FINAL SM 57 - Printemps 2014

Nom :

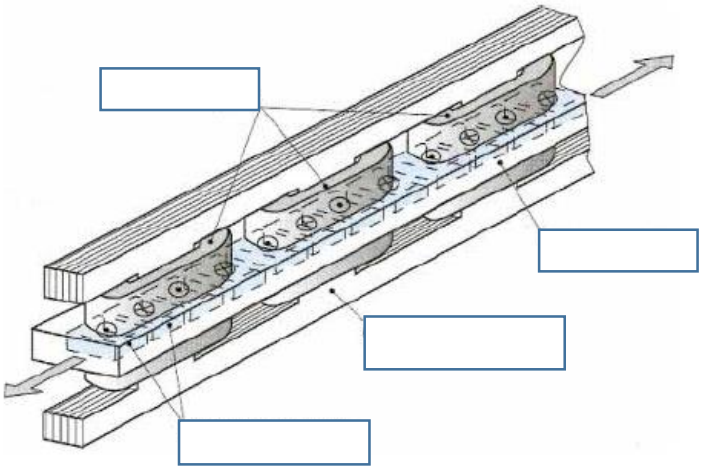
Aucun document autorisé

Prénom :

Calculatrice autorisée

I. Questions de cours

1. Expliquer le principe de fonctionnement d'un moteur à courant continu sans balai (BLDC) ?
2. Quel type de capteur est utilisé pour la commande d'un moteur BLDC? Comment est réalisé un asservissement en vitesse avec ce moteur ? Justifier la question par un schéma.
3. Compléter le schéma ci-dessous :



4. A partir de la figure précédente : Quel est le type de cet actionneur ? Expliquer son principe de fonctionnement.
5. Cochez la ou les bonnes réponses :
   Un moteur à rotor sans pièce polaire :
   - ☐ Dispose d'aimants dans son rotor
   - ☐ Ne dispose pas d'aimants dans son rotor
   - ☐ Le rotor génère un champ magnétique dans l'entrefer
   - ☐ Le rotor ne génère pas de champ magnétique dans l'entrefer
   - ☐ Ne tourne que dans un seul sens de rotation

6. Quelle est la différence entre une pile et un accumulateur ?
7. Donner une définition de l'électrolyte
8. Quels sont les avantages et inconvénients d'une batterie Lithium-Ion ?
9. Tracer l'allure de la courbe de décharge d'une batterie au Plomb (Ne pas oublier les légendes des axes !).
10. Classer par ordre croissant les accumulateurs suivant en fonction de leur énergie spécifique (Wh/kg) : Li-ion / Ni-Cd / Li-Phosphate / Plomb / Ni-Mh / Li-Polymère

11. Question bonus : (+0,5 points non comptabilisé dans le barème) : Qui a découvert l'accumulateur au nickel-cadmium ? Quel société à-il fondé?

II. Problème : Etude d'un bloc de climatisation

Un véhicule est équipé d'un bloc de climatisation pour le confort des passagers. Le bloc se décompose à partir des éléments suivants:

- Capteur de température relevant de manière analogique la température de l'habitacle ;
- Interface utilisateur avec écran LCD affichant la température désirée ainsi que des boutons poussoir pour régler la température ;
- Calculateur basé sur un microcontrôleur DSPic30F4011 utilisant le système d'exploitation PICOS, gérant les entrées capteurs et boutons poussoir, réalisant l'asservissement et le pilotage du bloc de refroidissement en fonction de la température désirée

Le circuit électronique de l'ECU est composé d'un dsPIC30F4011 utilisant le système d'exploitation PICOS comme suit :

- La tâche 0 est utilisée pour la lecture du capteur de température
- La tâche 1 est utilisée pour l'asservissement du bloc de refroidissement
- La tâche 2 est utilisée pour incrémenté la température désirée
- La tâche 3 est utilisée pour décrémenté la température désirée
- La tâche 4 est utilisée pour affiché la température désirée sur l'écran LCD

2.1 Le Timer 1 est utilisé pour définir la base de temps du compteur interne Counter_kernel de PICOS. Déterminer la valeur à placer dans le registre de période PR1 pour fixer la base de temps à 1 ms dans le cas où Fquartz = 20MHz, PLL = 8x, Prescaler = 1 :8

2.2 Compléter le code permettant de déclarer l'alarme permettant la lecture du capteur de température dans TASK0. Déclarer l'évènement associé à cette alarme.

