Architecture interne d'un microprocesseur

Les différents constituants d'un microprocesseur (*µp*) peuvent être regroupés dans deux blocs principaux, l'unité de calcul et l'unité de contrôle.

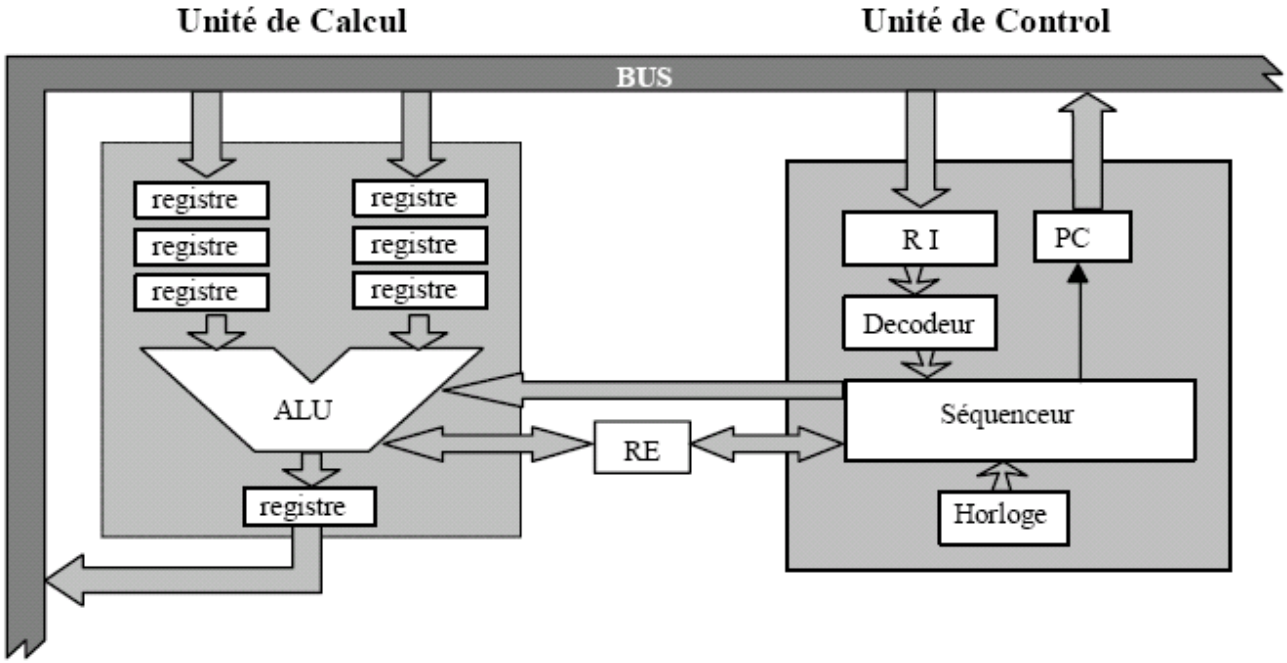


Fig. 1.1 : Architecture simplifiée d'un processeur

RI : registre d'instruction
PC : compteur de programme
RE : registre d'état

- **L'unité de calcul**

Elle est constituée de l'Unité Arithmétique et logique UAL et d'un certain nombre de registres

- **ALU** : Unité arithmétique et logique
- **Les registres :** Ce sont des mémoires élémentaires pouvant contenir chacun un opérande. Les registres peuvent être de 8, 16 ou 32 bits.

- **L'unité de contrôle**

C'est l'unité de contrôle qui supervise le déroulement de toutes les opérations au sein du *µp*.
Elle est constituée principalement de :

- **Horloge :** C'est l'horloge qui génère les signaux qui permettent le cadencement et la synchronisation de toutes les opérations.
- **Le compteur programme PC :** (*Program Counter*) contient l'adresse de la case mémoire où est stockée la prochaine instruction à charger. Au début de l'exécution d'un programme, le PC est initialisé à l'adresse mémoire où est stockée la première instruction du programme. Le compteur programme est incrémenté chaque fois qu'une instruction est chargée dans le ***µp.***
- **Le registre d'instruction RI :** C'est là où le *µp* stocke l'instruction en cours d'exécution.
- **Le décodeur :** C'est lui qui va "décoder" l'instruction contenue dans ***RI*** et générer les signaux logiques correspondants et les communiquer au séquenceur.
- **Le séquenceur** : Il gère le séquencement des opérations et génère :
  - les signaux du bus de commande (RD, WR, etc.),
  - les signaux internes aux µP (gestion des registres, de l'A.L.U., aiguillages des bus internes, etc.).

Le séquenceur est réalisé avec une structure qui comprend une mémoire ROM intégrée. Celle-ci contient des micro-instructions (à ne pas confondre avec les instructions contenues dans la mémoire programme). La mémoire des micro-instructions n'est pas accessible à l'utilisateur. Chacune des instructions du ***µp*** nécessite plusieurs micro-instructions et donc plusieurs cycles d'horloge.

- **Le registre d'état** : Le registre d'état est formé de plusieurs bits appelés drapeaux ou indicateurs (Flags) qui sont positionnés par l'ALU après chaque opération. On dispose d'un jeu d'instructions conditionnées par l'état de différents drapeaux. Par exemple l'indicateur **Z** indique, quand il est positionné, que le résultat de l'opération est égal à Zéro. L'indicateur **C** indique que l'opération a généré une retenue. Le bit N indique que le résultat est négatif …

Structure d'un système minimum à µp

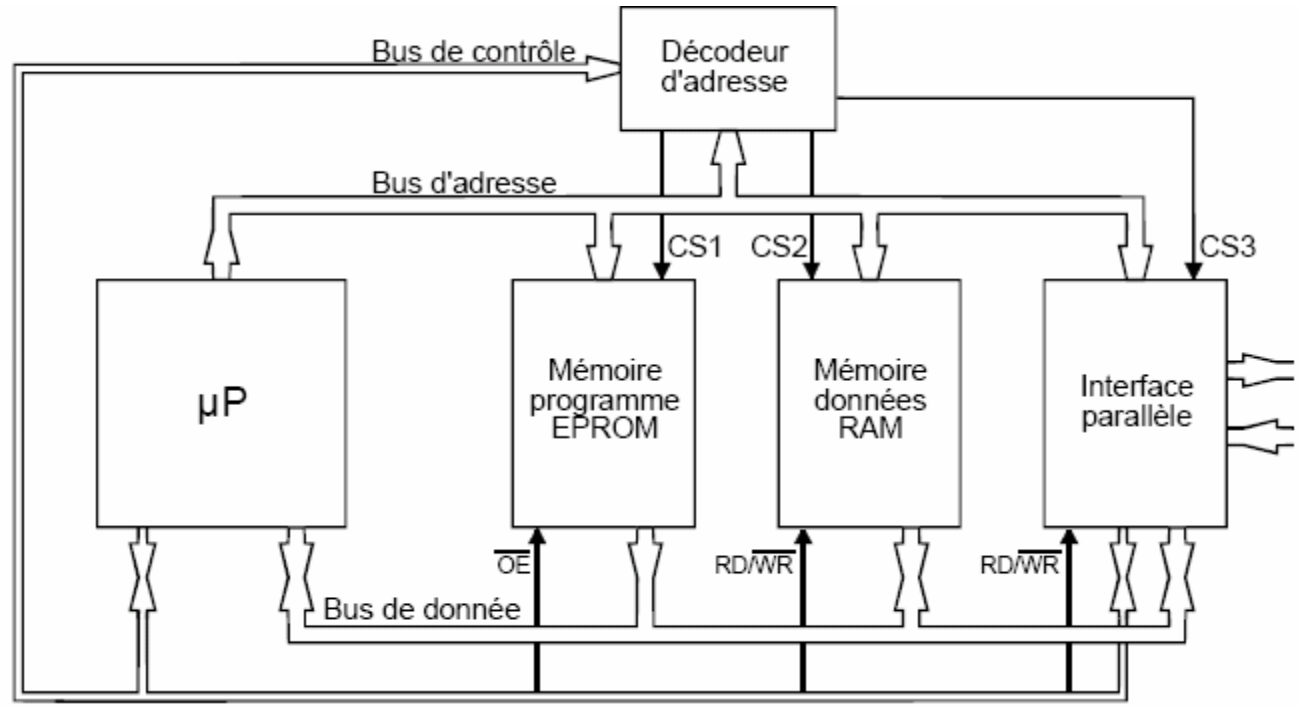
On appelle système minimum à *µP* l'ensemble comprenant le *µP* et les composants indispensables à son fonctionnement. Un système minimum à *µP* est constitué de :

- une unité centrale de traitement (*CPU - Central Processing Unit*) : le ***µP***.
- une unité de stockage du programme et des données : ***les mémoires***.
- une unité d'échanges : ***les interfaces d'E/S***, L'unité d'échange est vue par l'unité centrale comme un ensemble de registres accessibles. Les informations vers l'extérieur transitent par certains de ces registres.
- Les différentes unités sont réunies par des canaux d'échanges : ***les bus***.

