PIC18 : Le module ECAN

Pour les PIC18Fxx8 et/ou PIC18Fxxx8

Introduction

Cette fiche est destinée à la description des modules ECAN présents sur les PIC 18Fxx8 et 18Fxx8x.

Ce module permet le pilotage d'un bus CAN à partir d'un tel PIC.

Le module ECAN

ECAN pour Enhanced Controller Area Network. Ce module est un contrôleur de communications implémentant le protocole ***CAN2.0 dans ses versions A ou B*** (selon les spécifications définies par Bosh). Ce module permet aussi le support des protocoles CAN 1.2, CAN 2.0B « active » et CAN 2.0B « passive ».

Les principales caractéristiques du module ECAN sont :

- ✗ Implémentation des versions 1.2, 2.0A et 2.0B du protocole CAN
- ✗ Support du filtrage de données DeviceNet^{tm}
- ✗ Support des trames de données standard et étendues
- ✗ Taille des données comprise entre 0 et 8 octets
- ✗ Débit programmmable, jusqu'à 1 Mbps
- ✗ Trois modes de fonctionnement :
  - ✗ Mode 0 : Legacy mode (fonctionnement classique)
  - ✗ Mode 1 : Enhanced Legacy Mode with DeviceNet support
  - ✗ Mode 2 : FIFO mode with DeviceNet support
- ✗ Support de la gestion automatisée des « remote frames »
- ✗ Récepteur à « double buffer avec gestion des priorités »
- ✗ Six buffers programmables comme buffers d'émission ou de réception
- ✗ Plusieurs filtres et masques permettant de gérer l'acceptation des trames
- ✗ ...

« Overview »

Le module ECAN est en fait un gestionnaire de protocole associé à un système de de stockage et de gestion de messages. Le gestionnaire de protocole CAN fournit les fonctionnalités permettant de recevoir et d'émettre des messages sur le bus. Les messages sont émis après avoir chargé les registres adéquats avec les valeurs adéquates. La vérification des états, gestion des erreurs se fait à travers des registres spécifiques.

Chaque message détecté sur le bus CAN est contrôlé par rapport à d'éventuelles erreurs, puis comparé aux différents filtres afin de déterminer s'il doit être reçu et stocké dans un des deux registres de réception.

Le module CAN gère les types de trames suivants :

- ✗ Trame de données standard : Standard Data Frame
- ✗ Trame de données étendue : Extended Data Frame
- ✗ Remote Frame
- ✗ Trame d'erreur : Error Frame
- ✗ Overload Frame Reception
- ✗ Interframe Space Generation/Detection

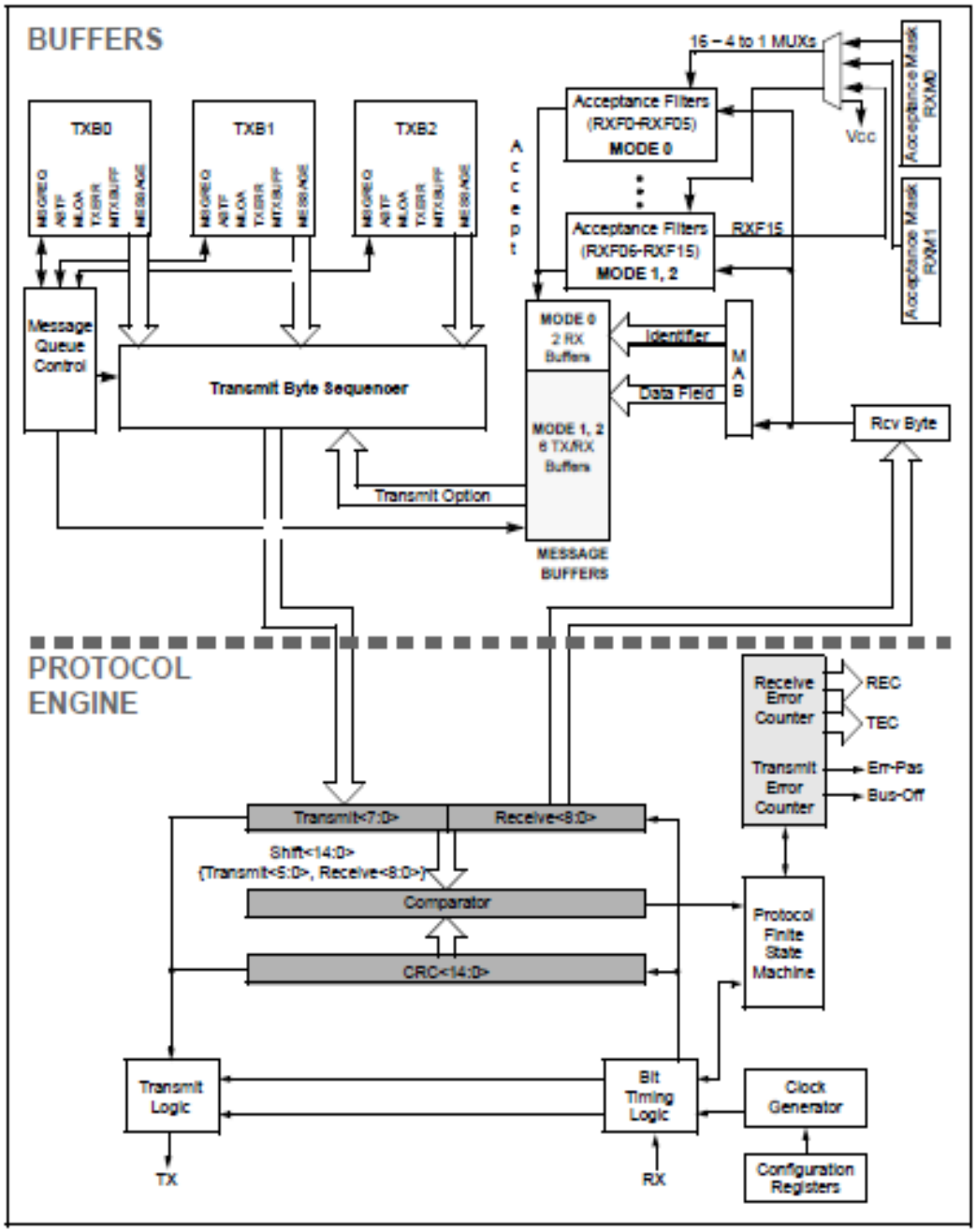
Le module CAN utilise les lignes RB2/CANTX et RB3/CANRX pour interfacer le bus. En fonctionnement normal, le module CAN gère le bit TRISB2. ***L'utilisateur doit s'assurer que le bit TRISB3 est positionné à '1'***.

Mise en oeuvre du module CAN

La procédure suivante doit être respectée pour que le module CAN soit capable d'émettre et de recevoir des données CAN :

1. Positionner le module ECAN dans le mode configuration.
2. Choisir le mode de fonctionnement (0, 1 ou 2), 0 (Legacy) est choisi par défaut.
3. Configurer les registres fixant le débit (baud rate registers).
4. Configurer les registres associés aux filtres et aux masques.
5. Positionner le module ECAN dans le mode « Normal » (ou dans un des autres modes spécifiques, en fonction des besoins de l'application).

Schéma bloc du module ECAN



Principaux modes

Le module ECAN propose six principaux modes « of operation » (à ne pas confondre avec les 3 modes de fonctionnement 0,1 ou 2, qui font partie du mode « normal ») :

- Configuration Mode
- Disable mode
- Normal Operation mode
- Listen Only mode
- Loopback mode
- Error Recognition mode

Mis à part le « Error Recegnition mode », chacun des modes est sélectionné à l'aide des bits ***REQOP*** du registre ***CANCON*** (CANCON<7:5>). Le mode « Error Recognition » est quant à lui sélectioné par le bit ***RXM*** du (des) registre(s) « ***Receive Buffer*** ».

