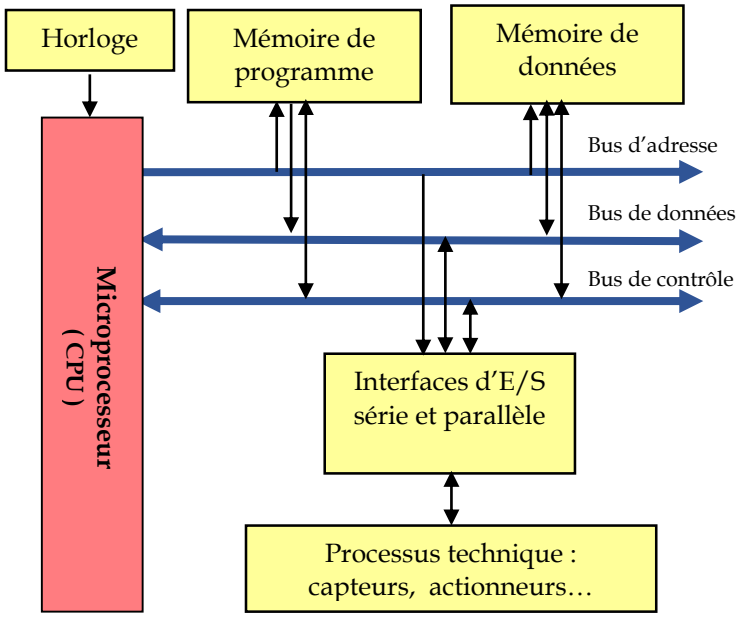
SYSTEMES PROGRAMMABLES

Environnement microinformatique minimal

Le contrôle d'un processus technique fait souvent appel à un dispositif de traitement programmable

Un système programmable comporte deux éléments complémentaires indissociables :

- Logiciel : le programme qui décrit la gestion du processus technique
- Matériel : le dispositif électronique, à base de µP, qui héberge le programme et lui donne vie



Ci-contre, la structure d'un système programmable minimal :

1. *Microprocesseur* (ou CPU pour Central Processing Unit)

Le µP exécute le programme situé dans la mémoire de programme .Toute l'activité du µP est cadencée par une horloge

2. *Les bus*

Un bus est ensemble de fils destiné à véhiculer les données entre les différents composants du système. On distingue :

- Le bus de données : à travers lequel transitent les données
- Le bus d'adresse : à travers lequel le µP applique l'adresse de la donnée à lire ou à écrire
- Le bus de contrôle : à travers lequel le µP dialogue avec le composant adressé

3. *Les mémoires*

Ce sont des composants chargés de conserver des contenus. En fonction des propriétés des mémoires, on peut les classer selon l'arborescence suivante :

	Lecture	écriture	volatile
RAM	Oui	oui	Oui
ROM	Oui	Non (programmée une seule fois au moment de la fabrication)	Non
PROM	Oui	Programmable une seule fois par l'utilisateur	Non
EEPROM	Oui	Effaçable électriquement à l'aide d'une broche spécifique et reprogrammable	Non

- Une mémoire volatile perd son contenu à la coupure de l'alimentation
- En principe, la RAM sert de mémoire de données tandis que la ROM sert de mémoire de programme
- Une variante de l'EEPROM est la mémoire FLASH qui est rapide et accepte d'être reprogrammée un grand nombre de fois

4. *Périphériques*

Ils ont pour rôle de dialoguer avec le processus à gérer

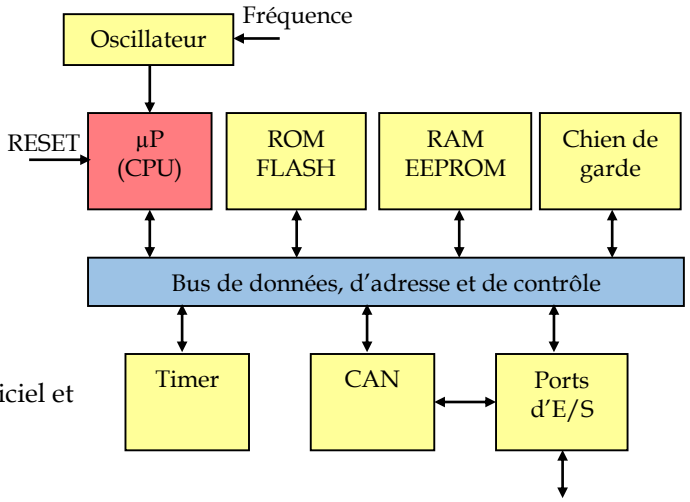
- ⇨ *Interface parallèle* : les données sont transmises par paquets de 8, 16 …bits
- ⇨ *Interface série* **:** les données sont transmises bit pat bit sur un fil unique (exemples : USB, RS232, RS485)

Le microcontrôleur PIC 16F877

Généralités

Le µC se présente sous la forme d'un circuit intégré réunissant tous les éléments d'une structure à base de µP ainsi que certains modules auxiliaires :

- Un µP
- De la mémoire de données (RAM, EEPROM)
- De la mémoire de programme (FLASH, EEPROM)
- Des ports pour la connexion des entrées/sorties
- Des timers pour générer des impulsions, compter des évènements, mesurer des fréquences…
- Des CAN pour le traitement des signaux analogiques
- Un chien de garde : qui détecte s'il y a un problème logiciel et provoque un RESET

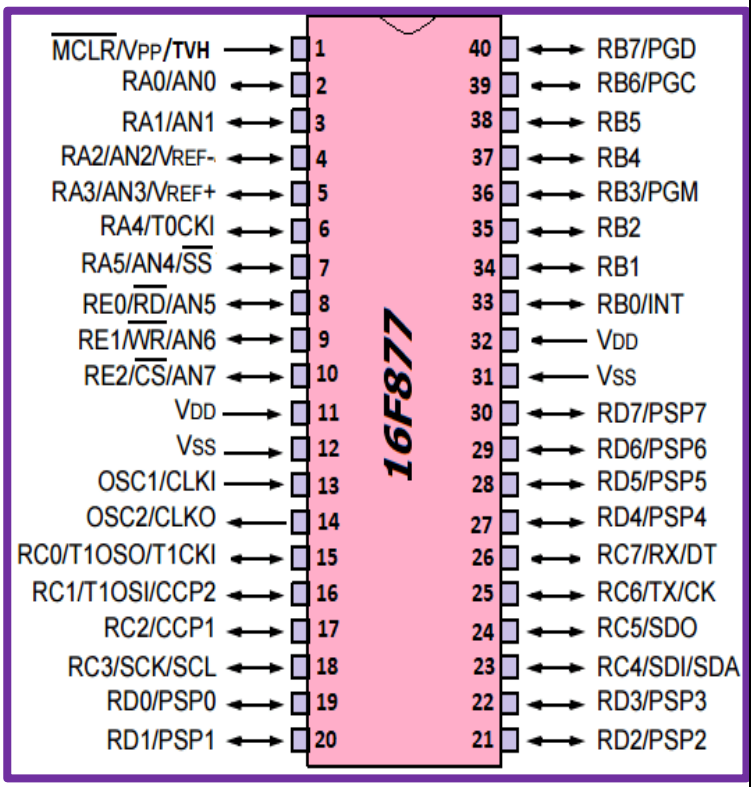


Aujourd'hui on trouve les µC presque partout : cafetière, réfrigérateur, four à micro ondes, téléviseur, téléphone portable, imprimante, scanner, voiture (airbags, climatisation, alarme…)…

Architecture externe (brochage) du PIC 16F877 de Microchip

La popularité du 16F877 vient de sa facilité de mise en œuvre et la disposition d'un environnement de développement complet totalement gratuit : MPLAB.

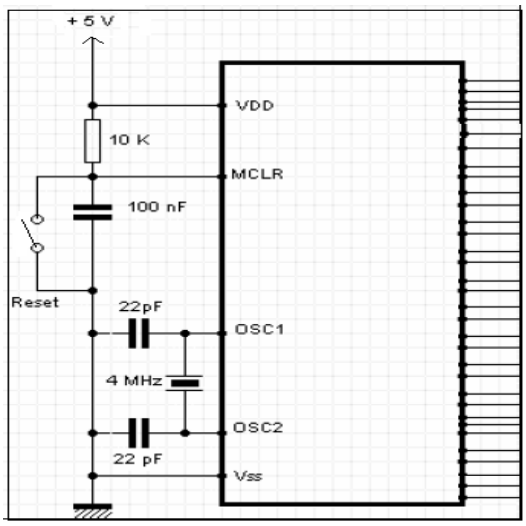
- L'alimentation du circuit est assurée par les pattes **VDD = +5V** et **VSS = 0V**
- **OSC1** et **OSC2** : broches qui reçoivent un quartz ou un circuit RC pour l'horloge interne
- **MCLR** (Master Clear Reset) : broche de remise à zéro, active au niveau bas (0 V)
- **RA0 à RA5** : 6 broches E/S du port A
  RB0 à RB7 : 8 broches E/S du port B
  RC0 à RC7 : 8 broches E/S du port C
  RD0 à RD7 : 8 broches E/S du port D
  RE0 à RE2 : 3 broches E/S du port E



Mise en œuvre (Montage minimum)

Ici un montage minimum avec :

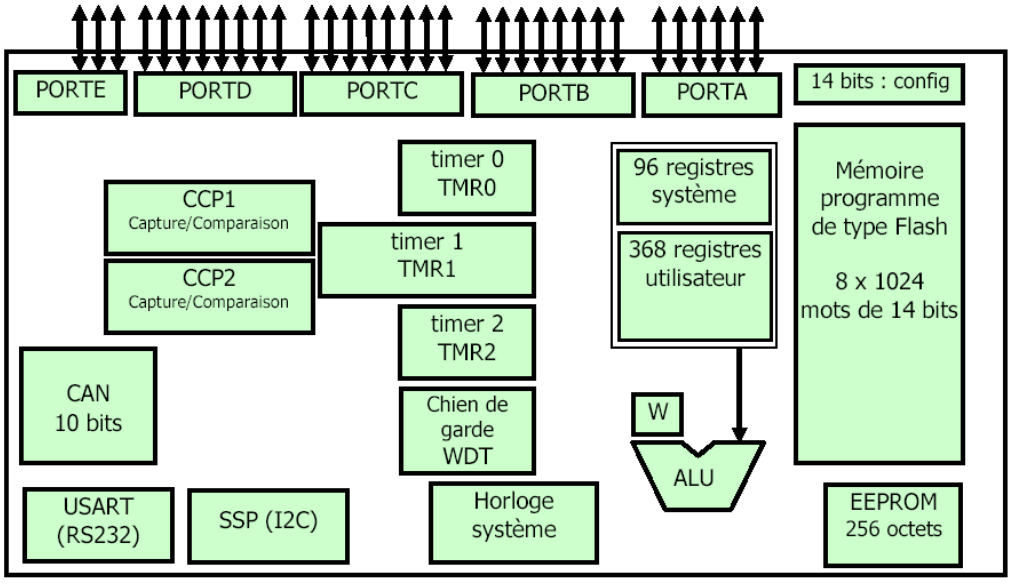
- Alimentation du composant (VDD = +5V et VSS = 0V)
- Pilotage par quartz, ici de 4 MHz (OSC1 et OSC2)
- RESET manuel par un bouton poussoir
- RESET automatique à chaque mise sous tension. En effet, le condensateur initialement déchargé (MCLR = 0), se charge jusqu'à VDD (MCLR = 1)



