Charlotte Junod et Nicolas Fuchs
charlotte.junod@edu.hefr.ch et nicolas.fuchs@edu.hefr.ch

Systèmes embarqués I – journal

TP05

Heures de travail en dehors des cours : 3

Feedback :

Dans ce TP, nous avons créé un pilote qui utilise plusieurs périphériques du Beaglebone. Nous avons ainsi agrandi nos connaissances sur la gestion de ce périphériques ainsi que sur les structures et pointeurs C. Nous avons eu de la peine à réaliser le pilote du timer (heureusement nous avons pu demander de l'aide à nos camarades). Par contre, le reste de l'implémentation n'a pas posé trop de souci malgré quelques difficultés avec la détection du bouton de la roue encodeuse.

Questions :

Quelle est la signification du qualificatif volatile ?

Avec le qualificatif volatile on s'assure que chaque opération sur la variable écrite dans le code est bien effectuée par le code compilé et que le compilateur ne doit pas « optimiser » les opérations sur la variable, c'est-à-dire qu'à chaque fois qu'on appelle cette variable, il ira effectivement chercher sa valeur dans la mémoire.

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

Cela permet de reporter la propriété « volatile » sur un pointeur qui désignerait une variable volatile. Si on ne le fait pas, les opérations faites avec le pointeur peuvent être optimisées par le compilateur.

Ce qualificatif est utile en programmation concurrente, lorsque plusieurs processus accèdent à la même ressource.

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

Avec un pointeur sur une structure contenant des variables dont le type est de même longueur qu'un registre (par exemple uint32_t pour des registres de 32b).

Si on attribue au pointeur l'adresse de départ de la banque de registres qu'on veut utiliser, on peut alors par la suite accéder à ces registres avec les champs de notre structure.

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

En se servant de la méthode décrite à la question précédente et en donnant l'adresse des registres de notre périphérique au pointeur.

Charlotte Junod et Nicolas Fuchs
charlotte.junod@edu.hefr.ch et nicolas.fuchs@edu.hefr.ch

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

Ils sont placés directement à la suite selon leur ordre de déclaration.

Etant donné que l'on doit travailler avec des pointeurs, l'adresse en mémoire est importante. Par exemple, dans notre TP, nous avons dû placer des espaces de mémoire occupée pour placer les pointeurs aux bons endroits.

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

Cela est utile si on a une donnée dans un certain format et qu'on veut changer, par exemple pour la donner en paramètre d'une fonction qui demande un certain type. Par exemple, on peut avoir une fonction qui demande un int. Si on a un uint à disposition, il faudra le caster en int avant de le donner à notre fonction.

Il existe 2 types de de (type casting) : implicite et explicite.
Le casting implicite est utilisé par exemple dans des calculs pour transformer des double ou char en int. Il se fait automatiquement.
Le casting explicite permet de voir une source sous différents types. Ce type de casting est voulu et écrit par le programmeur.

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

Il suffit de d'ajouter entre parenthèses le type dans lequel on veut caster avant le nom de la variable qu'on veut caster, par exemple :

```
float x=1.0 ;

double y=(double)x ;
```