L'accès à un mode se vérifie à l'aide des bits ***OPMODE***. En effet, un changement de mode n'est réellement effectué que lorsque toutes les transmissions en attente sont terminées. L'utilisateur doit donc prendre la précaution de vérifier si le changement de mode est effectif.

Configuration Mode

Il est nécessaire d'initialiser le module ECAN, ceci se fait en étant dans ce mode. L'accès au mode de configuration est demandé en positionnant le bit ***REQOP2 à 1***. La configuration peut commencer dès que le bit ***OPMODE2*** est détecté à 1. Une fois cet état atteint, il est possible d'écrire dans les registres de configuration, les registres « filtres », « masques »... Le ***module est réactivé*** en positionnant les bits ***REQOP à 0***.

En dehors du mode de configuration, il est impossible de modifier les registres :

- De configuration
- De sélection du mode de fonctionnement
- De configuration de débit
- De configuration des filtres et masques

Dans ce mode, le module n'émet ni ne reçoit aucune donnée, les compteurs d'erreurs sont remis à 0 et les flags d'interruption sont inchangés.

Disable Mode

Dans ce mode, le module ne transmet ni ne reçoit. Par contre, le module peut positionner à 1 le bit ***WAKIF*** en cas d'activité sur le bus. Les flags d'interruption ainsi que les compteurs d'erreurs restent inchangés.

L'accès au mode « Disable » se fait en positionnant les bits ***REQOP<2:0> à 001***.

Normal Mode

C'est le fonctionnement standard du module. Dans ce mode, le périphérique surveille activement les messages sur le bus, génère les acquittements, les trames d'erreurs... C'est uniquement dans ce mode qu'il sera possible d'émettre des données sur le bus.

Listen Only Mode

Dans ce cas de figure, le module reçoit tous les messages, y compris ceux contenant des erreurs. Ce mode est utile pour surveiller, débugger une application... C'est aussi par ce mode qu'il faut passer pour mettre en place un système « hot plug » avec détection automatique du débit.

Loopback Mode

Dans ce mode, il est possible de transférer directement le contenu d'un buffer d'émission vers un buffer de réception, sans passer par le bus. Ce mode est utilisé dans des situations de test.

Error Recognition Mode

Le module peut être configuré pour ignorer les erreurs. C'est l'objet de ce mode.

Modes de fonctionnement

En plus des différents « modes » utilisables, le module ECAN propose 3 modes de fonctionnement. Ils sont définis comme mode 0, 1 et 2.

Ce document ne traitera que du ***mode 0 : Legacy Mode***. Ce mode est le mode de fonctionnement par défaut, il est sélectionné après un RESET. Les ressources suivantes sont disponibles dans le mode 0 :

- Trois buffers d'émission : ***TXB0, TXB1 et TXB2***.
- Deux buffers de réception : ***RXB0 et RXB1***.
- Un masque d'acceptation par buffer de réception : ***RXM0 et RXM1***.
- Six filtres associés : ***deux pour RXB0 et quatre pour RXB1*** : ***RXF0 à RXF5***.

Buffers de messages

Les PIC 18Fxx8x ***implémentent trois buffers d'émission dédiés*** : TXB0, TXB1 et TXB2. Chacun de ces buffers occupe 14 octets de SRAM et est « mappé » parmi les registres SFR. Ces buffers sont les seuls accessibles en mode 0.

Chaque buffer d'émission est composé de :

- Un registre de contrôle : ***TXBnCON***
- Quatre registres identifiants : ***TXBnSIDL, TXBnSIDH, TXBnEIDL et TXBnEIDH***.
- Un registre compteur de longeur de donnée : ***TXBnDLC***
- Huit registres de données : ***TXBnDm***

Les buffers de réception (dédiés) sont au nombre de ***deux : RXB0 et RXB1***. Là encore, chaque buffer occupe 14 octets de RAM dans la zone SFR et ces deux buffers sont les seuls accessibles en mode 0.

Chaque buffer de réception des composé de :

- Un registre de contrôle : ***RXBnCON***
- Quatre registres identifiants : ***RXBnSIDL, RXBnSIDH, RXBnEIDL et RXBnEIDH***.
- Un registre compteur de longeur de donnée : ***RXBnDLC***
- Huit registres de données : ***RXBnDm***

On peut aussi noter la présence d'un buffer d'assemblage de message (Message Assembly Buffer : MAB). Le MAB est destiné à récupérer les messages sur le bus, les assembler et les transférer vers un registre de réception si le message satisfait aux filtres et masques.

Emission d'un message vers le bus CAN

Pour que le processeur puisse écrire dans un buffer message, le ***bit TXREQ doit être à 0***. Cela veut dire qu'il n'y a aucun message en attente sur ce buffer. Les registres ***SIDH, SIDL et DLC***, au minimum, doivent être renseignés. Si des données sont à envoyer, les registres de données doivent aux aussi être renseignés. De même, si un identifiant étendu est utilisé, les registres ***EIDH et EIDL*** sont utilisés et le bit ***EXIDE positionné à 1***.

Pour démarrer une émission, le bit ***TXREQ*** correspondant au buffer utilisé doit être ***positionné à 1***. A ce moment, les bits ***TXABT, TXLARB et TXERR sont mis à 0***. Une émission est considérée comme réussie s'il y a au moins un noeud compatible (débit identique) sur le réseau.

Remarque :

Le fait de positionner TXREQ à 1 ne suffit pas à démarrer une émission. En fait, cela ne fait que qu'indiquer qu'un message est prêt à être émis. La transmission ne débutera réellement que lorsque le medium sera détecté libre.

Lorsque la transmission est terminée (avec succès), ***TXREQ repasse à 0***, ***TXBnIF passe à 1***. Une ***interruption est donc générée si TXBnIE est égal à 1***.

Dans le cas où la transmission échoue, TXREQ reste à 1 (indiquant que le message est toujours en attente d'émission) et un des drapeaux suivants est positionné :

- Si le message a commencé à être émis mais a rencontré une erreur, ***TXERR et IRXIF*** passent à 1
- Si le message a ***perdu un arbitrage, le bit TXLARB est positionné à 1***.

Priorités des buffers d'émission

Les buffers d'émission peuvent se voir affecter une priorité. Celle-ci n'a rien à voir avec la priorité du message (inscrite dans l'identifiant). Le buffer de transmission ayant la priorité la plus élevée sera traité le premier. Quatre niveaux de priorité existent et sont programmés avec les bits ***TXP***. TXP= « 11 » représente le niveau de priorité le plus élevé, tandis que TXP= « 00 » correspond à la priorité la plus basse.

Dans le cas où deux buffers ont le même niveau de priorité, c'est le buffer ayant le plus haut « numéro » qui est traité d'abord.

Réception d'un message

Lorsqu'un message est stocké dans un des buffers de réception (après être passé par le MAB, les filtres...) le ***bit RXFUL correspondant passe à 1***. Ce bit doit être repositionné à 0 après que les données ont été traitées pour permettre la réception d'un nouveau message. Si les interruptions en réception sont autorisées, une interruption est générée.

Une fois que le message est stocké dans un buffer de réception, il faut que le programme détermine quel filtre a permis la réception. Ceci est fait en testant les « Filter Hit Bits » du registre RXBnCON ou BnCON.

Dans le mode 0, les bits ***FILHIT<3:0> du registre RXBnCON*** servent de « Filter Hit Bits ».

