

PIC18Fxx2



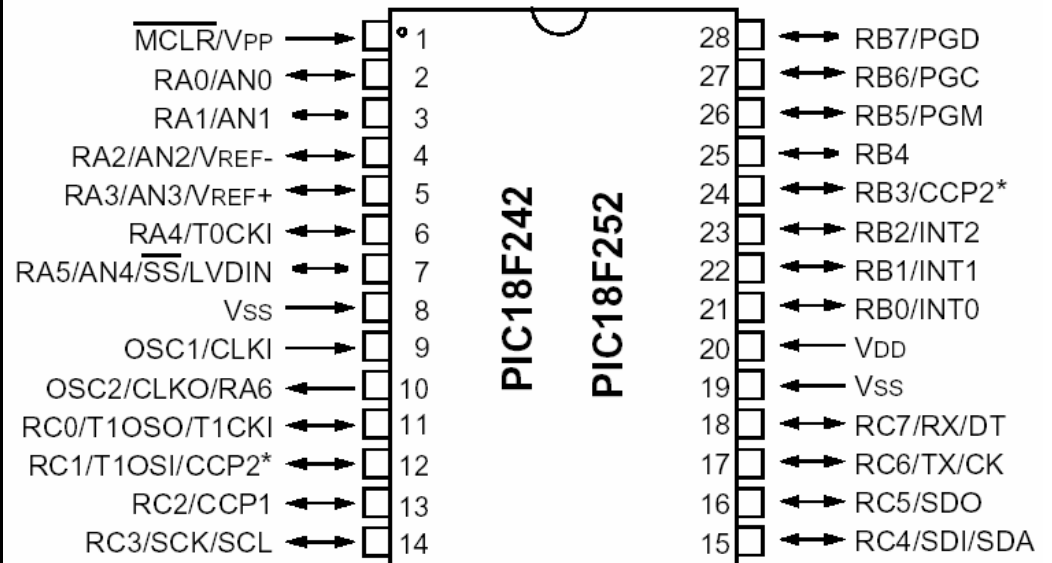
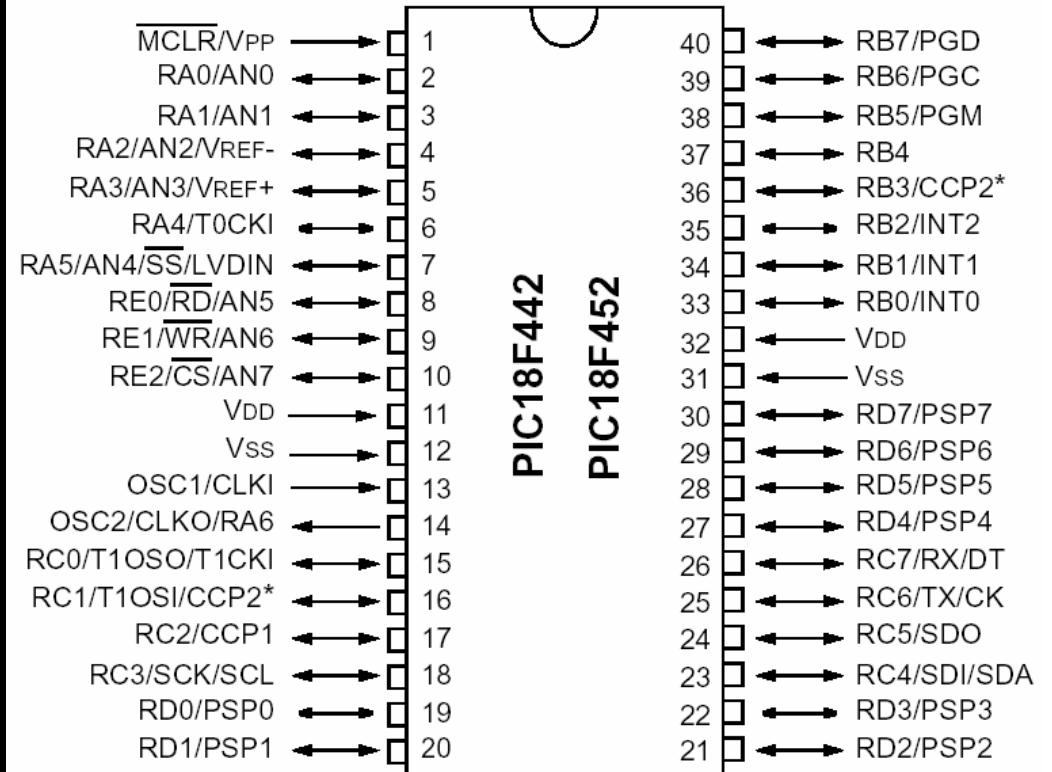
Christian Dupaty
Lycée Fourcade 13120 Gardanne

c.dupaty@aix-mrs.iufm.fr

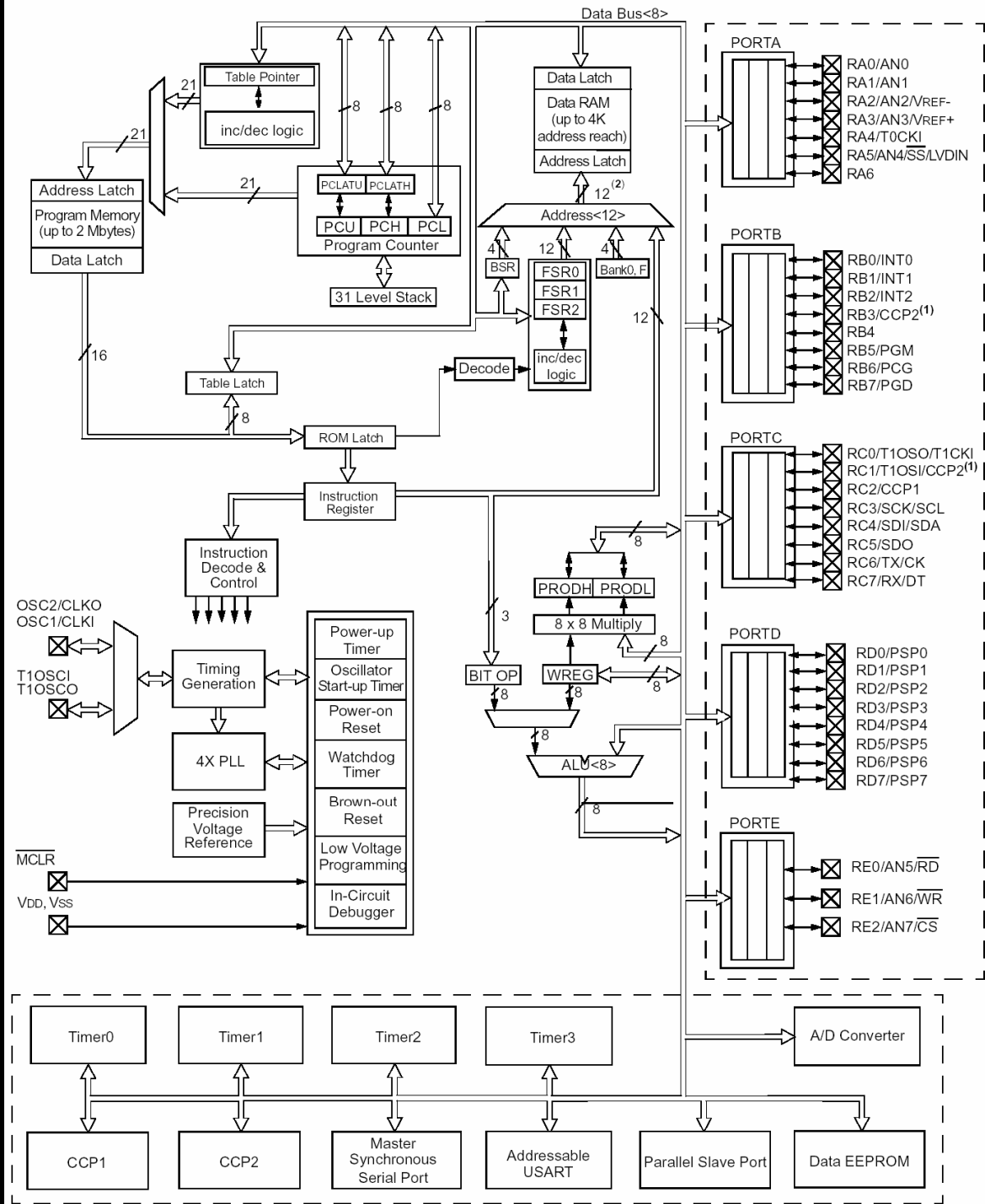
CONSTITUTION

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
Boitiers	28-pin DIP 28-pin SOIC	40-pin DIP 44-pin PLCC 44-pin TQFP

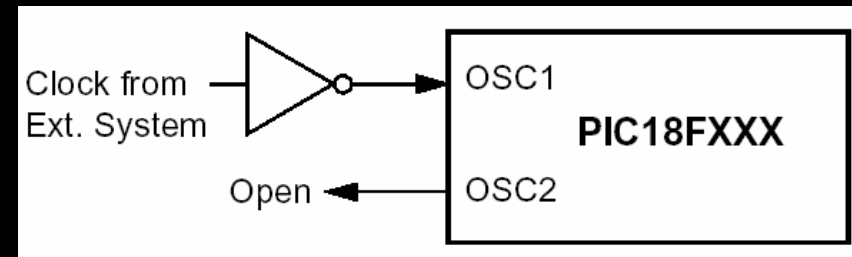
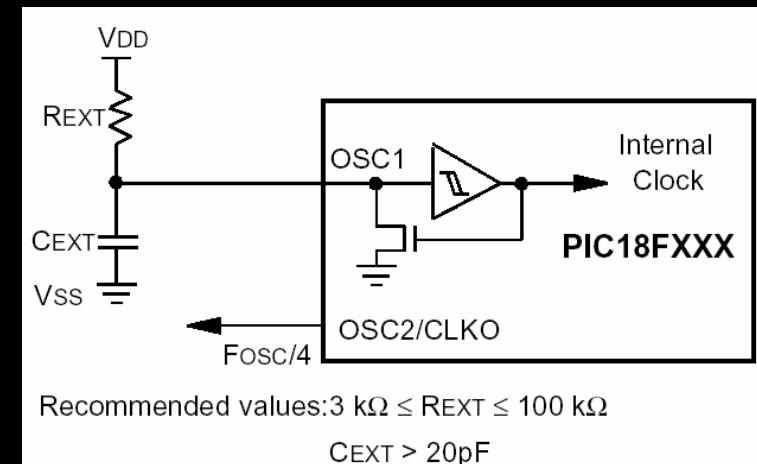
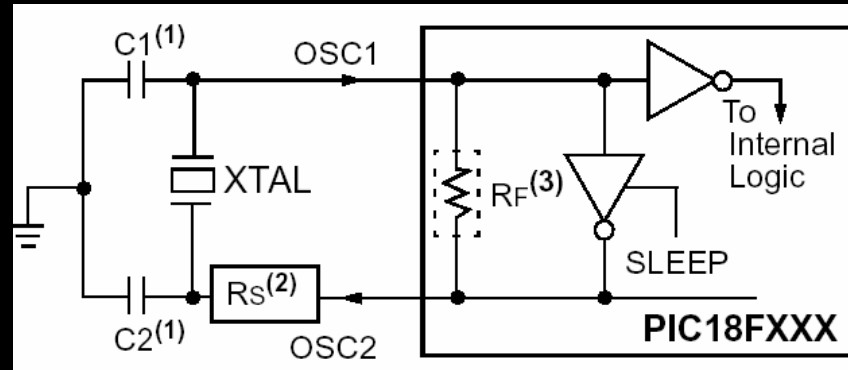
Brochages



