UTBM **Département GESC**

Automne 2010 **Durée : 2 heures. Sans documents**

Examen TR57

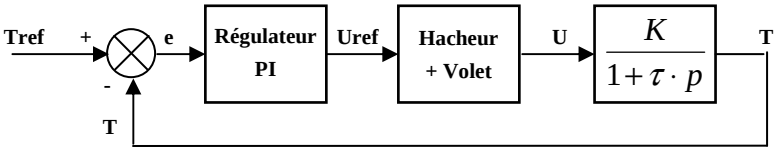
Régulation de la température d'un véhicule par noyau temps réel PICOS

Les différents organes électriques d'un véhicule sont pilotés par un microcontrôleur PIC de la famille dsPIC30F, et le noyau temps réel PICOS.

Le sujet traite dans un premier temps la mise en œuvre d'une seule tâche PICOS (tâche5) permettant de contrôler la température dans l'habitacle du véhicule à l'aide d'un contrôleur PI dont la sortie agit sur la commande d'un hacheur série alimentant un ventilateur et sur la commande d'un volet de sélection tout ou rien air chaud / air froid. La communication par bus CAN entre le dsPIC30F et l'ordinateur de bord est ensuite étudiée.

I Régulation de température

L'asservissement et le modèle de l'ensemble : système de chauffe + habitacle est modélisé de la manière suivante :

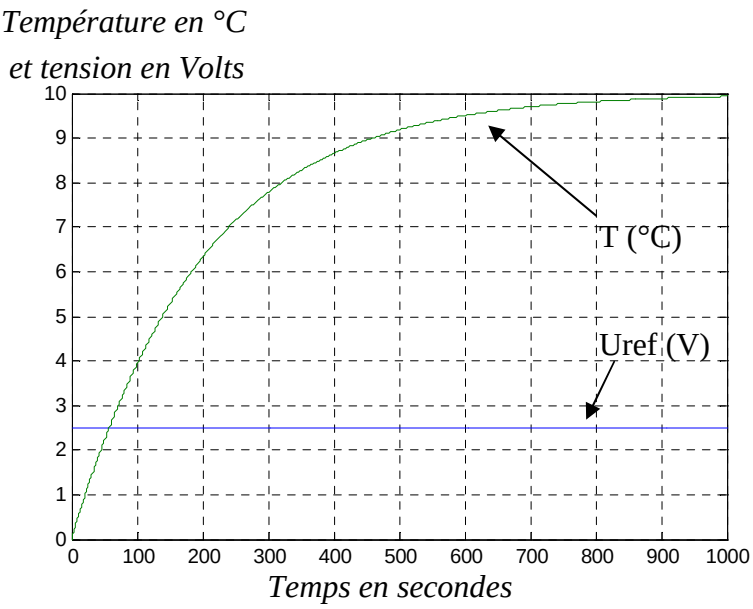


Le hacheur permet de piloter la tension d'alimentation du ventilateur. Elle dépendra du rapport cyclique de la PWM générée par le microcontrôleur. Sa fonction de transfert est assimilée à H(p)=1.
Le volet permet de sélectionnée l'arrivée d'air : chaud (moteur) ou froid (extérieur). Il est piloté par l'intermédiaire d'une sortie numérique du microcontrôleur : 0 : sélection de l'air froid, et 1 : sélection de l'air chaud.
La tension de commande du système de chauffe fonctionne de la manière suivante :

- Si Uref > 0 : Réchauffement de l'habitacle :
  - La consigne du hacheur sera égale à abs(Uref) (= Uref dans ce cas)
  - La consigne du volet sera positionnée sur chaud : sortie numérique à 1
- Si Uref < 0 : Refroidissement de l'habitacle :
  - La consigne du hacheur sera égale à abs(Uref) (= -Uref dans ce cas)
  - La consigne du volet sera positionnée sur froid : sortie numérique à 0

I.1 Identification du système de chauffe

La courbe ci-dessous représente la réponse en température à un échelon de commande Uref = 2.5V en boucle ouverte. En déduire graphiquement les paramètres K et τ du modèle.



