

ANNEXE

LE MICROCONTROLEUR PIC32MX

1 Introduction :

Microchip a une nouvelle gamme de microcontrôleurs à 32 bits avec peu de broches, dans des boîtiers de 5 mm x 5 mm, à faible coût avec de fortes contraintes.

Avec l'USB On-The-Go (OTG), ils sont idéaux pour le développement d'accessoires audio et autres applications grand public.

Ce μ C a une architecture RISC 32-bits et est basé sur un cœur MIPS 32 avec un pipeline à 5 étages.

Il peut atteindre des performances de 1,56DMIPS/MHz (soit 125MIPS à 80MHz). Il possède un mode MIPS 16e permettant de réduire jusqu'à 40 la taille du code. Il est capable d'exécuter une multiplication 32x16 à chaque coup d'horloge.

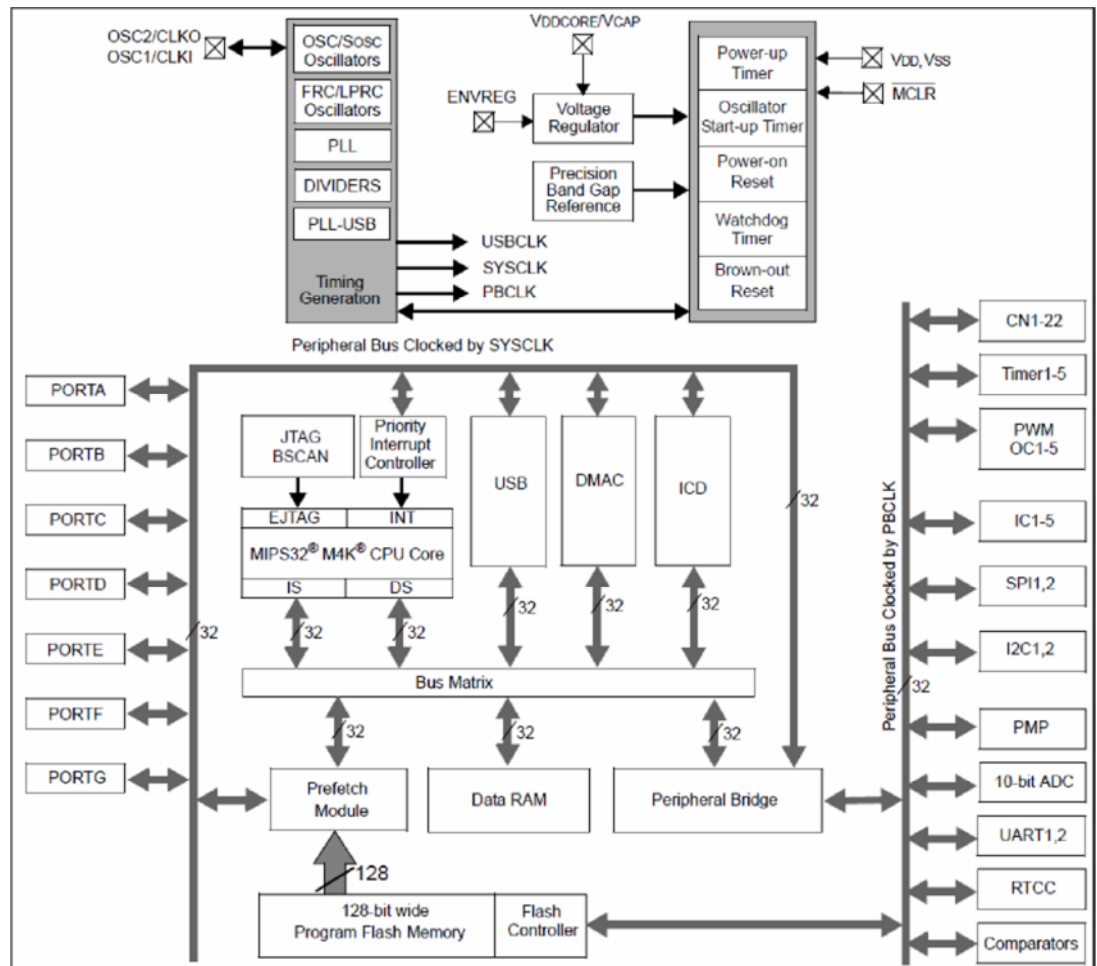


Figure 1.1 : Schéma bloc du PIC32MX

Un schéma bloc est donné à la Figure 1.1 et un schéma des pattes, à la Figure 1.2. Ce μ C possède plusieurs périphériques intégrés tels que :

- des entrées et sorties logiques,
- des interfaces séries (UART, I2C, SPI),
- des canaux DMA programmables,
- un module USB incluant 1 canaux DMA dédiés,
- un convertisseur A/N 10 pouvant être connecté à 16 entrées,
- des timers associés à des circuits de capture et comparaison,
- un module de port parallèle maître (PMP) permettant diverses configurations, des sorties PWM,
- une mémoire Flash de 512Ko,
- et une RAM de 128Ko.