Pour l'organisation des différentes unités, il existe deux architectures :

- l'architecture *Von Neumann*
- l'architecture *Harvard*

L'architecture Von Neumann

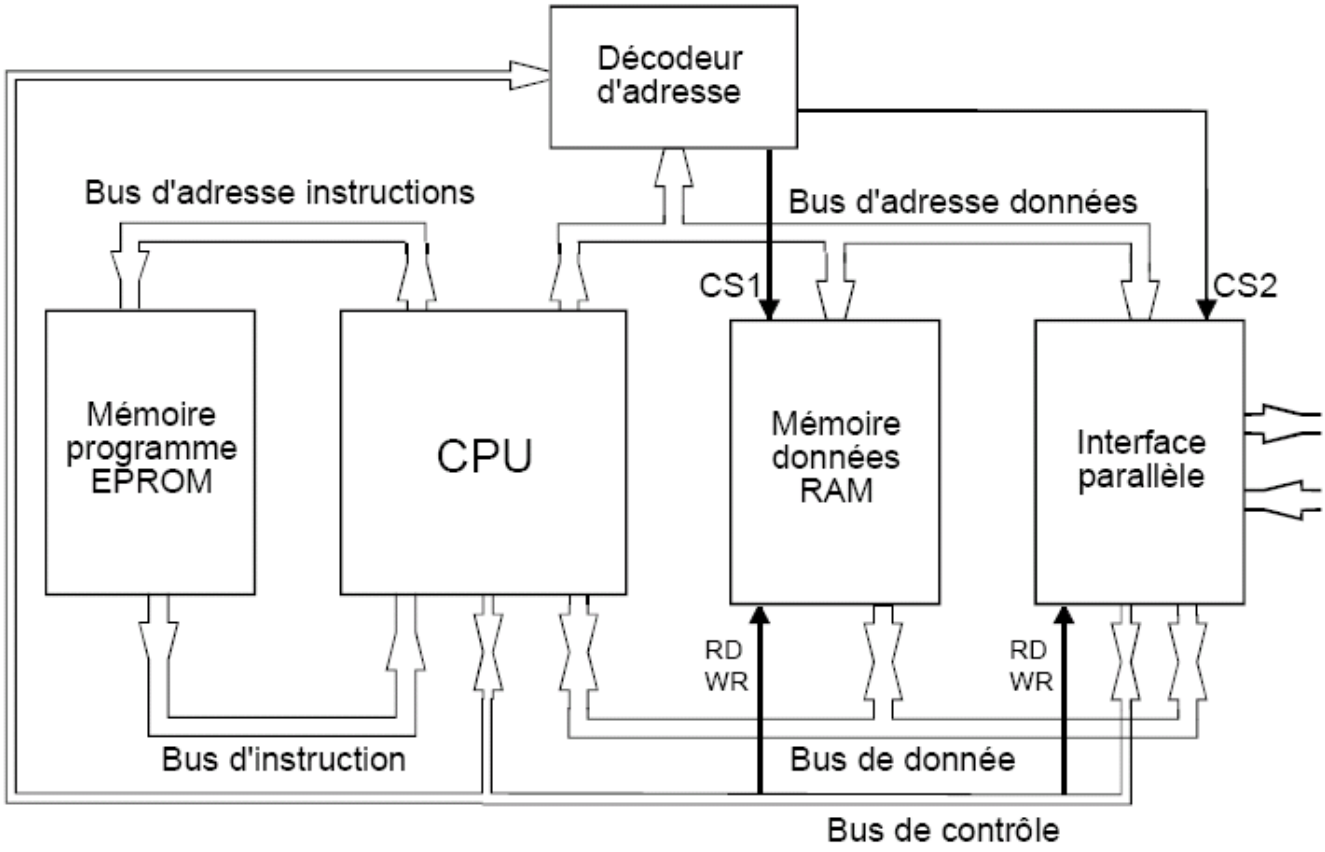


Le *µP* utilise le même canal d'échange pour lire les instructions et manipuler des données : ***le bus de données***. Les tailles les plus courantes des bus de données sont :

- 8 bits pour les petites applications embarquées
- 16 bits pour les applications embarquées de moyenne complexité
- 32 bits, 64 bits pour les gros calculateurs, les ordinateurs et consoles de jeux

Les adresses délivrées par le μP sont véhiculées par le bus d'adresses

L'architecture Harvard



La CPU utilise 2 canaux d'échange pour lire les instructions et manipuler les données : le ***bus instruction*** (ou ***bus programme***) et le ***bus de donnée***. Les tailles de ces bus sont en général différentes. La taille la plus courante du bus de données est de 8 bits.
Les deux bus distincts ***programme*** et ***donnée*** permettent de réaliser simultanément une recherche d'une instruction et l'exécution de l'instruction précédente.

Type d'architecture d'un microprocesseur

- **Architecture CISC (Complex Instruction Set Computer)**

C'est une architecture avec un grand nombre d'instructions. Le processeur doit exécuter des taches complexes par instruction unique. Donc, pour une tache donnée, une machine CISC exécute un petit nombre d'instructions mais chacun nécessite un plus grand nombre de cycles d'horloge (Intel 8086, Pentium…, Motorola 68000, PowerPC). Actuellement les deux technologies convergent : les processeurs CISC (Pentium par exemple) utilisent des instructions de plus en plus simples et exécutent parfois plusieurs instructions en un cycle d'horloge.

- **Architecture RISC (Reduced Instruction Set Computer)**

Architecture dans laquelle les instructions sont en nombre réduit (chargement, branchement, appel sous-programme) et elles sont fréquemment utilisées. Le but est d'éliminer les instructions rarement employées et de consacrer les ressources matérielles à exécuter les instructions relativement simples en un cycle d'horloge et à émuler les autres instructions à l'aide de séquences basées sur les instructions élémentaires. On trouve donc une meilleure performance à une vitesse donnée (le gain en performance envisageable est important mais dépend de la qualité du compilateur). Processeurs RISC : PowerRISC (IBM/Motorola), SPARC (SUN), PA-RISC (HP).

Les processeurs Spécialisés

➢ Microcontrôleurs (μC)

Ils contiennent un CPU, de la RAM, de la ROM, quelques ports d'E/S parallèles, des ports séries, des compteurs programmables (timers), des CAN/CNA, des interfaces pour réseaux de terrain ...
Ils sont en général utilisés pour contrôler des simples machines (appareils électroménagers, lecteurs de carte à puce...). Exemples de circuits :

- 80C186XX (80186, 16 bits, Intel)
- 68HC11, 68HC12 (6809, 8 bits, Motorola)
- 68HC16 (68000, 16 bits, Motorola)
- PIC 16F84, 16F877 (Microchip)

➢ Digital Signal Processor (DSP)

Ce sont des processeurs dédiés aux traitements des signaux numériques. Une architecture particulière leur permet un traitement efficace des fonctions complexes telles que FFT, convolution, filtrage numérique ...
Exemples :

- TMS320 (Texas Instrument)
- 2100 et 21000 (Analog Device)
- 56000 (Motorola)
- PIC30F4011 (Microchip)

Présentation du microcontrôleur PIC 16F877

Introduction

Les PICs sont des microcontrôleurs à architecture RISC (Reduced Instructions Set Computer), ou encore composant à jeu d'instructions réduit. L'avantage est que plus on réduit le nombre d'instructions, plus leur décodage sera rapide ce qui augmente la vitesse de fonctionnement du microcontrôleur.
Les PICs sont subdivisés en 3 grandes familles :

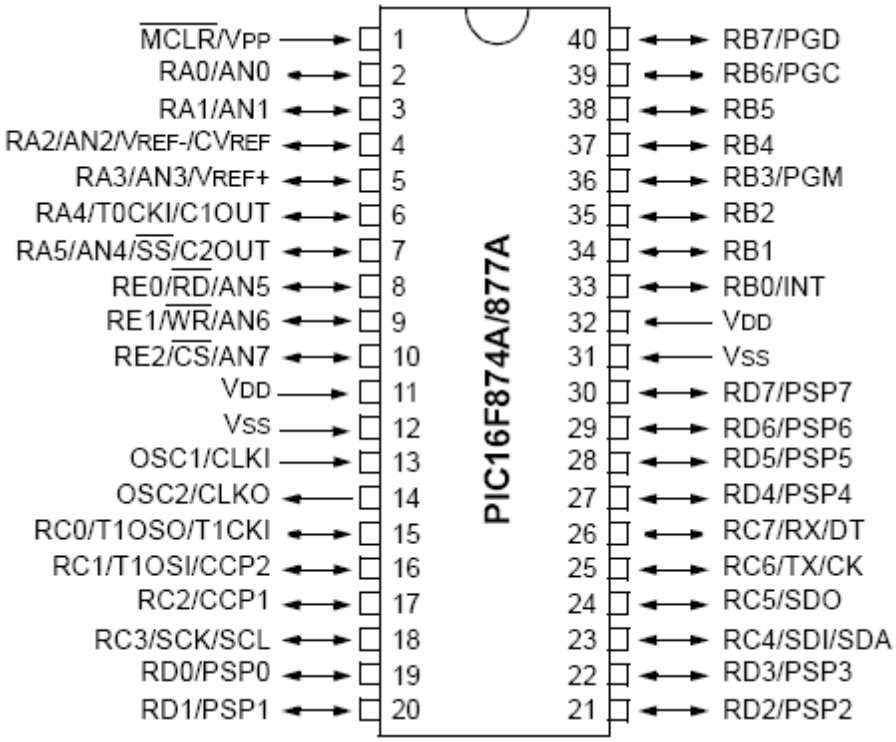
- La famille ***Base-Line***, qui utilise des mots d'instructions de 12 bits,
- la famille ***Mid-Range***, qui utilise des mots de 14 bits (et dont font partie les 16F8xx),
- et la famille ***High-End***, qui utilise des mots de 16 bits (les PIC 18Fxxx).