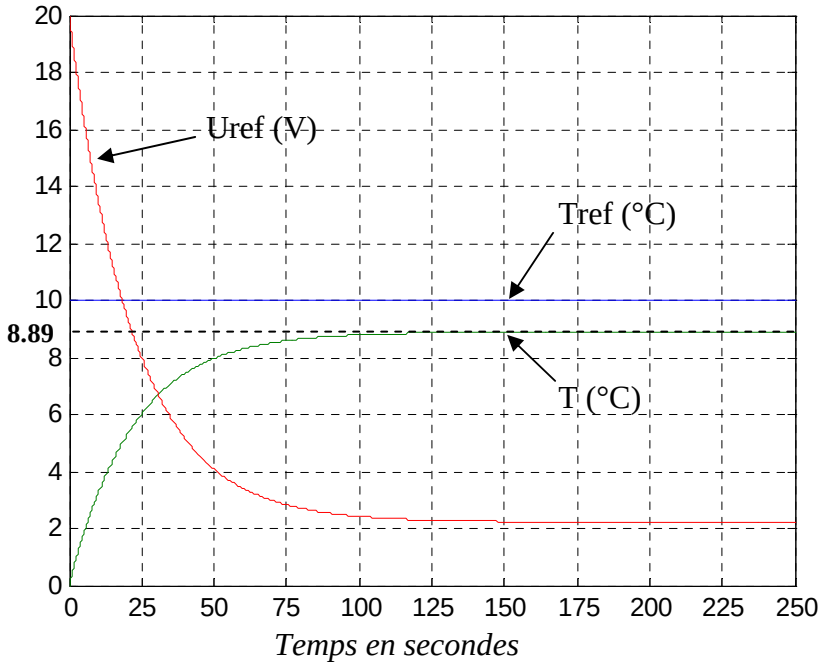
K =

τ =

I.2 Périodicité des tâches d'asservissement (période d'échantillonnage)

Le modèle obtenu précédemment asservi par un correcteur proportionnel de gain Kp = 2 est simulé. La consigne de température Tref est fixée à 10°C. Les courbes sont représentées ci-dessous (Uref est représentée à titre indicatif afin d'évaluer les saturations dues au choix du gain Kp).

Déterminer graphiquement la constante de temps $\tau 0$ et la bande passante F0 ($\omega 0/2\pi$) du système en boucle fermée, ainsi que l'erreur statique $\varepsilon 0$.



$\tau 0$ =

F0 =

$\varepsilon 0$ =

Déterminer l'expression de la fonction de transfert en boucle fermée et en déduire les valeurs littérales et numériques de la bande passante F0 et de la constante de temps $\tau 0$:

$$\frac{T(p)}{Tref(p)} =$$

F0 =

$\tau 0$ =

En déduire la fréquence d'échantillonnage Fe si l'on désire qu'elle soit 25 fois plus grande que la bande passante en boucle fermée. Exprimer également la période d'échantillonnage en millisecondes.

Fe =

Te =

I.3 Alarme permettant la périodicité de la tâche

Pour effectuer la tâche d'asservissement de la température, on utilise une alarme de période Te dont l'évènement posté ASSERVT_EVENT permet de synchroniser l'exécution de la tâche à chaque période Te.

- Sans détailler le code, indiquer quelles sont les étapes permettant de déclarer l'alarme
- Indiquer comment configurer la fréquence de l'alarme à Fe.

I.4 Synthèse du correcteur PI

- Rappeler la fonction de transfert en p du correcteur PI parallèle Uref(p)/e(p).
- Déterminer la fonction de transfert en z Uref(z)/e(z).
- En déduire l'équation récurrente exprimant Uref(n).