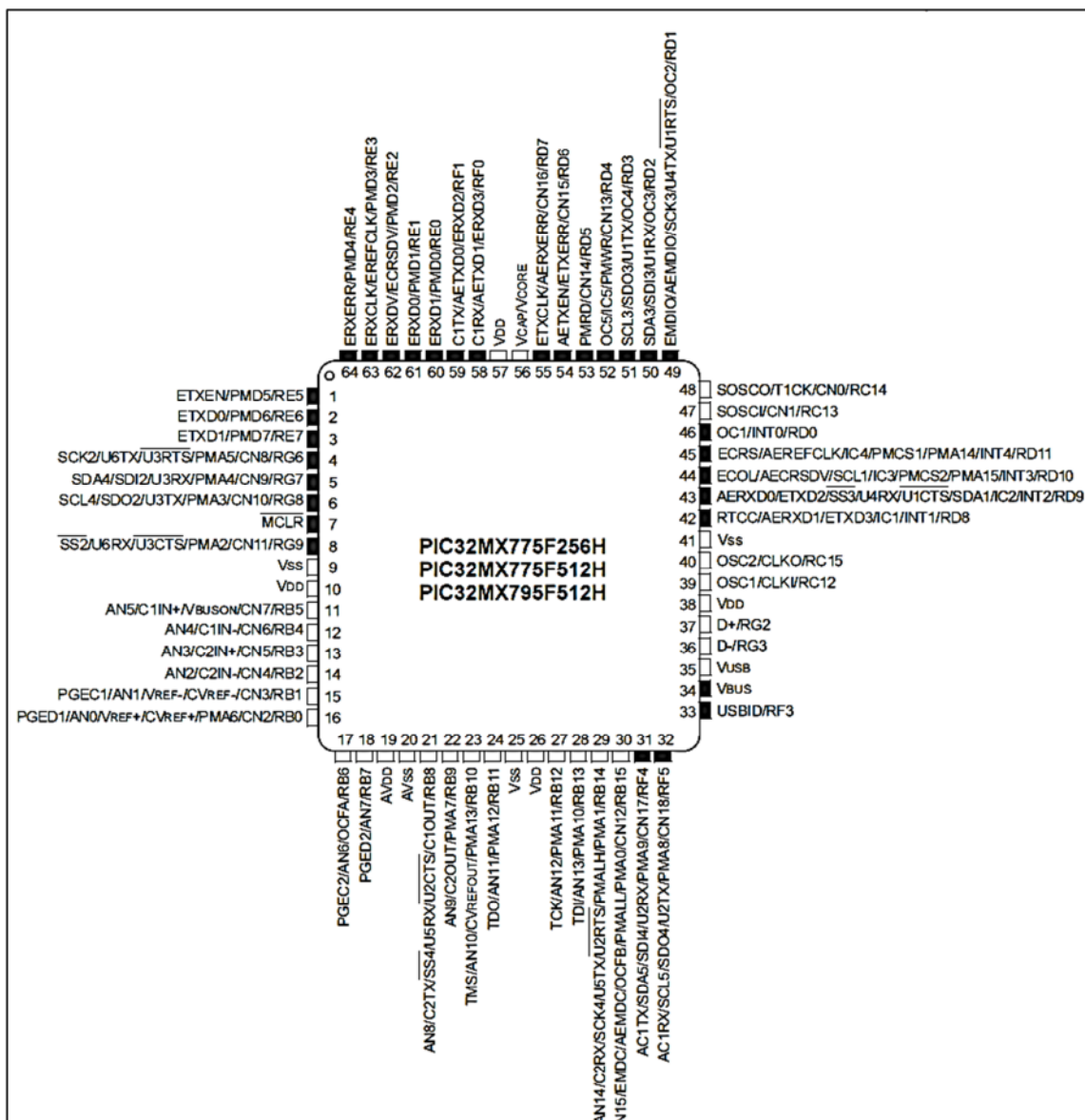


Figure 1.2 : Schéma des pattes du PIC32MX795F512H

2 DMA(Direct Memory Access)

Le contrôleur de DMA (Direct Memory Access) permet de transférer des données entre des périphériques internes ou la mémoire sans l'intervention du CPU. La source et la destination d'un transfert DMA peuvent être n'importe quel module du PIC32MX mappé en mémoire comme la mémoire, le SPI, l'I²C ou l'UART.

Le PIC 32MX759F512H possède 4 canaux DMA. Chacun d'eux ont un pointeur d'adresse pour la source et un pour la destination qui sont automatiquement incrémentés. Les données sont transférées par octet et un transfert peut comprendre jusqu'à 256 octets.

La source et la destination ont des tailles indépendamment configurables qui sont indépendantes du nombre d'octets à transférer.

Un transfert d'une cellule peut être démarré par le software ou par n'importe quelle requête d'interruption du PIC 32MX. La taille d'une cellule est définie par le registre DCH x CSIZ. Un transfert d'un bloc définit le nombre d'octets transférés lorsqu'un canal est activé. Ce nombre est égal au maximum entre les tailles de la source et de la destination. Un transfert d'un bloc comprend un ou plusieurs transferts d'une cellule.

Dès qu'un transfert est initialisé, le contrôleur de DMA effectuera un transfert d'une cellule et le canal restera activé jusqu'à la fin du transfert d'un bloc. Tant qu'un canal est désactivé, tous transferts ne seront pas effectués jusqu'à ce que le canal soit réactivé.

Un transfert DMA peut être arrêté par le software, par n'importe quelle requête d'interruption ou lorsque une donnée transférée est égale à un pattern. Le transfert sera également arrêté si une erreur d'adresse est détectée.

Un vecteur d'interruption est associé à chaque canal DMA qui possède plusieurs sources d'interruptions configurables.

Lorsque des transferts de plusieurs canaux DMA sont en attente, l'arbitration du démarrage d'un canal est effectuée selon la priorité configurée de chaque canal.

La fin d'un transfert d'un bloc d'un canal peut automatiquement démarrer un transfert d'un autre canal. Ce mode est appelé « Channel chaining ».

Le numéro du dernier canal et l'adresse du dernier transfert réalisé sont disponibles pour le débogage, dans des registres spécifiques (respectivement DMASTAT et DMAADDR).

Le module DMA intègre un module de fonctions spéciales partagé par tous les canaux. Ces fonctions sont les calculs de code détecteur d'erreur de type CRC.

Le diagramme de la Figure 1.3 résume le principe d'un transfert DMA .

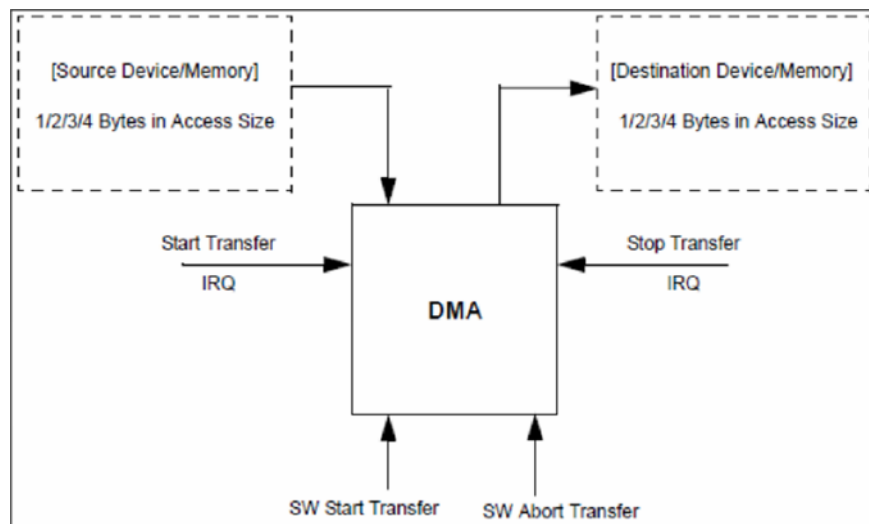


Figure 1.3 : Fonctionnement du DMA

2.1 Registres de contrôles :

Le module DMA comporte 3 registres de contrôle généraux et 12 registres spécifiques à chaque canal :