Architecture interne

Les caractéristiques principales du 16F877 sont :

- Une mémoire programme de type flash de 8K mots de 14 bits de h'0000' à h'1FFF' (1K = 2^{10} = 1024)
- Une mémoire RAM constituée :
  - 96 registres de configuration ayant des fonctions spéciales
  - 368 registres d'espace utilisateur
- Une mémoire EEPROM Data de données de 256 octets
- 3 Timers avec prédiviseur programmable : 2 timers 8 bits TMR0, TMR2 et un autre 16 bits TMR1
- Un chien de garde WDT
- CAN 10 bits à approximations successives à 8 entrées multiplexées
- 33 Entrée-Sortie bidirectionnelles réparties sur 5 ports : PORTA à 6 bits, PORTB, PORTC, PORTD à 8 bits et PORTE à 3 bits
- 1 registre de travail (accumulateur) W à 8 bits.
- Bus DATA sur 8 bits.
- Jeu de 35 instructions de durée 1 ou 2 cycles.
- Plusieurs sources d'interruption
- Autres modules : USART pour transmission série, interface I2C, 2 modules pour PWM…



La RAM de données

Page 0		Page 1		Page2		Page 3	
00	INDF	80	INDF	100	INDF	180	INDF
01	TMR0	81	OPTION	101	TMR0	181	OPTION
02	PCL	82	PCL	102	PCL	182	PCL
03	STATUS	83	STATUS	103	STATUS	183	STATUS
04	FSR	84	FSR	104	FSR	184	FSR
05	PORTA	85	TRISA	105		185	
06	PORTB	86	TRISB	106	PORTB	186	TRISB
07	PORTC	87	TRISC	107		187	
08	PORTD	88	TRISD	108		188	
09	PORTE	89	TRISE	109		189	
0A	PCLATH	8A	PCLATH	10A	PCLATH	18A	PCLATH
0B	INTCON	8B	INTCON	10B	INTCON	18B	INTCON
0C	PIR1	8C	PIE1	10C	EEDATA	18C	EECON1
0D	PIR2	8D	PIE2	10D	EEADR	18D	EECON2
0E	TMR1L	8E	PCON	10E	EEDATH	18E	
0F	TMR1H	8F		10F	EEADRH	18F	
10	T1CON	90		110		190	
11	TMR2	91	SSPCON2				
12	T2CON	92	PR2				
13	SSBUF	93	SSPADD				
14	SSPCON	94	SSPSTAT				
15	CCPR1L	95					
16	CCPR1H	96			RAM 16 octets		RAM 16 octets
17	CCP1CON	97					
18	RCSTA	98	TXSTA				
19	TXREG	99	SPBRG				
1A	RCREG	9A					
1B	CCPR2L	9B					
1C	CCPR2H	9C					
1D	CCP2CONL	9D					
1E	ADRESH	9E	ADRESL				
1F	ADCON0	9F	ADCON1	11F		19F	
20		A0		120		1A0	
	RAM 96 octets		RAM 80 octets		RAM 80 octets		RAM 80 octets
		EF		16F		1EF	
		F0		170		1F0	
7F		FF		17F		1FF	

Programmation du PIC 16F877 par le langage assembleur

Jeu d'instructions du 16F877

{W, F ? d} signifie que le résultat va soit dans W si d=0 ou w, soit dans F si d= 1 ou f

INSTRUCTIONS OPERANT SUR UN REGISTRE			Indicateurs	Cycles
ADDWF	**F, d**	W+F → {W, F ? d}	C, DC, Z	1
ANDWF	**F, d**	W AND F → {W, F ? d}	Z	1
CLRF	**F**	0 → F	Z	1
CLRW		0 → W	Z	1
CLRWDT		0 → WDT	TO', PD'	1
COMF	**F, d**	Complémente F → {W, F ? d}	Z	1
DECF	**F, d**	Décrémente F → {W, F ? d}	Z	1
DECFSZ	**F, d**	Décrémente F → {W, F ? d} et saut si 0		1(2)
INCF	**F, d**	Incrémente F → {W, F ? d}	Z	1
INCFSZ	**F, d**	Incrémente F → {W, F ? d} et saut si 0		1(2)
IORWF	**F, d**	W OR F → {W, F ? d}	Z	1
MOVF	**F, d**	F → {W, F ? d}	Z	1
MOVWF	**F**	W → F		1
RLF	**F, d**	Rotation à gauche de F à travers C → {W, F ? d}	C	1
RRF	**F, d**	Rotation à droite de F à travers C → {W, F ? d}		1
SUBWF	**F, d**	F – W → {W, F ? d}	C, DC, Z	1
SWAPF	**F, d**	Permute les 2 quartets de F → {W, F ? d}		1
XORWF	**F, d**	W XOR F → {W, F ? d}	Z	1
INSTRUCTIONS OPERANT SUR UN BIT				
BCF	**F, b**	Mise à 0 du bit b du registre F		1
BSF	**F, b**	Mise à 1 du bit b du registre F		1
BTFSC	**F, b**	Teste le bit b de F et saut si 0		1(2)
BTFSS	**F, b**	Teste le bit b de F et saut si 1		1(2)
INSTRUCTIONS OPERANT SUR UNE DONNEE				
ADDLW	**K**	W + K → W	C, DC, Z	1
ANDLW	**K**	W AND K → W	Z	1
IORLW	**K**	W OR K → W	Z	1
MOVLW	**K**	K → W		1
SUBLW	**K**	K – W → W	C, DC, Z	1
XORLW	**K**	W XOR K → W	Z	1
INSTRUCTIONS GENERALES				
CALL	**L**	Branchement à un sous-programme de label L		2
GOTO	**L**	Branchement à la ligne de label L		2
NOP		Pas d'opération		1
RETURN		Retour d'un sous-programme		2
RETFIE		Retour d'interruption		2
RETLW	**K**	Retour d'un sous-programme avec K dans W		2
SLEEP		Mode standby	TO', PD'	1

⇨ *Accès à la RAM*

Registre STATUS

IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C

Les bits **RP1** et **RP0** du registre **STATUS** permettent la sélection des pages de la RAM

RP1	*RP0*	*Page sélectionnée*
0	0	Bank 0
0	1	Bank 1
1	0	Bank 2
1	1	Bank 3

⇨ *Expression des nombres*

L'assembleur considère les nombres dans la base indiquée par la spécification adjointe

Base	*Indicateur*	*Exemple*
Décimal	d	d'27'
Hexadécimal	h 0x	h'1B' ou 0x1B 1Bh
Binaire	b	b'00011011'

Exercice

En utilisant le plan d'organisation de la RAM, trouver la page concernée

Registre	*Bank*
PORTA	…….
TRISA	…….
TMR0	…….
ADCON1	…….

Exercice

Question	Réponse
Sélectionner la page 0	BCF 0x03, 6 ou bien BCF STATUS, RP1 ……..... ……
Sélectionner la page 2	…… ……
Mettre à 0 le registre W	…… ; par CLRW autrement …… ; par AND ou encore …… ; charger la valeur 0 dans W

Incrémenter le registre W	……. ; copier W dans un registre, d'adresse ….. par exemple ……. ; Incrémenter le registre avec résultat dans W ou mieux ……. ; incrémenter W par addition
Charger le registre PORTB par le mot 00000001	…… ; RAZ du registre PORTB …… ; forcer le bit b_0 à 1 autrement ……. ; charger l'octet '00000001' dans W …… … ; copier W dans PORTB
Copier le contenu de PORTA dans PORTB	……. ; copier le contenu de PORTA dans W …… … ; copier W dans PORTB

⇨ ***Les ports d'entrées/ sortie***

PORTA : pin RA0 à RA5
PORTB : pin RB0 à RB7
PORTC : pin RC0 à RC7
PORTD : pin RD0 à RD7
PORTE : pin RE0 à RE2

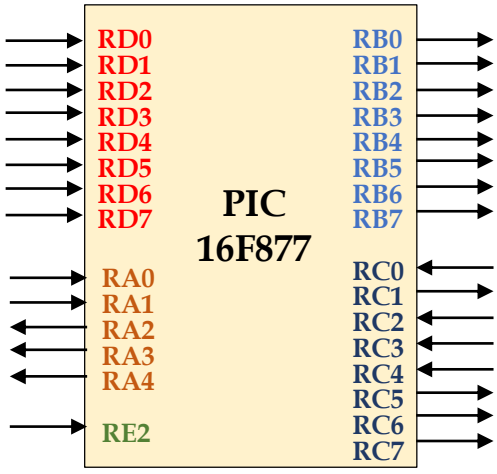
- La direction des données sur les ports est programmée à travers les registres TRISA, TRISB, TRISC, TRISD et TRISE (0 →sortie 1→ entrée)

Exemples : Mettre la valeur '00000011' dans TRISB configure les broches RB0 et RB1 en entrée et les autres en sortie

- La manipulation des données (lecture/écriture) sur les ports est programmée à travers les registres PORTA, PORTB, PORTC, PORTD et PORTE

Exemple : bsf PORTC, 7 → positionne la broche RC7 à 1 (écriture)
movwf PORTC → permet de sortir le contenu de w sur PORTC (écriture)
movf PORTC, w → récupère l'état de PORTC dans w (lecture)

Exercice *Donner le programme assembleur réalisant la configuration suivante :*

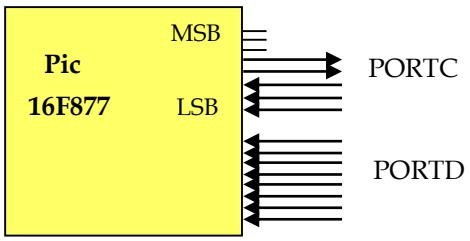


TRISA = ……
TRISB = ……
TRISC = ……
TRISD = ……
TRISE = ……

```
……                          ; activer banque 1
……
……                          ; configurer port B
……                          ; configurer port C
……
……                          ; configurer port D
……
……                          ; configurer port E
……                          ; configurer port A
……
……                          ; activer banque 0
……
```