I.5 Ecriture de la tâche5 réalisant l'asservissement

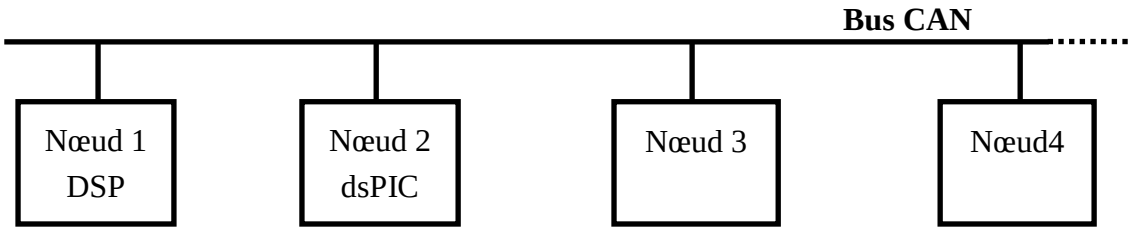
Le calcul de l'asservissement se compose des étapes séquentielles suivantes :

- Lancement des conversions analogiques numériques : fonction *lance_AD()*
- Attente de la fin des conversions : fonction *wait_AD()*
- Lecture des résultats de conversion Tref (registre ADCBUF0) et T (registre ADCBUF1)
- Calcul de Uref à partir de l'équation récurrente (la mise à l'échelle des grandeurs sera considérée comme déjà réalisée grâce aux valeurs des gains du correcteur)
- Pilotage du volet d'arrivée d'air : fonction *volet(x)* avec x = 0 : air froid et x = 1 : air chaud.
- Pilotage du ventilateur en agissant sur le rapport cyclique PDC1 qui est l'image de Uref.

Ecrire le code associé à la tâche5 en utilisant les fonctions données ci-dessus et sans tenir compte des initialisations des périphériques du microcontrôleur. Aucune interruption n'est utilisée pour cette tâche.

II Transfert de données par bus CAN

On étudie la communication par bus CAN entre l'ordinateur de bord (nœud 1) et le régulateur de température (nœud 2). Le nœud 1 est constitué d'un DSP contrôleur TMS320LF2407A cadencé à 40 MHz. Le nœud 2 est constitué du dsPIC30F4011 sur lequel est implantée la régulation de la température de l'habitacle. Seule la programmation du nœud 1 est étudiée.



Principe de communication :

- Le nœud 1 effectue la récupération d'informations depuis les autres nœuds à partir du bus CAN afin d'assurer la centralisation de la gestion du véhicule. Les échanges entre le nœud 1 et le nœud 2 s'effectuent à la fréquence de 10 Hz conformément au principe suivant : le nœud 1 envoie une trame de requête, le nœud 2 répond en envoyant une trame de 8 octets comprenant 4 mots de 16 bits (Tref, T, Uref et Iv le courant dans le moteur du ventilateur)
- Les transmissions CAN s'effectuent au standard 2.0A à 1Mbauds.
- Le nœud 1 utilise la mailbox 2 pour envoyer la trame de requête et recevoir la réponse du nœud 2.
- La trame de requête envoyée par le nœud 1 au nœud 2 a l'identificateur 11 bits 0x400.
- Aucune autre trame de requête d'identificateur 0x400 n'est envoyée tant que le nœud 2 n'a pas répondu. Le sémaphore *T_ok* permet de mémoriser une réception correcte.
- La réception des trames par le nœud 1 s'effectue par interruption (CANMBINT -> INT1) et mise en buffer circulaire.

II.1 Filtrage des messages

Dans les deux cas suivants, indiquer quels identificateurs seront acceptés par la mailbox de réception numéro 2 configurée avec l'identificateur 0x400 :

- lorsque les masques de filtrage ne sont pas utilisés
- lorsque le masque de filtrage suivant est utilisé : LAM1_H = 0x100C

II.2 Arbitrage

En cas d'émissions simultanées de plusieurs autres nœuds, indiquer quels seront les messages prioritaires par rapport à celui d'identificateur 0x400.

