Examen TR57

I Mise en œuvre d'un régulateur PI parallèle sur TMS320LF2407

La grandeur de consigne et la rétroaction de l'asservissement sont exprimées numériquement avec le facteur k1 par rapport aux valeurs réelles : X_DSP = k1·X_reel. La grandeur de sortie du régulateur est exprimée numériquement avec le facteur k2 par rapport aux valeurs réelles : U_DSP = k2·U_reel.

I.1 Rappeler l'expression de la fonction de transfert continue (en p) d'un régulateur PI parallèle.

I.2 Déterminer l'équation récurrente exprimant Up_reel(n) la sortie du correcteur P seul.

I.3 En déduire l'équation récurrente exprimant la sortie Up_DSP(n) en fonction des grandeurs d'entrée manipulées par le DSP.

I.4 Déterminer l'expression de la fonction de transfert échantillonnée (en z) du correcteur I en utilisant la méthode d'approximation du rectangle inférieur.

I.5 Déterminer l'équation récurrente correspondante exprimant la sortie Ui_DSP(n).

I.6 Ecrire en langage C la partie du sous-programme d'interruption correspondant au régulateur PI en tenant compte des éventuels restes de divisions entières.

II Réception SCI par interruption et buffer circulaire sur TMS320LF2407

II.1 Déclarer le buffer circulaire d'une taille de 2000 mots, ainsi que les variables et sémaphores nécessaires à la gestion du buffer circulaire.

II.2 Ecrire en langage C la fonction ***interrupt void inter_SCI(void)*** gérant la réception par buffer circulaire (l'interruption en réception du SCI est orientée vers l'interruption CPU INT5).

II.3 Ecrire en langage C la fonction ***int kbhit(void)*** testant la présence d'une valeur dans le buffer de réception.

II.4 Ecrire en langage C la fonction ***int getch(void)*** réalisant la lecture d'une valeur présente dans le buffer de réception.

II.5 On désire que l'interruption ***inter_SCI*** puisse préempter une autre interruption ***inter_ADC***. Indiquer comment permettre cela sans que la gestion des autres interruptions ne soit perturbée.

III Création d'une tâche périodique dans le noyau temps réel PICOS18

On désire mettre en œuvre la tâche 0 faisant clignoter une LED témoin à la fréquence fixe de 10 Hz. L'arrêt du clignotement de la LED ou le non respect de la fréquence de clignotement signalerait un problème de résolution des contraintes temporelles par PICOS18.

III.1 Le projet de départ est constitué des fichiers suivants : main.c, tascdesc.c, tsk_tsk0.c, int.c, define .h, picos18.lib, picos18iz.o et 18f458.lkr. Rappeler le rôle de chacun de ces fichiers.

III.2 Indiquer suivant quel principe l'évènement associé à une alarme est déclaré et identifié par rapport à d'autres évènements (la déclaration de l'alarme à l'index 0 du tableau Alarm_list[] n'est pas demandée).

III.3 Indiquer comment configurer cette alarme pour permettre un clignotement de la LED à 10 Hz par appels à intervalles de temps réguliers à la fonction ***COMMUTE_LED()***.

III.4 Ecrire le code en langage C de la tâche 0 (fonction ***TASK(TASK0)***).

IV Configuration du contrôleur CAN du TMS320LF2407

On désire utiliser la mailbox 0 pour recevoir les trames d'identificateur standard 0x300 contenant des données de 8 octets.

IV.1 Donner le contenu des registres initialisant l'identificateur et le champ de contrôle de la mailbox 0.

IV.2 Déterminer le contenu des registres BCR1 et BCR2 pour initialiser la fréquence de transmission à 1Mbauds (on précisera chaque paramètre du Bit Timing pour une fréquence d'horloge de 30 MHz).

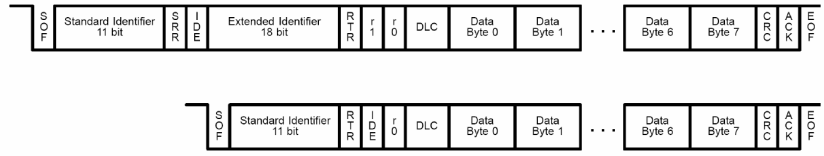
IV.3 Ecrire en langage C le sous-programme ***void init_CAN(void)*** d'initialisation de la mailbox 0 et du débit binaire du contrôleur CAN.