Architecture

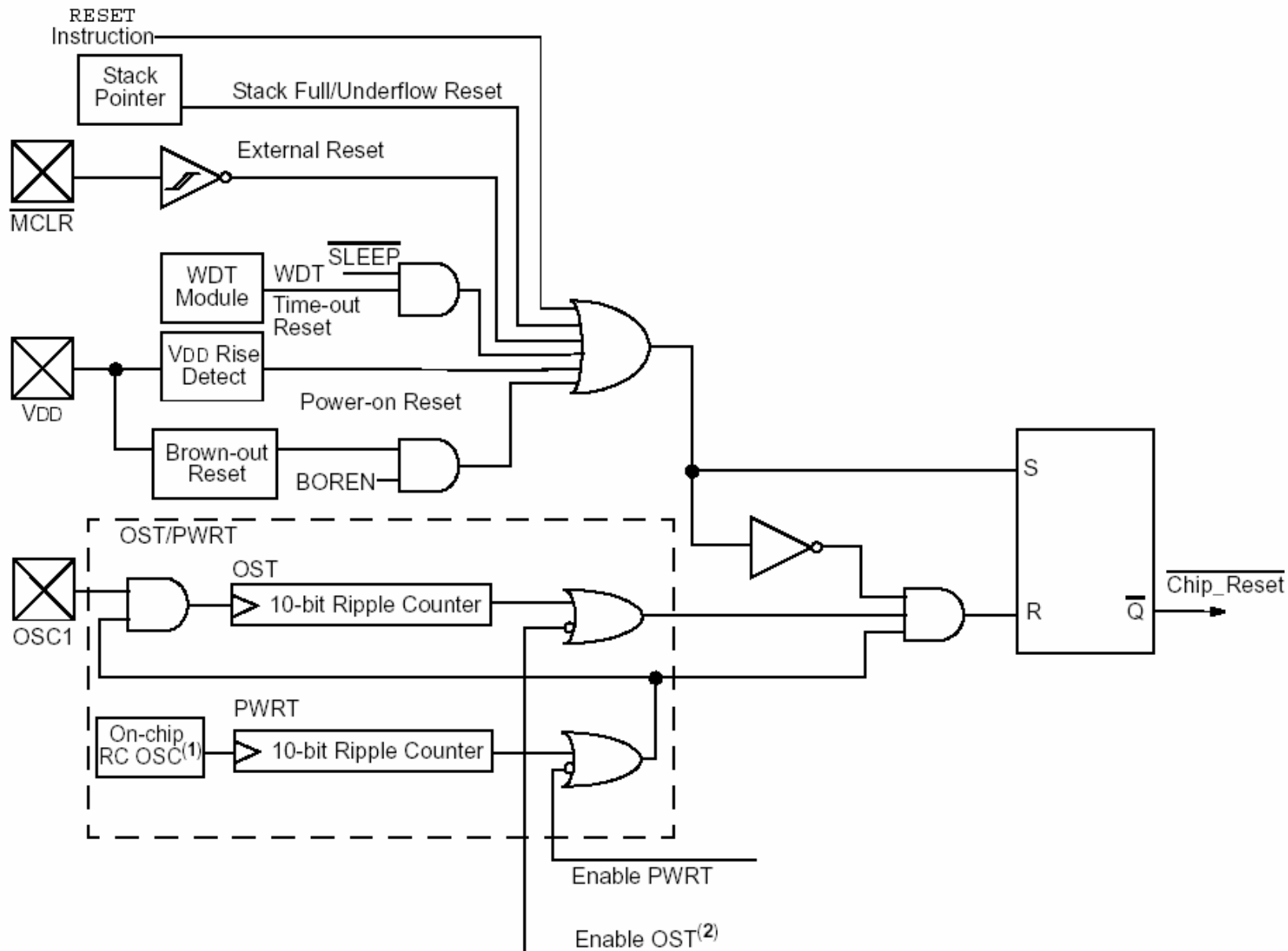


Horloges



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
		Brown Out Voltage	EC-OSC2 as Clock Out
300003	FE	Watchdog Timer	RC
		Watchdog Postscaler	HS
300005	FF	CCP2 Mux	XT
300006	7B	Low Voltage Program	LP
		Background Debug	Enabled

RESET (sources)



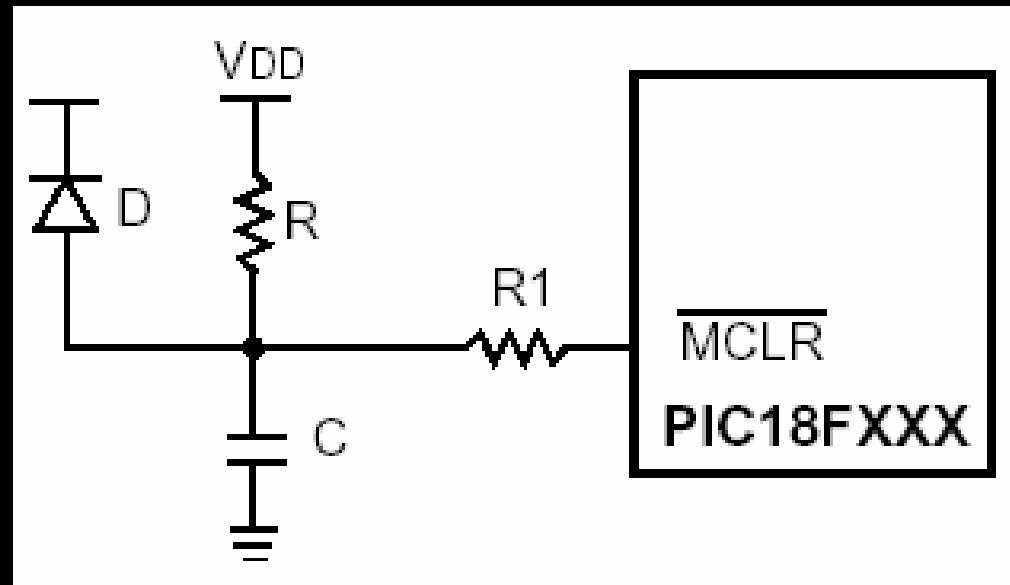
RESET - origines

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

RESET - circuit

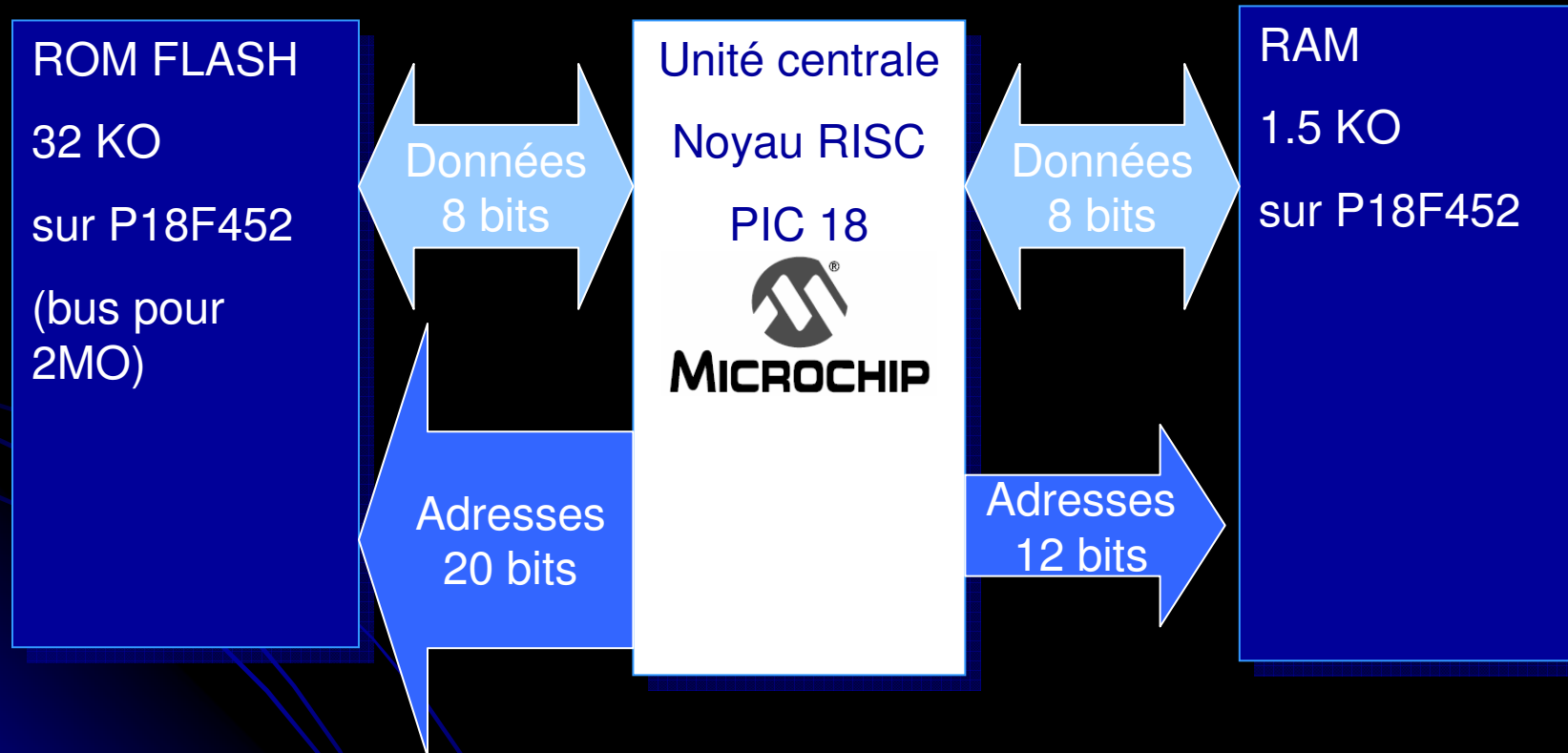


D permet une décharge plus rapide de C lorsque VDD descend.

$R < 40\text{KO}$, $C = 0.1\mu\text{F}$

$100 < R1 < 1\text{KO}$, R1 limite I dans /MCLR en cas de décharge électrostatique (ESD)

Architecture HARVARD



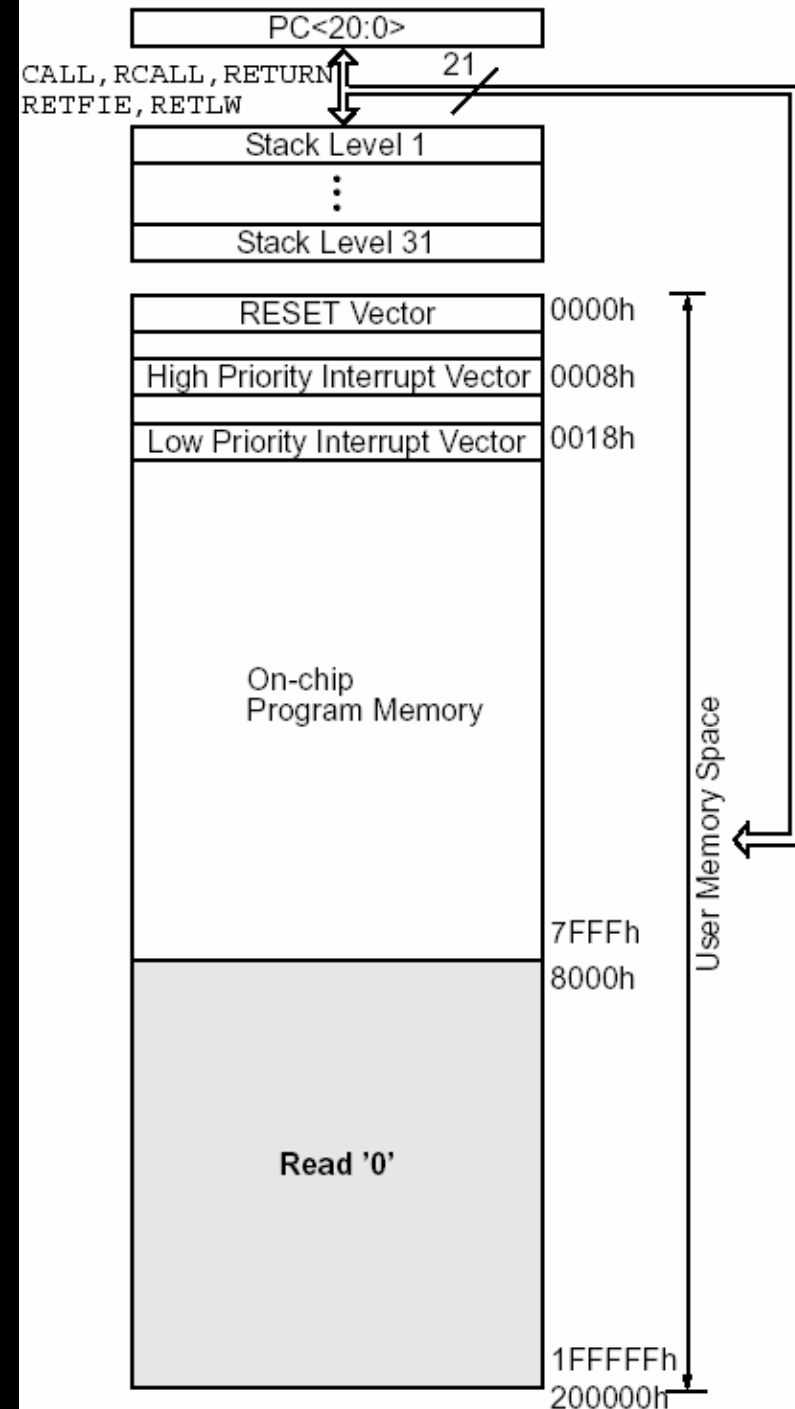
Organisation mémoire

L'espace mémoire va de 0x0000 à 0x200000 (soit 2MO).

Sur les PIC18Fx52 seuls 32 KO sont implantés.

