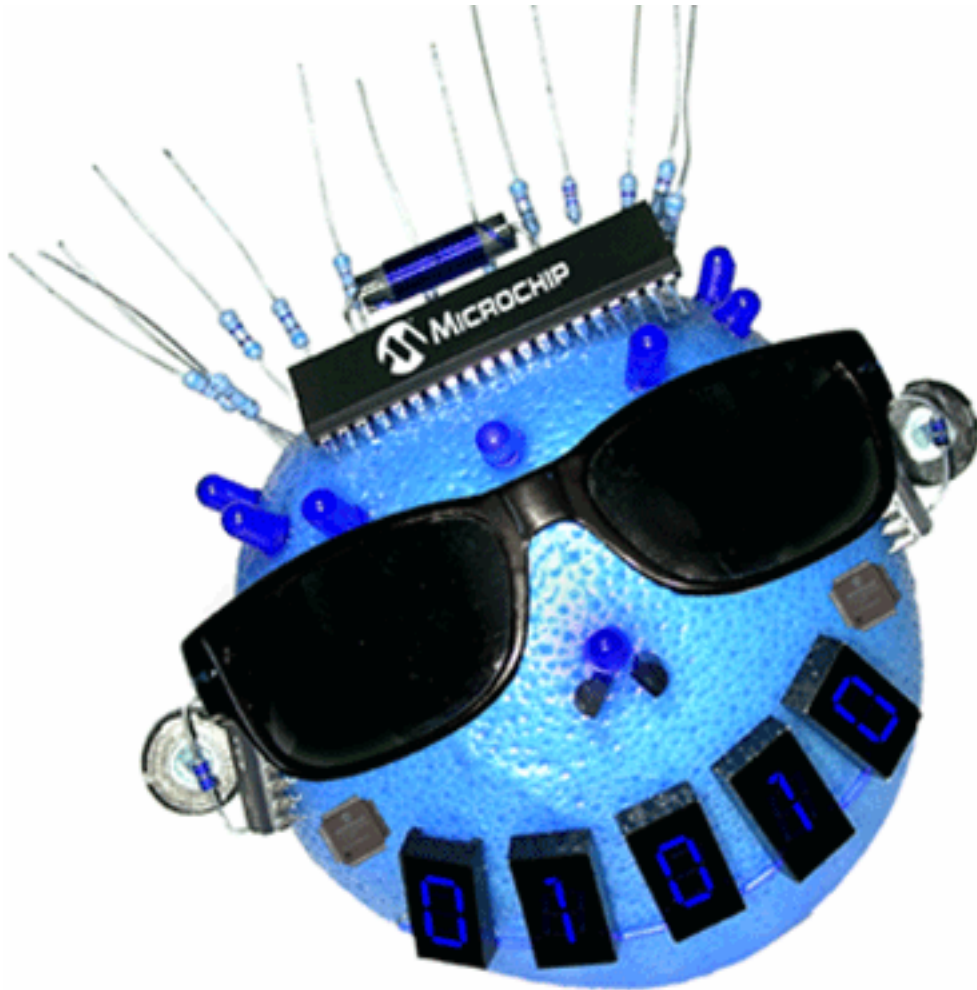




# PIC18Fxx2



V1.0

Christian Dupaty  
Lycée Fourcade  
13120 Gardanne

[c.dupaty@aix-mrs.iufm.fr](mailto:c.dupaty@aix-mrs.iufm.fr)



## Table des matières

1.	Famille et constitution.....	3
2.	Choix de l'horloge.....	5
3.	RESET.....	6
4.	Organisation mémoire.....	7
4.1.	PIC18F452, Mémoire programme.....	7
4.2.	Lecture / Ecriture FLASH.....	8
4.3.	PIC18F452, Mémoire RAM.....	8
4.4.	Adressage direct en RAM.....	9
4.5.	Adressage indirect en RAM.....	9
5.	Acces EEPROM / FLASH.....	10
6.	Registres internes.....	11
7.	Interruptions.....	13
8.	Jeu d'instructions.....	17
9.	Ports parallèles.....	22
10.	CAN 10bits.....	23
11.	Detection de faible tension (LVD).....	24
12.	TIMER0.....	25
13.	TIMER1.....	26
14.	TIMER2.....	27
15.	TIMER3.....	28
16.	Capture/Compare/PWM.....	29
16.1.	CAPTURE.....	29
16.2.	COMPARE.....	31
16.3.	PWM.....	33
17.	Chien de garde.....	34
18.	Communications séries asynchrones.....	35
19.	Communications séries synchrones : bus SPI.....	40
20.	Communications séries synchrones : bus I2C (IIC).....	43

Les images et tableaux proviennent du data sheet du PIC18F452 (39576b.pdf) disponible sur  
<http://www.microchip.com>

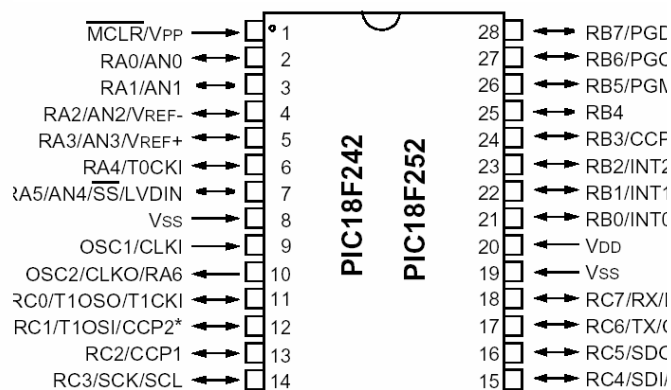
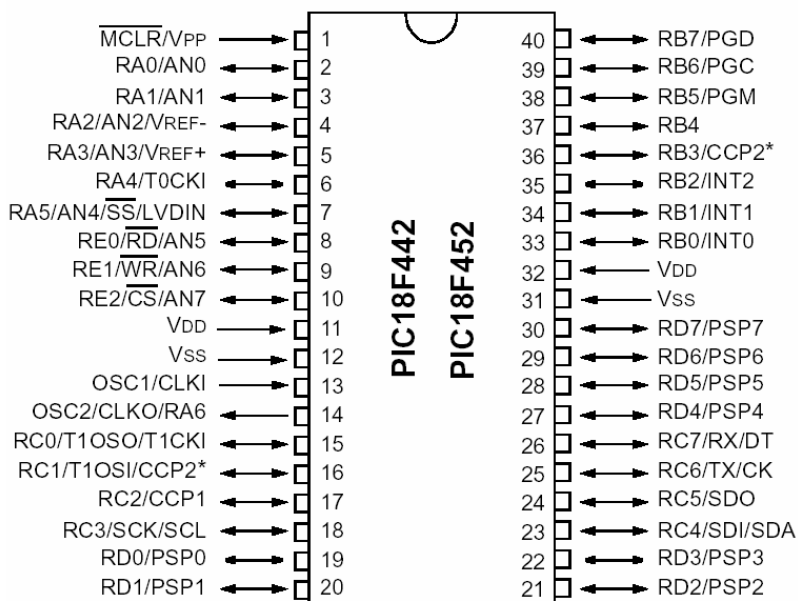
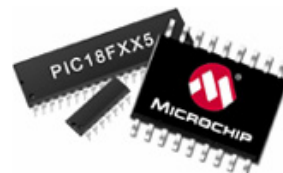


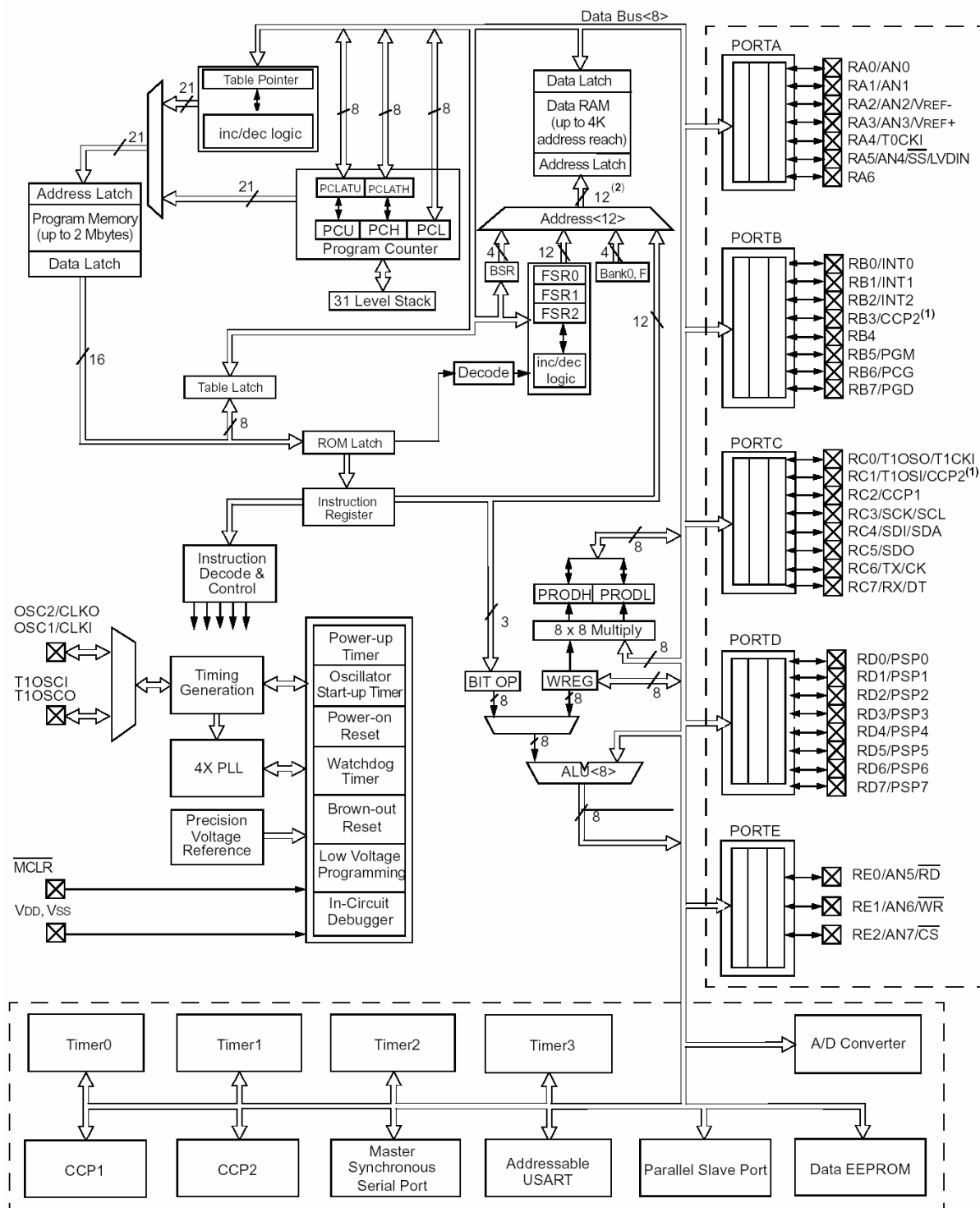
## 1. Famille et constitution

Caractéristiques	PIC18F252	PIC18F452
Fréquence Horloge MHz	DC-40 MHz	DC-40 MHz
Mémoire programme FLASH	32KO	32KO
Programme (Instructions)	16384	16384
Mémoire données	1536 Octets	1536 Octets
Mémoire EEPROM	256 Octets	256 Octets
Interruptions	17	18
Ports parallèles	A,B,C	A,B,C,D,E
Timers	4	4
Capture/Compare/PWM	2	2
Communications séries	SPI / I2C / USART	SPI / I2C / USART
Communications Parallèles	—	PSP
CAN 10-bit	5 entrées	8 entrées
RESETS	POR, BOR, RESET Instruction, Stack Full, Stack Underflow (PWRT, OST)	POR, BOR, RESET Instruction, Stack Full, Stack Underflow (PWRT, OST)
Détection de VDD faible programmable	oui	oui
Instructions	75	75
Boîtiers	28-pin DIP 28-pin SOIC	40-pin DIP 44-pin PLCC, 44-pin TQFP

- Architecture HARVARD
- Horloge max de 40 MHz par multiplication interne avec PLL (avec quartz 10MHz)
- PIC18F avec rom programme FLASH
- Programmation et debug sur cible (In-Circuit Serial Programming, In-Circuit debug, perte de RB6, RB7)
- Chaque instruction sur 2 octets
- Mémoire EEPROM à accès aléatoire (sauvegarde de données)
- TIMERS : Microchip appelle TIMER des compteurs
- Capture : permet la mesure de temps
- Compare : permet la production de signaux rectangulaires
- SPI : communications séries synchrones sans protocole logiciel
- I2C : standard Philips, communications séries synchrones avec protocole logiciel
- USART : communication séries asynchrones (RS232 et RS485)
- CAN : convertisseur analogique numérique à 10 entrées multiplexées

Le PIC18 possède une multiplication 8x8 matérielle, extrêmement rapide (100ns à 1µs) ce qui lui confère des possibilités de DSP particulièrement utiles pour le traitement numérique du signal





Le principe du flux de données en pipeline permet un temps d'exécution sur 1 cycle d'horloge

Exemple :

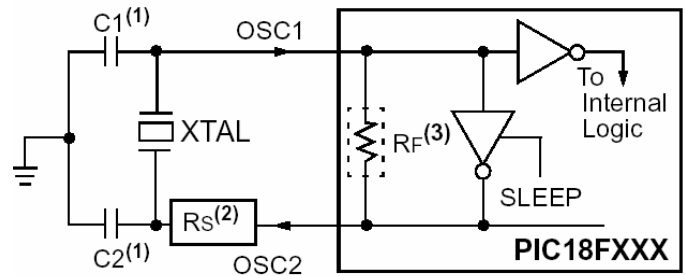
	T CY0	T CY1	T CY2	T CY3	T CY4	T CY5
1. MOVLW 55h	Fetch 1	Execute 1				
2. MOVWF PORTB		Fetch 2	Execute 2			
3. BRA SUB_1			Fetch 3	Execute 3		
4. BSF PORTA, BIT3 (force un NOP)				Fetch 4	vide (NOP)	
5. Instruction @ l'adresse SUB_1					Fetch SUB_1	Execute SUB_1



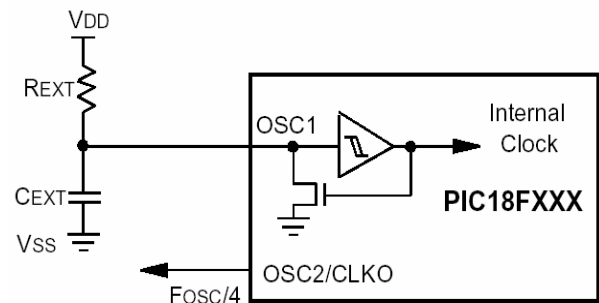
## 2. Choix de l'horloge

1. LP Quartz faible puissance
2. XT Quartz ou oscillateur externe
3. HS Quartz ou oscillateur externe rapide
4. HS + PLL (F x 4)

Types	Fréquences	C1	C2
LP	200kHz	15 pF	15 pF
XT	200 kHz	47-68pF	47-68 pF
	1.0MHz	15 pF	15 pF
	4.0MHz	15 pF	15 pF
HS	4.0 MHz	15 pF	15 pF
	8.0MHz	15-33 pF	15-33 pF
	20.0MHz	15-33pF	15-33 pF

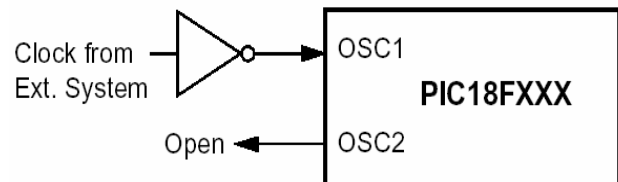


5. RC Résistance/capacité externe (Fosc/4 sortie sur OSC2)
6. RCIO Résistance/capacité externe (RA6 est un port //)



Recommended values:  $3 \text{ k}\Omega \leq R_{EXT} \leq 100 \text{ k}\Omega$   
 $C_{EXT} > 20 \text{ pF}$

7. EC Source d'horloge externe (Fosc/4 sortie sur OSC2)
8. ECIO Source d'horloge externe (RA6 est un port //)



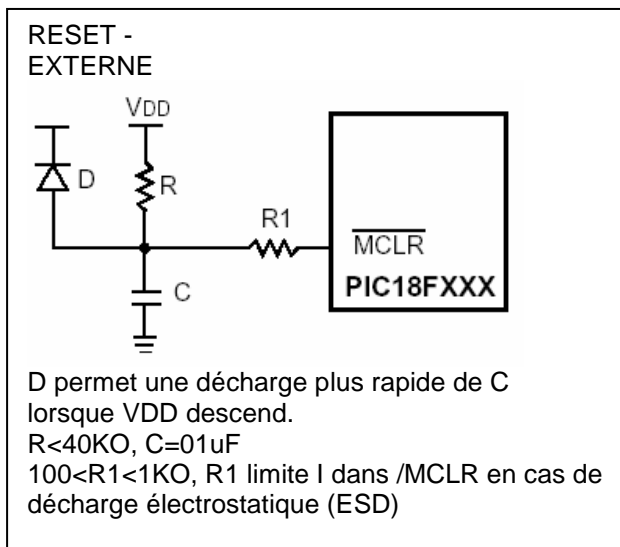
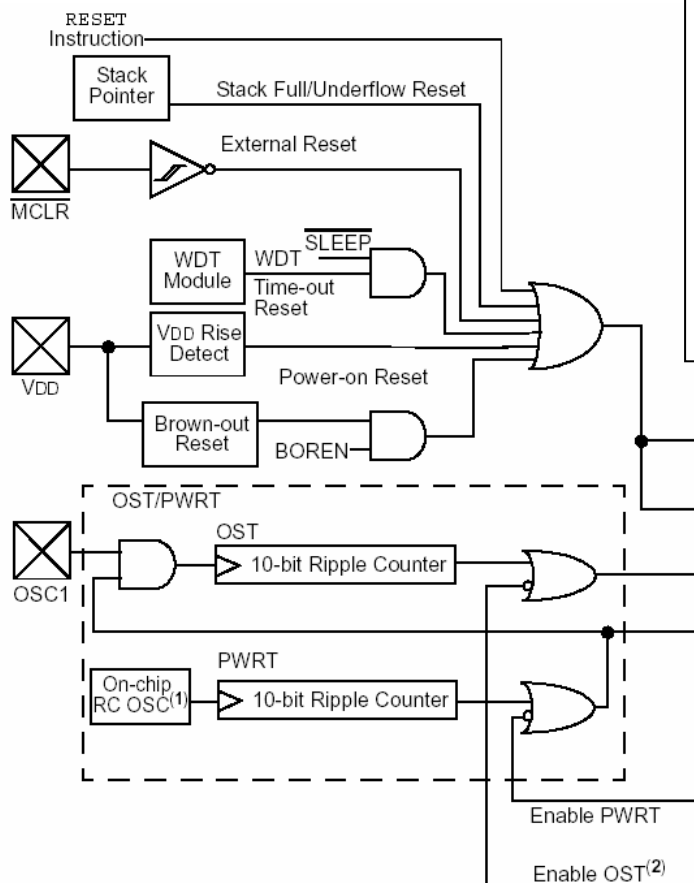
**Remarque** : le type d'oscillateur doit être déclaré dans la MPLAB dans le menu configure-> configuration bits

Configuration Bits			
Address	Value	Category	Setting
300001	FA	Oscillator	HS
300002	FF	Osc. Switch Enable	RC-OSC2 as RA6
		Power Up Timer	HS-PLL Enabled
		Brown Out Detect	EC-OSC2 as RA6
300003	FE	Brown Out Voltage	EC-OSC2 as Clock Out
		Watchdog Timer	RC
300005	FF	Watchdog Postscaler	HS
300006	7B	CCP2 Mux	XT
		Low Voltage Program	LP
		Background Debug	Enabled



### 3. RESET

Les différentes sources de l'interruption RESET



Après un RESET l'origine de celui-ci peut être déterminée en lisant le **REGISTRE RCON (0xFD0)**

7	6	5	4	3	2	1	0
IPEN	-	-	/RI	/TO	/PD	/POR	/BOR

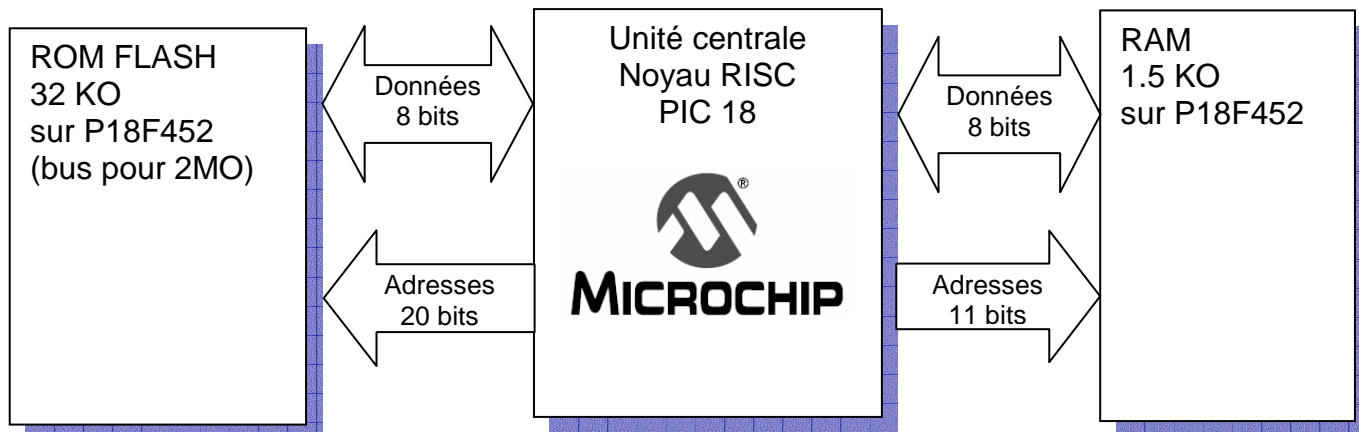
Condition	Program Counter	RCON Register	RI	TO	PD	POR	BOR	STKFUL	STKUNF
Power-on Reset	0000h	0--1 1100	1	1	1	0	0	u	u
MCLR Reset during normal operation	0000h	0--u uuuu	u	u	u	u	u	u	u
Software Reset during normal operation	0000h	0--0 uuuu	0	u	u	u	u	u	u
Stack Full Reset during normal operation	0000h	0--u uu11	u	u	u	u	u	u	1
Stack Underflow Reset during normal operation	0000h	0--u uu11	u	u	u	u	u	1	u
MCLR Reset during SLEEP	0000h	0--u 10uu	u	1	0	u	u	u	u
WDT Reset	0000h	0--u 01uu	1	0	1	u	u	u	u
WDT Wake-up	PC + 2	u--u 00uu	u	0	0	u	u	u	u
Brown-out Reset	0000h	0--1 11u0	1	1	1	1	0	u	u
Interrupt wake-up from SLEEP	PC + 2	u--u 00uu	u	1	0	u	u	u	u

IPEN = 1 valide les niveaux de priorités des interruptions



## 4. Organisation mémoire

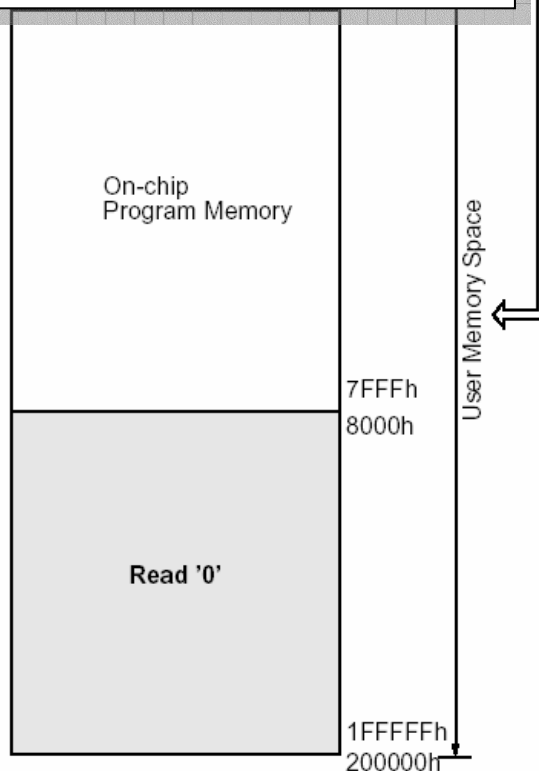
Les PIC18 sont à architecture HAVARD. Les espaces mémoires programmes et données (appelés registres, les registres des périphériques sont appelés registres spéciaux) sont distincts. Ceci implique la création d'instructions et de processus différents pour l'accès données en ROM et en RAM.



