マイクロチップ社

PIC32MX Section34　CAN モジュール

完全日本語訳　*サンプル*

(Microchip 社の原典の工業所有権表示は次頁に掲載。)

尚、日本語訳文から派生する利用者のいかなる不利益もテック・ハンゾウガネ諸橋義明は責任を負いません。

Note the following details of the code protection feature on Microchip devices:

- Microchip products meet the specification contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is one of the most secure families of its kind on the market today, when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods used to breach the code protection feature. All of these methods, to our knowledge, require using the Microchip products in a manner outside the operating specifications contained in Microchip's Data Sheets. Most likely, the person doing so is engaged in theft of intellectual property.
- Microchip is willing to work with the customer who is concerned about the integrity of their code.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as "unbreakable."

Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our products. Attempts to break microchip's code protection feature may be a violation of the Digital Millennium Copyright Act. If such acts allow unauthorized access to your software or other copyrighted work, you may have a right to sue for relief under that Act.

Microchip received QS-9000 quality system certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona in July 1999 and Mountain View, California in March 2002. The Company's quality system processes and procedures are QS-9000 compliant for its PICmicro® 8-bit MCUs, KEELOQ® code hopping devices, Serial EEPROMs, microperipherals, non-volatile memory and analog products. In addition, Microchip's quality system for the design and manufacture of development systems is ISO 9001 certified.

MICROCHIP

Section 34. Controller Area Network (CAN)

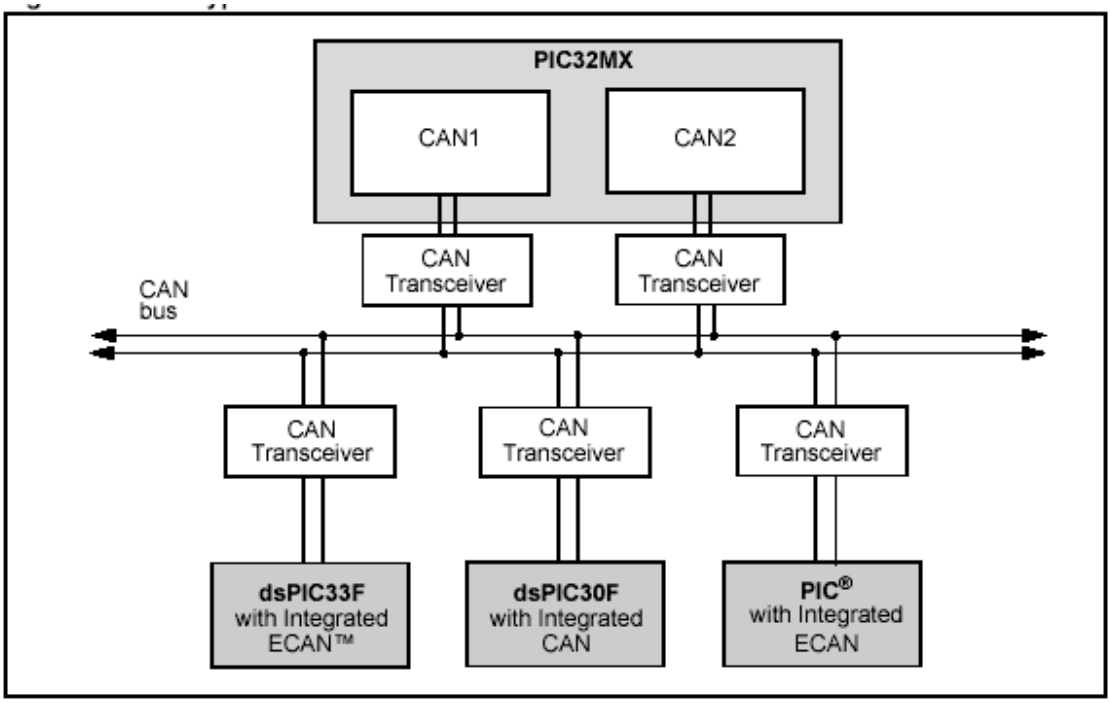
ハイライト

マニュアルのこのセクションは次のトピックを含みます：

34.1 イントロダクション

PIC32MXコントローラーエリアネットワーク（CAN）モジュールは産業用な、そして自動車アプリケーションに主に使われるCAN 2.0Bプロトコルを実装します。この非同期のシリアルデータコミュニケーションプロトコルは電気的ノイズが多い環境で信頼できるコミュニケーションを提供します。 PIC32MXデバイスファミリーは最高2つのCANモジュールを統合します。 図34-1が典型的なCANバストポロジを例示します。