On trouve aussi des familles de dsPIC pour le traitement du signal et d'autres microcontrôleurs spécialisés !!
Dans ce cours, on va étudier les microcontrôleurs de la catégorie 16F87x (x=3, 4, 6, 7) qui sont les PIC les plus performants de la famille mid-range de Microchip.

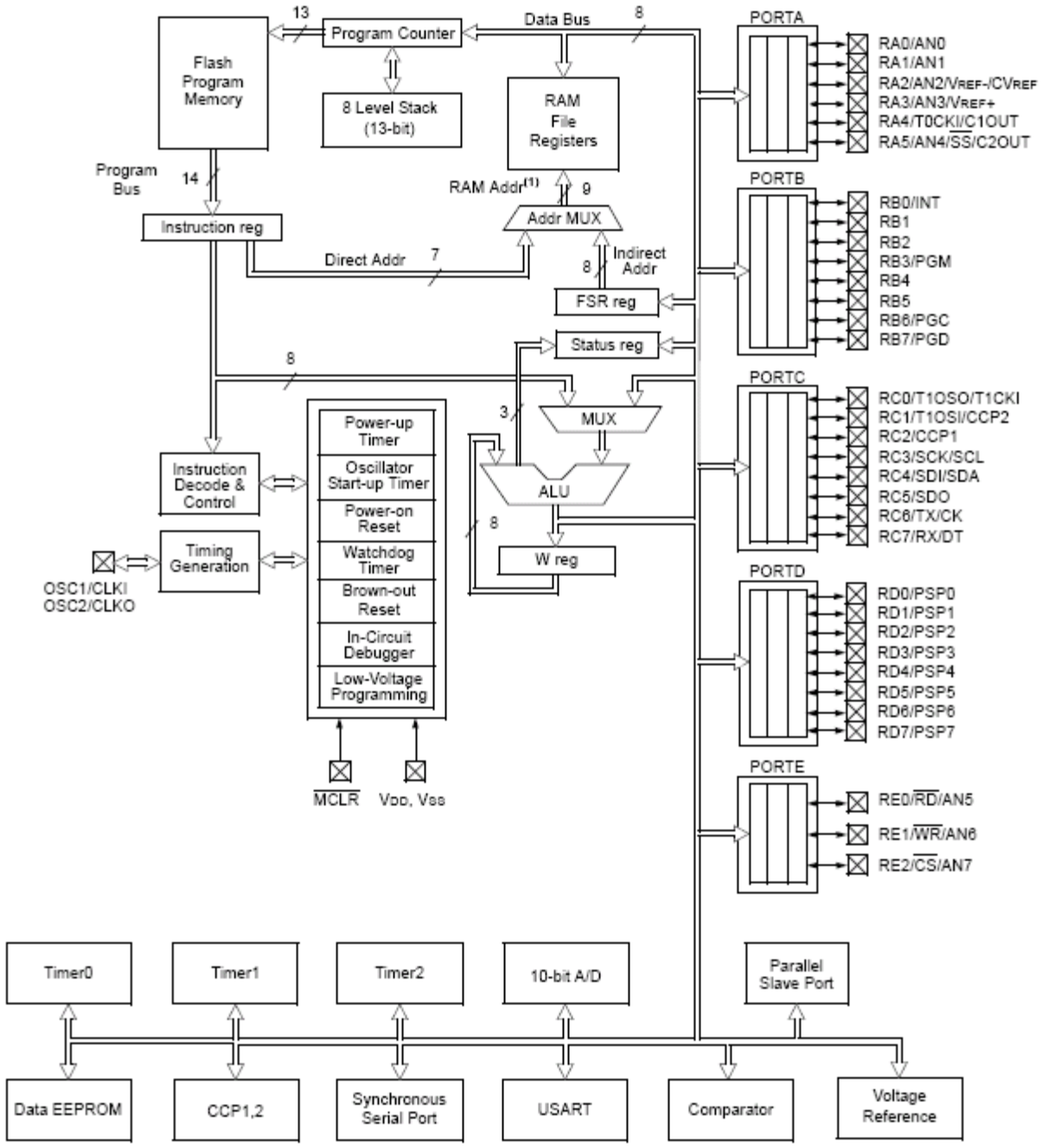
Les éléments essentiels du PIC 16F877 sont :

- Une mémoire programme de type EEPROM flash de 8K mots de 14 bits,
- Une RAM donnée de 368 octets,
- Une mémoire EEPROM de 256 octets,
- 5 ports d'entrée sortie, A (6 bits), B (8 bits), C (8 bits), D (8 bits) et E (3 bits)
- Convertisseur Analogique numérique 10 bits à 8 entrées sélectionnables,
- USART, Port série universel, mode asynchrone (RS232) et mode synchrone
- SSP, Port série synchrone supportant I2C
- 3 TIMERS avec leurs Prescalers, TMR0, TMR1, TMR2
- 2 modules de comparaison et Capture CCP1 et CCP2
- 15 sources d'interruption,
- Générateur d'horloge, à quartz (jusqu' à 20 MHz)
- Protection de code,
- Tension de fonctionnement de 2 à 5V,
- Jeux de 35 instructions

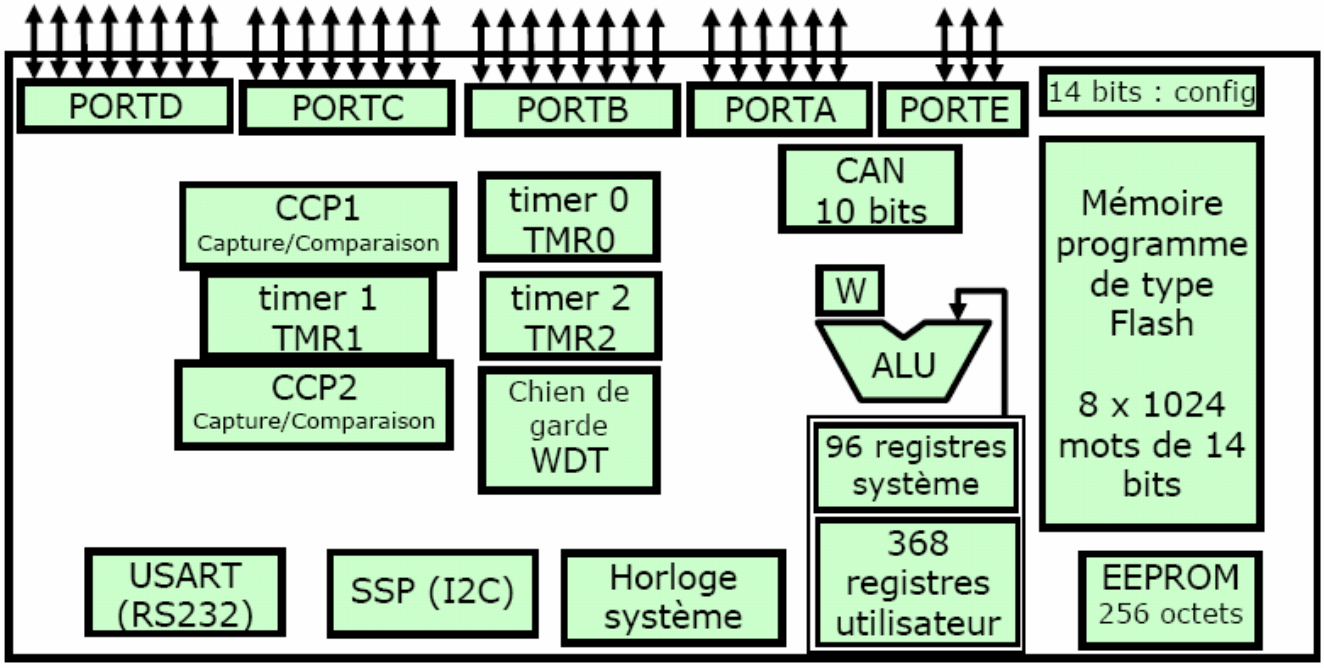
Description et architecture externe



Description et architecture interne



Les éléments constitutifs du PIC 16f877



Le PIC 16F877 a une :

- ***Architecture matériel de Harvard***
- ***Architecture RISC des instructions***

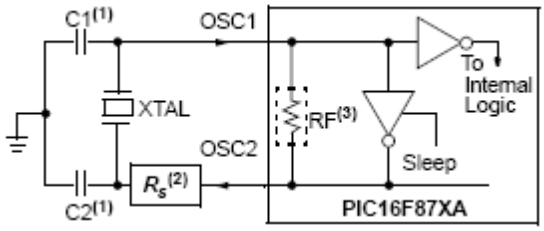
Minimum pour démarrer

L'horloge

Le rôle de l'horloge est de cadencer le rythme d'exécution des instructions. Il y a 4 modes possibles pour réaliser l'horloge :

