People's Democratic Republic of Algeria
Ministry of Higher Education and Scientific Research
University of Science and Technology of Oran - Mohamed BOUDIAF
Faculty of Electrical Engineering
Vice-Dean, Postgraduate, Scientific Research and External Relations
وزارة التعليم العالي والبحث العلمي
جامعة وهران للعلوم والتكنولوجيا محمد بوضياف
كلية الهندسة الكهربائية
نيابة عمادة الكلية لما بعد التدرج والبحث العلمي والعلاقات الخارجية

Département d'Electronique

Polycopié Pédagogique

TITRE

Les Systèmes à Microcontrôleurs

Cours et Travaux-Dirigés destinés aux étudiants de :

Master (spécialité et niveau) : Instrumentation Biomédicale
(Master1 – 2ème Semestre)

Auteur : Mr LAKHDARI Fethi

Experts : Mr ZAFRANE Amine et Mme Mazouz Nacera

Année Universitaire : 2024/2025

Table des matières

1 Introduction générale **1**

2 Le microcontrôleur PIC18F458 **5**
2.1 Introduction 5
2.2 Caractéristiques Techniques du PIC18F458 5
2.2.1 Le CPU 8
2.2.2 Les registres internes 8
2.2.3 Organisation de la mémoire 9
2.2.4 Les circuits d'horloge et de reset 11

3 L'assembleur des PIC18F **15**
3.1 Introduction 15
3.2 La forme d'une instruction assembleur 15
3.3 Les modes d'adressage 19
3.3.1 Le mode littérale 19
3.3.2 L'adressage de la mémoire 19
3.4 Les instructions de base 22
3.4.1 Les instructions affectant un octet 23
3.4.2 Les instructions affectant un bit d'un octet 23
3.4.3 Les boucles et les branchements (instructions de contrôle) 23
3.5 La pile et les sous-programmes 25
3.5.1 Les instructions CALL, RCALL 26
3.5.2 Les instructions RETURN, RETLW 26
3.5.3 Le mode 'FAST' 27
3.6 Exercices 27

4 Les ports d'entrée/sortie numérique **29**
4.1 Introduction 29
4.2 Le PORTA 31
4.3 Le PORTB 31
4.4 Le PORTC 31
4.5 Le PORTD 32

4.6 Le PORTE 32
4.7 Le mode PSP : 32

4.8 Exercices 34

5 Les modules Timers et CCP **37**
5.1 Introduction 37
5.2 Les Timers 37
5.2.1 Le Timer 0 37
5.2.2 Le Timer 1 44
5.2.3 Le Timer 2 44
5.2.4 Le Timer 3 45
5.3 Le module CCP (Capture/Compare/Pwm) 46
5.3.1 Le module 'Compare' 47
5.3.2 Le module 'PWM' 48
5.3.3 Le module ECCP (enhanced CCP) 49

5.4 Exercices 52

6 Les interruptions **55**
6.1 Introduction 55
6.2 Les interruption du PIC18F458 55
6.2.1 Gestion de la priorité 56
6.2.2 Réponse à une interruption 57
6.2.3 Les interruptions externes (INT) 57
6.2.4 Les interruptions des Timers 57
6.3 Exercices 59

7 La communication série **61**
7.1 Introduction 61
7.2 Le module USART 62
7.2.1 Le mode asynchrone 63
7.2.2 Le mode synchrone 67
7.2.3 Le module : Master Synchronous Serial Port 68

7.2.4 Le mode : Serial Peripheral Interface 69
7.2.5 Le mode I2C 70
7.3 Exercices 72

8 La conversion analogique/numerique **75**
8.1 Introduction 75
8.2 Le principe de base de la conversion 75
8.3 L' ADC du PIC18F458 76
8.3.1 Les étapes d'une conversion 78
8.4 Exercices 79

9 Annexe **81**

9.1 Annexe 1 :Rappel géneral 81
9.2 La représentation numérique 81
9.2.1 La représentation des entiers 81
9.2.2 La représentation des réels 83
9.2.3 Le code ASCII 84
9.3 Annexe 2 :Les registres SFR (Special Function Register) 85
9.4 Annexe 3 :Le jeu d'instructions des PIC18F 86

Chapitre 1

Introduction générale

Ce polycopié de cours est destiné aux étudiants du département d'électronique et à l'ensemble des étudiants de la faculté du génie électrique. Il s'agit d'une introduction plus ou moins détaillée au système à microcontrôleurs où un microcontrôleur de la série des PIC18F de Microchip est pris comme exemple d'étude.

La série des microcontrôleurs PIC18F de Microchip est largement utilisée dans le domaine de l'électronique embarquée vu qu'elle offre un bon équilibre entre performance, flexibilité, et coût. Cette série de microcontrôleurs est pratiquement une optimisation des caractéristiques et des performances de l'ancienne série des PIC16F.

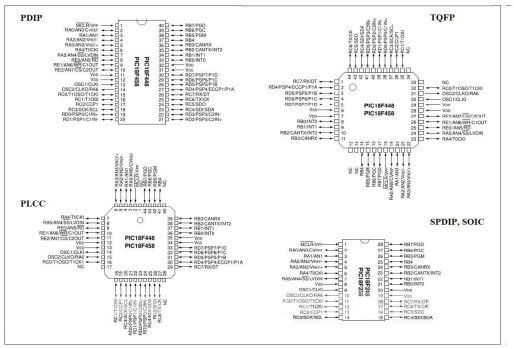


Figure 1 : les differents boitiers des microcontroleurs PIC18F

–

Du point vue fabrication, les PIC18F sont réalisés à l'aide du processus CMOS (Complementary Metal-Oxide-Semiconductor), ce qui permet en particulier une faible consommation d'énergie avec une haute densité de circuit, chose

qui facilite l'intégration d'un nombre élevé de périphériques et de fonctionnalités dans un circuit donné.

Les principales performances et caractéristiques des PIC18F peuvent être résumés ainsi :

1. Une architecture 8 bits :

Ce qui les rend adaptés aux applications où le traitement ne nécessite pas une puissance élevée de calcul.

2. Une mémoire :

- Mémoire Flash : pour le stockage du programme, typiquement entre 4 Ko et 128 Ko.

- Mémoire RAM : pour les données temporaires, avec des tailles variant généralement entre 256 octets et 4 Ko.

- Mémoire EEPROM : Certains modèles possèdent de la mémoire EEPROM avec des tailles allant de 128 octets et 1 Ko.

3. Le boitier et nombre de broches :

Les PIC18F sont disponibles dans plusieurs configurations de boîtiers (PDIP, PLCC , TQFP,. . .) typiquement allant de 8 à 64 broches (Figure 1).

4. L'horloge :

La fréquence d'horloge atteint jusqu'à 64 MHz dans certains cas, ce qui permet des vitesses de traitement élevées.

5. Les périphériques Intégrés :

- Convertisseurs Analogique-Numérique (ADC) : avec des résolutions allant jusqu'à 10 bits.

- Timers : Plusieurs timers (de 8 à 16 bits)

- Module PWM (Pulse Width Modulation) : Pour la génération d'un signal de contrôle modulé en largeur de d'impulsion.

- Module de communication série : avec la disponibilité de diffèrents protocoles tel que USART, SPI, I2C, can, et même USB dans certains microcontrôleurs de cette série.

6. Le Jeu d'Instructions :

Pour la série des PIC18F un jeu d'instructions amélioré (comparé au PIC16F) est implémenté avec une exécution en un seul cycle (instrucrtion) pour la plupart des instructions.

7. La gestion de l'énergie :

Les modes veille sont disponibles dans cette série des PIC18F permettent une gestion efficace de l'énergie (une réduction de la consommation électrique).

8. Les entrées/Sorties :

Les ports d'entrée/sortie (I/O) numériques sont généralement bidirectionnels et permettent une interaction flexible avec les périphériques externes.

Ce document est constitué d'un ensemble de chapitres, dont le premier est dédié a la description de l'architecture du microcontrôleur PIC18F458 qui sera l'exemple traité par la suite. Le second chapitre est consacré à la programmation assembleur, ou le jeu d'instruction des PIC18F est présenté d'une manière plus ou moins détaillée.

Les chapitres suivants sont consacrés à la présentation et l'étude des différents modules et modes de fonctionnement tels que les interruptions, les ports E/S, les timers, les ports série, et la conversion analogique/numérique.

Pour une meilleur assimilation un ensemble d'exemples et d'exercices sont proposés dans chaque chapitre, dont certains même nécessite le recours aux datasheets des constructeurs, pour familiariser l'étudiant aux bonnes pratiques dans le domaine de la micro-informatique.

Chapitre 2

Le microcontrôleur PIC18F458

2.1 Introduction

Le tableau 1 fournie par le constructeur [1] résume les caractéristiques des PIC18FXX8, qui sont conçus pour offrir des performances élevées et une flexibilité dans les applications embarquées. Ces circuits sont particulièrement appréciés pour leurs capacités à gérer différents protocoles de communications tels que le I2C, le SPI, ou le bus CAN ce qui les rend idéal pour des applications industrielles.

Si le microcontrôleur 8 bits PIC18F458 est souvent l'exemple traité dans les chapitres suivant, Les explications et les exemples traités restent valables et facilement extensibles au reste des microcontrôleurs de la série des 18FXX8, est même à d'autres circuits tels que les 18f45xx, ou le 18F45K22.

2.2 Caractéristiques Techniques du PIC18F458

Cette section donne plus de détails sur les caractéristiques du PIC18F458 résumées dans le tableau 1, et illustrées dans le diagramme bloc de la figure 1.

Features		PIC18F248	PIC18F258	PIC18F448	PIC18F458
Operating Frequency		DC – 40 MHz	DC – 40 MHz	DC – 40 MHz	DC – 40 MHz
Internal Program Memory	Bytes	16K	32K	16K	32K
	# of Single-Word Instructions	8192	16384	8192	16384
Data Memory (Bytes)		768	1536	768	1536
Data EEPROM Memory (Bytes)		256	256	256	256
Interrupt Sources		17	17	21	21
I/O Ports		Ports A, B, C	Ports A, B, C	Ports A, B, C, D, E	Ports A, B, C, D, E
Timers		4	4	4	4
Capture/Compare/PWM Modules		1	1	1	1
Enhanced Capture/Compare/ PWM Modules		—	—	1	1
Serial Communications		MSSP, CAN, Addressable USART	MSSP, CAN, Addressable USART	MSSP, CAN, Addressable USART	MSSP, CAN, Addressable USART
Parallel Communications (PSP)		No	No	Yes	Yes
10-bit Analog-to-Digital Converter		5 input channels	5 input channels	8 input channels	8 input channels
Analog Comparators		No	No	2	2
Analog Comparators VREF Output		N/A	N/A	Yes	Yes
Resets (and Delays)		POR, BOR, RESET Instruction, Stack Full, Stack Underflow (PWRT, OST)	POR, BOR, RESET Instruction, Stack Full, Stack Underflow (PWRT, OST)	POR, BOR, RESET Instruction, Stack Full, Stack Underflow (PWRT, OST)	POR, BOR, RESET Instruction, Stack Full, Stack Underflow (PWRT, OST)
Programmable Low-Voltage Detect		Yes	Yes	Yes	Yes
Programmable Brown-out Reset		Yes	Yes	Yes	Yes
CAN Module		Yes	Yes	Yes	Yes
In-Circuit Serial Programming™ (ICSP™)		Yes	Yes	Yes	Yes
Instruction Set		75 Instructions	75 Instructions	75 Instructions	75 Instructions
Packages		28-pin SPDIP 28-pin SOIC	28-pin SPDIP 28-pin SOIC	40-pin PDIP 44-pin PLCC 44-pin TQFP	40-pin PDIP 44-pin PLCC 44-pin TQFP

Tableau1 : les principales caracteristiques des PIC18FXX8 [1]

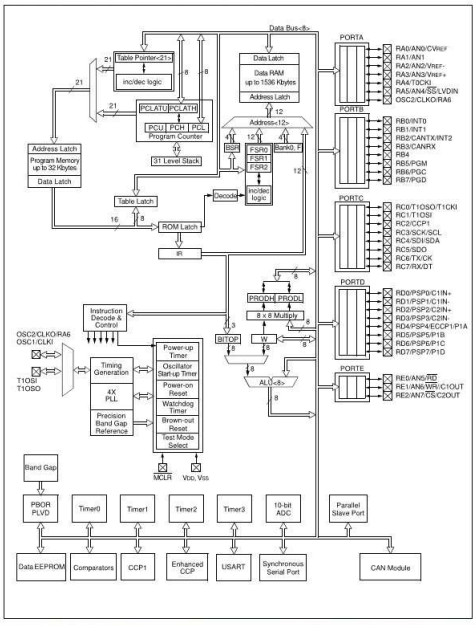


Figure 1 : Le diagramme bloc des microcontroleurs PIC18F 448/458 [1]

1. Mémoire

• **Mémoire Flash :**

- Capacité de 32 Ko.
- Permet jusqu'à 100 000 cycles d'écriture/effacement.
- Protection contre l'écriture en mémoire programme.

• **RAM :**

1536 octets, utilisée pour le stockage temporaire des données.

• **EEPROM :**

- Capacité de 256 octets.
- Permet jusqu'à 1 000 000 cycles d'écriture/effacement.

2. Performance

• Fréquence d'Horloge : Jusqu'à 40 MHz, offrant une exécution rapide des instructions.

• Vitesse d'Exécution : Jusqu'à 10 MIPS (Million d'Instructions Par Seconde).

3. Broches d'E/S

Nombre Total de Broches d'E/S : 36 broches numériques, configurables individuellement comme entrées ou sorties.

4. Interfaces de Communication

• **EUSART :**

Enhanced Universal Synchronous Asynchronous Receiver Transmitter pour la communication série.

• **MSSP :**

Master Synchronous Serial Port qui prend en charge les protocoles **I2C** et **SPI** pour la communication série synchrone.

• **CAN :**

- Implémentation du protocole **CAN 2.0B**.
- Supporte jusqu'à 1 Mbps, idéal pour les applications industrielles nécessitant des communications robustes.

5. Convertisseurs A/N

- Nombre de Canaux : 10 canaux de conversion analogique-numérique (A/N) avec une résolution de 10 bits.
- Plage de Tension : Convertit des signaux analogiques allant de 0 à la tension d'alimentation.

6. Timers et module CCP

• **Quatre (04) Timers** sont implémentés :

- Timer 0 : 8/16 bits
- Timer 1 : 16 bits
- Timer 2 : 8 bits
- Timer 3 : 16 bits

- **Deux modules de capture/compare/PWM (Pulse Width Modulation) :**

7. Protection et Sécurité

- **Brown-out Reset (BOR) :**

Protège le microcontrôleur contre les fluctuations de tension en redémarrant le système en cas de basse tension.

- **Watchdog Timer (WDT) :**

Un Timer qui réinitialise le microcontrôleur si le programme ne le réinitialise pas à temps, prévenant les blocages.

- **Code Protection** :

Mécanismes pour protéger le code en mémoire contre la lecture et l'écriture non autorisées.

8. Tension et Consommation

- Plage de Tension de Fonctionnement : 2.0V à 5.5V, permettant une utilisation dans des systèmes à faible consommation d'énergie.
- Mode Sleep : Réduit la consommation d'énergie en éteignant les parties non essentielles du microcontrôleur lorsqu'il est inactif.

9. Packages Disponibles

- **DIP** (Dual In-line Package) à 40 broches : Facile à manipuler pour les projets de prototypage.
- **PLCC** (Plastic Leaded Chip Carrier) à 44 broches : Offre une densité de broches plus élevée pour des conceptions compactes.
- **TQFP** (Thin Quad Flat Package) à 44 broches : Adapté pour les applications à faible encombrement sur PCB.