図34-1：典型的なCANバスネットワーク



CANモジュールは次の重要な特徴をサポートします：

•標準準拠：

- フルCAN 2.0B準拠
- プログラマブルビットレート最高1Mbps

• メッセージ受信と送信：

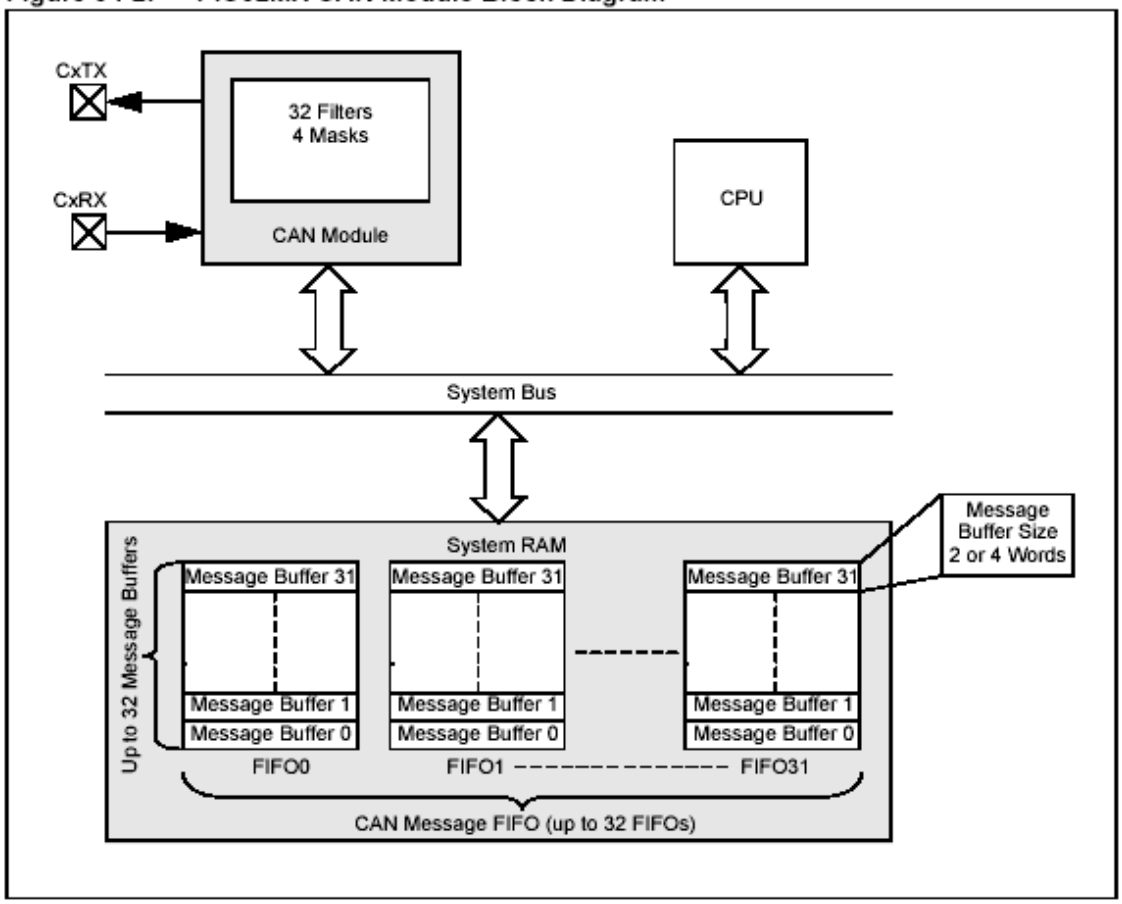
- 32メッセージFIFO
- それぞれのFIFOが1024のメッセージのトータルで最高32のメッセージを持つことができます
- FIFOは送信メッセージFIFOあるいは受信メッセージFIFOが可能です
- メッセージFIFOののためのユーザー定義プライオリティレベルは送信のために使われます
- メッセージフィルタリングのための32のアクセプタンスフィルター
- メッセージフィルタリングのための4つのアクセプタンスフィルターマスクレジスタ
- リモート送信リクエストのための自動応答
- DeviceNet™アドレッシングサポート