### 4.1. PIC18F452, Mémoire programme

CALL  
RET

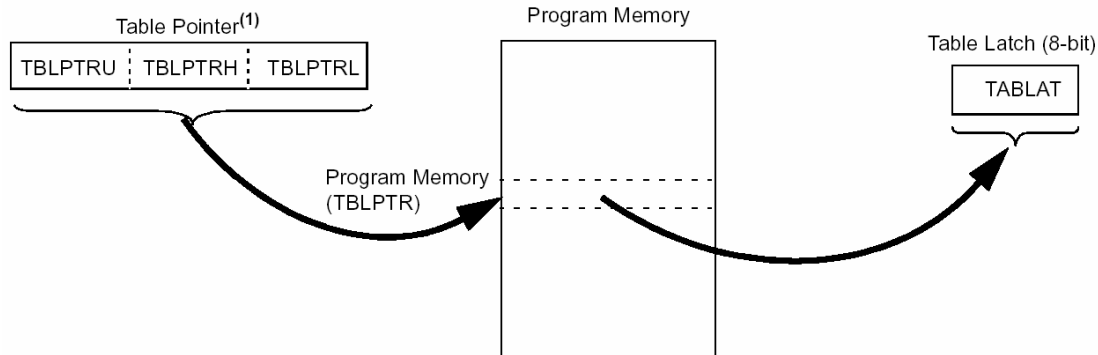
Il existe trois adresses d'interruption : RESET, HPI et LPI. Microchip utilise abusément le mot vecteur pour désigner ces adresses, en effet ces adresses sont celles des sous programmes à exécuter et non les vecteurs sur ces adresses. La taille de la pile n'est pas modifiable, elle contient 31 niveaux. L'espace mémoire va de 0x0000 à 0x200000 (soit 2MO). Sur les PIC18Fx52 seuls 32 KO sont implantés.





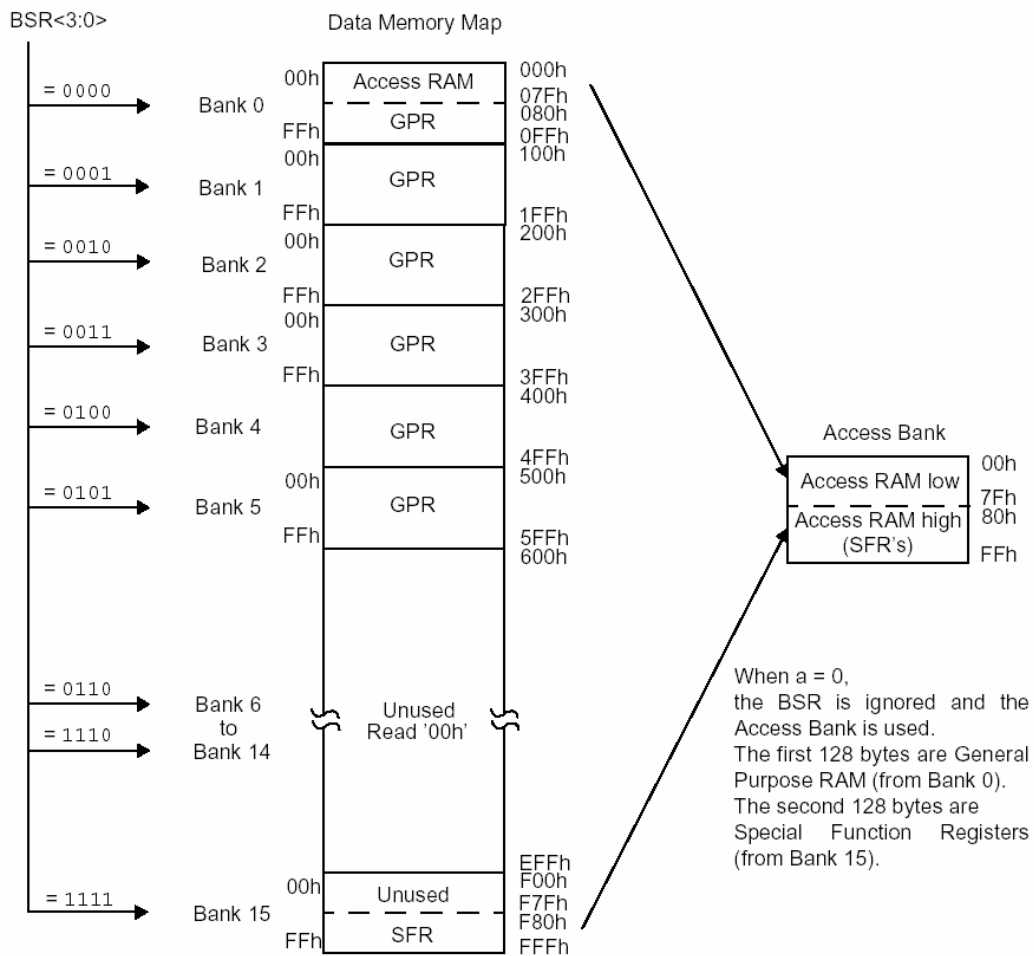
## 4.2. Lecture / Ecriture FLASH

Instruction: TBLRD\*



Attention on ne peut écrire dans la mémoire FLASH que des blocs de 64 octets, pour cela il faut écrire dans une zone de RAM appelée « Holding Register », puis exécuter une procédure d'écriture en ROM FLASH relativement complexe. **Il est recommandé de placer les données à conserver de manière permanente en EEPROM**

## 4.3. PIC18F452, Mémoire RAM



When a = 1,  
the BSR is used to specify the  
RAM location that the instruc-  
tion uses.

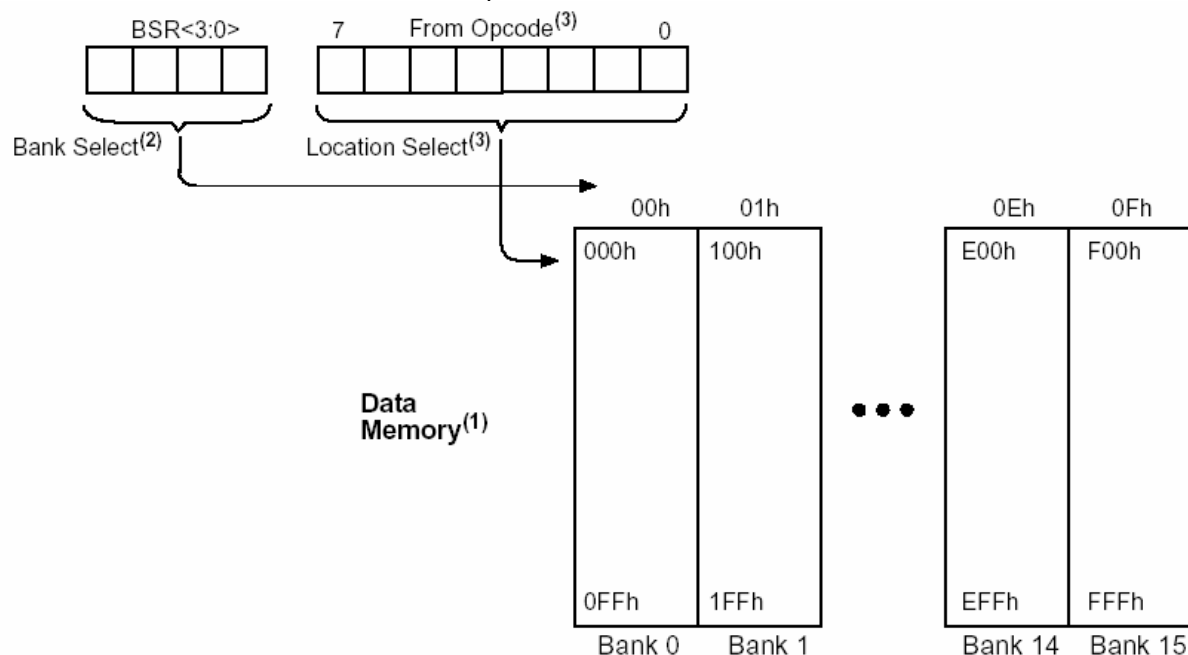
When a = 0,  
the BSR is ignored and the  
Access Bank is used.  
The first 128 bytes are General  
Purpose RAM (from Bank 0).  
The second 128 bytes are  
Special Function Registers  
(from Bank 15).





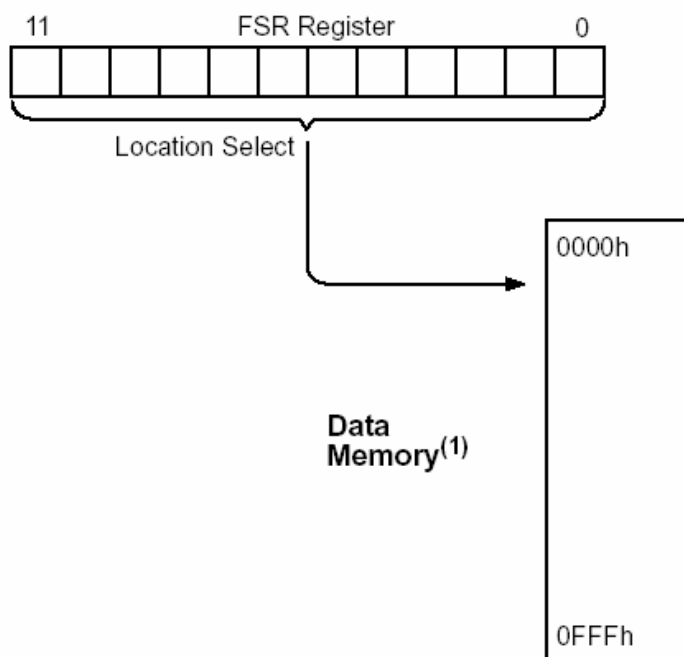
## 4.4. Adressage direct en RAM

BSR contient le numéro d'une des 8 banques de données



## 4.5. Adressage indirect en RAM

Il existe 3 registres d'indirection (ou pointeurs) FSR (FSR0, FSR1, FSR2)



Exemple : Mise à 0 de la BANK 1

```

LFSS FSR0 , 0x100 ;FSR0 pointe sur BANK
NEXT CLR F POSTINC0 ; Efface le contenu
; de l'adresse pointée
; par FSR0 puis
; incrémente FSR0
BTFSS FSR0H, 1 ; Test si FSR0=0x200
GOTO NEXT ; Non, efface le
; suivant
... CONTINUE ;
  
```



## 5. Acces EEPROM / FLASH

256 octets d'EEPROM (0x00 à 0xFF)

La lecture s'effectuent en mode indexé (l'adresse doit être olacée dans le pointeur EEADR).

La mise à 1 du bit RD de EECON1 provoque la lecture de l'EEPROM, le résultat se trouve dans EEDATA.

L'écriture s'effectuent en mode indexé, l'adresse doit être olacée dans le pointeur EEADR, la donnée à écrire dans le registre EEDATA.. La mise à du bit WREN de EECON1 autorise les écritures.

Un séquence d'écriture dans EECON2 (0x55 puis 0xAA) est nécessaire. La mise à 1 du bit WR de EECON1 provoque l'écrite de la donnée à l'adresse pointé par EEADR.

En fin d'écriture WR repasse à 0.

Le drapeau EEIF indique également la fin de l'écriture (possibilité de gestion par IT)

Exemple : Lecture de l'EEPROM

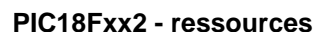
```
MOVLW DATA_EE_ADDR ; Adresse à lire dans W
MOVWF EEADR          ; W dans pointeur adresse
BCF EECON1,EEPGD ;Sel accès EEPROM (pas FLASH)
BSF EECON1,RD        ; lecture EEPROM, dans EEDATA
MOVF EEDATA,W         ; W = EEDATA
```

Exemple : Ecriture dans l'EEPROM

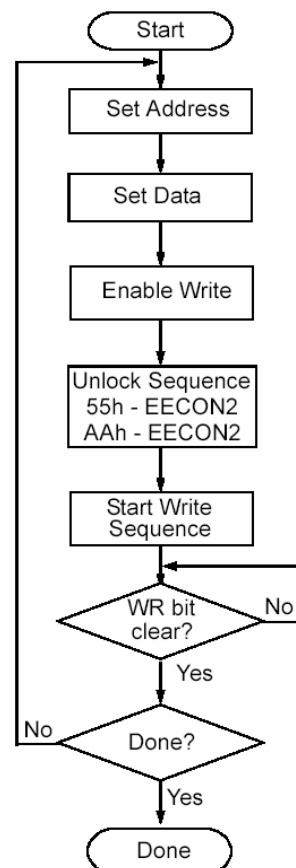
```
MOVLW DATA_EE_ADDR ;
MOVWF EEADR          ; Data Memory Address to write
MOVLW DATA_EE_DATA ;
MOVWF EEDATA         ; Data Memory Value to write
BCF EECON1,EEPGD ; Point to DATA memory
BSF EECON1,WREN ; Enable writes
BCF INTCON,GIE ; Disable Interrupts
MOVLW 55h ;
MOVWF EECON2 ; Write 55h
MOVLW AAh ;
MOVWF EECON2 ; Write AAh
BSF EECON1,WR ; Set WR bit to begin write
BSF INTCON,GIE ; Enable Interrupts
SLEEP ; Wait for interrupt
BCF EECON1,WREN ; Disable writes
```

### Registre EECON1 (0xFA6)

7	6	5	4	3	2	1	0
EEPGD	EEFS	—	FREE	WRERR	WREN	WR	RD



0 = ne pas commencer un cycle de lecture



## 6. Registres internes

ADR	REGISTRE	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR	page	
FFFh	<b>TOSU</b>	—	—	—	Top-of-Stack upper Byte (TOS<20:16>)					---0 0000	37	
FFEh	<b>TOSH</b>	Top-of-Stack High Byte (TOS<15:8>)									0000 0000	37
FFDh	<b>TOSL</b>	Top-of-Stack Low Byte (TOS<7:0>)									0000 0000	37
FFCh	<b>STKPTR</b>	STKFUL STKUNF		—	Return Stack Pointer					00-0 0000	38	
FFBh	<b>PCLATU</b>	—	—	—	Holding Register for PC<20:16>						0 0000	39
FFAh	<b>PCLATH</b>	Holding Register for PC<15:8>									0000 0000	39
FF9h	<b>PCL</b>	PC Low Byte (PC<7:0>)									0000 0000	39
FF8h	<b>TBLPTRU</b>	—	—	bit21	Program Memory Table Pointer Upper Byte (TBLPTR<20:16>)						--00 0000	58
FF7h	<b>TBLPTRH</b>	Program Memory Table Pointer High Byte (TBLPTR<15:8>)									0000 0000	58
FF6h	<b>TBLPTRL</b>	Program Memory Table Pointer Low Byte (TBLPTR<7:0>)									0000 0000	58
FF5h	<b>TABLAT</b>	Program Memory Table Latch								0000 0000	58	
FF4h	<b>PRODH</b>	Product Register High Byte									xxxx xxxx	69
FF3h	<b>PRODL</b>	Product Register Low Byte									xxxx xxxx	69
FF2h	<b>INTCON</b>	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	0000 000x	73	
FF1h	<b>INTCON2</b>	RBPUP	INTEDG0	INTEDG1	INTEDG2	—	TMR0IP	—	RBIP	1111 -1-1	74	
FF0h	<b>INTCON3</b>	INT2IP	INT1IP	—	INT2IE	INT1IE	—	INT2IF	INT1IF	11-0 0-00	75	
FEFh	<b>INDF0</b>	Uses contents of FSR0 to address data memory - value of FSR0 not changed									n/a	50
FEeh	<b>POSTINC0</b>	Uses contents of FSR0 to address data memory - value of FSR0 post-incremented									n/a	50
FEDh	<b>POSTDEC0</b>	Uses contents of FSR0 to address data memory - value of FSR0 post-decremented									n/a	50
FECh	<b>PREINC0</b>	Uses contents of FSR0 to address data memory - value of FSR0 pre-incremented									n/a	50
FEbh	<b>PLUSW0</b>	value of FSR0 offset by value in WREG									n/a	50
FEAh	<b>FSR0H</b>	—	—	—	—	Indirect Data Memory Address Pointer 0 High Byte					xxxx	50
FE9h	<b>FSR0L</b>	Indirect Data Memory Address Pointer 0 Low Byte									xxxx xxxx	50
FE8h	<b>WREG</b>	Working Register									xxxx xxxx	n/a
FE7h	<b>INDF1</b>	Uses contents of FSR1 to address data memory - value of FSR1 not changed									n/a	50
FE6h	<b>POSTINC1</b>	Uses contents of FSR1 to address data memory - value of FSR1 post-incremented									n/a	50
FE5h	<b>POSTDEC1</b>	Uses contents of FSR1 to address data memory - value of FSR1 post-decremented									n/a	50



FE4h	PREINC1	Uses contents of FSR1 to address data memory - value of FSR1 pre-incremented								n/a	50
FE3h	PLUSW1	value of FSR1 offset by value in WREG									
FE2h	FSR1H	—	—	—	—	Indirect Data Memory Address Pointer 1 High Byte			xxxx	50	
FE1h	FSR1L	Indirect Data Memory Address Pointer 1 Low Byte								xxxx xxxx	50
FE0h	BSR	—	—	—	—	Bank Select Register			0000	49	
FDFh	INDF2	Uses contents of FSR2 to address data memory - value of FSR2 not changed								n/a	50
FDEh	POSTINC2	Uses contents of FSR2 to address data memory - value of FSR2 post-incremented								n/a	50
FDDh	POSTDEC2	Uses contents of FSR2 to address data memory - value of FSR2 post-decremented								n/a	50
FDCh	PREINC2	Uses contents of FSR2 to address data memory - value of FSR2 pre-incremented								n/a	50
FDBh	PLUSW2	value of FSR2 offset by value in WREG								n/a	50
FDAh	FSR2H	—	—	—	—	Indirect Data Memory Address Pointer 2 High Byte			xxxx	50	
FD9h	FSR2L	Indirect Data Memory Address Pointer 2 Low Byte								xxxx xxxx	50
FD8h	STATUS	—	—	—	N	OV	Z	DC	C	x xxxx	52
FD7h	TMR0H	Timer0 Register High Byte								0000 0000	103
FD6h	TMR0L	Timer0 Register Low Byte								xxxx xxxx	103
FD5h	T0CON	TMR0ON	T08BIT	T0CS	T0SE	PSA	T0PS2	T0PS1	T0PS0	1111 1111	101
FD4h											
FD3h	OSCCON	—	—	—	—	—	—	—	SCS	0	21
FD2h	LVDFCON	—	—	IRVST	LVDFEN	LVDL3	LVDL2	LVDL1	LVDL0	--00 0101	189
FD1h	WDTCON	—	—	—	—	—	—	—	SWDTE	0	201
FD0h	RCON	IPEN	—	—	RI	TO	PD	POR	BOR	0--1 11qq	53, 28,
FCFh	TMR1H	Timer1 Register High Byte								xxxx xxxx	105
FCEh	TMR1L	Timer1 Register Low Byte								xxxx xxxx	105
FCDh	T1CON	RD16	—	T1CKPS1	T1CKPS0	T1OSCEN	T1SYNC	TMR1CS	TMR1ON	0-00 0000	105
FCCh	TMR2	Timer2 Register								0000 0000	109
FCBh	PR2	Timer2 Period Register								1111 1111	110
FCAh	T2CON	—	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0	-000 0000	109
FC9h	SSPBUF	SSP Receive Buffer/Transmit Register								xxxx xxxx	123
FC8h	SSPADD	SSP Address Register in I <sup>2</sup> C			C Slave Mode. SSP Baud Rate Reload Register in I <sup>2</sup> C Master Mode.					0000 0000	132
FC7h	SSPSTAT	SMP	CKE	D/A	P	S	R/W	UA	BF	0000 0000	124
FC6h	SSPCON1	WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0	0000 0000	125
FC5h	SSPCON2	GCEN	ACKSTAT	ACKDT	ACKEN	RCEN	PEN	RSEN	SEN	0000 0000	135
FC4h	ADRESH	A/D Result Register High Byte								xxxx xxxx	185
FC3h	ADRESL	A/D Result Register Low Byte								xxxx xxxx	185
FC2h	ADCON0	ADCS1	ADCS0	CHS2	CHS1	CHS0	GO/DONE	—	ADON	0000 00-0	179
FC1h	ADCON1	ADFM	ADCS2	—	—	PCFG3	PCFG2	PCFG1	PCFG0	00-- 0000	180
FC0h											
FBFh	CCPR1H	Capture/Compare/PWM Register1 High Byte								xxxx xxxx	119
FBEh	CCPR1L	Capture/Compare/PWM Register1 Low Byte								xxxx xxxx	119,
FBDh	CCP1CON	—	—	DC1B1	DC1B0	CCP1M3	CCP1M2	CCP1M1	CCP1M0	--00 0000	115
FBCh	CCPR2H	Capture/Compare/PWM Register2 High Byte								xxxx xxxx	119, 121
FBHh	CCPR2L	Capture/Compare/PWM Register2 Low Byte								xxxx xxxx	119, 121
FBAh	CCP2CON	—	—	DC2B1	DC2B0	CCP2M3	CCP2M2	CCP2M1	CCP2M0	--00 0000	115
FB4h – FB9h											
FB3h	TMR3H	Timer3 Register High Byte								xxxx xxxx	111
FB2h	TMR3L	Timer3 Register Low Byte								xxxx xxxx	111
FB1h	T3CON	RD16	T3CCP2	T3CKPS1	T3CKPS0	T3CCP1	T3SYNC	TMR3CS	TMR3ON	0000 0000	111
FB0h											
FAFh	SPBRG	USART1 Baud Rate Generator								0000 0000	166,
FAEh	RCREG	USART1 Receive Register								0000 0000	172
FADh	TXREG	USART1 Transmit Register								0000 0000	170,
FACH	TXSTA	CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D	0000 -010	164
FABh	RCSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	0000 000x	165
FAAh											
FA9h	EEADR	Data EEPROM Address Register								0000 0000	65, 68



FA8h	<b>EEDATA</b>	Data EEPROM Data Register								0000 0000	68
FA7h	<b>EECON2</b>	Data EEPROM Control Register 2 (not a physical register)									65, 68
FA6h	<b>EECON1</b>	EEPGD	CFGFS	—	FREE	WRERR	WREN	WR	RD	xx-0 x000	66
<b>FA3h – FA5h</b>											
FA2h	<b>IPR2</b>	—	—	—	EEIP	BCLIP	LVDIP	TMR3IP	CCP2IP	1 1111	81
FA1h	<b>PIR2</b>	—	—	—	EEIF	BCLIF	LVDIF	TMR3IF	CCP2IF	0 0000	77
FA0h	<b>PIE2</b>	—	—	—	EEIE	BCLIE	LVDIE	TMR3IE	CCP2IE	0 0000	79
F9Fh	<b>IPR1</b>	PSPIP	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP	1111 1111	80
F9Eh	<b>PIR1</b>	PSPIF	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	76
F9Dh	<b>PIE1</b>	PSPIE	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	78
<b>F97h – F9Ch</b>											
F96h	<b>TRISE</b>	IBF	OBF	IBOV	PSPMODE	—	Data Direction bits for PORTE			0000 -111	96
F95h	<b>TRISD</b>	Data Direction Control Register for PORTD								1111 1111	94
F94h	<b>TRISC</b>	Data Direction Control Register for PORTC								1111 1111	91
F93h	<b>TRISB</b>	Data Direction Control Register for PORTB								1111 1111	88
F92h	<b>TRISA</b>	—	TRISA6	Data Direction Control Register for PORTA						-111 1111	85
<b>F8Eh – F91h</b>											
F8Dh	<b>LATE</b>	—	—	—	—	—	Read PORTE Data Latch, Write PORTE Data Latch			-xxx	97
F8Ch	<b>LATD</b>	Read PORTD Data Latch, Write PORTD Data Latch								xxxx xxxx	93
F8Bh	<b>LATC</b>	Read PORTC Data Latch, Write PORTC Data Latch								xxxx xxxx	91
F8Ah	<b>LATB</b>	Read PORTB Data Latch, Write PORTB Data Latch								xxxx xxxx	88
F89h	<b>LATA</b>	—	LATA6	Read PORTA Data Latch, Write PORTA Data Latch						-xxx xxxx	85
<b>F85h – F88h</b>											
F84h	<b>PORTE</b>	Read PORTE pins, Write PORTE Data Latch								-000	97
F83h	<b>PORTD</b>	Read PORTD pins, Write PORTD Data Latch								xxxx xxxx	93
F82h	<b>PORTC</b>	Read PORTC pins, Write PORTC Data Latch								xxxx xxxx	91
F81h	<b>PORTB</b>	Read PORTB pins, Write PORTB Data Latch								xxxx xxxx	88
F80h	<b>PORTA</b>	—	RA6	Read PORTA pins, Write PORTA Data Latch						-x0x 0000	85

## 7. Interruptions

IPEN : interrupt priority enable. Cette fonction peut être désactivée pour avoir une compatibilité logicielle avec l'unité centrale PIC16. Si IPEN est à 1, chaque source d'interruption peut être configurée comme prioritaire ou non (entre autres : registres IPR1 et IPR2). Si elle est prioritaire, une autre source d'interruption sera prise en compte seulement à la fin de l'interruption prioritaire.

