FINAL SM 57 - Printemps 2015

Aucun document autorisé

Calculatrice autorisée

I. Questions de cours /5

1. Expliquer le principe de fonctionnement d'un moteur à courant continu à ballais? Justifier votre réponse avec un schéma
2. Quelle est la différence entre une pile et un accumulateur ?
3. Quels sont les avantages et inconvénients d'une batterie au plomb ?
4. Classer par ordre croissant les accumulateurs suivant en fonction de leur énergie spécifique (Wh/kg) : Li-ion / Ni-Cd / Plomb / Ni-Mh

II. Problème : Etude d'un coffre à ouverture électrique /15

Le nouveau C4 Grand Picasso est équipé en option d'une ouverture/fermeture électrique du coffre dans sa dernière version. Cette fonctionnalité est réalisée grâce à une paire de vérins commandés électriquement ainsi que différents capteurs réalisant les fonctions de sécurité quant à l'ouverture/fermeture du coffre:

- Vérins électrique avec capteurs de fin de course à chaque extrémité : un moteur BLDC couplé à une crémaillère permet un mouvement de translation qui ouvre/ferme le coffre. 2 capteurs de fin de course permettent de déterminer la position du vérin pour sa commande. La commande du moteur BLDC est effectuée par un circuit externe à l'ECU. L'ECU donne l'ordre de pilotage du moteur avec deux sorties digitales : l'une pour le mode marche/arrêt et l'autre pour le sens de rotation

- Capteur de pression fixé sur l'axe en translation du vérin : Evalue la force résistive exercé sur l'axe translateur à tout instant. Permet de détecter un obstacle à l'ouverture/fermeture du coffre (exemple : véhicule contre un mur ne permettant pas d'ouvrir le coffre totalement).

- Capteur anti-pincement intégré dans le joint du haillon : résistance variable en fonction de la pression exercée sur le joint placé sur le pourtour du haillon, permet de détecter un objet coincé entre le haillon et la carrosserie du véhicule pendant la fermeture du coffre (exemple : utilisateur laissant sa main sur la jonction haillon/carrosserie lors de la fermeture du coffre). Il en résulte un signal analogique 0-5V.

- Capteur radio réceptionnant la demande d'ouverture de la clé du véhicule : un circuit d'adaptation (non étudié ici) décode un signal radio provenant de la clé du véhicule puis commande une sortie tout ou rien.

Dans cette étude, seule la **fermeture** électrique du coffre sera étudiée :
Le circuit électronique de l'ECU pilotant le coffre est composé d'un dsPIC30F4011 utilisant le système d'exploitation PICOS comme suit :

- La tâche 0 est utilisée pour la lecture du capteur de pression
- La tâche 1 est utilisée pour la réception du signal de fermeture du coffre via la clé
- La tâche 2 est utilisée pour la lecture du capteur de pincement
- La tâche 3 est utilisée pour le pilotage des vérins

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

Tâche 0 : Lecture du capteur de pression

Le capteur de pression dispose d'un boitier de communication CAN pour transmettre ses données. La communication repose sur le principe requête/réponse : L'ECU envoie une trame de requête au capteur qui répond avec la donnée relevée. Les caractéristiques de la trame sont les suivantes :

- Identifiant standard 0x100
- Attente d'une trame de requête pour émettre une trame de réponse
- Donnée codé sur 1 octet, gamme 0 à 10Nm.

La demande de donnée sera envoyée toutes les 200ms, une alarme est donc associée à la tâche 0 pour réaliser cette temporisation. La réception de la trame de donnée ne sera pas faite par interruption mais par simple attente d'une donnée dans le buffer de réception de la boite 0. On considère l'initialisation du module CAN effectué avant le démarrage de PICOS et les boites TXB0 et RXB0 sont utilisées pour l'envoi et la réception.