II.3 Configuration du contrôleur CAN du DSP

Ecrire en langage C la fonction *void initCAN(void)* réalisant l'activation, l'initialisation du contrôleur CAN (aucun masque de filtrage n'est utilisé) ainsi que la déclaration et l'initialisation du buffer circulaire de réception.

II.4 Configuration du Timer 3

Le Timer 3 du TMS320LF2407A est utilisé pour déclencher une interruption (T3PINT->INT2) à la fréquence de 10 Hz. Ecrire en langage C la fonction *void initTimer3(void)* réalisant l'activation et l'initialisation du Timer 3.

II.5 Emission des requêtes par le nœud 1

Ecrire en langage C le sous-programme d'interruption *interrupt void interT3PINT(void)* réalisant la transmission de la trame de requête à la fréquence de 10 Hz.

II.6 Réception des trames par interruption et mise en buffer circulaire

Ecrire en langage C le sous-programme d'interruption *interrupt void interT3PINT(void)* réalisant la réception et le stockage dans le buffer circulaire des données reçues sur le bus CAN (on considère pour simplifier que toutes les données reçues proviennent de la mailbox2).

II.7 Lecture des trames envoyées par le nœud 2

Ecrire en langage C le programme principal réalisant la lecture des paramètres Tref, T, Uref et Iv stockés dans le buffer de réception (le traitement des données n'est pas traité ici).

II.8 Bande passante occupée par les transferts noeud1 – nœud 2

Déterminer le pourcentage de la bande passante utilisé par les seuls transferts entre le nœud 1 et le nœud 2.

Annexes

—	CLKSRC	LPM1	LPM0	CLK PS2	CLK PS1	CLK PS0	—	SCSR1
ADC CLKEN	SCI CLKEN	SPI CLKEN	CAN CLKEN	EVB CLKEN	EVA CLKEN	—	ILLADR	

Registres du contrôleur CAN

Registres CAN

Address	Name	Description
7100h	MDER	Mailbox Direction/Enable Register (bits 7 to 0)
7101h	TCR	Transmission Control Register (bits 15 to 0)
7102h	RCR	Receive Control Register (bits 15 to 0)
7103h	MCR	Master Control Register (bits 13 to 6, 1, 0)
7104h	BCR2	Bit Configuration Register 2 (bits 7 to 0)
7105h	BCR1	Bit Configuration Register 1 (bits 10 to 0)
7106h	ESR	Error Status Register (bits 8 to 0)
7107h	GSR	Global Status Register (bits 5 to 3, 1, 0)
7108h	CEC	CAN Error Counter Register (bits 15 to 0)
7109h	CAN_IFR	Interrupt Flag Register (bits 13 to 8, 6 to 0)
710Ah	CAN_IMR	Interrupt Mask Register (bits 15, 13 to 0)
710Bh	LAM0_H	Local Acceptance Mask for MBOX0 and 1 (bits 31, 28 to 16)
710Ch	LAM0_L	Local Acceptance Mask for MBOX0 and 1 (bits 15 to 0)
710Dh	LAM1_H	Local Acceptance Mask for MBOX2 and 3 (bits 31, 28 to 16)
710Eh	LAM1_L	Local Acceptance Mask for MBOX2 and 3 (bits 15 to 0)
710Fh	Reserved	Accesses assert the CAADDRx signal from the CAN peripheral (which will assert an Illegal Address error)

Note: All unimplemented register bits are read as zero; writes have no effect. All register bits are initialized to zero unless otherwise stated in the definition.