IV.4 Représenter schématiquement l'interconnexion du DSP au bus CAN, en faisant apparaître les cœurs DSP, les contrôleurs CAN, les drivers CAN et le bus CAN.

IV.5 Indiquer maintenant comment permettre la réception par la mailbox 0 de toutes les trames dont l'identificateur est compris entre 0x300 et 0x30F inclus.

Annexes

Illustration des formats de trame étendu et standard.



Registre d'identification des messages (MSB) : MSGIDnH

15	14	13	12–0
IDE	AME	AAM	IDH[28:16]
RW	RW	RW	RW

Note: R = Read access; W = Write access

Bit 15 **IDE.** Identifier Extension Bit.
- 0 The received message has a standard identifier (11 bits).† The message to be sent has a standard identifier (11 bits).‡
- 1 The received message has an extended identifier (29 bits).† The message to be sent has an extended identifier (29 bits).‡

† In case of a receive mailbox
‡ In case of a transmit mailbox

Bit 14 **AME.** Acceptance Mask Enable Bit.
- 0 No acceptance mask will be used. All identifier bits in the received message and the receive MBOX must match in order to store the message.
- 1 The corresponding acceptance mask is used.

This bit will not be affected by a reception.

This bit is relevant for receive mailboxes only. Hence, it is applicable for MBOX0 and MBOX1 and also for MBOX2 and MBOX3, if they are configured as receive mailboxes. It is a *don't care* for mailboxes 4 and 5.

Bit 13 **AAM.** Auto Answer Mode Bit.

0	Transmit mailbox	The mailbox does not reply to remote requests automatically. If a matching identifier is received, it is not stored.
	Receive mailbox	No influence on a receive mailbox.
1	Transmit mailbox	If a matching remote request is received, the CAN peripheral answers by sending the contents of the mailbox.
	Receive mailbox	No influence on a receive mailbox.

This bit is only used for mailboxes 2 and 3. This bit will not be affected by a reception.

Bits 12–0 **IDH[28:16].** Upper 13 Bits of extended identifier. For a standard identifier, the 11-bit identifier will be stored in bits 12 to 2 of the MSGID's upper word.

Registre d'identification des messages (LSB) : MSGIDnL

15–0
IDL[15:0]
RW

Note: R = Read access; W = Write access

Bits 15–0 **IDL[15:0].** The lower part of the extended identifier is stored in these bits.

Rq : L'identificateur ne peut être modifié que si la Mailbox concernée est désactivée (MEn = 0 dans MDER)

Registre champ contrôle des messages : MSGCTRLn

15–5	4	3–0
Reserved	RTR	DLC[3:0]
	RW	RW

Note: R = Read access; W = Write access

Bits 15–5 **Reserved.**

Bit 4 **RTR.** Remote Transmission Request bit.
- 0 Data frame.
- 1 Remote frame.

Bits 3–0 **DLC.** Data Length Code.

This value determines how many data bytes are used for transmission. This field will be updated by the received data frame (i.e., the DLC value of the data frame received will be copied in this field).

0001	1 byte
0010	2 bytes
0011	3 bytes
0100	4 bytes
0101	5 bytes
0110	6 bytes
0111	7 bytes
1000	8 bytes

Registre masque d'acceptation local : LAMnL (LSB) et LAMnH (MSB)

15–0
LAMn[15:0]
RW-0

Note: R = Read access; W = Write access; value following dash (–) = value after reset

Bits 15–0 **LAMn[15:0].** Lower part of the local acceptance mask. These bits enable the masking of any identifier bit of an incoming message.

0 Received identifier bit value must match the identifier bit of the receive mailbox.

1 Accept a 0 or a 1 (don't care) for the corresponding bit of the receive identifier.

15	14–13	12–0
LAMI	Reserved	LAMn[28:16]
RW-0		RW-0

Note: R = Read access; W = Write access; value following dash (–) = value after reset

Bit 15 **LAMI.** Local acceptance mask identifier extension bit.