Un message reçu est considéré comme ayant un identifiant standard si le bit ***EXID du registre RXBnSIDL est à 0 ou si le registre BnSIDL est à 0***. A contrario, le bit ***EXID à 1*** indique un message au format ***étendu***.

Dans le cas d'un message ***standard, l'identifiant est contenu dans SIDL et SIDH*** . Pour un message au format ***d'identifiant étendu***, il faut lire ***SIDL, SIDH, EIDL et EIDH***.

Si le registre ***RXBnDLC*** (ou BnDLC) contient une valeur différente de 0, il faut ***lire les données reçues dans les registres RXBnDm*** (ou BnDm).

Filtres d'acceptation des messages et masques

Ces éléments permettent de déterminer si un message reçu par le MAB doit être transféré dans un buffer de réception. Lorsqu'un message est reçu par le MAB, le champ « Identifiant » du message est comparé avec les valeurs des filtres. En cas de correspondance, le message sera transféré dans un buffer de réception.

Les masques associés aux filtres spécifient quels sont les bits de l'identifiant à examiner.

La table suivante montre la technique de comparaison utilisée ainsi que l'action correspondante :

Mask bit n	Filter bit n	Message Identifier bit n001	Accept or Reject bit n
0	x	x	Accept
1	0	0	Accept
1	0	1	Reject
1	1	0	Reject
1	1	1	Accept

Legend: x = don't care

Le masque sert principalement à déterminer quels sont les bits à examiner : un bit du masque à 0 indique une acceptation sans tenir compte du filtre.

Si le bit du masque est égal à 1, un bit sera accepté seulement si le bit du filtre lui correspondant possède la même valeur que lui.

Dans le mode 0, les filtres d'acceptation ***RXF0 et RXF1*** ainsi que le masque ***RXM0*** sont associés au ***buffer RXB0***.

Les filtres ***RXF2, RFX3, RFX4 et RFX5*** ainsi que les masque ***RXM1*** sont associés à ***RXB1***.

Lorsqu'un message et un filtre concordent, le message est transféré dans le buffer de réception et le ***numéro du filtre ayant accepté le message*** est chargé dans les bits ***FILHIT***. Dans le ***mode 0, pour RXB1***, le registre ***RXB1CON*** contient les ***bits FILHIT<2:0>***. Le codage employé est le suivant :

- FILHIT<2:0> = 000 : Filtre 0 (RXF0)
- FILHIT<2:0> = 001 : Filtre 1 (RXF1)
- FILHIT<2:0> = 010 : Filtre 2 (RXF2)
- FILHIT<2:0> = 011 : Filtre 3 (RXF3)

- FILHIT<2:0> = 100 : Filtre 4 (RXF4)
- FILHIT<2:0> = 101 : Filtre 5 (RXF5)

Remarque :

Les valeur 000 et 001 ne peuvent apparaître que si le bit RXB0DBEN est à 1 dans RXB0CON, permettant les messages dans RXB0 à passer dans RXB1.

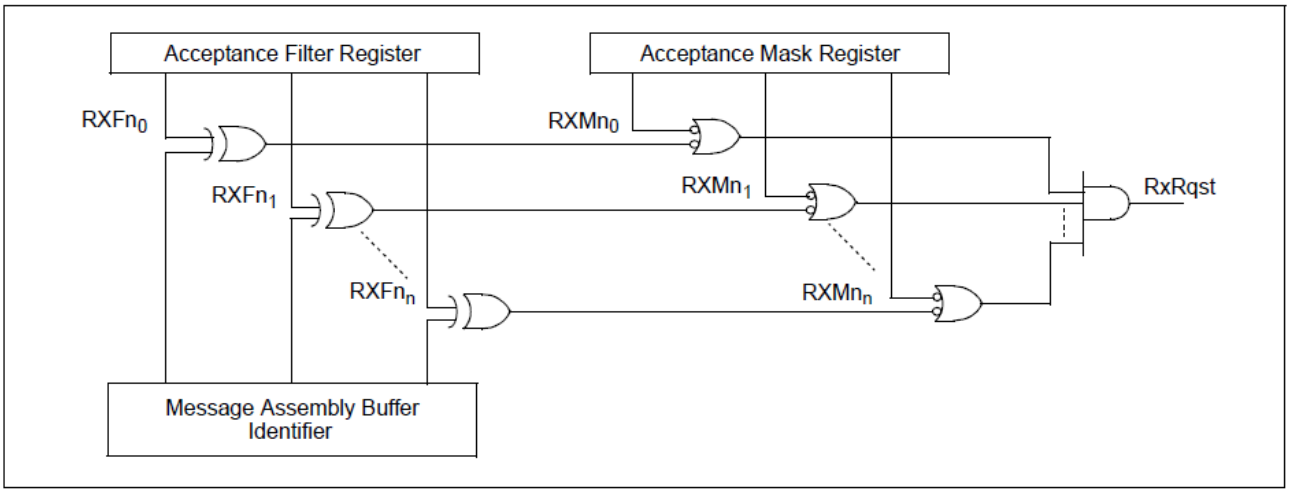
La mise en oeuvre de RXB0DBEN permet d'utiliser ces 3 bits comme « Hitbits » pour RXB0 et pour distinguer s'il y a eu transfert entre RXB0 et RXB1 :

- FILHIT<2:0> = 111 : Filtre 1 (RXF1)
- FILHIT<2:0> = 110 : Filtre 0 (RXF0)
- FILHIT<2:0> = 001 : Filtre 1 (RXF1)
- FILHIT<2:0> = 000 : Filtre 1 (RXF0)

Si RXB0DBEN est à 0, on se trouve face à 6 combinaisons associées aux 6 filtres. Si ce bit est à 1, deux codes supplémentaires correspondent à RXF0, RXF1 ainsi qu'un transfert vers RXB1.

Lorsque plusieurs filtres acceptent le message, c'est la valeur correspondante au plus petit indice qui est codée.

FIGURE 23-3: MESSAGE ACCEPTANCE MASK AND FILTER OPERATION



Configuration du débit

Tous les noeuds situés sur un bus CAN doivent être configurés pour le même débit. Le protocole CAN met en place un code NRZ sans codage d'horloge dans la trame. Il faut donc que les noeuds soient configurés correctement.

Le débit nominal correspond au nombre de bits transmis par seconde dans le cas idéal (oscillateur parfait, sans resynchronisations...).

Le ***débit nominal maximum est de 1 Mbps***. A partir du débit nominal, on définit le « Bit time » nominal :

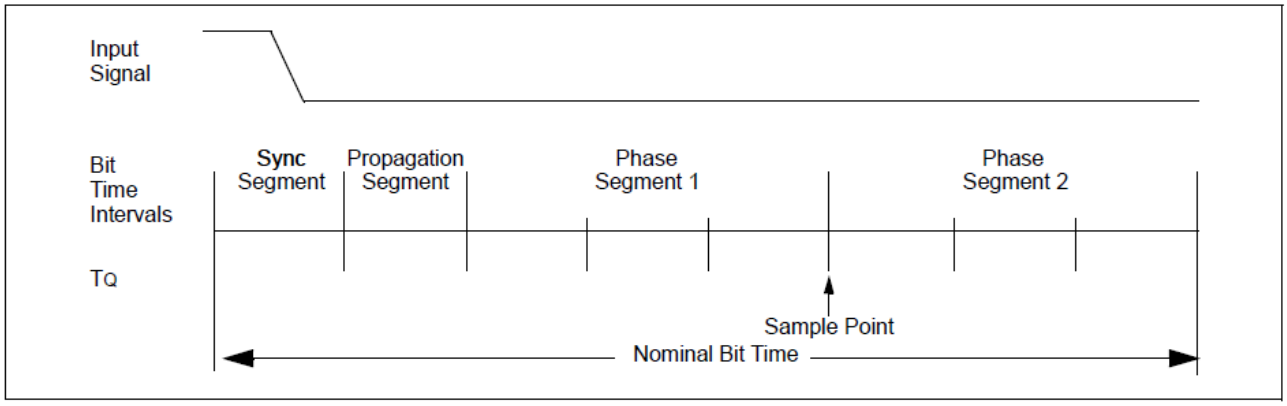
$$T_{BIT} = \frac{1}{Debit\ Nominal}$$

Le bit time nominal doit être pensé comme étant divisé en « segments temporels » distincts et successifs. On distingue 4 segments temporels :