- DMACON: registre de configuration générale
- DMASTAT : registre de statut général
- DMAADDR : registre contenant l'adresse de l'accès du DMA le plus récent
- DCHxCON : registre de configuration du canal x
- DCHxECON : registre de configuration des événements démarrant et arrêtant un transfert
- DCHxINT : registre de contrôle des interruptions du canal x
- DCHxSSA : registre de l'adresse de départ de la source
- DCHxDSA : registre de l'adresse de départ de la destination
- DCHxSSIZ : registre de la taille de la source
- DCHxDSIZ : registre de la taille de la destination
- DCHxSPTR : registre du pointeur de la source
- DCHxDPTR : registre du pointeur de la destination
- DCHxCSIZ : registre de la taille de la cellule
- DCHxCPTR : registre du pointeur de la cellule
- DCHxDAT : registre contenant le pattern à reconnaître

2.2 Initialisation d'un canal DMA

La procédure à suivre pour initialiser le canal DMA x est la suivante :

- Désactivation de l'interruption du canal DMA x ;
- Remettre à 0 le flag d'interruption du canal DMA x ;
- Activer le contrôleur DMA (DMACON) ;
- Configurer le mode de fonctionnement d'un canal DMA (DCH x CON) ;
- Configurer le mode de démarrage du transfert (DCHxECON), ne mettez pas le bit CFORCE à 1 ;
- Initialiser l'adresse physique de départ de la source et de la destination (DCH xSSA et DCHxDSA) ;
- Initialiser la taille de la source, de la destination et de la cellule (DCHxSSIZ, DCHxDSIZ et DCHxCSIZ) ;
- Effacer tous les évènements existant et activer les sources d'interruptions voulues (DCHxINT) ;
- Si besoin, activer l'interruption du canal ;
- Activer le canal (DCHxCON) ;
- Si démarrage par software, démarrer le transfert à l'aide du bit CFORCE de DCHxECON.

Un transfert d'une cellule est initialisé soit en mettant à 1 le bit CFORCE, soit quand l'évènement configurer dans CHSIRQ a lieu et que SIRQEN est à 1 (DCHxECON).

Le contrôleur DMA transfèrera la taille d'une cellule lorsqu'un transfert est initialisé. Le canal restera activé tant que le canal DMA n'a pas transféré le maximum de DCHxSSIZ et de DCHxDSIZ (taille du transfert d'un bloc).

Si la taille de la cellule est plus grande que la taille de la source et de la destination, le maximum de DCHxSSIZ et de DCHxDSIZ sera transféré.

Lorsqu'un transfert d'une cellule est terminé, le canal retournera dans l'état inactif et attendra qu'un nouvel évènement pour démarrer un transfert d'une cellule.

Lorsqu'un transfert d'un bloc est terminé ou arrêté, le canal est automatiquement désactivé en hardware en mettant CHEN à 0.

Chaque canal garde une trace du nombre d'octets transférés de la source et de la destination dans les registres DCHxSPTR et DCHxDPTR.

Une requête de transfert DMA peut être réinitialisée lors :

- D'une écriture dans le bit CABORT (DCHxECON)
- D'une reconnaissance du pattern si celle-ci est activée ;
- De l'apparition d'un évènement d'interruption correspondant à celui choisi dans CHAIRQ et que AIRQEN est activé ;
- D'une détection d'une erreur d'adresse ;
- De la fin d'un transfert d'une cellule ;
- De la fin d'un transfert d'un bloc.

Lorsqu'un évènement d'interruption arrête un canal, le flag CHTAIF est mis à 1 permettant ainsi à l'utilisateur de le détecter et de prendre ses dispositions.

Les pointeurs de source et de destination sont mis à jour après chaque transaction (une lecture et une écriture). Ils sont remis à 0 lorsque :

- Le contrôleur de DMA est désactivé ;
- Un transfert de bloc est terminé ;
- Le pattern est reconnu si ce mode est activé ;
- Le transfert est annulé en mettant le bit CABORT à 1 ;
- Les adresses de départ de la source et de la destination sont modifiées.

Le contrôleur de DMA vérifie, indépendamment du contrôleur d'interruption, les flags d'interruption pour détecter les IRQ de démarrage et d'arrêt de transfert. Il n'est donc pas nécessaire d'activer les IRQ correspondante dans les registres IECx.

2.3 Reconnaissance d'un pattern :

La reconnaissance d'un pattern permet l'utilisateur d'arrêter un transfert lorsqu'un octet de donnée écrit durant une transaction est équivalent à un pattern spécifique défini dans le registre DCHxDAT. La reconnaissance d'un pattern est traitée de la même manière que la fin d'un transfert d'un bloc.

Par exemple, la reconnaissance du pattern du caractère NULL (0x00) peut être utilisée pour arrêter un transfert DMA d'un message dont on ne connaît pas la taille, envoyé via l'UART

2.4 Suspendre un transfert DMA :

L'application peut suspendre le contrôleur DMA empêchant toute transaction sur le bus, en mettant le bit SUSPEND du registre DMACON à 1.

Chaque canal peut être suspendu grâce au bit CHEN. Si un transfert DMA est en cours et que CHEN est mis à 0, la transaction courante se terminera et toutes autres transactions du canal seront suspendues. La mise à 0 de CHEN n'affectera pas les pointeurs du canal.

2.5 Priorité des canaux DMA :

Le contrôleur DMA a une priorité naturelle associée à chaque canal. Le canal 0 a la plus grande priorité naturelle.

Chaque canal a deux bits définissant sa priorité. Lorsque que plusieurs canaux ont des transferts en attente, le prochain canal à transmettre des données est sélectionné de la manière suivante :

Le canal avec la plus grande priorité terminera tous ses transferts de cellule avant les canaux de priorité moins élevée ;
Si plusieurs canaux ont la même priorité, le contrôleur cyclera entre les canaux à cette

priorité. Chaque canal avec un transfert d'une cellule en cours à la plus haute priorité sera autorisé pour une seule transaction du transfert actif d'une cellule avant que le contrôleur autorise une seule transaction du canal suivant à cette priorité ;

Si un canal de plus haute priorité demande un transfert pendant qu'un autre canal de priorité moins élevée a une transaction en cours, la transaction se terminera avant de passer au canal de priorité plus élevée.

Ces différents cas sont illustrés à la Figure 1.4.