0 The identifier extension bit stored in the mailbox determines which messages are received (standard or extended).

1 Standard and extended frames can be received. In case of an extended frame, all 29 bits of the identifier are stored in the mailbox and all 29 bits of the global acceptance mask register are used for the filter. In case of a standard frame, only the first eleven bits (bits 12–2 of LAMn_H) of the identifier and the local acceptance mask are used.

Bits 14–13 **Reserved.**

Bits 12–0 **LAMn[28:16].** Upper 13 bits of the local acceptance mask.

0 Received identifier bit value must match the identifier bit of the receive mailbox. For example, if bit 27 of LAM is zero, then bit 27 of the transmitted MSGID and bit 27 of the receive mailbox MSGID must be the same.

1 Accept a 0 or a 1 (*don't care*) for the corresponding bit of the receive identifier.

Registre Direction/Autorisation des mailboxes : MDER

15–8							
Reserved							
7	6	5	4	3	2	1	0
MD3	MD2	ME5	ME4	ME3	ME2	ME1	ME0
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

Note: R = Read access; W = Write access; value following dash (–) = value after reset

Bits 15–8 **Reserved.**

Bits 7–6 **MDn.** Mailbox direction for mailbox n. Mailboxes 2 and 3 can be configured as a transmit or receive mailbox.

Mailbox direction bits are defined as follows:

0 Transmit mailbox.

1 Receive mailbox.

After power-up, all bits are cleared.

Bits 5–0 **MEn.** Mailbox-enable for mailbox n. Each mailbox can be enabled or disabled. If the bit MEn is 0, the corresponding mailbox n is disabled. The mailbox must be disabled before writing to any identifier field.

If the corresponding bit in ME is set, the write access to the identifier of a message object is denied and the mailbox is enabled for the CAN module.

Mailboxes that are disabled may be used as additional memory for the DSP.

Mailbox enable bits are defined as follows:

0 Disable mailbox.

1 Enable mailbox.

Registre de configuration binaire 1 : BCR1

15–11	10	9–8
Reserved	SBG	SJW[1:0]
	RW-0	RW-0

7	6–3	2–0
SAM	TSEG1-[3:0]	TSEG2-[2:0]
RW-0	RW-0	RW-0

Note: R = Read access; W = Write access; value following dash (–) = value after reset

Bits 15–11 **Reserved.**

Bit 10 **SBG.** Synchronization edge.

0 The CAN module resynchronizes on the falling edge only.

1 Reserved.

Bits 9–8 **SJW.** Synchronization jump width.

SJW indicates by how many units of TQ a bit is allowed to be lengthened or shortened when resynchronizing with the receive data stream on the CAN bus. The synchronization is performed with the falling edge (SBG = 0) of the bus signal. SJW is programmable from 1 to 4 TQ.

Note: Since the SJW[1:0] value is enhanced by one by the CAN module, the value that is written in bits [1:0] is actually (SJW – 1), where SJW is the timing segment referred to in Figure 10–17, CAN Bit Timing.

Bit 7 **SAM.** Sample point setting.

This parameter sets the number of samples used by the CAN module to determine the actual level of the CAN bus. When the SAM bit is set, the level determined by the CAN bus corresponds to the result from the majority decision of the last three values. The sample points are at the sample point and twice before with a distance of 1/2 TQ.

0 The CAN module samples only once.

1 The CAN module samples three times and makes a majority decision.

Bits 6–3 **TSEG1[3:0].** Time segment 1.

This parameter specifies the length of the TSEG1 segment in TQ units.

TSEG1 combines PROP SEG and PHASE SEG1 segments (CAN protocol).

TSEG1 = PROP SEG + PHASE SEG1.

The value of TSEG1 is programmable from 3 to 16 TQ and must be greater than or equal to TSEG2.

Note: Since the TSEG1[3:0] value is enhanced by one by the CAN module, the value that is written in bits [3:0] is actually (TSEG1 – 1), where TSEG1 is the timing segment referred to in Figure 10–17, CAN Bit Timing.

Bits 2–0 **TSEG2[2:0].** Time segment 2.

TSEG2 defines the length of PHASE SEG2 in TQ units.