Adresses des boîtes à lettres (Mailbox)

	Mailboxes					
Registers	**MBOX 0**	**MBOX 1**	**MBOX 2**	**MBOX 3**	**MBOX 4**	**MBOX 5**
MSG ID*n*L	7200	7208	7210	7218	7220	7228
MSG ID*n*H	7201	7209	7211	7219	7221	7229
MSG CTRL*n*	7202	720A	7212	721A	7222	722A
			Reserved			
MBX*n*A	7204	720C	7214	721C	7224	722C
MBX*n*B	7205	720D	7215	721D	7225	722D
MBX*n*C	7206	720E	7216	721E	7226	722E
MBX*n*D	7207	720F	7217	721F	7227	722F

Initialisation des Mailbox

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.

Adresses de stockage des données des Mailboxes en RAM

Register/Databyte	Address	Register/Databyte	Address
MSG – ID0H	7201h	MSG – ID0L	7200h
Reserved	7203h	MSG – CTRL0	7202h
Databyte 0, Databyte 1 (DBO = 1) Databyte 3, Databyte 2 (DBO = 0)	7205h (B)	Databyte 2, Databyte 3 (DBO = 1) Databyte 1, Databyte 0 (DBO = 0)	7204h (A)
Databyte 4, Databyte 5 (DBO = 1) Databyte 7, Databyte 6 (DBO = 0)	7207h (D)	Databyte 6, Databyte 7 (DBO = 1) Databyte 5, Databyte 4 (DBO = 0)	7206h (C)
MSG – ID1H	7209h	MSG – ID1L	7208h
Reserved		MSG – CTRL1	720Ah
Databyte 0, Databyte 1 (DBO = 1) Databyte 3, Databyte 2 (DBO = 0)	720Dh	Databyte 2, Databyte 3 (DBO = 1) Databyte 1, Databyte 0 (DBO = 0)	720Ch
Databyte 4, Databyte 5 (DBO = 1) Databyte 7, Databyte 6 (DBO = 0)	720Fh	Databyte 6, Databyte 7 (DBO = 1) Databyte 5, Databyte 4 (DBO = 0)	720Eh
•			
•			
•			
•			
•			
•			
MSG – ID5H	7229h	MSG – ID5L	7228h
Reserved		MSG – CTRL5	722Ah
Databyte 0, Databyte 1 (DBO = 1) Databyte 3, Databyte 2 (DBO = 0)	722Dh	Databyte 2, Databyte 3 (DBO = 1) Databyte 3, Databyte 2 (DBO = 0)	722Ch
Databyte 4, Databyte 5 (DBO = 1) Databyte 7, Databyte 6 (DBO = 0)	722Fh	Databyte 6, Databyte 7 (DBO = 1) Databyte 5, Databyte 4 (DBO = 0)	722Eh

† The DBO (data byte order) bit is located in the MCR register and is used to define the order in which the data bytes are stored in the mailbox when received and the order in which the data bytes are transmitted. Byte 0 is the first byte in the message and Byte 7 is the last one shown in the CAN message.

Configuration des Mailboxes

Modes de configuration

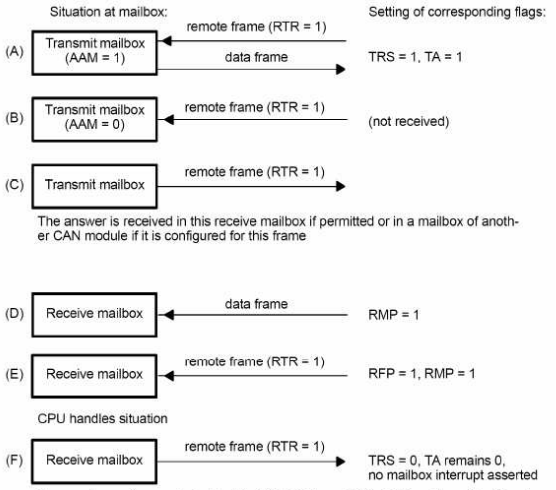
A mailbox can be configured in four different ways:

- ❑ Transmit mailbox (mailboxes 4 and 5 or 2 and 3 configured as transmit) can only transmit messages.
- ❑ Receive mailbox (mailboxes 0 and 1) can only receive messages.
- ❑ Mailboxes 2 and 3 configured as receive mailboxes can transmit a remote request frame and wait for the corresponding data frame if the TRS bit is set.
- ❑ Mailboxes 2 and 3 configured as transmit mailboxes can transmit a data frame wherever a remote request frame is received for the corresponding identifier, if the AAM bit is set.

