Introduction aux microcontrôleurs et à leur assembleur
Illustration par le PIC16F877

F. Senny

Université de Liège

Faculté des Sciences Appliquées

Plan

- Qu'est-ce qu'un micro-contrôleur ?
- Architecture du PIC16F877
  - Ports I/O
  - Mémoires programme et données
  - Autres modules
- Programmation du PIC
  - Projet MPLAB
  - Exemple de code
  - Les interruptions

Un microquoi ?

- Microcontrôleur : microprocesseur + périphériques internes → traitement d'info
- Les PICs
  - 3 grandes familles
    - Base-Line (12 bits)
    - Mid-Range (14 bits)
    - High-End (16 bits)
  - 16F877 = Mid-Range, mémoire FLASH (F)

Carte de développement

LEDs

Alim.

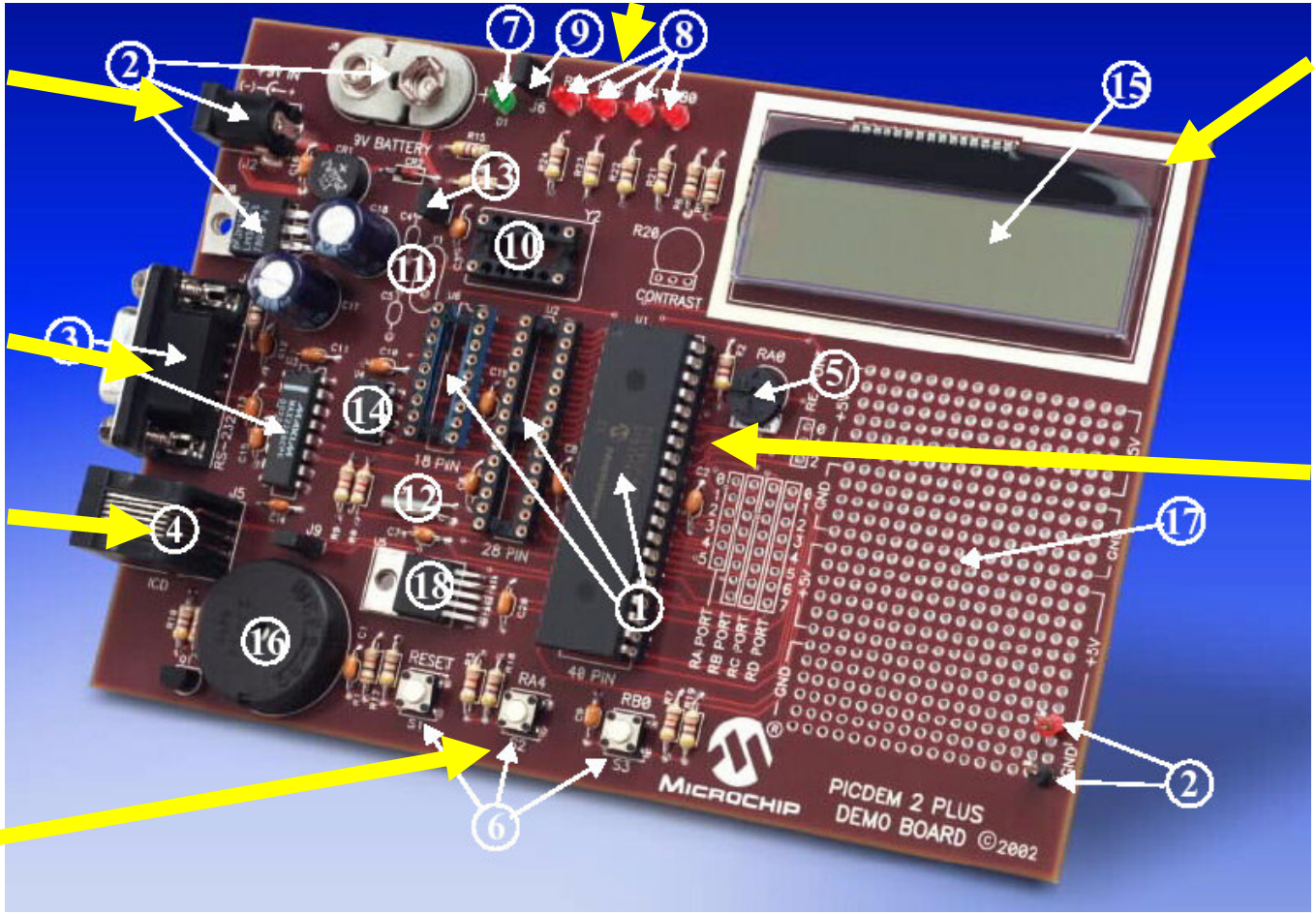
RS232

ICD2

boutons

LCD

PIC



Architecture PIC16F877

Classifications des microcontrôleurs à deux niveaux

- Au niveau du processeur:
  - **RISC** : Reduced Instruction Set Computer
  - **CISC** : Complex Instruction Set Computer
- Au niveau de l'organisation de la mémoire
  - Architecture **Von Neumann** : une mémoire unique et pour le programme et pour les données
  - Architecture **Harvard** : le programme et les données sont stockées dans des mémoires physiquement séparées