The value of TSEG2 is programmable from 2 to 8 TQ in compliance with the formula:

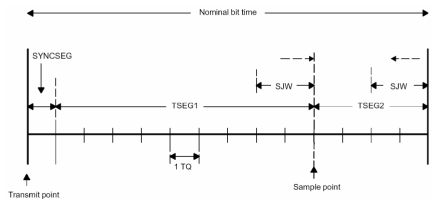
$$(SJW + 1) \le TSEG2 \le 8.$$

Note: Since the TSEG2[2:0] value is enhanced by one by the CAN module, the value that is written in bits [2:0] is actually (TSEG2 – 1), where TSEG2 is the timing segment referred to in Figure 10–17, CAN Bit Timing.

Note:

The user-defined values for the SJW, TSEG1, and TSEG2 parameters are enhanced by one (by the internal logic) when the CAN module accesses these parameters.

CAN Bit Timing



$$\text{Baud Rate} = \frac{I_{CLK}}{(BRP + 1) \times \text{Bit Time}} = \frac{1}{TQ * \text{Bit Time}} \quad \text{(in bits per second)}$$

$$\boxed{\text{Bit Time} = TSEG1 + TSEG2 + 1}$$

(avec TSEG1=TSEG1[3..0]+1 et TSEG2=TSEG2[2..0]+1)

CAN Bit Timing Examples for I_{CLK} = 40 MHz

TSEG1[3..0]	TSEG2[2..0]	Bit Time	BRP	Sampling Point	Baud Rate
4	3	10	3	60%	1 Mbit/s
10	7	20	3	60%	500 Kbit/s
9	4	16	9	68.8%	250 Kbit/s
14	8	25	15	64%	100 Kbit/s
11	6	20	39	65%	50 Kbit/s

Registre de configuration binaire 2 : BCR2

15–8
Reserved

7–0
BRP[7:0]
RW-0

Note: R = Read access; W = Write access; value following dash (–) = value after reset

Bits 15–8 **Reserved.**

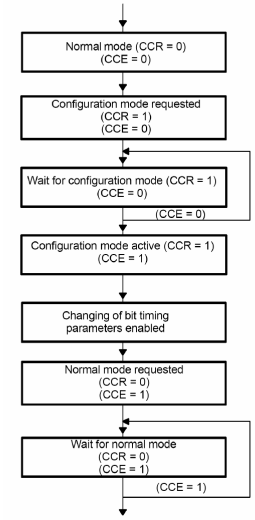
Bits 7–0 **BRP.** Baud Rate Prescaler.

Bits 7 to 0 of this field specify the duration of a time quantum (TQ) in CAN module system clock units. The length of one TQ is defined by:

$$TQ = \frac{BRP + 1}{I_{CLK}}$$

where I_{CLK} is the frequency of the CAN module system clock,which is the same as CLKOUT.

Principe de configuration du Bit Timing



Registre de contrôle maître : MCR

15–14	13	12	11	10	9	8
Reserved	SUSP	CCR	PDR	DBO	WUBA	CDR
	RW-0	RW-1	RW-0	RW-0	RW-0	RW-0

7	6	5–2	1–0
ABO	STM	Reserved	MBNR[1:0]
RW-0	RW-0		RW-0

Note: R = Read access; W = Write access; value following dash (–) = value after reset

Bits 15–14 **Reserved.**

Bit 13 **SUSP.** Action on emulator suspend. The value of the SUSP bit has no effect on the receive mailboxes.

0 *Soft mode.* The peripheral shuts down during suspend after the current transmission is completed.

1 *Free mode.* The peripheral continues to run in suspend.

Bit 12 **CCR.** Change Configuration Request.

0 The CPU requests normal operation. It also exits the bus-off state after the obligatory bus-off recovery sequence.

1 The CPU requests write access to the bit configuration registers (BCRn). Flag CCE in the GSR indicates if the access is granted. CCR must be set while writing to bit timing registers BCR1 and BCR2. This bit will automatically be set to 1 if the bus-off condition is valid and the ABO bit is not set. Thus, it has to be reset to exit the bus-off mode.

Bit 11 **PDR.** Power-Down Mode Request.

Before the CPU enters its IDLE mode (if IDLE shuts off the peripheral clocks), it must request a CAN power down by writing to the PDR bit. The CPU must then poll the PDA bit in the GSR, and enter IDLE only after PDA is set.