Note:

After successful transmission of a remote frame, the TRS bit is reset but no transmit acknowledge (TA) or mailbox interrupt flag is set.

Trames de requête



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 contrôle des transmissions : TCR

15	14	13	12	11	10	9	8
TA5	TA4	TA3	TA2	AA5	AA4	AA3	AA2
RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0
7	6	5	4	3	2	1	0
TRS5	TRS4	TRS3	TRS2	TRR5	TRR4	TRR3	TRR2
RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0

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

Bits 15–12 **TAn.** Transmission Acknowledge (for mailbox n).

If the message in mailbox n was sent successfully, bit TAn is set.

Bits TAn are reset by writing a 1 from the CPU. This also clears the interrupt if an interrupt was generated. Writing a 0 has no effect. If the CPU tries to reset the bit while the CAN tries to set it, the bit is set.

These bits set a mailbox interrupt flag (MIFx) in the IF register. The MIFx bits initiate a mailbox interrupt if enabled; that is, if the corresponding interrupt mask bit in the IM register is set.

Bits 11–8 **AAn.** Abort Acknowledge (for mailbox n).

If transmission of the message in mailbox n is aborted, bit AAn is set and the AAIF bit in the IF register is set. The AAIF bit generates an error interrupt if enabled.

Bits AAn are reset by writing a 1 from the CPU. Writing a 0 has no effect. If the CPU tries to reset a bit and the CAN tries to set the bit at the same time, the bit is set.

Bits 7–4 **TRSn.** Transmission Request Set (for mailbox n).

In order to initiate a transfer, the TRSn bit has to be set in the TCR register. After this, the entire transmission procedure and possible error handling is done without any CPU involvement.

If TRSn is set, write access to the corresponding mailbox is denied, and the message in mailbox *n* will be transmitted. Several TRS bits can be set simultaneously.

TRS bits can be set by the CPU (user) or the CAN module and reset by internal logic. If the CPU tries to set a bit while the CAN tries to clear it, the bit is set. TRS bits are set by the user writing a 1. Writing a 0 has no effect.

In the event of a remote frame request, the TRS bits are set by the CAN module for mailboxes 2 and 3.

The TRSn bits are reset after a successful or an aborted transmission (if an abort is requested).

A write to a mailbox with TRS set will have no effect and will generate the WDIF interrupt if enabled. A successful transmission initiates a mailbox interrupt, if enabled.

TRS bits are used for mailboxes 4 and 5, and also for 2 and 3 if they are configured for transmission.

Bits 3–0 **TRRn.** Transmission Request Reset (for mailbox n).

TRR bits can only be set by the CPU (user) and reset by internal logic. In case the CPU tries to set a bit while the CAN module tries to clear it, the bit is set. The TRR bits are set by the user writing a 1. Writing a 0 has no effect.

If TRRn is set, write access to the corresponding mailbox*n* is denied. A write access will initiate a WDIF interrupt, if enabled. If TRRn is set and the transmission which was initiated by TRSn is not currently processed, the corresponding transmission request will be cancelled. If the corresponding message is currently processed, this bit is reset in the event of:

1) A successful transmission
2) An abort due to a lost arbitration
3) An error condition detected on the CAN bus line

If the transmission is successful, the status bit TAn is set. If the transmission is aborted, the corresponding status bit AAn is set. In case of an error condition, an error status bit is set in the ESR.

The status of the TRR bits can be read from the TRS bits. For example, if TRS is set and a transmission is ongoing, TRR can only be reset by the actions described above. If the TRS bit is reset and the TRR bit is set, no effect occurs because the TRR bit will be immediately reset.