PIC16F877 → RISC (4 cycles d'horloge/instr.), Harvard

Schéma bloc PIC16F877

Mémoires

Contrôle ALU

Timers, EEPROM, A/D, UART

Ports I/O

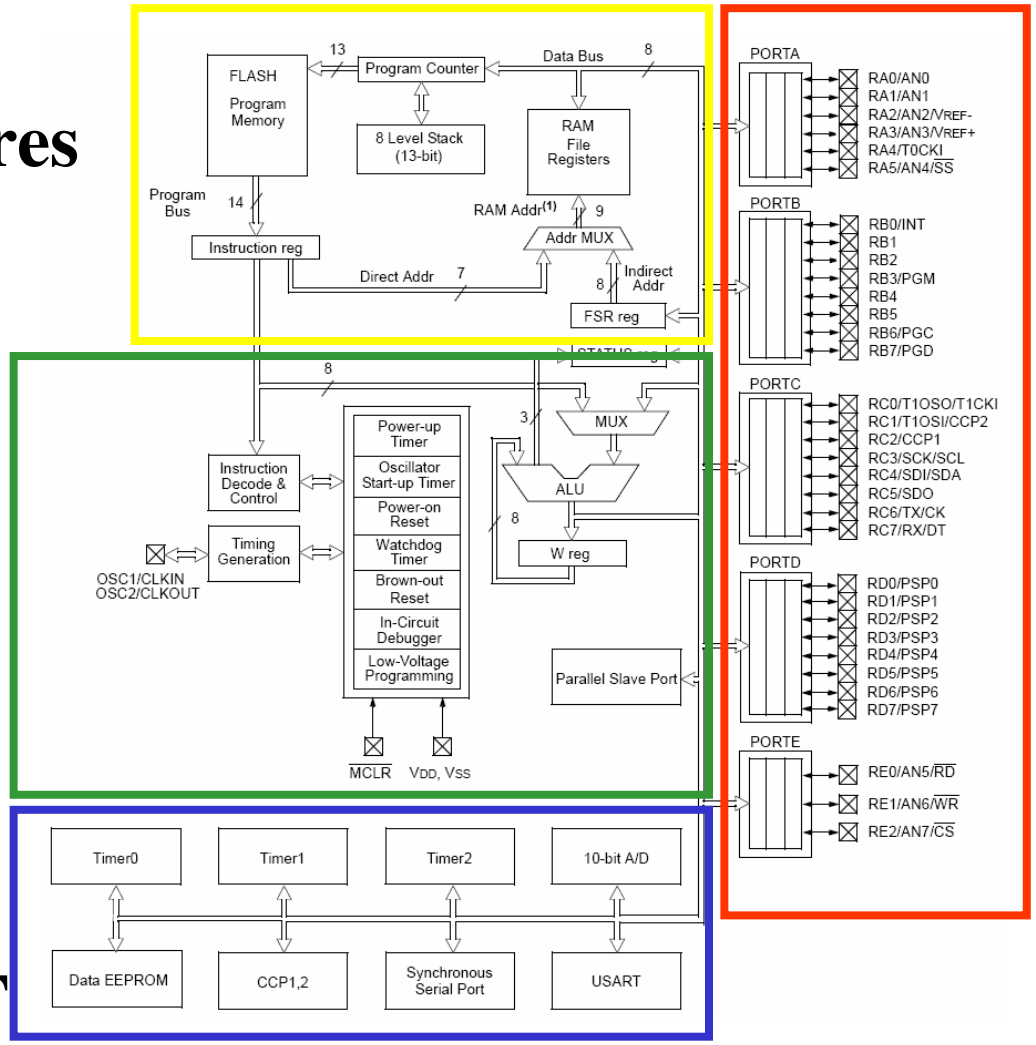
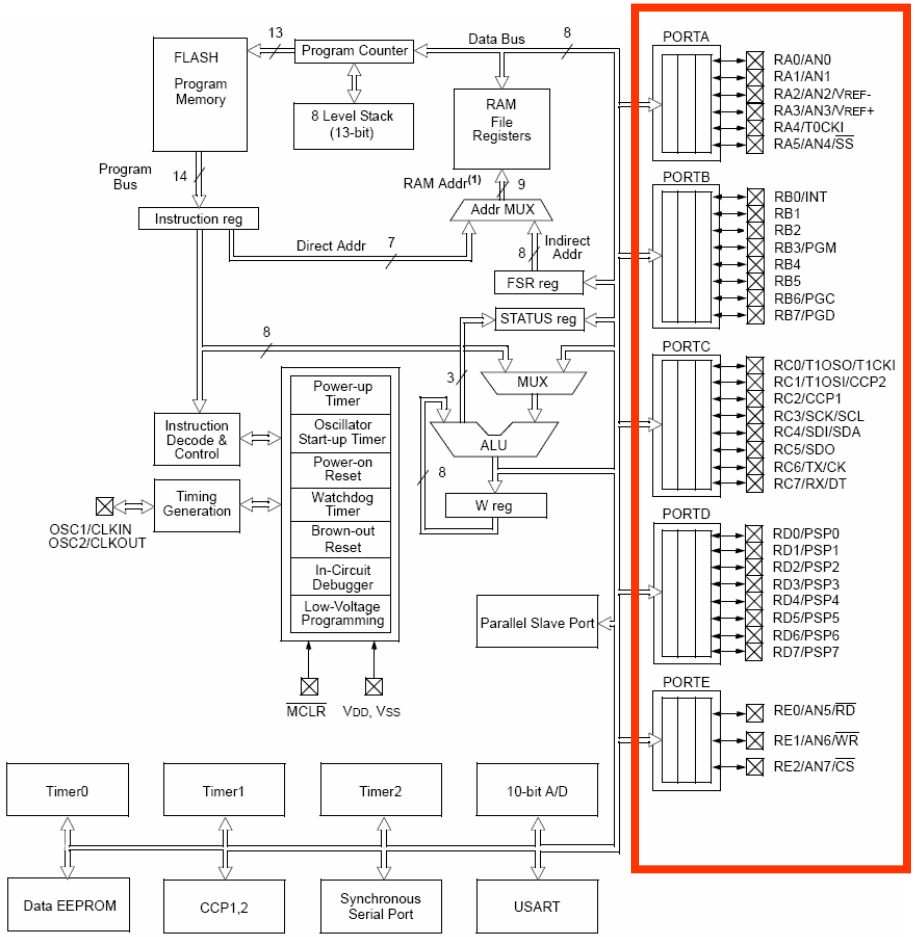


