Systèmes Embarqués

Journal - TP.05 : Pilote d'un timer du AM335x

Temps effectué hors des heures de classe

Nous avons effectué 8h de plus en dehors des deux périodes mises à disposition pour ce TP.

Synthèse des acquis

Acquis :
- Base du langage C
- Utilisation des boutons, leds, 7-segments et de la roue crantée en C

Acquis, mais à exercer encore :
- Initialisation et utilisation du timer 1ms en C

Questions

Quelle est la signification du qualificatif volatile ?

Le qualificatif volatile permet d'empêcher le compilateur de procéder aux optimisations qu'il réalise sur les variables normales. Il signifie au compilateur qu'une variable qualifiée de volatile peut être modifiée par un moyen externe au programme, comme un composant matériel. Ce qualificatif est uniquement utilisé lorsque l'on travaille avec des éléments hardware comme le timer.

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

Il est associé à un pointeur lorsqu'il s'agit d'accéder à une ressource matérielle, ou autrement dit un registre agissant sur un composant hardware. Dans le cadre de ce TP, nous l'avons utilisé afin d'accéder aux différents registres liés au timer 1ms de notre BeagleBone.

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

Grâce à l'utilisation d'une structure et d'un pointeur, comme nous l'avons fait durant le TP afin d'initialiser et d'utiliser le timer 1ms.

```
1 struct timer1_ctrl {
2   uint32_t tidr;
3   uint32_t unused[3];
4     // ...
5   uint32_t towr;
6 };
7
8 static volatile struct timer1_ctrl* timer1 = (volatile struct timer1_ctrl*) 0x44e31000;
```

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

En gardant la même initialisation que pour la question précédente, il est possible d'accéder à chaque registre , et d'en modifier le contenu, en utilisant une ligne de code comme ceci :

```
1 timer1 -> tldr = 0;
```

Pour accéder à un autre registre, enregistré dans la structure, il suffit de le renseigner après la ->.

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

Les éléments d'une structure sont placés les un après les autres dans la mémoire.

```
1 struct myStruc {
2 long a;
3 long b;
4 short c;
5 char[6] d;
6 };
```

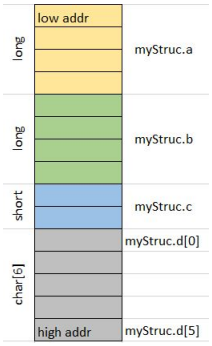

FIGURE 1 – Exemple d'une structure en mémoire

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

Une conversion de type permet de convertir une variable d'un type, par exemple un *integer*, vers un autre type, par exemple en *float* ou en *char*. Cette conversion a pour but de profiter de fonctions qui s'appliqueraient à un autre type ou d'obtenir la précision souhaitée lors de certaines opérations, par exemple si nous divisons 2 integer et que nous souhaiterions avoir le résultat exact et non un résultat tronqué.

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

Il existe deux "type cast" possibles, auxquels il faut penser lorsque l'on procède à une conversion : la conversion implicite ou explicite. La conversion implicite se réalise sans opération particulière et s'utilise lors de l'évaluation de type de base du langage. La conversion explicite est comme son nom l'indique explicitement réalisé par le programmeur et nécessite l'utilisation d'une instruction comme celle-ci :

```
1 type1 variable;
2 type2 result = (type2) variable;
```

Voici un exemple un peu plus concret :

```
1 double da = 3.3;
2 double db = 3.3;
3 int result = (int)da + (int)db;                    // result = 6
```

Remarques

Le timer peut fonctionner selon diverses fréquences d'entrée. Il est nécessaire d'indiquer laquelle doit être redirigé sur son entré clock à l'aide du registre PRCM. Pour utiliser l'horloge 24MHz, ajouter cette ligne dans l'initialisation :

```
1 am335x_clock_enable_timer_module(AM335X_CLOCK_TIMER1)
```

Feedback

Il était très intéressant de faire un projet un peut plus conséquent et d'avoir plusieurs TP pour le réaliser. Le temps supplémentaire comparé à un TP sur 11 heures nous a permis de mieux comprendre le fonctionnement du hardware et de tester diverses variantes de code afin de tester les fonctionnalités du BeagleBone.