•追加の特徴：

- ループバック、リスンオールメッセージとセルフテストのためのリスンオンリーモード、システム診断とバスモニタリング
- ローパワーのオペレーティングモード
- CANモジュールはPIC32MXシステムバス上でバスマスターです
- DMAの使用は必要とされません
- 専用のタイムスタンプタイマー
- データオンリーメッセージ受信モード

図３４-２がCANモジュールの一般的な構造を例示します。

Figure 34-2: PIC32MX CAN Module Block Diagram



CANモジュールはプロトコルエンジン、メッセージアクセプタンスフィルターとメッセージアセンブリバッファで構成されます。プロトコルエンジンは（CANバス2.0Bプロトコルに従って）CANバスからのメッセージを送受信します。受信メッセージは受信メッセージアセンブリバッファで受信されます。 受信メッセージはそれからメッセージアクセプタンスフィルターによってフィルターされます。それがプロトコルエンジンによって処理されるとき、送信メッセージアセンブリバッファは送信されるメッセージとして保持します。CANメッセージバッファはシステムRAM上に存在します。 CANモジュールにCANメッセージバッファはありません。 そのために、すべてのメッセージはシステムRAMにストアされます。 CANモジュールはPIC32MXシステムバス上でバスマスターであって、そして要求されるように、システムRAMにデータをリードとライトをするでしょう。 CANモジュールはそのオペレーションのためにDMAを使いません。 CANモジュールはDMAあるいはＣＰＵ介入無しでシステムRAMからのメッセージをフェッチするでしょう。

34.3 CANレジスタ

CANモジュールレジスタは次のグループ中にそれらの機能によって分類されることができます：

- モジュールとCANビットレートコンフィギュレーションレジスタ
- 割り込みとステータスレジスタ
- マスクとフィルターコンフィギュレーションレジスタ
- FIFOコントロールレジスタ

34.3.1 モジュールとCANビットレートコンフィギュレーションレジスタ

• CiCON: CANコントロールレジスタ

このレジスタはCANモジュールオペレーショナルモードとDeviceNetアドレッシングをセットアップするために使われます。

• CiCFG: CANのボーレートコンフィギュレーションレジスタ

このレジスタは、ボーレートプレスケーラーを使って、それぞれのタイムカンタムのピリオドを達成するためにコントロールビットを含んでいて、そしてタイムカンタム（時間量）の項目で同期化ジャンプ幅（SJW）を指定します。 それは同じくそれぞれのCANビットセグメントでのタイムカンタム（時間量）数をプログラムするために使われます、伝搬とフェーズセグメント1と2含んでいる。

34.3.2 割り込みとステータスレジスタ

• CiINT: CAN割込みレジスタ

このレジスタは種々のCANモジュール割り込みソースがイネーブルで、そしてディスエーブルであることを可能にします。 それは同じく割り込みステータスフラグを含んでいます。

• CiVEC: CAN割込みコードレジスタ

このレジスタはCANモジュール割り込みソースとメッセージフィルターヒットに関するインフォメーションを提供するステータスビットを提供します。 これらの値は、異なるケースを処理することに対して、ジャンプテーブルを実行するために使われることができます。