Exercice

- *Configurer le port D ainsi que les 5 bits de poids faible du port C conformément à la figure*
- *Allumer les 2 LED supposées connectées aux broches RC3 et RC4 de PORTC*
  Lire le contenu de PORTD pour le stocker dans le registre d'adresse 7F



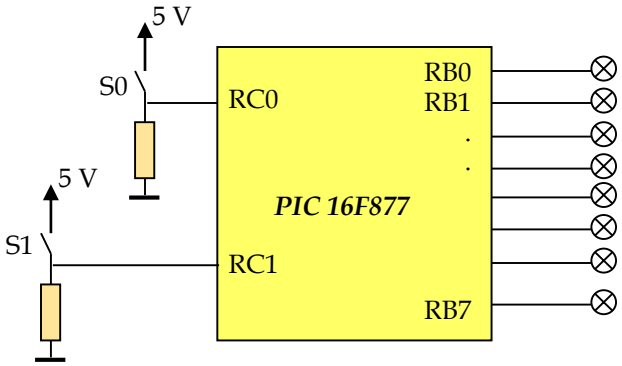
TRISC	……
TRISD	……

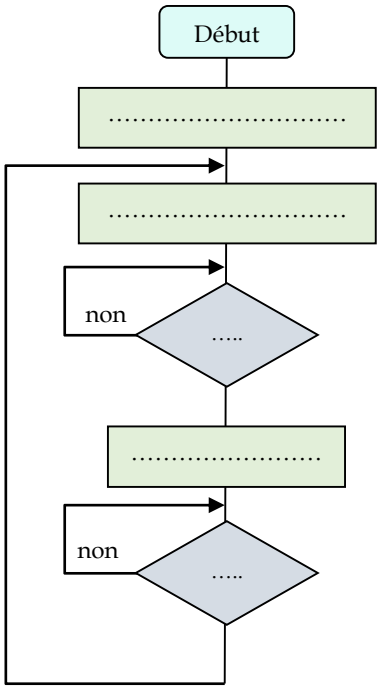
```
……                          ; activer  page 1
……
……                          ; configurer le port C
……
……                          ; configurer le port D
……
……                          ; retour à page 0
……
……                          ; allumer les 2 Leds du portC
……
……                          ; lire et ranger la valeur du port D
……                          ; dans le registre 7F
```

Exercice

Au repos les LEDs sont éteintes.
Une action sur le bouton poussoir S1 allume toutes les LEDs.
Une action sur le bouton poussoir S0 éteint toutes les LEDs.

Fournir l'organigramme et le programme





TRISB	……
TRISC	……

```
        .....                   ; page 1
        .....
        .....                   ; configurer  PORTB en sortie
        .....                   ; configurer  PORTC
        .....
        .....                   ; page 0
        .....

boucle1 .....                   ; éteindre les LEDs
boucle2 .....                   ; Si RC1 = 1 (signifiant si S1 appuyé)
        .....                   ; sinon retester si RC1 = 1
        .....                   ; si oui allumer les LEDs
        .....
boucle3 .....                   ; Si RC0 = 1
        .....                   ; sinon retester si RC0 = 1
        .....                   ; Se brancher à  'boucle1' pour éteindre de nouveau
```

Techniques de comparaison

Bits indicateurs d'état Z et C du registre STATUS

Le bit **Z** est l'indicateur **Zéro**; **Z = 1** si le résultat d'une opération arithmétique ou logique est **zéro**.
Z = 0 si le résultat est **différent de zéro**

Le bit **C** est l'indicateur de **signe**; **C = 1** si le résultat d'une soustraction est **positif**
C = 0 si le résultat est **négatif**

F1 et F2 étant deux registres, compléter ce tableau :

Question	Réponse
Si F1 = 0	..... ; copier F1 dans W puis contrôler le bit Z .....
Si F1 = F2	..... ; effectuer l'opération F1-F2 puis contrôler le bit Z ....
Si F1 < F2	..... ; effectuer l'opération F1-F2 puis contrôler le bit C ....
Si F1 ≥ 0	..... ; effectuer l'opération F1-0 puis contrôler le bit C ....

⇨ *Temporisation et sous-programme*

Un sous-programme est une séquence d'instructions chargée d'effectuer un traitement particulier.
Les sous-programmes permettent ainsi de décomposer un programme en plusieurs sous-parties

Programme principal

```
.....
.....
CALL X
.....
.....
CALL X
.....
```

Sous-programme

```
X .....
  .....
  .....
  RETURN
```

CALL : instruction d'appel du sous-programme
RETURN : instruction de retour au programme appelant

Exercice

Faire clignoter à, 1 Hz, les 8 LED (à la fois) supposées montées sur PORTB
On suppose déclaré un sous-programme "tempo" qui réalise une temporisation de 0,5 s

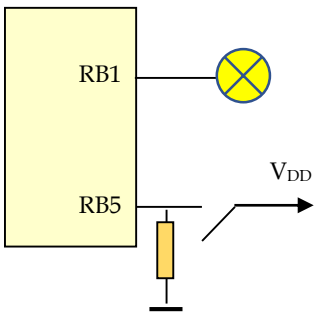
```
loop    ……..        ; Eteindre  les LED
        ……..        ; Temporiser 0,5 s
        ……..        ; Allumer les LED
        ……..
        ……..        ; Temporiser  0,5 s
        ……..        ; répéter
```

Variante utilisant l'instruction **comf F, d**

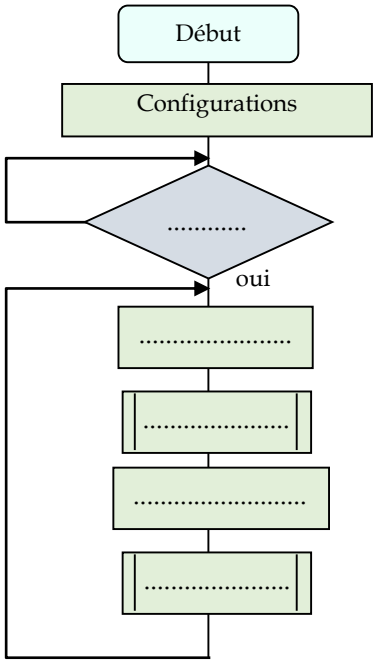
```
        ……..        ; Eteindre les LED
loop    ……..        ; Temporiser
        ……..        ; complémenter PORTB
        ……..        ; répéter
```

Exercice

Faire clignoter la LED du montage suivant ; le bouton permet de lancer les clignotements
Réaliser la temporisation par une boucle de retard (temporisation logicielle) et la confier à un sous-programme

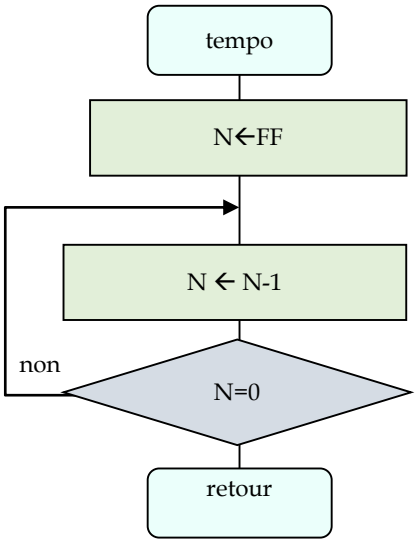


Programme




```
        N  equu 0x20          ; Initialisation des variables

                              ; page 1
                              ;
                              ; configurer port B
                              ;
                              ; page 0
                              ;
test    .....                 ; tester si RB5 = 1 et saut si vrai
                              ;  revenir à 'test'
lab     .....                 ; allumer
                              ; appel du sous-programme 'tempo'
                              ; éteindre
                              ; temporisation
                              ; revenir à 'lab'

tempo   Movlw 0xFF            ; sous prog. de temporisation
        movwf  N
boucle  Decfsz  N, f
        Goto boucle
        Return
```

Sous-programme de temporisation




```
tempo   .....          ; N ← FF
        .....
loop    .....          ; décrémente N et test si N=0
        .....          ; sinon retour à l'étiquette 'loop'
        return         ; retour au programme principal
```

Durée de temporisation du sous-programme

La fréquence f de fonctionnement interne du PIC est le quart de celle du quartz soit f = Fosc /4

- Pour un quartz de Fosc = 20 MHz ⇒ f = 5 MHz ⇒ 1 cycle = 1/f = 0.2 µs
- Pour un quartz de Fosc = 4 MHz ⇒ f = 1 MHz ⇒ 1 cycle = 1 µs

- Pour un quartz de 4MHz, 1 cycle instruction = 1µs
- Toutes les instructions du sous prog. prennent 1 cycle sauf goto et return qui en prennent 2

La temporisation est approximativement :

……

……

Multiples durées de temporisation

Augmenter la durée de temporisation peut être obtenu par imbrication des boucles de retard

```
tempo  .....
       .....
loop   .....
       .....
       return
```

Durée T = …… avec un quartz de ……

```
tempo1 .....
       .....
loop1  .....
       .....
       .....
       return
```

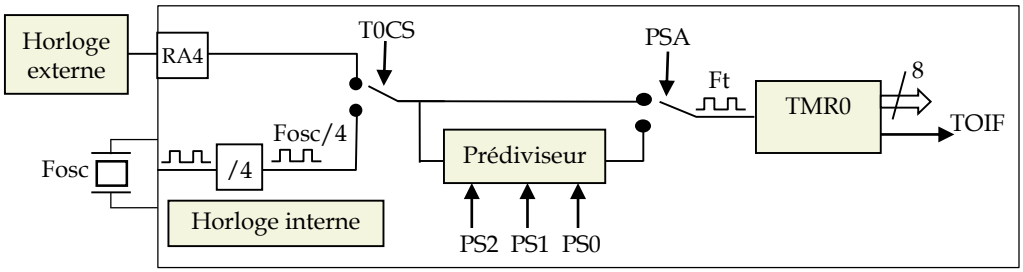
Durée T1 = ……

```
tempo2 .....
       .....
loop2  .....
       .....
       .....
       return
```

Durée T2 = ……

⇨ *Le module Timer 0*

- Le Timer 0 est un compteur 8 bits qui peut servir de temporisateur (timer) ou de compteur d'évènements
- TMR0 est un registre associé au Timer 0



Configuration

Timer 0 est configuré par le registre OPTION_REG

Registre **OPTION_REG** (pages 1 et 3)

RBPU	INTEDG	*T0CS*	T0SE	*PSA*	*PS2*	*PS1*	*PS0*

- L'horloge peut être interne ou externe
  - T0CS = 0 → horloge interne (de fréquence Fosc/4)
  - T0CS = 1 → horloge externe appliquée à la broche RA4

- On peut ou non utiliser le prédiviseur.
  - PSA = 0 → le prédiviseur est affecté au timer TMR0
  - PSA = 1 → le prédiviseur est affecté au Watchdog WDT