- Le segment de synchronisation : SYNC_SEG
- Le segment « temps de propagation » : PROP_SEG
- Le segment « Phase buffer 1 » : PHASE_SEG1
- Le segment « Phase buffer 2 » : PHASE_SEG2

Ces segments temporels (et donc le Tbit nominal) sont formés à partir d'unités de temps entières appelées « ***Time Quanta*** » (Quantum temporel) : T_Q.

FIGURE 23-4: BIT TIME PARTITIONING



Par définition, le bit time nominal est programmable ***entre un minimum de 8.T_Q et un maximum de 25.T_Q***.
Toujours par définition, le ***bit time nominal minimum*** (pour un débit maximum de 1 Mbps) est de ***1µs***.
La relation suivante est utilisée pour calculer T_Q :

$$T_{BIT} = T_Q * (\text{SYNC\_SEG} + \text{PROP\_SEG} + \text{PHASE\_SEG1} + \text{PHASE\_SEG2})$$

Le quantum temporel est dérivé de la période de l'oscillateur. Il est programmable par le un prédiviseur (Baud Rate Prescaler) dont la valeur varie entre 1 et 64.
La relation mathématique liant toutes ces valeurs est la suivante :

$$T_Q(\mu s) = \frac{(2 * (BRP + 1))}{F_{OSC}(MHz)}$$

BRP est une valeur entière comprise entre 0 et 63 et correspondante aux bits ***BRGCON1<5:0>***.

Remarque :

La fréquence de l'oscillateur Fosc, est ici la vraie fréquence « interne ».
Un quartz de 10MHz utilisé en mode HS donne Fosc = 10
Un quartz de 10MHz utilisé en mode HS-PLL donne Fosc = 40

Exemples de calculs

TQ (µs) = (2 * (BRP + 1))/FOSC (MHz)

TBIT (µs) = TQ (µs) * number of TQ per bit interval

Nominal Bit Rate (bits/s) = 1/TBIT

This frequency (FOSC) refers to the effective frequency used. If, for example, a 10 MHz external signal is used along with a PLL, then the effective frequency will be 4 x 10 MHz which equals 40 MHz.

CASE 1:

For FOSC = 16 MHz, BRP<5:0> = 00h and Nominal Bit Time = 8 TQ:

TQ = (2 * 1)/16 = 0.125 µs (125 ns)

TBIT = 8 * 0.125 = 1 µs (10^{-6}s)

Nominal Bit Rate = $1/10^{-6}$ = 10^6 bits/s (1 Mb/s)

CASE 2:

For FOSC = 20 MHz, BRP<5:0> = 01h and Nominal Bit Time = 8 TQ:

TQ = (2 * 2)/20 = 0.2 µs (200 ns)

TBIT = 8 * 0.2 = 1.6 µs (1.6 * 10^{-6}s)

Nominal Bit Rate = 1/1.6 * 10^{-6}s = 625,000 bits/s (625 Kb/s)

CASE 3:

For FOSC = 25 MHz, BRP<5:0> = 3Fh and Nominal Bit Time = 25 TQ:

TQ = (2 * 64)/25 = 5.12 µs

TBIT = 25 * 5.12 = 128 µs (1.28 * 10^{-4}s)

Nominal Bit Rate = 1/1.28 * 10^{-4} = 7813 bits/s

Retour sur les segments temporels

SYNC_SEG	Ce segment permet de synchroniser les différents noeuds sur le bus. Le front du signal doit se produire durant ce segment. La durée du segment de synchronisation est de ***1 T_Q***.
PROP_SEG	Le rôle de ce segment est de compenser les temps propagation physiques sur le support. Il s'agit de programmer des délais internes au noeuds. La durée de ce segment est programmable ***entre 1 et 8 T_Q***. Les bits ***PRSEG2:PRSEG0*** sont utilisés pour effectuer cette configuration.
PHASE_SEG1 et PHASE_SEG2	Les deux segments de phase permettent de déterminer précisément la position du point d'acquisition du bit reçu. Ce point est situé entre les deux segments. Ceux-ci peuvent être réduits ou agrandis par le processus de resynchronisation. La fin du segment de phase n°1 indique le point d'acquisition à l'intérieur du bit time, la durée de ce segment est programmable de ***1 à 8 T_Q***. Le second segment de phase met en place un délai avant la transition suivante. La durée est elle aussi comprise entre ***1 et 8 T_Q***. Cependant, pour des raisons dues au temps de traitement de l'information (IPT : Information Processing Time), la durée minimale de PHASE_SEG2 est de ***2.T_Q***. L'expérience montre que le ***point d'acquisition placé à 80% du bit time*** donne de très bons résultats.

Remarque : la resynchronisation

Le système est capable de gérer des séquences de resynchronisation, pour rattraper une dérive sur un oscillateur par exemple. Une resynchronisation se traduit par un allongement du segment de phase 1 ou un raccourcissement du segment de phase n°2.

La taille maximale du décalage est configurée, c'est le SJW (Synchronisation Jump Width). Le SJW est programmable entre 1 et 4 T_Q.

Programmation des segments temporels

Quelques contraintes sont à prendre en compte :

- ✗ PROP_SEG + PHASE_SEG1 ≥ PHASE_SEG2
- ✗ PHASE_SEG2 ≥ SJW

Les registres associés au « Baud Rate Control » sont ***BRGCON1, BRGCON2 et BRGCON3***.

BRGCON1

Les bits ***BRP*** contrôlent le prédiviseur. Les bits ***SJW<1:0>*** sont à utiliser pour programmer le ***SJW*** (en multiples de T_Q).

BRGCON1: BAUD RATE CONTROL REGISTER 1

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
SJW1	SJW0	BRP5	BRP4	BRP3	BRP2	BRP1	BRP0
bit 7							bit 0

bit 7-6 **SJW1:SJW0:** Synchronized Jump Width bits

11 = Synchronization jump width time = 4 x TQ
10 = Synchronization jump width time = 3 x TQ
01 = Synchronization jump width time = 2 x TQ
00 = Synchronization jump width time = 1 x TQ

bit 5-0 **BRP5:BRP0:** Baud Rate Prescaler bits

111111 = TQ = (2 x 64)/FOSC
111110 = TQ = (2 x 63)/FOSC
:
:
000001 = TQ = (2 x 2)/FOSC
000000 = TQ = (2 x 1)/FOSC

BRGCON2

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
SEG2PHTS	SAM	SEG1PH2	SEG1PH1	SEG1PH0	PRSEG2	PRSEG1	PRSEG0
bit 7							bit 0

bit 7 **SEG2PHTS:** Phase Segment 2 Time Select bit

1 = Freely programmable
0 = Maximum of PHEG1 or Information Processing Time (IPT), whichever is greater

bit 6 **SAM:** Sample of the CAN bus Line bit

1 = Bus line is sampled three times prior to the sample point
0 = Bus line is sampled once at the sample point

bit 5-3 **SEG1PH2:SEG1PH0:** Phase Segment 1 bits

111 = Phase Segment 1 time = 8 x TQ
110 = Phase Segment 1 time = 7 x TQ
101 = Phase Segment 1 time = 6 x TQ
100 = Phase Segment 1 time = 5 x TQ
011 = Phase Segment 1 time = 4 x TQ
010 = Phase Segment 1 time = 3 x TQ
001 = Phase Segment 1 time = 2 x TQ
000 = Phase Segment 1 time = 1 x TQ

bit 2-0 **PRSEG2:PRSEG0:** Propagation Time Select bits

111 = Propagation time = 8 x TQ
110 = Propagation time = 7 x TQ
101 = Propagation time = 6 x TQ
100 = Propagation time = 5 x TQ
011 = Propagation time = 4 x TQ
010 = Propagation time = 3 x TQ
001 = Propagation time = 2 x TQ
000 = Propagation time = 1 x TQ