• CiTREC: CAN送信/受信エラーカウントレジスタ

このレジスタは送信に関するインフォメーションと受信エラーカウンタ値を提供します。 それは同じく種々の警告ステートを示すビットを持っています。

• CiFSTAT: CANFIFOステータスレジスタ

このレジスタはすべてのFIFOのために割り込みステータスフラグを含んでいます。

• CiRXOVF: CAN受信FIFOオーバフローステータスレジスタ

このレジスタはすべてのFIFOのためにオーバーフロー割り込みステータスフラグを含んでいます。

• CiTMR: CANタイマレジスタ

このレジスタはCANメッセージタイムスタンプタイマーとプレスケーラーを含みます。

34.3.3 マスクとフィルターコンフィギュレーションレジスタ

• CiRXMn: CANアクセプタンスフィルターマスク nレジスタ（n = 0、1、2あるいは3）

これらのレジスタはフィルターマスクのコンフィギュレーションを許容します。 4つのマスクの合計が利用可能です。

• CiFLTCONn: CAN アクセプタンスフィルターコントロールレジスタ n（n = 0から7まで）

これらのレジスタはフィルターでFIFOとマスクのアソシエーションを許します。 フィルターがどんな1つのマスクでともアソシエーションされることができます。 それは同じくフィルターイネーブル/ディスエーブルビットを含んでいます。

• CiRXFn: CANアクセプタンスフィルターnレジスタ (n ＝ 0から31まで)

これらのレジスタは受信メッセージに適用されるフィルターを指定します。 32のフィルターの合計が利用可能です。

ノート: レジスタアイデンテファイアーで示される「i」はCAN1あるいはCAN2を意味します

34.3.4 FIFOコントロールレジスタ

• CiFIFOBA: CANメッセージバッファベースアドレスレジスター

このレジスタはCANメッセージバッファエリアのベース (スタート) アドレスを持ちます。 これは物理アドレスです。

• CiFIFOCONn: CAN FIFOコントロールレジスタ (n ＝ 0から31まで)

これらのレジスタはCANメッセージFIFOのコントロールとコンフィギュレーションを許容します。

• CiFIFOINTn: CAN FIFO割り込みレジスタ (n ＝ 0から31まで)

これらのレジスタは個別のFIFO割り込みソースがイネーブルにされるか、あるいはディスエーブルにされることを許容します。 ・それらは同じく割り込みステータスビットを含んでいます。

• CiFIFOUAn: CAN FIFOユーザーアドレスレジスタ (n ＝ 0から31まで)

これらのレジスタはそこから次のメッセージが読まれることができる、あるいは次のメッセージが書き込まれるべきであるCANメッセージFIFOでメモリーロケーションのアドレスを提供します。

• CiFIFOCIn: CANモジュールメッセージインデックスレジスタ (n ＝ 0から31まで)

これらのレジスタはCANモジュールが送信するであろう次のメッセージについて、あるいは次の受信メッセージがセーブされるであろうところに (メッセージFIFOで) メッセージバッファインデックスを提供します。表34-1がすべてのCAN関連レジスタの要約を提供します。 対応するレジスタが要約とその後に、それぞれのレジスタの詳細な記述の後に現われます。 すべての実装されていないレジスタと／あるいはビットはレジスタ中でゼロとリードされる。