2.2 Compléter le code permettant de déclarer l'alarme permettant l'envoi d'une trame CAN de requête 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 */
},
        };
```

Dans tsk_task0.c :

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

2.3 Le constructeur définit la limite de pression admissible à 4Nm. A partir de quelle valeur de donnée de la trame faut-il stopper l'action du vérin ?

2.4 Ecrire le programme de la Tâche 0 mettant la variable error_pression à 1 si la pression exercé sur le vérin est trop élevée, sinon à 0 (la variable est globale et sera déclarée en externe). On vérifiera que la trame reçu est bien l'ID 0x100.

Tâche 1 : réception du signal de fermeture du coffre via la clé

Le signal de sortie du module radio est relié à la broche INT0 du microcontrôleur. On considère l'initialisation de l'interruption INT0 effectuée comme suit :

IEC0bits.INT0IE=1 ; IFS0bits.INT0IF=0 ;

2.5 Ecrire en language C la fonction **void __attribute__((__interrupt__)) _INT0Interrupt(void)** pour qu'elle active la tâche 1.

2.6 Ecrire le code de la tâche 1 permettant l'activation de l'évènement CMD_EVENT lors de l'appui sur la clé, utilisé par la tâche 3 pour piloter les vérins.

Tâche 2 : lecture du capteur de pincement

Le capteur est une résistance variable en fonction de la pression exercé sur le joint en silicone par un objet se retrouvant coincé entre le haillon et la carrosserie lors de la fermeture de la porte. Un courant fixe le traverse permettant d'avoir en sortie en tension variable de 0 à 5V pour respectivement de 0 à 4Nm. La tension varie linéairement en fonction de la pression. Le convertisseur Analogique Numérique (ADC) du DSpic30F4011 est un convertisseur 10 bits.

Comme pour le capteur de pression, on souhaite réaliser la mesure toutes les 200ms. Une alarme est donc assignée cette tâche :

2.7 Le constructeur définit la limite de pression admissible à 1,5Nm. A partir de quelle valeur de donnée du résultat de l'ADC faut-il stopper l'action du vérin ?

2.8 Ecrire le programme de la Tâche 2 mettant la variable error_incement à 1 si la pression exercé sur le joint est trop élevée, sinon à 0 (la variable est globale et sera déclarée en externe). On utilisera la fonction **int lectureADC(void)** qui renvoi la valeur numérisée par le convertisseur.

L'alarme est définie par :

```
#define ALARM_TSK2 /* indice de l'alarme dans le tableau Alarm_list */
```

Tâche 3 : le pilotage des vérins

Les capteurs de fin de course qui envoient l'information du vérin fermé ou ouvert sont respectivement branchés sur RB0 et RB1 (0 pour tête de vérin non présente, 1 pour tête de vérin présente). Exemple : si le vérin est à mi-parcours : B0=B1=0. (Rappel : code C pour lire B0 dans la variable resultat: resultat=PORTBbits.RB0;)

La commande des 2 vérins se fait via les sorties RC0 pour ON/OFF (1 pour marche, 0 pour arrêt) et RC1 pour le sens de rotation (0 pour fermeture, 1 pour ouverture).

La tâche 3 est elle aussi cadencée par une alarme définie par :