GEIH : global interrupt enable high (validation des interruptions prioritaires, adresse 0x0008)

GEIL : global interrupt enable low (validation des interruptions non prioritaires, adresse 0x0018)

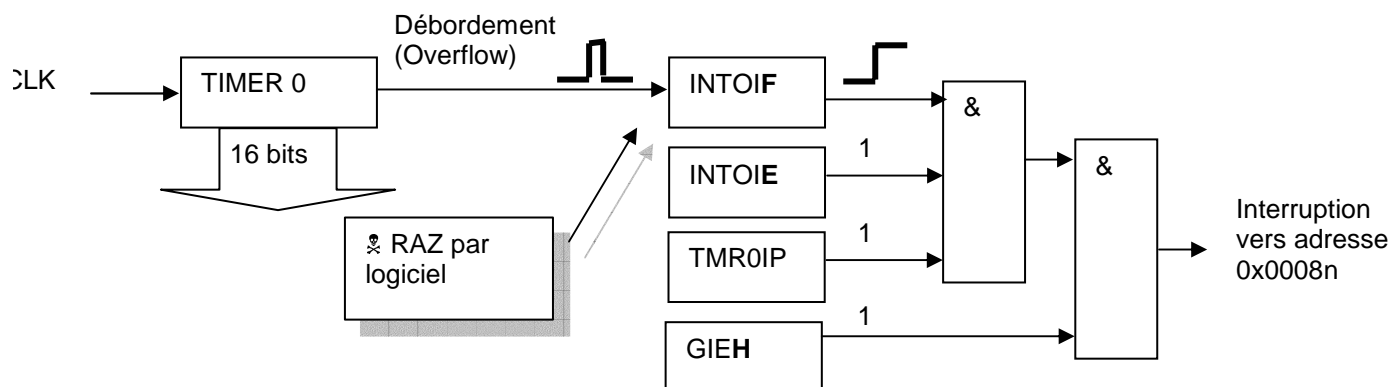
### Chaque source d'interruption possède

- un bit d'autorisation E (enable) ce bit doit être à 1 pour valider l'interruption
- un bit d'état F (flag) qui indique s'il y a eu ou non un événement

**Exemple :** Pour utiliser l'interruption générée lors du débordement du TIMER0 il faut

- mettre INTOIE à 1
- mettre GEIH ou GEIL à 1

L'interruption est déclenchée lors du débordement, il faut OBLIGATOIREMENT remettre INTOIF à 0 avant de ressortir de l'interruption, sinon elle reste active.





### INTCON (0xFF2): un 1 valide l'it concernée

GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF
----------	-----------	--------	--------	------	--------	--------	------

**GIE/GIEH:** Global Interrupt Enable bit

Valide toutes les interruptions non masquées si IPEN = 0

Valide toutes les interruptions prioritaires si IPEN = 1

**PEIE/GIEL:** Peripheral Interrupt Enable bit

Valide toutes les interruptions de périphériques si IPEN = 0

Valide toutes les interruptions non prioritaires si IPEN = 1

**TMR0IE:** TMR0 Overflow Interrupt Enable bit

Valide l'interruption de débordement de TMR0

**INT0IE:** INT0 External Interrupt Enable bit

Valide l'interruption externe INT0

**RBIE:** RB Port Change Interrupt Enable bit

Valide l'interruption sur un changement sur PB4-PB7

**TMR0IF:** TMR0 Overflow Interrupt Flag bit

Drapeau d'indication d'IT TMR0

**INT0IF:** INT0 External Interrupt Flag bit

Drapeau d'indication d'IT INT0

**RBIF:** RB Port Change Interrupt Flag bit

Drapeau d'indication d'IT RBI (au moins une ligne RB4-RB7 a changé)

### INTCON2 (0xFF1): Détection de front / Débordement TIMER0 / PORTB

RBPUP	INTEDG0	INTEDG1	INTEDG2	—	TMR0IP	—	RBIP
-------	---------	---------	---------	---	--------	---	------

**RBPUP:** PORTB Pull-up Enable bit

Pas de pull up sur PORB

**INTEDG0:** External Interrupt0 Edge Select bit

**INTEDG1:** External Interrupt1 Edge Select bit

**INTEDG2:** External Interrupt2 Edge Select bit

**TMR0IP:** TMR0 Overflow Interrupt Priority bit

**RBIP:** RB Port Change Interrupt Priority bit

1 pour front montant 0 pour front descendant

1 pour haute priorité.

### INTCON3 (0xFF0): Interruptions externes

INT2IP	INT1IP	—	INT2IE	INT1IE	—	INT2IF	INT1IF
--------	--------	---	--------	--------	---	--------	--------

**INT2IP:** INT2 External Interrupt Priority bit

**INT1IP:** INT1 External Interrupt Priority bit

**INT2IE:** INT2 External Interrupt Enable bit

**INT1IE:** INT1 External Interrupt Enable bit

**INT2IF:** INT2 External Interrupt Flag bit

**INT1IF:** INT1 External Interrupt Flag bit

1 pour haute priorité.

1 valide l'interruption

1 signal qu'une l'interruption a eu lieu

### PIR1 (0xF9E): ADC / USART / CCP1 / TIMER1 et 2

PSPIF	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF
-------	------	------	------	-------	--------	--------	--------

**PSPIF:** Parallel Slave Port Read/Write Interrupt Flag bit

Drapeau indiquant qu'une opération de lecture/écriture a eu lieu

**ADIF:** A/D Converter Interrupt Flag bit

1 indique une fin de conversion

**RCIF:** USART Receive Interrupt Flag bit

1 indique que RCREG est plein (une donnée a été reçue)

**TXIF:** USART Transmit Interrupt Flag bit

1 indique que TXREG, is vide

**SSPIF:** Master Synchronous Serial Port Interrupt Flag bit

1 indique que la transmission est terminée

**CCP1IF:** CCP1 Interrupt Flag bit

Mode capture : 1 indique qu'une capture a eu lieu dans TMR1

Mode comparaison : 1 indique qu'une égalité de comparaison a eu lieu dans TMR1

Mode PWM : inutilisé

**TMR2IF:** TMR2 to PR2 Match Interrupt Flag bit

1 indique que TMR2 a été égale à PR2

**TMR1IF:** TMR1 Overflow Interrupt Flag bit

1 indique un débordement sur TMR1

### PIR2 (0xFA1): Ecriture EEPROM / Bus collision / Faible VDD / TIMER3/ CCP2

—	—	—	EEIF	BCLIF	LVDIF	TMR3IF	CCP2IF
---	---	---	------	-------	-------	--------	--------

**EEIF:** Data EEPROM/FLASH Write Operation Interrupt Flag bit



1 indique une fin d'écriture

**BCLIF**: Bus Collision Interrupt Flag bit

1 indique qu'une collision s'est produite

**LVDIF**: Low Voltage Detect Interrupt Flag bit

1 indique une détection de faible tension

**TMR3IF**: TMR3 Overflow Interrupt Flag bit

1 indique un débordement sur TMR3

**CCP2IF**: CCPx Interrupt Flag bit

Idem PIR1 ci dessus

### PIE1 (0xF9D) : ADC / USART / I2C/ SPI / CCP1 / TIMER 1 et 2

PSPIE(1)	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE
----------	------	------	------	-------	--------	--------	--------

**PSPIE(1)**: Parallel Slave Port Read/Write Interrupt Enable bit

Autorise l'interruption PSP

**ADIE**: A/D Converter Interrupt Enable bit

Autorise l'interruption ADC (fin de conversion)

**RCIE**: USART Receive Interrupt Enable bit

Autorise l'interruption en réception sur l'USART

**TXIE**: USART Transmit Interrupt Enable bit

Autorise l'interruption en émission sur l'USART

**SSPIE**: Master Synchronous Serial Port Interrupt Enable bit

Autorise l'interruption SPI

**CCP1IE**: CCP1 Interrupt Enable bit

Autorise l'interruption CCP1 (pour capture ou compare)

**TMR2IE**: TMR2 to PR2 Match Interrupt Enable bit

Autorise l'interruption lors d'une égalité entre TMR2 et PR2

**TMR1IE**: TMR1 Overflow Interrupt Enable bit

Autorise l'interruption en cas de débordement de TMR1

### PIE2 (0xFA0): EEPROM / BUS collision / Faible VDD / TIMER 2 / CPP2

—	—	—	EEIE	BCLIE	LVDIE	TMR3IE	CCP2IE
---	---	---	------	-------	-------	--------	--------

**EEIE**: Data EEPROM/FLASH Write Operation Interrupt Enable bit

Autorise l'interruption de fin d'écriture

**BCLIE**: Bus Collision Interrupt Enable bit

Autorise l'interruption lors d'un collision

**LVDIE**: Low Voltage Detect Interrupt Enable bit

Autorise l'interruption lors de la détection d'un tension faible

**TMR3IE**: TMR3 Overflow Interrupt Enable bit

Autorise l'interruption lors du débordement de TMR3

**CCP2IE**: CCP2 Interrupt Enable bit

Autorise l'interruption CCP2 (pour capture ou compare)

### IPR1 (0xF9F) : Priorités

PSPIP	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP
-------	------	------	------	-------	--------	--------	--------

**PSPIP**: Parallel Slave Port Read/Write Interrupt Priority bit

**ADIP**: A/D Converter Interrupt Priority bit

**RCIP**: USART Receive Interrupt Priority bit

**TXIP**: USART Transmit Interrupt Priority bit

**SSPIP**: Master Synchronous Serial Port Interrupt Priority bit

**CCP1IP**: CCP1 Interrupt Priority bit

**TMR2IP**: TMR2 to PR2 Match Interrupt Priority bit

**TMR1IP**: TMR1 Overflow Interrupt Priority bit

1 haute priorité  
0 basse priorité

### IPR2 (0xFA2) : Priorités

—	—	—	EEIP	BCLIP	LVDIP	TMR3IP	CCP2IP
---	---	---	------	-------	-------	--------	--------

**EEIP**: Data EEPROM/FLASH Write Operation Interrupt Priority

**BCLIP**: Bus Collision Interrupt Priority bit

**LVDIP**: Low Voltage Detect Interrupt Priority bit

**TMR3IP**: TMR3 Overflow Interrupt Priority bit

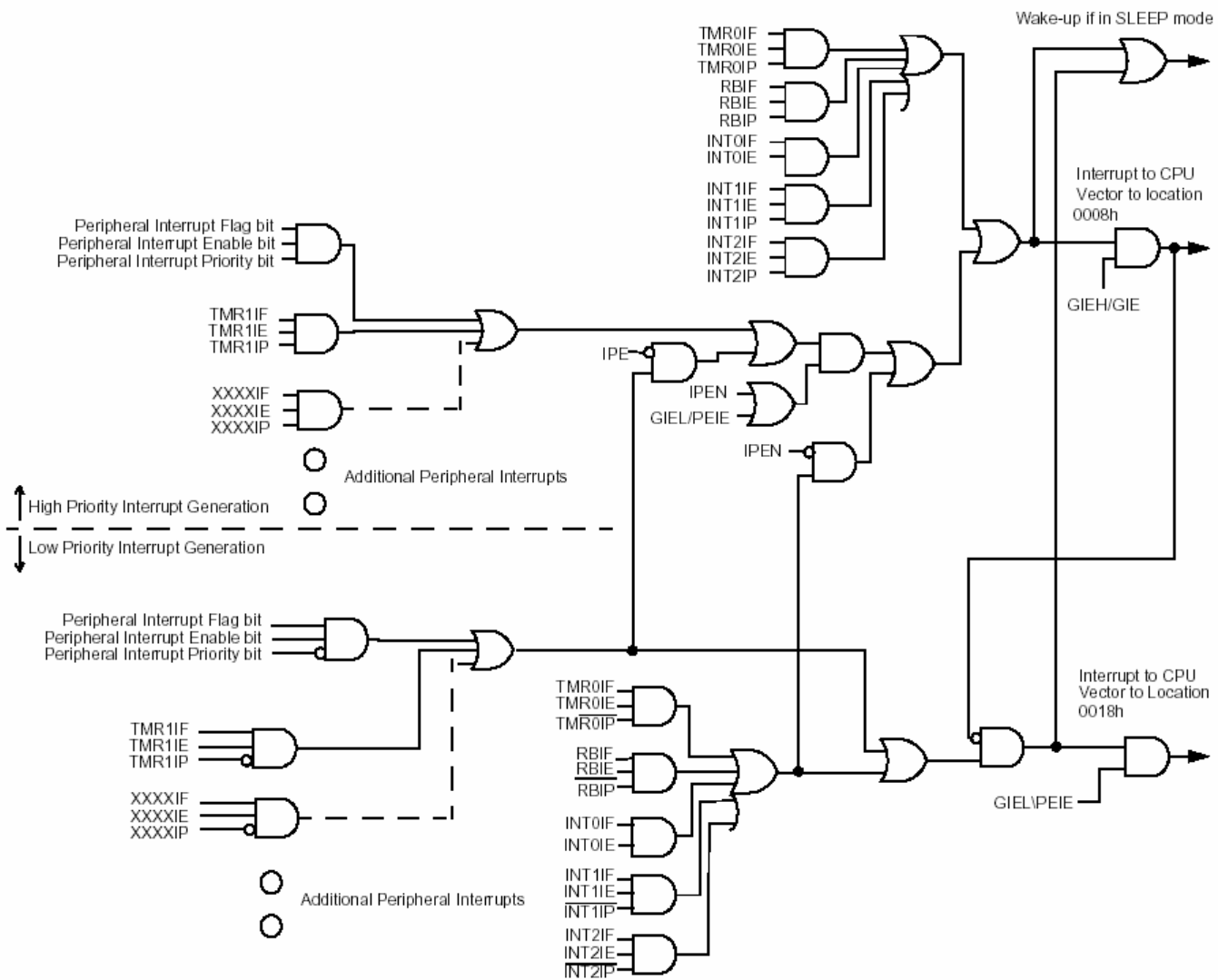
**CCP2IP**: CCP2 Interrupt Priority bit

1 haute priorité  
0 basse priorité

### RCON (0xFD0) : RESET Control

IPEN	—	—	RI	TO	PD	POR	BOR
------	---	---	----	----	----	-----	-----

IPEN active les priorités entre les interruptions







## 8. Jeu d'instructions

Toutes les opérations arithmétiques et logiques et les échanges de données entre registre passent par le registre de travail W.

**Exemple** : ADDWF, cette instruction ajoute le contenu W à un registre F.

ADDWF 50h,0,0 ; ajoute W au registre 50h, le résultat est dans W, ACCES RAM uniquement

ADDWF 50h,0,0 ; ajoute W au registre 50h, le résultat est dans 50h

ADDWF 50h,0,1 ; ajoute W au registre 50h, le résultat est dans W, Bank spécifiée par BSR

Le PIC18 possède une multiplication 8x8 matérielle, extrêmement rapide et particulièrement utile pour le traitement numérique du signal, cette fonction est appelée par l'instruction **MULWF f,a**.

### L'assembleur :

En plus du jeu d'instruction l'assembleur Microchip possède de nombreuses directives, une directive ne génère pas de code machine.

#### Exemples :

**List** : permet de définir le processeur cible ex : list p=18F452

**#include** : ajouter un fichier ex : #include <p18F452.inc> (<> indique le répertoire par défaut)

**#define** : comme en C c'est une définition, remplacement d'un texte par un autre avant assemblage

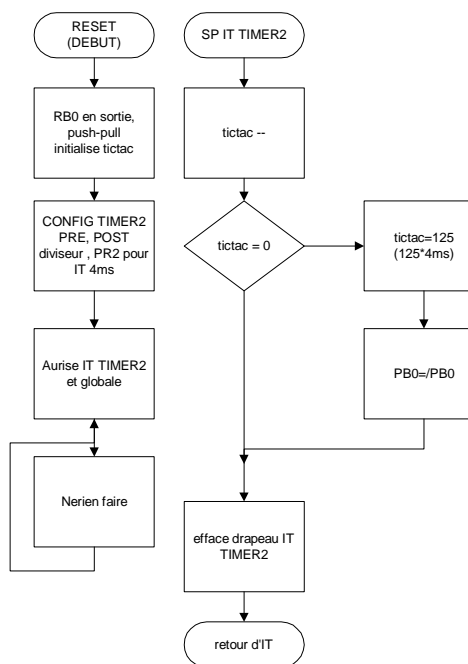
**equ** : équivalence entre un texte et une valeur numérique

**cblock / endc** : pour la réservation mémoire de données

**org** : origine pour spécifier l'adresse d'assemblage

**end** : fin du programme

### Exemple de programme assembleur



```
;*****
;Clignotement d'une LED sur PB0 (tempo par IT avec TIMER2
;(d'après Bigonoff) Q=4Mhz, t=1uS
;*****
list      p=18F452      ; Définition de processeur pour l'assembleur
#include <p18F452.inc>   ; fichier de définition pour PIC18
#define    LED TRISB,0 ; LED de sortie
tictac equ    d'124'
;VARIABLES
cblock 0x20          ; Début de la zone (0x20 à 0x6F)
compteur : 1          ; compteur de passages dans tmr2 (1 octet)
endc                ; Fin de la zone
;DEMARRAGE SUR RESET
org    0x000
goto   init
; SOUS PROGRAMME D'INTERRUPTION TMR2
; Un passage dans cette routine tous les 32*125*1µs = 4ms.
org    0x0008          ; adresse d'interruption prioritaire
decfsz compteur,f      ; décrémenter compteur d'IT
goto   attend          ; pas 0, ne rien faire
movlw  tictac          ; recharge le compteur d'IT
movwf  compteur
movlw  B'00000001'     ; inverser LED
xorwf  PORTB,f
attend bcf    PIR1,TMR2IF ; effacer flag interrupt tmr2
retfie                ; retour d'interruption
; INITIALISATIONS
init    bcf    LED          ; RB0 en sortie
        bsf    INTCON2,7    ; Pas de R pull up sur PORTB
        movlw  tictac       ; le tmr2 compte jusque (124+1)*32*1µs = 4ms
        movwf  PR2         ; dans PR2
        movlw  B'00101110' ; postdiviseur à 2,prédiviseur à 16,timer ON
        movwf  T2CON        ; dans registre de contrôle TIMER2
        movlw  tictac+1     ; pour 125 passages dans tmr2 = 125*4ms = 500
        movwf  compteur     ; dans compteur de passage interruption
        bsf    PIE1,TMR2IE ; autorise IT sur TIMER2
        bsf    INTCON,GIE  ; valider interruptions
        bsf    INTCON,GIEL
; PROGRAMME PRINCIPAL
debut   goto   debut        ; boucle sans fin (l'IT est asynchrone)
END      ; fin de programme
```

**Le registre STATUS (0xFD8)**

7	6	5	4	3	2	1	0
-	-	-	N	OV	Z	DC	C

Très important ce registre indique quel a été le type de résultat de l'instruction précédente. Il est utilisé entre autre par les instructions de branchement conditionnel.

**N** si négatif

**OV** s'il y a eu un débordement dans une opération en complément à 2

**Z** : si le résultat est nul

**DC** : demi retenue (le bit4 est passé à 1)

**C** : s'il y a eu une retenue (résultat supérieur à 0xFF)

**Le jeux d'instructions**

Champ	Description
a	RAM access bit a = 0: RAM location in Access RAM (BSR register is ignored) a = 1: RAM bank is specified by BSR register
bbb	Bit address within an 8-bit file register (0 to 7)
BSR	Bank Select Register. Used to select the current RAM bank.
d	Destination select bit; d = 0: store result in WREG, d = 1: store result in file register f.
dest	Destination either the WREG register or the specified register file location
f	8-bit Register file address (0x00 to 0xFF)
fs	12-bit Register file address (0x000 to 0xFFF). This is the source address.
fd	12-bit Register file address (0x000 to 0xFFF). This is the destination address.
k	Literal field, constant data or label (may be either an 8-bit, 12-bit or a 20-bit value)
label	Label name
mm * *+ *- +*	The mode of the TBLPTR register for the Table Read and Table Write instructions Only used with Table Read and Table Write instructions: No Change to register (such as TBLPTR with Table reads and writes) Post-Increment register (such as TBLPTR with Table reads and writes) Post-Decrement register (such as TBLPTR with Table reads and writes) Pre-Increment register (such as TBLPTR with Table reads and writes)
n	The relative address (2's complement number) for relative branch instructions, or the direct address for Call/ Branch and Return instructions
PRODH	Product of Multiply high byte
PRODL	Product of Multiply low byte
*	Fast Call / Return mode select bit. s = 0: do not update into/from shadow registers s = 1: certain registers loaded into/from shadow registers (Fast mode)
u	Unused or Unchanged
WREG	Working register (accumulator)
x	The assembler will generate code with x = 0. It is the recommended form of use for compatibility with all
Microchip software tools.	
TBLPTR	21-bit Table Pointer (points to a Program Memory location)
TABLAT	8-bit Table Latch
TOS	Top-of-Stack
PC	Program Counter
PCL	Program Counter Low Byte
PCH	Program Counter High Byte
PCLATH	Program Counter High Byte Latch
PCLATU	Program Counter Upper Byte Latch
GIE	Global Interrupt Enable bit
WDT	Watchdog Timer
TO	Time-out bit
PD	Power-down bit
C, DC, Z, OV, N	ALU status bits Carry, Digit Carry, Zero, Overflow, Negative
[]	Optional
()	Contents
□	Assigned to
< >	Register bit field
In the set of	
italics	User defined term (font is courier)



Mnemonic,	Op	Description	Cycles	16-Bit Instruction Word				Status Affected
				MSb			LSb	
<b>BYTE-ORIENTED FILE REGISTER OPERATIONS</b>								
ADDWF	f, d, a	Add WREG and f	1	0010	01da	ffff	ffff	C, DC, Z, OV, N
ADDWFC	f, d, a	Add WREG and Carry bit to f	1	0010	00da	ffff	ffff	C, DC, Z, OV, N
ANDWF	f, d, a	AND WREG with f	1	0001	01da	ffff	ffff	Z, N
CLRF	f, a	Clear f	1	0110	101a	ffff	ffff	Z
COMF	f, d, a	Complement f	1	0001	11da	ffff	ffff	Z, N
CPFSEQ	f, a	Compare f with WREG, skip =	1 (2 or 3)	0110	001a	ffff	ffff	None
CPFSGT	f, a	Compare f with WREG, skip >	1 (2 or 3)	0110	010a	ffff	ffff	None
CPFSLT	f, a	Compare f with WREG, skip <	1 (2 or 3)	0110	000a	ffff	ffff	None
DECF	f, d, a	Decrement f	1	0000	01da	ffff	ffff	C, DC, Z, OV, N
DECFSZ	f, d, a	Decrement f, Skip if 0	1 (2 or 3)	0010	11da	ffff	ffff	None
DCFSNZ	f, d, a	Decrement f, Skip if Not 0	1 (2 or 3)	0100	11da	ffff	ffff	None
INCF	f, d, a	Increment f	1	0010	10da	ffff	ffff	C, DC, Z, OV, N
INCFSZ	f, d, a	Increment f, Skip if 0	1 (2 or 3)	0011	11da	ffff	ffff	None
INFSNZ	f, d, a	Increment f, Skip if Not 0	1 (2 or 3)	0100	10da	ffff	ffff	None
IORWF	f, d, a	Inclusive OR WREG with f	1	0001	00da	ffff	ffff	Z, N
MOVF	f, d, a	Move f	1	0101	00da	ffff	ffff	Z, N
MOVFF	f <sub>s</sub> , f <sub>d</sub>	Move f <sub>s</sub> (source) to 1st word f <sub>d</sub> (destination) 2nd word	2	1100	ffff	ffff	ffff	None
MOVWF	f, a	Move WREG to f	1	0110	111a	ffff	ffff	None
MULWF	f, a	Multiply WREG with f	1	0000	001a	ffff	ffff	None
NEGF	f, a	Negate f	1	0110	110a	ffff	ffff	C, DC, Z, OV, N
RLCF	f, d, a	Rotate Left f through Carry	1	0011	01da	ffff	ffff	C, Z, N
RLNCF	f, d, a	Rotate Left f (No Carry)	1	0100	01da	ffff	ffff	Z, N
RRCF	f, d, a	Rotate Right f through Carry	1	0011	00da	ffff	ffff	C, Z, N
RRNCF	f, d, a	Rotate Right f (No Carry)	1	0100	00da	ffff	ffff	Z, N
SETF	f, a	Set f	1	0110	100a	ffff	ffff	None
SUBFWB	f, d, a	Subtract f from WREG with borrow	1	0101	01da	ffff	ffff	C, DC, Z, OV, N
SUBWF	f, d, a	Subtract WREG from f	1	0101	11da	ffff	ffff	C, DC, Z, OV, N
SUBWFB	f, d, a	Subtract WREG from f with borrow	1	0101	10da	ffff	ffff	C, DC, Z, OV, N
SWAPF	f, d, a	Swap nibbles in f	1	0011	10da	ffff	ffff	None
TSTFSZ	f, a	Test f, skip if 0	1 (2 or 3)	0110	011a	ffff	ffff	None
XORWF	f, d, a	Exclusive OR WREG with f	1	0001	10da	ffff	ffff	Z, N
<b>BIT-ORIENTED FILE REGISTER OPERATIONS</b>								
BCF	f, b, a	Bit Clear f	1	1001	bbba	ffff	ffff	None
BSF	f, b, a	Bit Set f	1	1000	bbba	ffff	ffff	None
BTFSC	f, b, a	Bit Test f, Skip if Clear	1 (2 or 3)	1011	bbba	ffff	ffff	None
BTFSS	f, b, a	Bit Test f, Skip if Set	1 (2 or 3)	1010	bbba	ffff	ffff	None
BTG	f, d, a	Bit Toggle f	1	0111	bbba	ffff	ffff	None
<b>CONTROL OPERATIONS</b>								
BC	n	Branch if Carry	1 (2)	1110	0010	nnnn	nnnn	None
BN	n	Branch if Negative	1 (2)	1110	0110	nnnn	nnnn	None
BNC	n	Branch if Not Carry	1 (2)	1110	0011	nnnn	nnnn	None
BNN	n	Branch if Not Negative	1 (2)	1110	0111	nnnn	nnnn	None
BNOV	n	Branch if Not Overflow	1 (2)	1110	0101	nnnn	nnnn	None
BNZ	n	Branch if Not Zero	2	1110	0001	nnnn	nnnn	None
BOV	n	Branch if Overflow	1 (2)	1110	0100	nnnn	nnnn	None
BRA	n	Branch Unconditionally	1 (2)	1101	0nnn	nnnn	nnnn	None
BZ	n	Branch if Zero	1 (2)	1110	0000	nnnn	nnnn	None
CALL	n, s	Call subroutine 1st word 2nd word	2	1110	110s	kkkk	kkkk	None
CLRWDT	—	Clear Watchdog Timer	1	0000	0000	0000	0100	TO, PD
DAW	—	Decimal Adjust WREG	1	0000	0000	0000	0111	C
GOTO	n	Go to address 1st word 2nd word	2	1110	1111	kkkk	kkkk	None
NOP	—	No Operation	1	0000	0000	0000	0000	None
NOP	—	No Operation (Note 4)	1	1111	xxxx	xxxx	xxxx	None
POP	—	Pop top of return stack (TOS)	1	0000	0000	0000	0110	None
PUSH	—	Push top of return stack (TOS)	1	0000	0000	0000	0101	None