Table 34-1: CAN Controller Register Summary

Address Offset	Name	Bit Range	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
0x00	C1CON[1,2,3]	31:24	—	—	—	—	ABAT	REQOP<2:0>		
		23:16	OPMOD<2:0>			CANCAP	—	—	—	—
		15:8	ON	—	SIDLE	—	CANBUSY	—	—	—
		7:0	—	—	—	DNCNT<4:0>				
0x10	C1CFG[1,2,3]	31:24	—	—	—	—	—	—	—	—
		23:16	—	WAKFIL	—	—	—	SEG2PH<2:0>		
		15:8	SEG2PHTS	SAM	SEG1PH<2:0>			PRSEG<2:0>		
		7:0	SJW<1:0>		BRP<5:0>					
0x20	C1INT[1,2,3]	31:24	IVRIE	WAKIE	CERRIE	SERRIE	RBOVIE	—	—	—
		23:16	—	—	—	—	MODIE	CTMRIE	RBIE	TBIE
		15:8	IVRIF	WAKIF	CERRIF	SERRIF	RBOVIF	—	—	—
		7:0	—	—	—	—	MODIF	CTMRIF	RBIF	TBIF
0x30	C1VEC[1,2,3]	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	—	—	—	FILHIT<4:0>				
		7:0	—	ICODE<6:0>						
0x40	C1TREC[1,2,3]	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	TXBO	TXBP	RXBP	TXWARN	RXWARN	EWARN
		15:8	TEC<7:0>							
		7:0	REC<7:0>							
0x50	C1FSTAT[1,2,3]	31:24	FIFOIP31	FIFOIP30	FIFOIP29	FIFOIP28	FIFOIP27	FIFOIP26	FIFOIP25	FIFOIP24
		23:16	FIFOIP23	FIFOIP22	FIFOIP21	FIFOIP20	FIFOIP19	FIFOIP18	FIFOIP17	FIFOIP16
		15:8	FIFOIP15	FIFOIP14	FIFOIP13	FIFOIP12	FIFOIP11	FIFOIP10	FIFOIP9	FIFOIP8
		7:0	FIFOIP7	FIFOIP6	FIFOIP5	FIFOIP4	FIFOIP3	FIFOIP2	FIFOIP1	FIFOIP0
0x60	C1RXOVF[1,2,3]	31:24	RXOVF31	RXOVF30	RXOVF29	RXOVF28	RXOVF27	RXOVF26	RXOVF25	RXOVF24
		23:16	RXOVF23	RXOVF22	RXOVF21	RXOVF20	RXOVF19	RXOVF18	RXOVF17	RXOVF16
		15:8	RXOVF15	RXOVF14	RXOVF13	RXOVF12	RXOVF11	RXOVF10	RXOVF9	RXOVF8
		7:0	RXOVF7	RXOVF6	RXOVF5	RXOVF4	RXOVF3	RXOVF2	RXOVF1	RXOVF0
0x70	C1TMR[1,2,3]	31:24	CANTS<15:8>							
		23:16	CANTS<7:0>							
		15:8	CANTSPRE<15:8>							
		7:0	CANTSPRE<7:0>							
0x80	C1RXM0[1,2,3]	31:24	SID<10:3>							
		23:16	SID<2:0>			—	MIDE	—	EID<17:16>	
		15:8	EID<15:8>							
		7:0	EID<7:0>							
0x90	C1RXM1[1,2,3]	31:24	SID<10:3>							
		23:16	SID<2:0>			—	MIDE	—	EID<17:16>	
		15:8	EID<15:8>							
		7:0	EID<7:0>							
0xA0	C1RXM2[1,2,3]	31:24	SID<10:3>							
		23:16	SID<2:0>			—	MIDE	—	EID<17:16>	
		15:8	EID<15:8>							
		7:0	EID<7:0>							

Register 34-20: CiFIFOCONn: CAN FIFO Control Register (n = 0 through 31)(1,2,3)

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 31							bit 24

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	FSIZE<4:0>(4)				
bit 23							bit 16

U-0	R/W-0	R/W-0	R/W-0	U-0	U-0	U-0	U-0
—	FRESET	UINC	DONLY(4)	—	—	—	—
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
TXEN	TXABAT(5)	TXLARB(6)	TXERR(6)	TXREQ	RTREN	TXPR<1:0>	
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

ビット31-21**実装されていない**：「0」としてリードします

ビット20-16 **FSIZE <4：0>**：FIFOサイズビット (4)

11111 ＝ FIFOは32メッセージ深さです

00010 ＝ FIFOは3メッセージ深さです

00001 ＝ FIFOは2メッセージ深さです

00000 ＝ FIFOは1メッセージ深さです

ビット15 **実装されていない**：「0」としてリードします

ビット14 **FRESET**：FIFOリセットビット

ビットがセットされるとき、1＝FIFOはリセットされるでしょう、FIFOがリセットされるときハードウェアによってクリアされます。ユーザーは0＝ノーエフェクトのどんなアクションを取る前にクリアされるこのビットをチェック (ポール) するべきです