```
#define ALARM_TSK3 /* indice de l'alarme dans le tableau Alarm_list */
```

2.9 Ecrire le programme de la Tâche 3 réalisant dans l'ordre :

- Initialisation des entrées/sorties et alarm/event
- Attente de l'évènement CMD_EVENT
- Si le vérin n'est pas en position ouvert, piloter le moteur en ordre d'ouverture jusqu'à ce que le coffre soit complètement ouvert
- Si aucun défaut n'est présent, piloter la fermeture du vérin toutes les 200ms jusqu'à la fermeture complète du coffre.

2.10 Pour chaque tâche, donner un numéro de priorité afin que le fonctionnement global du système soit correct.

3 Registres associés à la transmission :

Register 23-2: CiTXnCON: Transmit Buffer Status and Control Register

Upper Byte:							
U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 15							bit 8

Lower Byte:							
U-0	R-0	R-0	R-0	R/W-0	U-0	R/W-0	R/W-0
—	TXABT	TXLARB	TXERR	TXREQ	—	TXPRI<1:0>	
bit 7							bit 0

bit 15-7 **Unimplemented:** Read as '0'

bit 6 **TXABT**: Message Aborted bit
1 = Message was aborted
0 = Message has not been aborted
Note: This bit is cleared when TXREQ is set.

bit 5 **TXLARB:** Message Lost Arbitration bit
1 = Message lost arbitration while being sent
0 = Message did not lose arbitration while being sent
Note: This bit is cleared when TXREQ is set.

bit 4 **TXERR:** Error Detected During Transmission bit
1 = A bus error occurred while the message was being sent
0 = A bus error did not occur while the message was being sent
Note: This bit is cleared when TXREQ is set.

bit 3 **TXREQ:** Message Send Request bit
1 = Request message transmission
0 = Abort message transmission if TXREQ already set, otherwise no effect
Note: The bit will automatically clear when the message is successfully sent.

bit 2 **Unimplemented:** Read as '0'

bit 1-0 **TXPRI<1:0>:** Message Transmission Priority bits
11 = Highest message priority
10 = High intermediate message priority
01 = Low intermediate message Priority
00 = Lowest message priority

Register 23-3: CiTXnSID: Transmit Buffer n Standard Identifier

Upper Byte:							
R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	U-0	U-0	U-0
SID<10:6>					—	—	—
bit 15							bit 8

Lower Byte:							
R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
SID<5:0>						SRR	TXIDE
bit 7							bit 0

bit 15-11 **SID<10:6>:** Standard Identifier bits

bit 10-8 **Unimplemented:** Read as '0'

bit 7-2 **SID<6:0>:** Standard Identifier bits

bit 1 **SRR:** Substitute Remote Request bit
When TXIDE = 0
1 = Message will request a remote transmission
0 = Normal message
When TXIDE = 1, the SRR bit must be set to '1'.

bit 0 **TXIDE:** Extended Identifier bit
1 = Message will transmit extended identifier
0 = Message will transmit standard identifier

Register 23-4: CiTXnEID: Transmit Buffer n Extended Identifier

Upper Byte:							
R/W-x	R/W-x	R/W-x	R/W-x	U-0	U-0	U-0	U-0
EID<17:14>				—	—	—	—
bit 15							bit 8

Lower Byte:							
R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
EID<13:6>							
bit 7							bit 0

bit 15-12 **EID<17:14>:** Extended Identifier bits 17-14

bit 11-8 **Unimplemented:** Read as '0'

bit 7-0 **EID<13:6>:** Extended Identifier bits 13-6

Register 23-5: CiTXnDLC: Transmit Buffer n Data Length Control

Upper Byte:							
R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
EID<5:0>						TXRTR	TXRB1
bit 15							bit 8

Lower Byte:							
R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	U-0	U-0	U-0
TXRB0	DLC<3:0>				—	—	—
bit 7							bit 0

bit 15-10 **EID<5:0>:** Extended Identifier bits 5-0

bit 9 **TXRTR:** Remote Transmission Request bit
When TXIDE = 1,
1 = Message will request a remote transmission
0 = Normal message
When TXIDE = 0, the TXRTR bit is ignored.

bit 8-7 **TXRB<1:0>:** Reserved Bits
Note: User must set these bits to '0' according to CAN protocol.

bit 6-3 **DLC<3:0>:** Data Length Code bits

bit 2-0 **Unimplemented:** Read as '0'

Register 23-6: CiTXnBm: Transmit Buffer n Data Field Word m

Upper Byte:							
R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
CTXB<15:8>							
bit 15							bit 8

Lower Byte:							
R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
CTXB<7:0>							
bit 7							bit 0

bit 15-0 **CTXB<15:0>:** Data Field Buffer Word bits (2 bytes)

Registres associés à la réception :

Register 23-7: CiRX0CON: Receive Buffer 0 Status and Control Register

Upper Byte:							
U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 15							bit 8

Lower Byte:							
R/C-0	U-0	U-0	U-0	R-0	R/W-0	R/W-0	R-0
RXFUL	—	—	—	RXRTRRO	DBEN	JTOFF	FILHIT0
bit 7							bit 0