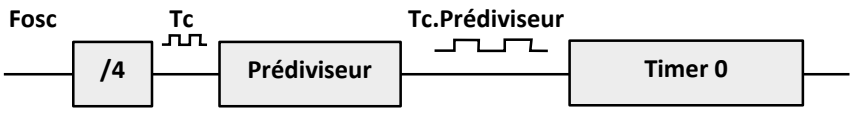
Le facteur de division de la fréquence est fixé par les bits PS2, PS1 et PS0

PS2 PS1 PS0	**000**	**001**	**010**	**011**	**100**	**101**	**110**	**111**
Division de fréquence de TMR0 par	**2**	**4**	**8**	**16**	**32**	**64**	**128**	**256**

Timer 0 en mode timer

Dans ce mode, Timer 0 est piloté par l'horloge interne du microcontrôleur et sert de temporisateur
La durée de temporisation est **T = Tc.Prédiviseur.(256 – TMR0)**

Tc : durée d'un cycle horloge
Prédiviseur : valeur de prédivision
TMR0 : valeur de départ du registre TMR0



Temporisation par scrutation du débordement du Timer 0

Au débordement du timer (passage de FF à 00), le bit TOIF du registre INTCON passe à 1

Registre **INTCON** (pages 0, 1, 2 et 3)

GIE	PEIE	TOIE	INTE	RBIE	*TOIF*	INTF	RBIF

Exercice

On désire obtenir une temporisation de 10 ms, trouver le mot de configuration(OPTION_REG) et la valeur initiale de TMR0. On donne Fosc = 4 MHz
Ecrire, ensuite, un sous-programme qui met en œuvre cette temporisation en scrutant le bit TOIF

En mode timer, on utilise l'horloge interne : T0CS = ….
Et si on choisit une prédivision = ……., le mot de configuration sera :

OPTION_REG	RBPU	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0
	.....	.....	.....	.....	.....	.....	.....	.....

Avec un quartz de Fosc = 4MHz, la période du signal de l'horloge interne est …….
……
……
……

Sous-programme de temporisation de 10 ms par scrutation du bit TOIF

```
        ..
          ……..……..        ; configurer  Timer 0 via OPTION_REG
          ……..…...
        ..
tempo10ms ..........        ; valeur de départ du registre TMR0
          ..........
          ..........        ; baisser le drapeau càd mettre TOIF = 0
attente   ..........        ; tester si TOIF = 1
          ..........        ; sinon revenir à 'attente'
          ..........        ; retour au programme principal
```

Exercice

Donner la valeur initiale de TMR0 et trouver le mot de configuration(OPTION_REG) pour obtenir la temporisation maximale possible. On donne Fosc = 4 MHz
Ecrire, ensuite, un sous-programme qui met en œuvre cette temporisation
……
……

OPTION_REG	RBPU	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0
	.....	.....	.....	.....	.....	.....	.....	.....

Sous-programme de temporisation maximale

```
        ..
          ……..……..        ; configurer  Timer 0 via OPTION_REG
          ……..…...
        ..
tempomax  ..........        ; valeur de départ du registre TMR0
          ..........        ; baisser le drapeau càd mettre TOIF = 0
attente   ..........        ; tester si TOIF = 1
          ..........        ; sinon revenir à 'attente'
          ..........        ; retour au programme principal
```

⇨ *Le module WDT Watchdog Timer ou chien de garde*

C'est un compteur 8 bits incrémenté par une horloge indépendante de l'horloge système (même si le µC est en mode sleep). Lorsqu'il déborde (après 18 ms), deux situations sont possibles :

- Si le µC est en fonctionnement normal, cela provoque un RESET.
- Si le µC est en mode SLEEP, cela provoque un WAKE-UP (l'exécution du programme reprend)

Configuration

Registre **OPTION_REG** (pages 1 et 3)

RBPU	INTEDG	T0CS	T0SE	*PSA*	*PS2*	*PS1*	*PS0*

Il est possible d'augmenter la durée par utilisation du prédiviseur (celui partagé avec le timer 0). Le coefficient de prédivision est fixé par les bits PS0, PS1 et PS2

- PSA = 0 → le prédiviseur est affecté au timer TMR0
- PSA = 1 → le prédiviseur est affecté au Watchdog WDT

Le coefficient de prédivision de la fréquence est fixé par les bits PS2, PS1 et PS0

PS2 PS1 PS0	**000**	**001**	**010**	**011**	**100**	**101**	**110**	**111**
Division de fréquence par	**1**	**2**	**4**	**8**	**16**	**32**	**64**	**128**

Le WDT est activé par la directive *__CONFIG _WDT_ON*

Le mode SLEEP

Dans ce mode, l'horloge système est arrêtée ce qui suspend l'exécution du programme.
Pour sortir du mode SLEEP et donc continuer l'exécution, il faut provoquer un WAKE-UP par :

- Un RESET externe dû à l'initialisation du PIC; dans ce cas, l'exécution recommence
- Ou par un débordement du chien de garde WDT s'il est activé; dans ce cas, l'exécution continue

Temporisation par le WDT combiné au mode Sleep

On met le µC dans le mode Sleep ; le programme est alors suspendu et reprendra après débordement du WDT.

Exercice

Faire clignoter une LED montée sur la ligne RC0

On optera pour une prédivision par …….. donc PSA = …….. et PS2 PS1 PS0 = ……..….

Donc

OPTION_REG	RBPU	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0
	.....	.....	.....	.....	.....	.....	.....	.....

La durée du WDT étant de ……
Avec le prédiviseur elle devient ……

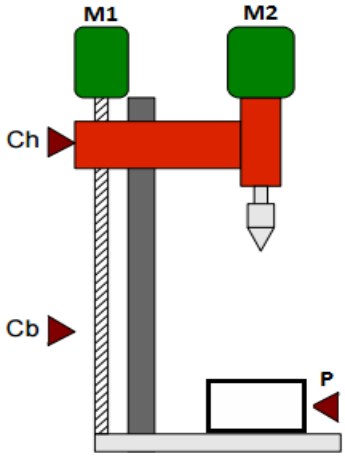
```
        ……                          ; activer WDT
        ..
        ……                          ; configurer  WDT via OPTION_REG → tempo=…….
        ……
        ..

loop    .....                       ; allumer la LED
        .....                       ; passer en mode sleep. Réveil dans ……  par  débordement de WDT
        .....                       ; éteindre la LED
        .....                       ; passer en mode sleep. Réveil dans ……  par  débordement de WDT
        goto loop
```

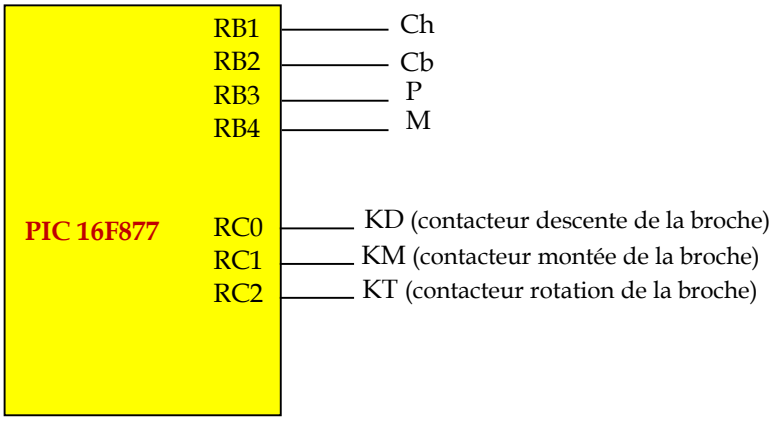
Durées possibles du WDT

PSA	PS2, PS1, PS0	Taux de prédivision du WDT	Mot de configuration (OPTION_REG)	Durée indicative
1	000	……..	……	……
1	001	……..	……	……
1	010	……..	……	……
1	011	……..	……	……
1	100	……..	……	……
1	101	……..	……	……
1	110	……..	……	……
1	111	……..	……	……

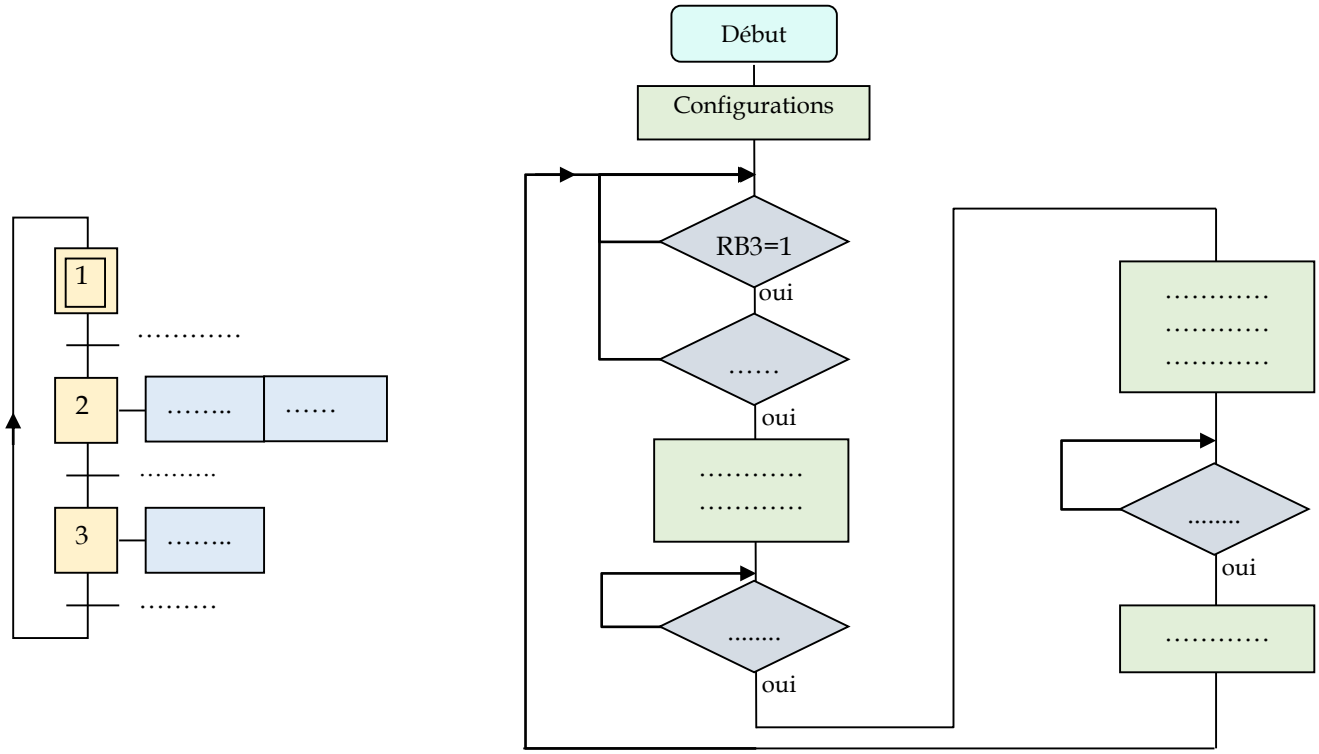
Implantation du GRAFCET dans un microcontrôleur