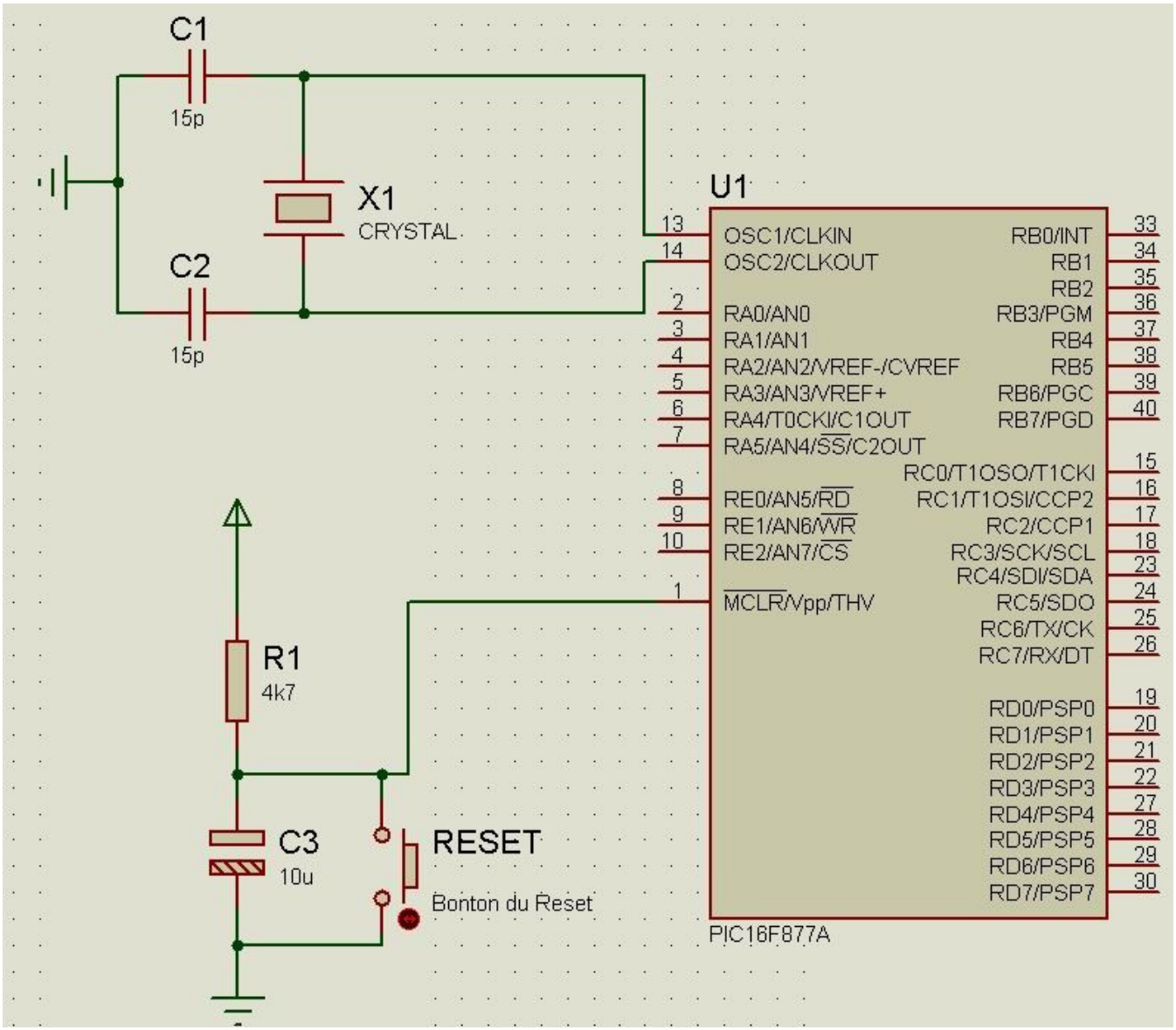
- LP Low-Power Crystal
- XT Crystal/Resonator
- HS High-Speed Crystal/Resonator
- RC Résistor/Capacitor

On utilise plus souvent un quartz (de 1MHZ jusqu'à 20MHZ) relié avec deux condensateurs de filtrage



Circuit de RESET

On utilise un circuit de reset externe qui permet la mise à la masse de l'entrée MCLR qui permet l'initialisation du PIC (Master Clear) à l'aide d'un bouton poussoir.
Un niveau bas sur l'entrée MCLR entraîne une réinitialisation complète du microcontrôleur.
D'une façon générale ce signal est activé à la mise sous tension. Un bouton poussoir est souvent rajouté afin qu'une réinitialisation manuelle soit possible.
Lorsque le signal de "RESET" est activé, tous les registres sont initialisé et le compteur programme se place à une adresse spécifique appelée "Vecteur de RESET".



Les mémoires internes

Il existe trois types de mémoire :

- RAM pour les registres internes et les données ;
- EEPROM de données ;
- EEPROM FLASH programme (données aussi)

Organisation de la mémoire RAM

L'espace mémoire RAM adressable est de **512** positions de 1 octet chacune : 96 positions sont réservées au SFR (Spécial Function Registers) qui sont les registres de configuration du PIC.
Les 416 positions restantes constituent les registres GPR (General Purpose Registers) ou RAM utilisateur.
Sur le 16F877, 3 blocs de 16 octets chacun ne sont pas implantés physiquement d'où une capacité de RAM utilisateur de 368 octets.

	File Address		File Address		File Address		File Address
Indirect addr.(*)	00h	Indirect addr.(*)	80h	Indirect addr.(*)	100h	Indirect addr.(*)	180h
TMR0	01h	OPTION_REG	81h	TMR0	101h	OPTION_REG	181h
PCL	02h	PCL	82h	PCL	102h	PCL	182h
STATUS	03h	STATUS	83h	STATUS	103h	STATUS	183h
FSR	04h	FSR	84h	FSR	104h	FSR	184h
PORTA	05h	TRISA	85h		105h		185h
PORTB	06h	TRISB	86h	PORTB	106h	TRISB	186h
PORTC	07h	TRISC	87h		107h		187h
PORTD(1)	08h	TRISD(1)	88h		108h		188h
PORTE(1)	09h	TRISE(1)	89h		109h		189h
PCLATH	0Ah	PCLATH	8Ah	PCLATH	10Ah	PCLATH	18Ah
INTCON	0Bh	INTCON	8Bh	INTCON	10Bh	INTCON	18Bh
PIR1	0Ch	PIE1	8Ch	EEDATA	10Ch	EECON1	18Ch
PIR2	0Dh	PIE2	8Dh	EEADR	10Dh	EECON2	18Dh
TMR1L	0Eh	PCON	8Eh	EEDATH	10Eh	Reserved(2)	18Eh
TMR1H	0Fh		8Fh	EEADRH	10Fh	Reserved(2)	18Fh
T1CON	10h		90h		110h		190h
TMR2	11h	SSPCON2	91h		111h		191h
T2CON	12h	PR2	92h		112h		192h
SSPBUF	13h	SSPADD	93h		113h		193h
SSPCON	14h	SSPSTAT	94h		114h		194h
CCPR1L	15h		95h		115h		195h
CCPR1H	16h		96h		116h		196h
CCP1CON	17h		97h	General Purpose Register 16 Bytes	117h	General Purpose Register 16 Bytes	197h
RCSTA	18h	TXSTA	98h		118h		198h
TXREG	19h	SPBRG	99h		119h		199h
RCREG	1Ah		9Ah		11Ah		19Ah
CCPR2L	1Bh		9Bh		11Bh		19Bh
CCPR2H	1Ch	CMCON	9Ch		11Ch		19Ch
CCP2CON	1Dh	CVRCON	9Dh		11Dh		19Dh
ADRESH	1Eh	ADRESL	9Eh		11Eh		19Eh
ADCON0	1Fh	ADCON1	9Fh		11Fh		19Fh
	20h		A0h		120h		1A0h
General Purpose Register 96 Bytes		General Purpose Register 80 Bytes		General Purpose Register 80 Bytes		General Purpose Register 80 Bytes	
	7Fh		EFh		16Fh		1EFh
Bank 0		accesses 70h-7Fh	F0h FFh	accesses 70h-7Fh	170h 17Fh	accesses 70h - 7Fh	1F0h 1FFh
		Bank 1		Bank 2		Bank 3	

Unimplemented data locations, read as "0"

(*) Not a physical register.