Dans define.h : déclaration de l'évènement associé à l'alarme

```
#define ALARM_EVENT 0x80 /* Le bit à 1 identifie l'emplacement du bit d'état qui sera
utiliser pour définir l'état de l'évènement ALARM_EVENT (actif/inactif). Ici on a choisi le bit b7 */
```

Dans taskdesc.c : déclaration de l'alarme

```
volatile AlarmObject Alarm_list[] =
{
/*************************************************************************
* ------------------------- First alarm -------------------------
*************************************************************************/
{
……, /* State */
……, /* AlarmValue */
....., /* Cycle */
(RefCounter)&Counter_kernel, /* ptrCounter */
……, /* TaskID2Activate */
……, /* EventToPost */
……. /* CallBack */
},
        };

 volatile Counter Counter_kernel =
 {
          {
          ……., /* maxAllowedValue */
          …, /* ticksPerBase */
          … /* minCycle */
```

```
},
….., /* CounterValue */
…... /* Nbr of Tick for 1 CPT */
};
```

Dans tsk_task0.c :

```
#define ……/* indice de l'alarme dans le tableau Alarm_list */
```

2.3 On désire utiliser les 2 boutons poussoirs reliés aux broches d'interruption externes INT0 pour l'incrémentation de la température et INT1 pour la décrémentation. Ecrire le code à rajouter lors de l'initialisation permettant d'autoriser ces deux interruptions à chaque front descendant des broches.

Tâche 0 : récupération de la valeur de la température

La réception de la valeur analogique est fait à partir d'un convertisseur analogique numérique et stockage par buffer circulaire resultat_temp[] ainsi qu'une variable pointeur RD.

2.4 Compléter le programme suivant décrivant la tâche 0 :

```
TASK(TASK0)
  {
  Unsigned int RegulTemp ;
  …
  RegulTemp=resultat_temp[RD++] ;
  If (RD== TAILLE_BUFFER) RD=0;
  …
  }
```

2.5 Le capteur de température étant très sensible aux parasites, une solution proposée est de faire la moyenne des 20 dernières mesures. La variable RegulTemp sera mise à jour uniquement avec cette moyenne. De plus, la tâche 1 s'occupant de l'asservissement du bloc de refroidissement doit être synchronisé avec la mise à jour de la variable avec l'activation de l'évènement *TASK1_EVENT* Modifier le programme précèdent (réécrire le programme) pour intégrer ces fonctionnalités.

Tâche 2 : Incrémentation de la valeur de la température désirée

La tâche 2 permet d'incrémenter la variable **Temp_desiree** associée aux heures sur pression du bouton BP1. La fonction d'interruption INT0 associée au bouton BP1 doit permettre d'activer la tâche 2.

2.6 Ecrire en langage C, le code permettant l'incrémentation de 1 de la variable Temp_desiree.

2.7 Ecrire le code en langage C de la fonction d'interruption INT0 pour qu'elle active la tâche 2 à chaque pression de BP1.

Tâche 4 : Rafraichissement de l'affichage LCD

La tâche 4 réalise l'affichage de la température sur l'écran LCD. La mise à jour de l'affichage est déclenchée par l'évènement UPDATE_EVENT qui peut être activé à chaque fois qu'un rafraîchissement est nécessaire.

2.8 Ecrire le code en langage C de la tâche 4 en utilisant les fonctions de gestion de l'afficheur LCD :

- *void SPI_Init(void) :* initialisation de l'interface SPI connectée au LCD.
- *void WriteLineSPI_to_LCD(char *s, unsigned int line):* écriture de la chaîne **s** à la ligne **line** (1 ou 2).