Schéma bloc PIC16F877



- 5 ports I/O (A → E)
- ≠ fonctions : ADC/DAC, oscillateurs ext., port série (TX/RX), port PSP, debugger ICD,…

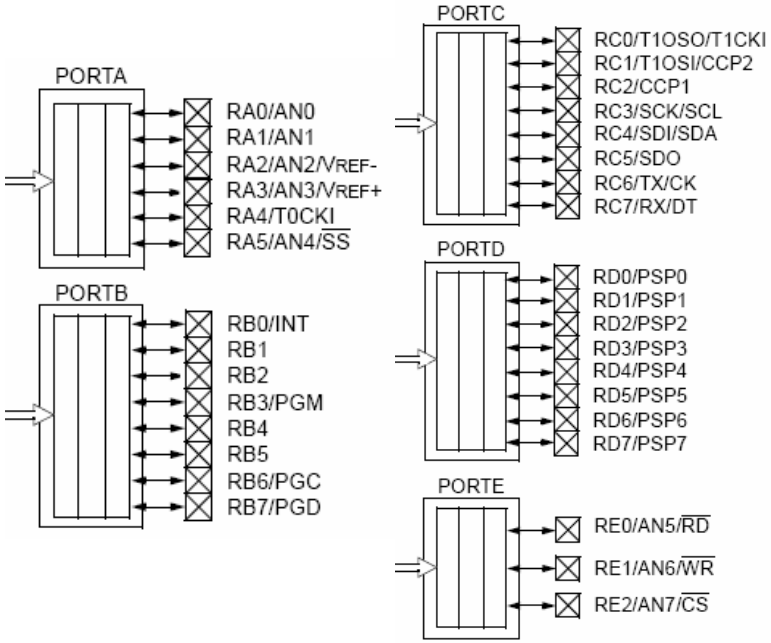
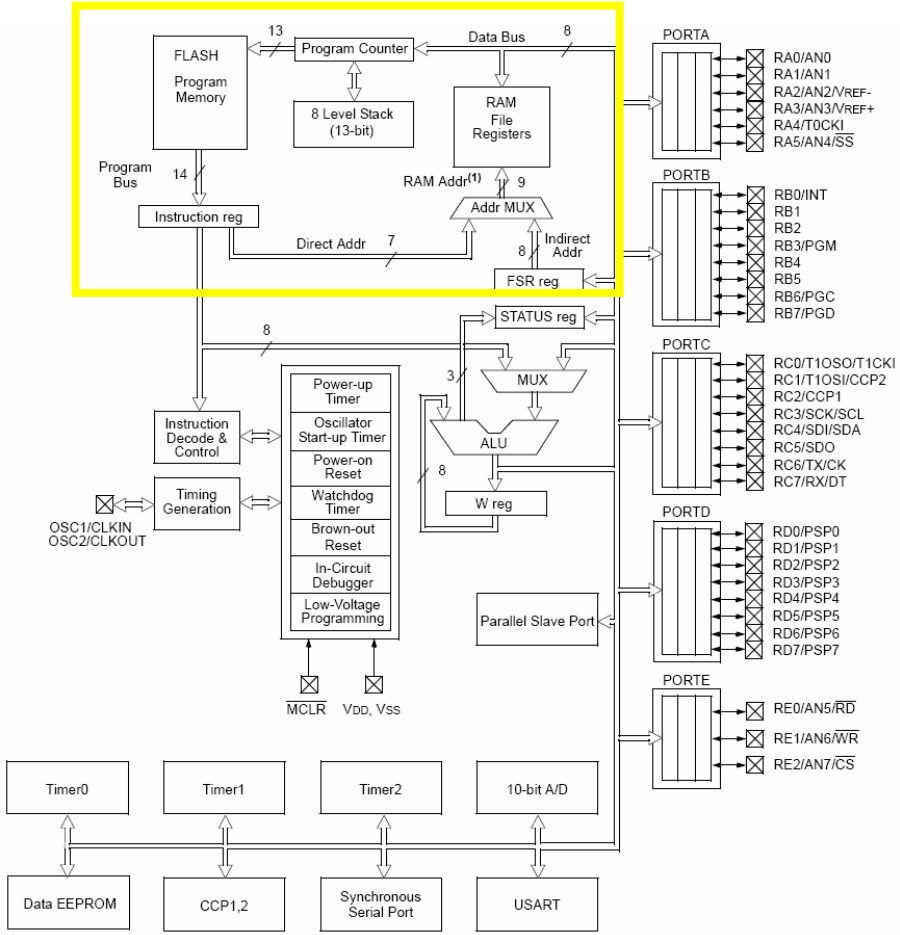


Schéma bloc PIC16F877



- Programme : Flash de 8k (mots de 14 bits)
- RAM : 368 bytes
- Pile matérielle (stack) : → jusqu'à 8 fonctions imbriquées