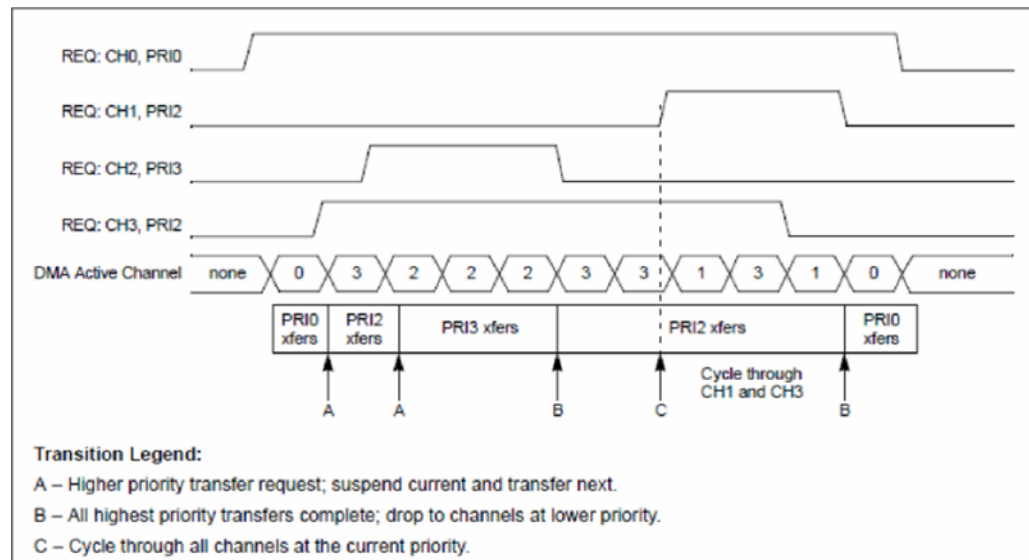


Figure 1.4 : Priorité des canaux DMA

3 Le PMP(Parallel Master Port) :

Le Port de Maître Parallèle (PMP) est un module d'entrée-sortie parallèle 8-bit/16-bit spécifiquement conçu pour communiquer avec une large variété de dispositifs parallèles comme des périphériques de communications, LCDs, des dispositifs de mémoire externes et des microcontrôleurs. Parce que les interfaces aux périphériques parallèles varient significativement, le module PMP est fortement configurable.

Les caractéristiques du module PMP incluent :

- Jusqu'à 16 lignes d'adresse programmables.
- Jusqu'à deux lignes de Sélection de Puce.
- Address auto-incrémente/auto-décrément
- Programmable multiplexage d'adresse/données
- Programmable polarité sur signaux de contrôle
- Legacy support de port parallèle d'esclave ;
- Enhanced support(assistance) d'esclave parallèle
 - adressent(abordent) au support(à l'assistance)
 - 4 octets de profondeur, auto-incrémentant amortisseur
- Schmitt la Détente ou TTL saisissent des amortisseurs
- Programmable Attendent états
- Freeze option pour débogage dans-circuit

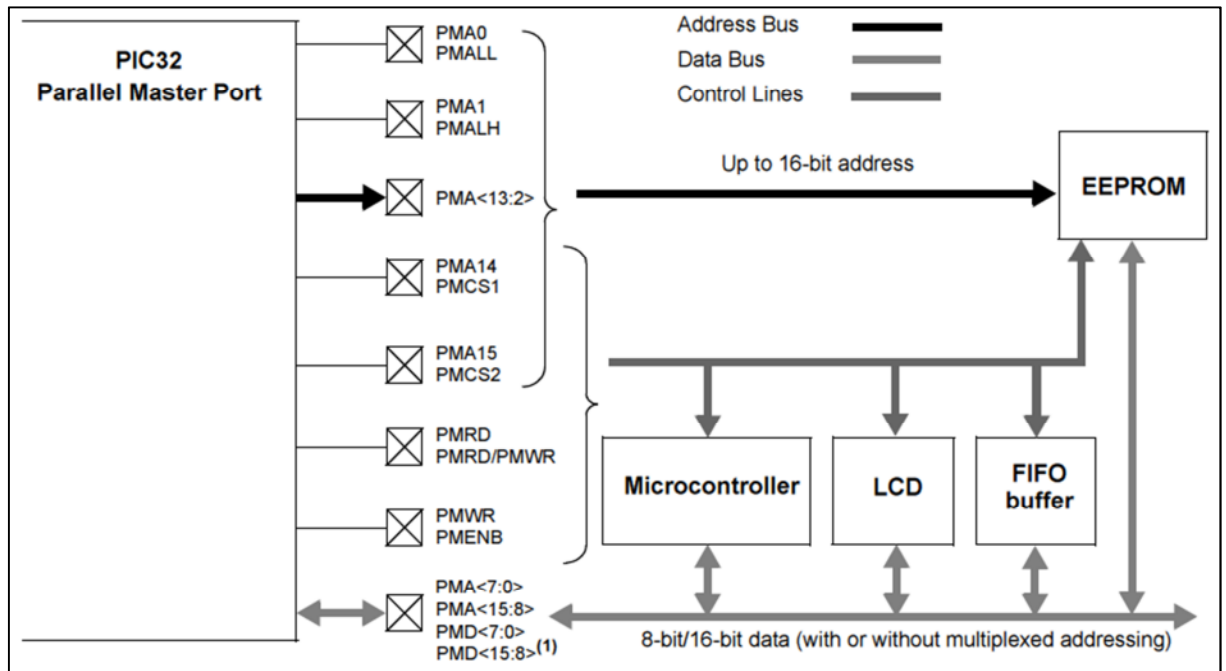


Figure 1.5

3.1 Registre de contrôle :

Le module PMP utilise ceux des Registres de Fonction Spéciaux (SFRs) :

- **PMCON** : Registre de Contrôle de Port Parallèle Ce registre contient les bits qui contrôlent beaucoup de fonctionnalité de base du module.
- **PMMODE** : Registre de Mode de Port Parallèle

Ce registre(enregistreur) contient les bits qui contrôlent les modes opérationnels du module.

- **PMADDR** : Registre d'Adresse de Port Parallèle
- **PMDOUT** : Registre de Sortie de données de Port Parallèle
- **PMDIN** : Registre de Saisie de données de Port Parallèle
- **PMAEN** : Registre activation de pin de Port Parallèle.
- **PMSTAT** : Registre de Statut de Port Parallèle (modes d'Esclave seulement)

Ce registre(enregistreur) contient des bits de statut associés à buffered des modes d'exploitation quand le port fonctionne comme un port d'esclave.

3.2 Modes de fonctionnement de maître :

Dans ses modes de maître, le module PMP peut fournir un 8 bit ou 16 bit de données, Jusqu'à 16 bits d'adresse et tous les signaux de contrôle nécessaires d'opérer une variété de dispositifs parallèles externes comme des dispositifs de mémoire, des périphériques et des microcontrôleurs d'esclave.