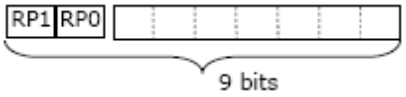
Accès à la mémoire RAM

Pour accéder à la RAM, on dispose de deux modes d'adressage :

- Adressage DIRECT

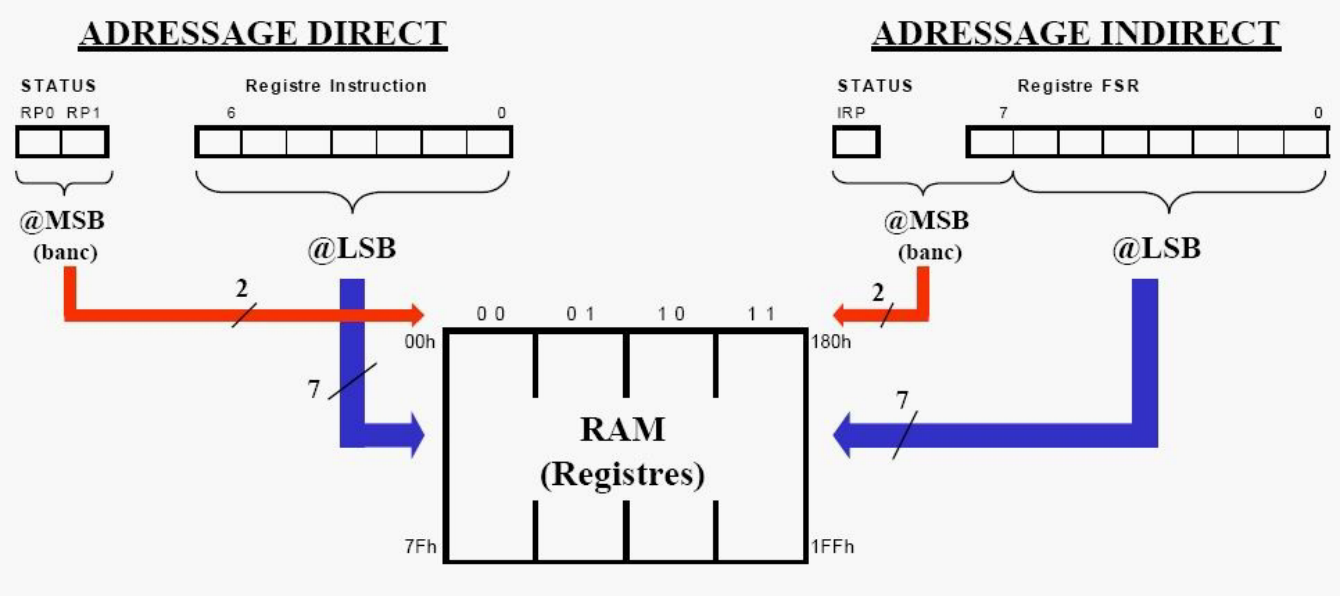
Avec ce mode d'adressage, on précise dans l'instruction la valeur de l'adresse à laquelle on veut accéder. Le jeu d'instruction du *µc* ne permet l'adressage des registres de la RAM que sur 7 bits (128 registres), l'espace mémoire est alors décomposé en 4 bank de 128 registres.
Il faut 9 bits d'adresse pour 512 octets. Le PIC complète les 7 bits par deux bits situés dans le registre STATUS. Ces bits sont appelés RP0 et RP1 et doivent être positionnés correctement avant toute instruction qui accède à la RAM par l'adressage direct.

RP1 RP0 : 00 → Bank 0
RP1 RP0 : 01 → Bank 1
RP1 RP0 : 10 → Bank 2
RP1 RP0 : 11 → Bank 3



- L'adressage INDIRECT

L'adressage indirect est possible en passant par un registre virtuel (INDF) dont l'adresse est contenue dans le registre FSR (File Select Register) et le bit IRP du registre STATUS.
Les deux types d'adressage sont présentés par la figure suivante :

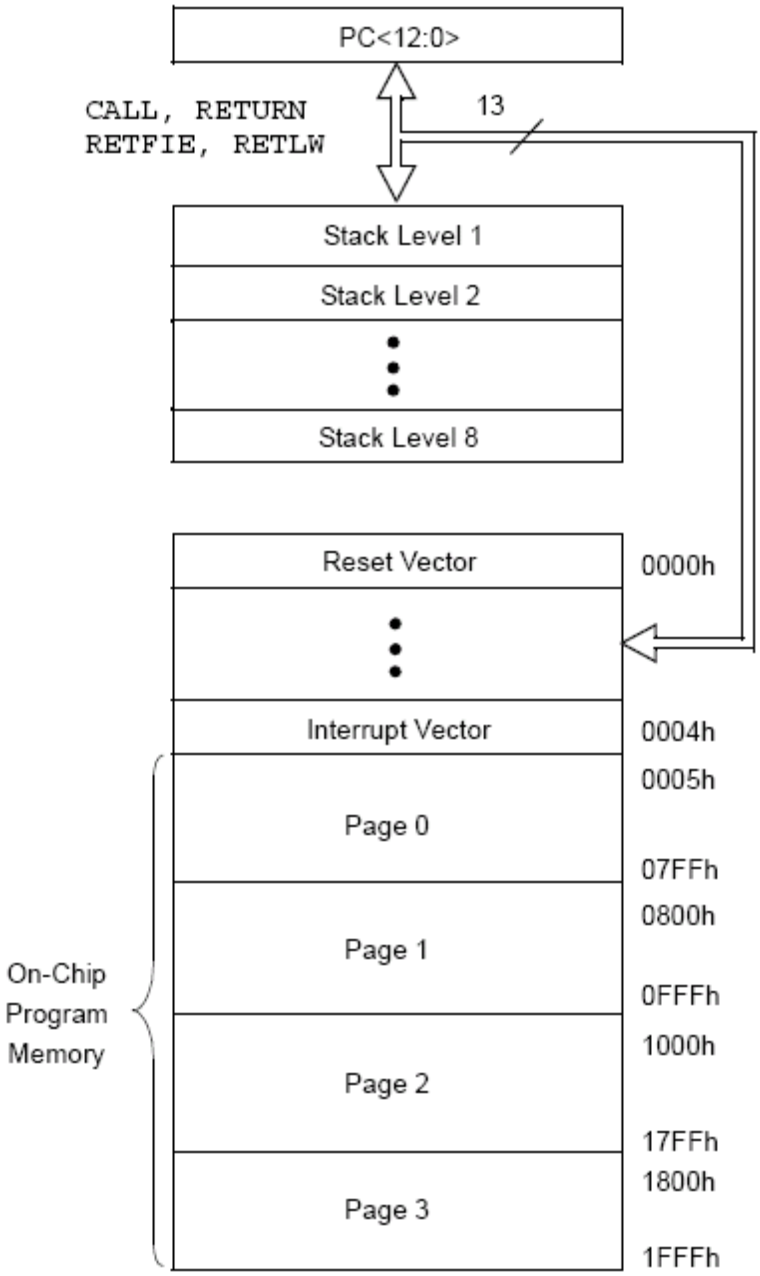


Mémoire programme

Le PIC 16F877 possède un compteur de programme de 13 bits ce qui permet l'adressage de 8K mots (instructions).
Il y a deux adresses réservées :

- ***Vecteur du RESET*** : 0000h adresse de début du programme
- ***Vecteur d'interruption*** : 0004h adresse de début du sous programme d'interruption (s'il y en a !!)

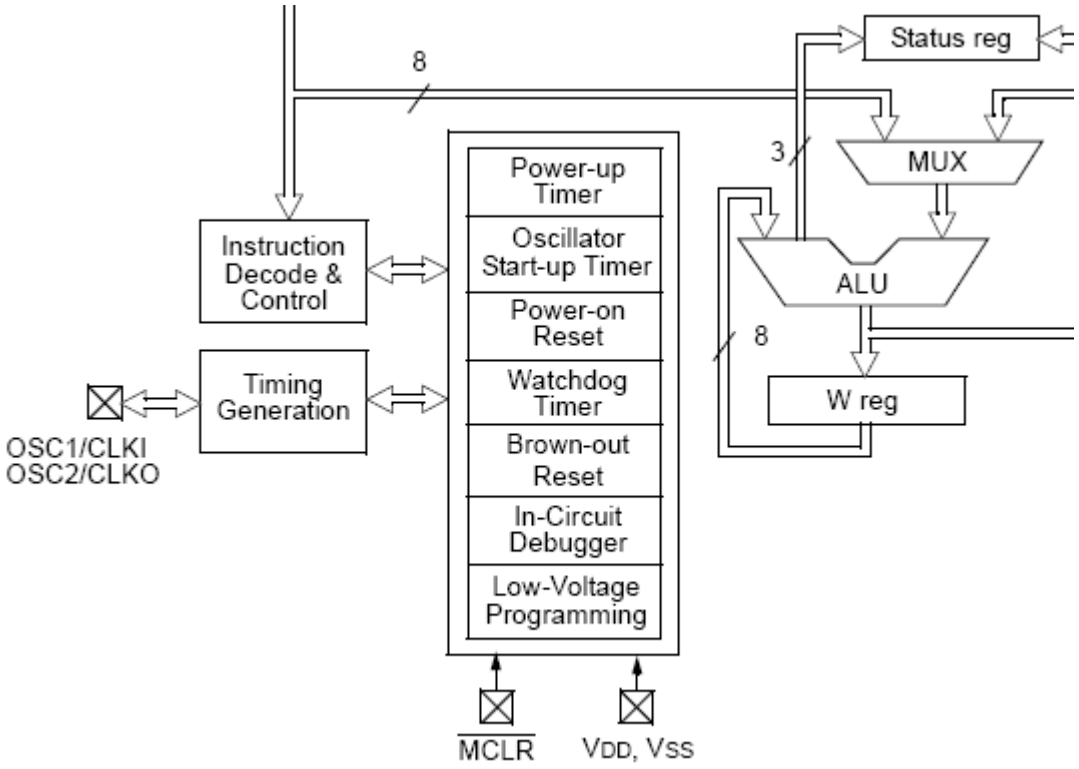
Le PIC possède une pile de 8 niveaux pour la sauvegarde des adresses de retours suite aux appels du sous programme.



Les Registres spéciaux

- L'accumulateur W (registre W)

L'ALU est une Unité Arithmétique et logique 8 Bits. L'accumulateur W est un registre de travail 8 bits, toutes les opérations à deux opérandes passent par ce registre W *(Work register)*.