bit 15-8 **Unimplemented:** Read as '0'

bit 7 **RXFUL:** Receive Full Status bit
1 = Receive buffer contains a valid received message
0 = Receive buffer is open to receive a new message
Note: This bit is set by the CAN module and should be cleared by software after the buffer is read.

bit 6-4 **Unimplemented:** Read as '0'

bit 3 **RXRTRRO:** Received Remote Transfer Request bit (read only)
1 = Remote Transfer Request was received
0 = Remote Transfer Request not received
Note: This bit reflects the status of the last message loaded into Receive Buffer 0.

bit 2 **DBEN:** Receive Buffer 0 Double Buffer Enable bit
1 = Receive Buffer 0 overflow will write to Receive Buffer 1
0 = No Receive Buffer 0 overflow to Receive Buffer 1

bit 1 **JTOFF:** Jump Table Offset bit (read only copy of DBEN)
1 = Allows Jump Table offset between 6 and 7
0 = Allows Jump Table offset between 0 and 1

bit 0 **FILHIT0:** Indicates Which Acceptance Filter Enabled the Message Reception bit
1 = Acceptance Filter 1 (RXF1)
0 = Acceptance Filter 0 (RXF0)
Note: This bit reflects the status of the last message loaded into Receive Buffer 0.

Register 23-8: CiRX1CON: Receive Buffer 1 Status and Control Register

Upper Byte:							
U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 15							bit 8

Lower Byte:							
R/C-0	U-0	U-0	U-0	R-0	R-0	R-0	R-0
RXFUL	—	—	—	RXRTRRO	FILHIT<2:0>		
bit 7							bit 0

Unimplemented: Read as '0'

RXFUL: Receive Full Status bit
1 = Receive buffer contains a valid received message
0 = Receive buffer is open to receive a new message
Note: This bit is set by the CAN module and should be cleared by software after the buffer is read.

Unimplemented: Read as '0'

RXRTRRO: Received Remote Transfer Request bit (read only)
1 = Remote transfer request was received
0 = Remote transfer request not received
Note: This bit reflects the status of the last message loaded into Receive Buffer 1.

FILHIT<2:0>: Indicates Which Acceptance Filter Enabled the Message Reception bits
101 = Acceptance filter 5 (RXF5)
100 = Acceptance filter 4 (RXF4)
011 = Acceptance filter 3 (RXF3)
010 = Acceptance filter 2 (RXF2)
001 = Acceptance filter 1 (RXF1) (Only possible when DBEN bit is set)
000 = Acceptance filter 0 (RXF0) (Only possible when DBEN bit is set)

Register 23-9: CiRXnSID: Receive Buffer n Standard Identifier

Upper Byte:							
U-0	U-0	U-0	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
—	—	—	SID<10:6>				
bit 15							bit 8

Lower Byte:							
R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
SID<5:0>						SRR	RXIDE
bit 7							bit 0

bit 15-13 **Unimplemented:** Read as '0'

bit 12-2 **SID<10:0>:** Standard Identifier bits

bit 1 **SRR:** Substitute Remote Request bit (Only when RXIDE = 1)
When RXIDE = 0,
1 = Remote transfer request occured
0 = No remote transfer request occured
When RXIDE = 1, the SRR bit can be ignored.

bit 0 **RXIDE:** Extended Identifier Flag bit
1 = Received message is an extended data frame, SID<10:0> are EID<28:18>
0 = Received message is a standard data frame

Register 23-10: CiRXnEID: Receive Buffer n Extended Identifier

Upper Byte:							
U-0	U-0	U-0	U-0	R/W-x	R/W-x	R/W-x	R/W-x
—	—	—	—	EID<17:14>			
bit 15							bit 8

Lower Byte:							
R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
EID<13:6>							
bit 7							bit 0

bit 15-12 **Unimplemented:** Read as '0'

bit 11-0 **EID<17:6>:** Extended Identifier bits 17-6

Register 23-12: CiRXnDLC: Receive Buffer n Data Length Control

Upper Byte:							
R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
EID<5:0>						RXRTR	RB1
bit 15							bit 8

Lower Byte:							
U-0	U-0	U-0	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
—	—	—	RB0	DLC<3:0>			
bit 7							bit 0

bit 15-10 **EID<5:0>:** Extended Identifier bits

bit 9 **RXRTR:** Receive Remote Transmission Request Control bit
When RXIDE = 1,
1 = Remote transfer request
0 = No remote transfer request
When RXIDE = 0, the RXRTR bit can be ignored
Note: This bit reflects the status of the RTR bit in the last received message.

bit 8 **RB1:** Reserved bit 1
Reserved by CAN Spec and read as '0'

bit 4 **RB0:** Reserved bit 0
Reserved by CAN Spec and read as '0'

bit 3-0 **DLC<3:0>:** Data Length Code bits (Contents of Receive Buffer)

Register 23-11: CiRXnBm: Receive Buffer n Data Field Word m

Upper Byte:							
R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
CRXB<15:8>							
bit 15							bit 8

Lower Byte:							
R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
CRXB<7:0>							
bit 7							bit 0

bit 15-0 **CRXB<15:0>:** Data Field Buffer Word bits (2 bytes)