Le système est une perceuse automatique simple. Lorsque la pièce à usiner est en place, un appui sur le bouton M lance le cycle : la machine descend jusqu'à Cb puis remonte pour s'arrêter en Ch





Fournir le GRAFCET, le traduire en organigramme puis en un programme assembleur pour le microcontrôleur PIC16F877.




```
        bcf   STATUS, RP1          ; page 1
        bsf   STATUS, RP0
        ……..                       ; configurer  PortB
        ……..
        ……..                       ; configurer  PortC
        bcf   STATUS, RP0          ; page 0
        bcf   STATUS, RP1

test1   ……..                       ; si RB3 = 1
        ……..
        ……..                       ; si RB4 = 1
        ……..
        ……..                       ; RC0 ← 1
        ……..                       ; RC2 ← 1
test2   ……..                       ; si RB2 = 1
        ……..
        ……..                       ; RC0 ← 0
        ……..                       ; RC2 ← 0
        ……..                       ; RC1 ← 1
test3   ……..                       ; si RB1 = 1
        ……..
        ……..                       ; RC1 ← 0
        ……..                       ; recommencer
```

⇨ *Les interruptions*

Une interruption provoque l'arrêt du programme principal pour aller exécuter un sous-programme, dit d'interruption, qui se trouve à l'adresse **0004**.
A la fin de cette procédure, le microcontrôleur reprend le programme principal à l'endroit où il l'a laissé.

Sources d'interruption

Interruption	*Source d'interruption*	*Bit de validation*	*Flag*	*Interruption périphérique*
TOI	Débordement de Timer0	TOIE	TOIF	Non
INT	Front sur la broche RB0/INT (le front de déclenchement est configuré par le bit INTEDG)	INTE	INTF	Non
RBI	Front sur une broche RB4-RB7	RBIE	RBIF	Non
ADI	Fin de conversion A/N	ADIE	ADIF	Oui
TMR1I	Débordement de Timer1	TMR1E	TMR1F	Oui
TMR2I	Timer2 a atteint la valeur maxi	TMR2E	TMR2F	Oui
EEI	Fin d'écriture en EEPROM	EEIE	EEIF	Oui

- Toutes les interruptions peuvent être validées/interdites à la fois par le bit GIE
- Chaque interruption peut être validée/interdite par son bit de validation individuel
- Un bit drapeau (flag) permet de savoir si l'évènement déclenchant l'interruption arrive

L'interruption INT

Cette interruption externe arrive via la broche RB0/INT, sa mise en œuvre nécessite de :

- Mettre à 1 le bit **GIE** pour l'autorisation globale de toutes les interruptions
- Mettre à 1 le bit **INTE** afin d'autoriser l'interruption sur RB0/INT.
- Choisir le front (montant ou descendant) qui déclenche l'interruption par le bit **INTEDG** du registre OPTION_REG

Le bit **INTF** (flag) est un indicateur, il est mis à 1 si une interruption arrive sur RB0/INT. Il doit être remis à 0 dans le programme de traitement de l'interruption.

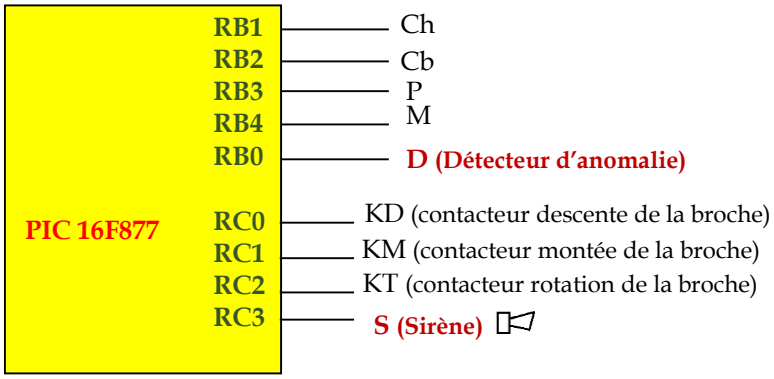
Registre **INTCON**

GIE	EEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF

Exercice : mise en œuvre de l'interruption INT

Il s'agit d'alerter les responsables en cas de danger autour de la perceuse de l'exercice traité plus haut : en cas d'anomalie signalée par le détecteur D sur la broche RB0, la sirène S montée sur RC3 entre en action

On va charger le programme principal de l'opération de perçage et la routine d'interruption du traitement de l'anomalie



Configuration de l'interruption INT (sur RB0)

Registre	bit	Action
INTCON	GIE = 1	Autorisation générale de toutes les interruptions
INTCON	INTE = 1	Autorisation de l'interruption INT (sur RB0)
OPTION_REG	INTEDG = 1	Choix du front montant sur RB0

```
        ORG  0000
        goto depart

        ORG  0004
        .....                ; déclencher la sirène
        .....                ; remettre à 0 le flag  INTF
        .....                ; retour au prog.principal

depart  ..
        .....                ; configurer l'interruption
        .....
        .....
        ..
```

Sous-prg d'interruption

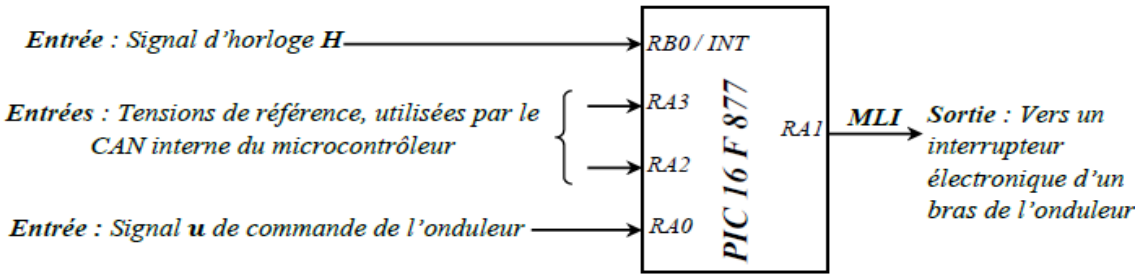
Prog.principal

Exercice : Examen national 2018, session normale

Tâche 2 : Commande de l'onduleur

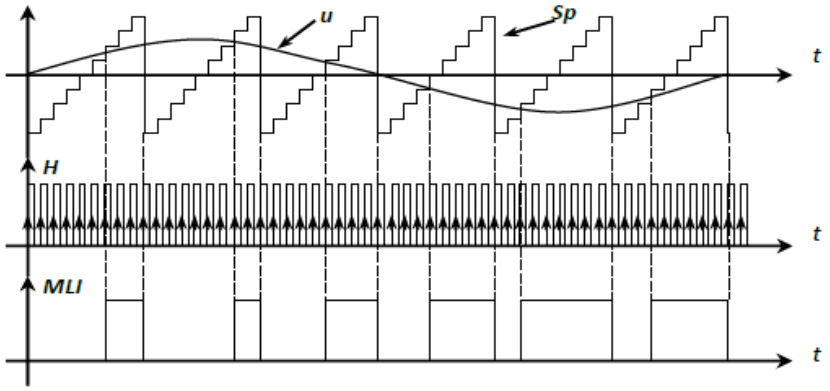
Dans cette partie, on étudie une technique de génération d'un signal **MLI** (Modulation de Largeur d'Impulsion) par le microcontrôleur **PIC 16F877**. Le signal **MLI** est un signal **TOR** dont le rapport cyclique est une image de la tension ***u*** de commande, appliquée à l'entrée **RA0** du **PORTA** du microcontrôleur.

Le schéma de principe de cette commande est le suivant :



Principe :

Sp est un signal en dents de scie interne généré par le microcontrôleur tel que :



H est le signal d'horloge de synchronisation, appliqué à l'entrée d'interruption **RB_0/INT** du microcontrôleur. **Cp** est un compteur **8 bits** (registre interne), incrémenté à chaque front montant du signal **H**. La routine d'interruption consiste alors, à incrémenter le contenu de la case mémoire **Cp** qui simule le signal **Sp** en dents de scie. **N** est l'image numérique de la tension ***u*** de commande ; ***u*** est convertie par le module **C.A.N** (Convertisseur Analogique Numérique) intégré au **PIC16F877**.

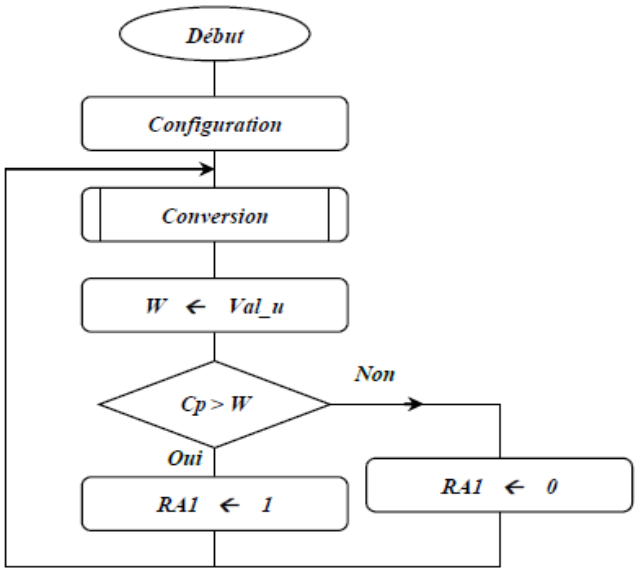
En comparant **Cp** à **N** (Nombre fourni par le **CAN** intégré au **PIC 16F877**), on pourra générer le signal **MLI** commandant un interrupteur électronique d'un bras de l'onduleur.

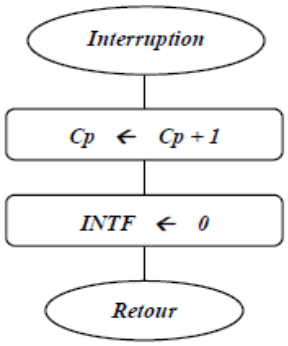
L'algorithme de cette commande est le suivant :

- Si **Cp** > **N**, alors **MLI** = 1 ;
- Sinon **MLI** = 0.

Le sous-programme **"Conversion"** est un sous-programme qui permet de convertir le signal de commande **u** en un nombre **N** sur **8 bits**, stocké dans la case mémoire appelée **Val_u**.

On donne ci-dessus l'organigramme de la commande d'un bras de l'onduleur.