RCALL	n	Relative Call	2	1101	1nnn	hnnn	nnnn	None
RESET		Software device RESET	1	0000	0000	1111	1111	All
RETFIE	s	Return from interrupt enable	2	0000	0000	0001	000s	GIE/GIEH, PEIE/GIEL
RETLW	k	Return with literal in WREG	2	0000	1100	kkkk	kkkk	None
RETURN	s	Return from Subroutine	2	0000	0000	0001	001s	None
SLEEP	—	Go into standby mode	1	0000	0000	0000	0011	TO, PD
<b>LITERAL OPERATIONS</b>								
ADDLW	k	Add literal and WREG	1	0000	1111	kkkk	kkkk	C, DC, Z, OV, N
ANDLW	k	AND literal with WREG	1	0000	1011	kkkk	kkkk	Z, N
IORLW	k	Inclusive OR literal with WREG	1	0000	1001	kkkk	kkkk	Z, N
LFSR	f, k	Move literal (12-bit) 2nd word to FSRx 1st word	2	1110	1110	00ff	kkkk	None
MOVLB	k	Move literal to BSR<3:0>	1	0000	0001	0000	kkkk	None
MOVLW	k	Move literal to WREG	1	0000	1110	kkkk	kkkk	None
MULLW	k	Multiply literal with WREG	1	0000	1101	kkkk	kkkk	None
RETLW	k	Return with literal in WREG	2	0000	1100	kkkk	kkkk	None
SUBLW	k	Subtract WREG from literal	1	0000	1000	kkkk	kkkk	C, DC, Z, OV, N
XORLW	k	Exclusive OR literal with WREG	1	0000	1010	kkkk	kkkk	Z, N
<b>DATA MEMORY</b>				<b>PROGRAM MEMORY OPERATIONS</b>				
TBLRD*		Table Read	2	0000	0000	0000	1000	None
TBLRD*+		Table Read with post-increment		0000	0000	0000	1001	None
TBLRD*-		Table Read with post-decrement		0000	0000	0000	1010	None
TBLRD*+		Table Read with pre-increment		0000	0000	0000	1011	None
TBLWT*		Table Write	2 (5)	0000	0000	0000	1100	None
TBLWT*+		Table Write with post-increment		0000	0000	0000	1101	None
TBLWT*-		Table Write with post-decrement		0000	0000	0000	1110	None
TBLWT*+		Table Write with pre-increment		0000	0000	0000	1111	None



### 8\*8 non signée :

```
MOVWF ARG1, W ;
MULWF ARG2 ; ARG1 * ARG2 ->
; PRODH:PRODL
```

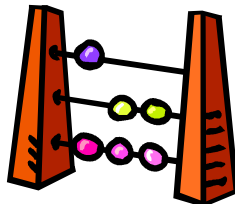
### 8\*8 signée

```
MOVWF ARG1, W ;
MULWF ARG2 ; ARG1 * ARG2 ->
; PRODH:PRODL
BTFSC ARG2, SB ; Test Sign Bit
SUBWF PRODH, F ; PRODH = PRODH
; - ARG1
MOVWF ARG2, W ;
BTFSC ARG1, SB ; Test Sign Bit
SUBWF PRODH, F ; PRODH = PRODH
; - ARG2
```

### 16 bits non signée

```
MOVWF ARG1L, W ;
MULWF ARG2L ; ARG1L * ARG2L ->
; PRODH:PRODL
MOVFF PRODH, RES1 ;
MOVFF PRODL, RES0 ;
MOVWF ARG1H, W ;
MULWF ARG2H ; ARG1H * ARG2H ->
; PRODH:PRODL
MOVFF PRODH, RES3 ;
MOVFF PRODL, RES2 ;
MOVWF ARG1L, W ;
MULWF ARG2H ; ARG1L * ARG2H ->
; PRODH:PRODL
MOVWF PRODL, W ;
ADDWF RES1, F ; Add cross
MOVWF PRODH, W ; products
ADDWFC RES2, F ;
CLRF WREG ;
ADDWFC RES3, F ;
MOVWF ARG1H, W ;
MULWF ARG2L ; ARG1H * ARG2L ->
; PRODH:PRODL
MOVWF PRODL, W ;
ADDWF RES1, F ; Add cross
MOVWF PRODH, W ; products
ADDWFC RES2, F ;
CLRF WREG ;
ADDWFC RES3, F ;
```

### Multiplications :



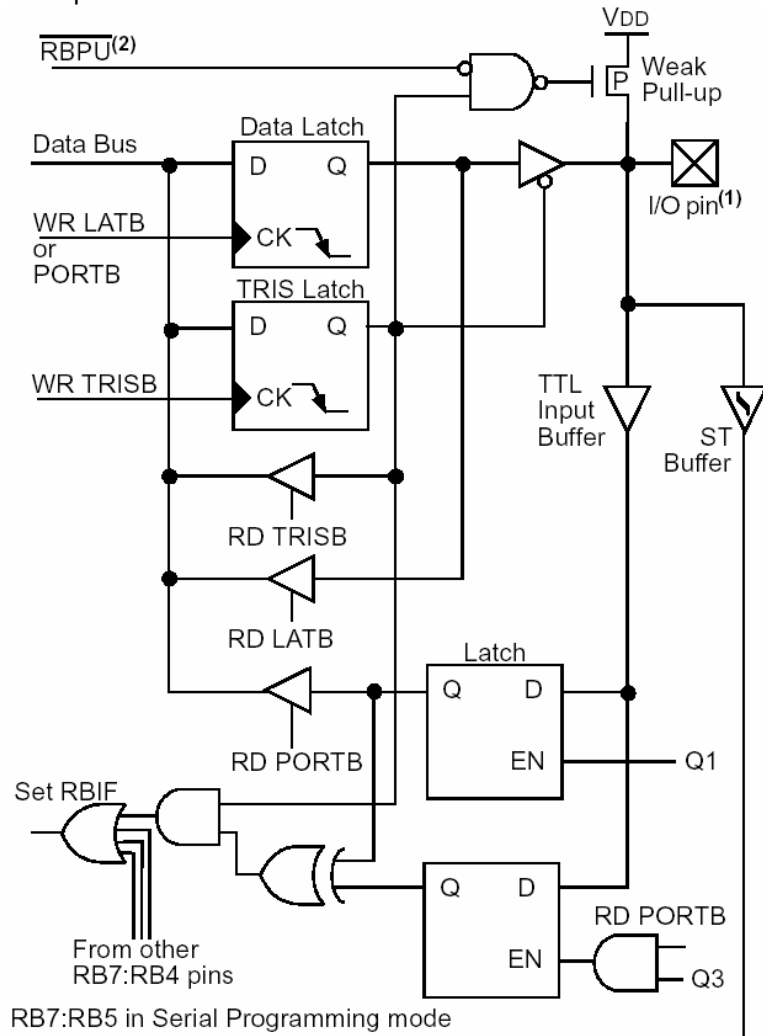
### 16 bits signée

```
MOVWF ARG1L, W ;
MULWF ARG2L ; ARG1L * ARG2L ->
; PRODH:PRODL
MOVFF PRODH, RES1 ;
MOVFF PRODL, RES0 ;
MOVWF ARG1H, W ;
MULWF ARG2H ; ARG1H * ARG2H ->
; PRODH:PRODL
MOVFF PRODH, RES3 ;
MOVFF PRODL, RES2 ;
MOVWF ARG1L, W ;
MULWF ARG2H ; ARG1L * ARG2H ->
; PRODH:PRODL
MOVWF PRODL, W ;
ADDWF RES1, F ; Add cross
MOVWF PRODH, W ; products
ADDWFC RES2, F ;
CLRF WREG ;
ADDWFC RES3, F ;
MOVWF ARG1H, W ;
MULWF ARG2L ; ARG1H * ARG2L ->
; PRODH:PRODL
MOVWF PRODL, W ;
ADDWF RES1, F ; Add cross
MOVWF PRODH, W ; products
ADDWFC RES2, F ;
CLRF WREG ;
ADDWFC RES3, F ;
BTFSS ARG2H, 7 ; ARG2H:ARG2L neg?
BRA SIGN ARG1 ; no, check ARG1
MOVWF ARG1L, W ;
SUBWF RES2 ;
MOVWF ARG1H, W ;
SUBWFB RES3
SIGN ARG1
BTFSS ARG1H, 7 ; ARG1H:ARG1L neg?
BRA CONT_CODE ; no, done
MOVWF ARG2L, W ;
SUBWF RES2 ;
MOVWF ARG2H, W ;
SUBWFB RES3
```



## 9. Ports parallèles

Exemple : PORTB



```
Example
CLRF PORTB      ; Initialize PORTB by
                  ; clearing output
                  ; data latches
CLRFB LATB      ; Alternate method
                  ; to clear output
                  ; data latches
MOVLW 0xCF      ; Value used to
                  ; initialize data
                  ; direction
MOVWF TRISB     ; Set RB<3:0> as inputs
                  ; RB<5:4> as outputs
                  ; RB<7:6> as inputs
```

```
#include <p18f452.h>
void main(void)
{
    char a=0,b=0x55 ;
    PORTB=0 ;
    TRISB=0b11110000 ;

    a=PORTB ;
    PORTB=b ;
    While(1) ;
}
```

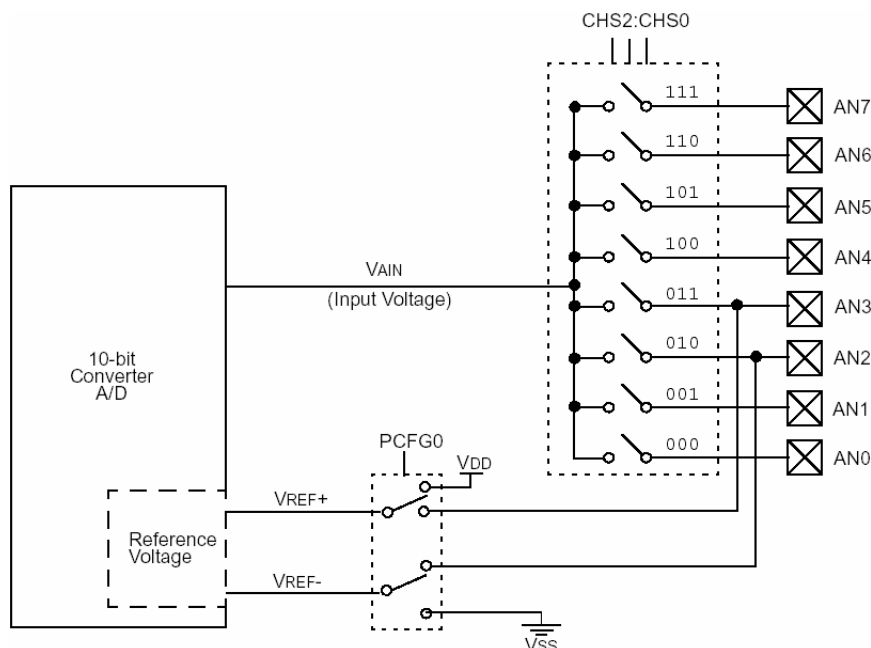
Chaque port a ses particularités, seules celle du PORTB sont détaillées ici. (voir data sheet)

### Particularités du PORTB

Les broches peuvent être configurées en Drain-Ouvert RBPU INTCON2<7> (INTCON2bits.RBPU)  
Un changement sur l'un des bits PB4 à PB7 peut déclencher une interruption (gestion d'un clavier par exemple) RBIF (INTCON<0>). (INTCONbits.RBIF), ce drapeau DOIT être effacé dans le sous programme d'interruption.



## 10. CAN 10bits



```
// Demo pour ADC
#include <pic18f452.h> //pour LCD
#define q 4.8828e-3 // quantum

void main(void)
{
    float res;
    // CAN on. CLOCK=FOSC/2. CANAL0 (RA)
    // seul AN0 est activé
    // VREF+=VDD VREF-=VSS
    ADCON0=1;
    ADCON1=0x8E;
    while(1){
        // déclenche SOC
        ADCON0bits.GO_DONE=1;
        // attend EOC
        while(ADCON0bits.GO_DONE)
            // calcule la tension
            res=(float)ADRES*q;
    }
}
```

### REGISTRE ADCON0

7	6	5	4	3	2	1	0
ADCS1	ADCS0	CHS2	CHS1	CHS0	GO/DONE	—	ADON

ADCON1 <ADCS2>	ADCON0 <ADCS1:ADCS0>	Horloge de conversion
0	00	Fosc/2
0	01	Fosc/8
0	10	Fosc/32
0	11	FRC (Provient de l'oscillateur RC interne)
1	00	Fosc/4
1	01	Fosc/16
1	10	Fosc/64
1	11	FRC (Provient de l'oscillateur RC interne)

CHS2:CHS0	Canal sélectionné
000	canal 0, (AN0)
001	canal 1, (AN1)
010	canal 2, (AN2)
011	canal 3, (AN3)
100	canal 4, (AN4)
101	canal 5, (AN5)
110	canal 6, (AN6)
111	canal 7, (AN7)

### REGISTRE ADCON1

7	6	5	4	3	2	1	0
ADFM	ADCS2	—	—	PCFG3	PCFG2	PCFG1	PCFG0

**ADFM:** Format du résultat (sur 16 bits)

1 = justification à droite, les 6 bits de poids fort d'ADRESH sont à 0.

0 = justification à gauche, les 6 bits de poids faible d'ADRESL sont à 0.

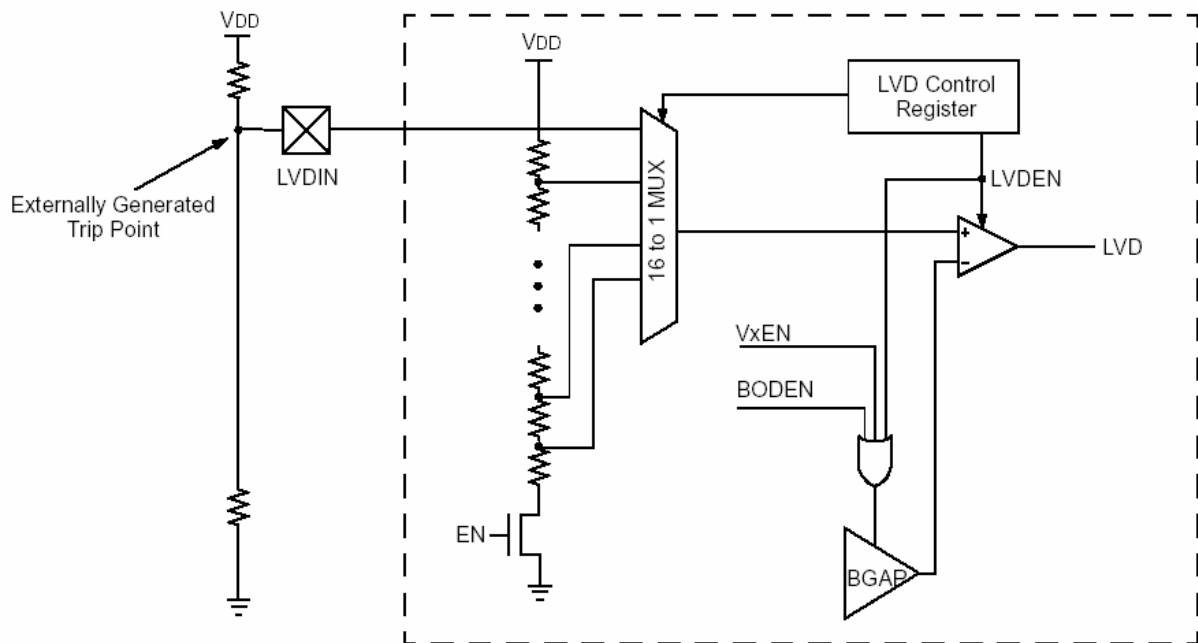
**ADCS2:** selection de l'horloge (voir tableau)

PCFG<3:0>	AN7	AN6	AN5	AN4	AN3	AN2	AN1	AN0	VREF+	VREF-	C/R
0000	A	A	A	A	A	A	A	A	VDD	VSS	8/00
0001	A	A	A	A	VREF+	A	A	A	AN3	VSS	7/1
0010	D	D	D	A	A	A	A	A	VDD	VSS	5/0
0011	D	D	D	A	VREF+	A	A	A	AN3	VSS	4/1
0100	D	D	D	D	A	D	A	A	VDD	VSS	3/0
0101	D	D	D	D	VREF+	D	A	A	AN3	VSS	2/1
011x	D	D	D	D	D	D	D	D	—	—	0/0
1000	A	A	A	A	VREF+	VREF-	A	A	AN3	AN2	6/2
1001	D	D	A	A	A	A	A	A	VDD	VSS	6/0
1010	D	D	A	A	VREF+	A	A	A	AN3	VSS	5/1
1011	D	D	A	A	VREF+	VREF-	A	A	AN3	AN2	4/2
1100	D	D	D	A	VREF+	VREF-	A	A	AN3	AN2	3/2
1101	D	D	D	D	VREF+	VREF-	A	A	AN3	AN2	2/2
1110	D	D	D	D	D	D	D	A	VDD	VSS	1/0
1111	D	D	D	D	VREF+	VREF-	D	A	AN3	AN2	1/2



## 11. Detection de faible tension (LVD)

Fonction low voltage detect



### REGISTRE LVDCON (0xFD2)

7	6	5	4	3	2	1	0
—	—	IRVST	LVDEN	LVDL3	LVDL2	LVDL1	LVDL0

**IRVST:** Internal Reference Voltage Stable Flag bit

1 = Indique que la référence de tension interne est stable

**LVDEN:** Low Voltage Detect

1 = Active la fonction LVD

0 = Fonction LVD désactivée

**LVDL3:LVDL0:** Choix de la tension limite avant détection

1111 = Tension externe (sur la broche LVDIN)

1110 = 4.5V - 4.77V

1101 = 4.2V - 4.45V

1100 = 4.0V - 4.24V

1011 = 3.8V - 4.03V

1010 = 3.6V - 3.82V

1001 = 3.5V - 3.71V

1000 = 3.3V - 3.50V

0111 = 3.0V - 3.18V

0110 = 2.8V - 2.97V

0101 = 2.7V - 2.86V

0100 = 2.5V - 2.65V

0011 = 2.4V - 2.54V

0010 = 2.2V - 2.33V

0001 = 2.0V - 2.12V

0000 = Reservé

```
// Demo pour LVD
#include <p18f452.h> //pour LCD
char test=0;
void main(void)
{
    // LVD active, pas d'IT
    // detection VDD<3.5v
    PIE2bits.LVDIE=0;
    LVDCON=0b00011001;
    While(!LVDCONbits.IRVST); //attend...
    while(1){
        if (PIR2bits.LVDIF)
        {
            PIR2bits.LVDIF=0;
            test=1;
        }
        else test=0;
    }
}
```

### Pour utiliser l'interruption :

LVDIF dans PIR2<2> (drapeau d'interruption)

LVDIE dans PIE2<2> (validation de l'interruption)

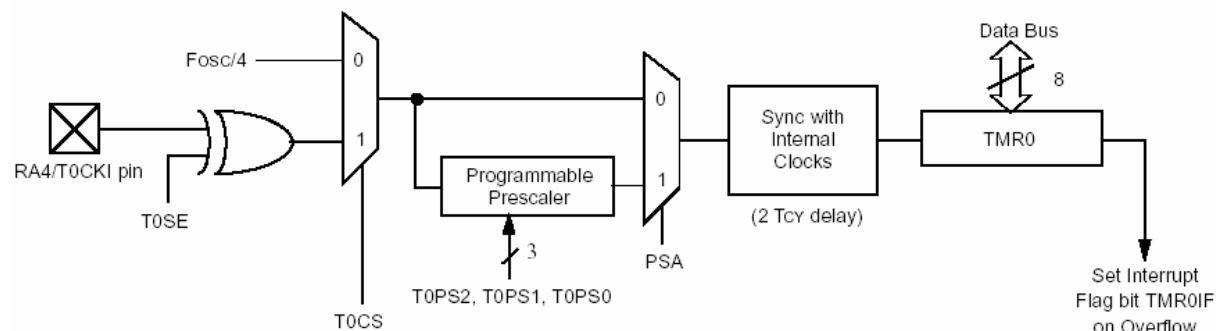
LVDIP dans IPR2<2> (priorité de l'interruption)



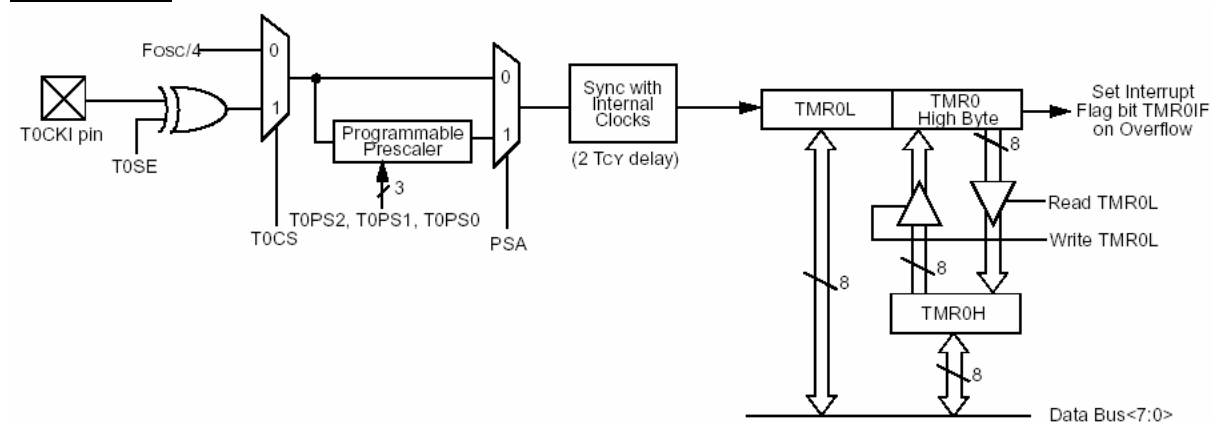


## 12. TIMER0

### Mode 8 bits



### Mode 16 bits



### REGISTRE T0CON

7	6	5	4	3	2	1	0
TMR0ON	T08BIT	T0CS	T0SE	PSA	T0PS2	T0PS1	T0PS0

**TMR0ON:** activation

1 = Active Timer0

0 = Stoppe Timer0

**T08BIT:** 8/16 bits

1 = Timer0 est un compteur 8-bits

0 = Timer0 est un compteur 16-bits

**T0CS:** sélection de l'horloge

1 = compte les fronts sur T0CKI

0 = compte les fronts sur l'horloge interne (CLKOUT)

**T0SE:** front détecté

1 = compte sur front descendant sur T0CKI

0 = compte sur front montant sur T0CKI

**PSA:** Pré diviser

1 = pas de pré diviseur.