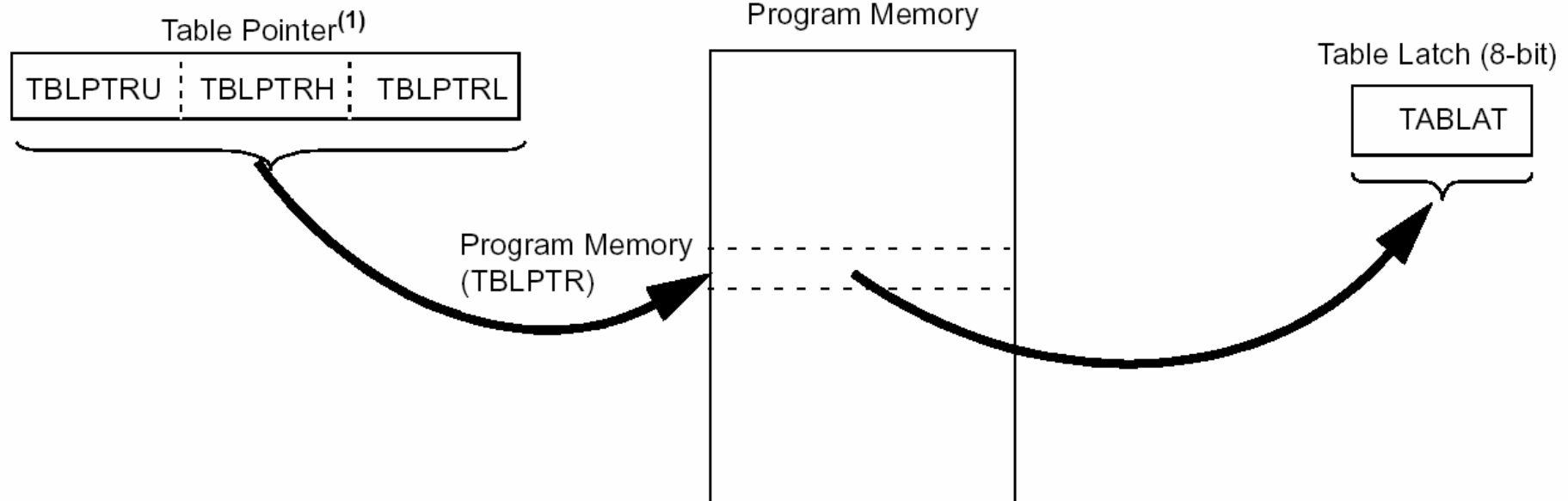
Il existe trois adresses d'interruption : RESET, HPI et LPI.

La taille de la pile n'est pas modifiable, elle contient 31 niveaux

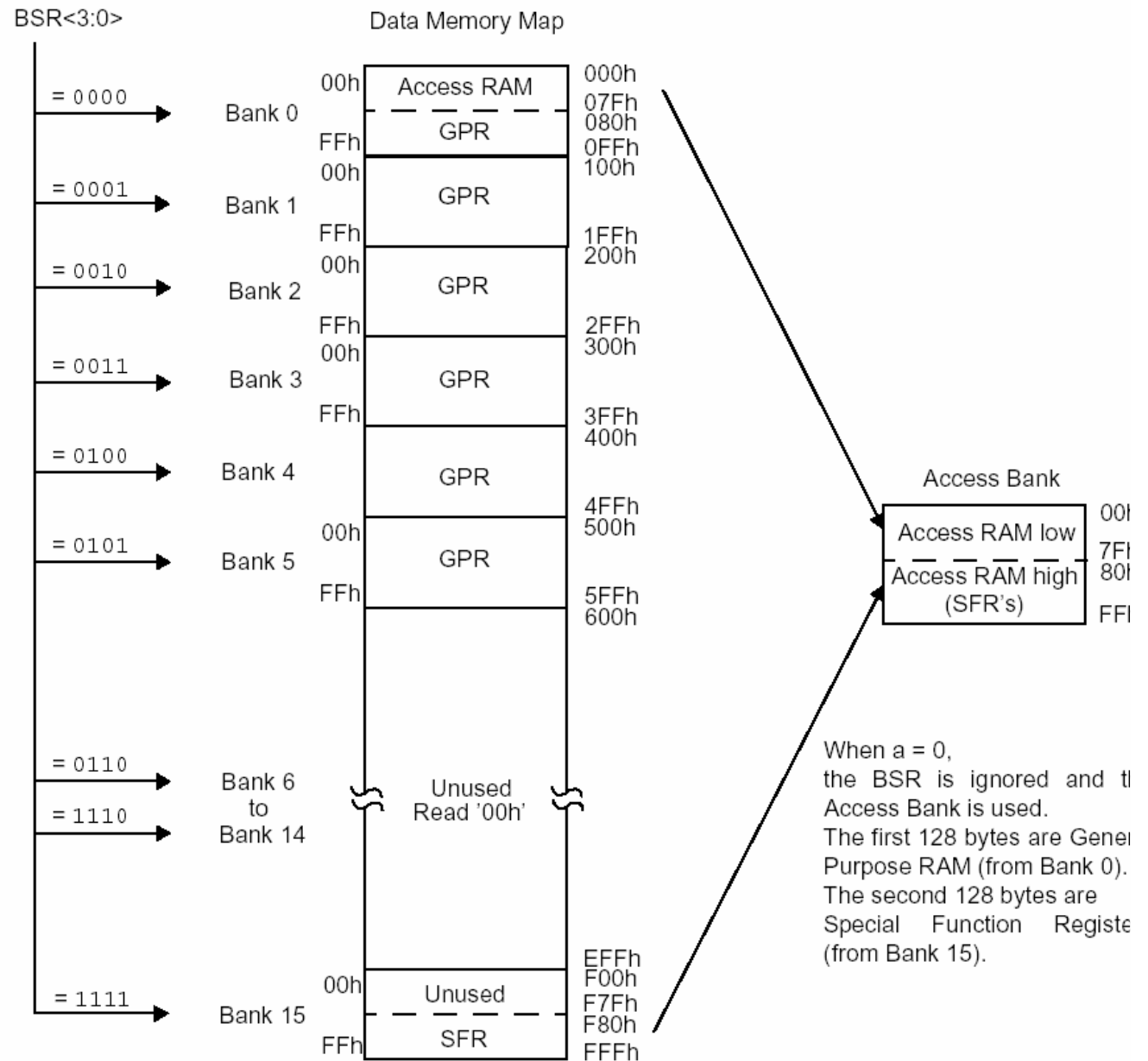


Lecture FLASH

Instruction: TBLRD*

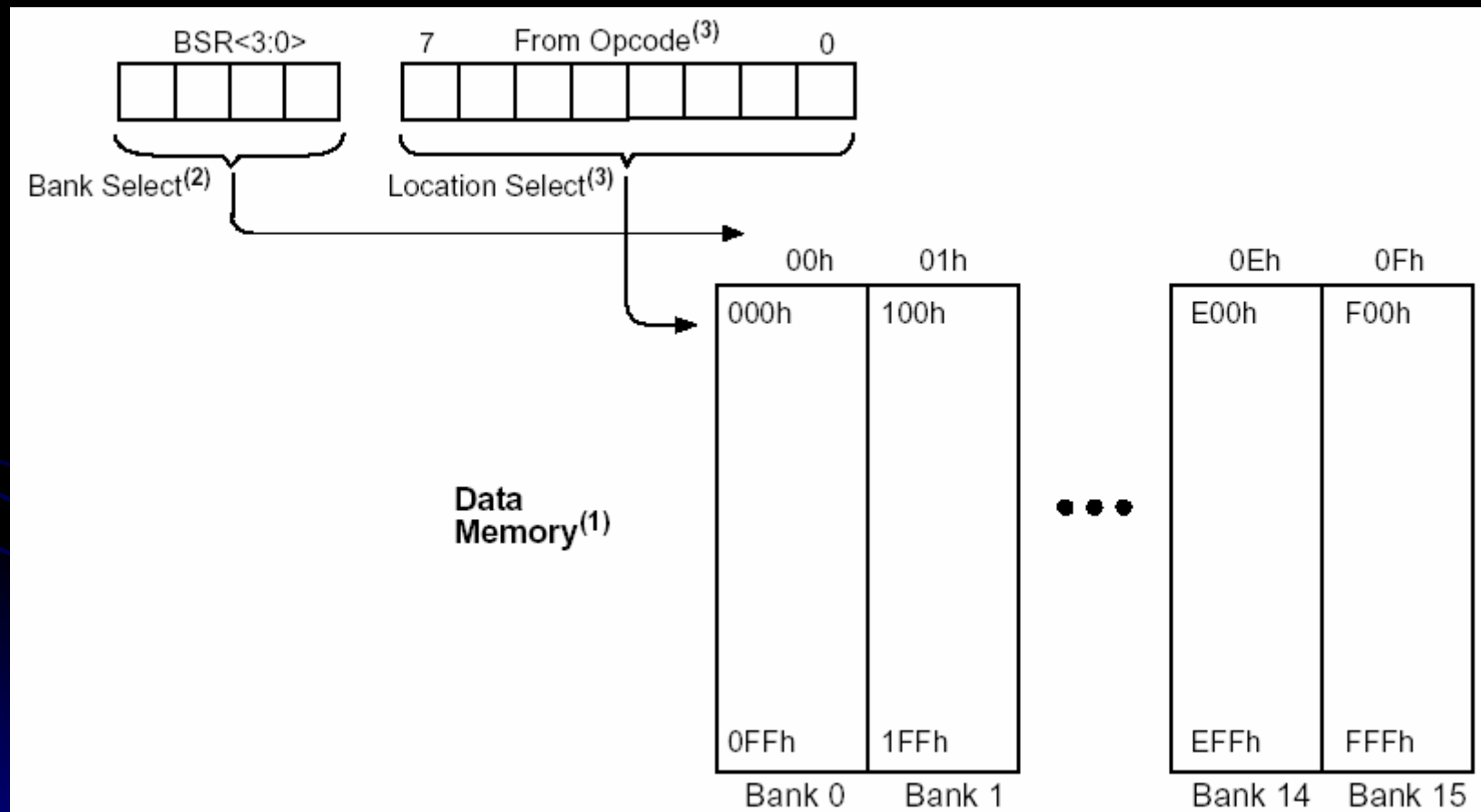


P18F452 - RAM

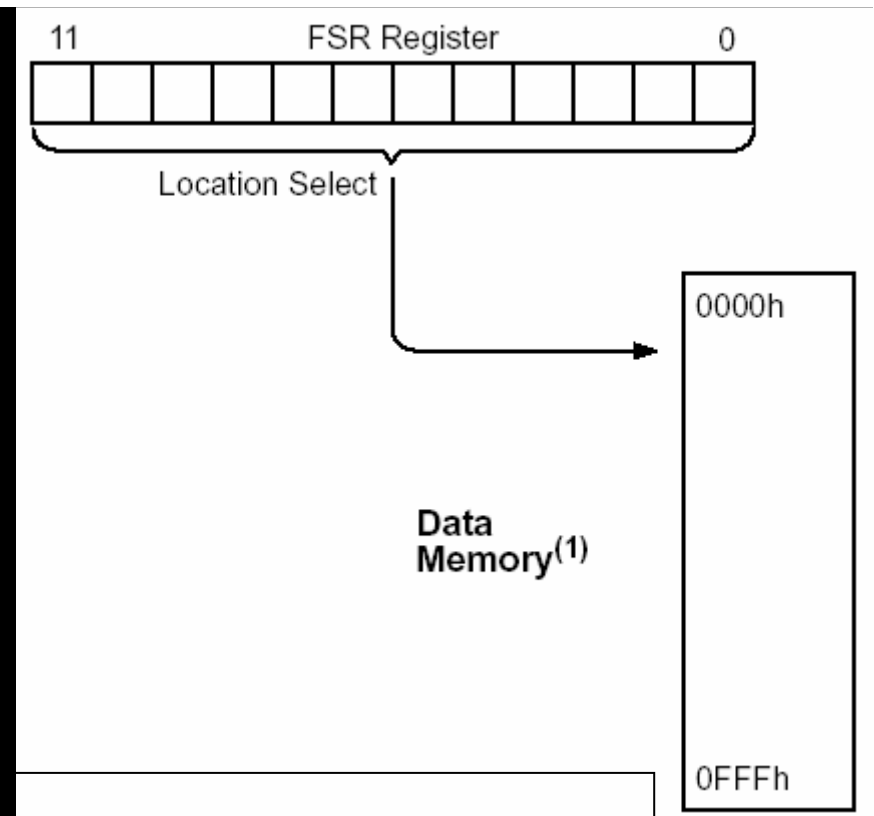


When a = 1, the BSR is used to specify the RAM location that the instruction uses.

