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

Alan Sueur
Jonathan Rial
Classe I-2a
Systèmes Embarqués
22.12.16

TP05 - Pilote d'un timer du AM335x

Réponses aux questions

Quelle est la signification du qualificatif *volatile* ?

Il dit au compilateur de ne pas optimiser.
Par exemple une variable some_int utilisée dans une boucle while(some_int) et qui n'est jamais modifiée par le programme, sera optimisée par le compilateur en while(true) et ce n'est peut-être pas le comportement que nous souhaitons. Pour pallier à ce problème, il suffit de mettre le qualificatif volatile.

Quelle est l'utilité du qualificatif volatile, quand il est associé à un pointeur ?

Il permet de signifier que la valeur pointée par le pointeur peut changer de façon inattendue et donc ceci demande au compilateur de ne pas optimiser l'accès à cette information et à ne pas la mettre en cache. Ceci est particulièrement utile si nous utilisons des informations générées par du matériel.

Comment peut-on efficacement définir des registres et leur contenu en c ?

Grâce à une structure. Nous l'avons fait pour le dmtimer1 :

```
struct dmtimer1_regs {
   uint32_t res1[3];
   uint32_t tidr;
   uint32_t tiocp_cfg;
   uint32_t tistat;
   ...
   uint32_t towr;
};
```

Comment peut-on accéder aux registres d'un contrôleur de périphérique situés dans l'espace d'adressage du μP ?

Grâce à un pointeur vers une adresse. Voici le pointeur que nous avons utilisé durant le TP :

```
static volatile struct dmtimer1_regs* timer1 = (struct dmtimer1_regs*) 0x44e31000;
```

Alan Sueur
Jonathan Rial
Classe I-2a

Comment sont placés les attributs d'une structure dans la mémoire ?

Ils sont alignés sur un multiple de la taille de la donnée la plus grande de la structure. Par exemple la structure suivante :

```
struct ma_struct {
    char a;
    float x;
    char b;
    float y;
    char c;
    float z;
    char d;
};
```

Est alignée comme ceci en mémoire :

Adresse	*Octet 0*	*Octet 1*	*Octet 2*	*Octet 3*
n	char a			
n+4	double x			
n+8	char b			
n+12	double y			
n+16	char c			
n+20	double z			
n+24	char d			

FIGURE 1 – Alignement en mémoire

Source : http ://ilay.org/yann/articles/mem

Quelle est l'utilité des conversions de type (type casting) en C ?

Comme dans les autres langages, il permet de changer le type d'une donnée dans un autre type. Ceci permet également de changer la taille en mémoire des données. Par exemple si nous voulons stocker un « long » en entier, nous pouvons faire une conversion de type de long à int.

Comment réalise-t-on un "type cast" en C ?

La syntaxe pour réaliser un « type cast » est la suivante :
$(type_name)expression$
Par exemple :

```
int sum = 17;
int count = 5;
double result;
result = (double) sum / count;
```

Alan Sueur
Jonathan Rial
Classe I-2a
Systèmes Embarqués
22.12.16

Heures de travail

En dehors des heures de cours, nous avons dû travailler 7h20. Durant le cours nous avons pu réaliser les tâches suivantes :

— Implémenter le code pour faire fonctionner les boutons.
— Implémenter le code pour faire fonctionner les leds.
— Une idée d'implémentation globale du projet et comment faire fonctionner le timer.

A la maison, nous avons dû finir les tâches suivantes :

— Répondre aux questions 1-7. [0h50]
— Comment fonctionne le timer, lire la doc et implémentation [∼1h00]
— Création du main [∼3h00]
— Correction de bugs, testing du projet [2h10]
— Rédiger ce rapport. [0h20]

Synthèse

Non acquis

— La création du timer est très compliquée, malgré les documentations.

Partiellement acquis

— Manque encore de la pratique pour les pointeurs.

Acquis

— Utilisation des différents composants du Beaglebone (leds, boutons, wheel, seg7)
— Utilisation désormais naturelle du code C.
— A l'aise avec l'environnement de travail en général (Eclipse, Git, Beaglebone)
— La recherche d'informations se fait plus naturelle et est plus rapide.
— Le débuggage est désormais plus rapide (dû aux nombreux bugs dont nous avons du faire face...)

Remarques & Feedback

Ce deuxième travail pratique orienté C n'a pas été simple à réalisé, nous avons dû travailler pas mal de temps en dehors de l'école pour faire fonctionner notre projet. Ceci nous a permis d'apprendre beaucoup de choses autour de ce nouveau langage et de nous améliorer dans ce domaine.
Ce tp était très intéressant malgré le timer qui était particulièrement compliqué à implémenter.
Nous avons rencontré un problème lorsque nous appuyions sur le bouton pause car le système passait plusieurs fois dans notre code et cela créait des problèmes. Nous avons réglé ce problème en comparant l'état précédant du système avec l'état présent.