Les modes de maître PMP fournissent une interface simple pour lire et écrire des données, mais ne pas exécuter des instructions de programme de dispositifs externes, comme SRAM ou des Mémoires flash.

Parce qu'il y a un certain nombre de dispositifs parallèles avec une variété de méthodes de contrôle, le module PMP est conçu pour la flexibilité pour accueillir une gamme de configurations.

Certaines de ces caractéristiques incluent :

- modes de données 8bit et 16 bits
- Configurable multiplexage d'adresse/données
- Jusqu'à deux lignes de Sélection de Puce
- Jusqu'à 16 lignes d'adresse sélectionnables
- Adresse auto-incrémentent et l'auto-décroissance
- Polarité sélectionner sur toutes les lignes de contrôle
- Configurable les états aux étapes Attendent .

3.3. Configuration de PMP :

Le PMP dans le mode de Maître supporte des données avec les largeurs de 8 et 16 bits. Par défaut, la largeur de données est 8 bit , $\text{MODE16 bit (PMMODE} < 10 >) = 0$. Pour choisir une largeur de données de 16 bit , mettez $\text{MODE16} = 1$. Quand configuré dans le mode de Données de 8 bit, 8 bits supérieurs du bus de données, $\text{PMD} < 15:8 >$, ne sont pas contrôlés par le module PMP et sont disponibles comme des épingles d'entrée-sortie de but générales.

3.3.1 Chip sélecte

Deux lignes de Sélection de Puce, PMCS1 et PMCS2, sont disponibles pour des modes de maître. Ces lignes sont communiquées en multiplex avec les bits les plus significatifs (MSBs) de bus d'adresse A14 et A15.

Quand' une pin est configurée comme une Puce de Sélection, il n'est inclus dans aucune adresse auto-incrément/décrément. Il est possible de permettre tant PMCS2 QUE PMCS1 comme des Sélections de Puce, ou permettre seulement PMCS2 comme une Sélection de Puce, permettant PMCS1 de fonctionner strictement comme la ligne d'adresse A14.

Il n'est pas possible de permettre PMCS1 seul. Les signaux de Sélection de Puce sont configurés utilisant la Fonction de Sélection de Puce mord CSF $< 1:0 >$ ($\text{PMCON} < 7:6 >$).

CSF<1 :0>	Fonction
10	PMCS2,PMCS1=ACTIVER
01	PMCS2= ACTIVER,PMCS1=A14
00	PMCS2=A15,PMCS1=A14

Tableau 1-1 Control de sélection

3.3.2 Contrôle de PIN:

Il y a plusieurs bits disponibles pour configurer la présence ou l'absence de contrôle et des signaux d'adresse dans le module. Ces bits sont PTWREN (PMCON < 9 >), PTRDEN (PMCON < 8 >) et PTEN <15:0> (PMAEN < 15:0 >). Ils fournissent la capacité de conserver des pin pour d'autres fonctions et permettre à la flexibilité de contrôler l'adresse externe. Quand n'importe lequel de ces bits est mis, la fonction associée est présente sur son pin associée; quand dégager, l' pin associée retourne à sa fonction de port d'entrée-sortie définie.

Le cadre d'un bit de PTEN permettra les pins associée comme une adresse pin et conduit les données correspondantes contenues dans le registre de PMADDR. Le dégagement de n'importe quel bit de PTEN forcera le pin de retourner à sa fonction d'entrée-sortie originale.

Pour les pin configurées comme Sélection de Puce (PMCS1 ou PMCS2) avec la correspondance PTEN bit, Les pin de Sélection de puce conduisent des données inactives quand un l'opération lire ou écriture n'est pas exécutée.

3.3.3 Contrôle de lire/écrire

Le module PMP supporte deux méthodes de signalisation distinct lire/écrire. Dans le mode de Maître 1, lire/écrire est combiné dans une seule ligne de contrôle PMRD/PMWR; une deuxième ligne de contrôle, PMENB, détermine quand l'action lire ou écriture doit être pris. Dans le mode de Maître 2, lire et écriture (PMRD et PMWR) est fourni sur des épingles séparées.

3.3.4 Polarité de ligne de contrôle :

Tous les signaux de contrôle (PMRD, PMWR, PMENB, PMALL, PMALH, PMCS1 et PMCS2) peuvent être individuellement configurés pour la polarité positive ou négative. La configuration est contrôlée par des bits séparés dans le registre PMCON, comme indiqué dans la Table 1-2.

Configuration du polarité	PMCON	polarité positive	polarité négative
PMRD	RDSP	1	0
PMWR	WRSP	1	0
PMALL	ALP	1	0
PMALH	ALP	1	0
PMCS1	CS1P	1	0
PMCS2	CS2P	1	0

Tableau 1-2

3.3.5 AUTO-INCREMENT/DECREMENT

Tandis que le module PMP opère dans un des modes de maître, les bits(PMMODE < 12:11 >) INCM <1:0> contrôlent le comportement de la valeur d'adresse. L'adresse dans le registre de PMADDR peut être faite pour AUTO-INCREMENT/DECREMENT, indépendamment de la largeur de données de transfert, après chaque opération lu et écrire est complétée et le bit (PMMODE < 15 >) va 'à 0'.

INCM<1:0>	Fonction
10	DECREMENT
01	INCREMENT
00	

Tableau 1-3

Si les signaux de Puce de Sélection sont mis hors de service et configurés comme des bits d'adresse, les bits participeront à l'incrément et décréments .

4 Timer :

Le PIC32MX possède 5 Timers classés selon deux catégories qui se différencient par de faibles variantes :

- Type A (Timer1)
- Type B (Timer2, Timer3, Timer4 et Timer5)

Tous les Timers ont des compteurs 16 bits, mais ceux de type B peuvent être combinés pour former un compteur 32 bits.

La configuration des Timers est réalisée à l'aide de trois registres :

- TMRx : registre du compteur 16 bits
- PRx : registre du comparateur 16 bits
- TxCON : registre de contrôle du Timer

Des registres permettant les opérations atomiques CLR, SET et INV sont associés à chacun de ces registres.

4.1 Variantes des Timers

Les Figure 1.6 et Figure 1.7 montrent des schémas blocs respectivement du Timer de type A et des Timers de type B.