2.2.1 Le CPU

L'unité centrale de traitement (Central Processing Unit : CPU) recoit l'instruction stockée dans la mémoire programme, la décode puis l'exécute.

Le CPU comprend principalement (figure 1) :

- L'unité de traitement arithmétique et logique : (UAL ou ALU).
- Le multiplicateur (hardware) 8bit x 8bits.
- Le registre de travail (worcking register : WREG)

2.2.2 Les registres internes

Tel qu'il est le cas dans l'architecture d'un circuit processeur, un registre 'compteur programme' (PC), un 'registre d'état' (STATUS), et un 'pointeur de pile' sont toujours implémentés.

Le PC (pointe l'instruction à exécuter, et dans le cas du 18F458 c'est un registre de 21 bits (figure 2.a).

Le registre d'état' STATUS indique via un ensemble de bits (flags) l'état du processeur après chaque opération (figure 2.b).

Remarque 1 *Les registres relatifs à la gestion de la pile seront présentés dans un chapitre ultérieur.*

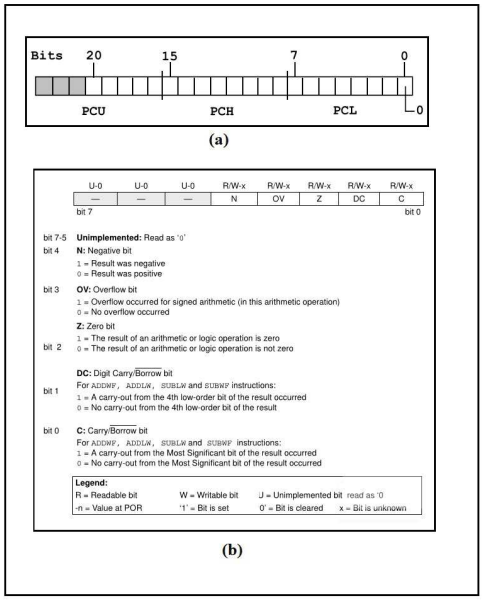


Figure 2 : Le registre PC (a), et le registre STATUS (b)

2.2.3 Organisation de la mémoire

Tel qu'il a été décrit dans les paragraphes introductifs, Le PIC18F458 possède trois différents types de mémoire :

Une mémoire flash pour les programme, une mémoire RAM, et une EEPROM.

La mémoire programme

Le compteur programme du PIC18F458 est de 21 bits, par conséquent la taille mémoire qu'il peut adresser est de 21M octets.

Effectivement la taille mémoire flash adressable est de 32k octets (figure 3). Cela signifie que le processeur peut 16K d'instructions.

Apres un reset le PC pointe l'adresse (000h), et la table du vecteur d'interruption occupe la zone (008h-018h) (Figure 3).

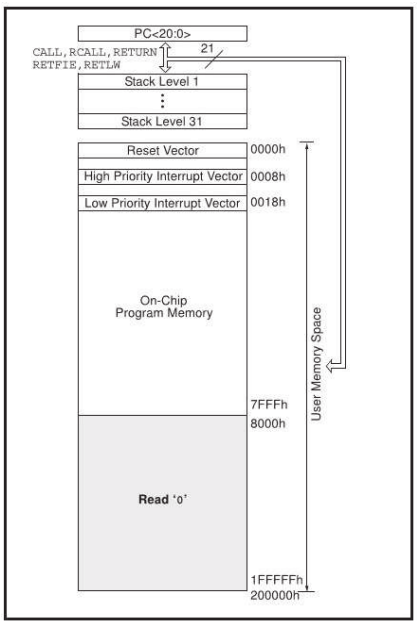


Figure 3 : Organisation de la memoire programme (a), et de la pile (b).

La mémoire données

La mémoire donnée est une mémoire de type RAM, dont l'adresse de chaque registre est codée sur 12 bits, permettant ainsi l'dressage de 4k octets de mémoire.

Cette mémoire est organisée via une répartition en 16 zones (Banks) de 256 octets (figure 4). Le registre BSR (BSR<3 :0>) permet la sélection d'une BANK parmi les seize.

La mémoire donnée comprend deux types de registres :

- GPRs (General Purpose Registers), utilisés pour le stockage des données des applications
- SFRs (Special Function Registers) : qui qui sont l'ensemble des registres de configuration du microcontrôleur et ses périphériques (Ports, Timers, ADC, interruption,. . .) .

L'accès à la mémoire peut etre effectué de deux façons différentes :

- Le mode direct : dans ce cas l'adresse complète du registre est donnée en spécifiant la valeur du BSR (le numero de la BANK).
- Le mode indirect : dans ce cas les 12 bits de l'adresse du registre sont spécifiés dans le FSR (File Selection Register).

Pour simplifier l'accès simultané aux GPRs ainsi que les registres SFRs sans recours au registre BSR, un mode dit 'Access Bank' est implémenté (figure 4). Une partie de la bank 0, et une partie de la bank 15 (les SFR) sont regroupés pour permettre un tel accès.

Remarque 2 *Des exemples d'accès à mémoire de données seront traités par la suite dans le chapitre consacré au software des PIC18F.*

Remarque 3 *La liste des SFRs (special function registers) est donnée dans l'annexe 2.*

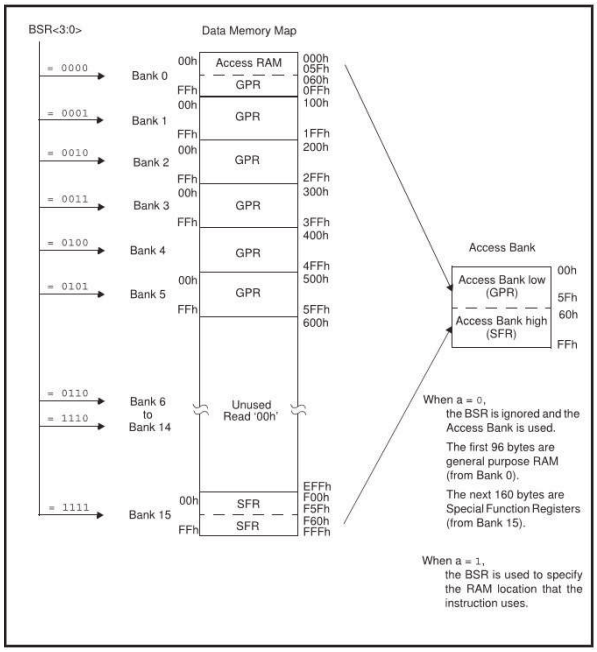


Figure 4 : Organisation de la memoire RAM

2.2.4 Les circuits d'horloge et de reset

Dans une conception de base (Figure 5), en plus d'assurer une alimentation électrique, un circuit d'horloge, et un circuit de reset sont nécessaires pour le fonctionnement d'un microcontrôleur.

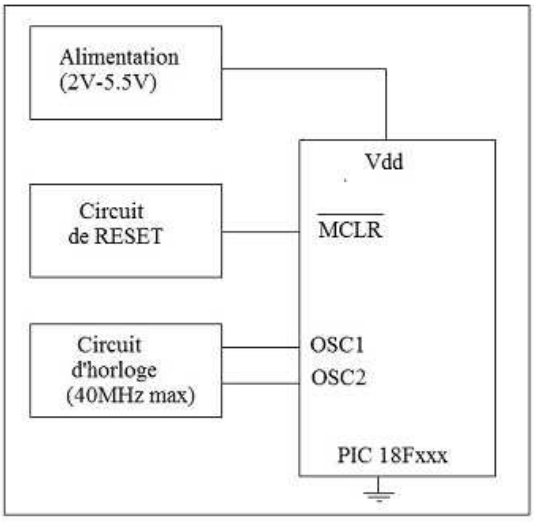


Figure 5 : Schema bloc de principe de circuit a base d'un microcontroleur

Le circuit d'horloge

Microchip simplifie la conception du circuit d'horloge en permettant plusieurs solutions.

En effet pour générer le signal d'horloge 8 modes différents de fonctionnement sont possibles (Tableau 2). Trois bits (Fosc0,1,2)d'un registre interne (accessible durant la phase de flashage du circuit) fixe le mode adopté.

LP :	Low-Power
XP :	Crystal Crystal/Resonator
HS :	High-Speed Crystal/Resonator
HS4 :	High-Speed Crystal/Resonator with PLL enabled
RC :	External Resistor/Capacitor
RCIO :	External Resistor/Capacitor with I/O pin enabled
EC :	External Clock
ECIO :	External Clock with I/O pin enabled

Tableau 2 : Les differents modes d horloge

Pour chaque mode une plage de fréquences d'horloge est possible. Les valeurs des composants passifs à ajouter sont spécifiées dans le datasheet du constructeur.

Dans les quatre premiers modes (LP, XP, HS, HS4) un circuit oscillateur est branché entre les pins OSC1et OSC2 (figure 6.a).

Le signal d'horloge peut être aussi générer via un circuit passif (modes : RC, RCIO) ou via une horloge externe (modes EC, ECIO). Dans le cas de ces modes, la pin OSC2 peut être configurée comme sortie de fréquence fixe (Fosc/4) ou comme une pin d'entrée/sortie générale (figure 4.b, c, d).

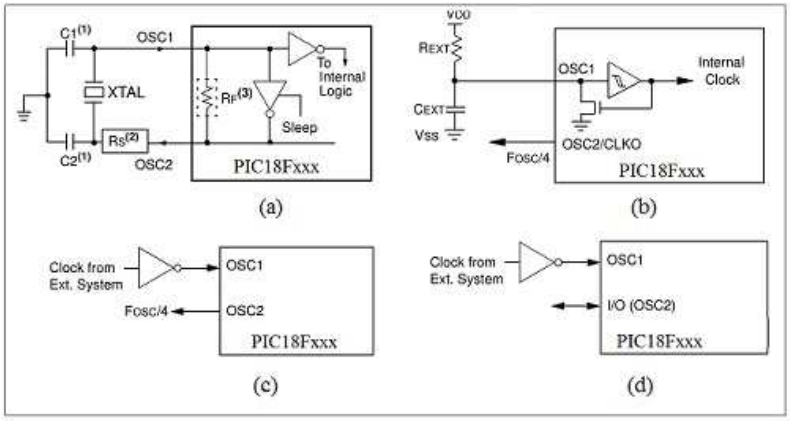


Figure 6 : Les configurations du circuit d'horloge

Le circuit de RESET

Une réinitialisation (RESET) du microcontrôleur (telle que le cas de sa mise sous tension) provoque automatiquement le branchement vers l'adresse mémoire programme 0x000.Outre la mise sous tension le PIC18F458 possède en tout huit (08) sources possibles de REST (Figure 7).

La nature du signal de RESET peut être soit logicielle (via l'instruction RESET), ou matériel tel qu'un niveau bas sur la ligne MCLR , ou le débordement du compteur d'un Timer.

Le RESET-MCLR

Apres la mise sous tension la pin MCLR du microcontrôleur doit être maintenu à un niveau haut (figure 8.a). Cette pin est utilisée pour forcer la réinitialisation du system via sa mise à zéro en utilisant par exemple un bouton poussoir (figure 8.b).Le constructeur propose dans son datasheet l'ajout d'une diode pour accélérer la décharge du condensateur dans le cas où la vitesse d'extinction de la source d'alimentation est relativement lente (figure 8.c).

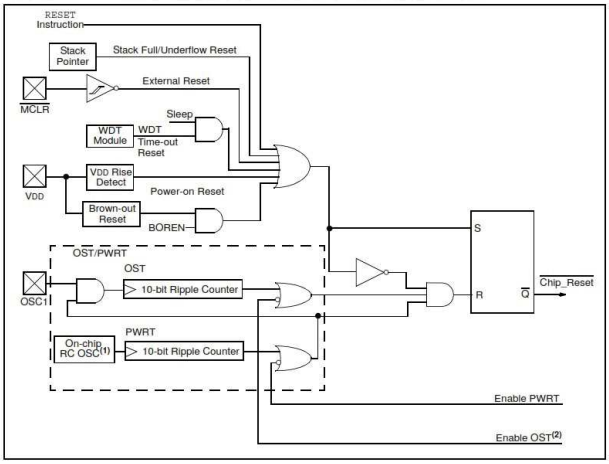


Figure 7 : shema bloc simplifie des sources de RESET

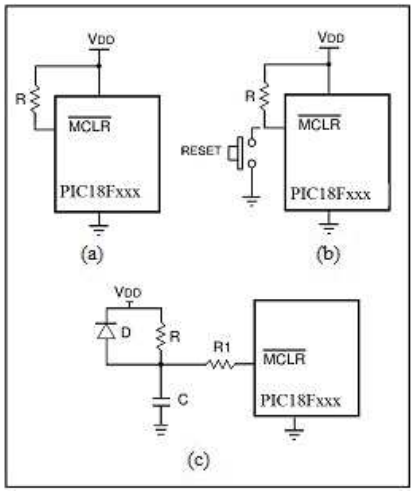


Figure 8 : Exemples de configuration de la pin RESET-MCLR

Chapitre 3

L'assembleur des PIC18F

3.1 Introduction

Tel qu'il a été introduit au chapitre premier, un microcontrôleur (ou un microprocesseur) exécute séquentiellement les instructions (codes machine) stockées dans la mémoire programme et pointées par le registre PC (program counter).

Au cours d'un cycle instruction, un mécanisme de lecture, et de décodage du code machine précède la phase finale d'exécution.

En microinformatique, l'assembleur est définit comme étant le langage de programmation de niveau bas. Cela signifie qu'il est proche du code machine, mais sous une forme lisible et intelligible [3].

Le présent chapitre est une introduction à l'assembleur des PIC18F. Microchip propose un jeu de 75 instructions, codées dans le cas général sur 16 bits.

Un cycle instruction dure 4 périodes d'horloge, à l'exception de quelques instructions telles que les branchements qui ont une durée double.

Remarque 4 *Par abus de langage, le terme 'assembleur' souvent désigne aussi le logiciel ou compilateur utilisé pour traduire un code assembleur en code machine.*

Remarque 5 *le tableau de l'ensemble des instructions de l'assembleur du PIC18F est donné dans l'annexe 3.*

3.2 La forme d'une instruction assembleur

La forme d'une instruction en assembleur est liée au code machine de cette instruction et par conséquent a l'architecture adopté par le constructeur.

Dans le cas des pic18F (et 16F) de Microchip un registre accumulateur WREG (Working Register) est utilisé comme opérande(ou même comme opérande unique de certains cas).

Outre cette particularité, l'adresse d'un opérande dépend du mode d'adressage utilisé. Si l'adresse mémoire dépasse 8 bits, le registre BSR est utilisé pour compléter l'adresse de l'opérande.

Les figures 1 et 2 illustrent des exemples de code machine d'instructions affectant un octet de mémoire, ou un bit particulier d'une case mémoire.

On note que :

- Le terme 'file' (ou la lettre f) indique une case de la mémoire

- Le bit 'd' désigne la destination du résultat.

Si d=0, le résultat est affecté au registre WREG

Si d=1, le résultat est affecté à la case mémoire (file)

- Le bit 'a' désigne si l'adresse de la case mémoire doit être complétée avec le contenu du BSR

Si a=0, le BSR n'est pas considéré

Si a=1, le contenu du BSR représente les 4 bits de poids forts de l'adresse mémoire (file).

Le tableau du jeu d'instruction fourni par le constructeur (annexe 3) précise la forme exacte de chaque instruction.