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.

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 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:

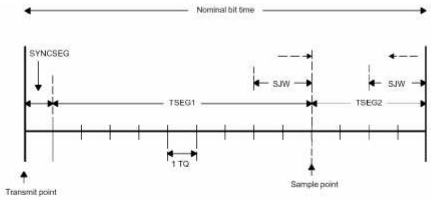
$$(SJW + 1) \leq TSEG2 \leq 8.$$

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.**

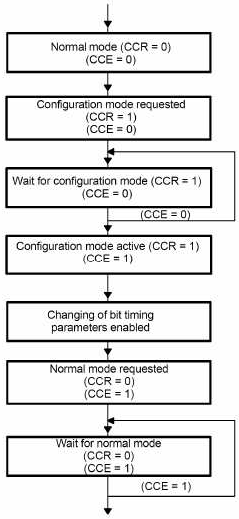
Bits 7–0 **BRP.** Baud Rate Prescaler.

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



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

Registre de contrôle général des timers GPTCONA

15	14	13	12–11	10–9	8–7
Reserved	T2STAT	T1STAT	Reserved	T2TOADC	T1TOADC
RW-0	R-1	R-1	RW-0	RW-0	RW-0

6	5–4	3–2	1–0
TCOMPOE	Reserved	T2PIN	T1PIN
RW-0	RW-0	RW-0	RW-0

Note: R = Read access, W = Write access, -n = value after reset

Bit 15 **Reserved**. Reads return zero; writes have no effect.

Bit 14 **T2STAT**. GP timer 2 Status. Read only.
- 0 Counting downward
- 1 Counting upward

Bit 13 **T1STAT**. GP timer 1 Status. Read only.
- 0 Counting downward
- 1 Counting upward

Bits 12–11 **Reserved**. Reads return zero; writes have no effect.

Bits 10–9 **T2TOADC**. Start ADC with timer 2 event.
- 00 No event starts ADC
- 01 Setting of underflow interrupt flag starts ADC
- 10 Setting of period interrupt flag starts ADC
- 11 Setting of compare interrupt flag starts ADC

Bits 8–7 **T1TOADC**. Start ADC with timer 1 event.
- 00 No event starts ADC
- 01 Setting of underflow interrupt flag starts ADC
- 10 Setting of period interrupt flag starts ADC
- 11 Setting of compare interrupt flag starts ADC

Bit 6 **TCOMPOE**. Compare output enable. If $\overline{\text{PDPINTx}}$ is active this bit is set to zero.
- 0 Disable all GP timer compare outputs (all compare outputs are put in the high-impedance state)
- 1 Enable all GP timer compare outputs

Bits 5–4 **Reserved**. Reads return zero; writes have no effect.

Bits 3–2 **T2PIN**. Polarity of GP timer 2 compare output.
- 00 Forced low
- 01 Active low
- 10 Active high
- 11 Forced high

Bits 1–0 **T1PIN**. Polarity of GP timer 1 compare output.
- 00 Forced low
- 01 Active low
- 10 Active high
- 11 Forced high

Registre de contrôle individuel des timers : TxCON (x=1, 2, 3 ou 4)

15	14	13	12	11	10	9	8
Free	Soft	Reserved	TMODE1	TMODE0	TPS2	TPS1	TPS0
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

7	6	5	4	3	2	1	0
T2SWT1/ T4SWT3†	TENABLE	TCLKS1	TCLKS0	TCLD1	TCLD0	TECMPR	SELT1PR/ SELT3PR†
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

Note: R = Read access, W = Write access, -0 = value after reset
† Reserved in T1CON andT3CON

Bits 15–14 **Free, Soft.** Emulation control bits.
- 00 Stop immediately on emulation suspend
- 01 Stop after current timer period is complete on emulation suspend
- 10 Operation is not affected by emulation suspend
- 11 Operation is not affected by emulation suspend

Bit 13 **Reserved**. Reads return zero, writes have no effect.