ノート1：このレジスタは0x4バイトのオフセット値において関連づけられたクリアレジスタ (CiFIFOCONnCLR) を持っています。クリアレジスタにおけるどんなビットポジションにでも「1」を書き込むことは関連づけられたレジスタでバリッドビットをクリアするでしょう。 クリアレジスタからの読み取りは無視されるべきです。

2： このレジスタは0x8バイトのオフセット値において関連づけられたセットレジスタ (CiFIFOCONnSET) を持っています。 セットレジスタにおけるどんなビットポジションにでも「1」を書き込むことはバリッドビットを関連づけられたレジスタにセットするでしょう。 セットレジスタからの読み取りは無視されるべきです。

3： このレジスタは0xCバイトのオフセット値において関連づけられた反転レジスタ (CiFIFOCONnINV) を持っています。 反転レジスタにおけるどんなビットポジションにでも「1」を書き込むことは関連づけられたレジスタでバリッドビットを反転するでしょう。 反転レジスタからの読み取りは無視されるべきです。

4： CANモジュールがコンフィギュレーションモード (OPMOD <2：0> (CiCON <23：21>) ＝ 100) にあるとき、これらのビットはただ修正されることができるだけです。

5： メッセージが完了する（あるいはアボートする）とき、あるいはFIFOがリセットされるとき、このビットはアップデートされます。

6： このビットはこのレジスタのどんなリードででもあるいはFIFOがリセットされるときリセットされます。

レジスタ34-20：CiFIFOCONn: CAN FIFOコントロールレジスタ（n＝0から31まで）（1,2,3）（続く）

ビット13 **UINC**：増加ヘッド/テイルビット

TXEN ＝ 1：（FIFOは送信FIFOとして構成されました）

このビットがセットされるときFIFOヘッドは一つのメッセージによってインクリメントされるであろう

TXEN ＝ 0：（FIFOがFIFOを受信として構成しました）

このビットがセットされるとき、FIFOテイルは一つのメッセージによってインクリメントされるであろう

ビット12 **DONLY**：ストアメッセージデータの唯一のビット **(1)**

TXEN ＝ 1：（FIFOは送信FIFOとして構成されました）

このビットは使われなくて、そして影響を持っていません。

TXEN ＝ 0：（FIFOは受信FIFOをとして構成されました）

1 ＝ データバイトのみがFIFOにストアされるでしょう

0 ＝ 全メッセージが、アイデンテファイアーを含めてストアされる

ビット11-8 **実装されていない**：「0」としてリードします

ビット7 **TXEN**：TX / RXバッファセレクションビット

1 ＝ FIFOは送信FIFOです

0 ＝ FIFOは受信FIFOです

ビット6 **TXABAT**：メッセージアボーテッドビット **(5)**

1 ＝ メッセージはアボートさせられました

0 ＝ メッセージは成功裏に完了されます

ビット5 **TXLARB**：メッセージロストアービトレーションビット**(6)**

1 ＝メッセージは送られる間にアービトレーションを失いました

0 ＝送られる間に、メッセージがアービトレーションを失いませんでした

ビット4 **TXERR**：送信ビット間に検出されたエラー**(6)**

1 ＝メッセージが送られていた間に、バスエラーが発生しました

0 ＝メッセージが送られていた間に、バスエラーが発生しませんでした

ビット3 **TXREQ**：メッセージがリクエストを送信した

TXEN ＝ 1：（FIFOは送信FIFOとして構成されました）

このビットを「1」にセットすることはメッセージを送ることをリクエストします。

セット（「1」）がメッセージアボートをリクエストするであろう間に、ビットはFIFO上で待ち行列に入れられるすべてのメッセージが「0」にビットをクリアして成功裏に送る時に自動的にクリアするでしょう。