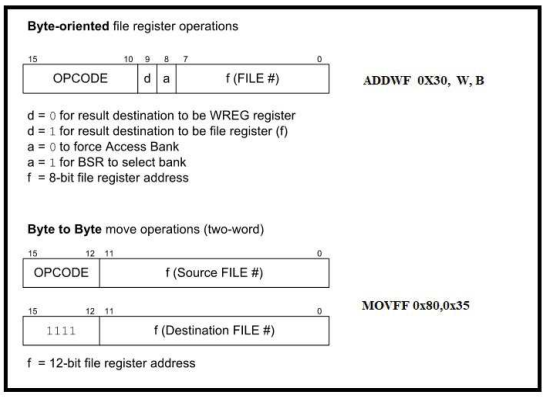


Figure1 : Exemples de code machine
d'instructions affectants des octets

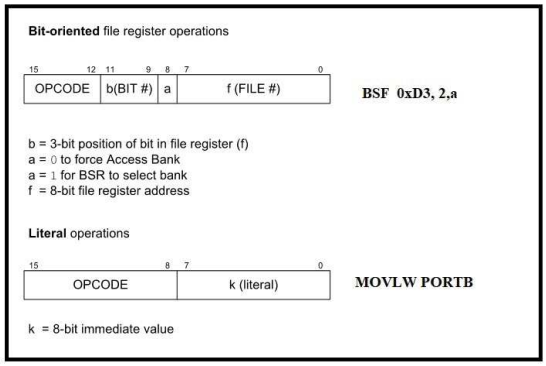


Figure 2 : Codes machines d'instructions

Dans le cas général elle peut s'écrire sous la forme :

Mnémonique de l'instruction f , d, a
f : *l'adresse mémoire*
d={ 0 ou w => résultat dans WREG
1 ou f => résultat dans la memoire
a={ 0 ou A => BSR n'est pas inclu dansl'adresse
1 ou Banked => BSR est inclu dans l'adresse

MOVLW 0x30	WREG reçoit la valeur $(30)_{16}$
ADDLW 0x0A	La valeur du WREG devient $(30+0A)_{16}=(3A)_{16}$
MOVLB 0x02	BSR reçoit la valeur $(02)_{16}$
MOVWF 0x50,0 MOVWF 0x50,A	L'octet 0x50 reçoit le contenu du WREG
MOVWF 0x50,1 MOVWF 0x50,BANKED	L'octet 0x250 reçoit le contenu du WREG
ADDWF 0x50,f,A ADDWF 0x50,1,0	addition de WREG et l'octet 0x50(résultat dans 0x50)
ADDWF 0x50,w,BANKED ADDWF 0x50,0,1	addition deWREG et l'octet 0x250(résultat dans WREG)

Exemple 6 (Un template de programme)

Le programme ci-dessous est un exemple de programme assembleur pour microcontrôleurs pic18F458.

Outre les instructions assembleur on trouve aussi un ensemble de directives.

On rappelle que les directives sont des ordres destinés au compilateur utilisé pour effectuer certaines tâches, et par conséquent ne possèdent pas de code machine.

```
 LIST p=18f458 , =decr
       INCLUDE <p18f458.inc>
       CONFIG Fosc=HSMP,WDTEN=OFF
val    EQU 0xA0
       ORG 0x00
       GOTO main
main MOVLW 'A'
       MOVWF 20h
       MOVLB 0x01 ;bank 1
       MOVLW 'B'
       MOVWF val
       MOVLB 0x02 ;bank 2
       MOVLW 'C'
       MOVLB 0x03 ;bank 3
       MOVLW 'D'
       MOVWF PORTB
       END
```

Dans cet exemple les directives utilisées sont :

- **LIST** : définit certains paramètres
  p : le processeur
  r : la base par défaut (DEC=10,HEX=16,BIN=2)
- **INCLUDE** : indique au compilateur les fichiers de définition. Dans le cas de cet exemple, le fichier 'p18f458.inc' du constructeur contient les définitions des symboles tels que STATUS, PORTB, BSR...
- **ORG** : indique l'adresse de début de la séquence d'instructions qui la suit.
- **EQU** : directive d'équivalence.
- **END** : indique la fin du fichier à compiler.
- **CONFIG** : indique au compilateur les paramètres de configuration du microcontrôleur. Cette zone (16 bits) n'est accessible que pendant la phase de flashage du circuit. Parmi ces paramètres on trouve :

- Des bits fixant le type du circuit d'horloge utilisé (déja décrits dans le tableau 1) ,et qui sont remplacé par la directive FOSC :

FOSC={	**HSMP**	Oscillateur medium power (4-16MHz)
	HSFOSC=HP	Oscillateur high power (>16MHz)
	ECHP	Horloge externe high power
	RC	Oscillateur externe (circuit RC)

- Un bit activant ou désactivant le Watch Dog Timer (WDT)

WDTEN={ 1 WDT activé
0 WDT désactivé

3.3 Les modes d'adressage

Un mode d'adressage définit la façon avec laquelle est exprimé l'opérande dans une instruction donnée. Le jeu d'instruction du PIC18F possède différents modes dont les principaux sont définit dans cette section.

3.3.1 Le mode littérale

Dans ce cas l'opérande est exprimé litéralement par une valeur numérique directe. La destination est généralement le registre accumulateur WREG, et dans certains cas le BSR, ou le FSR.

Instructions	**Conséquences**
MOVLW 0xD5	**WREG** reçoit la valeur $(D5)_{16}$
ADDLW 0x0A	La valeur du **WREG** devient $(WREG + 0A)_{16}$
MOVLB 0x05	**BSR** reçoit la valeur $(05)_{16}$
LFSR 0x3AB	**FSR** reçoit la valeur $(3AB)_{16}$

3.3.2 L'adressage de la mémoire

Le principe des modes d'adressage de la mémoire a été déjà introduit lors de la présentation de la forme d'une instruction de l'assembleur PIC18F.

On distingue différents modes :

L'adressage direct simple

Dans ce cas l'instruction est codée sur 16 bit (figure 1). Le bit 'a' indique si l'adresse de l'opérande doit être étendue à 12 bits avec le contenu du BSR.

Exemple 7

L'instruction d'addition du contenu du **WREG** avec le contenu d'une case mémoire s'écrit sous la forme générale : (**ADDWF f, d, a**)

En mode 'ACCESS BANK' (Adresse sur 8 bit)

Instructions	**Conséquences**
ADDWF 0x80, f, A (ou **ADDWF** 0x80, 1, 0)	Addition du **WREG** et l'octet 0x80 (résultat dans la case 0x80)
ADDWF 0x80, w, A (ou **ADDWF** 0x80, 0, 0)	Même opération d'addition (résultat dans **WREG)**

En mode 'BANKED' (Adresse sur 12 bit, BSR inclu)

Instructions	Conséquences
MOVLB 0x04 **ADDWF** 0x80, f, BANKED (ou **ADDWF** 0x80, 1, 1)	**BSR** reçoit la valeur 4 Addition du **WREG** et dela case mémoire 0x480, Résultat affecté à la case 0x480
MOVLB 0x04 **ADDWF** 0x80, w, BANKED (ou **ADDWF** 0x80, 0, 1)	Même opération, avec le résultat dans le **WREG**

Le mode direct long

Dans ce mode l'instruction est codée sur 32 bits, et les adresses complètes (12 bits) des deux opérandes sont inclues.

Ce mode concerne l'instruction de transfert : (**MOVFF** source, destination).

Exemple 8

Instructions	Conséquences
MOVFF 0x10, 0x350	Transfert du contenu de 0x10 vers 0x350
MOVFF 0xC0, PORTB	Transfert du contenu de 0xC0 vers le **PORTB**
MOVFF WREG,0x3FC	Transfert du contenu du **WREG** vers 0x3FC

L'adressage indirect simple

Dans ce mode, l'adresse de l'opérande est spécifiée dans l'un des 3 registres d'adresse (FSR0, FSR1, FSR2).

Pratiquement, vu l'architecture 8 bits du microcontrôleur, chaque registre $FSR_{(0,1,2)}$ est implémenté par deux registres pour couvrir les 12 bits d'adresse :

FSR0H(4bit)+FSR0L(8bit)→FSR0

FSR1H(4bit)+FSR1L(8bit)→FSR1

FSR2H(4bit)+FSR2L(8bit)→FSR2

A chaque pointeur $FSR_{(0,1,2)}$ est associé un registre de donnée $INDF_{(0,1,2)}$ permettant l'accès au contenu de la case mémoire pointée.

Il est important de noter que les registres $INDF_{(0,1,2)}$ ne sont pas physiquement implémentés.

Exemple 9

Instructions	Conséquences
MOVLW 0x100	**WREG** reçoit la valeur 0x100
MOVWF FSR0	**FSR0** reçoit la valeur 0x100
LFSR 02,0x300	**FSR2** reçoit la valeur 0x300
MOVWF,INDF2	L'adresse 0x300 reçoit la valeur 0x100
MOVLW 0xA8	**WREG** reçoit la valeur 0xA8
MOVWF,INDF0	L'adresse 0x100 reçoit la valeur 0xA8

l'adressage indirect post-décrémenté

Dans ce mode indirecte le pointeur FSR(0,1, 2) est décrémenté automatiquement après l'exécution de l'instruction.

Cela est possible en utilisant le mot réservé POSTDEC(0,1, 2).

Instructions	Conséquences
LFSR 02,0x205	FSR2 reçoit la valeur 0x205
MOVFF POSTDEC2,0x500	Transfert du contenu de 0x205 vers 0x500 Décrémenter FSR2=205-1=$(204)_{16}$

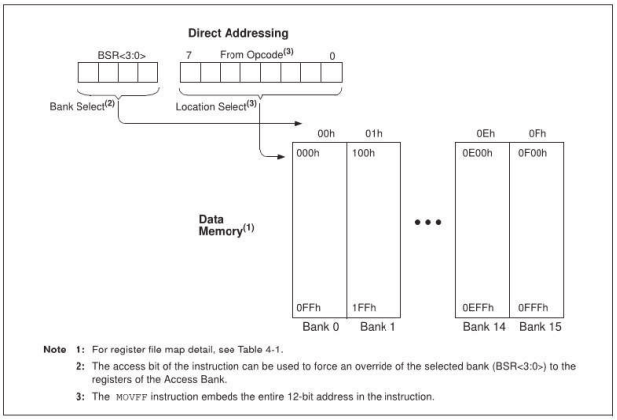


Figure 3 : Principe de l'adressage de la memoire

l'adressage indirect post-incrémenté

Dans ce cas le pointeur FSR(0,1,2) est incrémenté automatiquement après l'exécution de l'instruction.

Cela est possible en utilisant le mot réservé POSTINC(0, 1, 2)

Instructions	**Conséquences**
LFSR 02,0xAB	FSR2 reçoit la valeur 0xAB
MOVFF POSTINC2,0x100	Transfert du contenu de la case 0xAB vers 0x100 Incrémenter FSR2=AB+1=$(AC)_{16}$

l'adressage indirect pré-incrémenté

Dans ce mode l'incrémentation du pointeur est effectuée avant l'exécution de l'instruction via le mot réservé PREINC(0, 1, 2).

Instructions	**Conséquences**
LFSR 02,0xAB	FSR2 reçoit la valeur 0xAB
MOVFF PREINC2,0x100	Incrémenter FSR2=AB+1=$(AC)_{16}$ Transfert du contenu de la case mémoire 0xAC vers 0x100

L'adressage indirect pré-indexé

Dans le mode pré-indexé, l'adresse de l'opérande est la somme du FSR et du WREG. Dans ce cas le contenu du FSR n'est pas affecté, et le mot réservé est PLUSxW (x=0, 1, 2). En fin de compte ce mode permet un déplacement du pointeur par rapport à une référence.

Instructions	**Conséquences**
LFSR 03,0x20	FSR3 reçoit la valeur 0xAB
MOVLW 0x05	WREG reçoit 0x05
MOVFF PLUS3W,0x100	Transfert du contenu de 0x25 vers 0x100 FSR3 ne change pas

3.4 Les instructions de base

L'utilisation des instructions que permet le langage assembleur doit etre conforme au tableau fournit par le constructeur (annexe 3). Dans ce tableau on précise la forme de l'instruction, son code machine, ainsi que son effet sur les bits (flags) du registre d'état STATUS.

Différents exemples ont déjà été donnés au cours des sections précédentes.

Dans cette section les formes de quelques instructions de base sont données.

3.4.1 Les instructions affectant un octet

MOVWF f,a	Transfert de WREG vers l'adresse mémoire f
MOVF f,d,a	Transfert l'adresse mémoire f vers WREG (ou f)
CLRF f,a	Remise à zéro de la case mémoire f
COMF f,d,a	Complément à 1 de la case mémoire f
ADDWF f,d,a	Résultat de l'opération (WREG + f) transféré vers WREG (ou f)
XORWF f,d,a	Ou exclusif de WREG avec f, et résultat dans WREG (ou f)

3.4.2 Les instructions affectant un bit d'un octet

BCF f,b,a	Remise à 'zéro' d'un bit 'b' de l'octet 'f'
BSF f,b,a	Mise à 'un' d'un bit 'b' de l'octet 'f'
BSF f,b,a	Inversion d'état du bit 'b' de l'octet 'f

D**'autres instructions :**

NOP	Aucune opération (mais dure 4 cycle d'horloge)
RESET	Réinitialisation software du microcontrôleur

3.4.3 Les boucles et les branchements (instructions de contrôle)

Le jeu d'instructions du PIC18F offre deux types de branchement conditionnel, et inconditionnel.

Dans le cas conditionnel les bit flags du registre d'états STATUS sont utilisés.

Apres l'exécution d'une opération les instructions de contrôle permettent de tester le résultat via les flags (N,OV,Z,DC,C) du registre d'état (figure 4), pour ensuite par exemple effectuer un branchement conditionnel.

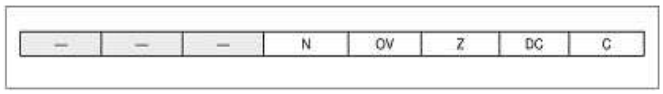


Figure 4 : Les bits flags du registre STATUS

Le tableau suivant liste quelques instructions de branchement

BC	Branchement si la retenue C=1
BNC	Branchement si la retenue C=0
BN	Branchement si le résultat est négatif
BZ	Branchement si le résultat est zéro
BNZ	Branchement si le résultat n'est pas nul
BTFSS f,b,a	Sauter l'instruction suivante si le bit 'b' de f est égale à 1
BTFSC f,b,a	Sauter l'instruction suivante si le bit 'b' de f est égale à 0
BRA n	Branchement inconditionnel
GOTO n	Branchement inconditionnel

Exemple 10 *L'organigramme de la figure 5 peut être traduite en assembleur comme suit*

```
            ORG 0x00
            ; Debut du programme
            ...
            BZ T2;
            Tache 1
            ...
            GOTO SUITE
T2          ; Tache 2
            ...
SUITE
            ; Suite du programme
            ...
        END
```

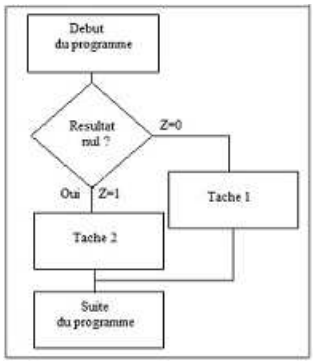


Figure 5 : Exemple d'organigramme d'un branchement conditionnel

3.5 La pile et les sous-programmes

Une pile (stack) est une zone mémoire de type LIFO (Last In First Out) pointé avec un registre pointeur de pile (stack pointer :STKPR). Cette zone mémoire est utilisée en particulier pour la sauvegarde des adresses de retour dans le cas d'appel de sous-programmes.

Une opération d'écriture dans la pile est précédée par une décrémentation du pointeur pour assurer un empilement dans la mémoire.

Par contre, une opération de lecture correspond à une incrémentation du pointeur assurant ainsi un dépilement de la pile.

Dans le cas du PIC18f458 (Figure 6), la pile est une zone mémoire de 31 mots (de 21 bits), par conséquent permettant 31 appel de sous-programmes (ou d'interruptions) imbriqués.