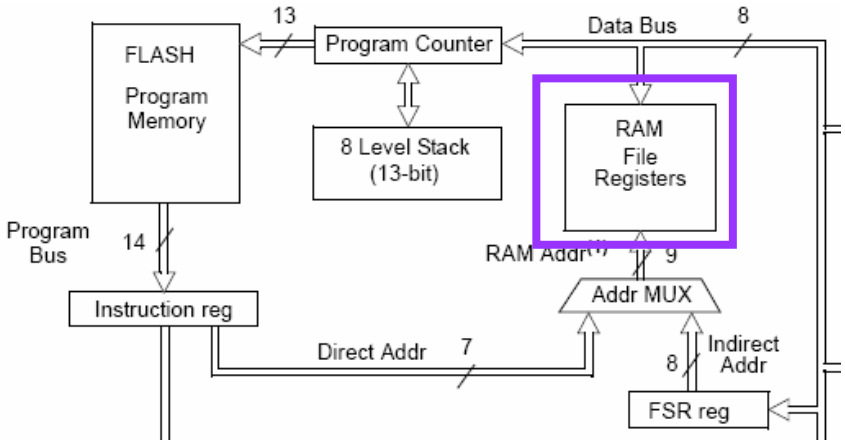


Schéma bloc PIC16F877 - RAM

- 4 banques
  - RP0, RP1 de STATUS
- → registres de
  - **configuration**
  - **données**

  1 ADR = 1 mot de 8 bits !

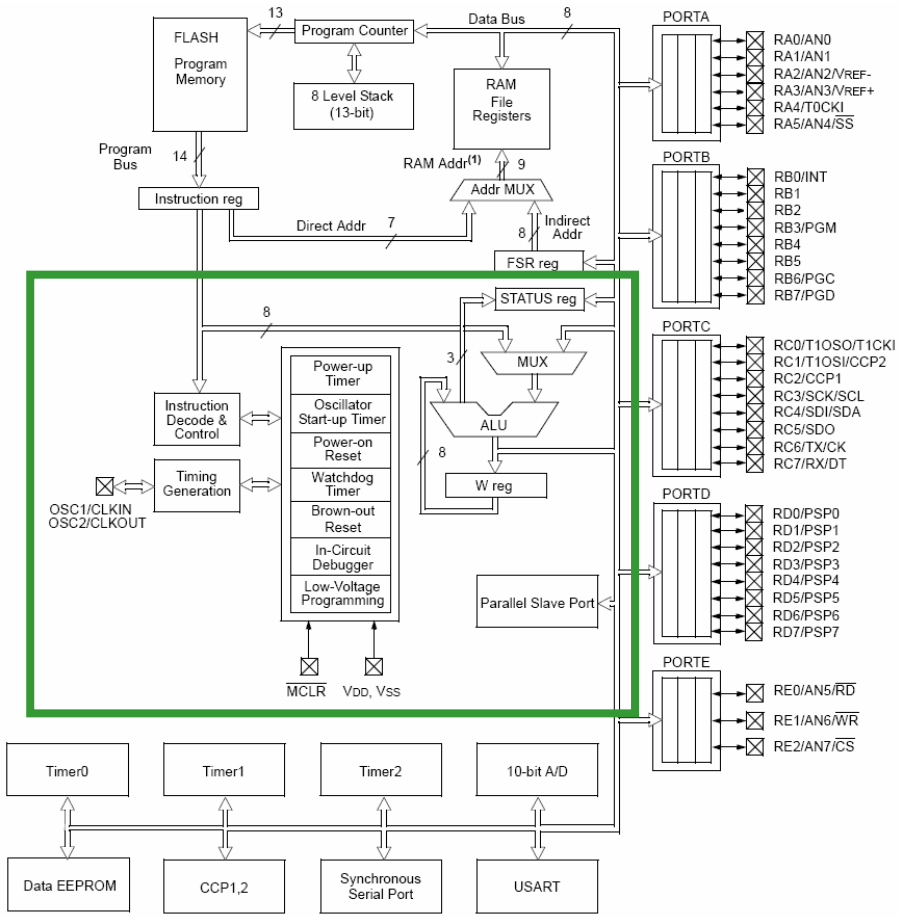
  Certains registres dupliqués dans les autres banques

  Ex : STATUS, PCL,…

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

Unimplemented data memory locations, read as '0'.
* Not a physical register.