TXEN ＝ 0：（FIFOが受信FIFOとして構成されました）このビットは影響を持っていません。

ビット2 **RTREN**：自動RTRイネーブルビット

1 ＝ リモート送信が受信される時、TXREQはセットされるでしょう

0 ＝ リモート送信が受信される時す、TXREQは影響されないでしょう

ビット1-0 **TXPR <1：0>：**メッセージ送信プライオリティビット

11 ＝ 最高メッセージプライオリティ

10 ＝ 高い中間のメッセージプライオリティ

01 ＝ 低い中間のメッセージプライオリティ

00 ＝ 最低メッセージプライオリティ

ノート1：このレジスタは0x4バイトのオフセット値において関連づけられたクリアレジスタ（CiFIFOCONnCLR）を持っています。クリアレジスタにおけるどんなビットポジションにでも「1」を書き込むことは関連づけられたレジスタでバリッドビットをクリアするでしょう。 クリアレジスタからの読み取りは無視されるべきです。

2： このレジスタは0x8バイトのオフセット値において関連づけられたセットレジスタ（CiFIFOCONnSET）を持っています。 セットレジスタにおけるどんなビットポジションにでも「1」を書き込むことはバリッドビットを関連づけられたレジスタにセットするでしょう。 セットレジスタからの読み取りは無視されるべきです。

3： このレジスタは0xCバイトのオフセット値において関連づけられた反転レジスタ（CiFIFOCONnINV）を持っています。 反転レジスタにおけるどんなビットポジションにでも「1」を書き込むことは関連づけられたレジスタでバリッドビットを反転するでしょう。 反転レジスタからの読み取りは無視されるべきです。

4： CANモジュールがコンフィギュレーションモード（OPMOD <2：0>（CiCON <23：21>） ＝ 100）にあるとき、これらのビットはただ修正されることができるだけです。

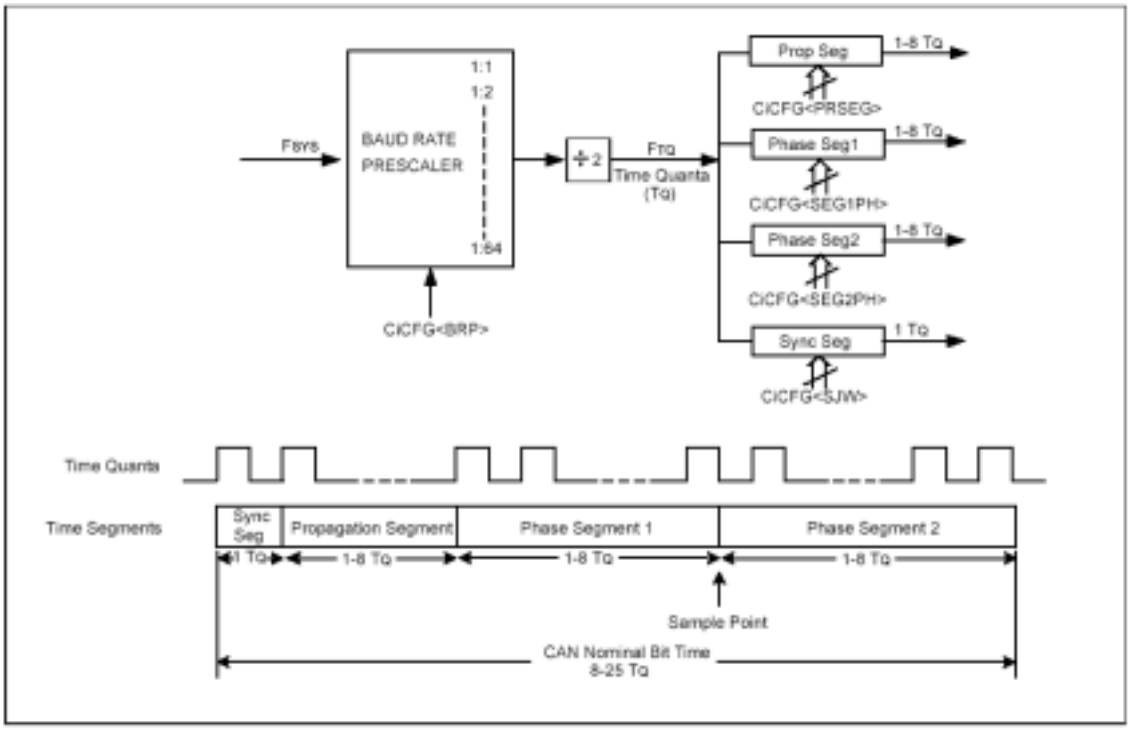
5： メッセージが完了する（あるいはアボートする）とき、あるいはFIFOがリセットされるとき、このビットはアップデートされます。