Ainsi La valeur du PC est empilée dans la pile après la validation d'une interruption ou l'exécutions des instructions : PUSH , CALL, ou RCALL.

L'adresse est récupérée par le PC après l'exécution des instructions : RETURN, RETLW, RETFIE.

On note que les instructions de type CALL provoquent l'incrémentation du pointeur de pile STKPTR avant la sauvegarde du contenu du PC, et vice versa les instructions de type CALL provoquent sa décrémentation après la récupération de l'adresse de retour.

Remarque 11 *La dernière adresse sauvegardée dans la pile (Top-Of-Stack) est accessible via le registre TOS (TOSU, TOSH, et TOSL).*

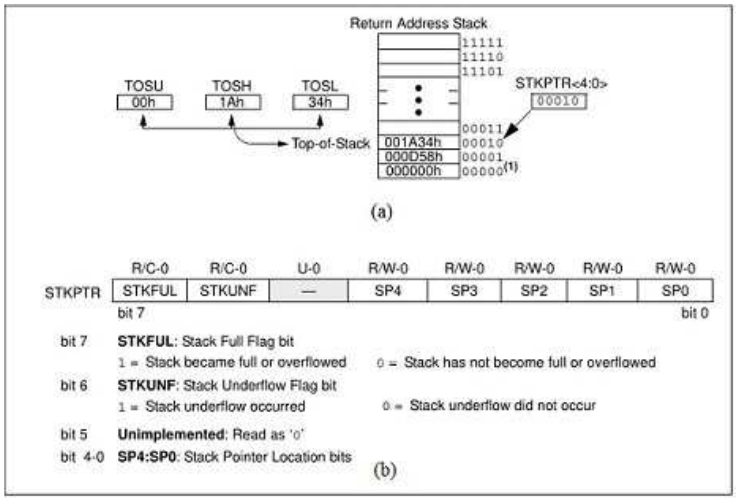


Figure 6 : La pile (a), et son registre pointeur (b)

3.5.1 Les instructions CALL, RCALL

L'instruction **CALL** provoque le branchement vers une adresse k (du sous-programme), et la sauvegarde de l'adresse de retour dans la pile.

CALL k,s : { 0≤k≤1048575 (20 bits)
PC+4 (adresse de retour)→**TOS** (Top Of Stack)
K (adresse de sous-prog)→ **PC**
Mode **FAST** si s=1

L'instruction **RCALL** est une variante de call où le déplacement est relatif, et codé sur 10 bits

RCALL k : { -1024≤k≤1023 (10 bits)
PC+2(adresse de retour)→**TOS** (Top Of Stack)
(**PC**)+2+K(adresse de sous-prog)→ **PC**

3.5.2 Les instructions RETURN, RETLW

A la fin d'un sous-programme la restitution de l'adresse de retour, et la décrémentation du pointeur de pile sont nécessaires. Pour cela, deux instructions sont disponibles :

RETURN s (mode FAST si s=1)
RETLW K (0≤k≤255)

L'instruction RETLW permet un retour avec une valeur literale dans le register WREG.

Remarque 12 *Les instructions d'appel et de retour de sous-programme durent 2 cycle instruction (8 cycles horloge).*

Exemple 13

Le programme	**Description**
function **DECFSZ** 80h,f **RETLW** 1 **RETLW** 0	Une fonction qui décrémente la case mémoire 80h, et retourne soit 0 ou 1 dans le registre WREG

Exemple 14 (une temporisation logicielle)

On exploite le fait que les instructions durent un cycle ou deux (cas des branchements), et que le cycle instruction vaut 4 périodes d'horloge.

Le programme		Nombre de cycles instruction
n **EQU** 6		
	ORG 0x00	
	; le programme principal	
	...	
	...	
	CALL Temp	2
	...	
	...	
Temp	**MOVLW** n	1
	MOVWF 70h	1
ret	**DECFSZ** 70h,f	1 ou 2 (s'il y a branchement)
	GOTO ret	2
	RETURN	2

Nombre total de cycles instruction=2+1+1+(n-1)(1+2)+2+2=(3n+5)
Si la fréquence de l'horloge est 4Mhz, alors :
Le cycle instruction=4. (1/4MHz)=1us
Pour n=6 la durée de la temporisation logicielle est par conséquent 23us

3.5.3 Le mode 'FAST'

Dans le cas où le mode FAST est activé, les registres STATUS, WREG, et BSR sont sauvegardés dans des registres internes d'ombre (STATUSS, WREGS, BSRS) avant l'exécution du sous-programme et sont restitués après l'instruction de retour.

3.6 Exercices

Exercice 1 :

Exécuter le programme ci-dessous pas à pas,

-Donner le résultat , ainsi que le contenu du registre PC, et des indicateurs (N,OV,Z,DC,C)du registre STATUS après l'exécution de chaque instruction ?

- Déterminer le temps d'exécution du programme par un microcontrôleur cadencé avec une horloge externe dev 8MHz ?

```
        ORG 0x0000
        GOTO START
START   MOVLW 0xF5
        ADDLW 0x06
        SUBLW 0xFB
        XORLW 0XFF
        MOVLB 0X02
        MOVWF 50H,0
        MOVWF 50H,1
        NOP
        GOTO START
        END
```

Exercice 2 :

Donner un programme assembleur qui copie :
45 dans la position 20h,
'B' dans la position A0h
-4 dans la position 110h,
35h dans la position 190h

Exercice 3 :

Donner un programme assembleur réalisant les opérations suivantes :
- L'addition de la constante 33 à la valeur de l'accumulateur (W+33)
- La soustraction de la constante 40h de la case mémoire 75h ([75h]-40h)
- La soustraction du contenu de la case mémoire 75h de l'accumulateur W avec le résultat dans W (W – [75h] —> W)
- La soustraction du contenu de la case mémoire 76h de l'accumulateur W avec le résultat dans la case mémoire (W – [76h] —> [76h])

Exercice 4 :

Comparer les contenus des cases mémoire 6Fh et EFh, s'il son égaux mettre à zéro tous les bits de la case 16Fh sinon mettre à 1 tous les bits de la case 1EFh

Exercice 5 :

Comparer les contenus des cases mémoire 6Fh et EFh,
- Si [6Fh] = [EFh] copier la lettre E dans la case mémoire 16Fh
- Si [6Fh] > [EFh] copier la lettre S dans la case mémoire 16Fh
- Si [6Fh] < [EFh] copier la lettre I dans la case mémoire 16Fh

Exercice 6 :

En utilisant un sous-programme, donner un programme qui copie l'alphabet majuscule dans la RAM à partir de la position 120h

Exercice 7 :

En utilisant un sous-programme, donner un programme réalisant la permutation des données de la zone mémoire A0h –EFh et ceux de la zone mémoire 120h-16Fh

Chapitre 4

Les ports d'entrée/sortie numérique

4.1 Introduction

Les pins d'entrée-sortie numériques du PIC18F458 sont organisées en 6 ports, dont le principe de fonctionnement est relativement le même, puisqu'il s'agit de pins bidirectionnelles et multiplexées fonctionnant avec d'autres modules tels que les Timers, l'ADC, ou le module de communication série.

Le tableau 1, donne le nombre de bits de chaque port, et les registres SFR associés à leurs configurations.

Le PORT	**Nombre de Bits**	**Registres associes**	**Remarque**
PORTA	7 (RA0-RA6)	PORTA, TRISA, LATA, ADCON1	5 pins multiplexées avec l'ADC
PORTB	8 (RB0-RB7)	PORTB, TRISB, LATB, INTCON2	
PORTC	8 (RC0-RC7)	PORTC, TRISC, LATC	
PORTD	8 (RD0-RD7)	PORTD, TRISD, LATD	
PORTE	3 (RE0-RE2)	PORTE, TRISE, LATE, ADCON1	3 pins multiplexées avec l'ADC

Tableau 1 : Les ports numériques,et les registres SFR associés

La conception des ports diffère d'un port à un autre et même d'une pin à une autre du même port. La figure 1 illustre deux exemples (RA5, RB0). RA5 serve aussi comme une entré du convertisseur analogique/numérique, par contre RB0 peut être programmée comme source d'interruption externe.

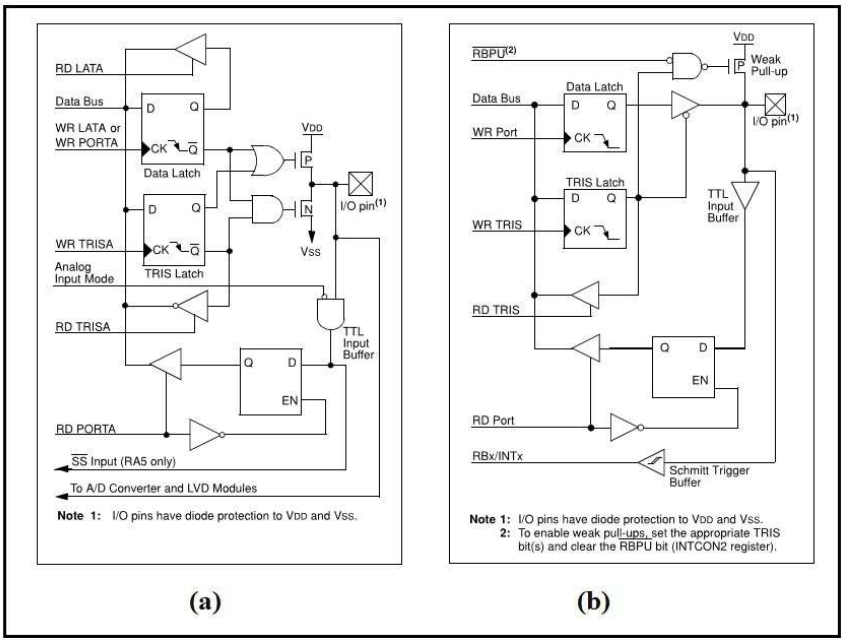


Figure : Diagramme bloc des pins :(a) RA5, (b) RB0

Le registre TRISx :

Les registres **TRISx(A, B, C, D, E)** sont les registres de direction permettant de programmé individuellement en entrée ou en sortie chaque pins :

- une pin particulière d'un port est configurée en sortie si le bit correspondant dans le registre **TRISx** est mis à 0. La mise à 1 correspond à une configuration en entrée :

{ si **TRISx**(i)=0 → la pin i est configuré en sortie
{ si **TRISx**(i)=1 → la pin i est configuré en entrée

Le registre LATx :

Une opération d'écriture dans un port correspond à une affectation du registre dit 'data latch' **LATx(A, B, C,D, E)**, qui mémorise la dernière valeur écrite dans le port (Figure 1).

Par contre une opération de lecture d'un **PORTx** correspond à la lecture de l'état de la pin en sortie. De ce fait les valeurs du PORTx et celle du **LATx** peuvent être différentes.

4.2 Le PORTA

Un port 7 bits, dont le sens de transfert (direction) est configuré avec le registre **TRISA**.

Les pins (**RA0-RA3**, et **RA5**) sont configurables comme entrée analogique via le registre **ADCON1**.

Si la valeur 7 est affectée à ce registre, alors tout le port en mode numérique [1]

La pin **RA4** possède la particularité d'être une sortie à drain ouvert, par conséquence nécessite une charge de 'pull up' pour un fonctionnement correcte.

Exemple 15

```
CLRF PORTA      ; Initialisation du PORTA
CLRF LATA       ; Une autre alternative de mise ;
                ; à zéro du PORTA
MOVLW 07h       ; Configuration du port
MOVWF ADCON1    ; en mode numérique
MOVLW 0CFh      ; Valeur pour registre de direction
MOVWF TRISA     ; RA3 :RA0 (pins en entrée)
                ; RA5 :RA4 (pins en sortie)
```

Remarque 16 *:*

Apres un reset les pins du port sont configurées comme entrées analogiques.

4.3 Le PORTB

Un port 8 bits bidirectionnelles configurables avec le registre **TRISB**.

Chaque pin du **PORTB** possède une faible charge de 'pull up' (figure 1.b) qui peut être désactivée via une mise à un (1) du bit ($\overline{\mathbf{RBU}}$) du registre **INTCON2**. Cette charge est automatiquement désactivée quand le port est configuré en sortie numérique.

Remarque 17 *:*

Quatre pins (**RB4-RB7**) peuvent être configurées comme source d'interruption externe. L'interruption n'est possible que si la pin du port est configurée comme entrée.

4.4 Le PORTC

Le **PORTC** est un 8 bits bidirectionnel configurable avec le registre **TRISC**.

Les lignes de ce port sont liées aussi à d'autres modules tels que le **Timer 1** ou la liaison serie (**USART, I2C, et SPI**).

4.5 Le PORTD

Le **PORTD** est un 8 bits bidirectionnel configurable avec le registre **TRISD**.

Il a la particularité de pouvoir fonctionner en mode **PSP** (Paralel Slave Port), c'est à dire un port contrôlé par des lignes externe. Ce fonctionnement en mode **PSP** est lié au **PORTE** (voire la section 4.7).

4.6 Le PORTE

Les 3 lignes du **PORTE** fonctionnent en 3 modes differents :
- Des pins E/S numériques (**RE0 , RE1, RE2**)
- Des lignes de contrôle **PSP ((**$\overline{\mathbf{RD}}$**), (**$\overline{\mathbf{WR}}$**),(**$\overline{\mathbf{CS}}$**)**)
- Des entrées analogiques (**AN5, AN6, AN7**).

Outre la direction des données que décident les bits <0,1,2> du registre **TRISE**, la sélection du mode **PSP** est réalisée via le bit <4>.

IBF	OBF	IBOV	PSPMODE	—	TRISE2	TRISE1	TRISE0

Par conséquent pour configurer le port en E/S numérique il faudra désactiver le mode **PSP** (**TRISE**<4>=0), ainsi que le mode analogique (par exemple affecter un 7 au registre **INTCON1**).

```
CLRF PORTE      ; initialisation du PORTE
                ; avec un zéro en sortie
CLRF LATE       ; une autre alternative
MOVLW 0CFh      ; valeur pour registre de direction
MOVWF TRISE     ; RE1 :RE0 en entrée, RE2 en sortie
                ; PSPMODE désactivé (RE4=0)
```

4.7 Le mode PSP :

Dans ce mode un module externe (maître) peut écrire et lire du PORTD du microcontrôleur (qui devient esclave) via les lignes de contrôle du PORTE (figure 2).

L'opération de lecture en mode PSP (figure 3.a) n'est réalisée qu'après le passage au niveau bas des lignes ($\overline{WR}$) , et ($\overline{CS}$) .

Par contre l'opération d'écriture est caractérisée par le passage des lignes ($\overline{RD}$) , ($\overline{CS}$) au niveau bas (figure 3.b).

Les bits IBF OBF IBOV du registre TRISE permettent à l'esclave de connaitre l'état de la transmission PSP :

- Si **IBF**=1 => Une donnée est reçue sur le **PORTD**
-Si **OBF** =0 => La donnée sur le **PORTD** a été lue
- Si **IBOV**=1 => Un débordement (overflow), ou une nouvelle écriture a eu lieu avant la lecture de l'ancienne donnée déposée.

