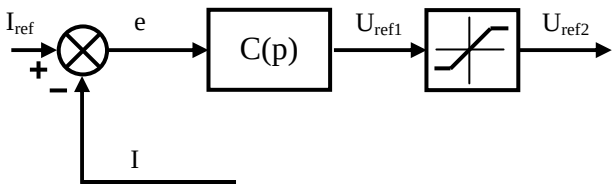
Examen TR57

I Asservissement de courant par TMS320LF2407A (11 points / 30)

On désire mettre en œuvre un asservissement de courant à l'aide du DSP contrôleur TMS320LF2407A cadencé à 40 MHz. Le correcteur C(p) est de type Proportionnel Intégral Dérivé (PID) sous sa forme parallèle. Le courant I à asservir est mesuré par un capteur de courant à effet Hall associé à un conditionneur. Le correcteur C(p) donne la consigne de tension U_{ref1} à appliquer au processus avant saturation, décomposable comme la somme des tensions de sortie des correcteurs P, I et D (U_{ref1} = Up + Ui + Ud).



$$C(p) = k_p + \frac{k_i}{p} + \frac{k_d\, p}{1 + \frac{k_d}{N \cdot k_p} p}$$

Les courants I et I_{ref} ainsi que les tensions U_{ref1} et U_{ref2} sont respectivement exprimés dans le programme du DSP par les variables globales COURANT, COURANTREF, TENSION1 et TENSION2.
On donne les relations suivantes :
COURANT = 16.I
COURANTREF = 16.I_{ref}
TENSION1 = 4.U_{ref1}
TENSION2 = TENSION1 si Vmin ≤ TENSION1 ≤ Vmax, sinon TENSION2 est saturée à Vmin ou Vmax.
UP = 4.Up; UI = 4.Ui; UD = 4.Ud avec TENSION1 = UP + UI + UD

I.1 Déterminer l'équation récurrente exprimant UP(n) la sortie du correcteur P seul en fonction de l'entrée et des paramètres du correcteur. *(1 point)*

I.2 Déterminer l'équation récurrente exprimant UI(n) la sortie du correcteur I seul en fonction de l'entrée et des paramètres du correcteur en utilisant la méthode d'approximation du rectangle inférieur. *(2 points)*

I.3 Déterminer l'équation récurrente exprimant UD(n) la sortie du correcteur D seul en fonction de l'entrée et des paramètres du correcteur en utilisant la méthode d'approximation du rectangle inférieur. *(2 points)*

I.4 Le régulateur PID est calculé par la fonction d'interruption ***interrupt void regul(void)*** liée au vecteur INT1. Indiquer comment initialiser le vecteur d'interruption INT1. *(1 point)*

I.5 Compléter en langage C les pointillés dans la fonction d'interruption ***interrupt void regul(void)*** ci-dessous (on ne tiendra pas compte des restes des éventuelles divisions entières) : *(3 points)*

```
#define     Vmax  995
#define     Vmin  (-995)

/* Interruption liée au module ADC/INT1 (echantillonnage) */
interrupt void regul(void)
{
ADCTRL2 = 0xC600; /* RAZ flag ADC */
IFR = ……;       /* RAZ flag INT1 */

/* saturation de la tension de référence */
……
```

```
/* Pilotage du moteur */
CMPR1 = 1000 + TENSION2;
CMPR2 = 1000 – TENSION2;

/* Lecture du courant de référence et à asservir (résultats de CA/N) */
COURANTREF = (RESULT0>>6)-512;
COURANT = (RESULT1>>6)-512;

/* Calcul de l'asservissement de courant */
……
}
```

I.6 Expliquer brièvement (2 lignes maximum) comment choisir la fréquence de déclenchement de l'interruption ***interrupt void regul(void)*** (limites max et min). *(1 point)*

I.7 Expliquer brièvement (4 lignes maximum) comment choisir la fréquence PWM permettant d'imposer au processus la tension calculée (limites max et min). *(1 point)*

II Communications par bus CAN (19 points / 30)

Une machine asynchrone triphasée réalisant l'entrainement linéaire de métaux dans une fonderie est pilotée par un onduleur triphasé à MLI. La précision du pilotage permet d'assurer la qualité et la cadence de production. Les ordres de contrôle du convertisseur de puissance et les asservissements sont gérés par une carte électronique à base du DSP contrôleur TMS320LF2407A cadencé à 40 MHz. Ce processus temps réel est relié à une IHM (Interface Homme Machine) permettant de donner des ordres élémentaires et de recevoir des données utilisées pour effectuer la surveillance du système. L'IHM et la carte DSP communiquent par l'intermédiaire d'une liaison CAN 2.0A configurée à une fréquence de 250 kbit/s.