RAM – address direct



RAM – adressage indirect



Exemple : Mise à 0 de la BANK 1

```

NEXT    LFSR FSR0 ,0x100    ;FSR0 pointe sur BANK1
        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           ;
    
```

A s s e m b l e u r

```
;*****
;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 défintion 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*1us = 4ms.
        org      0x0008          ; adresse d'interruption prioritaire
        decfsz   compteur,f      ; décrémente 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*1us = 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 = 500ms
        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
```

STATUS

Le registre STATUS (0xFD8)

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

N si négatif

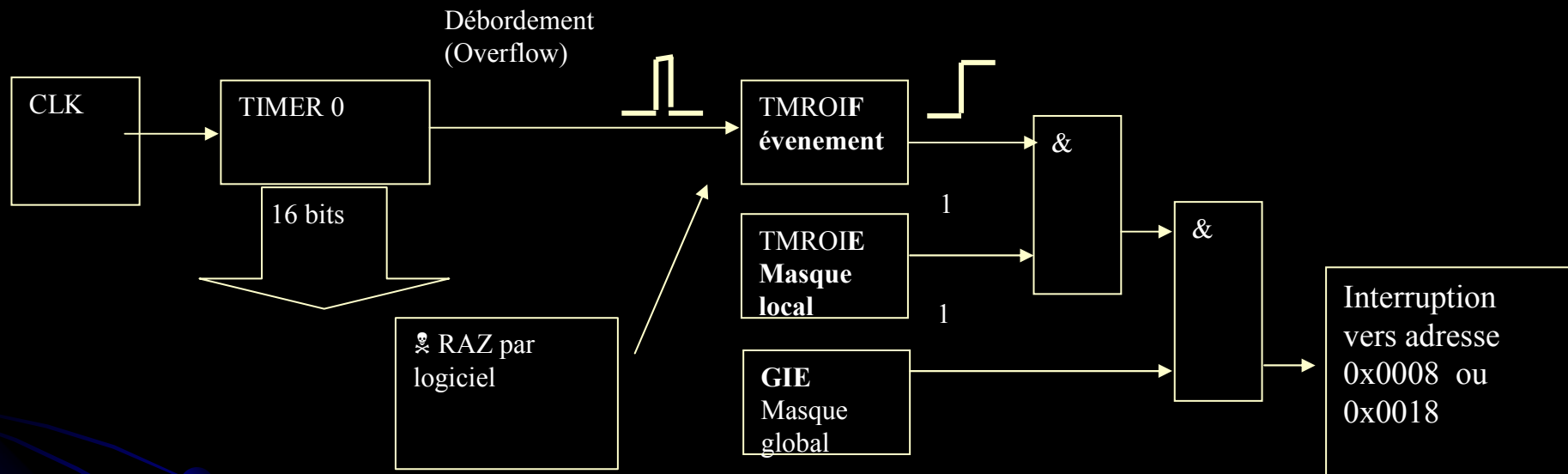
OV s'il y a eu un débordement dans une opération en complément à 2

Z : si le résultat est nul

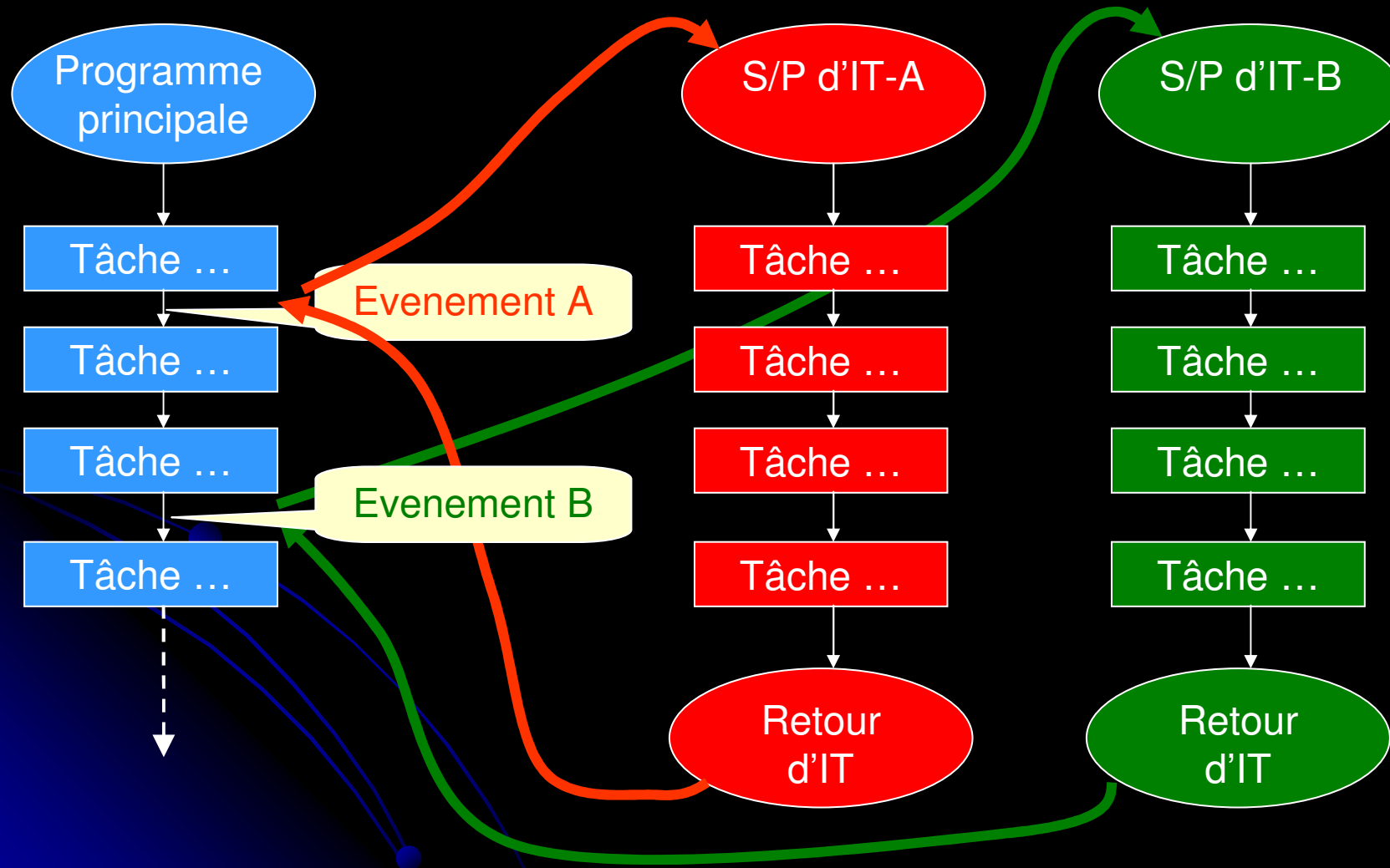
DC : demi retenue (le bit4 est passé à 1)

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

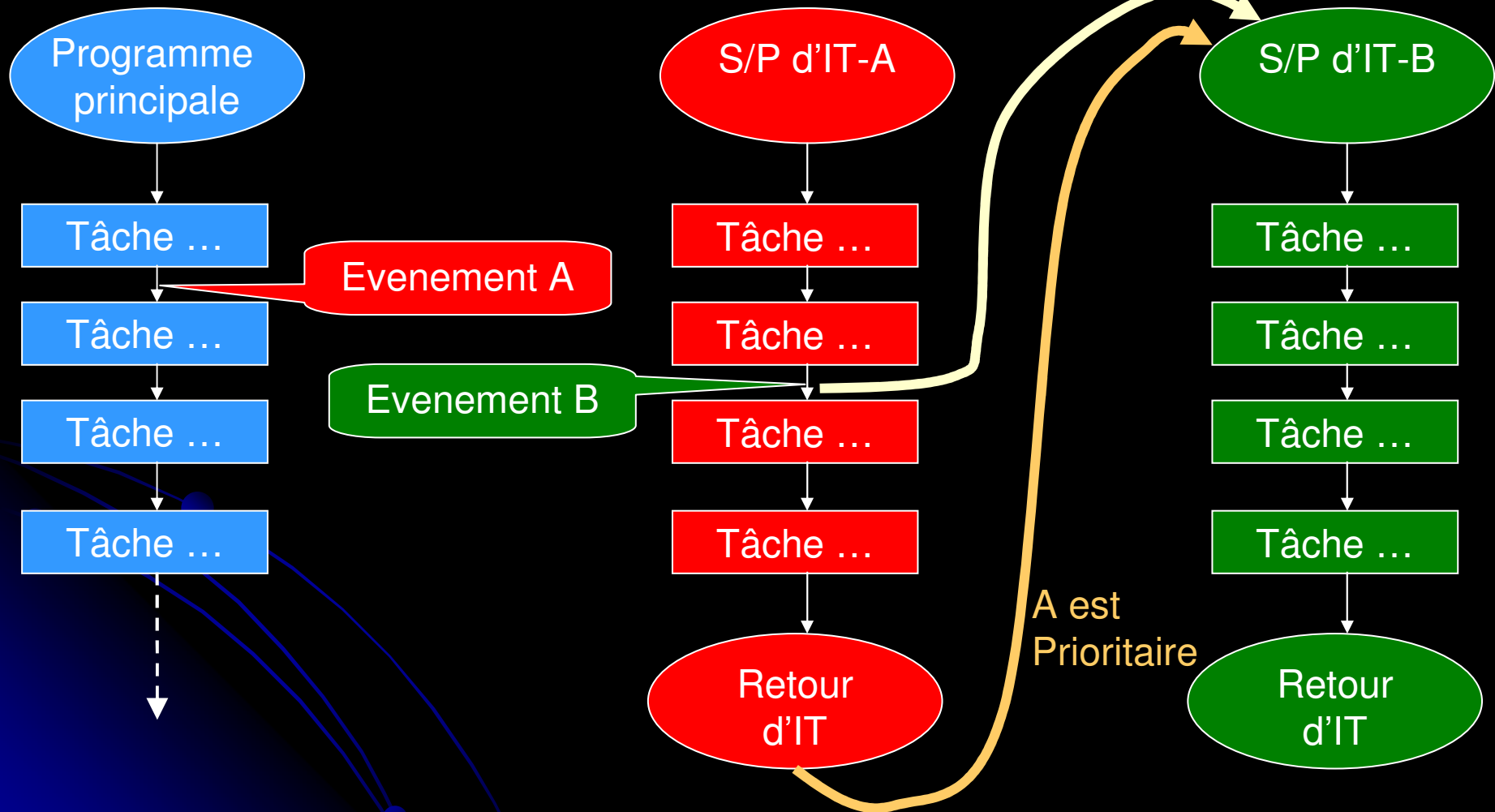
Interruptions- principes



Interruptions – Priorités (1)



Interruptions – Priorités (2)



Choix des priorités RCON et INTCON

REGISTRE RCON : IPEN : interrupt priority enable. Cette fonction peut être désactivée pour avoir une compatibilité logicielle avec l'unité centrale PIC16.

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

Registre INTCON:

GEIH, GEIL de INTCON si IPEN=1

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

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

7	6	5	4	3	2	1	0
GEIH	GEIL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF

Choix des priorités IPIR1 & IPIR2

Ces deux registres permettent de choisir pour chaque périphérique si sa source d'interruption sera prioritaire ou non (avec IPEN=1)

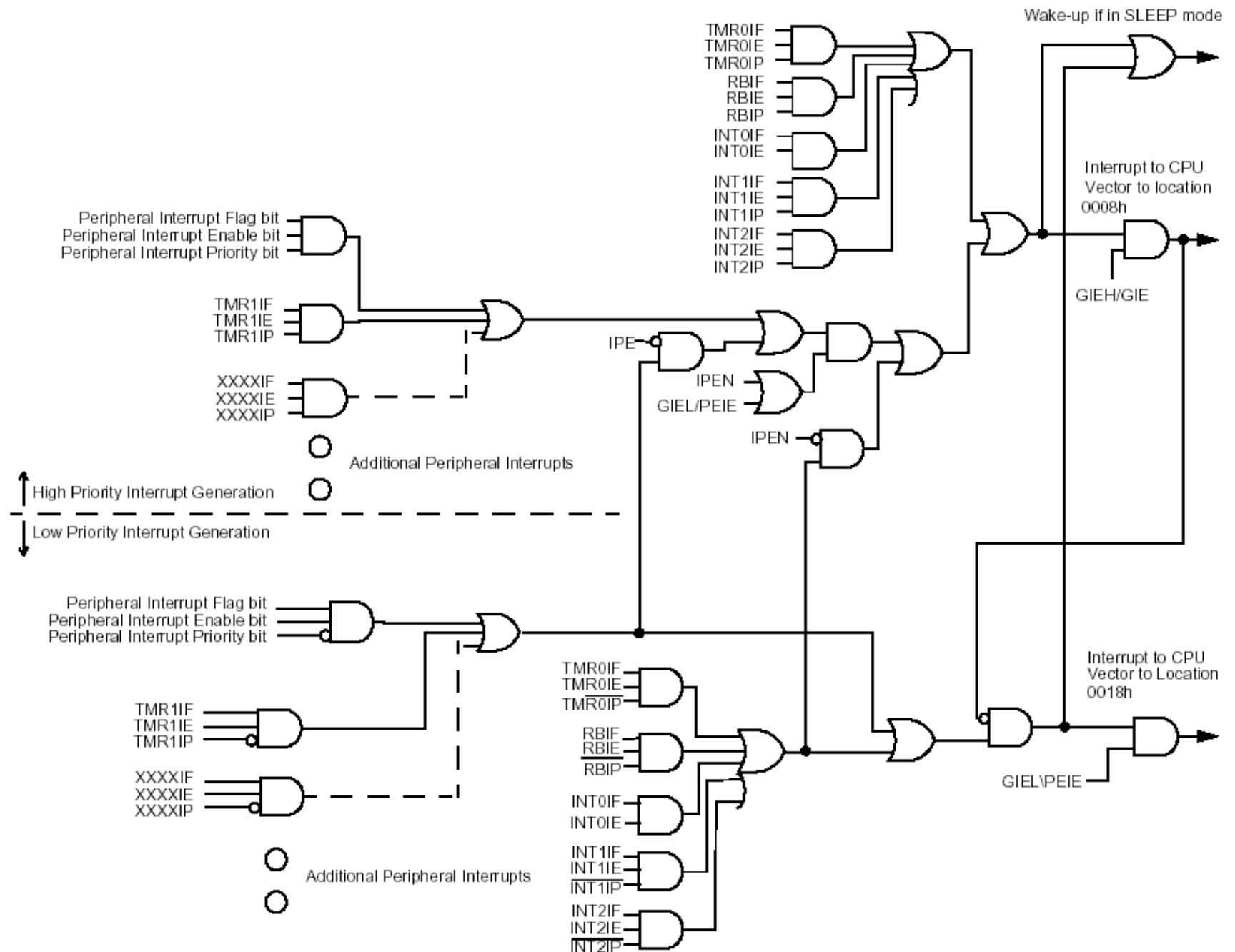
Ex : registre IPR1 priorité haute si le bit =1