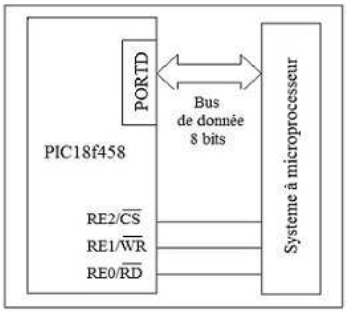


Figure 2 : Schema bloc de principe du mode PSP

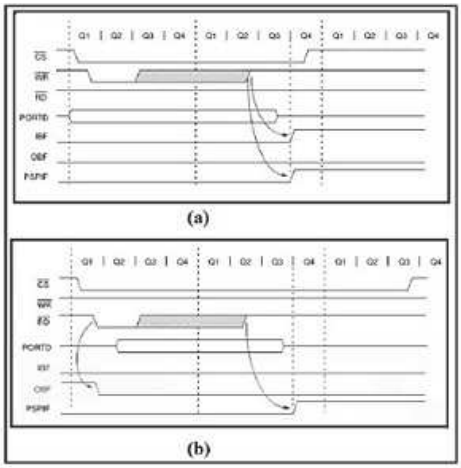


Figure 3. Chronogramme du mode PSP :(a) cycle d'ecriture (b) cycle de lectur [1]

4.8 Exercices

Exercice 1 :

Sachant qu'on dispose en mémoire d'une temporisation logicielle d'une seconde 1s (le sous programme time).

- Développer un programme assembleur permettant l'allumage séquentiel des LEDs connectés aux pins RB0-RB4 (Figure 1).

Ce même programme doit permettre l'arrêt de l'allumage après l'appui du bouton poussoir « B.P ».

- Si le microcontrôleur utilise un oscillateur 4Mhz proposer une temporisation logicielle de 100ms.

- Modifier le programme si si le port utilisé est le PORTA

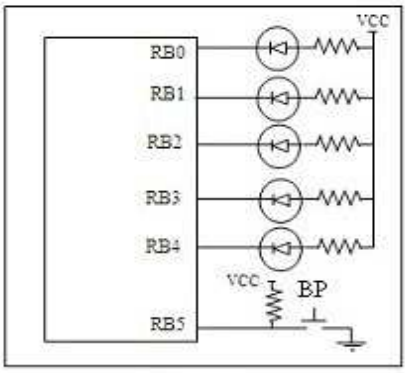


Figure 1

Exercice 2 :

En considérant le montage de la figure 1 développer un programme qui :

- Allume la LED branchée sur RB0
- Compte 150 impulsions sur l'entrée RB5
- Eteint la LED branchée sur RB0

Exercice 3 :

On branche un afficheur 7 segments cathode commune sur le port B suivant la figure 2.

– Développer un programme assembleur permettant d'incrémenter le contenu de l'afficheur a chaque appuie du bouton poussoir

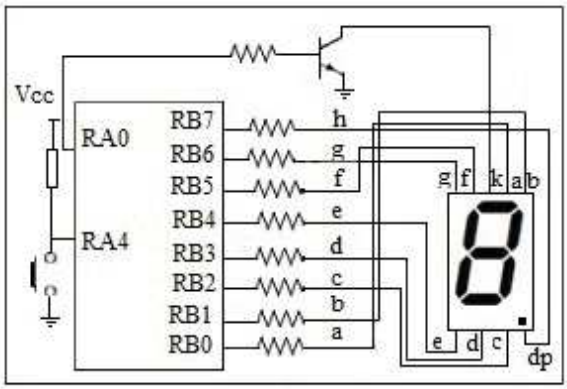

Vcc
RA0
RA4
RB7
RB6
RB5
RB4
RB3
RB2
RB1
RB0
h
g
f
e
d
c
b
a
g f k a b
e d c dp

Figure 2

Chapitre 5

Les modules Timers et CCP

5.1 Introduction

Ce chapitre est consacré à la présentation des Timers, et du module 'Capture /Compare /PWM' (CCCP). C'est deux modules sont pratiquement présents dans tous les microcontrôleurs de la série PIC18Fxxx

Dans Le cas du PIC18F458 quatre Timers de taille différentes sont intégrés (tableau 1), mais dont le principe de fonctionnement est identique. Le Timer 0 est presenté avec plus de détails que les autres en se limitera au fonctionnement en mode scrutation du bit flag **TOIF**. Le fonctionnement en mode interruption sera traité dans le chapitre suivant.

Module	**Taille**	**Fonction**
Timer 0	8-16 bits	Timer/ counter
Timer 1	16 bits	Timer/ counter
Timer 2	8 bits	Timer/ PWM (8 bits)
Timer 3	16 bits	Timer/ counter

Tableau 1 : les TIMERs du PIC18F458

5.2 Les Timers

Un Timer est un compteur synchrone dont le débordement (overflow) indique la fin de la temporisation. Ce débordement se produit lors du passage de 0xFF à 0x00 (mode 8 bits) ou de 0xFFFF à 0x0000 (mode 16 bits).

5.2.1 Le Timer 0

. Ce Timer a la possibilité de fonctionner soit en mode 8 bits soit 16 bits.
. Un prescalaire (codé sur 3 bits) est disponible

. L'horloge peut être interne (Fosc/4) ou externe (la pin **RB4/T0CKi**

.possibilité de sélectionner le flanc montant ou descendant du signal d'horloge (externe)

. L'overflow (passage de 0XFF à 0x00 ou de 0XFFFF à 0x0000) peut être programmé pour provoquer une interruption.

Les registres associés

. Deux compteurs 8bits sont utilisés : **TMR0H**, et **TMR0L** pour pouvoir fonctionner soit en 8 soit en 16 bits.

. Le registre **T0CON** (figure 1) permet une configuration complète du Timer :

Lancement/arrêt, sélection de l'horloge, sélection du pré-scalaire, et du flanc montant ou descendant du signal (cas d'horloge externe).

. le registre **INTCON** permet plusieurs tâches liées à la gestion des interruption

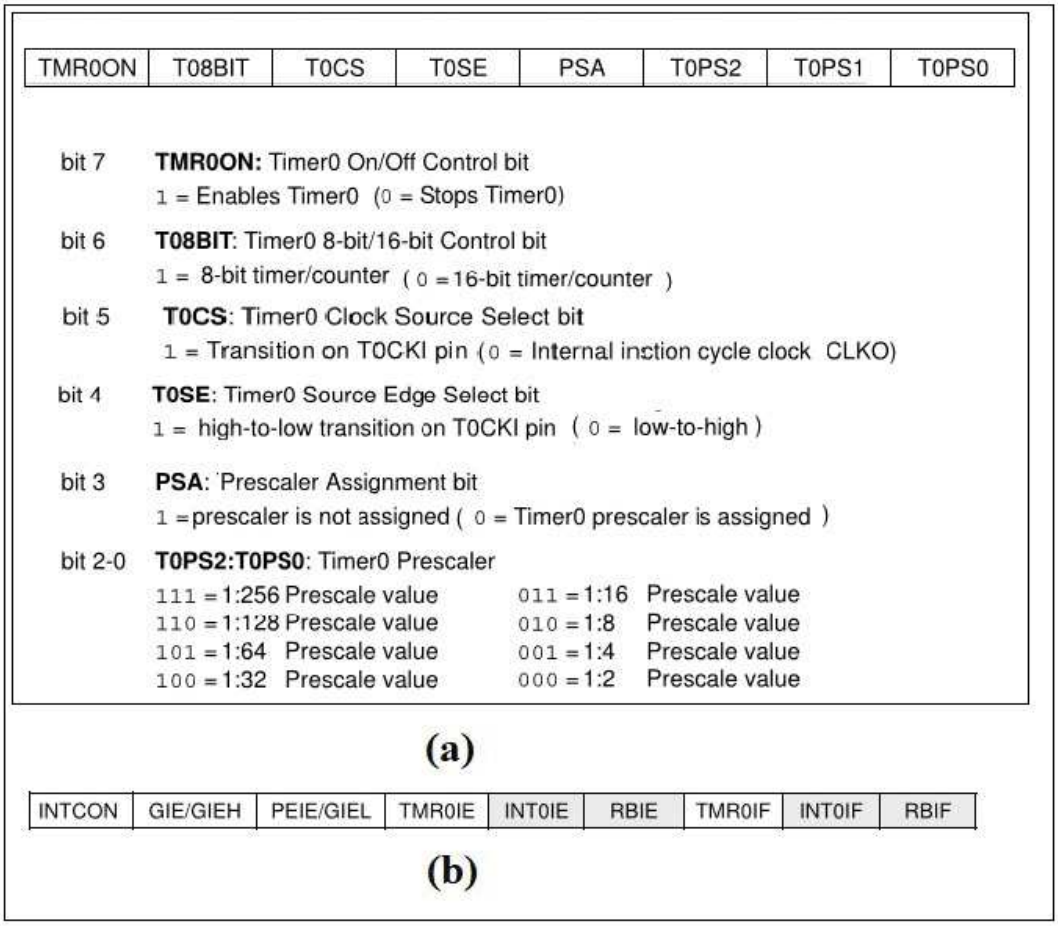


Figure 1 : Le registre de controle T0CON

Principe de fonctionnement

Le Timer 0 peut fonctionner soit comme timer ou comme compteur.

En mode timer il subit une incrémentation après chaque cycle d'instruction (si le pré-scalaire n'est pas activé). Par contre en mode compteur l'incrémentation se produit après chaque flanc montant (ou descendant) au niveau de la pin **RA4/T0CKI**.

Le pré-scalaire (pré-diviseur) est une valeur codée sur 8 bits qui permet de multiplier la période de la temporisation produite par des puissances de deux (2, 4,8,. . . .128 ,256)

La configuration du mode de fonctionnement et le choix du pre-scalaire sont réalisés via le registre de contrôle T0CON (figure 1.a).

Un bit (T0IF) dans le registre INCON (figure 1.b) est le flag indicateur de la fin de la temporisation et de la production d'une interruption Timer 0 (si elle activée).

Les principes de fonctionnement en mode 8 et 16 sont pratiquement identiques (figure 2). Le bit T0IF passe à 1 après le débordement du registre TMR0L (passage de 0xFF à 0x00 en mode 8 bits) , ou débordement de l'ensemble TMR0H-TMR0L (passage de 0xFFFF à 0x0000 en mode 16 bits).

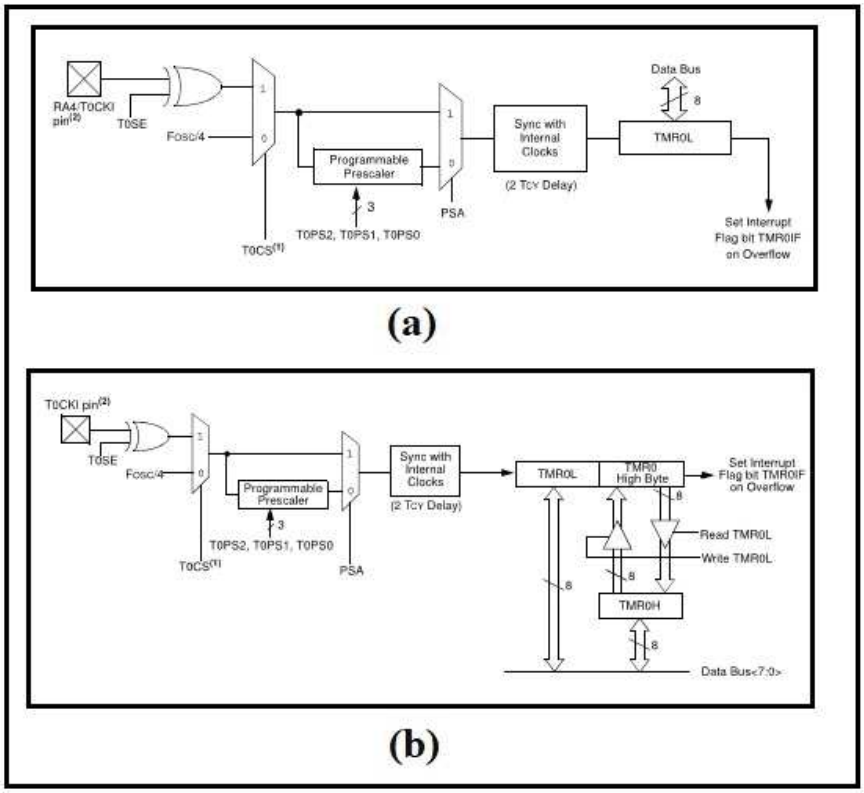


Figure 2 : Schema bloc de principe du Timer0 :mode 8 bits (a), mode 16 bits (b)

Calcul de la période

Si le timer est chargé avec une valeur initiale 'Ninit' la temporisation realisée est :

Ttotal= (FF+1- Ninit) . Tcycle . **PSC** (mode 8 bits)

Ttotal= (FFFF+1- Ninit) . Tcycle . PSC (mode 16 bits)

Où Tcycle est la durée d'un cycle instruction égale à 4/Fosc

PSC est la valeur du pre-scalaire

Soi un PIC18F458 cadencé avec une horloge Fosc de 4MHz.

Tcycle =4/Fosc=1us

En mode 8 bits, si **TMR0L** est chargé avec la valeur 00, alors le débordement se produira après 256 cycles.

Sans pré-scalaire, Ttotal=256us

Avec un pré-scalaire de 02, Ttotal=2 . 256us=512us

Avec un pré-scalaire de 256, Ttotal=256 . 256us=65.536ms

La fin d'une temporisation programmée avec le Timer 0 peut être configurée pour provoquer une interruption. En mode scrutation est bloqué sur le test du flag **INTCON.T0IF** (organigramme de la figure 4)

L'organigramme de la figure 3, génère une impulsion positive sur les lignes du **PORTB** en exploitant le **Timer 0**.

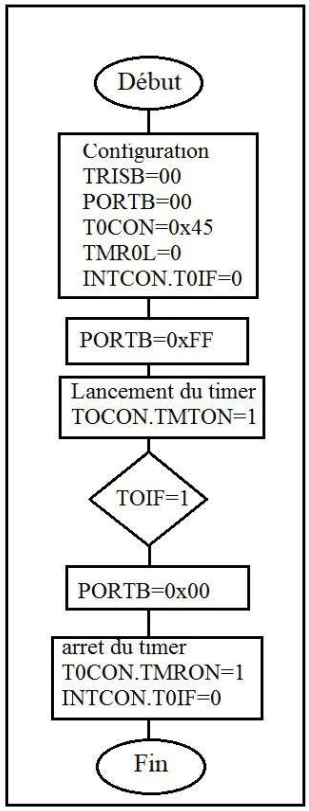


Figure 3 : exemple d'organigramme de scrutation du flag T0IF

. T0CON=0x45=(01000101)2

{ TIMER 0 { configuré en mode 8 bits
cadencé avec l'horloge interne
Pré-scalaire=64

. Si le microcontrolleur est cadencé avec une horloge Fosc de 4MHz, alors :

Tcycle =4/Fosc=1us.

. TMR0L =00, alors le débordement du timer se produit après :

Ttotal=256. 64 . 1us=16.384 ms

En considérant l'organigramme de la figure 3, le programme assembleur peut s'écrire :

```
        ORG 0X00
        CLRF PORTB
        CLRF TRISB ; PORTB en sortie

        MOVLW 0x45 ;Configuration du timer 0
        MOVWF T0CON
        CLRF TMR0L

        SETF PORTB
        BSF T0CON, 07 ; Lancement du timer 0
        BFSSC INTCON ,T0IF
RT      BRA  RT
        CLRF PORTB
        BCF 07 , T0CON
        BCF T0CON, T0IF
        ...
```

Question : en considérant l'implémentation proposée en assembleur, déterminer avec précision la durée de l'impulsion générée ?

En considere la configuration de l'exemple précédant, le code ci-dessous permet la generation d'un signal carré sur la pin RB0.

```
        ORG 0X00
        BCF PORTB,RB0
        BCF TRISB,RB0 ; RB0 en sortie
loop    MOVLW 0x45 ;Configuration du timer 0
        MOVWF T0CON
        CLRF TMR0L

        BSF T0CON, 07 ; Lancement du timer 0
RT      BFSSC INTCON ,T0IF
        BRA RT
        BTG PORTB,RB0
        BCF 07 , T0CON ; arret du timer 0
        BCF T0CON, T0IF ; reset du flag
        BRA loop
```

Remarque 18 *l'instruction **BTG** (bit toggle) permet l'inversion de l'etat d'un bit.*

Question : calculer avec précision la durée de l'impulsion générée ?