Question : 45. *En vous aidant du jeu d'instructions fourni en* ***DRES 03****, compléter le programme assembleur correspondant. (Les bits non utilisés de* ***TRISA*** *et* ***TRISB*** *sont à 0)* **[8 pts]**

Question :45.

LABEL	*CODE ASSEMBLEUR*		*COMMENTAIRE*
	BCF	*STATUS, 6*	
	.....		*; Accès à la banque 1*
	.....		*; Configuration TRISA*
	MOVWF	*TRISA*	
	.....		*; Configurationon TRISB*
	MOVWF	*TRISB*	
	Configuration des registres :		
	OPTION – INTCON – ADCON1 – ADCON0		
	BCF	*STATUS, 5*	*; Accès à la banque 0*
LAB	*CALL*	*CONVERSION*	*; Appel du sous-programme* ***CONVERSION***
	.....		*; Lecture du résultat de la conversion*
	.....		*; W = Cp - W*
	.....		*; Tester si le résultat est négatif*
	.....		*; Sinon MLI = 1*
	BTFSS	*STATUS, C*	*; Tester si le résultat est positif*
	BCF	*PORTA, 1*	*; Sinon MLI = 0*
	GOTO	*LAB*	*; Reprendre*

Interruption	*INCF*	*Cp, F*	*Incrémenter compteur Cp*
	BCF	*INTCON, INTF*	*Remise à zéro du flag d'interruption INT*
	.....		*Retour d'interruption*

Exercice : Examen national 2018, session de rattrapage

Tâche 4 : Test du niveau de la charge de la batterie *(voir document* ***DRES 04****)*

*A chaque mise sous tension du chariot, l'utilisateur est informé du niveau de la charge de la batterie par une signalisation sonore. Pour cela, un sous-programme « **Test** » permet de tester la tension de la batterie* E_{Bat} *et d'émettre des bips sonores :*

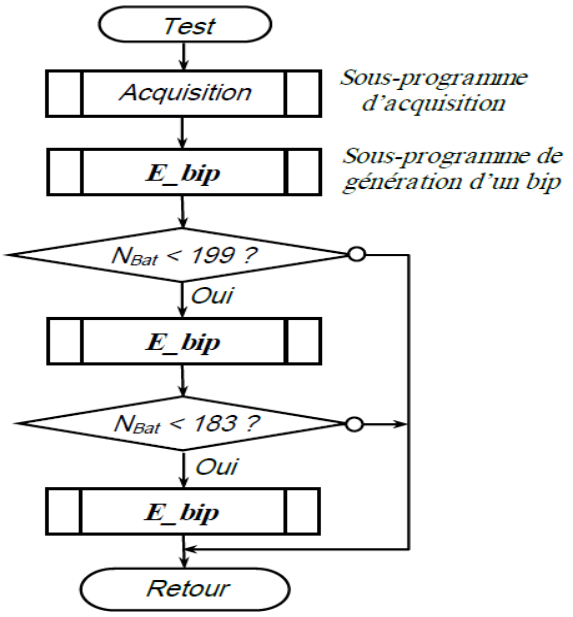
- $E_{Bat} \geq 12\ V$ ➔ *charge convenable :* ***1 bip*** *;*
- $11\ V \leq E_{Bat} < 12\ V$ ➔ *charge moyenne :* ***2 bips*** *;*
- $E_{Bat} < 11\ V$ ➔ *charge insuffisante (batterie déchargée) :* ***3 bips****.*

Le principe de ce test consiste à acquérir la tension U_{Bat} *(image de la tension de la batterie* E_{Bat}*) et à la convertir en une valeur numérique* N_{Bat} *; ensuite, on compare cette valeur aux seuils correspondants définis dans le tableau suivant :*

	$E_{Bat} = 11\ V$	$E_{Bat} = 12\ V$
U_{Bat} *(V)*	***3,59***	***3,92***
N_{Bat} *en décimal*	***183***	***199***

*L'organigramme du sous-programme « **Test** » est représenté ci-contre :*

On note ***adr_Nbat*** *l'adresse de la case mémoire contenant la valeur numérique* N_{Bat}*.*



Q39) *En utilisant le jeu d'instructions du microcontrôleur donné sur le document* ***DRES 05****, compléter le programme Assembleur correspondant à cet organigramme.* ***6 pts***

Q39) *Programme assembleur :*

Label	***Mnémonique***	***Opérande***	***Commentaire***
Test	. . . . .	. . . . .	; Appel au sous-programme Acquisition
	CALL	E_bip	; Appel au sous-programme E_bip
	. . . . .	. . . . .	; Charger W par la valeur 199
	SUBWF	Adr_NBat,W	; Comparer (adr_NBat) à W
	. . . . .	. . . . .	; Sauter si NBat < 199
	GOTO	Fin	; Aller à la fin
	CALL	E_bip	; Appel au sous-programme E_bip
	. . . . .	. . . . .	; Charger W par la valeur 183
	. . . . .	. . . . .	; Comparer (adr_NBat) à W
	BTFSC	STATUS,C	; Sauter si NBat < W
	. . . . .	. . . . .	; Aller à la fin
	CALL	E_bip	; Appel au sous-programme E_bip
Fin	RETURN		

Exercice : Examen national 2019, session de rattrapage

Tâche 2 : Programme de régulation du degré d'acidité pH

Le programme de régulation agit selon l'**algorithme** suivant :

- Acquisition du niveau de pH ;
- Affichage du niveau de pH sur l'indicateur (BARGRAPH) ;
- Si le pH mesuré est inférieur à 6,9 (N < N1) alors :
  - ✓ Injection pendant 5 minutes du correcteur de pH+ par la mini-pompe doseuse pH+ ;
  - ✓ Arrêt de la mini-pompe pendant 15 minutes, ce qui permet de stabiliser l'eau traitée ;
- Si le pH mesuré est supérieur à 7,7 (N > N2) alors :
  - ✓ Injection pendant 5 minutes du correcteur de pH- par la mini-pompe doseuse pH- ;
  - ✓ Arrêt de la mini-pompe pendant 15 minutes, ce qui permet de stabiliser l'eau traitée.

On dispose des sous programmes suivants :

- **Acquisition_pH** : qui permet de convertir le signal U_3 image du pH en un nombre N sur 8 bits, stocké dans la case mémoire appelée **Val_pH** ;
- **Affichage_pH** : qui permet d'afficher le pH mesuré sur le BARGRAPH à diodes LED connecté sur le port B ;
- **TEMPO_T1** : est un sous-programme de temporisation T1 = 5 min ;
- **TEMPO_T2** : est un sous-programme de temporisation T2 = 15 min.

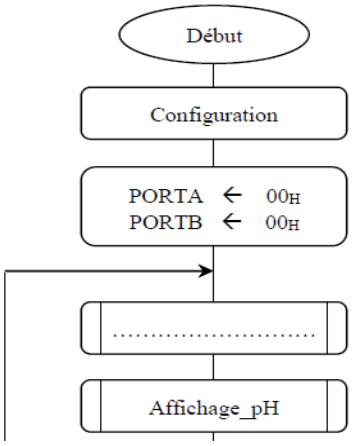
Q.43- *En vous aidant de l'algorithme ci-dessus, compléter l'**organigramme** correspondant.* **[4,5 pts**

La configuration du PIC16F876 est comme suit :

- RA2 : Sortie logique pour la commande de la mini-pompe doseuse pH- ;
- RA1 : Sortie logique pour la commande de la mini-pompe doseuse pH+ ;
- RA0 : Entrée analogique pour l'acquisition de la tension U_3 image du pH de l'eau de la piscine ;
- RB0...RB7 : Sorties logiques pour la commande du BARGRAPH.

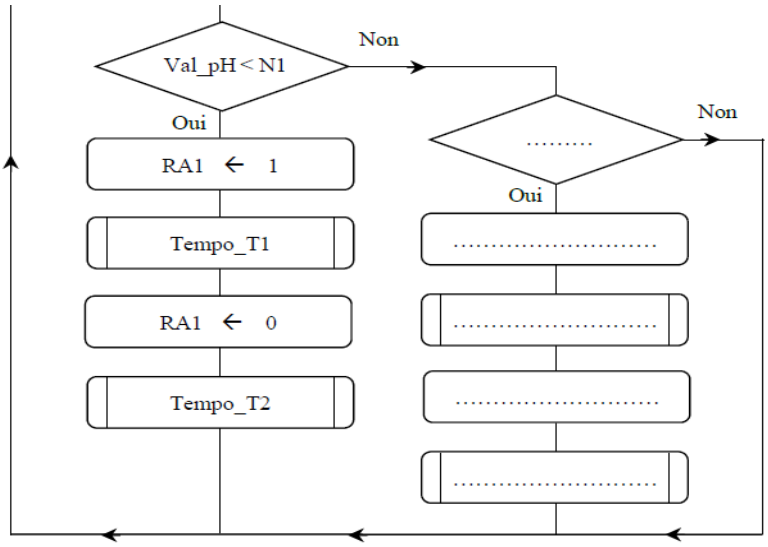
Q.44- *En vous aidant du jeu d'instructions fourni en **DRES 04** page **14**, compléter le programme assembleur de régulation du pH. (Les bits non utilisés du registre **TRISA** sont à 0).* **[5,5 pts]**

Question.43: **[4,5 pts]**



N1 : est la valeur de N correspondante à un pH de 6,9.

N2 : est la valeur de N correspondante à un pH de 7,7.