Schéma bloc PIC16F877



- Décode/contrôle les instructions (CLK)
- ALU → W et STATUS

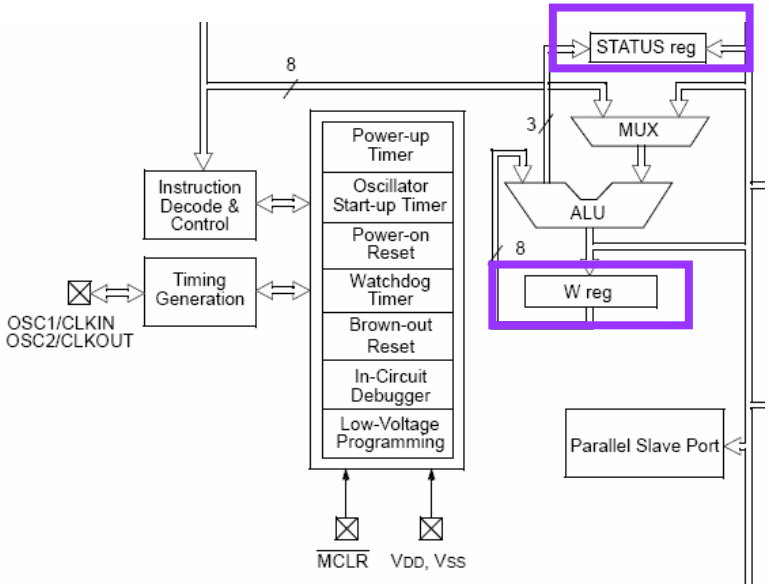
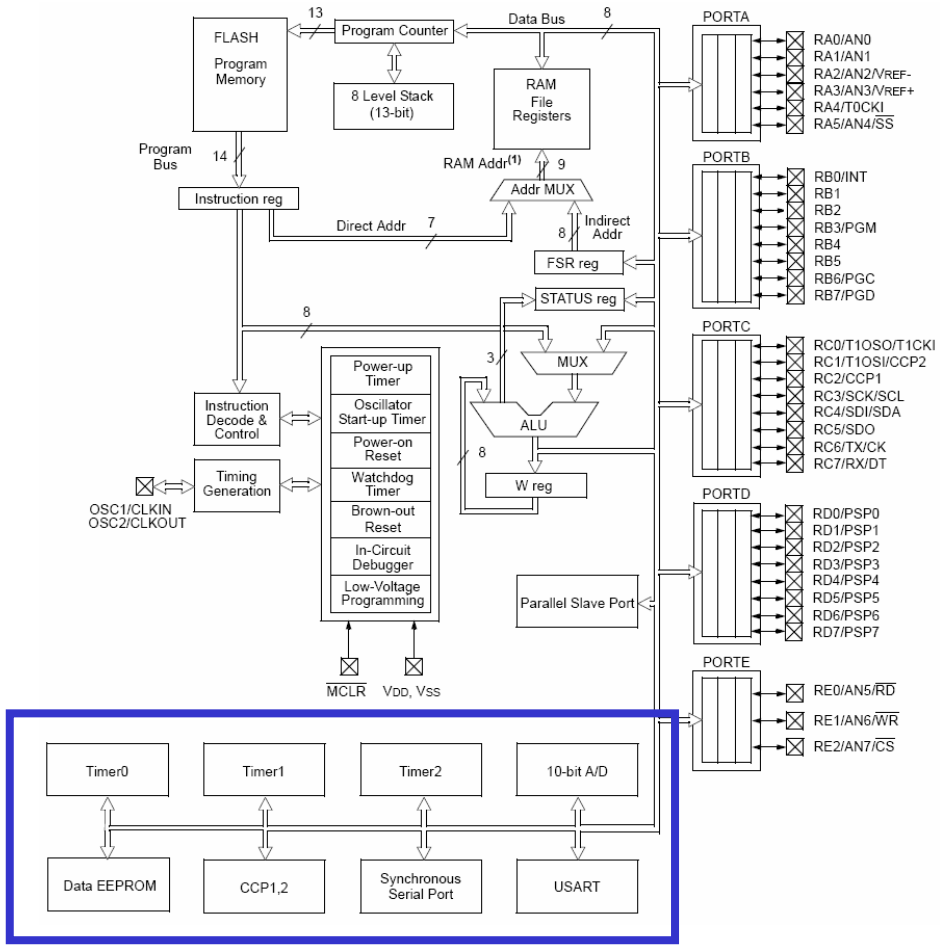
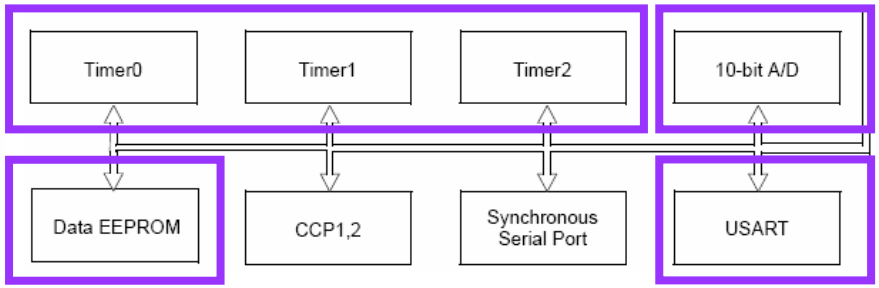


Schéma bloc PIC16F877



- 3 Timers : cpt, INTR
- A/D : 10 bits
- USART : série RS232
- EEPROM : nbre de cycles W limitée à 10^6



Programmation d'un PIC

- Logiciel : MPLAB (www.microchip.com)
- Langage :
  - ASM (assembleur)
  - C (compilateur PICC, C30, … selon le PIC)
- Simulation du code via MPASM
- Programmation :
  - ICD2 (In-Circuit Debugger), RS-232
  - module dédié comme le « dataman48 »

Manip1 - MPLAB IDE v7.60

File Edit View Project Debugger Programmer Tools Configure Window Help

Release | Checksum: 0xe76c

Manip1.mcw

- Manip1.mcp
  - Source Files
    - Led_cli_OK.asm
  - Header Files
  - Object Files
  - Library Files
  - Linker Scripts
  - Other Files

Projet

C:\...\P16F877A.INC

```
;
;==================================================

W                       EQU     H'0000'
F                       EQU     H'0001'

;----- Register Files------------------------------

INDF                    EQU     H'0000'
TMR0                    EQU     H'0001'
PCL                     EQU     H'0002'
STATUS                  EQU     H'0003'
FSR                     EQU     H'0004'
PORTA                   EQU     H'0005'
PORTB                   EQU     H'0006'
PORTC                   EQU     H'0007'
PORTD                   EQU     H'0008'
PORTE                   EQU     H'0009'
PCLATH                  EQU     H'000A'
INTCON                  EQU     H'000B'
PIR1                    EQU     H'000C'
PIR2                    EQU     H'000D'
TMR1L                   EQU     H'000E'
TMR1H                   EQU     H'000F'
T1CON                   EQU     H'0010'
TMR2                    EQU     H'0011'
T2CON                   EQU     H'0012'
SSPBUF                  EQU     H'0013'
SSPCON                  EQU     H'0014'
CCPR1L                  EQU     H'0015'
CCPR1H                  EQU     H'0016'
```