0 The power-down mode is not requested (normal operation).

1 The power-down mode is requested.

Bit 10 **DBO.** Data Byte Order.

0 The data is received or transmitted in the following order: Data byte 3,2,1,0,7,6,5,4.

1 The data is received or transmitted in the following order: Data byte 0,1,2,3,4,5,6,7.

Note:

The DBO bit is used to define the order in which the data bytes are stored in the mailbox when received and in which the data bytes are transmitted. Byte 0 is the first byte in the message and Byte 7 is the last one as shown in the figure of the CAN message (Figure 10–4).

Bit 9 **WUBA.** Wake Up on Bus Activity.

0 The module leaves the power-down mode only after the user writes a 0 to clear PDR.

1 The module leaves the power-down mode when detecting any dominant value on the CAN bus.

Bit 8 **CDR.** Change Data Field Request.

The CDR bit is applicable for mailboxes 2 and 3 only *and* in the following situation: 1) either (or both) of these mailboxes are configured for transmission and 2) the corresponding AAM bit (MSGIDxH.13) is set.

0 The CPU requests normal operation.

1 The CPU requests write access to the data field of the mailbox in MBNR (also located in MCR). The CDR bit must be cleared by the CPU after accessing the mailbox. The CAN module does not transmit the mailbox if the CDR is set. This is checked by the state machine before and after it reads the data from the mailbox to store it in the transmit buffer.

Bit 7 **ABO.** Auto Bus On.

0 The bus-off state may only be left after 128×11 consecutive recessive bits on the bus *and* after having reset the CCR bit.

1 After the bus-off state, the module goes back to the bus-on state after 128×11 consecutive recessive bits.

Bit 6 **STM.** Self-Test Mode.

0 The module is in normal mode.

1 The module is in Self-Test mode. In this mode, the CAN module generates its own ACK signal. Thus, it enables operation without a bus connected to the module. The message is not sent but read back and stored in the appropriate mailbox. The remote frame handling with Auto Answer mode set is *not* implemented in STM. The received message ID will not be stored in the receive mailbox in this mode.

Bits 5–2 **Reserved.**

Bits 1–0 **MBNR.** Mailbox Number (for CDR bit assertion).

The CPU requests a write access to the data field for the mailbox having this number and configured for Remote Frame Handling. These are mailboxes 2 (10) or 3 (11), but not 0, 1, 4 or 5.

Registre d'état global : GSR

15–8
Reserved

7–6	5	4	3	2	1	0
Reserved	SMA	CCE	PDA	Rsvd	RM	TM
	R-0	R-1	R-0		R-0	R-0

Note: R = Read access; value following dash (–) = value after reset

Bits 15–6 **Reserved.**

Bit 5 **SMA.** Suspend Mode Acknowledge.

0 The CAN peripheral is not in suspend mode.

1 The CAN peripheral has entered suspend mode.

This bit is set after a latency of 1 clock cycle up to the length of one frame after the SUSPEND signal is activated.

Bit 4 **CCE.** Change Configuration Enable.

0 Write access to the configuration registers is denied.

1 The CPU has write access to the configuration registers BCR while the CCR bit (MCR.12) is set. Access is granted after reset or when the CAN module reaches the idle state.

This bit is set after a latency of 1 clock cycle up to the length of one frame.

Bit 3 **PDA.** Power-Down Mode Acknowledge.

Before the CPU enters its IDLE mode (to potentially shut off ALL device clocks), it must request a CAN power down by writing to the PDR bit in MCR. The CPU must then poll the PDA bit and enter IDLE only after PDA is set.

0 Normal operation.

1 The CAN peripheral has entered the power-down mode.

This bit is set after a latency of 1 clock cycle up to the length of one frame.

Bit 2 **Reserved.**

Bit 1 **RM.** The CAN module is in the Receive Mode.

This bit reflects what the CBM is actually doing regardless of mailbox configuration.

0 The CAN core module is not receiving a message.

1 The CAN core module is receiving a message.

Bit 0 **TM.** The CAN module is in the Transmit Mode.

This bit reflects what the CBM is actually doing regardless of mailbox configuration.

0 The CAN core module is not transmitting a message.

1 The CAN core module is transmitting a message.