➢ Le registre STATUS

C'est le ***registre d'état*** du μc, qui permet :

- De lire les drapeaux (Flags) de l'ALU
- Choix du Bank de la RAM soit en adressage directe (RP1 , RP0) ou indirecte (IRP)

IRP	RP1	RP0			Z	DC	C

Les indicateurs d'état

Ce sont des indicateurs qui permettent de savoir comment une instruction s'est terminée. Toutes les instructions n'agissent pas sur les indicateurs,

Z : passe à 1 quand le résultat d'une instruction est nul (ZERO)

C : passe à 1 quand l'opération a généré une retenue (CARY)

DC : passe à 1 quand les 4 bits du poid faible génèrent une retenue (DIGIT CARY)

L'accès à ce registre par adressage directe, son adresse est toujours 03h quelque soit la Bank (0,1,2 ou 3)

➢ Les Ports d'entrée/sortie

Le PIC 16F877 comporte 5 ports d'entrée/sortie :

- PortA : 6 lignes I/O digital ou entrés analogiques ;
- PortB : 8 lignes I/O digital
- PortC : 8 lignes I/O digital avec d'autre fonctionnalités ;
- PortD : 8 lignes I/O digital
- PortE : 3 lignes I/O digital ou entrés analogiques ;

Ces ports sont bidirectionnels, leur configuration se fait par des registres spécifiques (TRISx) ; par exemple TRISB configure le PortB :

Bit *i* de TRISB = 0 → bit *i* de PORTB configuré en **sortie**

Bit *i* de TRISB = 1 → bit *i* de PORTB configuré en **entrée**

Courant max en sortie : 20mA/pin ; 200mA/port

Les périphériques internes

Le 16F877 possède

- 3 compteurs indépendants (Timer : T0, T1, T2)
- 1 convertisseur analogique/numérique 10 bits pour 8 entrées analogiques multiplexées
- 1 port série (USART)
- 1 port I²C
- 1 port SPI (pour connecter des PICs entre eux)
- Mémoire EEPROM de 256 octets

Le jeu d'instructions

1. Opérations sur des registres

W → accumulateur **W** ; **F** → un registre de la RAM

d = 0 → W est donc la destination, d = 1 → F est donc la destination

Mnémonique		Description	Indicateurs
ADDWF	**F,d**	W+F → {W,F ? d}	C,DC,Z
ANDWF	**F,d**	W and F → {W,F ? d}	Z
CLRF	**F**	Clear F	Z
COMF	**F,d**	Complémente F → {W,F ? d}	Z
DECF	**F,d**	décrémente F → {W,F ? d}	Z
DECFSZ	**F,d**	décrémente F → {W,F ? d} skip if 0	
INCF	**F,d**	incrémente F → {W,F ? d}	Z
INCFSZ	**F,d**	incrémente F → {W,F ? d} skip if 0	
IORWF	**F,d**	W or F → {W,F ? d}	Z
MOVF	**F,d**	F → {W,F ? d}	Z
MOVWF	**F**	W → F	
RLF	**F,d**	rotation à gauche de F a travers C → {W,F ? d}	C
RRF	**F,d**	rotation à droite de F a travers C → {W,F ? d}	
SUBWF	**F,d**	F – W → {W,F ? d}	C,DC,Z
SWAPF	**F,d**	permute les 2 quartets de F → {W,F ? d}	
XORWF	**F,d**	W xor F → {W,F ? d}	Z

2. Opérations sur des bits (registres)

F → un registre de la RAM ; **b** → position (7 – 0) du bit dans le registre F ;

BCF	**F,b**	RAZ du bit b du registre F	
BSF	**F,b**	RAU du bit b du registre F	
BTFSC	**F,b**	teste le bit b de F, si 0 saute une instruction	
BTFSS	**F,b**	teste le bit b de F, si 1 saute une instruction	

3. Opérations immédiates (W)

K → une donnée immédiate (valeur) ;

ADDLW	**K**	W + K → W	C,DC,Z
ANDLW	**K**	W and K → W	Z
IORLW	**K**	W or K → W	Z
MOVLW	**K**	K → W	
SUBLW	**K**	K – W → W	C,DC,Z
XORLW	**K**	W xor K → W	Z

3. Instructions de contrôle (W)

K → une donnée immédiate (valeur) ;

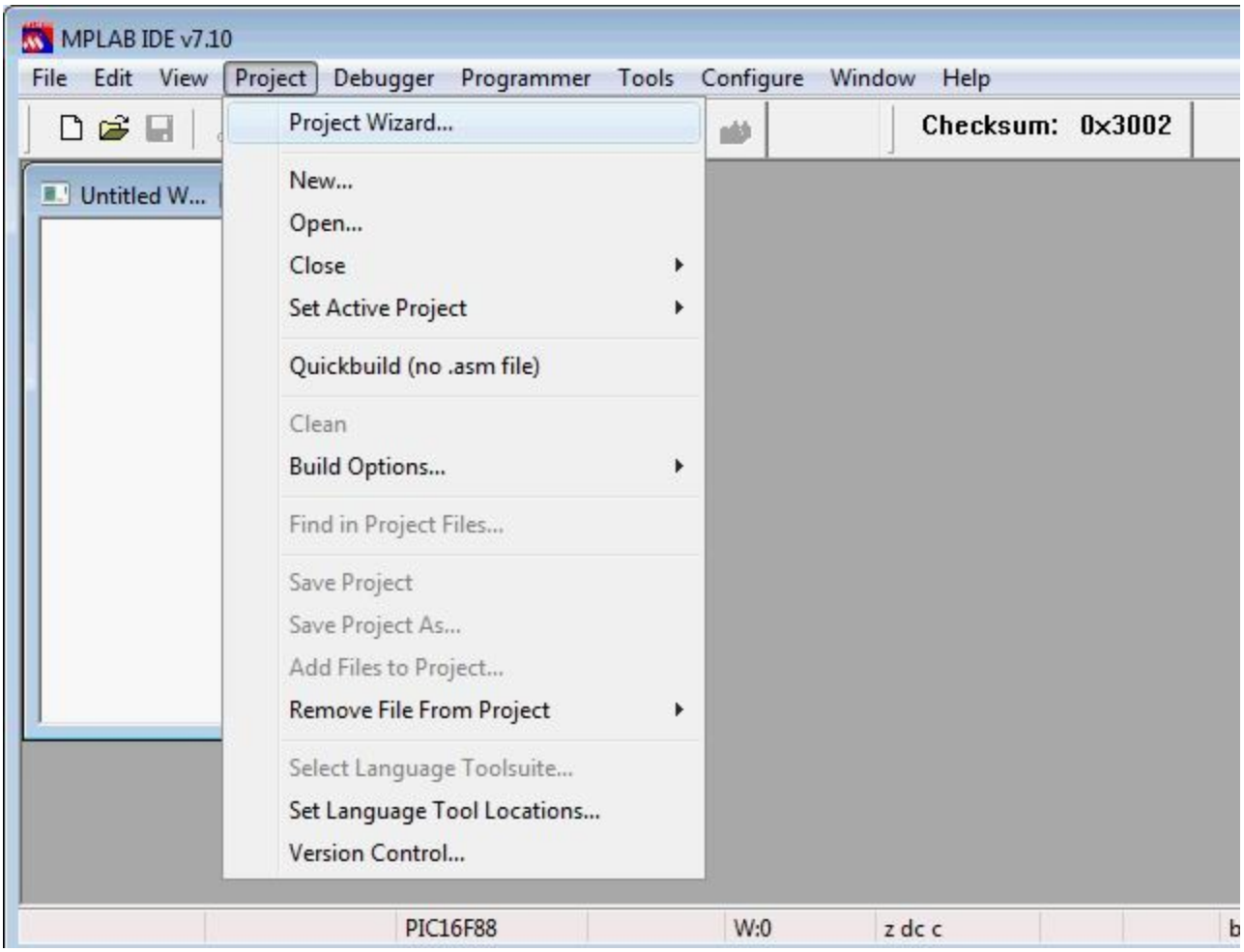
L → adresse (ou une étiquette)

CLRW		Clear W	Z
CLRWDT		Clear Watchdoc timer	TO', PD'
CALL	**L**	Branchement à un sous programme de label L	
GOTO	**L**	branchement à la ligne de label L	
NOP		No operation	
RETURN		retourne d'un sous programme	
RETFIE		Retour d'interruption	
RETLW	**K**	retourne d'un sous programme avec K dans W	
SLEEP		se met en mode standby	TO', PD'