.INC = définitions

C:\Manips\source_MPLAB\Pic-ASM\Led_cli_OK.asm

```
;*********************************************************************
;   PROGRAMME DE CLIGNOTEMENT D'UNE LED CONNECTEE SUR LE PORTB.1      *
;   D'UNE PIC16F877. PROGRAMME D'ENTRAINEMENT AU FONCTIONNEMENT       *
;   DES PICS.                                                         *
;*********************************************************************
;                                                                     *
;    NOM:       LED-CLI                                               *
;    Date:                                                            *
;    Version:   1.0                                                   *
;    Circuit:   Carte de développement Microchip                      *
;    Auteur:                                                          *
;*********************************************************************
;                                                                     *
;    Fichier requis: P16F877.inc                                      *
;*********************************************************************
;                                                                     *
;    Notes: Ce petit programme permet de faire clignoter une LED      *
;           sur le port B.1 à une fréquence de 1Hz                    *
;           Ce programme fait partie de la leçon 6 des cours          *
;*********************************************************************

    LIST      p=16F877A            ; Définition de processeur
    #include <p16F877A.inc>        ; Définitions de variables
```

.ASM/.C = code source

Program Memory

Line	Address	Opcode	Disassembly
1	0000	2801	GOTO 0x1
2	0001	0186	CLRF 0x6
3	0002	1683	BSF 0x3, 0x5
4	0003	3088	MOVLW 0x88
5	0004	0081	MOVWF 0x1
6	0005	1086	BCF 0x6, 0x1
7	0006	1283	BCF 0x3, 0x5

Watch

Add SFR | ADCON0 | Add Symbol | __16F877A

Address	Symbol Name	Value
020	cmpt1	0x00
021	cmpt2	0x00
022	cmpt3	0x00

Visualisation des mém. programmes et de données (→ simul.)

MPLAB ICD 2 | PIC16F877A | W:0 | z dc c | bank 0

Exemple de code

```
;*********************************************************************
;                              CONFIGURATION                          *
;*********************************************************************

    LIST      p=16F877A           ; Définition de processeur
    #include <p16F877A.inc>       ; Définitions de variables

    __CONFIG   _CP_OFF & _DEBUG_OFF & _WRT_OFF & _CPD_OFF & _LVP_OFF & _BODEN_OFF &
    _PWRTE_ON & _WDT_OFF & _HS_OSC

;*********************************************************************
;                              ASSIGNATIONS                           *
;*********************************************************************

OPTIONVAL EQU       H'0088'   ; Valeur registre option
                              ; Résistance pull-up OFF
                              ; Pas de préscaler

;*********************************************************************
;                             DEFINE                                  *
;*********************************************************************

#DEFINE LED         PORTB,1             ; Led rouge
```

Exemple de code (2)

```
;*********************************************************************
;                              MACRO                                 *
;*********************************************************************

LEDON macro
          bsf       LED
    endm

LEDOFF macro
          bcf       LED
    endm

    bcf             STATUS,RP0          ; repasser banque 0
    goto  start                   ; sauter au programme principal
;*********************************************************************
;                    DECLARATIONS DE VARIABLES                       *
;*********************************************************************

    CBLOCK 0x020                  ; début de la zone variables

    cmpt1 : 1                     ; compteur de boucles 1
    cmpt2 : 1                     ; compteur de boucles 2
    cmpt3 : 1                     ; compteur de boucles 3

        ENDC                      ; Fin de la zone
```

Exemple de code (3)

```
;*********************************************************************
;                        DEMARRAGE SUR RESET                         *
;*********************************************************************

    org 0x000                   ; Adresse de départ après reset
    goto    init                ; Adresse 0: initialiser

;*********************************************************************
;                         INITIALISATIONS                            *
;*********************************************************************

init
    clrf   PORTB                ; sorties portB à 0
    bsf    STATUS,RP0           ; sélectionner banque 1
    movlw  OPTIONVAL            ; charger masque
    movwf  OPTION_REG           ; initialiser registre option

                        ; initialisations spécifiques
                        ; ----------------------------
    bcf    LED          ; LED en sortie (banque1)
    ; LEDOFF             ; ou utiliser LEDOFF

; etc ...
END                     ; fin code
```

Registres importants
DATASHEET pages 17-22 du .pdf

- W : registre de travail
- STATUS
  - **IRP** : sélection de la banque (ADR indirect)
  - **RP1:RP0** : sélection de la banque (ADR direct)
  - **TO** : time-out bit
  - **PD** : power-down
  - **Z** : zero bit
  - **DC** : digital carry
  - **C** : carry

Registres importants (2)

- OPTION_REG
  - **RBPU** : PORTB pull-up (on/off)
  - **INTEDGS** : INT edge (montante/descendante)
  - **T0CS** : T0 clock source (RB4/CLK int)
  - **T0SE** : T0 source edge (montante/descendante)
  - **PSA** : pre-scaler on T0 ou WD
  - **PS2:PS0** : valeur prescaler

Registres importants (3)

- INTCON
  - **GIE:** Global Interrupt Enable
  - **PEIE**: Peripheral Interrupt Enable
  - **T0IE**: TMR0 Overflow Interrupt Enable
  - **INTE**: RB0/INT External Interrupt Enable
  - **RBIE**: RB Port Change Interrupt Enable
  - **T0IF**: TMR0 Overflow Interrupt Flag
  - **INTF**: RB0/INT External Interrupt Flag
  - **RBIF**: RB Port Change Interrupt Flag

Directives

ELLES NE SONT PAS TRADUITES EN OPCODE !