0 = Le pré diviseur est activé

**T0PS2:T0PS0:** Pré Division de l'horloge

111 = 1:256

110 = 1:128

101 = 1:64

100 = 1:32

011 = 1:16

010 = 1:8

001 = 1:4

000 = 1:2

### Pour utiliser l'interruption :

TMR0F dans INTCON <5> (drapeau d'interruption)

TMR0E dans INTCON <2> (validation de l'interruption)

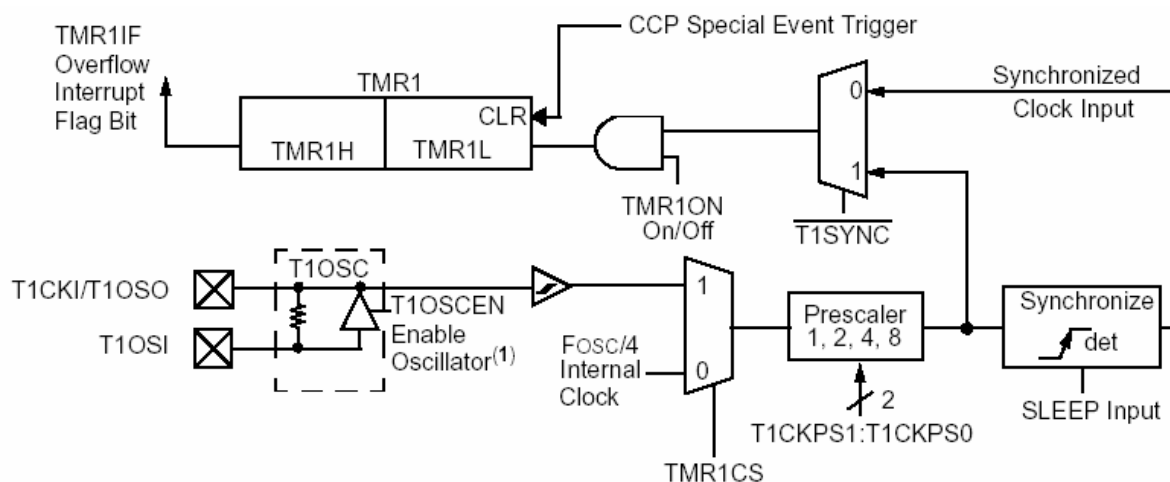
```
// Demo pour TMR0
#include <p18f452.h> //pour LCD

// sous programme d'interruption
#pragma interrupt itcomp
#pragma code interruption=0x8
void itcomp(void)
{
    PORTB^=0x01; // bascule PB0
    INTCONbits.TMR0IF=0;
}
#pragma code

void main(void)
{
    TRISB=0xFE; // PB0 en sortie
    // active Timer 16bits sur CLKOUT
    // prediviseur 1/8
    // avec Q=4MHz, CLK=1uS
    // IT toutes les 1*8*65536= 524mS
    T0CON=0b10000010;
    INTCONbits.TMR0IE=1; // autorise IT débordement
    RCONbits.IPEN=1; // Interruption prioritaires
    INTCONbits.GIE=1;
    While(1); // ne rien faire
}
```



## 13. TIMER1



### REGISTRE T1CON

7	6	5	4	3	2	1	0
RD16	—	T1CKPS1	T1CKPS0	T1OSCEN	T1SYNC	TMR1CS	TMR1ON

**RD16:** 16-bit Read/Write Mode Enable bit

1 = Accès à TMR1 par 16 bits

0 = Accès à TMR1 en 2 fois 8 bits

**T1CKPS1:T1CKPS0:** Valeur du prédiviseur

11 = 1:8

10 = 1:4

01 = 1:2

00 = 1:1

**T1OSCEN:** Validation de l'oscillateur (entrées sur T1OSO et T1OSI)

1 = activé

0 = désactivé

**T1SYNC:** Synchronisation avec l'horloge externe (pour le mode sleep)

Quand TMR1CS = 1:

1 = Ne pas synchroniser l'horloge externe

0 = Synchroniser l'horloge externe

**TMR1CS:** Choix de l'horloge

1 = Horloge externe

0 = Horloge interne (Fosc/4)

**TMR1ON:** Validation du TIMER1

1 = TIMER1 activé

0 = TIMER1 arrêté

### Remarques :

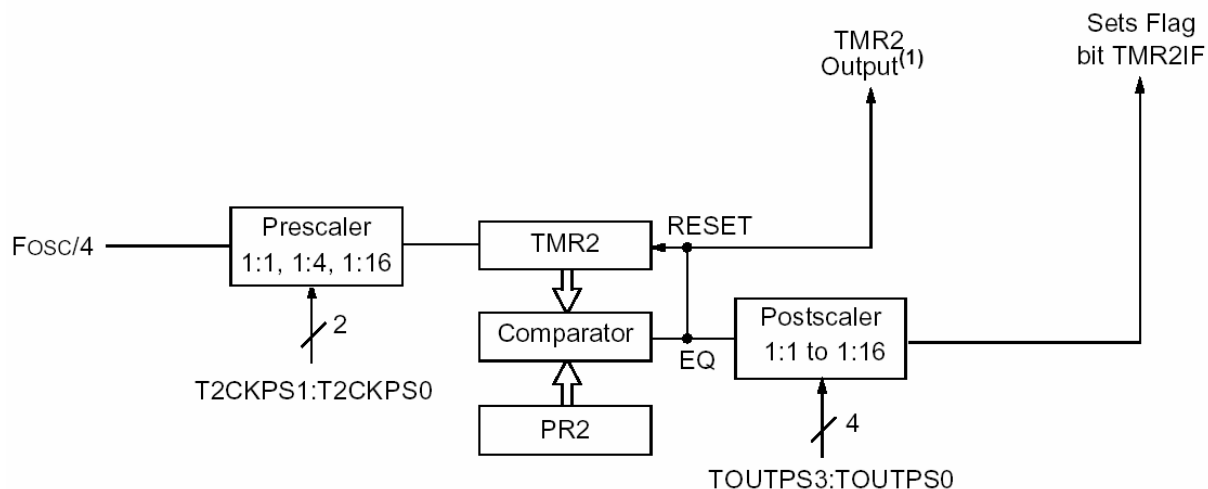
En plaçant un quartz de 32.768KHz sur T1OS, il est possible d'obtenir une base de temps de 1S.

Validation de l'IT de débordement par PIE1 <TMR1IE>

Drapeau d'IT PIR1<TMR1IF>



## 14. TIMER2



### REGISTRE TCON2

7	6	5	4	3	2	1	0
—	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0

**TOUTPS3:TOUTPS0:** Postdiviseur

0000 = 1:1

0001 = 1:2

1111 = 1:16

**TMR2ON:** Validation Timer2

1 = Timer2 activé

0 = Timer2 désactivé

**T2CKPS1:T2CKPS0:** Prédiviseur

00 = 1:1

01 = 1:4

1x = 1:16

Validation de l'IT de débordement par PIE1

<TMR2IE>

Drapeau d'IT PIR1<TMR2IF>

TMR2 et PR2 sont des registres 8 bits. Lorsqu'il y a égalité TMR2IF est mis à 1 et TMR2 à 0x00

TM2R peut servir d'horloge pour le mode PWM ou pour les communications synchrones (TMR2 output).

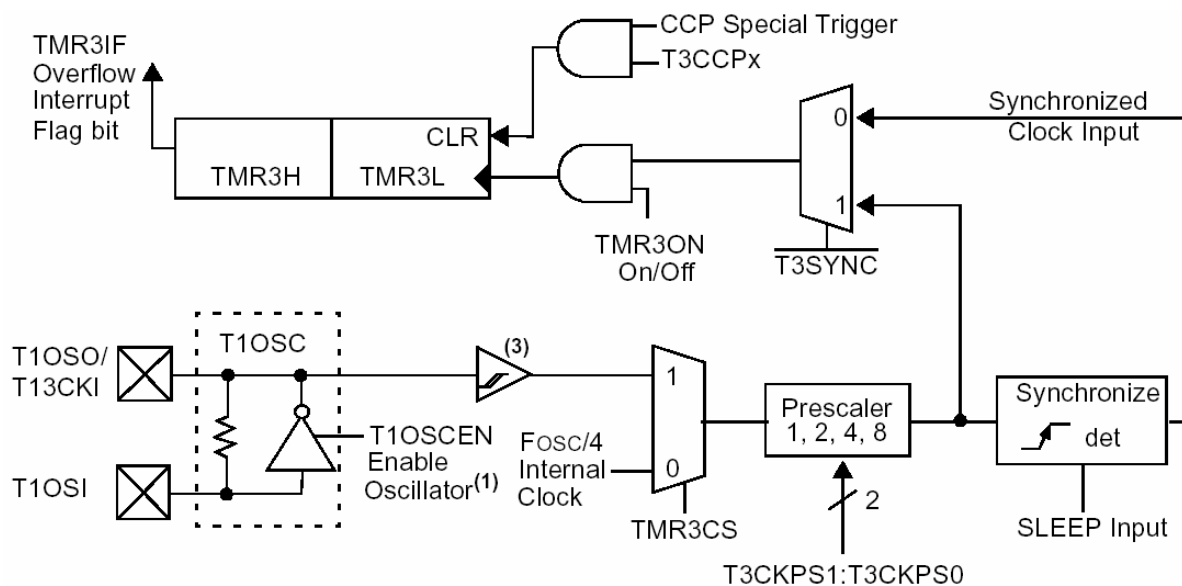
```
// Demo pour TMR2
#include <p18f452.h> //pour LCD

// sous programme d'interruption
#pragma interrupt itcomp
#pragma code interruption=0x8
void itcomp(void)
{static char tictac=7;
 if (!tictac--)
 {
    tictac=15; // environ 500ms
    PORTB^=0x01; // bascule PB0
 }
 PIR1.TMR2IF=0;
}
#pragma code

void main(void)
{
    TRISB=0xFE; // PB0 en sortie
    // active Timer2
    // prediviseur 1/16 post 1/16
    // avec Q=4MHz, CLK=1uS
    // IT toutes les T=1*16*16*125 = 32 mS
    PR2=125;
    TCON2=0b01111110;
    PIE1.TMR2IE=1; // autorise IT débordement
    RCONbits.IPEN=1; // Interruption prioritaires
    INTCONbits.GIE=1;
    While(1); // ne rien faire
}
```



## 15. TIMER3



REGISTRE TCON3

7	6	5	4	3	2	1	0
RD16	T3CCP2	T3CKPS1	T3CKPS0	T3CCP1	T3SYNC	TMR3CS	TMR3ON

**RD16:** Lecture/ écriture 16-bit Read/Write

1 = Accès à TMR3 sur 16 bits

0 = Accès à TMR3 sur 2x8 bits

**T3CCP2:T3CCP1:** Liaisons Timer3 et Timer1 et CCPx

1x = Timer3 est l'horloge du module compare/capture CCP

01 = Timer3 est l'horloge du module compare/capture CCP2,

Timer1 est l'horloge du module compare/capture CCP1

00 = Timer1 est l'horloge du module compare/capture CCP

**T3CKPS1:T3CKPS0:** Prédiviseur

11 = 1:8

10 = 1:4

01 = 1:2

00 = 1:1

**T3SYNC:** Synchronisation avec l'horloge externe (pour le mode sleep)

Quand TMR1CS = 1:

1 = Ne pas synchroniser l'horloge externe

0 = Synchroniser l'horloge externe

**TMR3CS:** Choix de l'horloge

1 = Horloge externe

0 = Horloge interne (Fosc/4)

**TMR3ON:** Validation du TIMER1

1 = TIMER1 activé

0 = TIMER1 arrêté

Validation de l'IT de débordement par PIE2 <TMR3IE>

Drapeau d'IT PIR2<TMR3IF>

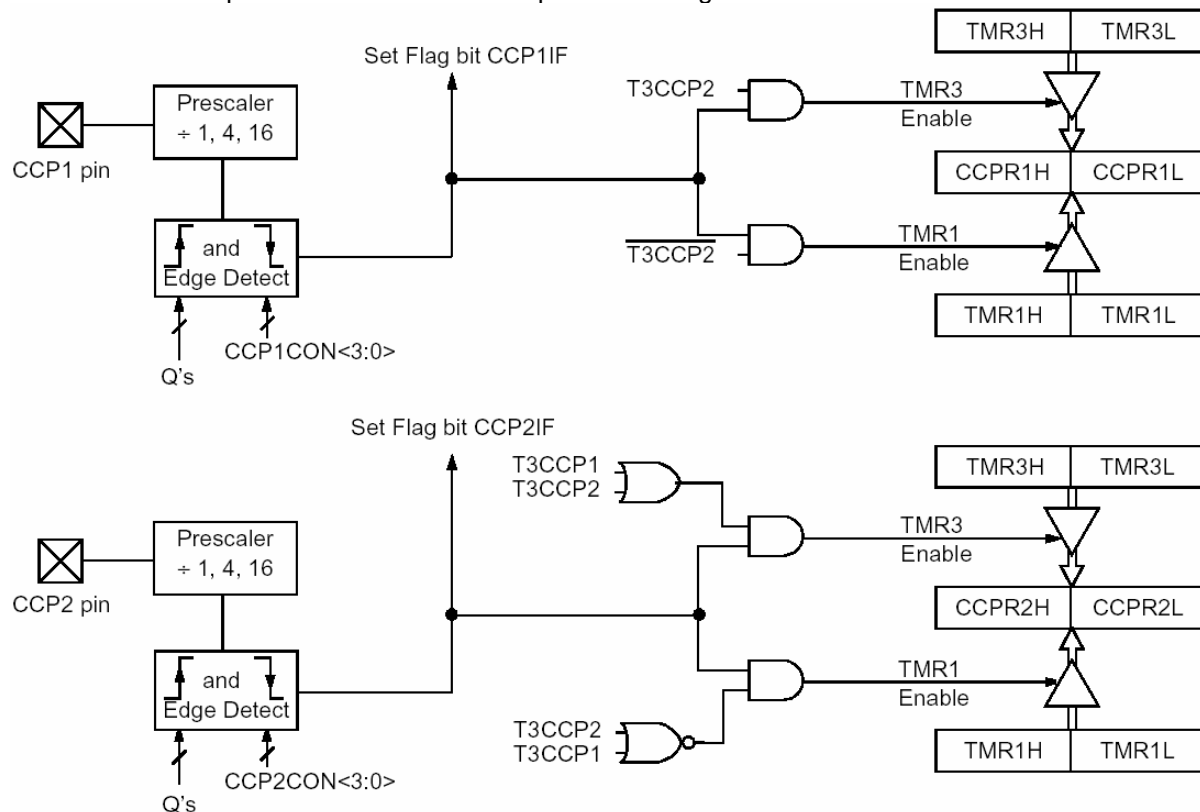


## 16. Capture/Compare/PWM

### 16.1. CAPTURE

La fonction capture permet de mesurer la durée d'une impulsion haute ou basse ou la période d'un signal rectangulaire. Le résultat est le nombre d'impulsions d'horloge entre deux fronts. Connaissant la période de l'horloge on en déduit un temps.

Il existe deux modules de capture CCP1 et CCP2. Lorsque que l'événement asynchrone attendu sur la broche CCP se produit un TIMER est recopié dans le registre CCPR



#### REGISTRE CCP1CON/CCP2CON

7	6	5	4	3	2	1	0
—	—	DCxB1	DCxB0	CCPxM3	CCPxM2	CCPxM1	CCPxM0

**DCxB1:DCxB0:** PWM rapport cyclique bit1 et bit0

Ces bits ne sont pas utilisés en mode capture/comparaison. Ils représentent les deux LSB du rapport cyclique (10bits) de la fonction PWM, les bits de poids forts se trouvent dans CCPRxL. Ces bits ne sont pas utilisés en mode capture/comparaison

**CCPxM3:CCPxM0:** Choix du mode CCPx

0000 = Capture/Compare/PWM désactivé

0001 = Réserve

0010 = comparaison, la sortie bascule lors de l'égalité (CCPxIF est à 1)

0011 = Réserve

0100 = capture sur front descendant

0101 = capture sur front montant

0110 = capture les 4 fronts descendants

0111 = capture tous les 4 fronts montants

1000 = comparaison, CCPx est initialisée à 0 et passe à 1 lors de l'égalité TMRx / CCPRx, CCPxIF est mis à 1

1001 = comparaison, CCPx est initialisée à 1 et passe à 0 lors de l'égalité TMRx / CCPRx, CCPxIF est mis à 1

1010 = comparaison, lors de l'égalité TMRx / CCPRx, CCPxIF est mis à 1, CCPx n'est pas modifié

1011 = comparaison, déclenche un « special event » CCPxIF est mis à 1

11xx = mode PWM

Validation de l'IT de capture pour CCP1 par PIE1 <CCP1IE>

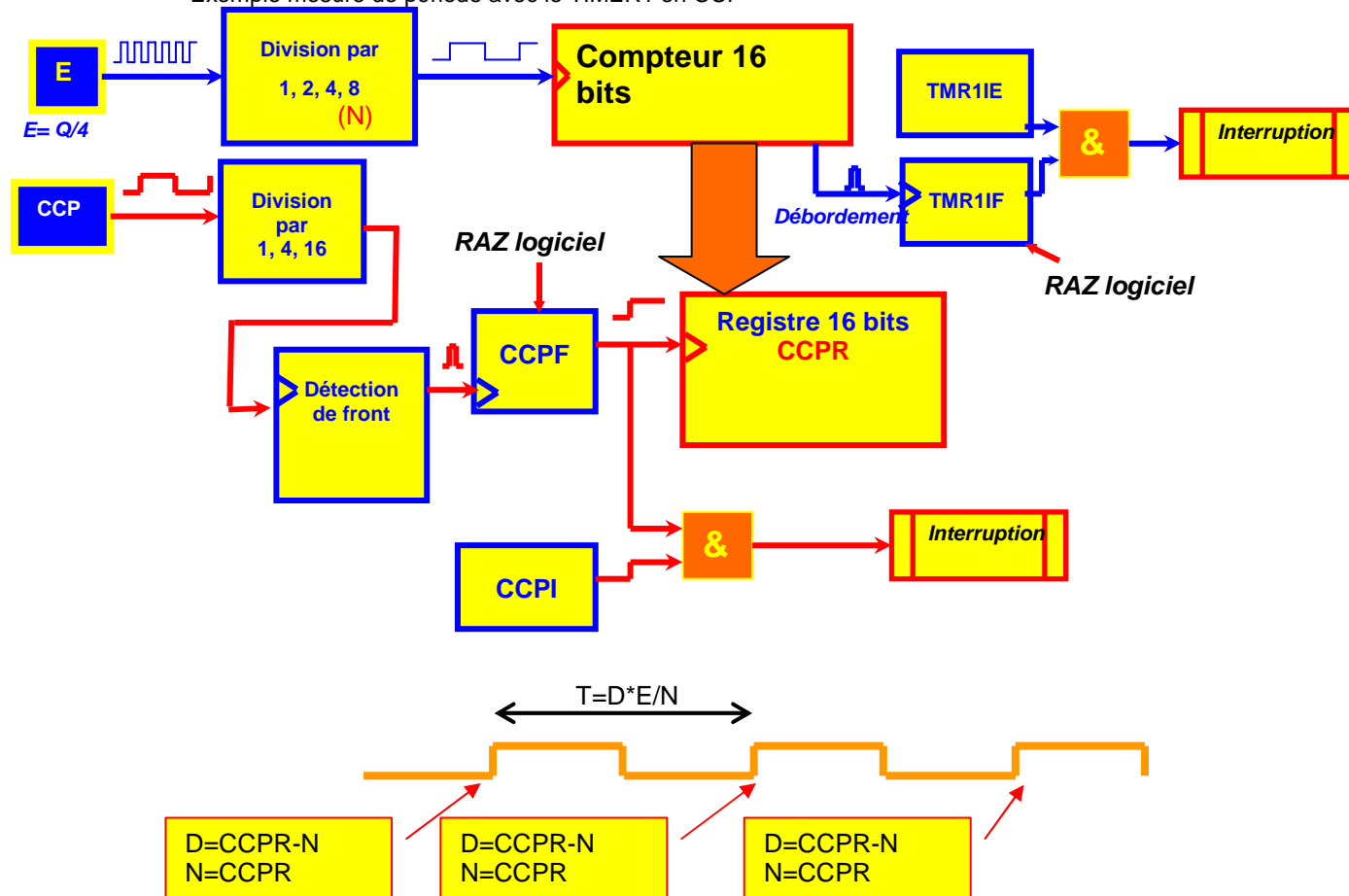
Drapeau d'IT PIR1<CCP1IF>

Validation de l'IT de capture pour CCP2 par PIE2 <CCP2IE>

Drapeau d'IT PIR2<CCP2IF>



# Exemple mesure de période avec le TIMER1 en CCP



A chaque front montant TIMER1 est recopié dans CCPR, une interruption est générée, le sous programme d'IT calcul  $D = CCPR - N$  puis  $N = CCPR$ . D représente la période du signal d'entrée

```
// Programme test de la fonction capture. Le nombre d'impulsions comptées
// entre deux fronts montants de CCPR1 est rangée dans la variable duree
#include <p18f452.h>
unsigned int duree; // représente le comptage entre 2 fronts
// sous programme d'interruption
#pragma interrupt itcomp
void itcomp(void)
{
    unsigned static int ancien;
    if (PIR1bits.CCP1IF) // l'IT provient d'une capture
    {
        duree = CCPR1 - ancien; // comptage entre les deux front
        ancien = CCPR1;
    }
    PIR1bits.CCP1IF = 0; // efface le drapeau d'IT
}
#pragma code interruption = 0x8
void fonction (void)
{
    _asm goto itcomp _endasm
}
#pragma code
void main(void)
{
    // configure PORTC CCP1
    DDRCbits.RC2 = 1; // RC2/CCP1 en entree
    // configure le TIMER1
    T1CONbits.RD16 = 0; // TMR1 mode simple (pas de RW)
    T1CONbits.TMR1CS = 0; // compte les impulsions sur internal clock
    T1CONbits.T1CKPS1 = 1; // prédiviseur = 1/8 periode sortie = 8us
    T1CONbits.T1CKPS0 = 1;
    T1CONbits.T1SYNC = 1; // pas de synchronisation sur sleep/Reset
    T1CONbits.TMR1ON = 1; // TMR1 Activé
    // configure le mode capture sur le TIMER1 avec IT sur CCP1
    T3CONbits.T3CCP2 = 0; // mode comparaison entre TMR1 et CCPR1
    CCP1CON = 0x05; // capture mode sur fronts montants
    PIE1bits.CCP1IE = 1; // active IT sur mode capture/comparaison CCP1
    RCONbits.IPEN = 1; // Interruption prioritaires activées
    INTCONbits.GIE = 1; // Toutes les IT démasquées autorisées
    while(1);
}
```

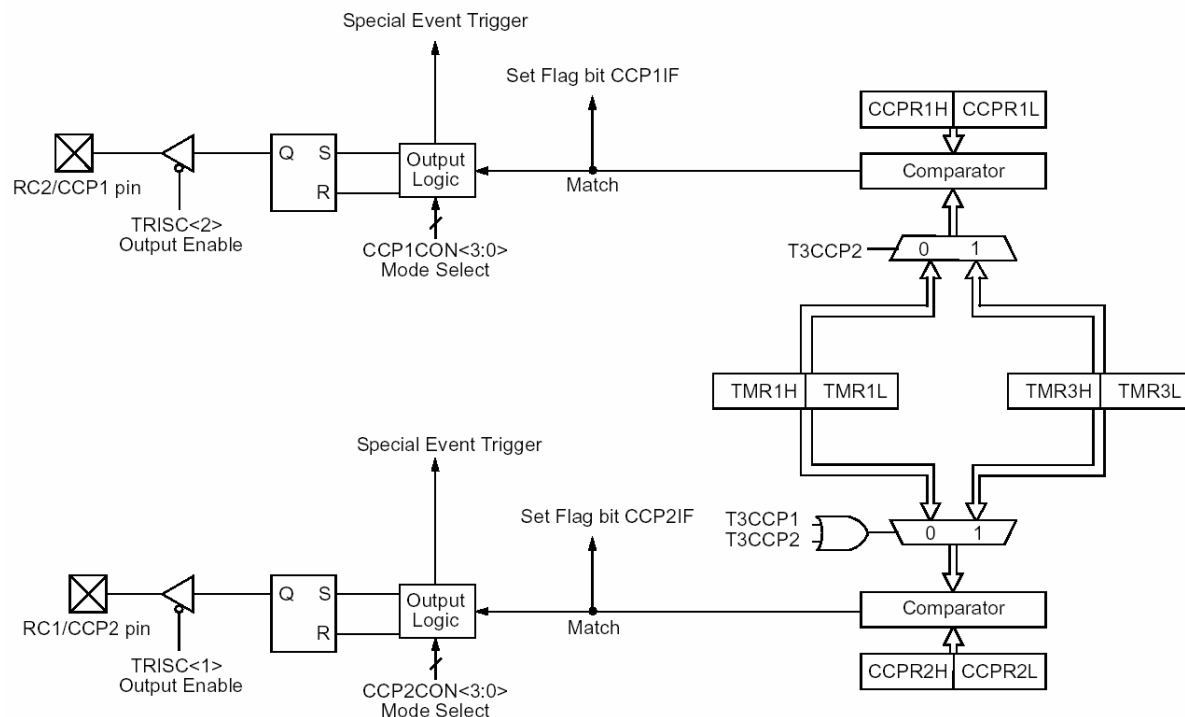


## 16.2. COMPARE

La fonction compare permet de produire des « durées », impulsions hautes ou basses calibrées ou des signaux rectangulaires périodiques. A chaque coïncidence TIMER/CCPR la broche CCP évolue en fonction de la configuration, une interruption est générée. Il suffit de recharger CCPR avec CCPR + nombre d'impulsions à compter, pour produire un nouveau changement après le temps désiré.