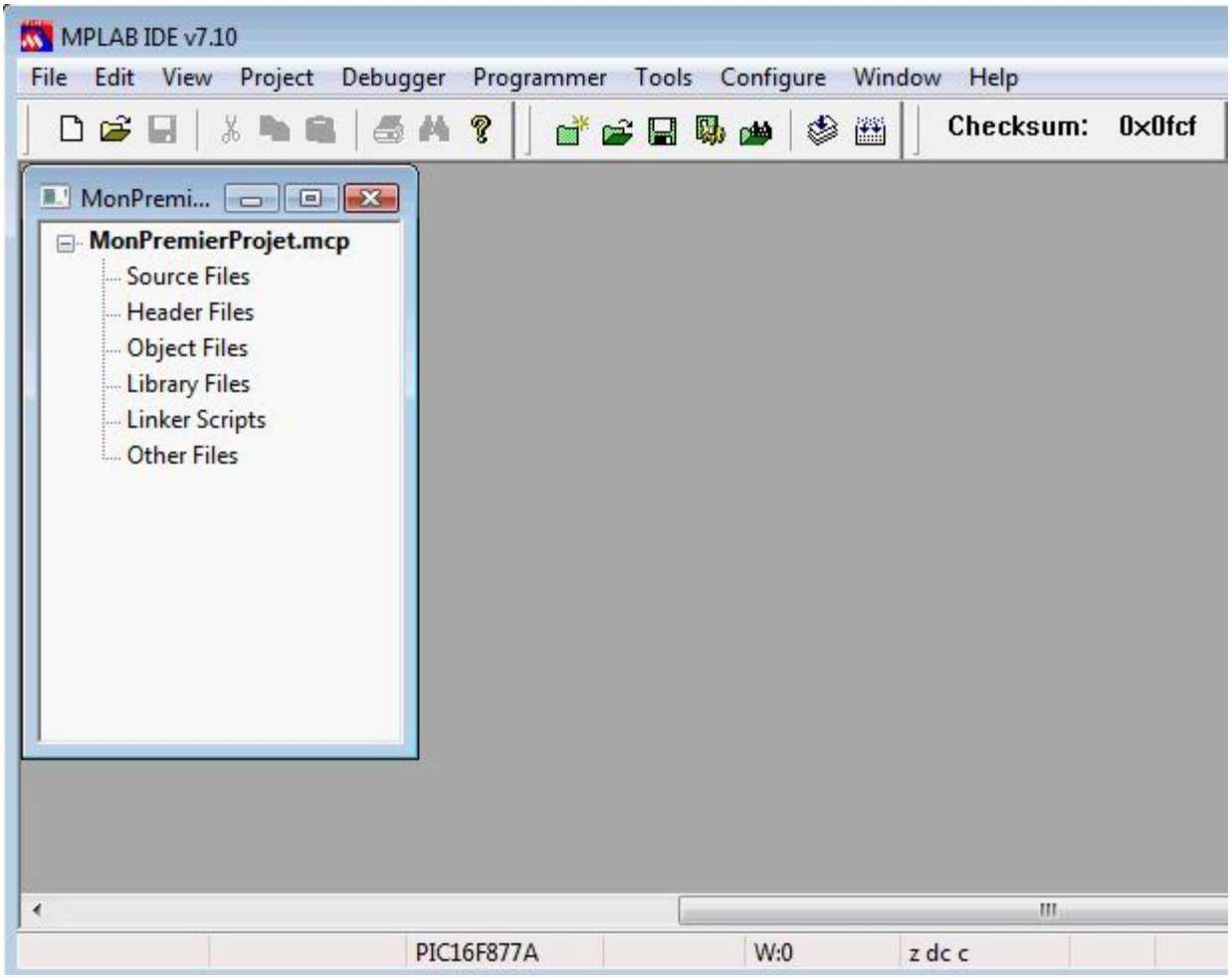
Programmation du microcontrôleur PIC 16F877

Assembleur & MPLAB

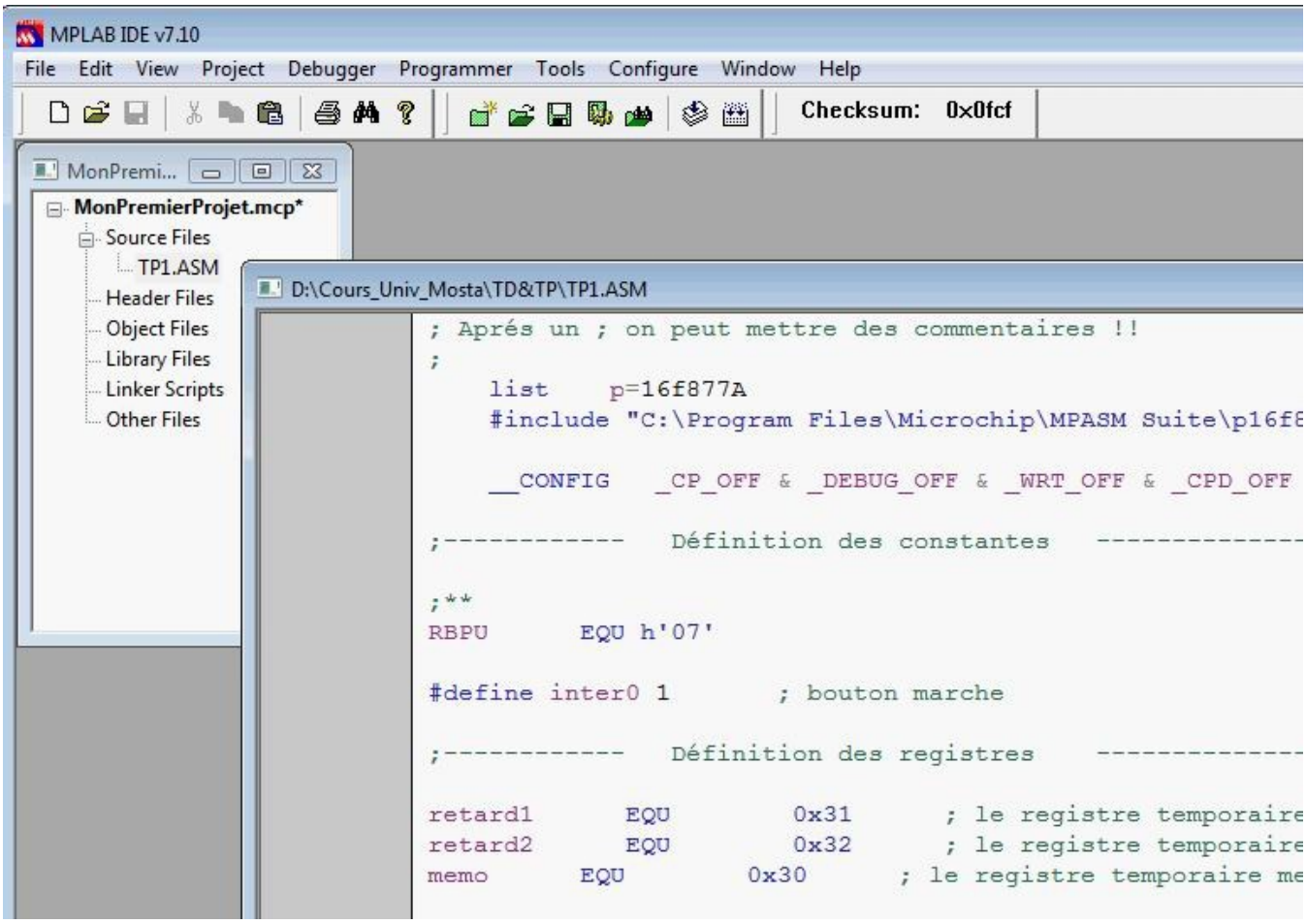
MPLAB est un environnement de programmation pour les PIC conçu par Microchip. MPLAB regroupe un éditeur de texte, un assembleur et un outil de simulation.



Une fois qu'on a créé un projet et choisi un PIC (en l'occurrence 16F877) en suivant les étapes du Wizard (le magicien en français) On aura la fenêtre suivante :



Il faut maintenant ouvrir un fichier source (.asm) dans le quel on écrira le programme en langage assembleur.
NB : il faut ajouter ce fichier dans la liste des fichiers comme ceci :



Un programme en Assembleur

```
; Après un ; on peut mettre des commentaires !!
        list p=16f877A
        #include "p16f877A.inc"
        __CONFIG _CP_OFF & _DEBUG_OFF & _WRT_OFF & _CPD_OFF & _LVP_OFF &
        _BODEN_OFF & _PWRTE_ON & _WDT_OFF & _HS_OSC
;------------ Définition des constantes ---------------
;**
RBPU EQU h'07'
#define inter0 1 ; bouton marche
;------------ Programme principal ---------------
ORG 0 ; l'adresse du début du programme est toujours '0'
        bsf STATUS,5 ; on met à 1 le 5eme bit du registre status pour accéder à la Bank1
        CLRF TRISC ; on met 00 dans le port C il est programmé en sortie
        bcf STATUS,5 ; revenir à Bank0
        MOVLW 0xFF ; on met FF dans le registre W
        MOVWF PORTC ; allumer toutes les LED
Boucle

        GOTO boucle ; retour au début de la boucle
END
```

Commentaires :

Après un ';' tout texte est ignoré, c-à-d qu'on peut écrire des commentaires !!

Les directives :

LIST

permet de définir un certain nombre de paramètres comme le processeur utilisé (p), la base par défaut pour les nombres (r), le format du fichier hex à produire (f) ainsi que d'autres paramètres.
Exemple : LIST p=16F876, r=dec, f=inhx8m

INCLUDE

permet d'insérer un fichier source. Par exemple le fichier p16f876.inc contient la définition d'un certain nombre de constante comme les noms des registres ainsi que les noms de certain bits;
exemple : INCLUDE « p16f876.inc »

EQU

permet de définir une constante ou une variable : var EQU 0x20

ORG

définit la position dans la mémoire programme à partir de laquelle seront inscrites les instructions suivantes.