- LIST : indique le type de processeur
- #include : indique les fichiers où sont regroupées les assignations
- CONFIG : fixe le fonctionnement du PIC à plusieurs niveaux (debugging, watchdog, protection du code, oscillateurs,…)

→ voir fichier datasheet et *.INC pour l'utilisation et la définition de ces variables de configuration

- EQU : remplacer un nombre par une chaîne de caractères

  Syntaxe : *assignations* EQU *nombre_a_remplacer*

- #DEFINE : remplacer un texte plus complexe par une chaîne de caractères.

  Syntaxe : *definition chaîne_a_substituer*

Directives (2)

- MACROS : remplacer des mots de code par une chaîne de caractères.

  Syntaxe : *nom_de_macro* MACRO
  code_de_macro
  endm

- CBLOCK : suivi d'une adresse, permet de définir le début d'une zone de variables.

  Syntaxe : CBLOCK *adresse_de_debut_de_zone*
  nom_de_variable : taille_de_zone (en bytes)
  …
  ENDC

- ORG : suivi d'une adresse, précise à quelle adresse les instructions qui suivent seront placées dans le PIC.
- END : fin du programme

Instructions

ELLES SONT TRADUITES EN OPCODE !

- Syntaxe : *étiquette* |_| *mnémonique opérande destination*
  - Etiquette : repère pour le programme
  - Tabulation (= |_|) : il est interdit d'écrire autre chose qu'une étiquette en première colonne.
  - Mnémonique : symbole littéral = instruction
  - Opérandes
  - Destination : W, F ou numéro de 0 à 7

BYTE-ORIENTED FILE REGISTER OPERATIONS									
ADDWF	f, d	Add W and f	1	00	0111	dfff	ffff	C,DC,Z	1,2
ANDWF	f, d	AND W with f	1	00	0101	dfff	ffff	Z	1,2
CLRF	f	Clear f	1	00	0001	1fff	ffff	Z	2
CLRW	-	Clear W	1	00	0001	0xxx	xxxx	Z	
COMF	f, d	Complement f	1	00	1001	dfff	ffff	Z	1,2
DECF	f, d	Decrement f	1	00	0011	dfff	ffff	Z	1,2
DECFSZ	f, d	Decrement f, Skip if 0	1(2)	00	1011	dfff	ffff		1,2,3
INCF	f, d	Increment f	1	00	1010	dfff	ffff	Z	1,2
INCFSZ	f, d	Increment f, Skip if 0	1(2)	00	1111	dfff	ffff		1,2,3
IORWF	f, d	Inclusive OR W with f	1	00	0100	dfff	ffff	Z	1,2
MOVF	f, d	Move f	1	00	1000	dfff	ffff	Z	1,2
MOVWF	f	Move W to f	1	00	0000	1fff	ffff		
NOP	-	No Operation	1	00	0000	0xx0	0000		
RLF	f, d	Rotate Left f through Carry	1	00	1101	dfff	ffff	C	1,2
RRF	f, d	Rotate Right f through Carry	1	00	1100	dfff	ffff	C	1,2
SUBWF	f, d	Subtract W from f	1	00	0010	dfff	ffff	C,DC,Z	1,2
SWAPF	f, d	Swap nibbles in f	1	00	1110	dfff	ffff		1,2
XORWF	f, d	Exclusive OR W with f	1	00	0110	dfff	ffff	Z	1,2
BIT-ORIENTED FILE REGISTER OPERATIONS									
BCF	f, b	Bit Clear f	1	01	00bb	bfff	ffff		1,2
BSF	f, b	Bit Set f	1	01	01bb	bfff	ffff		1,2
BTFSC	f, b	Bit Test f, Skip if Clear	1 (2)	01	10bb	bfff	ffff		3
BTFSS	f, b	Bit Test f, Skip if Set	1 (2)	01	11bb	bfff	ffff		3
LITERAL AND CONTROL OPERATIONS									
ADDLW	k	Add literal and W	1	11	111x	kkkk	kkkk	C,DC,Z	
ANDLW	k	AND literal with W	1	11	1001	kkkk	kkkk	Z	
CALL	k	Call subroutine	2	10	0kkk	kkkk	kkkk		
CLRWDT	-	Clear Watchdog Timer	1	00	0000	0110	0100	$\overline{TO}$,$\overline{PD}$	
GOTO	k	Go to address	2	10	1kkk	kkkk	kkkk		
IORLW	k	Inclusive OR literal with W	1	11	1000	kkkk	kkkk	Z	
MOVLW	k	Move literal to W	1	11	00xx	kkkk	kkkk		
RETFIE	-	Return from interrupt	2	00	0000	0000	1001		
RETLW	k	Return with literal in W	2	11	01xx	kkkk	kkkk		
RETURN	-	Return from Subroutine	2	00	0000	0000	1000		
SLEEP	-	Go into standby mode	1	00	0000	0110	0011	$\overline{TO}$,$\overline{PD}$	
SUBLW	k	Subtract W from literal	1	11	110x	kkkk	kkkk	C,DC,Z	
XORLW	k	Exclusive OR literal with W	1	11	1010	kkkk	kkkk	Z	

Exemples

- ADDWF : *ADDWF f,d*

  $W+f \rightarrow W$ (d=0) ou $\rightarrow f$ (d=1); [C,Z,DC]

- BCF : *BCF f,b*

  f[b]=0 ; [-]