BRGCON3

R/W-0	R/W-0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0
WAKDIS	WAKFIL	—	—	—	SEG2PH2(1)	SEG2PH1(1)	SEG2PH0(1)
bit 7							bit 0

bit 7 **WAKDIS:** Wake-up Disable bit
1 = Disable CAN bus activity wake-up feature
0 = Enable CAN bus activity wake-up feature

bit 6 **WAKFIL:** Selects CAN bus Line Filter for Wake-up bit
1 = Use CAN bus line filter for wake-up
0 = CAN bus line filter is not used for wake-up

bit 5-3 **Unimplemented:** Read as '0'

bit 2-0 **SEG2PH2:SEG2PH0:** Phase Segment 2 Time Select bits(1)
111 = Phase Segment 2 time = 8 x TQ
110 = Phase Segment 2 time = 7 x TQ
101 = Phase Segment 2 time = 6 x TQ
100 = Phase Segment 2 time = 5 x TQ
011 = Phase Segment 2 time = 4 x TQ
010 = Phase Segment 2 time = 3 x TQ
001 = Phase Segment 2 time = 2 x TQ
000 = Phase Segment 2 time = 1 x TQ

Note 1: Ignored if SEG2PHTS bit (BRGCON2<7>) is '0'.

Détection d'erreurs

Le protocole CAN met en oeuvre des techniques sophistiquées permettant la détection d'erreurs. Le tableau suivant indique les erreurs détectables et leur cause.

Type d'erreur	Cause
CRC Error	Un champ CRC est calculé par l'émetteur et transmis dans la trame. Ce CRC est recalculé à la réception et comparé à celui transmis. En cas de différence, un trame d'erreur est générée et la trame devra être réémise.
Acknowledge Error	Durant la transmission du champ ACK, l'émetteur vérifie que le bit présent est dominant (alors que l'émetteur avait envoyé un bit recessif). S'il n'y a pas d'ack, une trame d'erreur est émise et le message devra être réémis.
Form Error	Si un noeud détecte un bit dominant dans un des champs suivants : End Of Frame, Trou inter-trame, Délimiteur ACK ou délimiteur CRC, une erreur de format s'est produite. Une trame d'erreur est générée. Le message est réémis.
Bit Error	Une telle erreur se produit si un émetteur envoyant un bit ne détecte pas la valeur émise (envoi d'un bit récessif, lecture d'un bit dominant, ou l'inverse). Une telle erreur se produisant durant la transmission de l'identifiant ou durant la séquence d'acq ne génère pas le positionnement d'un drapeau d'erreur : il s'agit d'une situation d'arbitrage normale ou d'acquittement classique.
Stuff Bit Error	Si entre le Start Of Frame et le délimiteur CRC, une séquence de 6 bits consécutifs au même niveau est détectée, il s'agit d'une violation de la règle du « bit stuffing ». Une trame d'erreur est générée et le message est réémis.

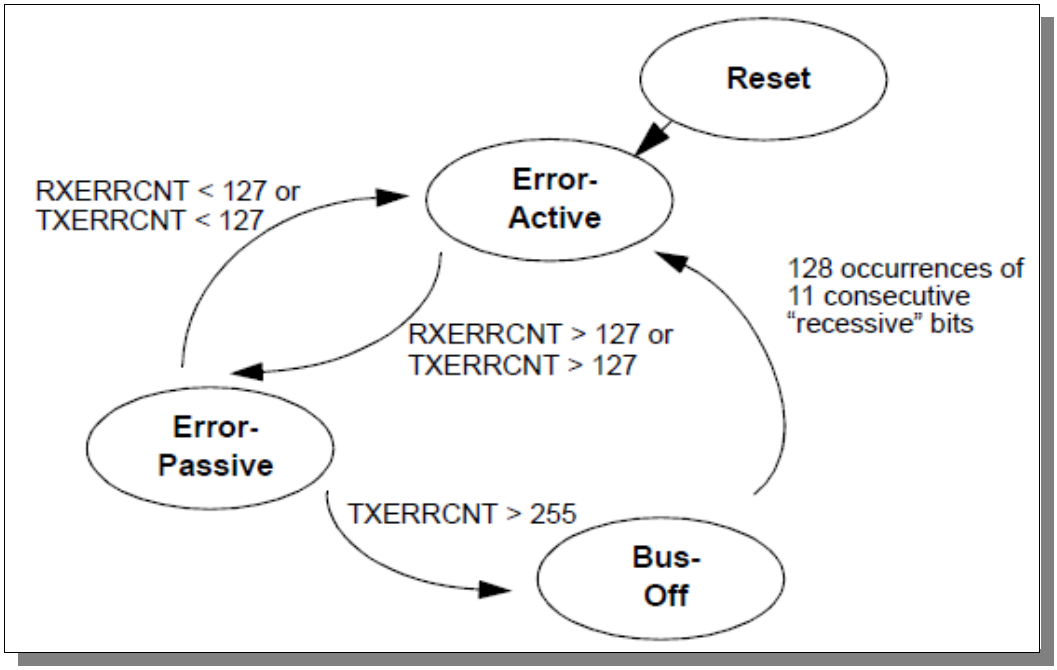
Toute erreur détectée est connue des noeuds sur le bus à travers les trames d'erreur. L'émission de la trame erronée est abandonnée aussitôt et la trame est réémise le plus tôt possible.

De plus, chaque noeud CAN peut être dans un des trois états suivants face aux erreurs, en fonction des valeurs de ses compteurs d'erreurs internes :

- ***Error Active*** : Etat « normal », le noeud peut émettre des messages et des trames d'erreur sans restrictions.
- ***Error Passive*** : Dans cet état, seules les trames « message » et « erreur passive » sont autorisés pour l'émission.
- ***Bus Off*** : Dans cet état, le noeud est temporairement « déconnecté » du bus.

Le module ECAN possède deux compteurs d'erreurs : ***RXERRCNT et TXERRCNT***. Ces compteurs sont incrémentés et décrémentés en accord le protocole CAN.

Le passage d'un état à un autres (Error Active, Error Passive ou Bus Off) est géré automatiquement selon ce diagramme :



Les interruptions liées au module ECAN

Le module ECAN possède de multiples sources d'interruption. Chacune d'elles peut être activée ou désactivée individuellement.

Le registre ***PIR3*** contient les flags IT tandis que les bits d'autorisation (pour les 8 sources principales) sont situés dans ***PIE3***. Un jeu de bits spécifiques dans le ***registre CANSTAT***, les bits ***ICODE*** peuvent être utilisés en association avec une table de saut pour une meilleure gestion des interruptions.

