Haute école d'ingénierie et d'architecture Fribourg
Hochschule für Technik und Architektur Freiburg

Systèmes Embarqués 1 & 2

tp.08 - Interruptions et exceptions

Classes T-2/I-2 // 2016-2017

Daniel Gachet | HEIA-FR/TIC

tp.08 | 03.03.2017

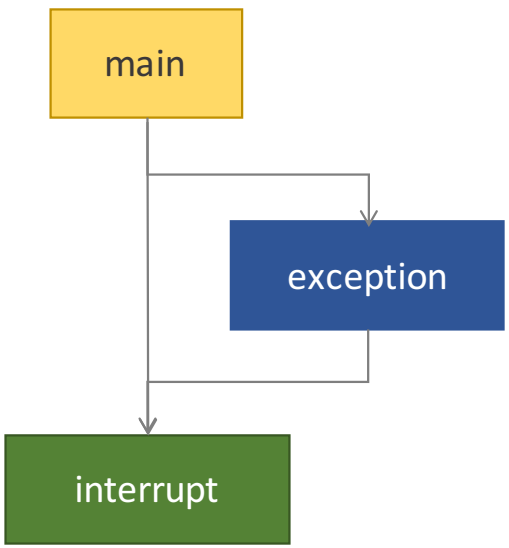
Hes·so
Haute Ecole Spécialisée
de Suisse occidentale
Fachhochschule Westschweiz

Objectifs

- A la fin du laboratoire, les étudiant-e-s seront capables de
  - Décrire le traitement des interruptions et exceptions des µP ARM
  - Concevoir et développer une petite application mixte C - assembleur permettant de traiter les interruptions et exceptions de bas niveau d'un µP ARM Cortex-A8
  - Débogguer une application mixte C - assembleur traitant des interruptions
  - Etudier le datasheet du µP ARM Cortex-A8 et du µP TI AM335x
- Durée
  - 1 séance de laboratoire (4 heures)
- Rapport
  - Rapport de laboratoire avec le code source

Travail à réaliser

- Développement d'une bibliothèque pour le traitement des interruptions et des exceptions d'un µP ARM Cortex-A8
- Integration de la bibliothèque dans un programme principale pour sa validation



- Le module "`interrupt`" servira au traitement de bas niveau des interruptions
- Le module "`exception`" permettra de traiter les exceptions

- L'interface du module "interrupt" fournira les services suivants
  - "interrupt_init()" pour initialiser le module et configurer le µP pour le traitement de tous les vecteurs d'interruptions et exceptions disponible sur le µP
  - "interrupt_attach" pour attacher une routine de traitement des interruptions à un vecteur donné. Cette méthode permettra de spécifier un paramètre, lequel sera passé comme argument avec l'identifiant du vecteur de l'interruption lors de l'appel de la méthode de traitement. Le service ne permettra d'attacher qu'une seule méthode à un vecteur d'interruption. En cas d'erreur la valeur -1 sera retournée, en case de succès 0.
  - "interrupt_detach" pour détacher la routine de traitement du vecteur d'interruption
  - "interrupt_enable" pour autoriser les interruptions matérielles
  - "interrupt_disable" pour bloquer les interruptions matérielles

- L'interface du module "`exception`" ne fournira qu'un service "`exception_init()`" pour initialiser le module et attacher une routine de traitement par défaut aux vecteurs correspondants.
- Le module "`exception`" n'implémentera qu'une seule routine générique de traitement. Cette dernière affichera un message approprié sur la console de la cible lorsqu'une exception est levée.
- Le programme principal "`main`" orchestrera l'ensemble de l'application. Il implémentera quelques instructions pour la génération d'exceptions.

- Pour télécharger le squelette du projet du dépôt centralisé

```
$ cd ~/workspace/se12/tp
$ git pull upstream master
```

- Pour mettre à jour les paths des includes dans eclipse
  - ouvrir `Properties` pour votre projet
  - aller `C/C++ General` → `Paths and Symbols`
  - ouvrir `Includes` → `GNU C`
  - ajouter `/home/lmi/workspace/se12/bbb/source`
- Le code et le rapport seront rendus au travers du dépôt Git centralisé
  - *sources* : *.../tp/tp.08*
  - *rapport* : *.../tp/tp.08/doc/report.pdf*
- Délai
  - Le journal et le code doivent être rendus au plus tard 5 jours après le TP à 23h59

Table des vecteurs

- Le µP ARM Cortex-A8 permet de choisir librement l'emplacement de la table des vecteurs
- Le co-processeur P15 offre un service pour spécifier l'adresse de la table des vecteurs

```
ldr   r0, =AM335X_VECTOR_BASE_ADDR
mcr   p15, #0, r0, c12, c0, #0
```

- Option
  Le µP TI AM335x dispose de plusieurs mémoires SRAM directement dans son chip. La mémoire L3 OCMC de 64KiB située à l'adresse 0x4030'0000 est parfaitement adaptée pour stocker
  - la table des vecteurs
  - les routines de traitement des interruptions de bas niveau
  - les piles pour les différents modes d'opération du µP

[0] 02_cortex_a8_technical_reference_manual, chapter 3.2.68

Traitement de l'interruption