



## Systèmes embarqués I – journal

### TP06

**Heures de travail en dehors des cours : 2<sup>1</sup>/<sub>2</sub>**

#### Feedback :

Dans ce TP, nous avons créé un pilote qui contrôle le module température du Beaglebone. Les boutons permettent de définir les registres haut et bas. Nous avons pu compléter nos connaissances sur la gestion des périphériques du Beaglebone ainsi que sur les structures et pointeurs C. Cependant, le codage du fichier epwm1.c n'a pas été facile; la documentation est difficile à comprendre (heureusement nous avons pu demander de l'aide à nos camarades).

#### Questions :

**Quelle est la fonction des 5 registres internes du thermomètre TMP102 ?**

Pointer Register :	Pointeur de 8 bits qui sert à pointer sur le registre (data) sur lequel on va lire ou écrire.
Temperature Register :	Registre de 12/13 bits qui permet de lire la température
Configuration Register :	Registre de 16 bits (lecture et écriture) qui stocke des bits pour le contrôle des modes d'opérations du capteur thermique.
T <sub>LOW</sub> Register :	Registre de 16 bits qui stocke la valeur de la limite de température inférieure.
T <sub>HIGH</sub> Register :	Registre de 16 bits qui stocke la valeur de la limite de température supérieure.

**Comment le TMP102 génère-t-il le signal d'alarme ALERT ?**

Le master envoie une commande d'alerte de 8 bits sur le bus. Si la pin ALERT est active, le périphérique répond en envoyant son adresse sur la ligne SDA. Le huitième bit (LSB) de cette adresse indique si l'alerte est due au fait que la température est au-dessus de T<sub>HIGH</sub> au-dessous de T<sub>LOW</sub>. (low si temp.  $\geq T_{HIGH}$ , high si temp.  $\leq T_{LOW}$ ).

**Pour quelle raison le TMP102 dispose d'un registre T<sub>HIGH</sub> et T<sub>LOW</sub> ?**

Lorsque la température dépasse le seuil fixé, le signal d'alarme est déclenché. Si on avait un seul seuil au lieu de deux et que le signal de température oscillait autour de ce seuil, alors l'alarme s'allumerait et s'éteindrait en boucle.

En utilisant deux seuils au lieu d'un, on évite cette oscillation du signal d'alarme. Il faut que la température descende en dessous de T<sub>LOW</sub> pour désactiver l'alarme et qu'elle remonte au-dessus de T<sub>HIGH</sub> pour réactiver l'alarme.



## **Quels sont les domaines d'application des pointeurs de fonctions ?**

L'appel de différentes fonctions est beaucoup plus flexible et modulable. C'est très utile spéciale pour l'appel de fonctions "mises" dans un tableau. On les utilise aussi très souvent pour concevoir des interfaces avec des fonctions de rappel (appelé aussi Callback). On arrive également avec ces pointeurs de fonctions à concevoir des algorithmes génériques pour les traitements de données.

## **Comment déclare-t-on un pointeur de fonction ?**

On peut le faire avec un typedef :

```
typedef return_type (*pointer_type) (function_parameters);
```

ou bien sans :

```
return_type (*pointer_name) (function_parameters);
```

## **Comment utilise-t-on un pointeur de fonction ?**

```
pointer_type pointer_name = &function_name; (le caractère '&' n'est pas obligatoire)
```

La fonction peut retourner une donnée :

```
type value = (*pointer_name) (function_parameters);  
(l'étoile et les parenthèses du pointeur sont pas obligatoires)
```

La fonction peut également ne rien retourner (void) :

```
(*pointer_name) (function_parameters);
```

## **Comment le compilateur implémente-t-il un pointeur de fonction en assembleur ?**

Ce sont des adresses mémoires. Quand on utilise notre pointeur de fonction, on est redirigé vers l'adresse où la fonction est effectivement stockée.