Chaque interruption possède une source, à l'exception de l'interruption liée à l'apparition d'une erreur. Si une interruption « erreur » apparaît, l'erreur doit être identifiée avec les registre d'état de la communication (Communication Status Register) : ***COMSTAT***.

Les interruptions peuvent être classées en deux catégories : les IT émission et réception.

Les interruptions « Réception » sont les suivantes :

- Receive ITs
- Wake-Up IT
- Receiver Overrun IT
- Receiver Warning IT
- Receiver Error-Passive IT

Les interruptions « Emission » sont :

- ✗ Transmit ITs
- ✗ Transmitter Warning IT
- ✗ Transmitter Error Passive IT
- ✗ Bus Off IT

Pour simplifier la gestion des interruptions CAN, le module utilise un jeu de bits spécifiques. Dans le mode 0, ces bits sont ***ICODE<3:1> du registre CANSTAT.***

Interruptions Transmission

Lorsque une IT « Transmission » est activée, elle se produit lorsque le buffer d'émission associé est vide et est prêt à émettre un nouveau message. Dans le mode 0, trois interruptions distinctes sont disponibles, une pour chacun des trois buffers. Les bits ***TXBnIF*** sont ici utilisés. Les bits d'autorisation sont ***TXBnIE***, les priorités de ces interruptions sont définies par les bits ***TXBnIP***. Ces bits sont situés dans les registres ***PIR3, PIE3 et IPR3***.

Interruptions Réception

Une IT « réception » se produit lorsque un message a été reçu et stocké dans un buffer de réception.

Dans le mode 0, les bits ***RXBnIF*** sont utilisés. Ces interruptions sont autorisées à l'aide des bits ***RXBnIE***.

Là encore, les registres ***PIR3, PIE3 et IPR3*** (priorités) sont les hôtes de ces bits.

Registre CANCON : CAN Control Register

Informations données pour le ***mode 0 seulement***.

Mode 0	R/W-1	R/W-0	R/W-0	R/S-0	R/W-0	R/W-0	R/W-0	U-0
	REQOP2	REQOP1	REQOP0	ABAT	WIN2	WIN1	WIN0	—

bit 7-5 **REQOP2:REQOP0:** Request CAN Operation Mode bits

1xx = Request Configuration mode
011 = Request Listen Only mode
010 = Request Loopback mode
001 = Request Disable mode
000 = Request Normal mode

bit 4 **ABAT:** Abort All Pending Transmissions bit

1 = Abort all pending transmissions (in all transmit buffers)
0 = Transmissions proceeding as normal

bit 3-1 Mode 0:

WIN2:WIN0: Window Address bits

These bits select which of the CAN buffers to switch into the access bank area. This allows access to the buffer registers from any data memory bank. After a frame has caused an interrupt, the ICODE3:ICODE0 bits can be copied to the WIN3:WIN0 bits to select the correct buffer. See Example 23-2 for a code example.

111 = Receive Buffer 0
110 = Receive Buffer 0
101 = Receive Buffer 1
100 = Transmit Buffer 0
011 = Transmit Buffer 1
010 = Transmit Buffer 2
001 = Receive Buffer 0
000 = Receive Buffer 0

Registre CANSTAT : CAN Status Register

Informations données pour le ***mode 0 seulement***.

Mode 0	R-1	R-0	R-0	R-0	R-0	R-0	R-0	U-0
	OPMODE2(1)	OPMODE1(1)	OPMODE0(1)	—	ICODE3	ICODE2	ICODE1	—

bit 7-5 **OPMODE2:OPMODE0:** Operation Mode Status bits(1)

111 = Reserved
110 = Reserved
101 = Reserved
100 = Configuration mode
011 = Listen Only mode
010 = Loopback mode
001 = Disable/Sleep mode
000 = Normal mode

bit 4 Mode 0:

Unimplemented: Read as '0'

bit 3-1 **ICODE3:ICODE1:** Interrupt Code bits

When an interrupt occurs, a prioritized coded interrupt value will be present in these bits. This code indicates the source of the interrupt. By copying ICODE3:ICODE1 to WIN2:WIN0 (Mode 0) or EICODE4:EICODE0 to EWIN4:EWIN0 (Mode 1 and 2), it is possible to select the correct buffer to map into the Access Bank area. See Example 23-2 for a code example. To simplify the description, the following table lists all five bits.

	Mode 0	Mode 1	Mode 2
No interrupt	00000	00000	00000
Error interrupt	00010	00010	00010
TXB2 interrupt	00100	00100	00100
TXB1 interrupt	00110	00110	00110
TXB0 interrupt	01000	01000	01000
RXB1 interrupt	01010	10001	-----
RXB0 interrupt	01100	10000	10000
Wake-up interrupt	00010	01110	01110
RXB0 interrupt	-----	10000	10000
RXB1 interrupt	-----	10001	10000
RX/TX B0 interrupt	-----	10010	10010
RX/TX B1 interrupt	-----	10011	10011(2)
RX/TX B2 interrupt	-----	10100	10100(2)
RX/TX B3 interrupt	-----	10101	10101(2)
RX/TX B4 interrupt	-----	10110	10110(2)
RX/TX B5 interrupt	-----	10111	10111(2)

Registre ECANCON : Enhanced CAN Control Register

R/W-0	R/W-0	R/W-0	R/W-1	R/W-0	R/W-0	R/W-0	R/W-0
MDSEL1(1)	MDSEL0(1)	FIFOWM(2)	EWIN4	EWIN3	EWIN2	EWIN1	EWIN0
bit 7							bit 0

bit 7-6 **MDSEL1:MDSEL0:** Mode Select bits(1)

00 = Legacy mode (Mode 0, default)
01 = Enhanced Legacy mode (Mode 1)
10 = Enhanced FIFO mode (Mode 2)
11 = Reserved

Registres TXBnCON : Transmit Buffer n Control Register (n = 0, 1 ou 2)

Mode 0	R/C-0	R-0	R-0	R-0	R/W-0	U-0	R/W-0	R/W-0
	TXBIF	TXABT(1)	TXLARB(1)	TXERR(1)	TXREQ(2)	—	TXPRI1(3)	TXPRI0(3)

bit 7 **TXBIF:** Transmit Buffer Interrupt Flag bit
1 = Transmit buffer has completed transmission of message and may be reloaded
0 = Transmit buffer has not completed transmission of a message

bit 6 **TXABT:** Transmission Aborted Status bit(1)
1 = Message was aborted
0 = Message was not aborted

bit 5 **TXLARB:** Transmission Lost Arbitration Status bit(1)
1 = Message lost arbitration while being sent
0 = Message did not lose arbitration while being sent

bit 4 **TXERR:** Transmission Error Detected Status bit(1)
1 = A bus error occurred while the message was being sent
0 = A bus error did not occur while the message was being sent

bit 3 **TXREQ:** Transmit Request Status bit(2)
1 = Requests sending a message. Clears the TXABT, TXLARB and TXERR bits.
0 = Automatically cleared when the message is successfully sent

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

bit 1-0 **TXPRI1:TXPRI0:** Transmit Priority bits(3)
11 = Priority Level 3 (highest priority)
10 = Priority Level 2
01 = Priority Level 1
00 = Priority Level 0 (lowest priority)

Note 1: This bit is automatically cleared when TXREQ is set.

2: While TXREQ is set, Transmit Buffer registers remain read-only. Clearing this bit in software while the bit is set will request a message abort.

3: These bits define the order in which transmit buffers will be transferred. They do not alter the CAN message identifier.

Registres associés à l'émission de l'identifiant message

Selon le format de l'identifiant (standard ou étendu) les registres mis en oeuvre ne contiennent pas les mêmes informations.