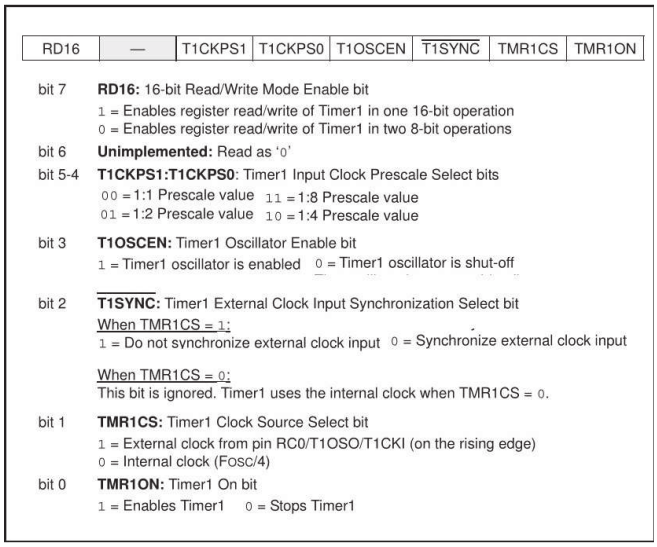
5.2.2 Le Timer 1

Le **Timer 1** est un timer/compteur 16bits qui peut être cadencé avec l'horloge interne (Fosc/4) ou externe.

. Le débordement du timer1 (représenté par les deux registres 8 bits :**TMR1H**, **TMR1L**) c'est-à-dire la transition 0xFFFF-0x0000 peut être configuré pour générer une interruption.

. Quatre valeur du pré-diviseur (pré-scalaire) de fréquence sont possible :1,2,4 , et 8.

. Le registre **TMR1CON** (figure 4) permet le contrôle et la configuration du **Timer 1**.



RD16	—	T1CKPS1	T1CKPS0	T1OSCEN	$\overline{\text{T1SYNC}}$	TMR1CS	TMR1ON

bit 7 **RD16:** 16-bit Read/Write Mode Enable bit
1 = Enables register read/write of Timer1 in one 16-bit operation
0 = Enables register read/write of Timer1 in two 8-bit operations

bit 6 **Unimplemented:** Read as '0'

bit 5-4 **T1CKPS1:T1CKPS0**: Timer1 Input Clock Prescale Select bits
00 = 1:1 Prescale value 11 = 1:8 Prescale value
01 = 1:2 Prescale value 10 = 1:4 Prescale value

bit 3 **T1OSCEN:** Timer1 Oscillator Enable bit
1 = Timer1 oscillator is enabled 0 = Timer1 oscillator is shut-off

bit 2 **$\overline{\text{T1SYNC}}$:** Timer1 External Clock Input Synchronization Select bit
When TMR1CS = 1:
1 = Do not synchronize external clock input 0 = Synchronize external clock input
When TMR1CS = 0:
This bit is ignored. Timer1 uses the internal clock when TMR1CS = 0.

bit 1 **TMR1CS:** Timer1 Clock Source Select bit
1 = External clock from pin RC0/T1OSO/T1CKI (on the rising edge)
0 = Internal clock (FOSC/4)

bit 0 **TMR1ON:** Timer1 On bit
1 = Enables Timer1 0 = Stops Timer1

Figure 04 : Le registre de controle : T1CON

5.2.3 Le Timer 2

Le **Timer 2** est un timer/compteur 8bits, dont le fonctionnement est relativement diffèrent du timer 0 et 1(figure 5). Il est constitué principalement de deux registres 8 bits (**TMR2**), et le registre période **PR2**.

Sa particularité d'avoir deux diviseurs de fréquence :un pré-scalaire, et un post-scalaire.

A chaque top d'horloge **TMR2** est incrémenté, et une fois qu'il atteint la valeur du PR2, il est réinitialisé, et une interruption peut être provoquée.

. Le Timer 2 est aussi contrôlé par le module **CCP** principalement pour générer un signal modulé en largeur d'impulsion MLI (PWM). Cette fonctionnalité sera discutée dans une section suivante.

. LeTimer 2 est configuré et controllé avec le registre **TMR2CON** (figure 6).

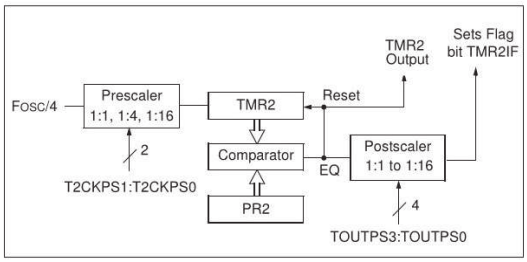


Figure 5 : Schema bloc du Timer 2

U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0
bit 7							bit 0

bit 7 **Unimplemented:** Read as '0'

bit 6-3 **TOUTPS3:TOUTPS0**: Timer2 Output Postscale Select bits
0000 = 1:1 Postscale
0001 = 1:2 Postscale
• • •
1111 = 1:16 Postscale

bit 2 **TMR2ON**: Timer2 On bit
1 = Timer2 is on 0 = Timer2 is off

bit 1-0 **T2CKPS1:T2CKPS0**: Timer2 Clock Prescale Select bits
00 = Prescaler is 1 01 = Prescaler is 4 1x = Prescaler is 16

Figure 6 : Le registre de controle : T2CON

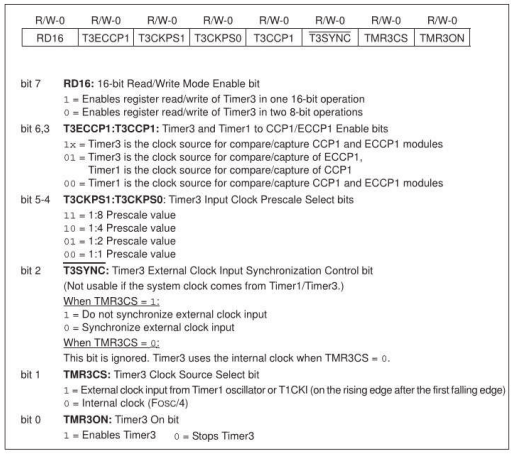
5.2.4 Le Timer 3

Le **Timer 3** est aussi un timer/compteur 16bits, dont le fonctionnement est lié aussi au module **CCP**.

. Le débordement du timer (représenté par les deux registres 8 bits :**TMR3H**, **TMR3L**) c'est-à-dire la transition de 0xFFFF-0x0000 peut être configuré pour générer une interruption.

. Quatre valeur du pré-diviseur (pré-scalaire) de fréquence sont possible :1,2,4 , et 8.

. Le registre **TMR3CON** (figure 7) permet le contrôle et la configuration du **Timer 3**.



R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
RD16	T3ECCP1	T3CKPS1	T3CKPS0	T3CCP1	$\overline{\text{T3SYNC}}$	TMR3CS	TMR3ON

bit 7 **RD16:** 16-bit Read/Write Mode Enable bit
1 = Enables register read/write of Timer3 in one 16-bit operation
0 = Enables register read/write of Timer3 in two 8-bit operations

bit 6,3 **T3ECCP1:T3CCP1:** Timer3 and Timer1 to CCP1/ECCP1 Enable bits
1x = Timer3 is the clock source for compare/capture CCP1 and ECCP1 modules
01 = Timer3 is the clock source for compare/capture of ECCP1,
Timer1 is the clock source for compare/capture of CCP1
00 = Timer1 is the clock source for compare/capture CCP1 and ECCP1 modules

bit 5-4 **T3CKPS1:T3CKPS0**: Timer3 Input Clock Prescale Select bits
11 = 1:8 Prescale value
10 = 1:4 Prescale value
01 = 1:2 Prescale value
00 = 1:1 Prescale value

bit 2 **T3SYNC:** Timer3 External Clock Input Synchronization Control bit
(Not usable if the system clock comes from Timer1/Timer3.)
When TMR3CS = 1:
1 = Do not synchronize external clock input
0 = Synchronize external clock input
When TMR3CS = 0:
This bit is ignored. Timer3 uses the internal clock when TMR3CS = 0.

bit 1 **TMR3CS:** Timer3 Clock Source Select bit
1 = External clock input from Timer1 oscillator or T1CKI (on the rising edge after the first falling edge)
0 = Internal clock (FOSC/4)

bit 0 **TMR3ON:** Timer3 On bit
1 = Enables Timer3 0 = Stops Timer3

Figure 7 : Le registre de controle : T3CON

5.3 Le module CCP (Capture/Compare/Pwm)

Le module **CCP** est basé sur l'utilisation d'un registre 16 pour assurer trois taches différentes : le mode 'Capture', le mode 'Compare', et le mode 'PWM'.

Le registre **CCP1CON** permet une configuration de base du module **CCP** (figure 8).

Le module 'Capture'

Dans ce mode (figure 9) le module **CCP1** est configuré pour fonctionner avec le **timer1** ou le **timer 3**, pour détecter un évènement précis sur la pin **RC2/CCP1**, et copier le contenu de **tmr1** (ou **tmr 3**) vers les registres 8 bits : **CCPR1H-CCPR1L** .

L'évènement détecté sur **RC2/CCP1** peut être :

- un flanc descendant

- un flanc montant

- le 4^{eme} flanc montant

- le 16^{eme} flanc montant

La détection de l'évènement sélectionné active automatiquement le flag **CCP1IF** (du registre **PIR**), et peut aussi produire une interruption. Apres chaque détection ce flag doit être remis à zéro.

En mode Capture **RC2/CCP1** doit être configurée comme d'entrée numérique **TRISC<2>**. Si par erreur la ligne est configurée comme sortie, alors une opération d'écriture peut être considéré comme une validation de la condition du mode capture [1].

—	—	DC1B1	DC1B0	CCP1M3	CCP1M2	CCP1M1	CCP1M0

bit 7-6 **Unimplemented:** Read as '0'

bit 5-4 **DCxB1:DCxB0**: PWM Duty Cycle bit 1 and bit 0

Capture mode: Unused. Compare mode: Unused.

PWM mode:
These bits are the two LSbs (bit 1 and bit 0) of the 10-bit PWM duty cycle. The upper eight bits (DCx9:DCx2) of the duty cycle are found in CCPRxL.

bit 3-0 **CCPxM3:CCPxM0**: CCPx Mode Select bits

0000 = Capture/Compare/PWM off (resets CCPx module)
0001 = Reserved
0010 = Compare mode, toggle output on match (CCPxIF bit is set)
0011 = Capture mode, CAN message received (CCP1 only)
0100 = Capture mode, every falling edge
0101 = Capture mode, every rising edge
0110 = Capture mode, every 4th rising edge
0111 = Capture mode, every 16th rising edge
1000 = Compare mode, initialize CCP pin low, on compare match force CCP pin high (CCPIF bit is set)
1001 = Compare mode, initialize CCP pin high, on compare match force CCP pin low (CCPIF bit is set)
1010 = Compare mode, CCP pin is unaffected (CCPIF bit is set)
1011 = Compare mode, trigger special event (CCP1IF bit is set; CCP resets TMR1 or TMR3 and starts an A/D conversion if the A/D module is enabled)
11xx = PWM mode

Figure 8 : Le registre de controle : CCP1CON

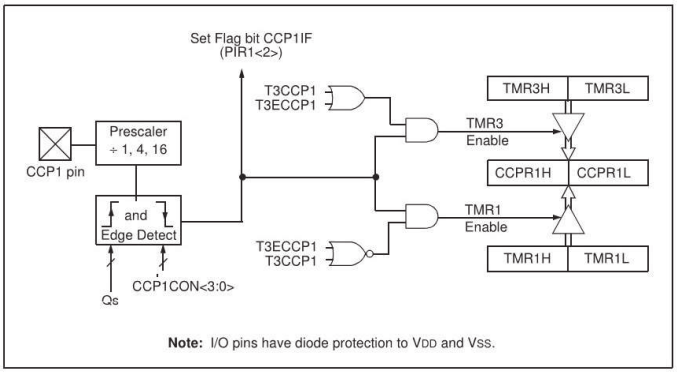


Figure 9 : Mode capture du module CCP

5.3.1 Le module 'Compare'

Dans ce mode (figure 10) les valeurs du registre **CCPR1** et **ECCPR1** sont constamment comparées à ceux du registre **TMR1** (ou **TMR3**). Dans le cas d'égalité la sortie **CCP1** subit l'une des actions suivantes :

- une mise au niveau haut
- une mise au niveau bas
- un basculement d'état (toggle)
- reste inchangé

Le choix de l'action est configuré avec le registre de contrôle **CCP1CON** (figure 8).

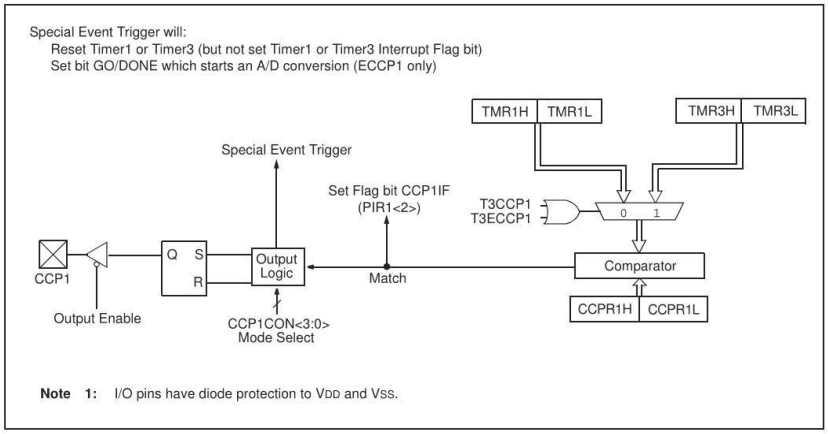


Figure 10 : Mode compare du module CCP

5.3.2 Le module 'PWM'

Le module PWM (Pulse Width Modulation) permet la génération d'un signal modulé en largeur d'impulsion (figure 10.a). On définit le rapport cyclique (duty cycle) comme étant le rapport entre la durée de l'état haut et la période du signal carré généré.

Dans le cas du PIC18F458 la sortie PWM du module **CCP1** est multiplexé avec la sortie **RC2** du **PORTC**.

Pour un fonctionnement en mode PWM, la pin **RC2/CCP1** doit être configurée comme sortie.

Principe de fonctionnement :

Le registre **PR2** fixe la fréquence du signal généré. Cette fréquence est donnée par la formule [1] :

freq=1/period=Fosc/((PR2+1).4.(prescal du tmr2))

Le rapport cyclique est définit par une valeur sur 10 bits (8 bits Msb dans le registre CCP1RL, et 2 bits Lsb dans le registre CCP1CON.

La valeur du rapport cyclique est déterminée par la formule [1] :

$dutycycle = ((CCPR1L : CCP1CON < 5 : 4 >).(prescaldutmr2))/Fosc$

La Configuration de base :

Pour générer une PWM sur **CCP1**, les étapes de base sont :

Fixer la période (fréquence) avec une valeur dans le **PR2**.

Fixer le rapport cyclique avec une valeur 10 bits

Configurer la pin **RC2/CCP1** en sortie.

Fixer la valeur du pré-scalaire, et valider le timer 2 avec le registre **T2CON**

Configurer le module **CCP1** en mode PWM.