Il existe deux modules de comparaison CCP1 et CCP2

Lors de l'égalité entre un compteur TMR et un registre CCPR, une action est déclenchée sur la broche CCP correspondante



### REGISTRE CCP1CON/CCP2CON

7	6	5	4	3	2	1	0
—	—	DCxB1	DCxB0	CCPxM3	CCPxM2	CCPxM1	CCPxM0

**DCxB1:DCxB0:** PWM rapport cyclique bit1 et bit0

Ces bits ne sont pas utilisés en mode capture/compare. Ils représentent les deux LSB du rapport cyclique (10bits) de la fonction PWM, les bits de poids forts se trouve dans CCPRxL. Ces bits ne sont pas utilisés en mode capture/compare

**CCPxM3:CCPxM0:** Choix du mode CCPx

0000 = Capture/Compare/PWM désactivé

0001 = Réservé

0010 = comparaison, la sortie bascule lors de l'égalité (CCPxIF is à 1)

0011 = Réservé

0100 = capture sur front descendant

0101 = capture sur front montant

0110 = capture les 4 fronts descendants

0111 = capture tous les 4 front montants

1000 = comparaison, CCPx est initialisée à 0 et passe à 1 lors de l'égalité TMRx / CCPRx, CCPIF est mis à 1

1001 = comparaison, CCPx est initialisée à 1 et passe à 0 lors de l'égalité TMRx / CCPRx, CCPIF est mis à 1

1010 = comparaison, lors de l'égalité TMRx / CCPRx, CCPIF est mis à 1, CCPx n'est pas modifié

1011 = comparaison, déclenche un « special event » CCPIF est mis à 1

11xx = mode PWM

Validation de l'IT de comparaison pour CCP1 par PIE1 <CCP1IE>

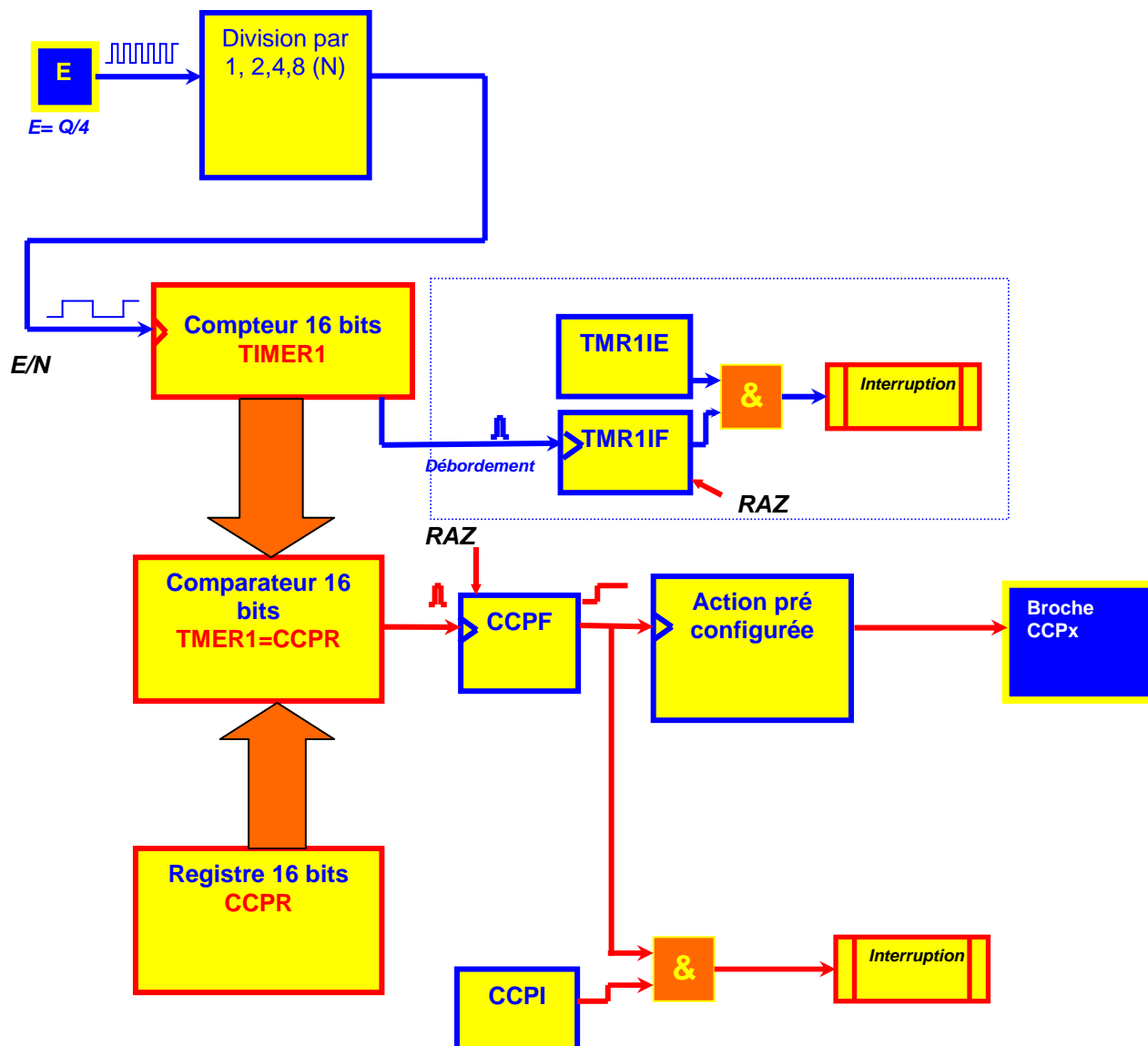
Drapeau d'IT PIR1<CCP1IF>

Validation de l'IT de comparaison pour CCP2 par PIE2 <CCP2IE>

Drapeau d'IT PIR2<CCP2IF>



## Exemple de production de signaux avec TIMER1



Valuer ajoutée à CCPR entre deux événements :  $D = \frac{\text{durée}}{ExN}$

Lors de chaque coïncidence la broche CCP évolue. Une interruption est générée, CCPR est incrémenté de la durée voulue.



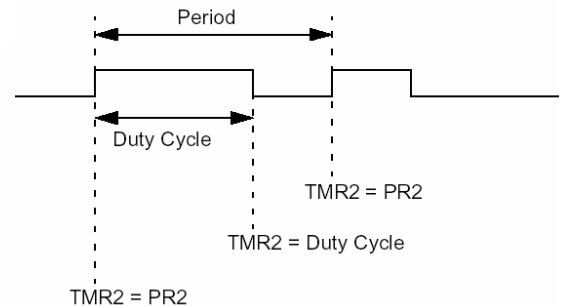
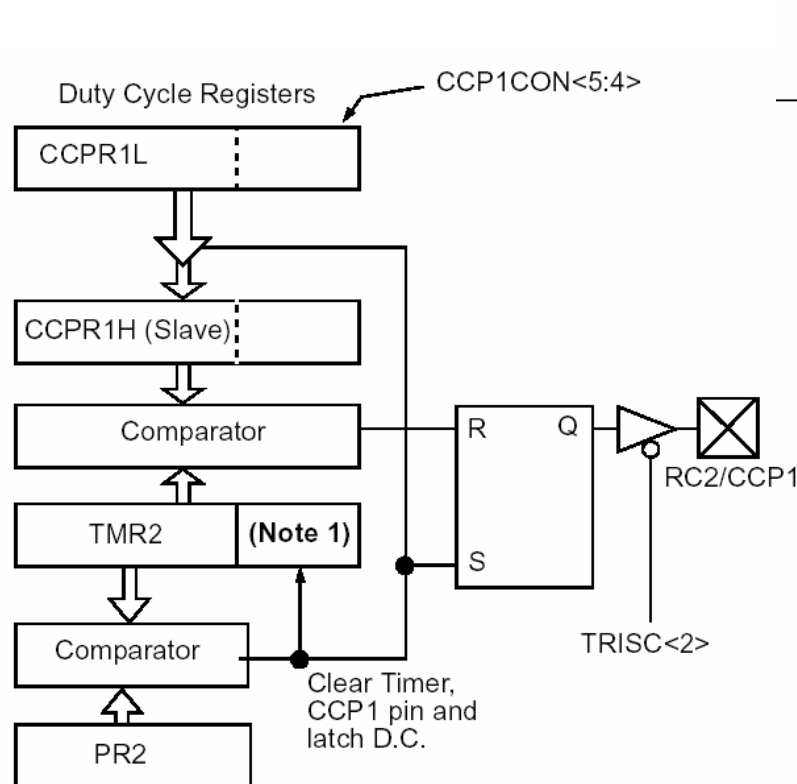
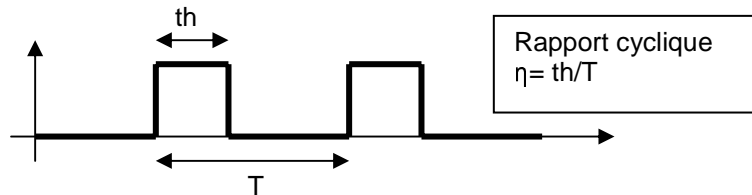


## 16.3. PWM

La modulation de largeur d'impulsion (ou de rapport cyclique) est couramment utilisée dans la commande des hacheurs (pour commander un moteur à courant continu par exemple).

La sortie CPP1 peut être configurée en PWM (TRISC<2> =0).

TMR2 cadence le processus, CCPR1 représente la durée de l'état haut et PR2 la période voir configuration de CCP1CON/CCP2CON (page 22)



CCP1=0  
Lorsque TMR2=PR2  
CCP1=1  
TMR2=0  
CCPR1H=CCPR1L  
Lorsque TMR2=CCPR1H  
CCP1=0  
Lorsque TMR2=PR2  
Etc...

PR2 représente la période T  
CCPR1L représente th

**Note:** 8-bit timer is concatenated with 2-bit internal Q clock or 2 bits of the prescaler to create 10-bit time-base.

### Exemples :

Fréquence PWM	2.44kHz	9.77kHz	39.06kHz	156.25kHz	312.50kHz	416.67kHz
Prédiviseur (1,4,16)	16	4	1	1	1	1
Valeur PR2	0xFF	0xFF	0xFF	0x3F	0x1F	0x17
Résolution Max bits)	14	12	10	8	7	6.58

Période PWM = (PR2 + 1) • 4 • TOSC • (TMR2 valeur du prédiviseur)

Rapport cyclique PWM= (CCPR1L:CCP1CON<5:4>) • TOSC • (TMR2 valeur du prédiviseur)

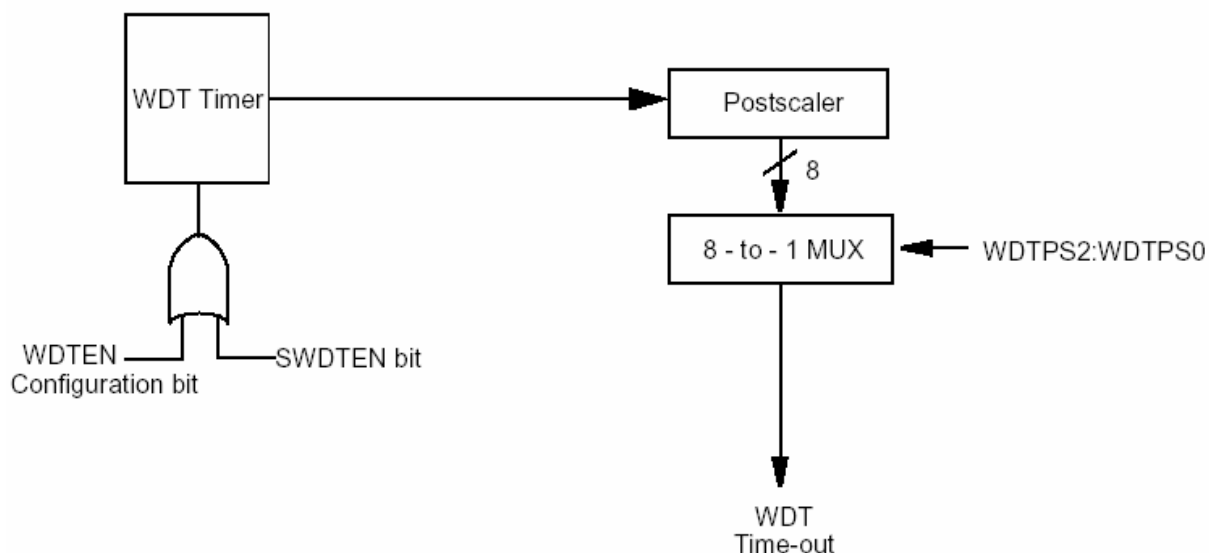
$$\text{Résolution(PWM) en bits} = \frac{\log \left( \frac{F_{OSC}}{F_{PWM}} \right)}{\log(2)}$$



## 17. Chien de garde

Watch Dog Timer, cette fonction peut être activée automatiquement par configuration dans MPLAB

Configuration Bits			
Address	Value	Category	Setting
300001	FA	Oscillator	HS
		Osc. Switch Enable	Disabled
300002	FF	Power Up Timer	Disabled
		Brown Out Detect	Enabled
		Brown Out Voltage	2.0V
300003	FE	Watchdog Timer	Disabled
		Watchdog Postscaler	Enabled
300005	FF	CCP2 Mux	Disabled
300006	7B	Low Voltage Program	Disabled
		Background Debug	Enabled



Le chien de garde est activé en mettant à 1 le bit 0 du registre WDTCON (SWDTEN) ou par WDTEEN de CONFIG2H si la configuration de départ n'a pas activée le chien de garde (voir configuration sur MPLAB).

### REGISTRE CONFIG2H

7	6	5	4	3	2	1	0
—	—	—	—	WDTPS2	WDTPS1	WDTPS0	WDTEEN

Les bits WDTPS2-WDTPS0 représente le rapport de division de la sortie WDT TIMER (1 à 8)

Après activation le chien de garde doit être réinitialisé avant la génération du Time-Out qui provoque un RESET.

Les instructions assembleur « clrwtd » et « sleep » remettent le TIMER à 0.

Durée de comptage avant Time-Out et sans prédiviseur  $7\text{mS} < T < 33\text{mS}$

Avec un prédiviseur de 5 :  $35\text{mS} < T < 165\text{mS}$



## 18. Communications séries asynchrones

**Remarque** : seul le mode asynchrone est traité.

Les communications séries asynchrones suivent le format NZR (No Return to Zero). Ils communiquent avec le microprocesseur par l'intermédiaire du bus de données et en série avec l'extérieur par une liaison série asynchrone (sans horloge). Ils peuvent par ailleurs piloter un modem.

**Exemple de frame** : asynchrone 1start, 8 bits, parité paire, 1 stop : nombre 10010001



**Parité paire** : le bit de parité est positionné pour que l'ensemble des bits de donnée et le bit de parité représente un nombre de bits à 1 pair

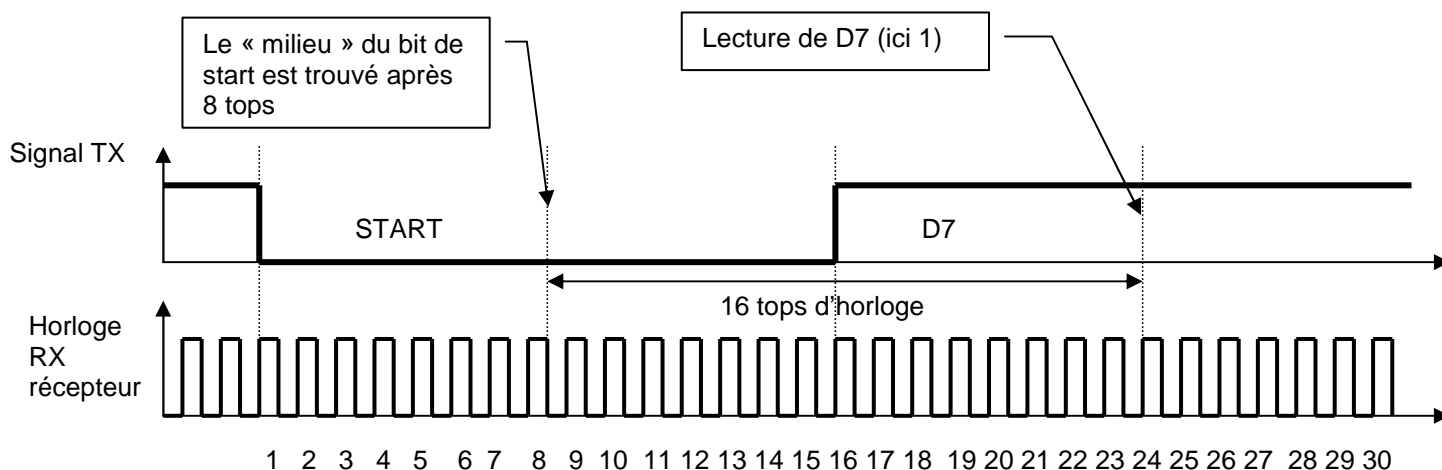
**Parité impaire** : le bit de parité est positionné pour que l'ensemble des bits de donnée et le bit de parité représente un nombre de bits à 1 impair

Dans ce type de transmission l'horloge de transmission est comprise dans le signal, le bit de start est utilisé par le récepteur pour se synchroniser avec l'émetteur. Cependant les deux horloges de transmission et de réception sont au départ très proche

L'horloge de réception possède une fréquence multiple de celle de transmission (en général x16 ou x64)

Dans le cas d'une division par 16 :

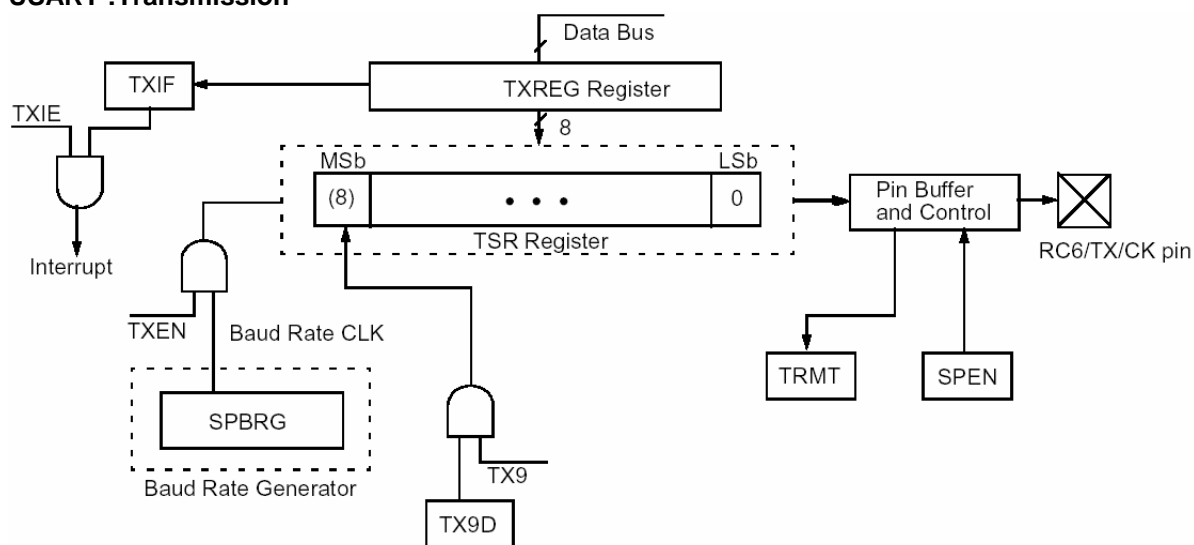
Lors de la réception du front descendant du bit de start, l'USART attend 8 tops d'horloge, le circuit reçoit alors théoriquement le milieu du bit de start, l'USART attend ensuite 16 tops d'horloge, le circuit reçoit alors le milieu de D7 et lit ce bit, l'USART attend ensuite 16 tops etc. L'horloge du récepteur est donc resynchronisée lors de la réception de chaque caractère.



L'horloge RX (réception) doit donc toujours être supérieure à celle de TX (transmission). En réalité les deux horloges sont identiques et TX est divisé dans l'USART pour produire la vitesse de transmission souhaité.