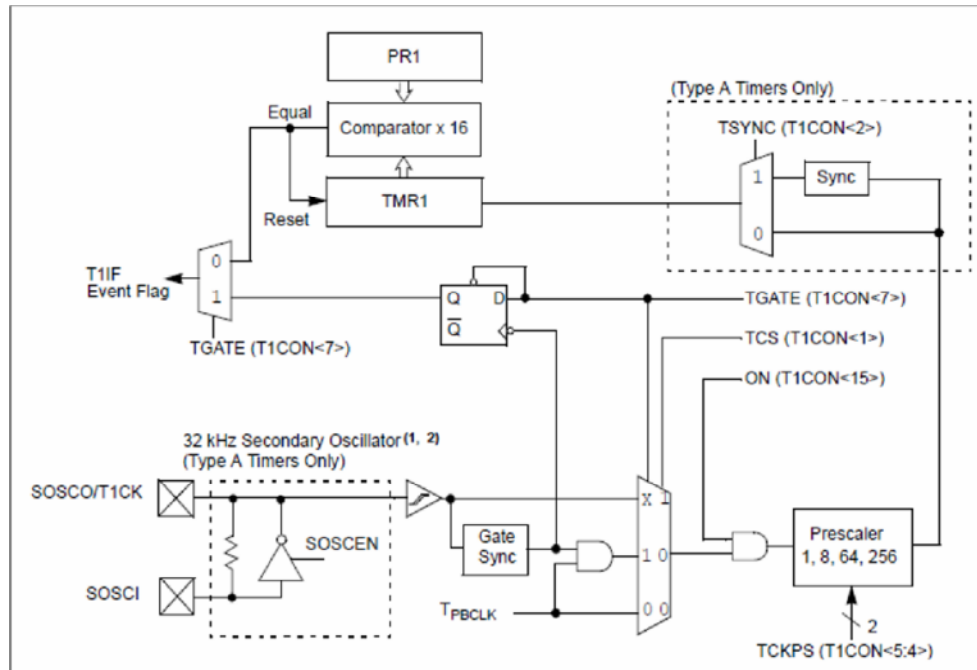


Figure 1.6 : Schéma bloc du Timer de type A

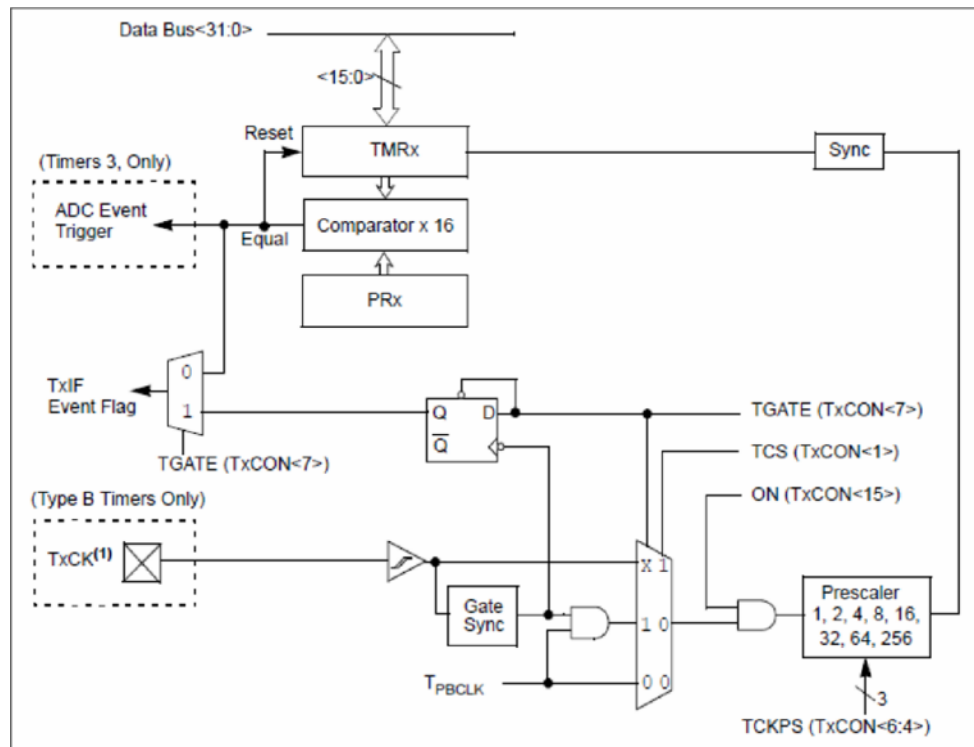


Figure 1.7: Schéma bloc du Timer de type B

Le Timer de type A possède deux fonctionnalités supplémentaires par rapport à ceux de type B :

- Il peut fonctionner dans le mode faible consommation à l'aide de l'oscillateur à 32kHz.
- Il peut fonctionner de manière asynchrone pour la source d'horloge externe alors que ceux de type B ont leur entrée externe synchronisée avec l'oscillateur du μ C.

Le « prescaler » du Timer 1 ne peut prendre que les valeurs 1, 8, 64 ou 256, tandis que les Timers de type B peuvent prendre 1, 2, 4, 8, 16, 32, 64 ou 256.

Le Timer 3 a la possibilité de déclencher automatiquement une conversion analogique/numérique dès que la comparaison est vérifiée.

4.2 Mode opératoire :

4.2.1 Timer 16 bits :

L'initialisation en mode Timer 16bits est relativement simple :

- Arrêt du Timer (TxCON) ;
- Configuration du mode de comptage (TxCON) ;
- Initialisation du Prescaler (TxCON) ;
- Remise à zéro du registre du compteur (TMRx) ;
- Chargement de la période (PRx) ;
- Initialisation de l'interruption ;
- Démarrage du Timer (TxCON).

Dans la routine d'interruption, il n'y a rien à faire du point de vue de la configuration du Timer car le registre TMRx se remet automatiquement à zéro lorsque la comparaison est vérifiée. Le compteur lui ne s'arrête jamais même pendant l'interruption, seule la remise à zéro du bit ON le permet.

4.2.2 Timer 32 bits

Un Timer 32 bits peut être formé en combinant deux Timers 16 bits de type B, un de numéro pair (TimerX) avec celui de numéro impair consécutif (TimerY). Pour cela, le bit T32 du registre de contrôle du TimerX doit être activé. Le TimerY contient alors les bits les plus significatifs et le TimerX, les bits les moins significatifs pour les registres du compteur et du comparateur. Le Tableau 1. 4 donne les combinaisons des Timers pour former un Timer 32 bits. La Figure 1.8 donne le schéma bloc du Timer 32 bits.