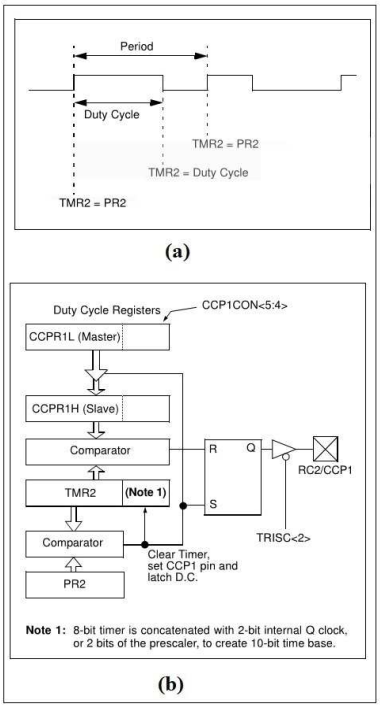


Figure 11 : Mode PWM : (a) Forme d'onde generee, (b) Schema bloc simplifie

5.3.3 Le module ECCP (enhanced CCP)

Le module ECCP intégré dans le pic18F458 est une optimisation des trois fonctions (capture, compare, PWM) du module conventionnel CCP. Cette optimisation apparait en particulier dans le mode PWM.

La figure 12 illustre un schéma bloc simplifie du mode PWM du module ECCP. Quatre (4) sorties PWM sont disponible (avec sélection de polarité). Le module peut être programmé pour un arrêt (shutdown) automatique en réponse à un évènement analogique ou numérique.

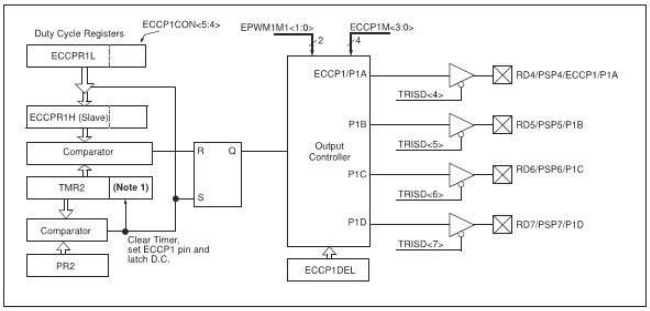


Figure 12 : Schema bloc simplifie du mode PWM du module ECCP

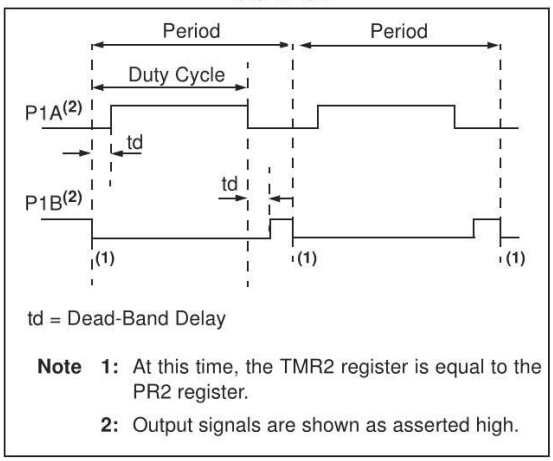


Figure 13 : Sorties PWM en mode demi-pont

Dans ce mode deux pins sont utilisées comme lignes de commande (figure 13). Sur la pin **RD4/P1A** est envoyé le signal PWM, et sur la pin **RD5/P1B** son complément. Ce mode est utilisable dans le cas des applications de commande de switches des circuits en demi-pont, et des circuits en pont-complet (figure 14).

Pour éviter le court-circuit d'un bras du pont un temps de retard (dead band delay) est programmable via le registre **ECCP1DEL**.

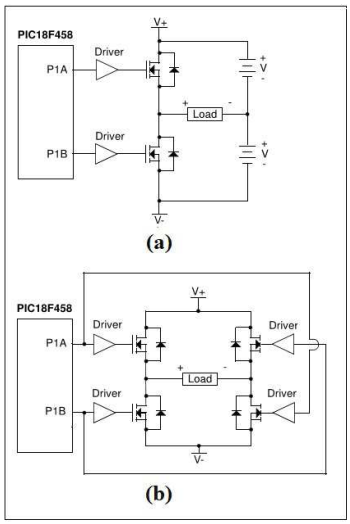


Figure 14 : exemple du mode 'PWM demi-pont' : Circuit demi-pont (push pull) (a), Circuit pont complet (b)

l'utilisation des 4 sorties PWM (PA1, PB1, PC1, PD1) comme lignes de commandes des 4 switches d'un pont complet est evidement possible (figure 15).

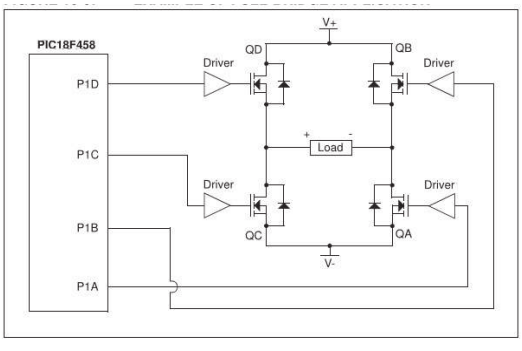


Figure 15 : exemple de commande en mode 'PWM 'pont complet'

5.4 Exercices

Dans les exercices suivants le microcontrôleur est supposé cadencé avec un quartz de 8Mhz.

Exercice 1 :

En exploitant le Timer 0 en mode scrutation, développer un programme permettant la génération d'un signal carré au niveau de la pin RA0 de fréquence de 500Hz.

Exercice 2 :

En exploitant le Timer 0 en mode scrutation, développer un programme permettant l'allumage séquentiel des LEDs connectés aux pins RB0-RB4 (figure 1). Chaque LED doit rester allumé pendant 1s (le reste des LEDs éteintes).

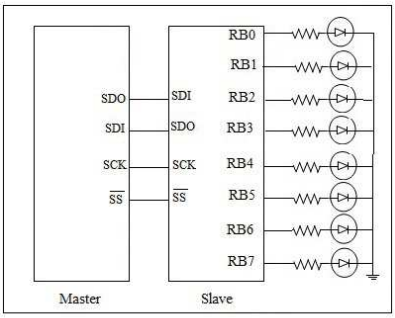


Figure 1

Exercice 3 :

On branche un afficheur 7 segments cathode commune sur le port B suivant la figure 2.

– Développer un programme assembleur permettant l'incrémentation du contenu (de 0 à 9) de l'afficheur chaque seconde.

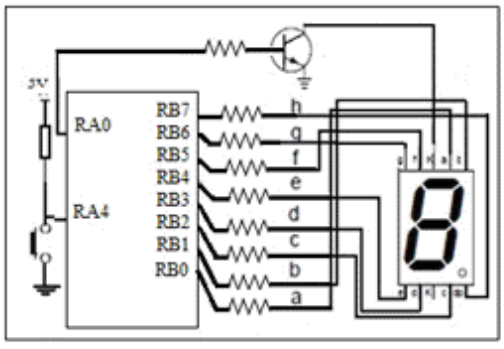


Figure 2

Exercice 4 :

En supposant que le code 7 segments de l'alphabet minuscule (a ..z) est stocké en mémoire à partir de l'adresse 0x20,

donner le programme assebleur permettant un affichage sequentiel et continu de cette alphabet sur le 7 segment de la figure 2.

utiliser le timer 1 pour fixer une durée d'affichage de 2s par caractere.

Exercice 5 :

- En exploitant le PIC18F458 en mode capture, donner l'organigramme et le programme assembleur permettant la mesure de la période d'un signal carré injecté à la broche CCP1, et la sauvegarde du résultat en mémoire.

- Modifier le programme pour la mesure de la durée d'une impulsion injecté à l même pin.

Exercice 6 :

Sur la figure 3 est donné un exemple simple de la commande de la vitesse d'un moteur à courant continu. Le signal modulé en largeur d'impulsion MLI (PWM) appliqué à la grille du MOSFET de puissance permet une variation de la tension qui alimente le moteur.

En exploitant le module PWM du PIC 18F458, donner l'organigramme et le programme assembleur permettant l'augmentation de la vitesse du moteur a chaque appui du bouton poussoir.

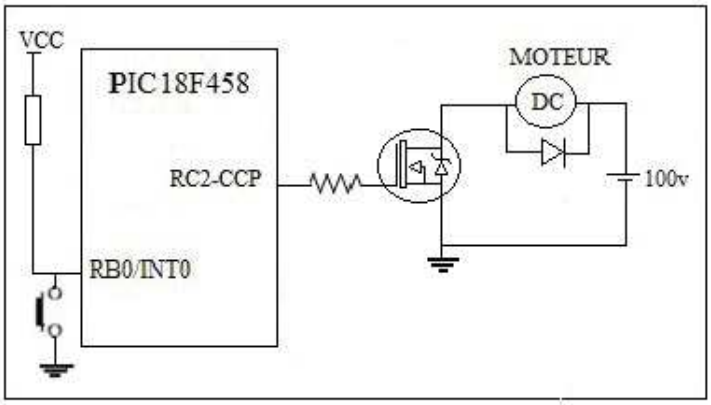


Figure 3

Chapitre 6

Les interruptions

6.1 Introduction

En microinformatique une interruption est un signal (matériel ou logiciel) qui provoque le déroutement du processeur vers une autre tâche [2].

Pratiquement après une interruption un processeur effectue une serie d'opérations :

- La sauvegarde de l'adresse de retours (du programme en cours) dans la pile.

- Le chargement de l'adresse du programme (sous-programme) d'interruption dans le **PC** (compteur programme).

Le retour d'une interruption nécessite en contrepartie l'opération inverse : la récupération de l'adresse de retour (de la pile vers le PC).

On rappelle que la notion de pile (ainsi que le mécanisme d'appel et de retour de sous-programmes) sont déjà introduits, et expliqué dans le chapitre 2.

Dans le cas des PIC18Fxxx, la majorité des modules intégrés ont la possibilité de fonctionner en mode interruption. Les cas considérés dans ce chapitre concerne les ports parallèle et les Timers dont le fonctionnement a été déjà présenté dans les chapitres précédents.

6.2 Les interruption du PIC18F458

Dans le cas du PIC18F458 differentes sources d'interruption existent, et peuvent être organisées en deux niveaux de priorité :

- un niveau haut, dont le vecteur d'interruption est placé à partir de l'adresse 0x000008

- un niveau bas, dont le vecteur d'interruption est placé à partir de l'adresse 0x000018

Evidement Les requêtes des sources de niveau haut peuvent interrompre celles de niveau bas.

La figure 1 illustre la logique de gestion des sources d'interruption, qui nécessite pour une configuration et un contrôle complets 13 registres différents (**RCON**, **INTCON, INTCON2, INTCON3, PIR1, PIR2, PIR3, PIE1, PIE2, PIE3, IPR1, IPR2, IPR3**).

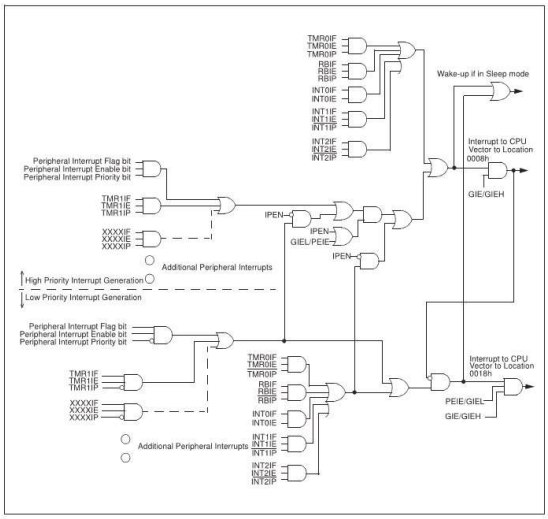


Figure 1 : logique de gestion des interruptions

Remarque 19 *Les détails concernant les registres de gestion d'interruption sont donnés dans le datasheet du constructeur [1].*

On se limitera dans ce document au registres (et bits) intervenant dans les exemples traités.

Pratiquement pour une source d'interruption donnée trois bits de contrôle sont disponible :

- un flag : indiquant l'occurrence de l'interruption.
- un bit : validant l'interruption, et par conséquent le branchement vers l'adresse du vecteur d'interruption.
- un bit : de sélection du niveau de priorité (haute ou basse)

6.2.1 Gestion de la priorité

- Dans le cas ou le bit **IPEN.RCON** est mis à zéro (pas de priorité), alors le bit **GIE.INTCON** deviendra le bit d'activation/désactivation des interruptions, en outre toute requête d'interruption déroutera le processeur vers l'adresse 0x00008.

- Si La priorité est validée (**IPEN.RCON** mis à 1), alors deux bits d'autorisation globale des interruptions sont disponible :

GIEH.INTCON valide toute les interruptions.

GIEL.INTCON valide les interruptions dont le bit de priorité est mis à zéro.

6.2.2 Réponse à une interruption

- En réponse à une requête d'interruption, l'adresse de retour est placée dans la pile. Et les interruptions sont désactivées (sauf si la priorité est validée)
- Le registre PC est chargé soit avec l'adresse 0x000008 ou 0x000018.
- La cause de l'interruption est déterminée par une scrutation des différents flags.
- Une fois déterminé, le flag associé à la requête doit être désactivé pour éviter des interruptions récursives.
-l'instruction **RETFIE** marque la fin de l'interruption, alors l'adresse de retour est récupérée de la pile vers le PC, et le bit d'autorisation des interruptions (**GIEL.INTCON**) est activé une nouvelle fois

6.2.3 Les interruptions externes (INT)

Trois pins du **PORTB** peuvent être programmées pour provoquer une interruption externe : **RB0/INT0, RB1/INT1, RB2/CANTX/INT2**.

La mise à 1 du bit **INTEDGx.INTCON2** (x=0,1,2) selectionne une sensibilité au flanc montant du signal d'entrée, et vice-versa une sensibilité au flanc descendant s'il est mis à zéro.

Pour chaque ligne un bit indicateur est associé **RBx/INTx** (x=0,1,2), ce flag est activé à chaque interruption, et évidement doit être remise à zéro dans le sous-programme d'interruption.

La priorité dans le cas des interruptions **INT1**, **INT2** peut etre selectionnée via les bits INT1IP et INT2IP du registre **INTCON3**. mais dans le cas de **INT0** l'interruption est par défaut de priorité de niveau haut.

6.2.4 Les interruptions des Timers

La fin de la temporisation de l'un des timers peut être programmée pour provoquer une interruption.

Evidement cette fin de temporisation est indiquée par le bit flag correspondant à chaque timer **TxIF** (x=0,1,2,3) (voir le chapitre précédent).

Dans l'exemple suivant un signal carré est généré sur **RB0**, en exploitant le timer 0 en mode interruption.

L'organigramme de la figure 2.a résume la séquence de configuration de base :
- désactivation de la priorité (**IPEN.RCON**=0)
- La validation générale des interruptions (GIE.INTCON=1), et l'autorisation des interruptions périphériques

PEIE.INTCON=1.
- La mise a zéro du flag (**TMR0IF.INTCON**=0)
- La validation de celle du timer 0 (**TMR0IE.INTCON**=1).

- La configuration du timer 0 (**TMROH,TMR0L, T0CON**)

- Lancement du timer 0

Le reste du programme est une boucle infinie à titre d'exemple.

L'organigramme de la figure 2.b résume Le programme d'interruption, dont la tâche est limitée à l'inversion de l'état de **RB0**, mais qui doit impérativement :

-Désactiver le flag T0IF pour éviter une récurrence d'interruption.

-Recharger le registre **TMR0L**, (et **TMR0H** si le mode est 16 bits) pour garder la même durée (si nécessaire).