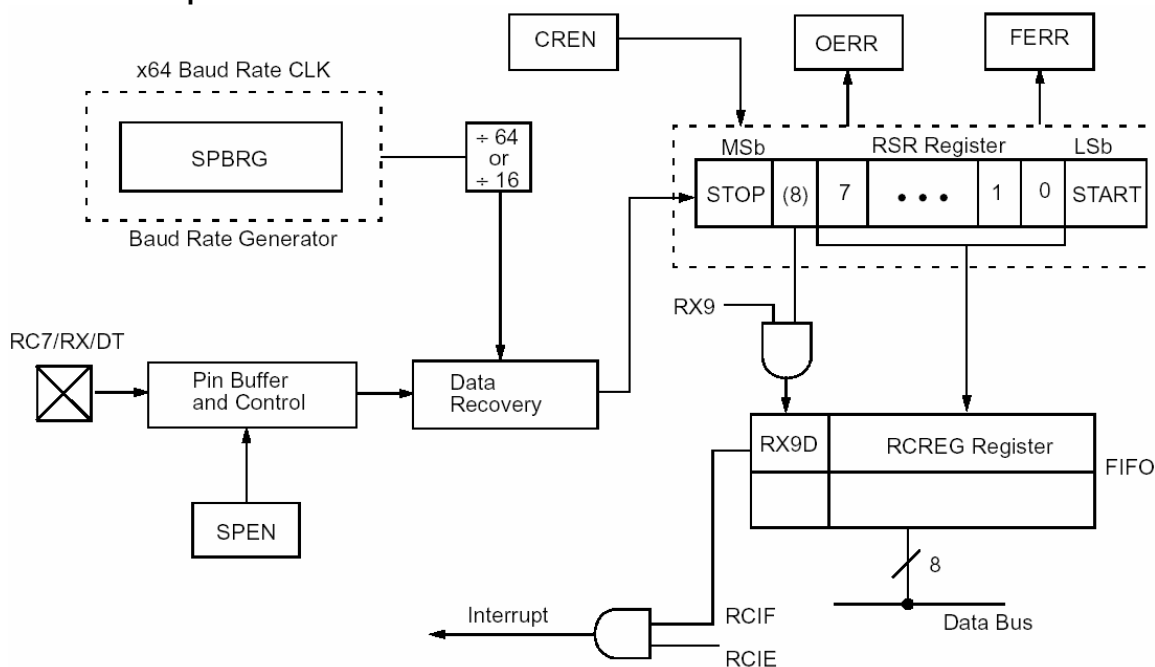


## USART :Transmission



TXREG : registre de transmission (tampon)  
 SPBRG : définit la vitesse de transmission (BAUD)  
 TXEN : valide l'horloge  
 TX9 : valide le 9<sup>ème</sup> bit  
 TX9D : 9<sup>ème</sup> bit (donnée, adresse ou parité)  
 TXIE : autorise l'interruption  
 TXIF : drapeau d'interruption, indique que TXREG est vide  
 SPEN : configure TX/RX pin pour USART  
 TMRT : indique si TSR est vide

## USART : Reception



CREN : active le récepteur asynchrone  
 SPBRG : définit la vitesse de transmission (BAUD)  
 SPEN : configure TX/RX pin pour USART  
 RCIE : autorise l'interruption en réception  
 RCIF : drapeau d'interruption de réception d'une donnée  
 RX9 : valide la prise en compte de D8 (adresse, donnée ou parité, traitement par logiciel)  
 OERR, FERR : indicateurs d'erreurs de réception



## Registres généraux

### TXSTA : Registre état et de contrôle des transmissions

7	6	5	4	3	2	1	0
CSRC	TX9	TXEN	SYNC	-	BRGH	TRMT	TX9D

**CSRC:** Clock Source Select bit

Non utilisé en mode asynchrone

1 = mode maître (horloge générée en interne depuis BRG)

0 = mode esclave (horloge externe)

**TX9:**

1 = transmission sur 9 bits

0 = transmission sur 8 bits

**TXEN:**

1 = transmissions activées

0 = transmissions désactivées

**Note:** SREN/CREN sont prioritaires sur TXEN en mode SYNC

**SYNC:** USART Mode Select bit

1 = mode synchrone

0 = mode asynchrone

**BRGH:** High Baud Rate Select bit

Mode asynchrone :

1 = grande vitesse

0 = petite vitesse

inutilisé en mode synchrone

**TRMT:** Transmit Shift Register Status bit

1 = TSR vide

0 = TSR plein

**TX9D:** 9th bit of Transmit Data

Peut être une adresse, une donnée ou un bit de parité

### RCSTA: RECEIVE STATUS AND CONTROL REGISTER

7	6	5	4	3	2	1	0
SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D

**SPEN:** Serial Port Enable bit

1 = Active le port série (configure RX/DT et TX/CK comme des broches de port série)

0 = Désactive le port série

**RX9:** 9-bit Receive Enable bit

1 = Réception sur 9 bits, 0 = Réception sur 8 bits

**SREN:** Single Receive Enable bit

Inutilisé en modes asynchrones et en mode synchrone esclave

En mode synchrone et maître

1 = Autorise une réception unique (effacé après la réception)

0 = interdit la réception

**CREN:** Continuous Receive Enable bit

En mode asynchrone

1 = Active le récepteur

0 = Désactive le récepteur

En mode synchrone

1 = Active la réception (CREN est prioritaire sur SREN)

0 = Désactive la réception

**ADDEN:** Address Detect Enable bit

En mode asynchrone sur 9 bits (RX9 = 1):

1 = Active la détection d'adresse, autorise l'interruption et ne charge pas la donnée dans le buffer de réception quand RSR<8>=1

0 = Désactive la détection d'adresse, tous les bits sont envoyés dans le buffer de réception et le 9<sup>ième</sup> bit peut être utilisé comme bit de parité

**FERR:** Framing Error bit

1 = Framing error (mis à jour par une lecture de RCREG et la réception du prochain octet)

0 = No framing error

**OERR:** Overrun Error bit

1 = Overrun error (effacé en effaçant CREN), 0 = No overrun error

**RX9D:** 9th bit of Received Data

Peut être un bit d'adresse ou de donnée ou de parité et doit être géré par logiciel

Générateur de BAUD en mode asynchrone (si BRGH=0) – valeurs à placer dans SPBRG

	Fosc = 40 MHz	33 MHz	25 MHz	20 MHz
--	---------------	--------	--------	--------



BAUD RATE (Kbps)	SPBRG value			SPBRG value			SPBRG value			SPBRG value		
	% (decimal)			% (decimal)			% (decimal)			% (decimal)		
	KBAUD	ERROR		KBAUD	ERROR		KBAUD	ERROR		KBAUD	ERROR	
0.3	NA	-	-	NA	-	-	NA	-	-	NA	-	-
1.2	NA	-	-	NA	-	-	NA	-	-	NA	-	-
2.4	NA	-	-	NA	-	-	NA	-	-	NA	-	-
9.6	NA	-	-	NA	-	-	NA	-	-	NA	-	-
19.2	NA	-	-	NA	-	-	NA	-	-	NA	-	-
76.8	76.92	+0.16	129	77.10	+0.39	106	77.16	+0.47	80	76.92	+0.16	64
96	96.15	+0.16	103	95.93	-0.07	85	96.15	+0.16	64	96.15	+0.16	51
300	303.03	+1.01	32	294.64	-1.79	27	297.62	-0.79	20	294.12	-1.96	16
500	500	0	19	485.30	-2.94	16	480.77	-3.85	12	500	0	9
HIGH	10000	-	0	8250	-	0	6250	-	0	5000	-	0
LOW	39.06	-	255	32.23	-	255	24.41	-	255	19.53	-	255

Fosc = 16 MHz												
10 MHz												
7.15909 MHz												
5.0688 MHz												
BAUD RATE (Kbps)	SPBRG value			SPBRG value			SPBRG value			SPBRG value		
	% (decimal)			% (decimal)			% (decimal)			% (decimal)		
	KBAUD	ERROR		KBAUD	ERROR		KBAUD	ERROR		KBAUD	ERROR	
0.3	NA	-	-	NA	-	-	NA	-	-	NA	-	-
1.2	NA	-	-	NA	-	-	NA	-	-	NA	-	-
2.4	NA	-	-	NA	-	-	NA	-	-	NA	-	-
9.6	NA	-	-	NA	-	-	9.62	+0.23	185	9.60	0	131
19.2	19.23	+0.16	207	19.23	+0.16	129	19.24	+0.23	92	19.20	0	65
76.8	76.92	+0.16	51	75.76	-1.36	32	77.82	+1.32	22	74.54	-2.94	16
96	95.24	-0.79	41	96.15	+0.16	25	94.20	-1.88	18	97.48	+1.54	12
300	307.70	+2.56	12	312.50	+4.17	7	298.35	-0.57	5	316.80	+5.60	3
500	500	0	7	500	0	4	447.44	-10.51	3	422.40	-15.52	2
HIGH	4000	-	0	2500	-	0	1789.80	-	0	1267.20	-	0
LOW	15.63	-	255	9.77	-	255	6.99	-	255	4.95	-	255

Fosc = 4 MHz												
3.579545 MHz												
1 MHz												
32.768 kHz												
BAUD RATE (Kbps)	SPBRG value			SPBRG value			SPBRG value			SPBRG value		
	% (decimal)			% (decimal)			% (decimal)			% (decimal)		
	KBAUD	ERROR		KBAUD	ERROR		KBAUD	ERROR		KBAUD	ERROR	
0.3	NA	-	-	NA	-	-	NA	-	-	0.30	+1.14	26
1.2	NA	-	-	NA	-	-	1.20	+0.16	207	1.17	-2.48	6
2.4	NA	-	-	NA	-	-	2.40	+0.16	103	2.73	+13.78	2
9.6	9.62	+0.16	103	9.62	+0.23	92	9.62	+0.16	25	8.20	-14.67	0
19.2	19.23	+0.16	51	19.04	-0.83	46	19.23	+0.16	12	NA	-	-
76.8	76.92	+0.16	12	74.57	-2.90	11	83.33	+8.51	2	NA	-	-
96	1000	+4.17	9	99.43	+3.57	8	83.33	-13.19	2	NA	-	-
300	333.33	+11.11	2	298.30	-0.57	2	250	-16.67	0	NA	-	-
500	500	0	1	447.44	-10.51	1	NA	-	-	NA	-	-
HIGH	1000	-	0	894.89	-	0	250	-	0	8.20	-	0
LOW	3.91	-	255	3.50	-	255	0.98	-	255	0.03	-	255

Générateur de BAUD en mode asynchrone (si BRGH=1) – valeurs à placer dans SPBRG

BAUD RATE (Kbps)	Fosc = 40 MHz			33 MHz			25 MHz			20 MHz		
	SPBRG value			SPBRG value			SPBRG value			SPBRG value		
	% (decimal)			% (decimal)			% (decimal)			% (decimal)		
	KBAUD	ERROR		KBAUD	ERROR		KBAUD	ERROR		KBAUD	ERROR	
0.3	NA	-	-	NA	-	-	NA	-	-	NA	-	-
1.2	NA	-	-	NA	-	-	NA	-	-	NA	-	-
2.4	NA	-	-	2.40	-0.07	214	2.40	-0.15	162	2.40	+0.16	129
9.6	9.62	+0.16	64	9.55	-0.54	53	9.53	-0.76	40	9.47	-1.36	32
19.2	18.94	-1.36	32	19.10	-0.54	26	19.53	+1.73	19	19.53	+1.73	15
76.8	78.13	+1.73	7	73.66	-4.09	6	78.13	+1.73	4	78.13	+1.73	3
96	89.29	-6.99	6	103.13	+7.42	4	97.66	+1.73	3	104.17	+8.51	2
300	312.50	+4.17	1	257.81	-14.06	1	NA	-	-	312.50	+4.17	0
500	625	+25.00	0	NA	-	-	NA	-	-	NA	-	-
HIGH	625	-	0	515.63	-	0	390.63	-	0	312.50	-	0
LOW	2.44	-	255	2.01	-	255	1.53	-	255	1.22	-	255

Fosc = 16 MHz												
10 MHz												
7.15909 MHz												
5.0688 MHz												
BAUD RATE (Kbps)	SPBRG value			SPBRG value			SPBRG value			SPBRG value		
	% (decimal)			% (decimal)			% (decimal)			% (decimal)		
	KBAUD	ERROR		KBAUD	ERROR		KBAUD	ERROR		KBAUD	ERROR	



0.3	NA	-	-	NA	-	-	NA	-	-	NA	-	-
1.2	1.20	+0.16	207	1.20	+0.16	129	1.20	+0.23	92	1.20	0	65
2.4	2.40	+0.16	103	2.40	+0.16	64	2.38	-0.83	46	2.40	0	32
9.6	9.62	+0.16	25	9.77	+1.73	15	9.32	-2.90	11	9.90	+3.13	7
19.2	19.23	+0.16	12	19.53	+1.73	7	18.64	-2.90	5	19.80	+3.13	3
76.8	83.33	+8.51	2	78.13	+1.73	1	111.86	+45.65	0	79.20	+3.13	0
96	83.33	-13.19	2	78.13	-18.62	1	NA	-	-	NA	-	-
300	250	-16.67	0	156.25	-47.92	0	NA	-	-	NA	-	-
500	NA	-	-	NA	-	-	NA	-	-	NA	-	-
HIGH	250	-	0	156.25	-	0	111.86	-	0	79.20	-	0
LOW	0.98	-	255	0.61	-	255	0.44	-	255	0.31	-	255

BAUD RATE (Kbps)	Fosc = 4 MHz			3.579545 MHz			1 MHz			32.768 kHz		
			SPBRG value			SPBRG value			SPBRG value			SPBRG value
	%		(decimal)	%		(decimal)	%		(decimal)	%		(decimal)
	KBAUD	ERROR		KBAUD	ERROR		KBAUD	ERROR		KBAUD	ERROR	
0.3	0.30	-0.16	207	0.30	+0.23	185	0.30	+0.16	51	0.26	-14.67	1
1.2	1.20	+1.67	51	1.19	-0.83	46	1.20	+0.16	12	NA	-	-
2.4	2.40	+1.67	25	2.43	+1.32	22	2.23	-6.99	6	NA	-	-
9.6	8.93	-6.99	6	9.32	-2.90	5	7.81	-18.62	1	NA	-	-
19.2	20.83	+8.51	2	18.64	-2.90	2	15.63	-18.62	0	NA	-	-
76.8	62.50	-18.62	0	55.93	-27.17	0	NA	-	-	NA	-	-
96	NA	-	-	NA	-	-	NA	-	-	NA	-	-
300	NA	-	-	NA	-	-	NA	-	-	NA	-	-
500	NA	-	-	NA	-	-	NA	-	-	NA	-	-
HIGH	62.50	-	0	55.93	-	0	15.63	-	0	0.51	-	0
LOW	0.24	-	255	0.22	-	255	0.06	-	255	0.002	-	255

Exemple :

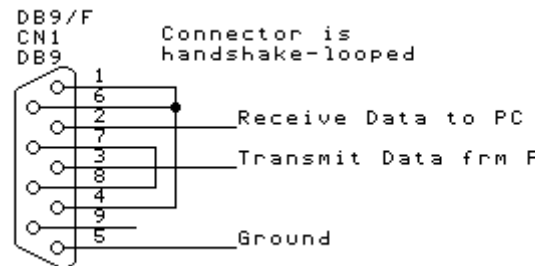
```
// Test des communications asynchrones sans IT
// connecter un émulateur de terminal sur le port série de PICDEM2+
#include <p18f452.h>
rom char mess[]="\nLes communications sont ouvertes\nTapez une touche ...\n\n";
// indique qu'un caractère est dans RCREG de l'USART
char data_recue(void) // reception d'une interruption
{
    if (PIR1bits.RCIF) // char reçu en reception*/
    {
        PIR1bits.RCIF=0; // efface drapeau
        return (1); // indique qu'un nouveau caractère est dans RCREG
    }
    else return (0); // pas de nouveau caractère reçu
}

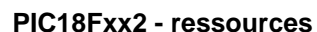
// envoie un caractère sur USART
void putch(unsigned char c) //putch est défini sur
{
    while(!TXSTAbits.TRMT); // pas de transmission
    TXREG=c; // envoie un caractère
    while(!PIR1bits.TXIF);
}

// envoie une chaîne en ROM
void putchaine(rom char* chaine)
{
    while (*chaine) putch(*chaine++);
}

void main(void)
{
    SPBRG = 25; // configure la vitesse (BAUD) 9600 N 8 1*/
    TXSTA = 0x24;
    RCSTA = 0x90; // active l'USART*/

    putchaine(mess); // intro
    while(1) // echo
    {
        if (data_recue()) putch(RCREG);
    }
}
```



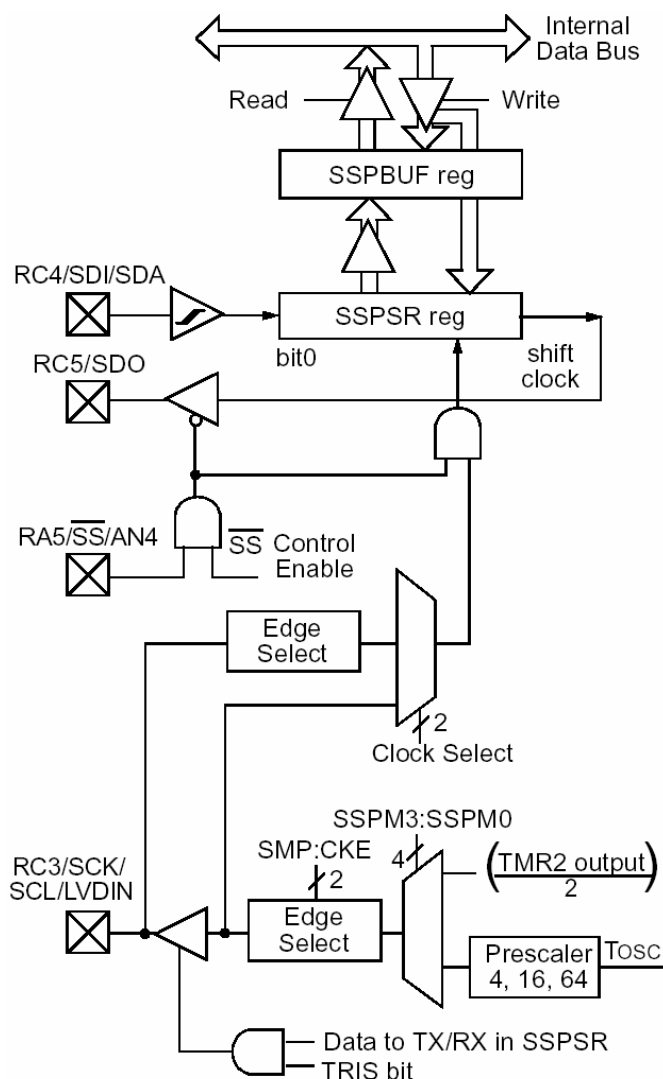


The diagram illustrates the hardware structure of the SPI Master SSPM3:SSPM0, which is part of PROCESSOR 1. The structure is enclosed in a dashed box and includes the following components and connections:

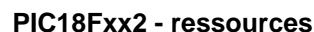
- SPI Master SSPM3:SSPM0 = 00xxb**: This label is positioned at the top left of the dashed box, indicating the specific hardware module.
- SDO**: The Serial Data Out signal line, which originates from the top right of the dashed box and connects to the **Serial Input Buffer (SSPBUF)**.
- Serial Input Buffer (SSPBUF)**: A rectangular block that receives data from the SDO line and outputs it to the **Shift Register (SSPSR)** via a large upward-pointing arrow.
- Shift Register (SSPSR)**: A rectangular block that receives data from the SSPBUF and outputs it to the **SDI** line. It is also labeled with **MSb** (Most Significant Bit) on the left and **LSb** (Least Significant Bit) on the right.
- SDI**: The Serial Data In signal line, which originates from the right side of the dashed box and connects to the Shift Register (SSPSR).
- SCK**: The Serial Clock signal line, which originates from the bottom right of the dashed box and connects to the **Serial Clock** input of the Shift Register (SSPSR).
- Serial Clock**: The label for the clock signal input to the Shift Register (SSPSR).
- PROCESSOR 1**: The label for the processor to which this hardware module belongs, located at the bottom center of the diagram.

En mode SPI, (Sérial Peripheral Interface) les données sont échangées entre deux registres à décalage. C'est le maître qui cadence l'échange avec son horloge. Il n'y a pas de protocole. L'horloge peut être programmée à  $F_{osc}/4$ . L'échange est alors extrêmement rapide, des précautions sont à prendre quant au câblage sur circuit imprimé. Ce type de communication est généralement réservé à l'échange de données entre deux circuits intégrés sur un même circuit imprimé (ex : avec CAN, CNA)

Lors de la configuration de l'émetteur, il faut s'assurer de la bonne synchronisation avec le récepteur (voir page 41)







The diagram illustrates the timing for four SPI clock modes, grouped by a bracket on the right labeled "4 Clock Modes". The modes are defined by the combination of CKP (Clock Polarity) and CKE (Clock Enable) values:

- SCK (CKP = 0, CKE = 0)
- SCK (CKP = 1, CKE = 0)
- SCK (CKP = 0, CKE = 1)
- SCK (CKP = 1, CKE = 1)

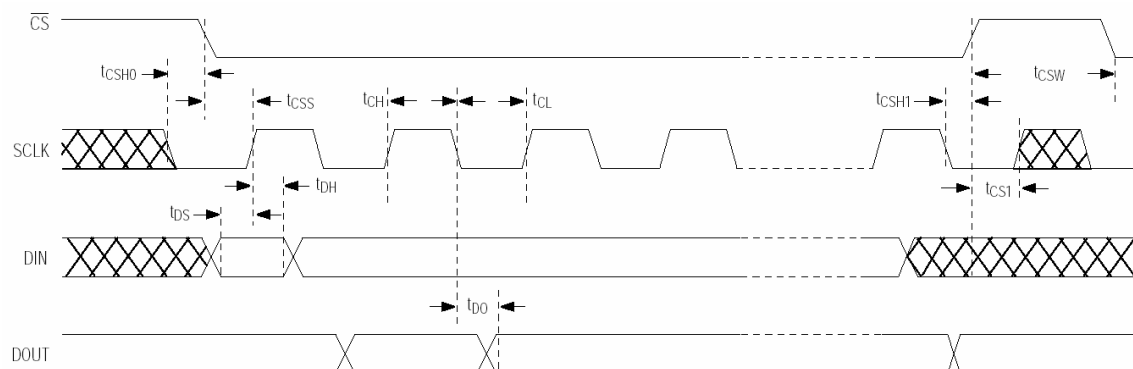
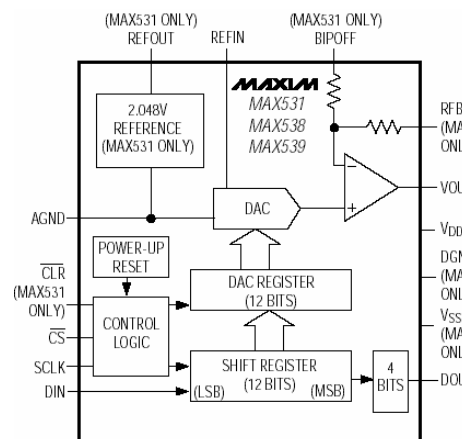
Below the SCK signals, the diagram shows the data bus signals and sampling:

- SDO (CKE = 0):** Data output signal. The first bit (bit7) is shown in a shaded box, indicating it is driven during the first clock cycle. Subsequent bits (bit6 to bit0) are shown in boxes with diagonal lines, indicating they are sampled during the first clock cycle.
- SDO (CKE = 1):** Data output signal. The first bit (bit7) is shown in a shaded box, indicating it is driven during the first clock cycle. Subsequent bits (bit6 to bit0) are shown in boxes with diagonal lines, indicating they are sampled during the first clock cycle.
- SDI (SMP = 0):** Data input signal. The first bit (bit7) is shown in a box with diagonal lines, indicating it is sampled during the first clock cycle. Subsequent bits (bit6 to bit0) are shown in boxes with diagonal lines, indicating they are sampled during the first clock cycle.
- SDI (SMP = 1):** Data input signal. The first bit (bit7) is shown in a box with diagonal lines, indicating it is sampled during the first clock cycle. Subsequent bits (bit6 to bit0) are shown in boxes with diagonal lines, indicating they are sampled during the first clock cycle.
- Input Sample (SMP = 0):** Sampling signal. Arrows indicate sampling at the midpoint of each clock cycle.
- Input Sample (SMP = 1):** Sampling signal. Arrows indicate sampling at the midpoint of each clock cycle.

At the bottom, the **SSPIF** (Serial SPI Interface Flag) signal is shown, which is set at the end of the first clock cycle. The **SSPSR to SSPBUF** signal is also shown, indicating the transfer of the first byte of data.

The diagram also shows the **Write to SSPBUF** signal, which is active during the first clock cycle. The **Next Q4 cycle after Q2↓** is indicated by a dashed line and an arrow pointing to the next clock cycle.