DE

pour déclarer des donnés qui seront stockée dans l'EEPROM de donnée au moment de l'implantation du programme sur le PIC,
exemple : *ORG 0x2100*
DE "Programmer un PIC, rien de plus simple", 70, 'Z'

END : indique la fin du programme

__CONFIG

permet de définir les 14 fusibles de configuration qui seront copiés dans l'EEPROM de configuration lors de l'implantation du programme dans le PIC

CP1	CP0	DEBUG	—	WRT	CPD	LVP	BODEN	CP1	CP0	PWRTE	WDTE	F0SC1	F0SC0

CP1/CP0
1 1 : Aucune protection (_CP_OFF)
1 0 : Protection de la zone 0x1F00 à 0x1FFF (_CP_UPPER_256)
0 1 : Protection de la zone 0x1000 à 0x1FFF (_CP_HALF)
0 0 : Protection de l'intégralité de la mémoire (_CP_ALL)

DEBUG
1 : RB6 et RB7 sont des I/O ordinaires (_DEBUG_OFF)
0 : RB6 et RB7 sont utilisés pour le debuggage sur circuit (_DEBUG_ON)

WRT
1 : Le programme peut écrire dans les zones non protégées par les bits CP1/CP0 (_WRT_ENABLE_ON)
0 : Le programme ne peut pas écrire en mémoire flash (_WRT_ENABLE_OFF)

CPD
1 : mémoire EEPROM non protégée (_CPD_OFF)
0 : mémoire EEPROM protégée contre la lecture externe via ICSP (_CPD_ON)

LVP
1 : La pin RB3 permet la programmation du circuit sous tension de 5V (_LVP_ON)
0 : La pin RB3 est utilisée comme I/O standard (_LVP_OFF)

BODEN : provoque le reset du PIC en cas de chute de tension
1 : En service (_BODEN_ON)
0 : hors service (_BODEN_OFF)

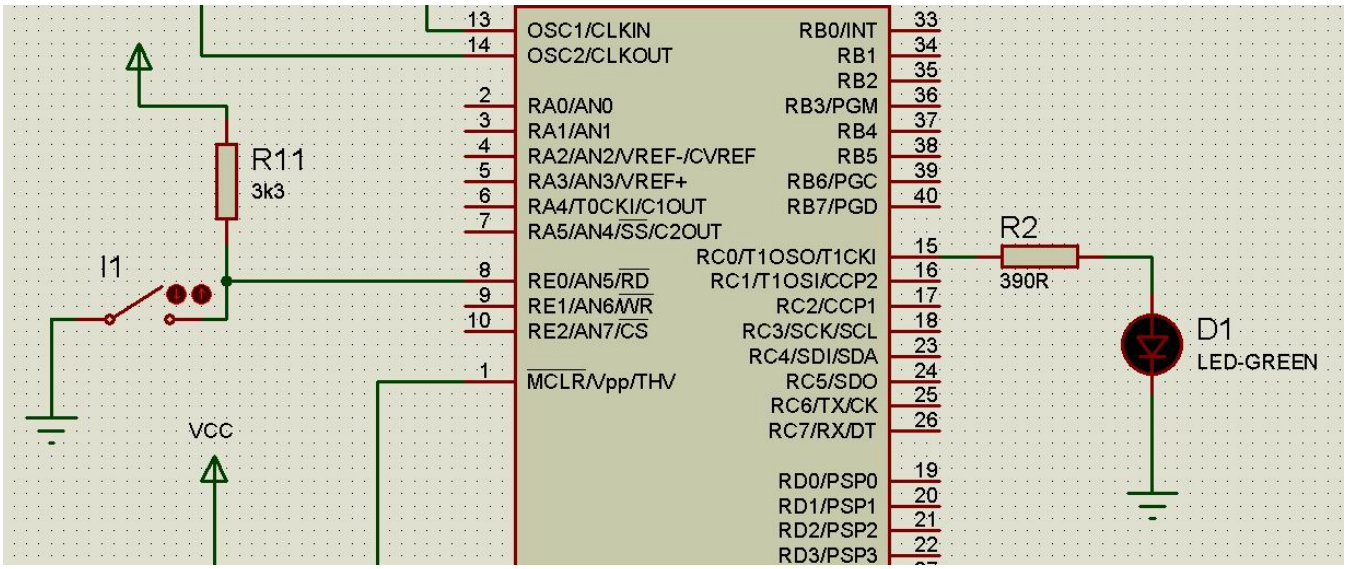
PWRTE : bit 3 : Délai de démarrage à la mise en service. Attention, est automatiquement mis en service si le bit BODEN est positionné.
1 : délai hors service (sauf si BODEN = 1) (_PWRTE_OFF)
0 : délai en service (_PWRTE_ON)

WDTE : bit 2 : Validation du Watchdog timer
1 : WDT en service (_WDT_ON)
0 : WDT hors service (_WDT_OFF)
FOSC1/FOSC0 : bits 1/0 : sélection du type d'oscillateur
11 : Oscillateur de type RC (_RC_OSC) (3K < R < 100k, C > 20 pF)
10 : Oscillateur haute vitesse (_HS_OSC) (4 Mhz à 20 Mhz)
01 : Oscillateur basse vitesse (_XT_OSC) (200 kHz à 4 Mhz)
00 : Oscillateur faible consommation (_LP_OSC) (32 k à 200 kHz)
Exemples d'utilisation :
__CONFIG B'11111100111001'
__CONFIG H'3F39'
__CONFIG _CP_OFF & _DEBUG_OFF & _WRT_ENABLE_ON & _CPD_OFF & _LVP_OFF & _BODEN_OFF & _PWRTE_OFF & _WDT_OFF & _XT_OSC

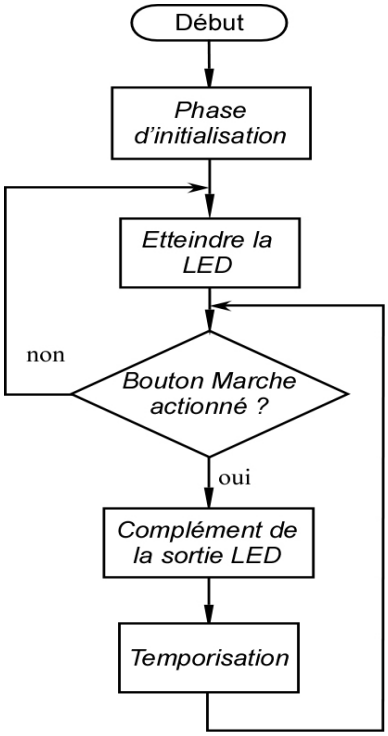
Exemples de programmation des Ports

Réaliser un clignotant pour une LED à une certaine fréquence avec la possibilité d'arrêter le clignotement à tout moment à l'aide d'un interrupteur

a- Le schéma



b- L'organigramme



C- Le programme

```
list p=16f877A
#include P16f877A.inc
__CONFIG _CP_OFF & _DEBUG_OFF & _WRT_OFF & _CPD_OFF & _LVP_OFF &
_BODEN_OFF & _PWRTE_ON & _WDT_OFF & _HS_OSC

;------------ Définition des constantes ---------------
#define inter  0                ; bouton marche

;------------ Définition des variables ---------------
memo   EQU   0x30               ; le registre temporaire memo se trouve à l'adresse 30h
retard1   EQU  0x31
retard2   EQU   0x32

;------------ Programme principal ---------------
        ORG 0x00      ; l'adresse du début du programme est toujours '0'
        BSF STATUS, RP0   ; on met à 1 le 5eme bit du registre status pour accéder
                          ; à la 2ème page mémoire (pour TRISA et TRISB)
        MOVLW 0x00        ; on met 00 dans le registre W
        MOVWF TRISC       ; on met 00 dans le port C il est programmé en sortie
        MOVLW 0xFF        ; on met 1F dans le registre W
        MOVWF TRISE       ; on met 1F dans le port TRISE, il est programmé en entrée
        MOVLW h'06'       ; le convertisseur AD désactivé !!!
        MOVWF ADCON1
        BCF STATUS, RP0   ; on remet à 0 le 5eme bit du registre status pour accéder  à
                          ; la 1ère page mémoire
        CLRF PORTC        ; on met 0 sur le port C (leds)
        CLRF memo         ; on met 0 dans le registre memo
```

```
;-------------------- Boucle principale ----------------------
debut           ; etiquette (Label)
        BTFSS PORTE, inter          ; interrupteur (marche) appuyé ? si oui on continu sinon ;va à debut
        GOTO Marche
        CLRF  PORTC                 ; éteindre la led
        GOTO debut
Marche
        COMF memo,W
        MOVWF PORTC
        MOVWF memo
        CALL tempo
        GOTO debut          ; retour au début de la boucle

;------------ Sous Programme de temporisation longue ---------------
tempo
        MOVLW 0xff          ; on met ff dans le registre W
        MOVWF retard1       ; on met W dans le registre retard1
        MOVWF retard2       ; on met W dans le registre retard2
attente
        DECFSZ retard1, F   ; on décrémente retard1 et on saute la prochaine instruction si
        GOTO attente        ; le registre retard1 = 0 sinon retour à attente
        MOVLW 0xff          ; on recharge retard1
        MOVWF retard1
        DECFSZ retard2, F   ; on décrémente retard2 et on saute la prochaine instruction si
        GOTO attente        ; le registre retard2 = 0 sinon retour à attente
        RETURN
        END                 ; fin programme
```