6： このビットはこのレジスタのどんなリードででもあるいはFIFOがリセットされるときリセットされます。

34.10 ビットタイミング

ごくわずかのビットレートは毎秒CANバスで送信されるビットの番号です。 名目上のビットタイム ＝ 1 ÷名目上ビットレート。発振器ドリフトあるいは伝搬遅延のためにどんな位相フェーズシフトでも補償するビットタイムに4つのタイムセグメントがあります。これらのタイムセグメントはお互いに重なりません、そしてタイムカンタム（TQ）に関して表されます。 1つのTQが発振器クロックから得られる時間の固定ユニットです。名目のビットタイムのタイムクアンタム(時間量)の合計数は8と25TQの間にプログラムされなくてはなりません。 図34-37がタイムクアンタム（FTQ）がどのようにシステムクロックから得られるか、そして異なったタイムセグメントが同じくどのようにプログラされるか示します。

図34-37：CAN™ビットタイミング



34.10.1 ビットセグメント

それぞれのビット伝送時間が4つのタイムセグメントから成り立ちます：

• **同期化セグメント** — このタイムセグメントはCANバスで接続された異なったノードを同期させます。 ビットエッジがこのセグメント中であることを予想されます。 CANプロトコルに基づいて、同期化セグメントは1つのタイムカンタム(時間量)であると考えられます。

• **伝搬セグメント** —このタイムセグメントはバスラインのためにあるいは種々のトランシーバーのために起こってもよい遅延がそのバスで接続したどんな時間でも補償します。

• **フェーズセグメント1** —このタイムセグメントはエッジでフェーズシフトのために発生してもよいエラーを補償します。 このタイムセグメントはフェーズシフトもための補償するように再同期化の間に長くされることができる。

• **フェーズセグメント2** —このタイムセグメントがエッジでフェーズシフトのために発生してもよいエラーを補償します。 タイムセグメントはフェーズを補償するために再同期化の間に短くされることができます。フェーズセグメント2タイムはいずれかプログラマブルであるために構成されるかフェーズセグメント1タイムによって指定されることができます。

34.10.2 サンプルポイント

サンプルポイントはサンプルが取得される、そしてバスステートがリードされて、そして解釈されるCANビットタイムにおけるポイントです。 それはフェーズセグメント1とフェーズセグメント2の間に位置しています。CANのボーレートコンフィギュレーションレジスタ（CiCFG <14>）でサンプルCANバスライン（SAM）ビットによって構成されるように、CANバスは1度、あるいは3度サンプルポイントでサンプルされることができます。

• もしCiCFG ＜SAM ＞＝1であるなら、CANバスはサンプルポイントで3回サンプルされます。 3つのサンプルの最も普通のものはビット値を決定します。

PIC32MX Family Reference Manual

詳細はホームページ　http://www5b.biglobe.ne.jp/~tekhanzo/
をご確認ください。

不許複製

PIC32MX Section34 CAN モジュール

完全日本語訳　サンプル

発行　2010 年 3 月 19 日　初版発行
翻訳者　諸橋　義明
発行元　テック・ハンゾウガネ(個人事業者)
〒940-0213
新潟県長岡市栃尾山田町 6-53
TEL　0258 (53) 0082　E-mail　tekhanzo@mta.biglobe.ne.jp
HP アドレス　http://www5b.biglobe.ne.jp/~tekhanzo/

MEMO