7	6	5	4	3	2	1	0
PSPiP	ADiP	RCiP	TXiP	SSPiP	CCP1iP	TMR2iP	TMR1iP

Adresses des SP d'IT

- Il n'y a que 16 instructions insérables entre les adresses 0x0008 et 0x0018.
- Il faudra pour les interruptions prioritaires (0x0008) placer à cette adresse un saut vers la SP d'IT. (goto)
- Pour les interruptions non prioritaires le SP d'IT pourra être placé à l'adresse 0x0018

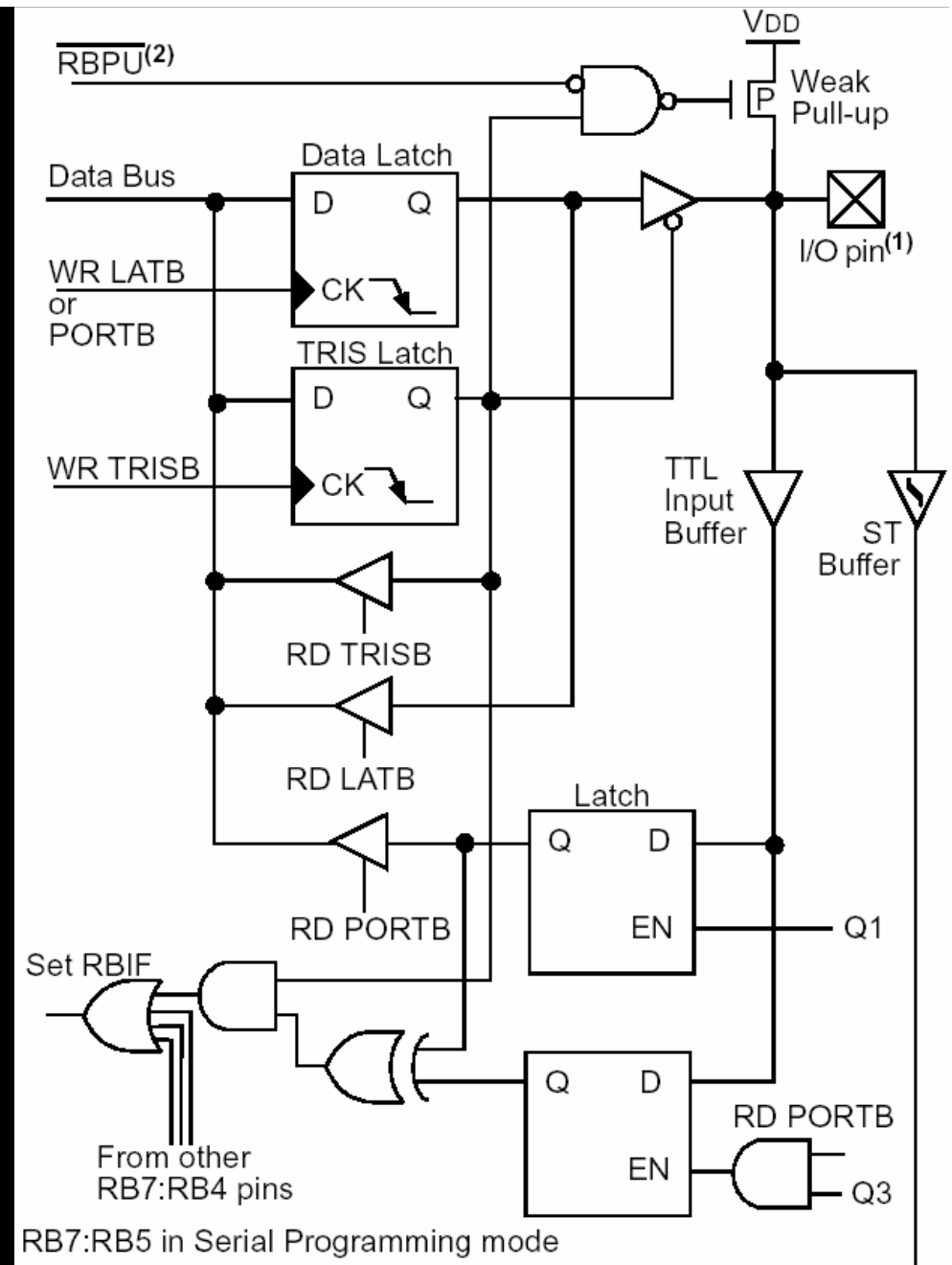
Interruptions - généralités



Ports parallèles (PRB)

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

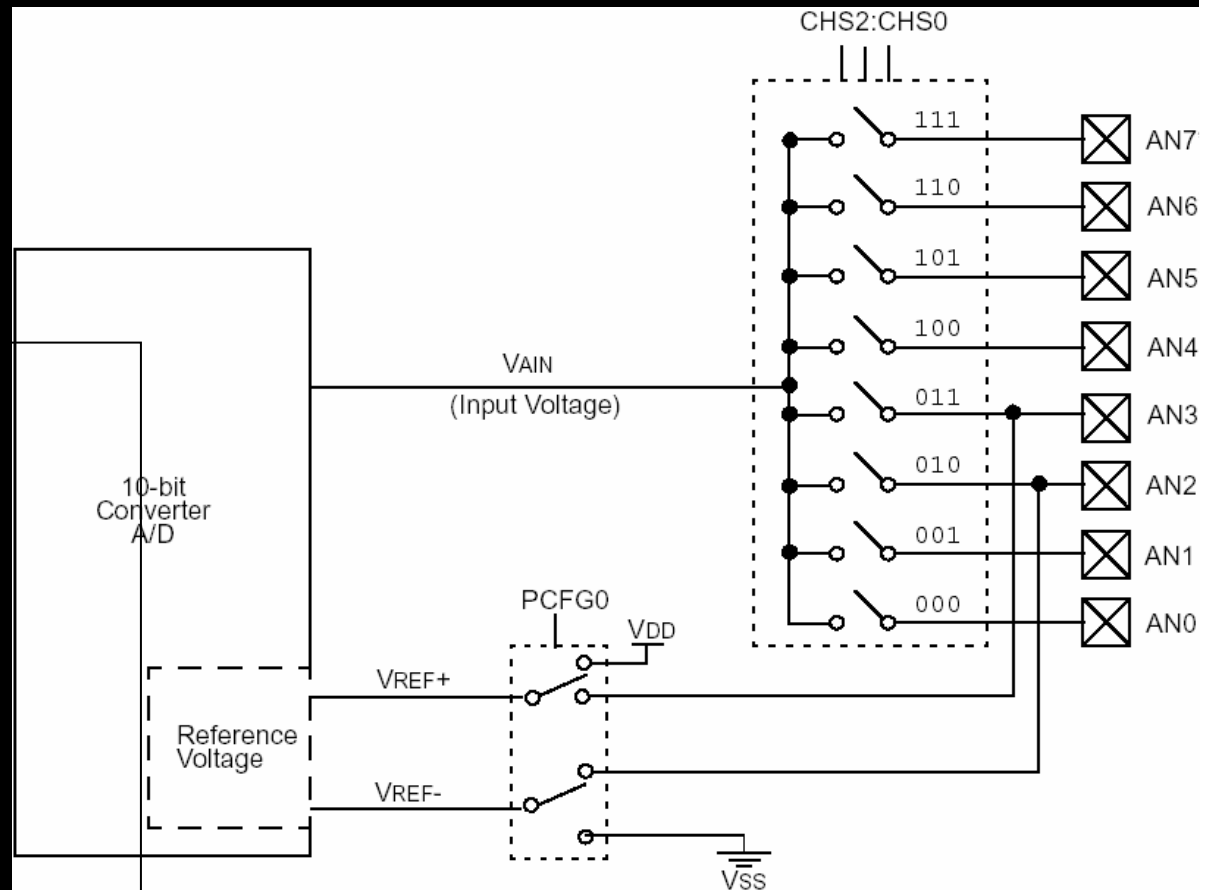
```
    a=PORTB ;
    PORTB=b ;
    While(1) ;
}
```