Question.44: **[5,5 pts]**

```
                BCF     STATUS, 6
                ..........................      ; Accès à la banque 1
                ..........................
                ..........................      ; Configuration PORTA
                ..........................      ; Configuration PORTB
                MOVLW   0X0E
                MOVWF   ADCON1          ; Configuration ADCON1
                BCF     STATUS, 5       ; Accès à la banque 0
                MOVLW   0X81
                MOVWF   ADCON0          ; Configuration ADCON0
                CLRF    PORTB           ;
                CLRF    PORTA           ; Initialisation des sorties du système
LAB1            CALL    Acquisition_pH  ; Appel du sous-programme Acquisition_pH
                CALL    Affichage_pH    ; Appel du sous-programme Affichage_pH
                ..........................      ;
                ..........................      ;
                ..........................      ;
                GOTO    LAB2            ; Si non sauter à LAB2
                ..........................      ;
                ..........................      ;
                ..........................      ;
                ..........................      ;
                GOTO    LAB1            ; Reprendre
LAB2            MOVF    Val_pH, W       ; Lecture du résultat de la conversion
                SUBLW   N2              ; W = N2 - W
                BTFSC   STATUS, C       ; W > N2?
                GOTO    LAB1            ; Si non sauter à LAB1
                BSF     PORTA, 2        ; Injection du pH-
                CALL    TEMPO_T1        ; Appel du sous-programme de temporisation 5 min
                BCF     PORTA, 2        ; Arrêt d'injection du pH-
                CALL    TEMPO_T2        ; Appel du sous- programme de temporisation 15 min
                GOTO    LAB1            ; Reprendre
```

Exercice : Examen national 2021, session normale

Tâche 2 : Commande du système de refroidissement à l'eau de mer

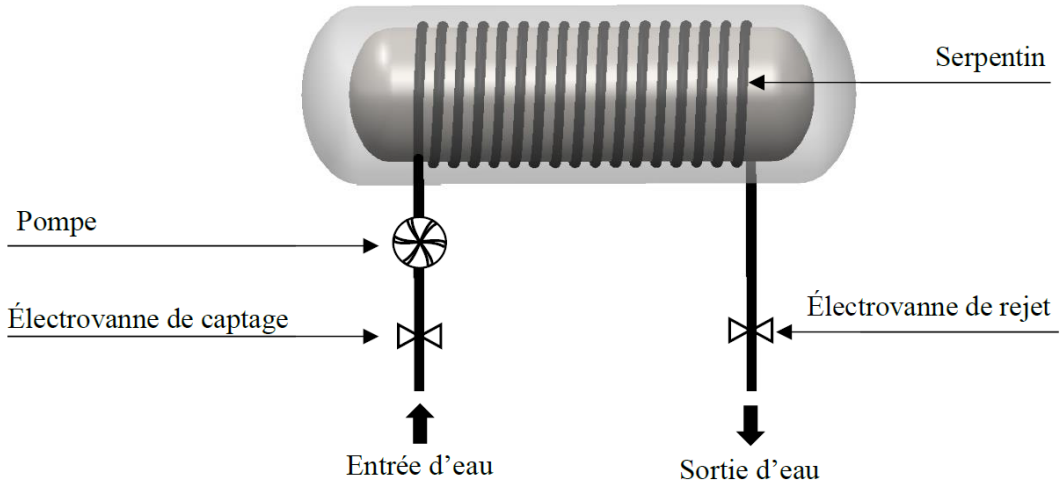
Les constituants électriques de l'hydrolienne, en particulier les convertisseurs électroniques de puissance, sont confinés à l'intérieur du convertisseur **AC/AC** (l'enceinte posée au fond marin et abritant l'ensemble du matériel électrique de l'hydrolienne).

De ce fait, les composants électroniques peuvent être portés à des températures excessives. Cela nécessite de prévoir un moyen de refroidissement afin de protéger l'électronique embarquée.

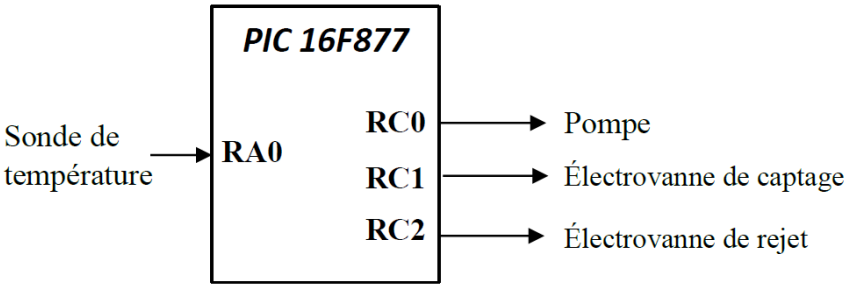
On envisage de maintenir la température à l'intérieur du convertisseur **AC/AC** entre **20 °C** et **30 °C** par un système de refroidissement à l'eau de mer.

Moyennant deux électrovannes et une pompe à eau de mer, le système fait circuler l'eau dans un serpentin (tube métallique enroulé en spirale) inséré dans la double coque du convertisseur **AC/AC**. C'est un refroidissement en circuit ouvert ; en effet, après circulation dans le serpentin, l'eau est rejetée dans la mer.

Une sonde mesure la température à l'intérieur du convertisseur **AC/AC.**



La commande de ce système de refroidissement est réalisée par un microcontrôleur **PIC 16F877** selon l'architecture suivante :



Programme de commande de ce système de refroidissement

Éléments du programme

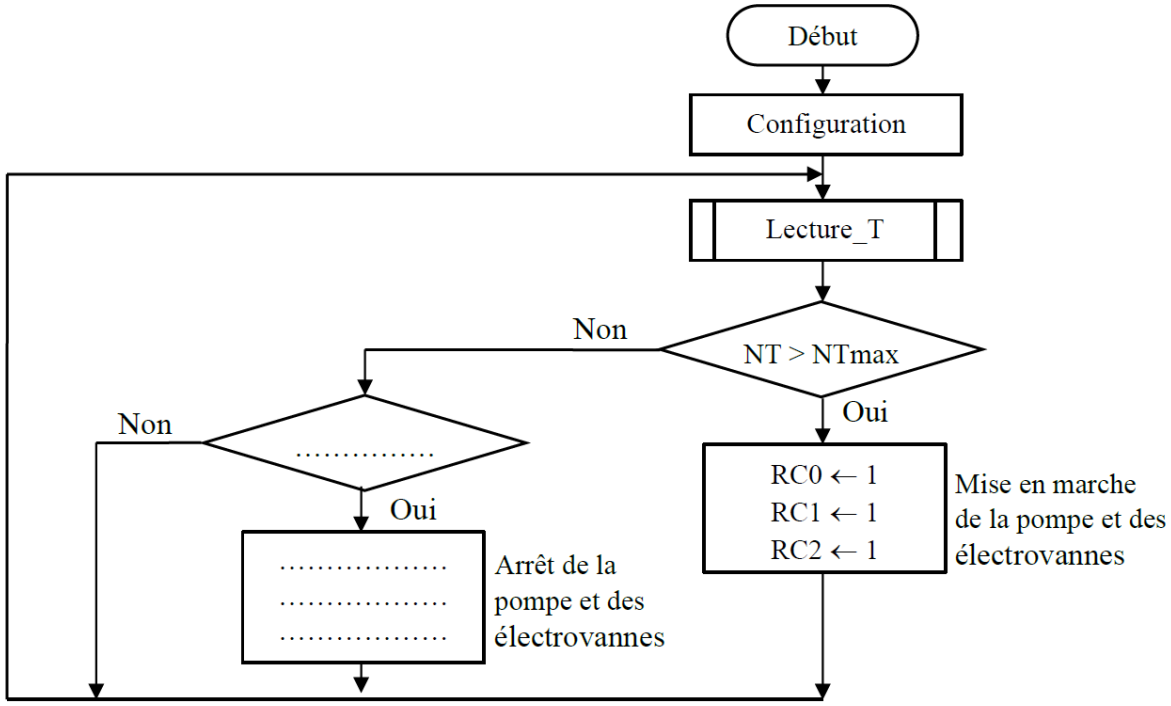
- **Lecture_T** : sous-programme chargé de lire, sur la ligne **RA0**, l'information issue de la sonde de température et de ranger le code numérique correspondant dans le registre **NT**.
- **NT** : registre qui contient le code numérique de la température mesurée par la sonde.
- **NT_{min}** et **NT_{max}** : sont les codes numériques qui correspondent respectivement à **20 °C** et **30 °C**.

Séquences du programme

- Appel du sous-programme **Lecture_T** ;
- Evaluation de la température lue :
  - Si **NT > NT_{max}**, on met en marche la pompe et les deux électrovannes ;
  - Si **NT < NT_{min}**, on arrête la pompe et les deux électrovannes ;
- Reprendre une nouvelle lecture de la température.

Question : 43. *Compléter l'organigramme traduisant les séquences ci-dessus.* **2 pts**

Question : 44. *En vous aidant du jeu d'instructions du **DRES 04**, compléter le programme assembleur correspondant.* **6 pts**



Instruction		Commentaire	
	BCF STATUS, RP1 ……	; Activer la page 1	
	……	; Configurer PORTC en sortie	
	BSF TRISA, 0	; Configurer la ligne RA0 en entrée	
	BCF STATUS, RP1 **BCF STATUS, RP0**	; Activer la page 0	
repeter	……	; appel du sous-programme "Lecture_T"	
	MOVLW NTmax **SUBWF NT, w** ……	; W ← NTmax ; W ← NT – NTmax ; tester si le bit C = 1	tester si NT > NTmax
	GOTO saut	; si non, se brancher à l'étiquette "saut"	
	BSF PORTC, 0 **BSF PORTC, 1** **BSF PORTC, 2**	; si oui, mettre en marche la pompe et les deux électrovannes	
	GOTO repeter	; reprendre	
saut	…… …… ……	; W ← NTmin ; W ← NT – NTmin ; tester si le bit C = 0	tester si NT < NTmin
	GOTO repeter	; si non, reprendre	
	BCF PORTC, 0 **BCF PORTC, 1** ……	; si oui, arrêter la pompe et les deux électrovannes	
	GOTO repeter	; reprendre	

Exercice : Examen national 2022, session normale

Tâche 2 : Affichage de la masse à déplacer

Le monte-charge étudié est équipé d'une carte d'acquisition et d'affichage de la masse $\mathbf{m_C}$ à déplacer (voir document ressources **DRES 04**). Cette carte est à base du microcontrôleur **PIC 16 F 876**.

L'affichage se fait en décimal sur 3 afficheurs à 7 segments, équipés chacun d'un décodeur BCD/7 segments et d'un Latch 4 bits (verrou à 4 bascules D).