Bits 12–11 **TMODE1–TMODE0.** Count Mode Selection.
- 00 Stop/Hold
- 01 Continuous-Up/-Down Count Mode
- 10 Continuous-Up Count Mode
- 11 Directional-Up/-Down Count Mode

Bits 10–8 **TPS2–TPS0.** Input Clock Prescaler.

000	x/1	100	x/16
001	x/2	101	x/32
010	x/4	110	x/64
011	x/8	111	x/128

x = device (CPU) clock frequency

Bit 7 **T2SWT1.** In the case of EVA, this bit is T2SWT1. (GP timer 2 start with GP timer 1.) Start GP timer 2 with GP timer 1's timer enable bit. This bit is reserved in T1CON.
T4SWT3. In the case of EVB, this bit is T4SWT3. (GP timer 4 start with GP timer 3.) Start GP timer 4 with GP timer 3's timer enable bit. This bit is reserved in T3CON.
- 0 Use own TENABLE bit
- 1 Use TENABLE bit of T1CON (in case of EVA) or T3CON (in case of EVB) to enable and disable operation ignoring own TENABLE bit

Bit 6 **TENABLE**. Timer enable.
- 0 Disable timer operation (the timer is put in hold and the prescaler counter is reset)
- 1 Enable timer operations

Bits 5–4 **TCLKS1, TCLKS0.** Clock Source Select.

5	4	Source
0	0	Internal
0	1	External
1	0	Reserved
1	1	QEP Circuit† (in case of Timer 2/Timer 4)
		Reserved (in case of Timer 1/Timer 3)

† This option is valid only if SELT1PR = 0

Bits 3–2 **TCLD1, TCLD0**. Timer Compare Register Reload Condition.
- 00 When counter is 0
- 01 When counter value is 0 or equals period register value
- 10 Immediately
- 11 Reserved

Bit 1 **TECMPR**. Timer compare enable.
- 0 Disable timer compare operation
- 1 Enable timer compare operation

Bit 0 **SELT1PR**. In the case of EVA, this bit is SELT1PR (Period register select). When set to1 in T2CON, the period register of Timer 1 is chosen for Timer 2 also, ignoring the period register of Timer 2. This bit is a reserved bit in T1CON.
SELT3PR. In the case of EVB, this bit is SELT3PR (Period register select). When set to1 in T4CON, the period register of Timer 3 is chosen for Timer 4 also, ignoring the period register of Timer 4. This bit is a reserved bit in T3CON.
- 0 Use own period register
- 1 Use T1PR (in case of EVA) or T3PR (in case of EVB) as period register ignoring own period register

Registre A des Flags d'interruption de l'EVA EVAIFRA

15–11	10	9	8
Reserved	T1OFINT FLAG	T1UFINT FLAG	T1CINT FLAG
R-0	RW1C-0	RW1C-0	RW1C-0

7	6–4	3	2	1	0
T1PINT FLAG	Reserved	CMP3INT FLAG	CMP2INT FLAG	CMP1INT FLAG	PDPINTA FLAG
RW1C-0	R-0	RW1C-0	RW1C-0	RW1C-0	RW1C-0

Note: R = Read access, W1C = Write 1 to clear, -0 = value after reset

Bits 15-11 et 6-4

Reserved. Reads return zero; writes have no effect.

Bits 10-7 et 3-2

Read: 0 Flag is reset
1 Flag is set
Write: 0 No effect
1 Resets flag

Registres d'interruption des EVA et EVB

Flag Register	Mask Register	EV Module
EVAIFRA	EVAIMRA	EVA
EVAIFRB	EVAIMRB	
EVAIFRC	EVAIMRC	
EVBIFRA	EVBIMRA	EVB
EVBIFRB	EVBIMRB	
EVBIFRC	EVBIMRC	