Rappel :

*Le mode **standard** est choisi en positionnant le bit **EXIDE à 0***

*Le mode **étendu** est chois en positionnant le bit **EXIDE à 1***

Selon le mode, l'identifiant est appelé ***SID*** (Standard Identier, 11 bits) ou ***EID*** (Extended Identifer 29 bits).

	Mode standard	Mode étendu
TXBnSIDH	**SID[10:3]**	**EID[28:21]**
TXBnSIDL	R/W-x R/W-x R/W-x U-0 R/W-x U-0 R/W-x R/W-x SID2 SID1 SID0 — EXIDE — EID17 EID16 bit 7 bit 0 bit 7-5 **SID2:SID0:** Standard Identifier bits (if EXIDE (TXBnSIDL<3>) = 0) Extended Identifier bits EID20:EID18 (if EXIDE = 1). bit 4 **Unimplemented:** Read as '0' bit 3 **EXIDE:** Extended Identifier Enable bit 1 = Message will transmit extended ID, SID10:SID0 become EID28:EID18 0 = Message will transmit standard ID, EID17:EID0 are ignored bit 2 **Unimplemented:** Read as '0' bit 1-0 **EID17:EID16:** Extended Identifier bits	
TXBnEIDH	Non utilisés en mode standard	**EID[15:8]**
TXBnEIDL		**EID[7:0]**

Registres TXBnDLC

Le nombre d'octets de données du message est codé sur les ***bits DLC[3:0] de 0000 à 1000 (0 à 8 octets)***.

Registre RXB0CON : Receive Buffer 0 Control Register

Pour le mode 0 seulement.

Mode 0	R/C-0	R/W-0	R/W-0	U-0	R-0	R/W-0	R-0	R-0
	RXFUL(1)	RXM1	RXM0	—	RXRTRRO	RXB0DBEN	JTOFF(2)	FILHIT0

b6 ***b5***	***RXM1:0***	**00**	Réception de tous les messages valides
		01	Réception des messages valides avec un identifiant standard (EXIDEN doit être à 0)
		10	Réception des messages valides avec un identifiant étendu (EXIDEN doit être à 1)
		11	Réception de tous les messages (y compris erronés) les filtres sont ignorés
b3	***RXRTRRO***	**1**	Une requête de transmission « Remote » a été reçue
b2	***RXB0DBEN***	**1**	Un débordement sur le buffer 0 sera « transmis » au buffer 1
		0	Le buffer 1 n'est pas utilisé en cas de débordement du buffer 0
b1	***JTOFF***		Jump Table offset bit (cf doc. Technique)
b0	***FILHIT0***	**0**	Indique que c'est le filtre RXF0 qui a accepté le message
		11	Indique que c'est le filtre RXF1 qui a accepté le message

Registre RXB1CON : Receive Buffer 1 Control Register

Mode 0	R/C-0	R/W-0	R/W-0	U-0	R-0	R/W-0	R-0	R-0
	RXFUL(1)	RXM1	RXM0	—	RXRTRRO	FILHIT2	FILHIT1	FILHIT0

Ce registre est semblable à RXB0CON à la différence qu'il n'existe pas de bit JTOFF et qu'il est doté de 3 bits « Filter Hit » :

bit 2-0 Mode 0:

FILHIT2:FILHIT0: Filter Hit bits

These bits indicate which acceptance filter enabled the last message receptio Receive Buffer 1.

111 = Reserved
110 = Reserved
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 RXB0DBEN bit is set
000 = Acceptance Filter 0 (RXF0), only possible when RXB0DBEN bit is set

Registre RXBnSIDH et RXBnSIDL

RXBnSIDH

R-x	R-x	R-x	R-x	R-x	R-x	R-x	R-x
SID10	SID9	SID8	SID7	SID6	SID5	SID4	SID3
bit 7							bit 0

bit 7-0 **SID10:SID3:** Standard Identifier bits (if EXID (RXBnSIDL<3>) = 0)
Extended Identifier bits EID28:EID21 (if EXID = 1).

RXBnSIDL

R-x	R-x	R-x	R-x	R-x	U-0	R-x	R-x
SID2	SID1	SID0	SRR	EXID	—	EID17	EID16
bit 7							bit 0

bit 7-5 **SID2:SID0:** Standard Identifier bits (if EXID = 0)
Extended Identifier bits EID20:EID18 (if EXID = 1).

bit 4 **SRR:** Substitute Remote Request bit
This bit is always '0' when EXID = 1 or equal to the value of RXRTRRO (RBXnCON<3>) when EXID = 0.

bit 3 **EXID:** Extended Identifier bit
1 = Received message is an extended data frame, SID10:SID0 are EID28:EID18
0 = Received message is a standard data frame

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

bit 1-0 **EID17:EID16:** Extended Identifier bits

Registres RXBnEIDH et RXBnEIDL

REGISTER 23-17: RXBnEIDH: RECEIVE BUFFER n EXTENDED IDENTIFIER REGISTERS, HIGH BYTE [0 ≤ n ≤ 1]

R-x	R-x	R-x	R-x	R-x	R-x	R-x	R-x
EID15	EID14	EID13	EID12	EID11	EID10	EID9	EID8
bit 7							bit 0

bit 7-0 **EID15:EID8:** Extended Identifier bits

REGISTER 23-18: RXBnEIDL: RECEIVE BUFFER n EXTENDED IDENTIFIER REGISTERS, LOW BYTE [0 ≤ n ≤ 1]

R-x	R-x	R-x	R-x	R-x	R-x	R-x	R-x
EID7	EID6	EID5	EID4	EID3	EID2	EID1	EID0
bit 7							bit 0

bit 7-0 **EID7:EID0:** Extended Identifier bits

Registre RXBnDLC

U-0	R-x	R-x	R-x	R-x	R-x	R-x	R-x
—	RXRTR	RB1	RB0	DLC3	DLC2	DLC1	DLC0
bit 7							bit 0

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

bit 6 **RXRTR:** Receiver Remote Transmission Request bit
1 = Remote transfer request
0 = No remote transfer request

bit 5 **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 **DLC3:DLC0:** Data Length Code bits
1111 = Invalid
1110 = Invalid
1101 = Invalid
1100 = Invalid
1011 = Invalid
1010 = Invalid
1001 = Invalid
1000 = Data length = 8 bytes
0111 = Data length = 7 bytes
0110 = Data length = 6 bytes
0101 = Data length = 5 bytes
0100 = Data length = 4 bytes
0011 = Data length = 3 bytes
0010 = Data length = 2 bytes
0001 = Data length = 1 bytes
0000 = Data length = 0 bytes

Registres RXFnSIDH et RXFnSIDL (Filtres d'acceptation)

REGISTER 23-37: RXFnSIDH: RECEIVE ACCEPTANCE FILTER n STANDARD IDENTIFIER FILTER REGISTERS, HIGH BYTE [0 ≤ n ≤ 15](1)

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
SID10	SID9	SID8	SID7	SID6	SID5	SID4	SID3
bit 7							bit 0

bit 7-0 **SID10:SID3:** Standard Identifier Filter bits (if EXIDEN = 0)
Extended Identifier Filter bits EID28:EID21 (if EXIDEN = 1).

Note 1: Registers RXF6SIDH:RXF15SIDH are available in Mode 1 and 2 only.

REGISTER 23-38: RXFnSIDL: RECEIVE ACCEPTANCE FILTER n STANDARD IDENTIFIER FILTER REGISTERS, LOW BYTE [0 ≤ n ≤ 15](1)

R/W-x	R/W-x	R/W-x	U-0	R/W-x	U-0	R/W-x	R/W-x
SID2	SID1	SID0	—	EXIDEN(2)	—	EID17	EID16
bit 7							bit 0

bit 7-5 **SID2:SID0:** Standard Identifier Filter bits (if EXIDEN = 0)
Extended Identifier Filter bits EID20:EID18 (if EXIDEN = 1).