ADC

```
// Demo pour ADC
#include <p18f452.h>
#define q 4.8828e-3// quantum

void main(void)
{
    float res;
    // CAN on. CLOCK=FOSC/2. CANAL0
    // 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

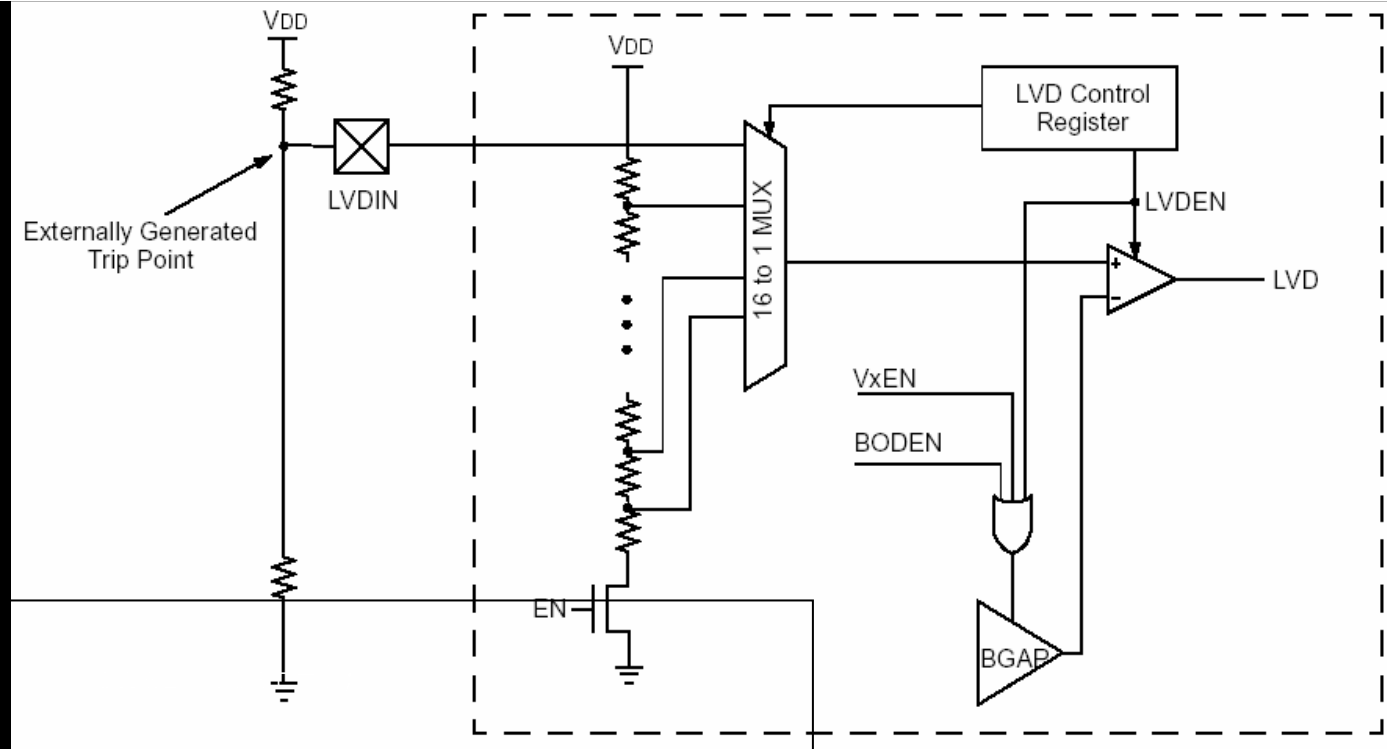
7	6	5	4	3	2	1	0
ADCS1	ADCS0	CHS2	CHS1	CHS0	GO/DONE	—	ADON

REGISTRE ADCON1

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

LVD

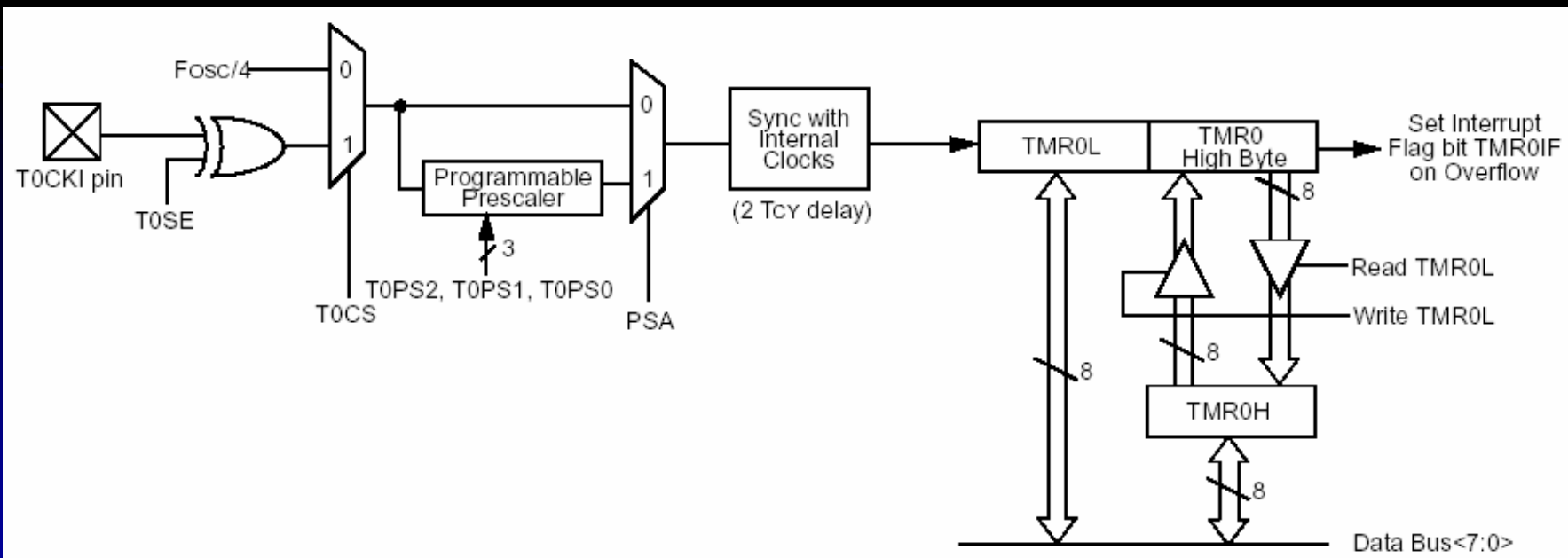
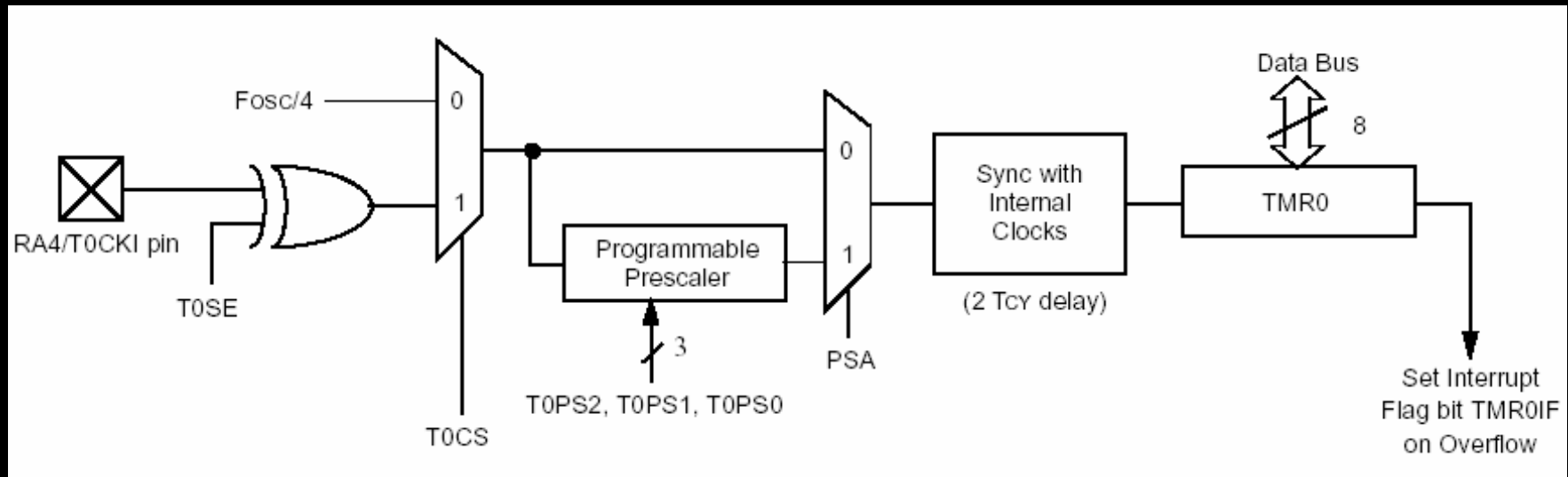
```
// Demo pour LVD
#include <p18f452.h>
void main(void)
{
    // LVD active, pas d'IT
    // detection VDD<3.5v
    PIE2bits.LVDIE=0;
    LVDCON=0b00011001;
    While(!LVDCONbits.IRVST); //attend...
        while(1){
            if (PIR2bits.LVDIF)
            {
                PIR2bits.LVDIF=0
  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



7	6	5	4	3	2	1	0
—	—	IRVST	LV DEN	LV DL3	LV DL2	LV DL1	LV DL0

REGISTRE **LVDCON** (0xFD2)