TimerX (LSB)	TimerY (MSB)
Timer2	Timer3
Timer4	Timer5

Tableau 1.4 : Combinaisons pour les Timers 32

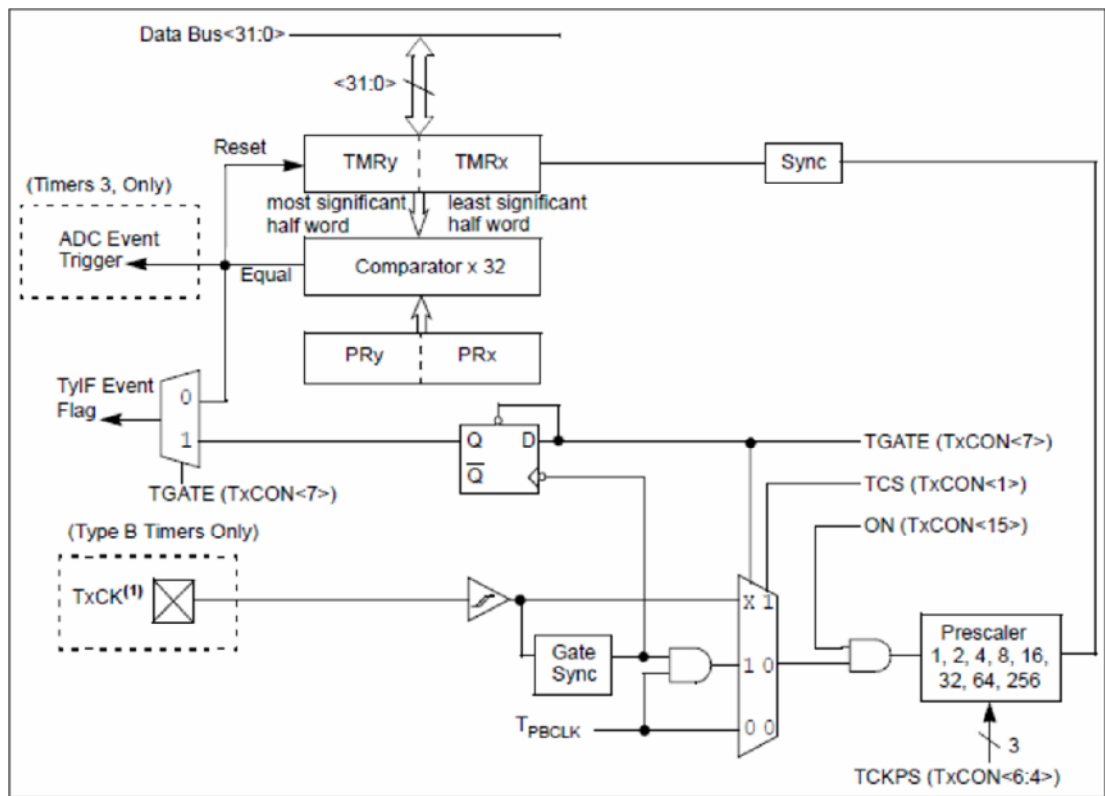


Figure 1.8 : Schéma bloc du Timer 32 bits

Lorsque des Timers sont configurés en 32 bits, seul le registre de contrôle du TimerX est requis pour initialiser et commander celui-ci. Par contre, pour les interruptions, seuls les bits correspondant au TimerY doivent être utilisés.

Les opérations d'écriture et de lecture des registres des compteurs (TMR et PR) peuvent être réalisées sur 32 bits en accédant aux registres du TimerX.

5 Ports I/O :

Le PIC32MX460F512L possède 64 pattes dont 55 peuvent être utilisé comme des entrées/sorties digitales. Les autres pattes servent à l'alimentation, au reset et à l'oscillateur. Toutes les IO sont partagées avec les périphériques (UART, SPI, ADC, «~»).

Une I/O comporte donc des multiplexeurs permettant de sélectionner sa fonction. Lorsqu'un périphérique est en fonction, les pattes correspondant ne peuvent pas être utilisées en I/O banale.

La structure d'une entrée/sortie est schématisée à la Figure 1.9.

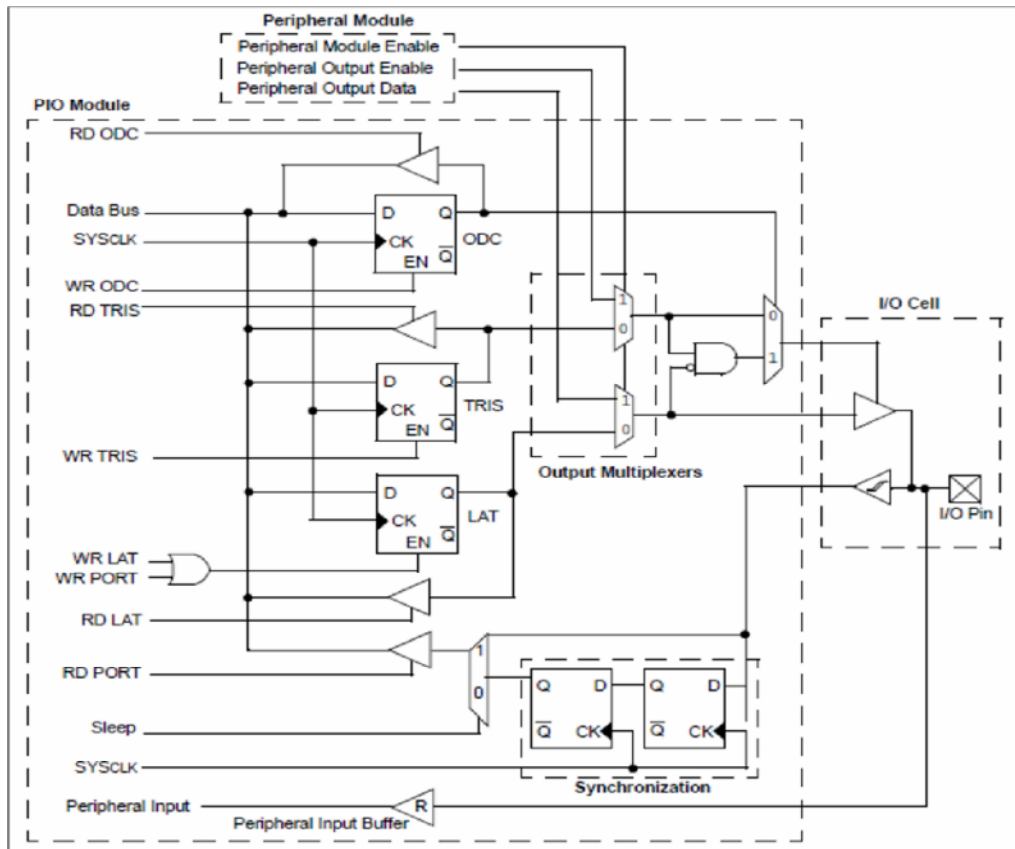


Figure 1.9 : Schéma bloc d'une I/O

Les I/O sont rassemblées en port identifié par un « R » suivi d'une lettre (A, B, C, D, E, F et G) sur la Figure 2. Un port est composé de maximum 16 I/O.