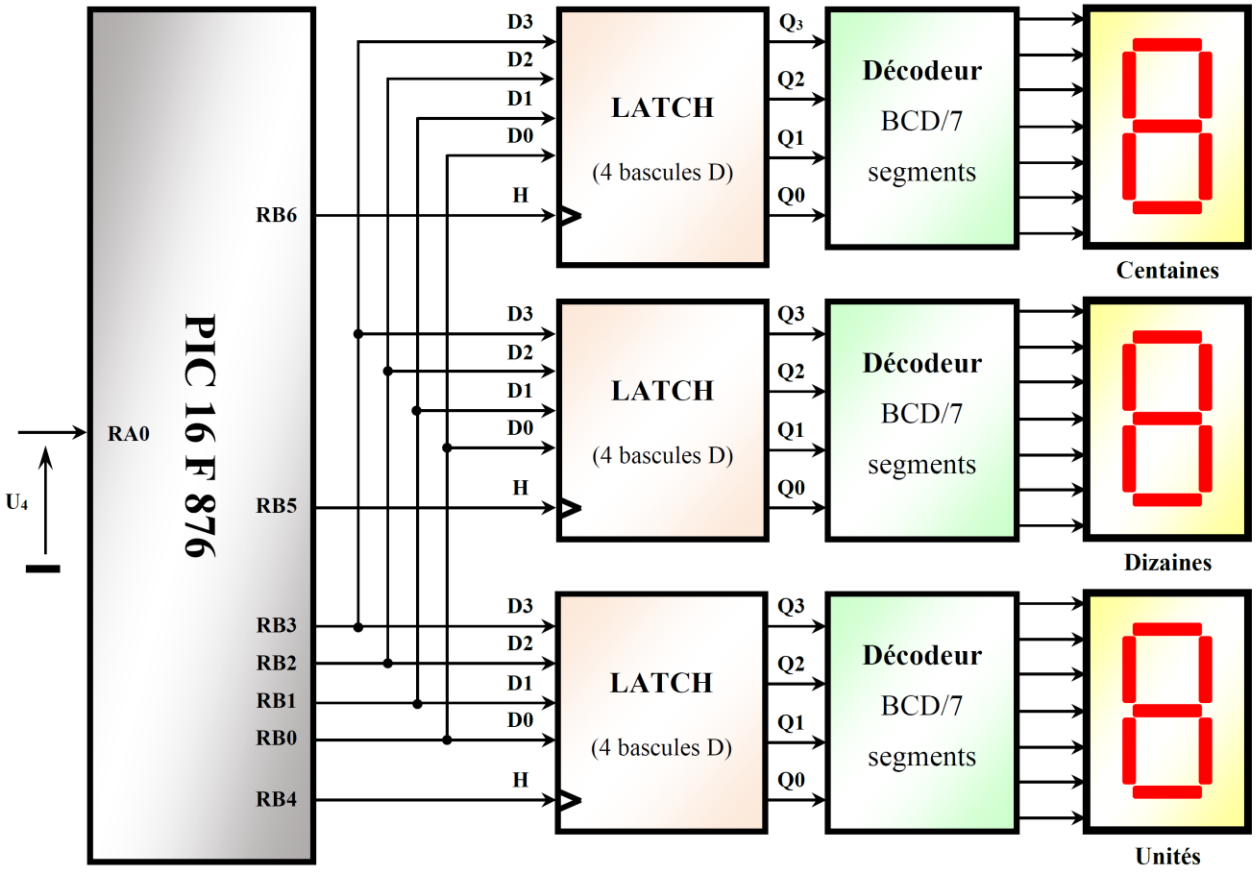
On dispose des sous programmes suivants :

- **Acquisition** : Sous-programme qui convertit le signal $\mathbf{U_4}$ image de la masse $\mathbf{m_C}$ à déplacer en un nombre **N** sur **10 bits**, stocké dans les registres internes du CAN (**ADRESH : ADRESL**).
- **Conv_Kg** : Sous-programme qui convertit la valeur numérique **N** résultat du convertisseur CAN (ADRESH : ADRESL) en un nombre **N'** exprimé en **Kg**, stocké dans deux cases mémoires appelées **Val_mc_H** et **Val_mc_L**.
- **Conv_BCD :** Sous-programme qui convertit le nombre **N'** en décimal (code BCD), stocké dans les cases mémoires appelées **Unite_mc** (pour les unités), **Dizaine_mc** (pour les dizaines) et **Centaine_mc** (pour les centaines) ;

Q.47 - *En vous aidant du document ressources **DRES 04**, compléter l'**organigramme** de l'acquisition et de l'affichage de la masse $\mathbf{m_C}$ à déplacer.* **4 pts**

Q.48 - *En vous aidant du jeu d'instructions fourni en document ressources **DRES 05**, compléter le **programme** assembleur correspondant.* **5 pts**

Carte d'acquisition et d'affichage de la masse à déplacer **DRES 04**



Principe d'affichage de la masse à déplacer

Si **N'= 2DE**$_H$ alors **N'= 734** en décimal donc **N'= 0111.0011.0100** en BCD

Donc, après appel du sous-programme **"Conv_BCD"** :

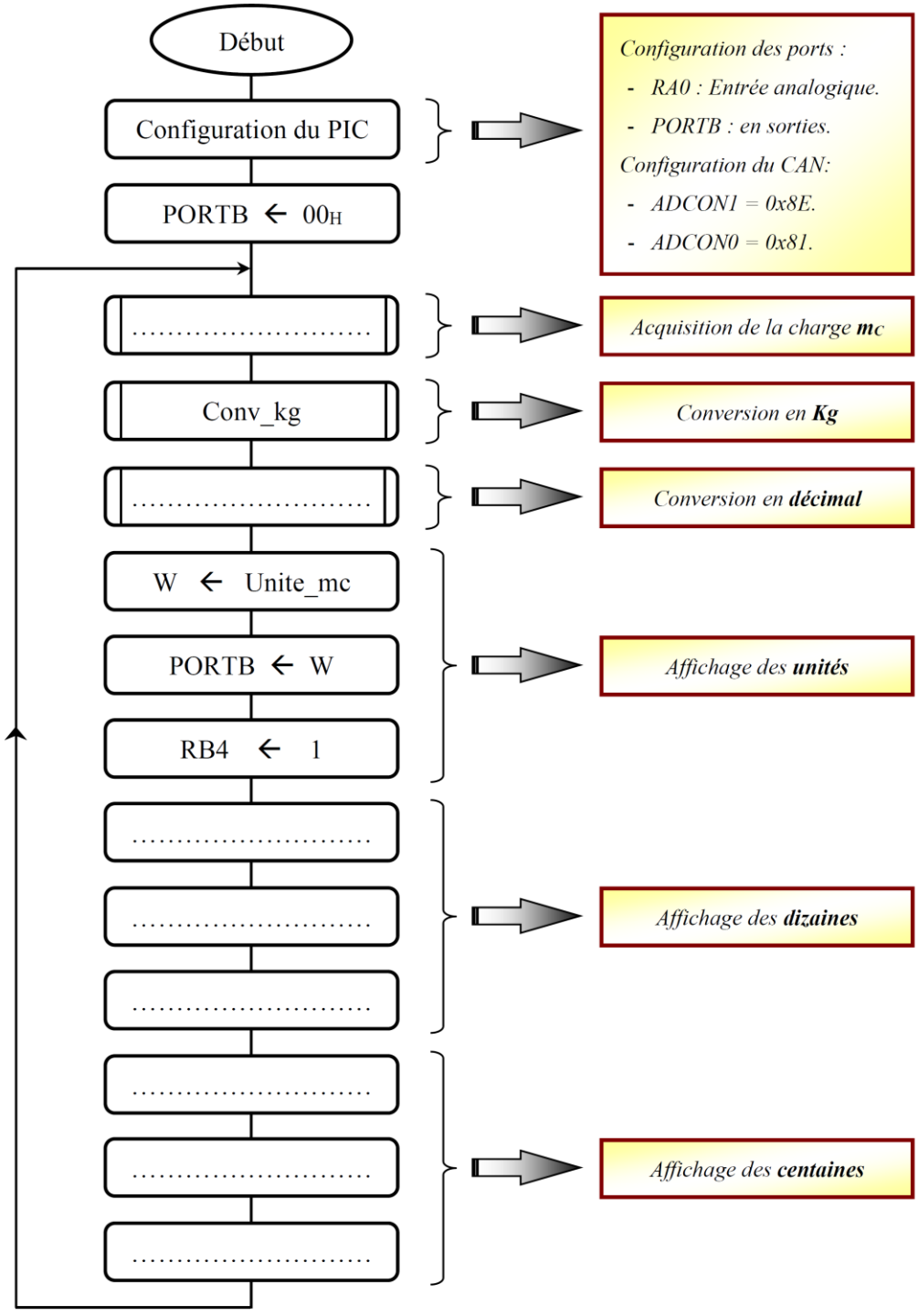
- Unité_mc = **0 0 0 0 . 0 1 0 0** (4)
- Dizaine_mc = **0 0 0 0 . 0 0 1 1** (3)
- Centaines_mc = **0 0 0 0 . 0 1 1 1** (7)

Pour afficher la masse à déplacer $\mathbf{m_C}$ en décimal sur les **trois** afficheurs (N'= **2DE** $_H$= **734** en décimal), on effectue les opérations suivantes :

- Acquisition de la masse à déplacer par appel du sous-programme **"Acquisition"** ;
- Convertir le nombre **N** résultat de la conversion analogique numérique en un nombre **N'**, exprimé en **kg** par appel du sous-programme **"Conv_Kg"** ;
- Convertir **N'** en BCD par appel du sous-programme **"Conv_BCD"** ;
- Écrire la valeur des unités '**4**' sur les entrées $D_3..D_0$ de l'afficheur des unités avec **H** = 0 ;
- Mettre **H** à 1 pour mémoriser la valeur des unités par les sorties $Q_3..Q_0$ (créer un front montant sur l'entrée d'horloge des unités) ;
- Écrire la valeur des dizaines '**3**' sur les entrées $D_3..D_0$ de l'afficheur des dizaines avec **H** = 0 ;
- Mettre **H** à 1 pour mémoriser la valeur des dizaines par les sorties $Q_3..Q_0$ (créer un front montant sur l'entrée d'horloge des dizaines) ;
- Écrire la valeur des centaines '**7**' sur les entrées $D_3..D_0$ de l'afficheur des centaines avec **H** = 0 ;
  Mettre **H** à 1 pour mémoriser la valeur des centaines par les sorties $Q_3..Q_0$ (créer un front montant sur l'entrée d'horloge des centaines).

Q.47 -

DREP 09


Début
Configuration du PIC
PORTB ← 00H
.....
Conv_kg
.....
W ← Unite_mc
PORTB ← W
RB4 ← 1
.....
.....
.....
.....
.....
.....
Configuration des ports :
- RA0 : Entrée analogique.
- PORTB : en sorties.
Configuration du CAN:
- ADCON1 = 0x8E.
- ADCON0 = 0x81.
Acquisition de la charge mc
Conversion en Kg
Conversion en décimal
Affichage des unités
Affichage des dizaines
Affichage des centaines

Q.48 -

DREP 10

```
                BCF     STATUS, 6       ;
                ....................    ; accès à la BANK 1
                ....................    ; PORTB en sortie
                ....................    ; Mot de commande du registre TRISA
                MOVWF   TRISA           ; RA0 en entrée
                MOVLW   0x8E            ; Mot de commande du registre ADCON1
                MOVWF   ADCON1          ; Configuration du CAN interne
                BCF     STATUS, 5       ; Retour en banque mémoire 0
                MOVLW   0x81            ; Mot de commande du registre ADCON0
                MOVWF   ADCON0          ; Configuration du CAN interne
                CLRF    PORTB           ; Initialisation des sorties
Loop            CALL    Acquisition     ; appel du sous-programme "Acquisition"
                ....................
                CALL    Conv_BCD        ; appel du sous-programme "Conv_BCD"
                ....................
                ....................
                ....................
                ....................
                ....................
                ....................
                MOVF    Centaine_mc, W  ; Lecture de la valeur des centaines
                MOVWF   PORTB           ; Ecriture des centaines dans le PORTB
                BSF     PORTB, 6        ; Affichage des centaines
                GOTO    Loop            ;
```