TIMER0



TIMER0 - exemple

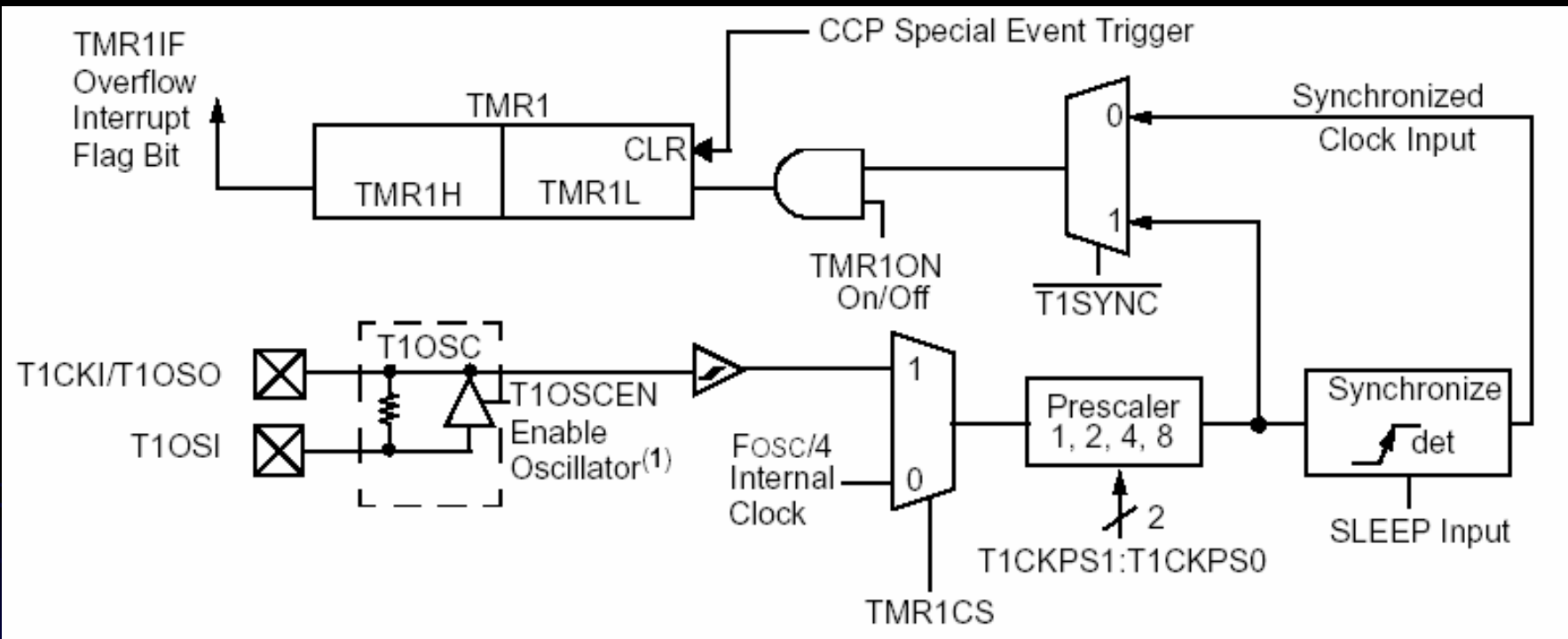
REGISTRE T0CON

7	6	5	4	3	2	1	0
TMR OON	T08B IT	T0CS	T0SE	PSA	T0PS 2	T0PS 1	T0PS 0

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

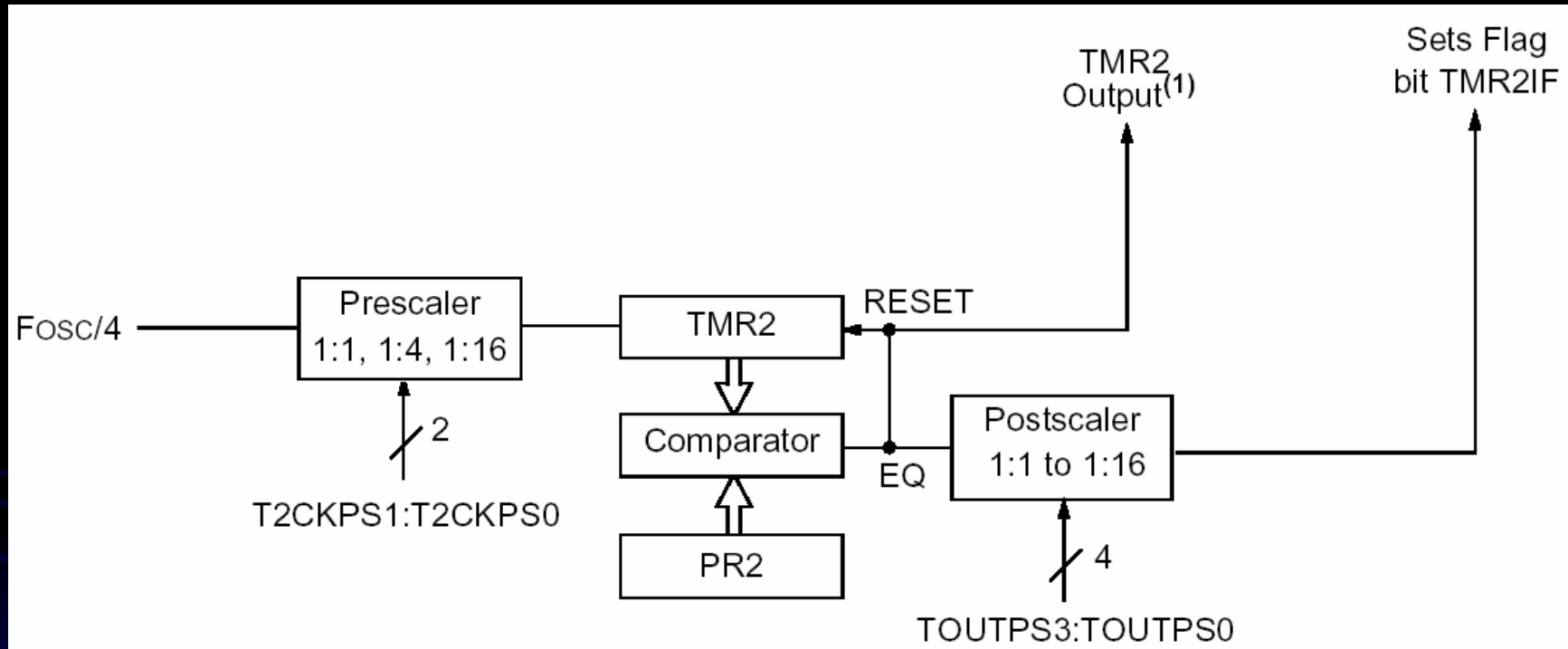
TIMER1



REGISTRE T1CON

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

TIMER2



REGISTRE TCON2

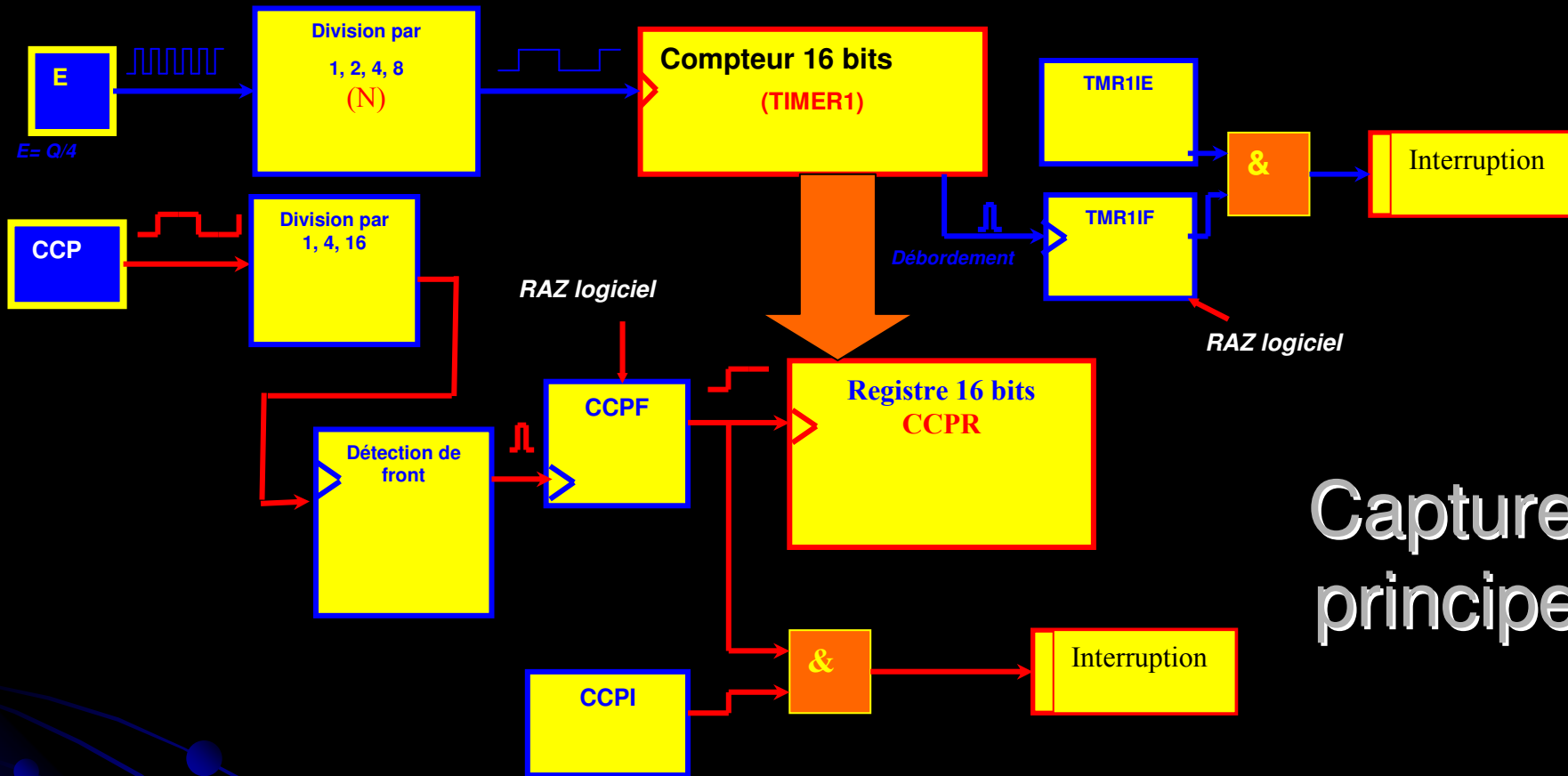
7	6	5	4	3	2	1	0
—	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0

TIMER2 - exemple

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

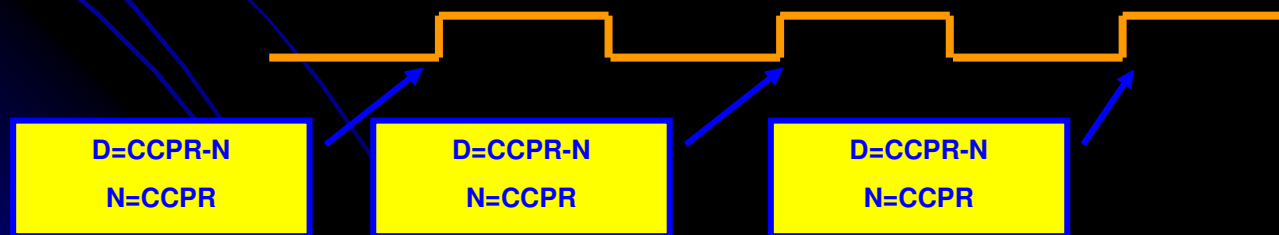
// sous programme d'interruption
#pragma interrupt itcomp
#pragma code interruption=0x8
void itcomp(void)
{static char tictac=7;
if (!tictac--)
    {
        tictac=15; // environ 500mS
        PORTB^=0x01; // bascule PB0
    }
PIR1.TMR2IF=0;
}
#pragma code

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

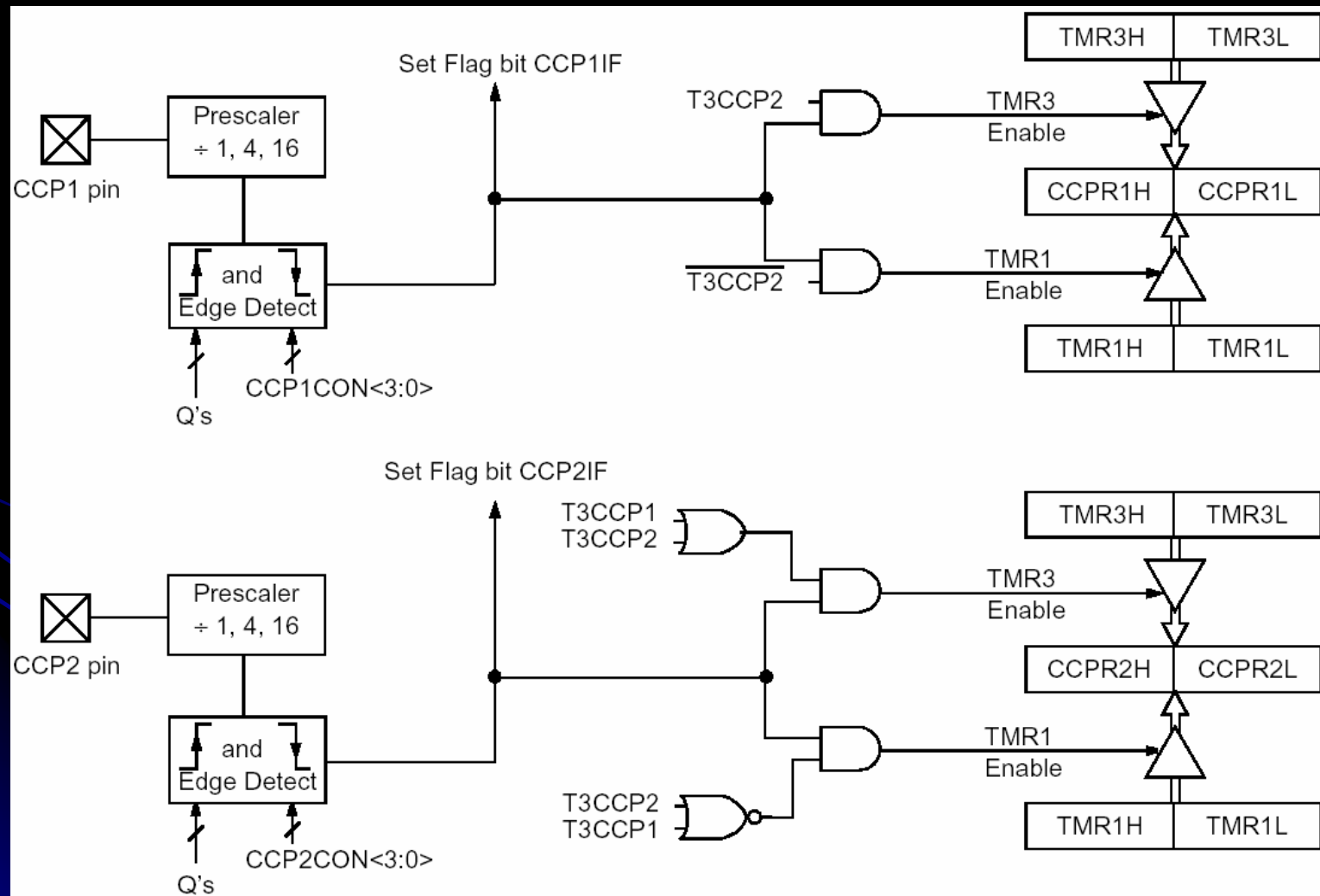


Capture -
principes

$$T = D * E / N$$



CAPTURE



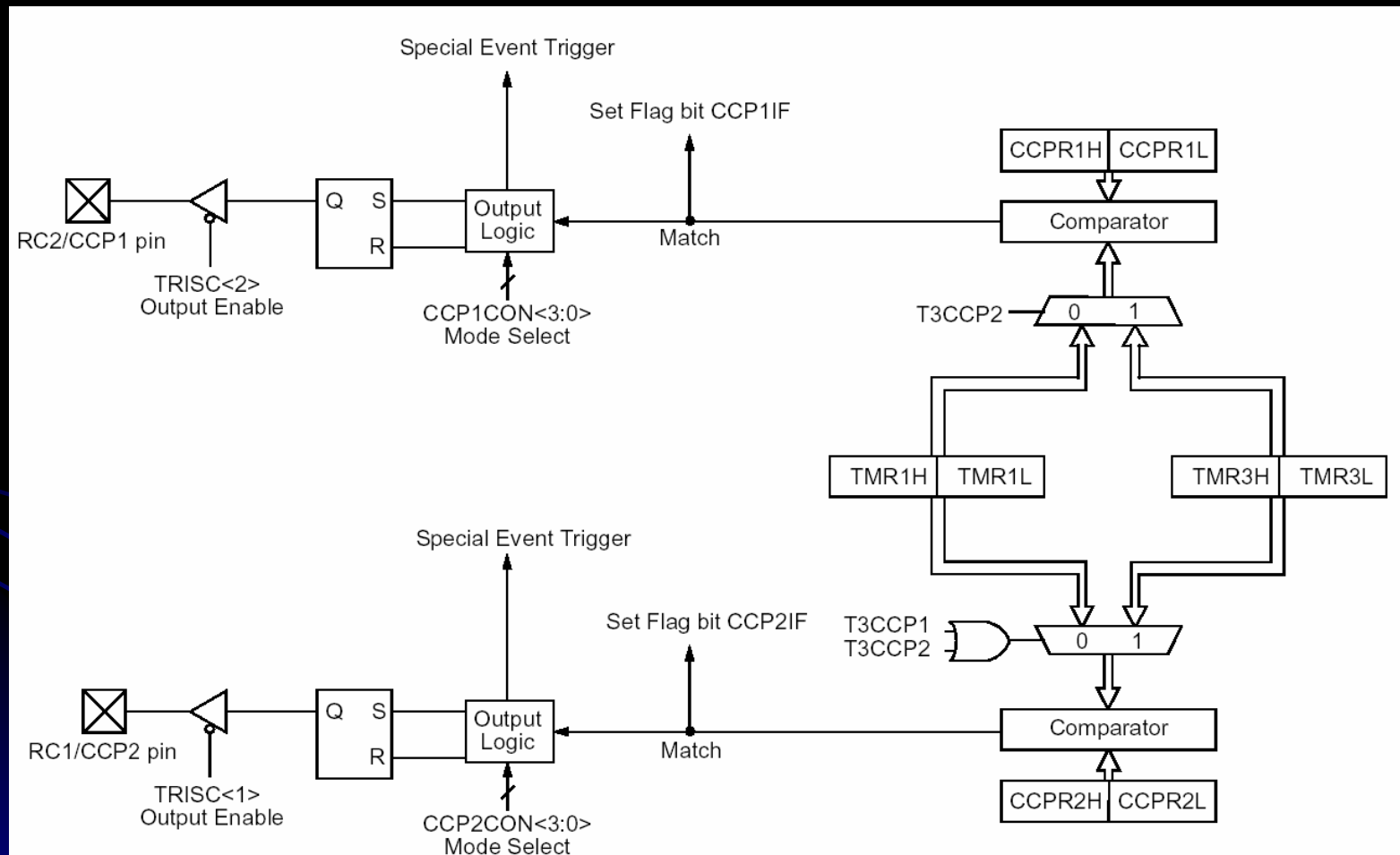
```

#include <p18f452.h>
unsigned int duree;          // représente le comptage entre 2 fronts
#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 fontion (void)
{__asm goto itcomp __endasm}
#pragma code
void main(void)
{
    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) ;
}