5.1 Registres de contrôle des I/O :

Quatre registres sont associés à chaque port, permettant de les configurer :

- TRISx : registre contrôlant la direction
- PORTx : registre d'I/O
- LATx : registre du « Data Latch »
- ODCx : registre configurant la sortie en drain ouvert

Le « x » représente une lettre qui identifie un port particulier (A, B,....).

Chaque I/O d'un port est associée à un bit dans chacun de ces quatre registres.

A ces quatre registres, les registres permettant les opérations atomiques CLR, SET et INV leurs sont associés.

5.1.1 Registre TRIS

Ce registre permet de définir si une patte est une entrée ou une sortie. Si un bit de ce registre est à 1, alors la patte associée est une entrée, tandis que si le bit est à 0, la patte est une sortie. Après un reset, toutes les pattes sont configurées en entrée.

Exemple de code :

Accès au registre TRIS du port B : `TRISB = 0x00F0;`

Accès au bit 2 du registre TRIS du port B : `TRISBbits.TRISB2 = 0;`

5.1.2 Registre PORT :

La donnée sur une patte est accessible par ce registre. Une lecture d'un bit de ce registre donne l'état de la patte associée (état haut ou état bas), tandis qu'une écriture impose l'état du « Data Latch » (Bistable LAT sur la Figure 1.9).

Comme il a été dit plus haut, l'accès à ce registre s'effectue en 32bits même pour une instruction ne modifiant qu'un seul bit. Ceci peut amener à des incohérences à cause de la rapidité d'exécution du μC et d'une charge capacitive importante sur la sortie. La Figure 8 montre un exemple de ce problème.

La première instruction modifie à 1 la donnée dans le « Data Latch » de la patte 0. L'état à la sortie de cette patte ne passe pas instantanément à l'état haut, car la capacité en sortie doit se charger. Le PIC32 fonctionnant à une grande fréquence (dans notre cas, le PIC32 exécute la plupart des instructions assembleurs en 16,7ns) commence à exécuter la seconde instruction qui met à 1 le « Data Latch » de la patte 1. Pour cela, il lit les 32 bits sur les pattes du port en question, mais à cet instant, la patte 0 est toujours considérée à l'état bas. Après avoir modifié la valeur du bit 1, il écrit le résultat sur les « Data Latch » du port. Il en résulte que la patte 0 est alors remise à l'état bas.

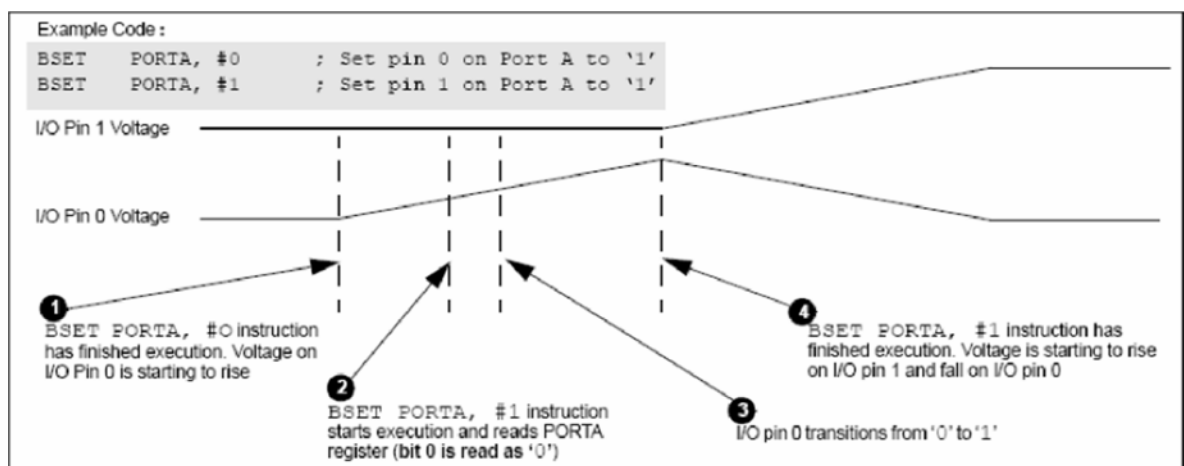


Figure 1.10 : Incohérence due à une charge capacitive

5.1.3 Registre LAT :

Ce registre élimine le problème qui peut apparaître avec les instructions « read-modify-write ». Lors d'une lecture de ce registre, la valeur est lue à la sortie du « Data Latch » au lieu de la patte.

En résumé, les différences entre les accès aux registres PORT et LAT sont :

- Une écriture dans le registre PORTx écrit la donnée dans le « Data Latch ».
- Une écriture dans le registre LATx écrit la donnée dans le « Data Latch ».
- Une lecture du registre PORTx lit la donnée sur la patte I/O.
- Une lecture du registre LATx lit la donnée stockée dans le « Data Latch ».

5.1.4 Registre ODC :

Chaque patte d'un port peut être individuellement configurée soit en sortie digitale soit en sortie à drain ouvert (Open Drain). La mise à 1 d'un bit de ce registre configure la sortie de la patte correspondante en drain ouvert.

Cette configuration offre la possibilité de générer des états de sortie supérieur à la tension d'alimentation (3,3V) en utilisant une résistance de pull-up externe. Cette configuration n'est pas supportée pour toute patte pouvant être configurée en entrée analogique. Les pattes supportant cette option s'identifient sur la Figure 2 par leur tolérance au 5V.

5.2 Multiplexage avec les périphériques internes :

Quand un périphérique est activé, le buffer de sortie de la patte associée est en général contrôlé par ce dernier, cependant certaines sorties doivent être configurées dans le code à l'aide des registres cités ci-dessus.

L'UART est un exemple de contrôle du buffer de sortie par le périphérique. Quand l'UART est activé les registres PORT et TRIS n'ont aucun effet et ne peuvent pas être utilisés pour écrire sur les pattes

RX et TX (pattes de réception et de transmission). La plupart des périphériques de communication contrôlent le buffer de sortie de leurs pattes associées.

Un exemple contraire est l'entrée d'interruption. Une entrée d'interruption peut être configurée comme une sortie, et par une écriture dans le bit LATx associé, une interruption peut être générée si elle est activée.