On désire étudier la configuration et la programmation du contrôleur CAN du TMS320LF2407A. On utilise la mailbox 0 pour recevoir les trames provenant de l'IHM, et la mailbox 5 pour émettre.
La mailbox 0 permet de recevoir tous les messages d'identificateur compris entre 0x200 et 0x20F inclus, en utilisant les masques de filtrage. La mailbox 5 envoie les messages d'identificateur 0x300.
Les trames transmises sont exclusivement des trames de données comprenant 8 octets de données. Le mode de réponse automatique n'est pas utilisé.
La lecture des messages CAN reçu s'effectue en utilisant l'interruption ***CANMBINT***, la lecture du message et de l'identificateur (compris entre 0x200 et 0X20F) et le stockage de ces valeurs dans un buffer circulaire.

II.1 Déterminer des valeurs valides des paramètres ***Bit Time***, ***TSEG1*** et ***TSEG2*** pour obtenir la fréquence voulue de 250 kbit/s. *(2 points)*

II.2 Déterminer le contenu des registres ***BCR1*** et ***BCR2*** pour les paramètres précédents. *(1 point)*

II.3 Configuration de la mailbox 0 : donner le contenu des registres ***MSGID0H***, ***MSGID0L***, ***LAM0H***, ***LAM0L*** et ***MSGCTRL0***. *(2 points)*

II.4 Configuration de la mailbox 5 : donner le contenu des registres ***MSGID5H***, ***MSGID5L*** et ***MSGCTRL5***. *(1 point)*

II.5 Configuration de l'interruption : donner le contenu des registres ***CAN_IMR*** et ***IMR*** pour configurer ***CANMBINT*** en priorité faible (orientée sur le vecteur INT5). *(2 points)*

II.6 Indiquer comment initialiser le vecteur d'interruption INT5 pour le lier à la fonction ***interrupt void receptMB0(void)***. *(1 point)*

II.7 Ecrire en langage C le sous-programme ***void init_CAN(void)*** réalisant la configuration du contrôleur CAN et des mailboxes 0 et 5. *(3 points)*

II.8 Donner l'organigramme de l'interruption de réception CAN sur la mailbox 0 avec stockage en buffer circulaire de l'identificateur et des octets reçus. *(2 points)*

II.9 Ecrire en langage C la fonction d'interruption correspondante et déclarer les tableaux et variables nécessaires. *(2 points)*

On désire étudier la programmation du DSP pour permettre la prise en charge du protocole CANOpen.

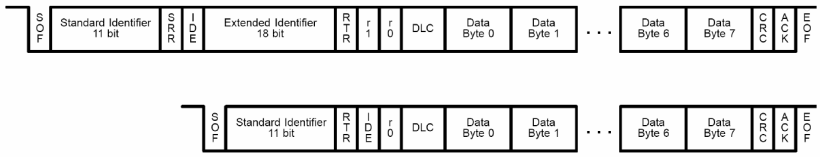
II.10 Décrire (5 lignes maximum) ce qu'est un dictionnaire d'objets. *(1 point)*

II.11 Décrire (2 lignes maximum) suivant quel principe un récepteur peut différentier si une trame est de type SDO ou PDO. *(1 point)*

II.12 Décrire (5 lignes maximum) pour les deux types de trame SDO et PDO, comment un paramètre (index et sous-index) est identifié lors de sa transmission. *(1 point)*

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:

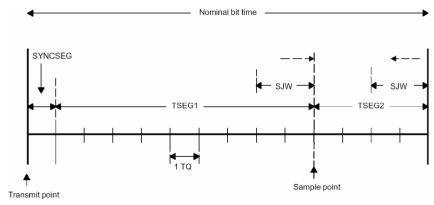
$$(SJW + 1) \le TSEG2 \le 8.$$

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.**

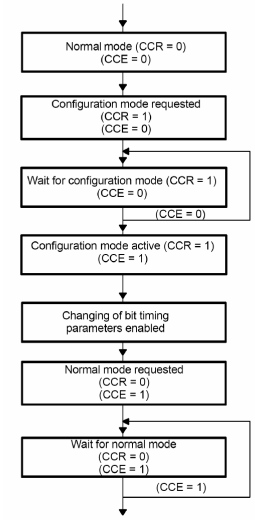
Bits 7–0 **BRP.** Baud Rate Prescaler.

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.

Registre de contrôle des réceptions : RCR

15	14	13	12	11	10	9	8
RFP3	RFP2	RFP1	RFP0	RML3	RML2	RML1	RML0
RC-0	RC-0	RC-0	RC-0	R-0	R-0	R-0	R-0
7	6	5	4	3	2	1	0
RMP3	RMP2	RMP1	RMP0	OPC3	OPC2	OPC1	OPC0
RC-0	RC-0	RC-0	RC-0	RW-0	RW-0	RW-0	RW-0

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

Bits 15–12 **RFPn.** Remote Frame Pending Register (for mailbox n).

Whenever a remote frame request is received by the CAN Peripheral, the corresponding bit RFPn is set.

It may be cleared by the CPU if the TRSn is not set; otherwise, it is reset automatically. If the CPU tries to reset a bit and the CAN Peripheral tries to set the bit at the same time, the bit is cleared.

If the AAM bit in the MSGIDn register is not set (and thus no answer is sent automatically), the CPU must clear bit RFPn after handling the event.

If the message is sent successfully, RFPn is cleared by the CAN Peripheral.

The CPU cannot interrupt an ongoing transfer.

Bits 11–8 **RMLn.** Receive Message Lost (for mailbox n).

If an old message is overwritten by a new one in mailbox n, bit RMLn is set. RMLn is not set in mailboxes that have the OPCn bit set. Thus, a message may be lost without notification.

These bits can only be reset by the CPU and can be set by the internal logic. They can be cleared by writing a 1 to RMPn. If the CPU tries to reset a bit and the CAN tries to set the bit at the same time, the bit is set.

If one or more RML bits in the RCR register are set, the RMLIF in the CAN_IFR register is also set. This may initiate an interrupt if the RMLIM bit in the CAN_IMR register is set.

Bits 7–4 **RMPn.** Received Message Pending (for mailbox n).

If a received message is stored in mailbox n, the RMPn bit is set.

The RMP bits can only be reset by the CPU and are set by the CAN internal logic. The RMPn and RMLn bits are cleared by writing a 1 to the RMPn bit at the corresponding bit location. If the CPU tries to reset a bit and the CAN tries to set the bit at the same time, the bit is set.

A new incoming message will overwrite the stored one if the OPCn bit is cleared. If not, the next mailboxes are checked for a matching identifier. When the old message is overwritten, the corresponding status bit RMLn is set.

The RMP bits in the RCR register set the mailbox interrupt flag (MIFx) bit in the CAN_IFR register if the corresponding interrupt mask bit in the CAN_IMR register is set. The MIFx flag initiates a mailbox interrupt if enabled.

Bits 3–0 **OPCn.** Overwrite Protection Control (for mailbox n).

If there is an overflow condition for mailbox n, the new message is stored/ignored depending on the OPCn value. If the corresponding OPCn bit is set to 1, the old message is protected against being overwritten by the new message. Thus, the next mailboxes are checked for a matching identifier. If no other mailbox is found, the message is lost without further notification. If OPCn bit is not set, the old message is overwritten by the new one.

Les interruptions du contrôleur CAN

Registre des indicateurs d'interruption CAN : CAN_IFR

15–14	13	12	11	10	9	8
Reserved	MIF5	MIF4	MIF3	MIF2	MIF1	MIF0
	R-0	R-0	R-0	R-0	R-0	R-0
7 6	5	4	3	2	1	0
Rsvd RMLIF	AAIF	WDIF	WUIF	BOIF	EPIF	WLIF
RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0

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

Bits 15–14 **Reserved.**

Bits 13–8 **MIFx.** Mailbox Interrupt Flag (receive/transmit).