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

bit 3 **EXIDEN:** Extended Identifier Filter Enable bit(2)
1 = Filter will only accept extended ID messages
0 = Filter will only accept standard ID messages

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

bit 1-0 **EID17:EID16:** Extended Identifier Filter bits

Note 1: Registers RXF6SIDL:RXF15SIDL are available in Mode 1 and 2 only.

2: In Mode 0, this bit must be set/cleared as required, irrespective of corresponding mask register value.

Registres RXFnEIDH et RXFnEIDL (Filtres d'acceptation)

REGISTER 23-39: RXFnEIDH: RECEIVE ACCEPTANCE FILTER n EXTENDED IDENTIFIER REGISTERS, HIGH BYTE [0 ≤ n ≤ 15](1)

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
EID15	EID14	EID13	EID12	EID11	EID10	EID9	EID8
bit 7							bit 0

bit 7-0 **EID15:EID8:** Extended Identifier Filter bits

Note 1: Registers RXF6EIDH:RXF15EIDH are available in Mode 1 and 2 only.

REGISTER 23-40: RXFnEIDL: RECEIVE ACCEPTANCE FILTER n EXTENDED IDENTIFIER REGISTERS, LOW BYTE [0 ≤ n ≤ 15](1)

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
EID7	EID6	EID5	EID4	EID3	EID2	EID1	EID0
bit 7							bit 0

bit 7-0 **EID7:EID0:** Extended Identifier Filter bits

Note 1: Registers RXF6EIDL:RXF15EIDL are available in Mode 1 and 2 only.

Registres RXMnSIDH et RXMnSIDL (Masques)

REGISTER 23-41: RXMnSIDH: RECEIVE ACCEPTANCE MASK n STANDARD IDENTIFIER MASK REGISTERS, HIGH BYTE [0 ≤ n ≤ 1]

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
SID10	SID9	SID8	SID7	SID6	SID5	SID4	SID3
bit 7							bit 0

bit 7-0 **SID10:SID3:** Standard Identifier Mask bits or Extended Identifier Mask bits EID28:EID21

REGISTER 23-42: RXMnSIDL: RECEIVE ACCEPTANCE MASK n STANDARD IDENTIFIER MASK REGISTERS, LOW BYTE [0 ≤ n ≤ 1]

R/W-x	R/W-x	R/W-x	U-0	R/W-0	U-0	R/W-x	R/W-x
SID2	SID1	SID0	—	EXIDEN(1)	—	EID17	EID16
bit 7							bit 0

bit 7-5 **SID2:SID0:** Standard Identifier Mask bits or Extended Identifier Mask bits EID20:EID18

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

bit 3 Mode 0:
Unimplemented: Read as '0'
Mode 1, 2:
EXIDEN: Extended Identifier Filter Enable Mask bit(1)
1 = Messages selected by the EXIDEN bit in RXFnSIDL will be accepted
0 = Both standard and extended identifier messages will be accepted

Note 1: This bit is available in Mode 1 and 2 only.

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

bit 1-0 **EID17:EID16:** Extended Identifier Mask bits

Registres RXMnEIDH et RXMnEIDL (Masques)

REGISTER 23-43: RXMnEIDH: RECEIVE ACCEPTANCE MASK n EXTENDED IDENTIFIER MASK REGISTERS, HIGH BYTE [0 ≤ n ≤ 1]

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
EID15	EID14	EID13	EID12	EID11	EID10	EID9	EID8
bit 7							bit 0

bit 7-0 **EID15:EID8:** Extended Identifier Mask bits

REGISTER 23-44: RXMnEIDL: RECEIVE ACCEPTANCE MASK n EXTENDED IDENTIFIER MASK REGISTERS, LOW BYTE [0 ≤ n ≤ 1]

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
EID7	EID6	EID5	EID4	EID3	EID2	EID1	EID0
bit 7							bit 0

bit 7-0 **EID7:EID0:** Extended Identifier Mask bits

Registre CIOCON : CAN I/O Control Register

U-0	U-0	R/W-0	R/W-0	U-0	U-0	U-0	U-0
—	—	ENDRHI(1)	CANCAP	—	—	—	—
bit 7							bit 0

bit 7-6 **Unimplemented:** Read as '0'

bit 5 **ENDRHI:** Enable Drive High bit(1)

1 = CANTX pin will drive VDD when recessive
0 = CANTX pin will be tri-state when recessive

Note 1: Always set this bit when using differential bus to avoid signal crosstalk in CANTX from other nearby pins.

bit 4 **CANCAP:** CAN Message Receive Capture Enable bit

1 = Enable CAN capture, CAN message receive signal replaces input on RC2/CCP1
0 = Disable CAN capture, RC2/CCP1 input to CCP1 module

bit 3-0 **Unimplemented:** Read as '0'

Registre PIR3

Mode 0

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
IRXIF	WAKIF	ERRIF	TXB2IF	TXB1IF(1)	TXB0IF(1)	RXB1IF	RXB0IF

bit 7 **IRXIF:** CAN Invalid Received Message Interrupt Flag bit
1 = An invalid message has occurred on the CAN bus
0 = No invalid message on CAN bus

bit 6 **WAKIF:** CAN bus Activity Wake-up Interrupt Flag bit
1 = Activity on CAN bus has occurred
0 = No activity on CAN bus

bit 5 **ERRIF:** CAN bus Error Interrupt Flag bit
1 = An error has occurred in the CAN module (multiple sources)
0 = No CAN module errors

bit 4 When CAN is in Mode 0:
TXB2IF: CAN Transmit Buffer 2 Interrupt Flag bit
1 = Transmit Buffer 2 has completed transmission of a message and may be reloaded
0 = Transmit Buffer 2 has not completed transmission of a message
When CAN is in Mode 1 or 2:
TXBnIF: Any Transmit Buffer Interrupt Flag bit
1 = One or more transmit buffers have completed transmission of a message and may be reloaded
0 = No transmit buffer is ready for reload

bit 3 **TXB1IF:** CAN Transmit Buffer 1 Interrupt Flag bit(1)
1 = Transmit Buffer 1 has completed transmission of a message and may be reloaded
0 = Transmit Buffer 1 has not completed transmission of a message

bit 2 **TXB0IF:** CAN Transmit Buffer 0 Interrupt Flag bit(1)
1 = Transmit Buffer 0 has completed transmission of a message and may be reloaded
0 = Transmit Buffer 0 has not completed transmission of a message

bit 1 When CAN is in Mode 0:
RXB1IF: CAN Receive Buffer 1 Interrupt Flag bit
1 = Receive Buffer 1 has received a new message
0 = Receive Buffer 1 has not received a new message
When CAN is in Mode 1 or 2:
RXBnIF: Any Receive Buffer Interrupt Flag bit
1 = One or more receive buffers has received a new message
0 = No receive buffer has received a new message

bit 0 When CAN is in Mode 0:
RXB0IF: CAN Receive Buffer 0 Interrupt Flag bit
1 = Receive Buffer 0 has received a new message
0 = Receive Buffer 0 has not received a new message
When CAN is in Mode 1:
Unimplemented: Read as '0'
When CAN is in Mode 2:
FIFOWMIF: FIFO Watermark Interrupt Flag bit
1 = FIFO high watermark is reached
0 = FIFO high watermark is not reached

Note 1: In CAN Mode 1 and 2, this bit is forced to '0'.