```

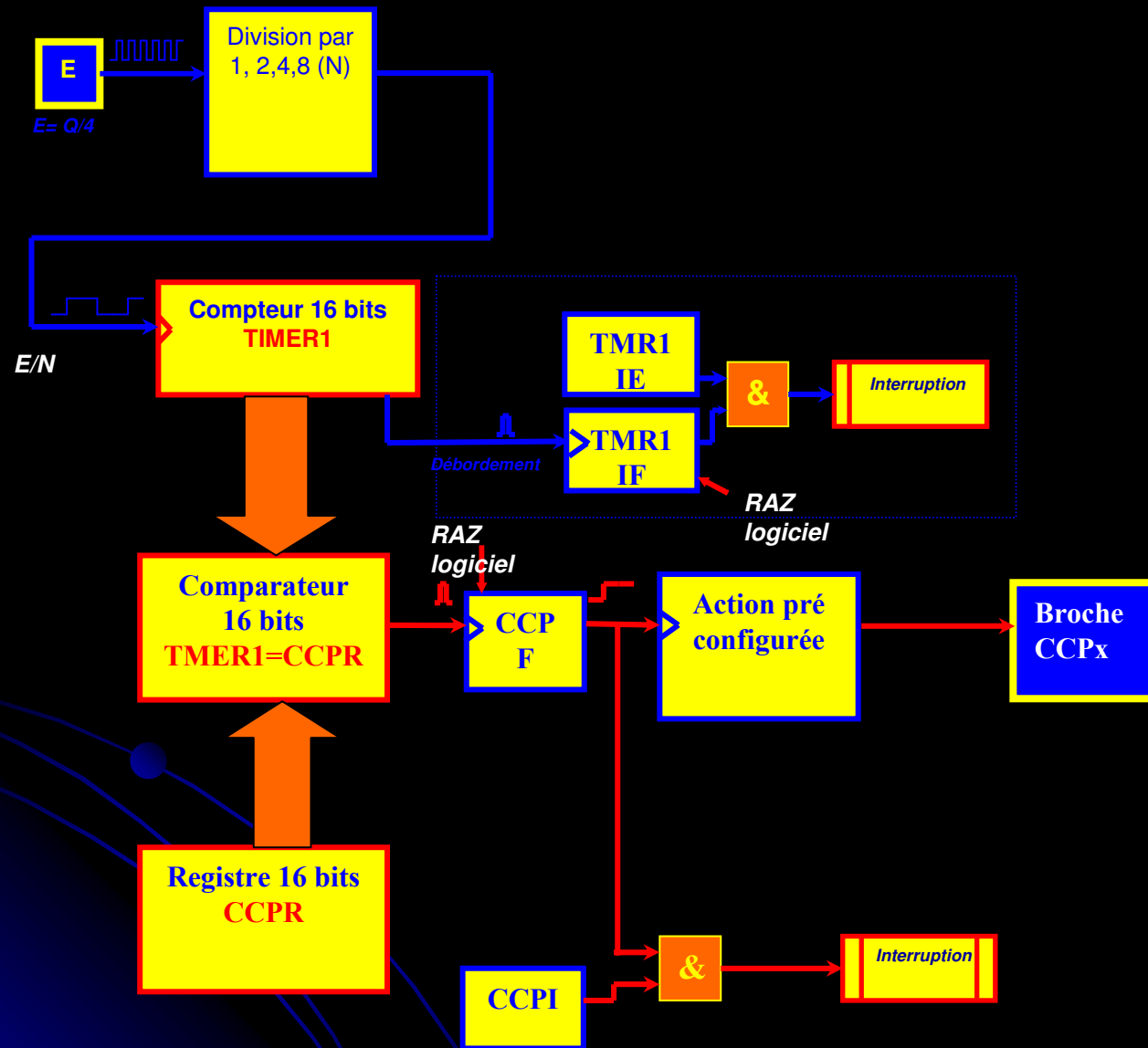
Capture - exemple

COMPARE



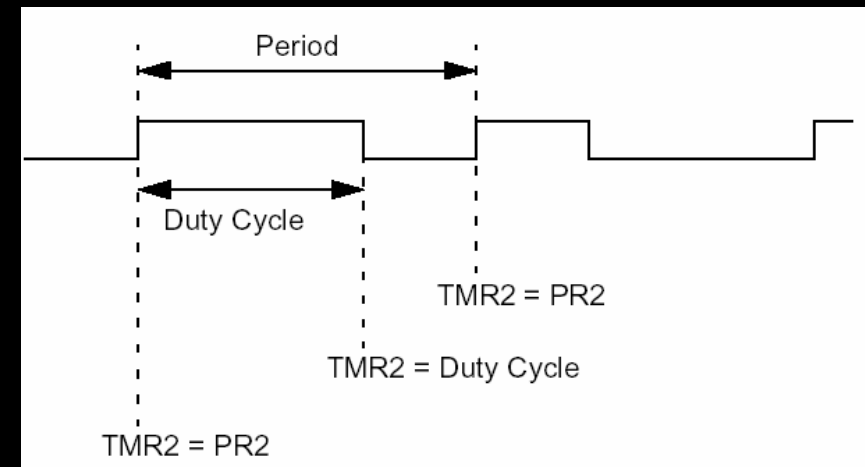
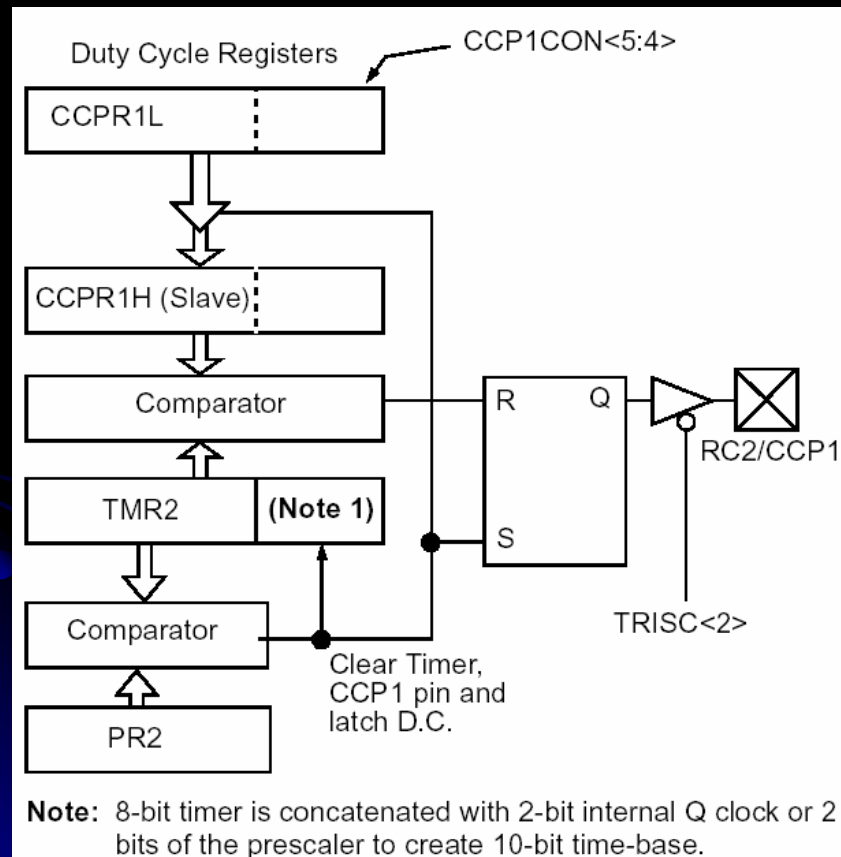
COMPARE

- Principes



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

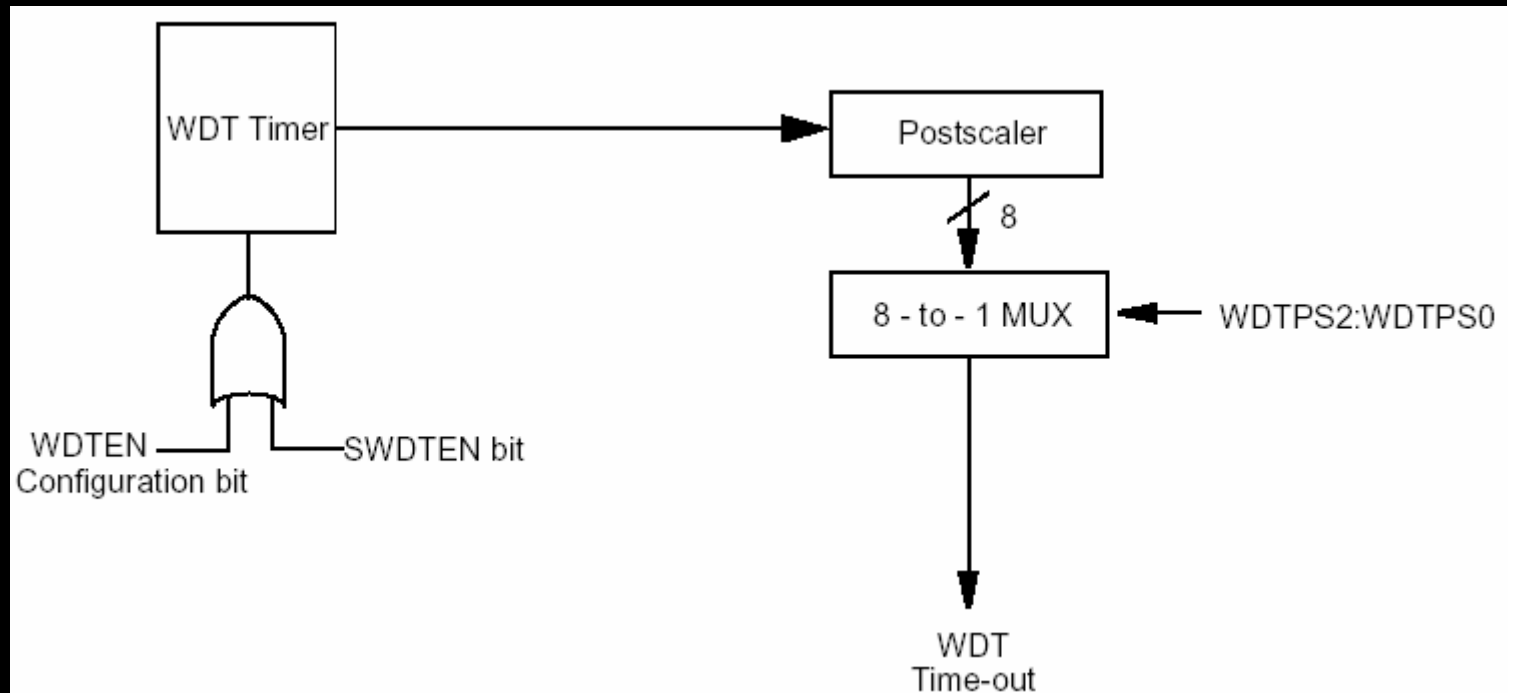
PWM



```
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

Watch Dog (WD)



REGISTRE CONFIG2H

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

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 « clrwtdt » 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 : $35\text{mS} < T < 165\text{mS}$

Acces EEPROM

Registre EECON1 (0xFA6)

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

EEPGD: Choix de la mémoire (FLASH ou EEPROM)

1 = Accès mémoire programme FLASH

0 = Accès mémoire de données EEPROM

CFGS: Accès mémoire ou configuration

1 = Accès aux registres de configuration ou de calibration

0 = Accès FLASH ou EEPROM

FREE: Validation d'effacement 64 octets en FLASH

1 = Efface la mémoire FLASH adressée par TBLPTR à la prochaine commande WR (RAZ du bit automatique)

0 = Effectue seulement une écriture

WRERR: Indication d'erreur en EEPROM

1 = une opération d'écriture a été arrêtée trop tôt

0 = l'opération d'écriture s'est terminée correctement

WREN: autorisation d'écriture en EEPROM

1 = cycle d'écriture autorisé

0 = cycle d'écriture interdit

WR: Contrôle d'écriture

1 = commence un cycle d'écriture ou d'effacement en EEPROM ou en FLASH

Initiates a data EEPROM erase/write cycle or a program memory erase cycle or write cycle.

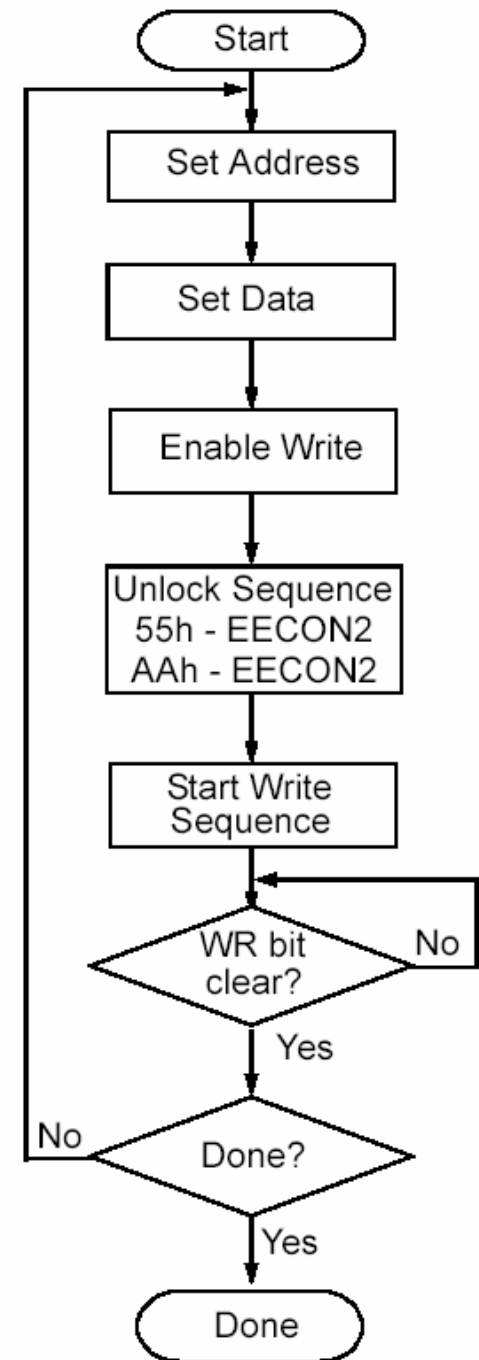
Ce bit est mis à 0 à la fin du cycle

0 = Cycle d'écriture terminé

RD: Contrôle de lecture

1 = commence une lecture en EEPROM (1 cycle machine) (RAZ automatique en fin de cycle)

0 = ne pas commencer un cycle de lecture



Accès EEPROM - Lecture

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
```

Accès EEPROM-Ecriture

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
```