0 No message was transmitted or received.

1 The corresponding mailbox transmitted or received a message successfully.

Each of the six mailboxes may initiate an interrupt. These interrupts can be a receive or a transmit interrupt depending on the mailbox configuration. If one of the configurable mailboxes is configured as Remote Request Mailbox (AAM set) and a remote frame is received, a transmit interrupt is set after sending the corresponding data frame. If a remote frame is sent, a receive interrupt is set after the reception of the desired data frame.

There is one interrupt mask bit for each mailbox. If a message is received, the corresponding RMPn bit in the RCR is set. If a message is sent, the corresponding TAn bit in the TCR register is set. The setting of the RMPn bit or the TAn bit also sets the appropriate MIFx flag in the CAN_IFR register if the corresponding interrupt mask bit is set. The MIFx flag generates an interrupt. The MIMx mask bits in the CAN_IMR register determine if an interrupt can be generated by a mailbox.

Bit 7 **Reserved.**

Bit 6 **RMLIF.** Receive Message Lost Interrupt Flag.

0 No message was lost.

1 An overflow condition has occurred in at least one of the receive mailboxes.

Bit 5 **AAIF.** Abort Acknowledge Interrupt Flag.

0 No transmission was aborted.

1 A "send transmission" operation was aborted.

Bit 4 **WDIF.** Write Denied Interrupt Flag.

0 The write access to the mailbox was successful.

1 The CPU tried to write to a mailbox but was not allowed to.

Bit 3 **WUIF.** Wake-Up Interrupt Flag.

0 The module is still in the sleep mode or in normal operation.

1 The module has left the sleep mode.

Bit 2 **BOIF.** Bus-Off Interrupt Flag.

0 The CAN module is still in the bus-on mode.

1 The CAN has entered the bus-off mode.

Bit 1 **EPIF.** Error Passive Interrupt Flag.

0 The CAN module is not in the error-passive mode.

1 The CAN module has entered the error-passive mode.

Bit 0 **WLIF.** Warning Level Interrupt Flag.

0 None of the error counters has reached the warning level.

1 At least one of the error counters has reached the warning level.

Registre des masques d'interruption CAN : CAN_IMR

15	14	13	12	11	10	9	8
MIL	Reserved	MIM5	MIM4	MIM3	MIM2	MIM1	MIM0
RW-0		RW-0	RW-0	RW-0	RW-0	RW-0	RW-0
7	6	5	4	3	2	1	0
EIL	RMLIM	AAIM	WDIM	WUIM	BOIM	EPIM	WLIM
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

Two additional control bits are included in this register:

Bit 15 **MIL.** Mailbox Interrupt Priority Level.

For the mailbox interrupts MIF5 – MIF0.

0 The mailbox interrupts generate high-priority requests.

1 The mailbox interrupts generate low-priority requests.

Bit 14 **Reserved.**

Bits 13–8 See section 10.6.1, *CAN Interrupt Flag Register (CAN_IFR)*, on page 10-34.

Bit 7 **EIL.** Error Interrupt Priority Level.

For the error interrupts RMLIF, AAIF, WDIF, WUIF, BOIF, EPIF, and WLIF.

0 The named interrupts generate high-priority requests.

1 The named interrupts generate low-priority requests.

Bits 6–0 See section 10.6.1, *CAN Interrupt Flag Register (CAN_IFR)*, on page 10-34.

Exemple de programmation du contrôleur CAN

- Activer le contrôleur et les broches (SCSR1, MCRB)
- Désactiver les Mailboxes (MDER)
- Initialiser l'ID et DLC (MSGIDnL, H et MSGCTRLn)
- Initialiser les masques de filtrage (LAM0(1)L(H))
- Programmer le Bit Timing (voir organigramme)
- Activer la ou les Mailboxes

Envois : - Initialiser les données MBXnA, B, C et D
- activer une requête d'envoi (TCR : bits TRSn)
- attendre l'acceptation de l'envoi (TCR : bits TAn)
- remettre TAn à 0

Réceptions : - attendre une réception (RCR : bits RMPn)
- lire la donnée
- remettre RMPn à 0