- CALL : *CALL sous-routine*

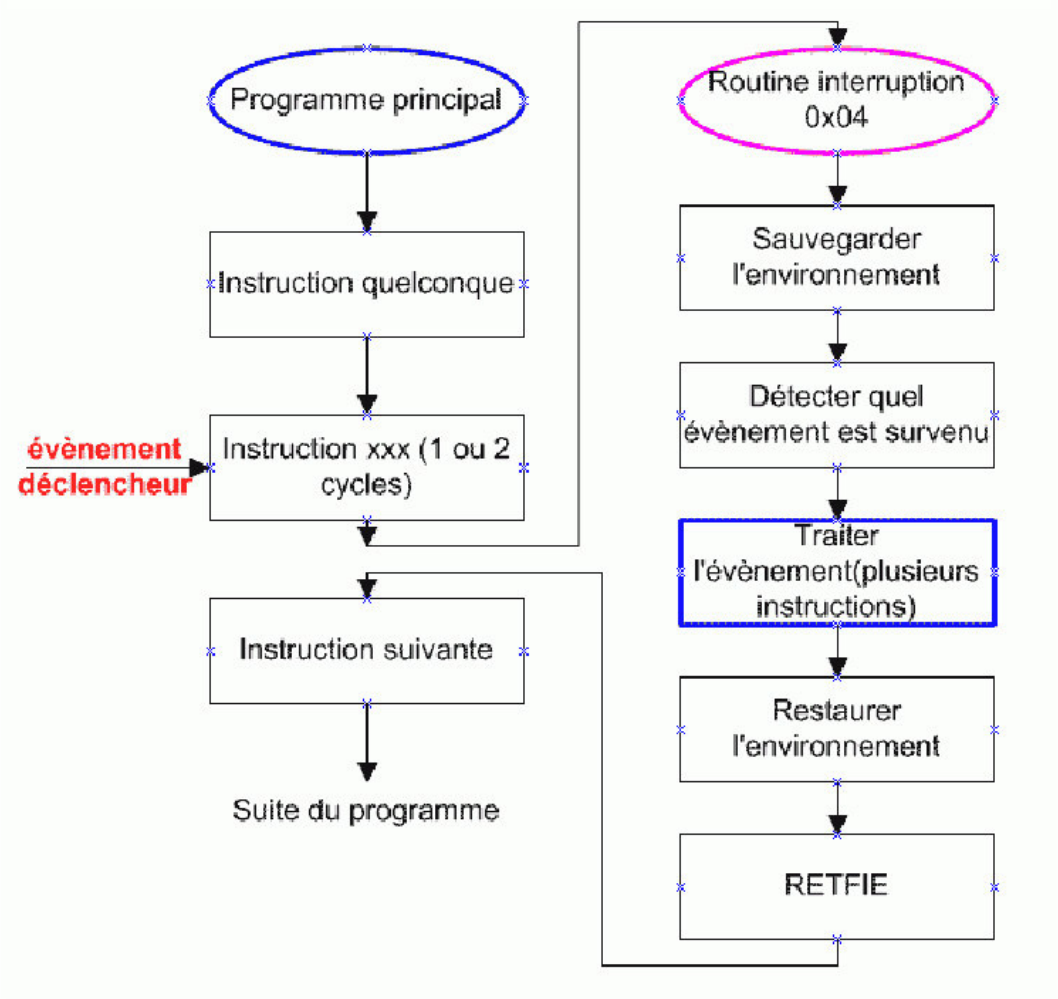
  appel de *sous-routine* (pile utilisée⇒retour)

- GOTO : GOTO *étiquette*

  saut inconditionnel (pile non-utilisée)

Interruptions (INTR)

- INTR = rupture du déroulement normal par un évènement dé-clencheur → exécution d'une routine d'INTR



Interruptions (2)

- Sources (1 seule à la fois !)
  - EEPROM, T0, RB0/INT, PORTB
- ADR d'INTR unique : 0x04
- Sauvegarde de l'état du système :

  Uniquement le PC est sauvé lors du saut vers la routine d'interruption → le reste du contexte (W, STATUS,…) doit être sauvé manuellement

Exemple de code

```
;*********************************************************************
;                        ROUTINE D'INTERRUPTION                      *
;*********************************************************************
                    ; sauvegarder des registres
                    ; --------------------------
    org 0x00                    ; Adresse d'interruptions
    movwf w_temp                ; sauvegarder le registre de travail
    swapf STATUS,w              ; swap STATUS avec w
    movwf status_temp           ; sauvegarder le registre de STATUS
                     ; switch vers différentes INTR
                     ; ----------------------------
; ici on teste éventuellement pour savoir d'où vient l'INTR
                     ; switch vers différentes INTR
                     ; ----------------------------
; ici on peut traiter l'INTR et effacer son FLAG
                     ; restauration des registres
                     ; --------------------------
    swapf status_temp,w         ; swap ancien STATUS, dans w
    movwf STATUS                ; restaurer STATUS
    swapf w_temp,f              ; inversion L et H de l'ancien W
                                ; sans mofifier Z
    swapf w_temp,w              ; ré-inversion de L et H dans W
                                ; W restauré sans modifier STATUS
    retfie                      ; return from interrupt
```

En résumé

- Un microcontrôleur est un composant programmable polyvalent
  - Processeur (ALU)
  - Mémoires programme, RAM et EEPROM
  - Périphériques (A/D, UART,…)
  - Gestion d'interruptions
- Programmation en ASM ou C (→ MPLAB)

Derniers messages

- Comprendre le fonctionnement d'un microcontrôleur
  - DATASHEET !!
  - Lecture du *BigOnOFF* sur le 16F84 (excellente base !!) ou des notes
  - Lecture du *BigOnOFF* sur le 16F877 (uniquement les différences avec le 16F84)
- Programmer un *µctrl* 16F877 au labo (chez soi)
- !! Configuration !!