**Exemple :** timing d'un CNA MAXIM 12bits MAX539, les bits sont transmis lors du front montant de SCLK (horloge du maitre, ici un P18Fxx2). Le CNA attend deux octets consécutifs avant de les placer dans son tampon de sortie (4 bits de poids fort à 0)



**SSPSTAT: MSSP STATUS REGISTER (SPI MODE)**

7	6	5	4	3	2	1	0
SMP	CKE	D/A	P	S	R/W	UA	BF

**SMP:** Sample bit (échantillonnage)

SPI Maître:

1 = La donnée en entrée est saisie à la fin du temps de la donnée en sortie

0 = La donnée en entrée est saisie au milieu du temps de la donnée en sortie

SPI Esclave: doit être mis à 0

**CKE:** SPI Clock Edge Select

Avec CKP = 0:

1 = La donnée est transmise sur le front montant de SCK

0 = La donnée est transmise sur le front descendant de SCK

Avec CKP = 1:

1 = La donnée est transmise sur le front descendant de SCK

0 = La donnée est transmise sur le front montant de SCK

**D/A:** Data/Address bit (inutilisé en mode SPI)

**P:** STOP bit (inutilisé en mode SPI)

**S:** START bit (inutilisé en mode SPI)

**R/W:** Read/Write bit information (inutilisé en mode SPI)

**UA:** Update Address (inutilisé en mode SPI)

**BF:** Buffer Full Status bit (pour le mode réception)

1 = réception terminée , SSPBUF est plein

0 = réception en cours , SSPBUF is vide

Ces 5 bits sont  
réservés au  
mode I2C

**SSPCON1: MSSP CONTROL REGISTER1 (SPI MODE)**

7	6	5	4	3	2	1	0
WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0

**WCOL:** Write Collision Detect bit (pour le mode émission seulement)

1 = Une écriture a eu lieu dans SSPBUF durant une transmission (le bit doit être effacé par logiciel)

0 = pas de collision

**SSPOV:** Receive Overflow Indicator bit

SPI Esclave:

1 = Une donnée est arrivée avant la lecture de SSPBUF (la donnée dans SSPSR est perdue) (le bit doit être effacé par logiciel)

0 = pas d'écrasement

**SSPEN:** Synchronous Serial Port Enable bit

1 = Active le port série et configure SCK, SDO, SDI, et SS comme des broches de port série SPI

*attention, les broches doivent être correctement configurée en entrée ou en sortie (TRISA et TRISC)*

0 = Désactive la fonction SPI

**CKP:** Clock Polarity Select bit

1 = au repos l'horloge est à l'état haut

0 = au repos l'horloge est à l'état bas

**SSPM3:SSPM0:** Synchronous Serial Port Mode Select bits

0101 = SPI mode esclave , horloge = SCK , /SS est désactivé et peut être utilisée en E/S

0100 = SPI mode esclave , horloge = SCK , un niveau bas sur /SS est nécessaire pour autoriser la réception

0011 = SPI mode maitre , horloge = sortie de TMR2 /2

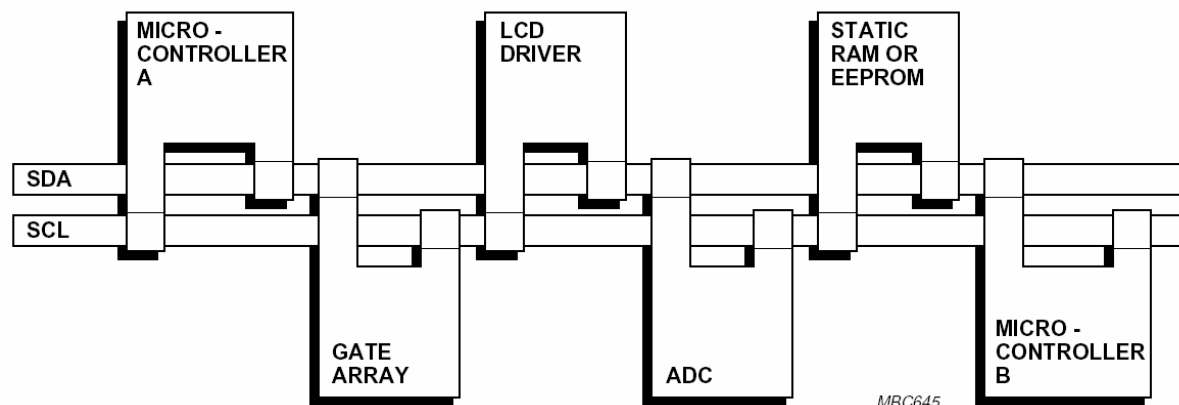
0010 = SPI mode maitre , horloge = Fosc/64

0001 = SPI mode maitre , horloge = Fosc/16

0000 = SPI mode maitre , horloge = Fosc/4



## 20. Communications séries synchrones : bus I2C (IIC)



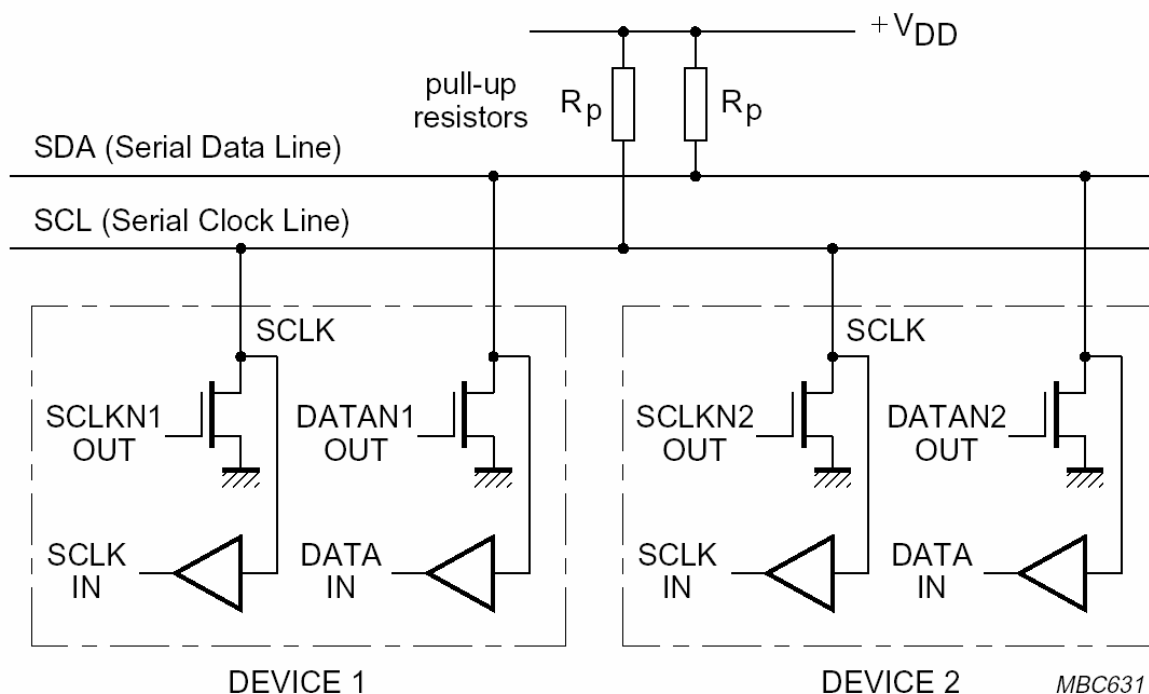
SDA : Serial Data  
SCL : Serial Clock



Les communications en mode I2C contrairement au mode SPI n'utilise que deux fils (SCL et SDA pour horloge et données), l'horloge est unique (donc synchrone) et générée par un maître. Les données transitent sur un seul fil en émission et en réception. Il peut y avoir plusieurs interfaces I2C sur un même fil de donnée avec une horloge synchrone commune. La contrepartie est la nécessité d'un protocole de communication.

Ce protocole est propriété de Philips (voir les spécifications ici : <http://www.semiconductors.philips.com/buses/i2c/>)

**Interface électrique :**

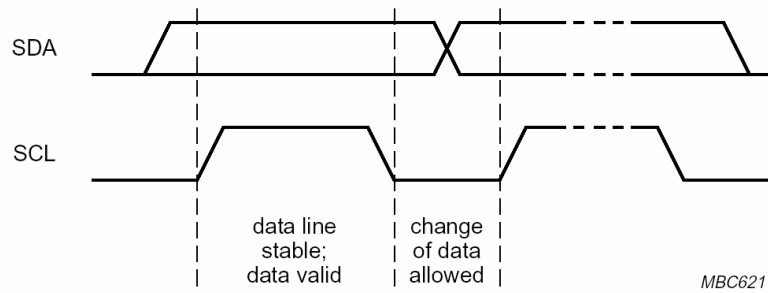


Au repos données et horloge (SDA et SCL) sont à l'état haut grâce à deux résistances de rappel (R<sub>p</sub>). Un état « actif » est donc un zéro électrique. Ce procédé permet d'éviter les courts circuits électriques au cas où deux périphériques voudraient prendre le ligne de donnée en même temps.

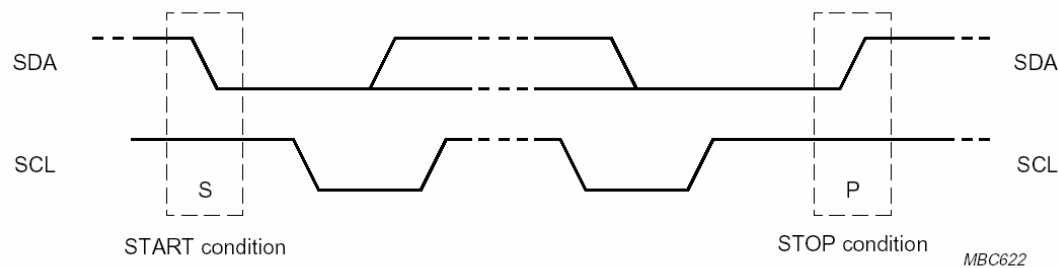


## Le protocole I2C :

### Emission d'un bit

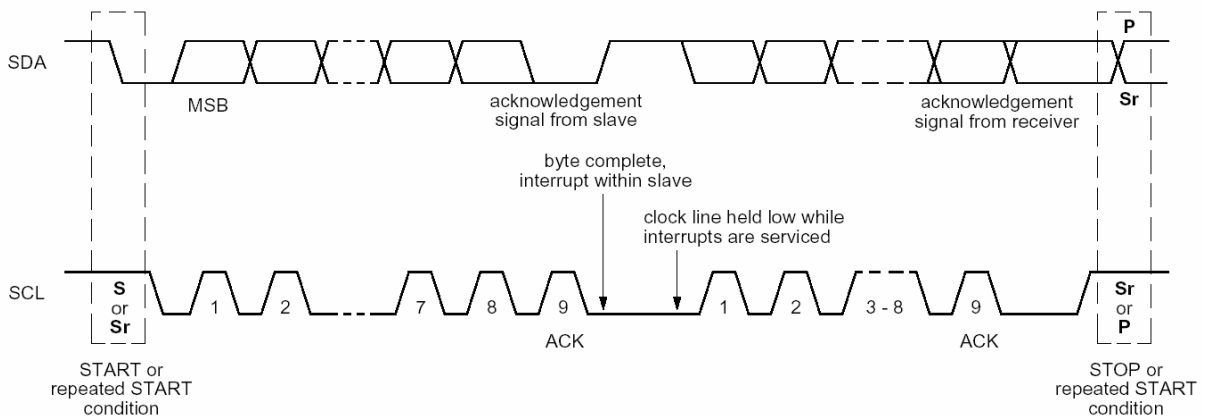


### Condition de début (start) et de fin (stop) de trame



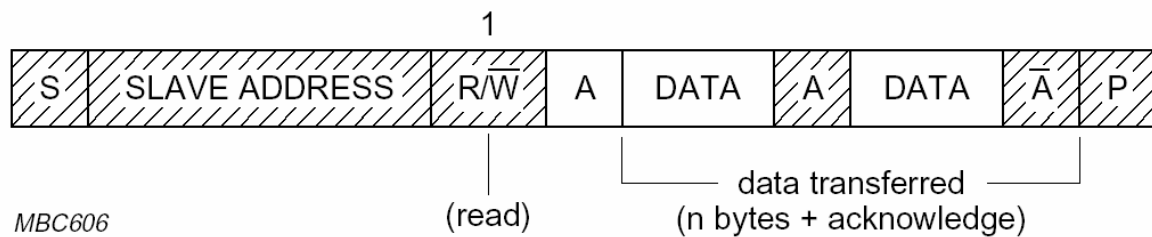
### Transmission d'une trame de donnée sur bus I2C

A la fin de la transmission de chaque octet c'est l'esclave qui place sur la ligne de donnée un 0 d'acquiescement (ack)



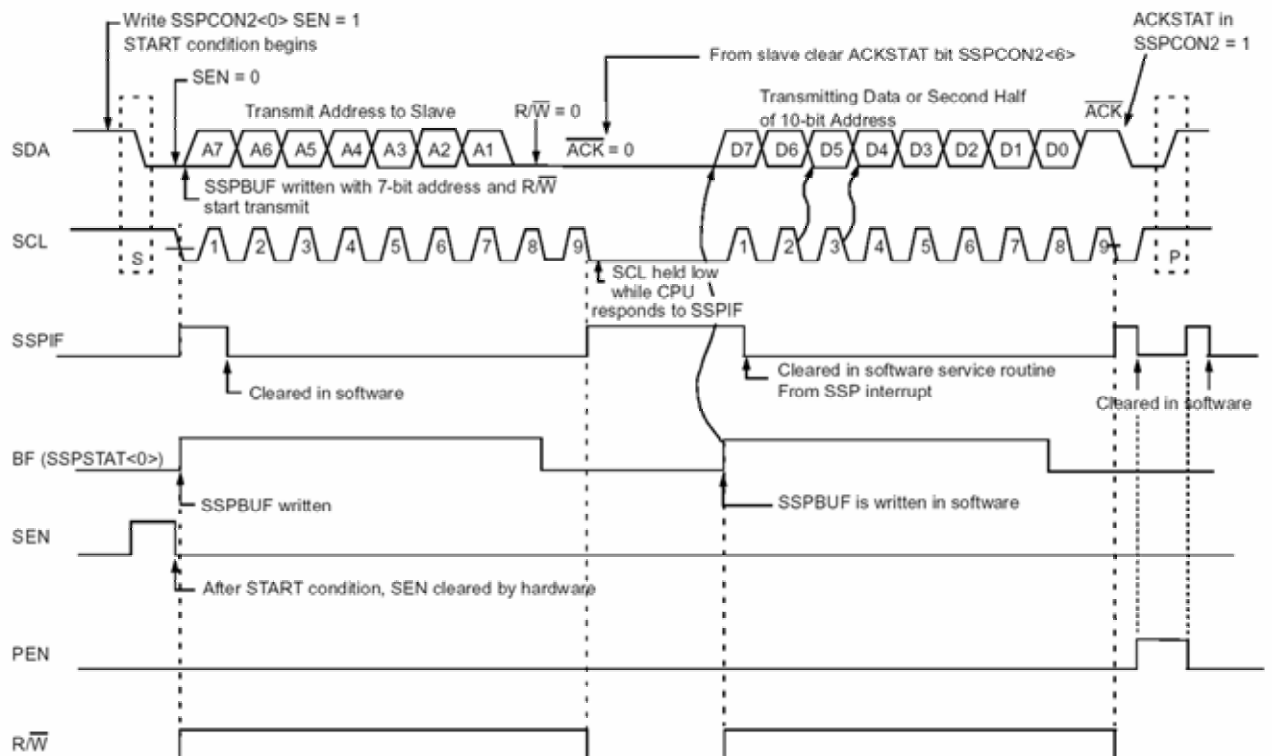
### Exemple : un maitre lit un esclave

S : start, P : stop, RW=1 pour une demande de lecture, A : ACK de l'esclave, /A : ACK du maitre

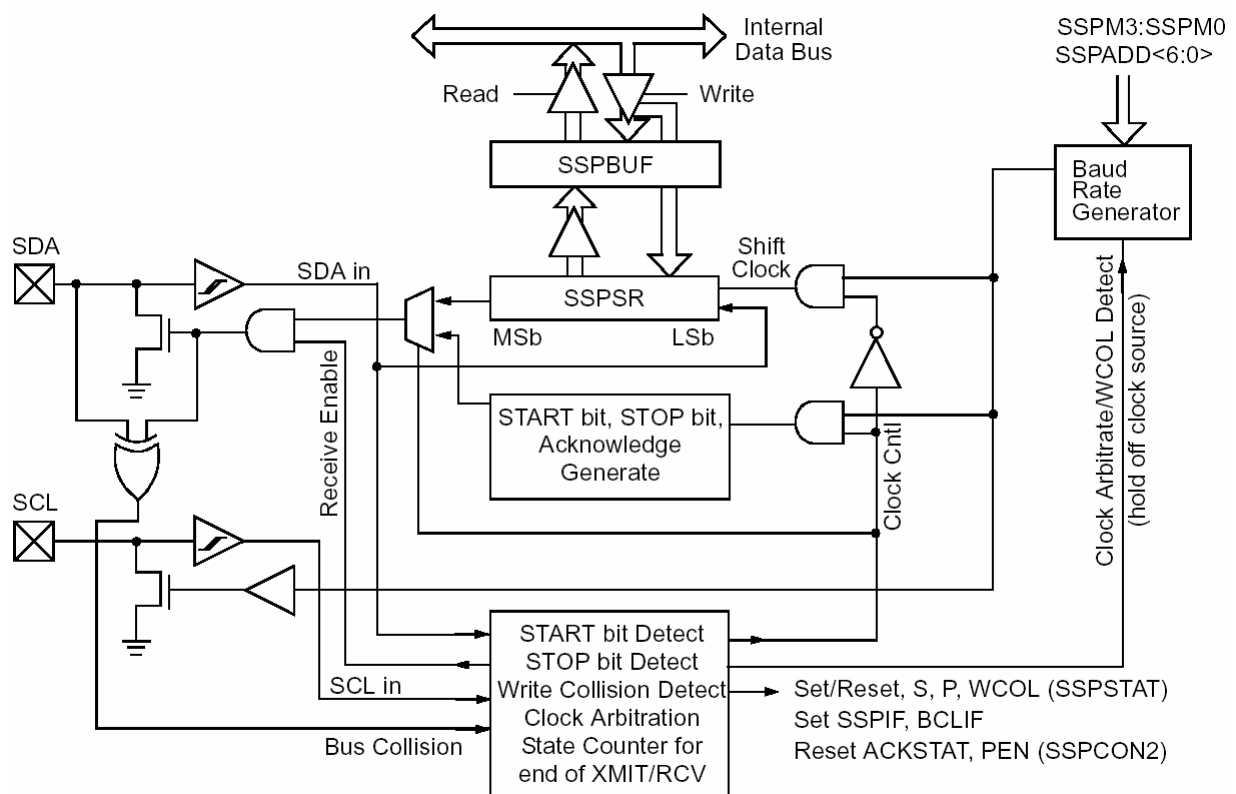


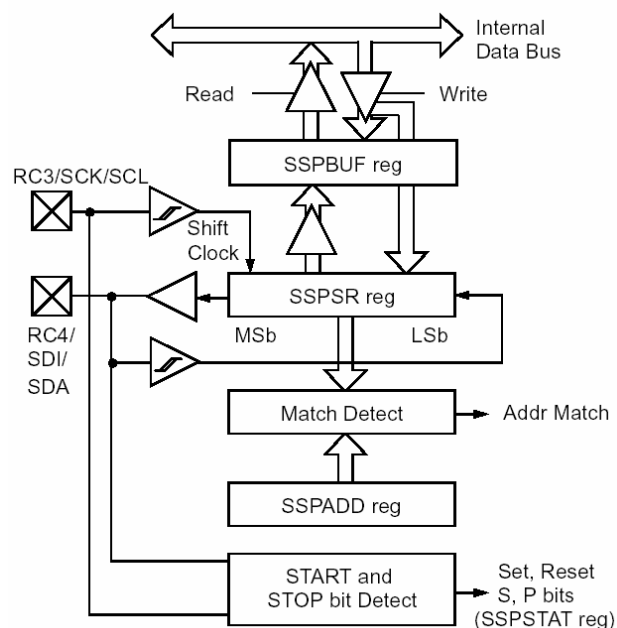


## Exemple de transmission en mode maitre avec un P18Fxx2



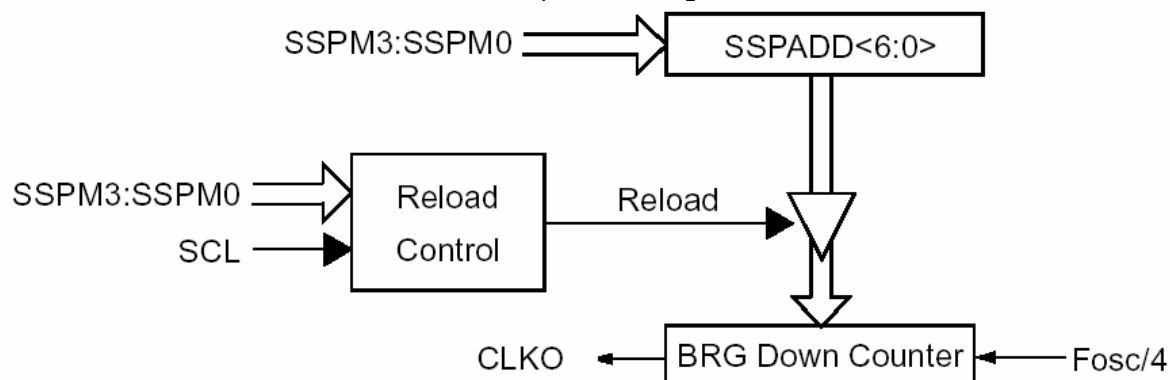
## Interface I2C en mode maitre du P18Fxx2 : structure





Le module MSSP (ci contre) utilise 6 registre pour contrôler les opération communes aux mode I2C maitre et esclave (le registre SSPSR n'est pas accessible)  
 SSPBUF contient la donnée à emttre ou la donnée reçue.  
 SSPADD contient l'adresse I2C du P18Fxx2

La vitesse de transmission en mode maitre dépende du registre BRG



FCY	FCY*2	BRG Value	FsCL <sup>(2)</sup> (2 Rollovers of BRG)
10 MHz	20 MHz	19h	400 kHz <sup>(1)</sup>
10 MHz	20 MHz	20h	312.5 kHz
10 MHz	20 MHz	3Fh	100 kHz
4 MHz	8 MHz	0Ah	400 kHz <sup>(1)</sup>
4 MHz	8 MHz	0Dh	308 kHz
4 MHz	8 MHz	28h	100 kHz
1 MHz	2 MHz	03h	333 kHz <sup>(1)</sup>
1 MHz	2 MHz	0Ah	100kHz
1 MHz	2 MHz	00h	1 MHz <sup>(1)</sup>



## Registres de contrôle en mode I2C

**SSPSTAT: MSSP STATUS REGISTER (I2C MODE)**

7	6	5	4	3	2	1	0
SMP	CKE	D/A	P	S	R/W	UA	BF

**SMP:** Slew Rate Control bit

1 = Slew rate control disabled for Standard Speed mode (100 kHz and 1 MHz)

0 = Slew rate control enabled for High Speed mode (400 kHz)

**CKE:** SMBus Select bit

1 = active le mode particulier SMBus

0 = désactivé

**D/A:** Data/Address bit

en mode esclave:

1 = indique que le dernier octet reçu était une donnée

0 = indique que le dernier octet reçu était une adresse

**P:** STOP bit

1 = indique qu'un bit de stop a été détecté

0 = pas de détection de bit de stop

**Note:** effacé par RESET ou quand SSPEN est effacé

**S:** START bit

1 = indique qu'un bit de start a été détecté

0 = pas de détection de bit de start

**Note:** effacé par RESET ou quand SSPEN est effacé

**R/W:** Read/Write bit Information

En mode esclave indique une opération de

1 = lecture

0 = écriture

En mode maître indique :

1 = une transmission en cours

0 = pas de transmission

**UA:** Update Address (pour le mode esclave avec adresse sur 10-bit)

1 = indique qu'il faut mettre à jour l'adresse dans le registre SSPADD (2 bits)

0 = rien à faire

**BF:** Buffer Full Status bit

En mode transmission

1 = Fin de réception, SSPBUF est plein

0 = Réception en cours, SSPBUF est vide

En mode réception

1 = Transmission en cours, SSPBUF est plein

0 = Transmission terminée, SSPBUF est vide

**SSPCON1: MSSP CONTROL REGISTER1 (I2C MODE)**

7	6	5	4	3	2	1	0
WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0

**WCOL:** Write Collision Detect bit

En mode maître

1 = Une écriture dans SSPBUF a eu lieu alors que les conditions de transfert n'étaient pas valide (doit être effacé par logiciel)

0 = pas de collision

En mode esclave

1 = Une écriture a eu lieu dans SSPBUF durant une transmission valide (doit être effacé par logiciel)

0 = pas de collision

**SSPOV:** Receive Overflow Indicator bit

En mode réception

1 = Un octet a été reçu alors que SSPBUF était plein (doit être effacé par logiciel)

0 = pas de débordement

**SSPEN:** Synchronous Serial Port Enable bit

1 = Active le port série et configure les broches SDA et SCL

0 = Désactive le port série I2C

**Note :** SCL et SDA doivent être correctement configurées en entrée ou en sortie

**CKP:** SCK Release Control bit

En mode esclave

1 = horloge libre

0 = Horloge bloquée à l'état bas

**SSPM3:SSPM0:** Synchronous Serial Port Mode Select bits



1111 = I2C mode esclave, adresse sur 10-bit avec interruptions sur START et STOP  
1110 = I2C mode esclave, adresse sur 7-bit avec interruptions sur START et STOP  
1011 = I2C Firmware Controlled Master mode (Slave IDLE)  
1000 = I2C mode maitre, horloge =  $F_{osc} / (4 * (SSPADD+1))$   
0111 = I2C mode esclave, adresse sur 10-bit  
0110 = I2C mode esclave, adresse sur 7-bit

**SSPCON2: MSSP CONTROL REGISTER 2 (I2C MODE)**

7	6	5	4	3	2	1	0
GCEN	ACKSTAT	ACKDT	ACKEN	RCEN	PEN	RSEN	SEN

**GCEN:** General Call Enable bit (seulement en mode esclave)

1 = Autorise l'interruption quand une adresse d'appelle (0x0000) est reçue dans SSPSR  
0 = interdit l'interruption

**ACKSTAT:** Acknowledge Status bit (pour les transmissions en mode maitre)

1 = pas d'acquiescement reçu de l'esclave  
0 = acquiescement reçu de l'esclave

**ACKDT:** Acknowledge Data bit (pour les réceptions en mode maitre)

1 = pas d'acquiescement  
0 = acquiescement

**ACKEN:** Acknowledge Sequence Enable bit (pour les transmissions en mode maitre)

1 = envoie un acquiescement

Effacement automatique

0 = pas d'acquiescement

**RCEN:** Receive Enable bit (mode maitre)

1 = active le mode réception en I2C  
0 = réception désactivée

**PEN:** STOP Condition Enable bit (mode maitre)

1 = effectue une condition stop sur SDA et SCL pins. Effacement automatique  
0 = pas de stop

**RSEN:** Repeated START Condition Enabled bit (mode maitre)

1 = effectue une condition start sur SDA et SCL pins. Effacement automatique  
0 = pas de start

**SEN:** START Condition Enabled/Stretch Enabled bit

En mode maitre

1 = effectue une condition start sur SDA et SCL pins. Effacement automatique  
0 = pas de start

En mode esclave

1 = Clock stretching activée pour les modes transmission et réception  
0 = Clock stretching activée pour le mode transmission uniquement (Legacy mode)





