- Définition d'un pointeur
- Initialisation d'un pointeur, accès par pointeur, opérations sur les pointeurs
- Pointeur constant et pointeur volatile
- L'attribut restrict
- Tableaux à une et à plusieurs dimensions
- Initialisation d'un tableau, accès à un tableau, opérations sur les tableaux
- Tableau de pointeurs

Structures et unions

- Déclaration d'une variable de type structure
- Déclaration d'un pointeur sur une structure
- Accès aux champs d'une structure
- Padding, directive de compilation #pragma pack
- Formats big et little endian
- Déclaration d'une structure à champs de bits
- Modéliser un registre de périphérique

- Tableau de structures
- Le type typedef
- Le type enum
- Déclaration d'une union
- Initialisation et manipulation d'une union

Fonctions

- Prototype de fonction (arguments, valeur de retour)
- Définition et déclaration d'une fonction
- Visibilité d'une fonction
- Pointeur de fonction
- Appel de fonction
- Passage de paramètres
- Fonctionnement de la pile
- Stack frame, pile d'appels
- La récursivité et la pile
- Macro ou fonction
- Pipeline et branchement
- Inlining de fonction
- Interfacer le C et l'assembleur

Présentation de la bibliothèque standard

- La bibliothèque stdio
- Les fonctions getchar et putchar
- La fonction memcpy
- Les fonctions printf et scanf
- Revue des fonctions d'accès aux fichiers

Troisième jour

Structures de données

- Programmer des FIFO
- Programmer des listes chaînées (simples et doubles)

Allocation dynamique

- Fonctions d'allocation dynamique : fonctions malloc et free
- L'opérateur sizeof
- Allocation dynamique de mémoire et allocation statique
- Pile et tas
- Présentation des algorithmes de gestion mémoire
  - Buddy System
  - Best fit / First Fit
  - Gestion par pools
- Erreurs de gestion mémoire

Le contexte embarqué

- Programmation des périphériques
- Accès aux registres de périphériques et accès mémoire
- Signé ou non signé
- Latence mémoire
- Cache
- Synchronisation

- Nécessité des interruptions dans un contexte embarqué
- Interruptions déclenchées sur niveau et sur impulsion
- Acquittement des interruptions
- Écriture d'un gestionnaire d'interruption
- Table des vecteurs
- Installation d'un vecteur

Présentation de l'architecture Cortex-M

- Présentation de l'architecture V7-M
- Architecture Harvard, bus I-Code, D-Code et System
- Registres (deux pointeurs de pile)
- États
- Les différents modes d'exécution et niveaux de privilège
- System Control Block
- Timer Systick
- Alignement et boutisme
- La bibliothèque CMSIS
- Présentation du mécanisme d'exception / interruption
- Table des vecteurs
- Présentation de l'entrée et du retour d'interruption
  - Tail-Chaining
  - Préemption (imbrication)
  - Contrôleur d'interruptions intégré NVIC
  - Gestion de la priorité des exceptions
- Présentation de l'interface de débogage
- Clarification de la séquence de démarrage

Quatrième jour

Conseils et astuces de compilation pour le Cortex-M

- Optimisations du compilateur
- Mélanger C et assembleur
- AAPCS
- Inlining de fonction
- Accès non alignés, padding
- Problèmes liés aux données locales et globales, alignement, structures

Présentation de l'architecture des MCU STM32F4

- Architecture à base de cœur ARM
- Description de l'architecture du SoC STM32F407X
- Clarification des chemins internes de données et d'instructions : Bus Matrix, interconnexion AHB-lite, bus périphériques, ponts AHB vers APB, DMA
- Organisation de la mémoire
  - Interface de lecture de la mémoire Flash
  - Accélérateur mémoire temps réel adaptatif, file de préchargement d'instructions et cache de branchement
  - Effacement par secteur et effacement total
  - SRAM internes
  - Accès concurrents aux blocs de 112 Ko et 16 Ko
- Cartographie du SoC
- Méthodes de programmation de la Flash
- Configuration du démarrage
- Présentation de l'alimentation, du reset et des horloges
- Présentation du DMA
- Présentation de l'UART