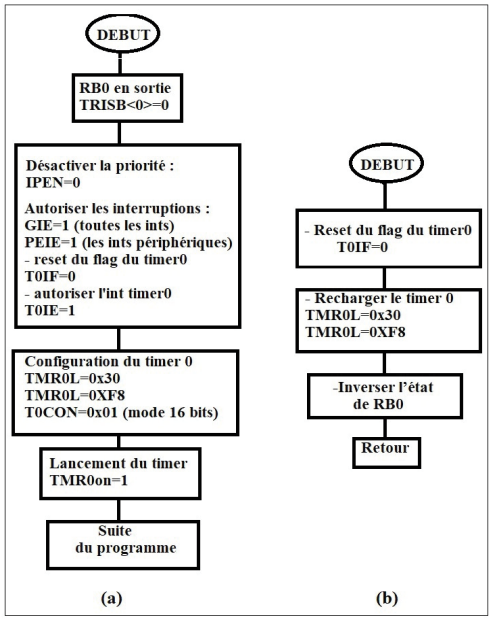


Figure 2 : Exemple d'organigramme pour Timer 0 en mode interruption

Par conséquent le code assembleur peut être le suivant :

```
        ORG 0X00
        BRA prog_P
        ;programme d'interruption
        ORG 0X08
        MOVLW 0x30
        MOVWF TMR0L
        MOVLW 0xF8
        MOVWF TMR0H
        BCF INTCON,T0IF ; reset du flag du timer 0
        BTG PORTB,0
        RETFIE

        ;programme principal
prog_P  BCF TRISB,0 ; RB0 en sortie

        BCF RCON,IPEN ; pas de priorité
        BSF INTCON,GIE ; validation générale des interruptions
        BSF INTCON,PEIE ;validation des ints des périphériques
        BCF INTCON,T0IF ; reset du flag du timer 0.
        MOVLW 0x45 ;Configuration du timer 0
        MOVWF T0CON
        MOVLW 0x30
        MOVWF TMR0L
        MOVLW 0XF8
        MOVWF TMR0H
        BSF INTCON,T0on ; lancement du timer 0
Ret     NOP
        BRA Ret
```

- Dans cet exemple, après la configuration et le lancement du timer 0, la tâche du programme principale est une boucle infinie de **NOP** (no operation).

-Dans le cas général ou plusieurs sources d'interruption sont autorisée, il est important de déterminer au début du sous-programme la cause de l'interruption, ce qui n'est pas le cas dans cet exemple.

- le programme d'interruption débute à partir de l'adresse 0x08 (la priorité des interruptions est désactivée).

6.3 Exercices

En considerant un microcontrôleur cadencé avec un quartz de 8Mhz, refaire les exercices 1 à 6 du chapitre précedant en exploitant le microcontrolleur en mode interruption.

Chapitre 7

La communication série

7.1 Introduction

Ce chapitre est consacré à la description des modules de communication série implémenté dans le microcontrôleur PIC 18F458.

On rappelle que le principe d'une communication série est la transmission/réception des données numériques (bit par bit) sur une seule ligne.

Dans sa forme de base (Figure 1) la sérialisation d'un octet de donnée s'effectue via un registre de décalage (shift register), et à chaque octet envoyé des bits de contrôle sont souvent ajoutés :

. Un bit de start (passage au niveau bas),

. Un bit de stop (retour au niveau haut),

. Un bit de parité pour vérifier la validité de la transmission.

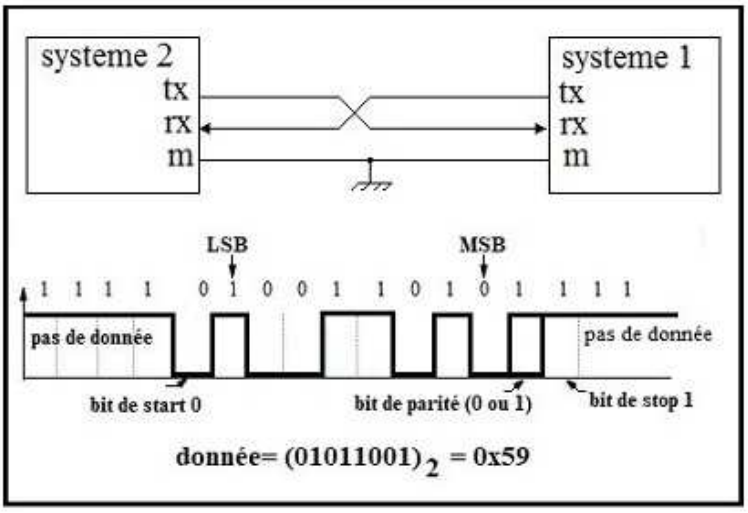


Figure 1 : schema de principe d'une communication serie.

Depuis les premières applications de la communication série différents protocoles et interfaces ont été conçus est appliqués tels que l' USART, le SPI, et le I2C....

En outre pour chaque technique des normes et des standards différents (, RS232, le RS485...) ont aussi été développés pour assurer une compatibilité entre les circuits et les équipements développés.

Dans le cas du PIC 18F458 trois modules différents de liaison série sont implémentés : l'USART, le MSSP, et le bus CAN.

L'objectif de ce chapitre n'est pas de décrire en détail chaque norme, mais de décrire de façon plus ou moins détaillé son implémentation dans le PIC18F458.

On s'intéressera en particulier au module USART, et on décrira en bref les principes des bus SPI et I2C et leurs implémentation.

7.2 Le module USART

Le module USART (Universal Synchronous Asynchronous Receiver Transmitter) connu aussi sous l'appellation SCI (serial communication interface) est l'un des plus anciens protocoles de transmission/réception série.

La figure 1, décrit la configuration minimale d'une communication série USART et la forme d'une trame dans le cas de ce protocole qui peut être constituée d'un bits de start, de 7 ou 8 bits de donnée, d'un bit de parité (optionnel), et d'un ou deux bits de stop.

Les vitesses de transmission sont normalisées : 300, 1200, 4800, 9600 bit/s...

Pour une transmission sur des lignes relativement longues, et pour assurer une certaine immunité au bruit généralement la norme RS232 est utilisée.

Cette norme permet en particulier d'adopter deux niveaux électriques bipolaires (±3v à ±25v), où une logique inverse est utilisé (0 pour une tension positive, 1 pour une tension négative)

On rappelle qu'une norme impose :

- Les caractéristiques mécaniques : type de connecteurs, les dimensions fils et des broches ...

- Les caractéristiques électriques : les niveaux électriques, la charge maximale ...

- La forme de la trame de donnée, et sa vitesse de transmission.

Remarque 20 *Apres le RS232 d'autre standards ont été développés tel que le RS422 ou le RS485*

7.2.1 Le mode asynchrone

La transmission asynchrone :

Le noyau de la transmission asynchrone est le registre a décalage **TSR** (Figure 2).

La donnée à transmettre est reçue par le TSR via le registre **TXREG**. L'écriture dans ce dernier n'est possible qu'après la transmission du bit de stop de la dernière donnée (Figure 3.a).

La mise à 1 du bit flag (**TXIF** du registre **PIR1**) indique que le registre **TXREG** est vide. Cet évènement peut être programmé pour provoquer une interruption (si le bit **TXIE** du registre PIE est activé).

Remarque 21 *Le bit **TRMT** du registre **TXSTA** indique aussi l'état du registre de transmission (mis à 1 quand le **TSR** est vide) sans être lié à l'interruption.*

Les étapes d'une transmission asynchrone :

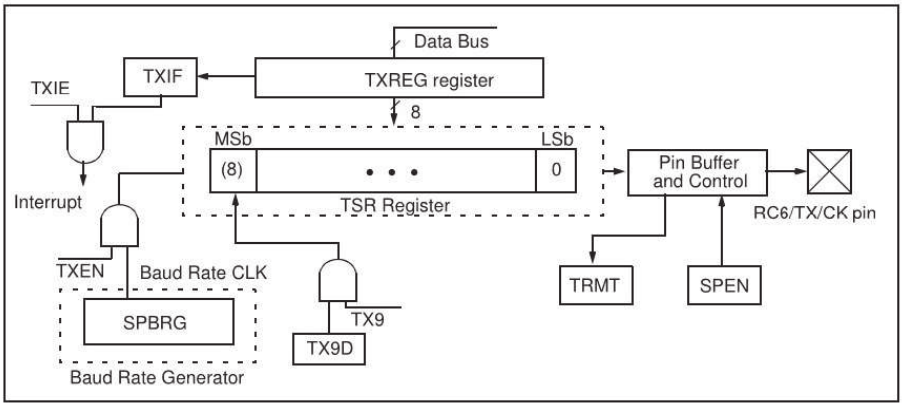


Figure 2 : Schema bloc du module USART (transmission asynchrone)

Les étapes d'une transmission série asynchrone sont décrit brièvement dans cette section, les détails concernant en particulier la sélection de la vitesse de transmission sont données dans le datasheet du constructeur [1].

Le tableau de la figure 3.b résume l'ensemble des registres et bits associés à cette opération de transmission.

Les étapes a effectuées sont :

1. Configuration de la vitesse de transmission via le registre SPBRG et le bit BRGH du registre **TXSTA** (si l'option :'haute vitesse de transmission' est choisie).

2. Validation du port série asynchrone (mise à 0 du bit **SYNC** et mise à 1 du bit **SPEN**).

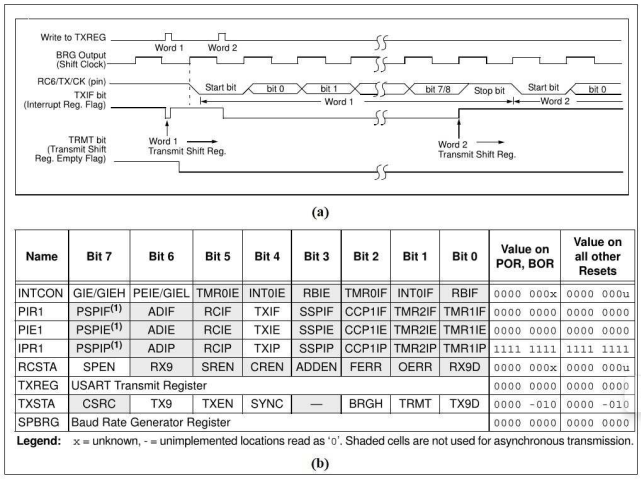
3. Dans le cas de recours à l'interruption le bit **TXIE** doit être activé (mis à 1).

4. Si une transmission de 9 bits est utilisée, le bit **TX9** doit être activé (mis à 1).

5. Validation de la transmission (mise à 1 du bit **TXEN**), cela activera le bit TXIF.

6. Si une transmission 9 bits est utilisée, le 9^{eme} bit doit être chargée dans le bit **TX9D**.

7. Charger la donnée dans le registre **TXREG**, et cela provoquera le lancement de l'opération de transmission.



Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Value on all other Resets
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	0000 000x	0000 000u
PIR1	PSPIF(1)	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
PIE1	PSPIE(1)	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
IPR1	PSPIP(1)	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP	1111 1111	1111 1111
RCSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	0000 000x	0000 000u
TXREG	USART Transmit Register								0000 0000	0000 0000
TXSTA	CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D	0000 -010	0000 -010
SPBRG	Baud Rate Generator Register								0000 0000	0000 0000

Legend: x = unknown, - = unimplemented locations read as '0'. Shaded cells are not used for asynchronous transmission.

(b)

Figure 3 : chronogramme d'une transmission asynchrone (a), et les registres associes (b)

La réception asynchrone

La réception série asynchrone (figure 04) est effectuée via la pin **RC7/RX/DT** est passe initialement par un bloc de décalage à vitesse élevé (16 fois la vitesse de transmission), par contre le registre de décalage principale (de réception) fonctionne a la fréquence d'horloge **FOSC**.

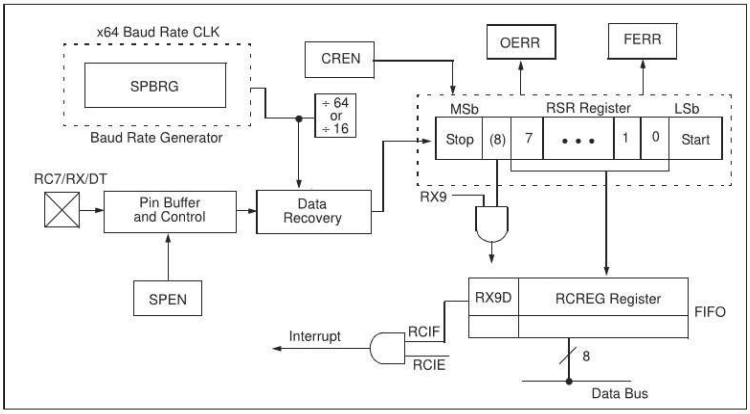


Figure 4 : Schema bloc du module USART (reception asynchrone)

Les étapes d'une réception asynchrone :

Le chronogramme de la figure 5.a résume les étapes d'une réception RS232 asynchrone. Les registres et bits associés à l'opération sont donnés dans le tableau de la figure 5.b.

On rappelle que les détails concernant la sélection de la vitesse de transmission sont données dans le datasheet du constructeur [1].

Les étapes a effectuées sont :

1. Configuration de la vitesse de transmission via le registre **SPBRG** et le bit **BRGH** du registre **TXSTA** (si l'option :'haute vitesse de transmission ' est choisie).

2. Validation du port série asynchrone (la mise à 0 du bit **SYNC** et la mise à 1 du bit **SPEN**).

3. Dans le cas de recours à l'interruption le bit **RCIE** doit être activé (mis à 1).

4. Dans le cas d'une réception 9 bits, le bit **TX9** doit être activé (mis à 1).

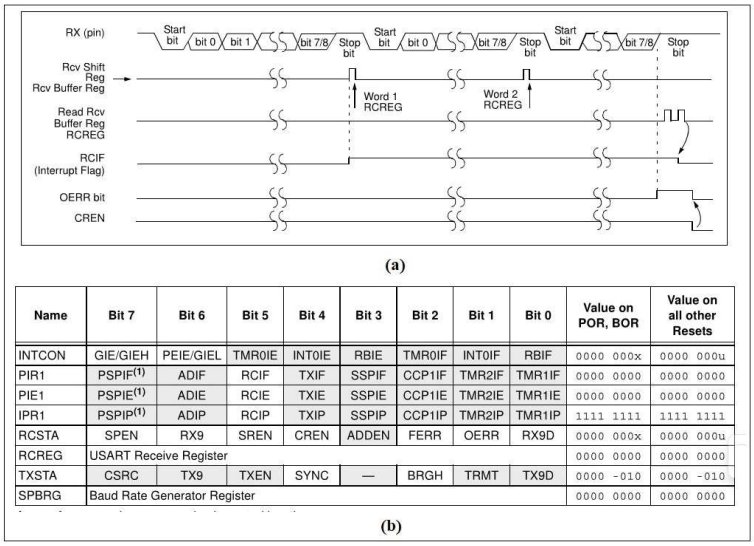
5. Validation de la reception (mise à 1 du bit **CREN**)

6. Le bit flag **RCIF** s'activera (passe à 1) une fois la réception est terminée, et une interruption sera provoquée si le bit RCIE a été activé.

7. la lecture du registre **RCSTA** détermine d'une part s'il y'a eu une erreur de réception, et d'autre part donne le 9^{eme} bit (s'il a été validé).

8. La lecture du registre **RCREG** délivre les 8 bits reçus.

9. En cas d'erreurs de réception le bit **CREN** doit etre mis à 0 pour annuler l'erreur.



(a)

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Value on all other Resets
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	0000 000x	0000 000u
PIR1	PSPIF(1)	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
PIE1	PSPIE(1)	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
IPR1	PSPIP(1)	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP	1111 1111	1111 1111
RCSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	0000 000x	0000 000u
RCREG	USART Receive Register								0000 0000	0000 0000
TXSTA	CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D	0000 -010	0000 -010
SPBRG	Baud Rate Generator Register								0000 0000	0000 0000

(b)

Figure 5 : chronogramme d'une reception asynchrone (a), et les registres associes (b)

7.2.2 Le mode synchrone

Dans le cas synchrone une horloge commune cadence le transfert entre l'émetteur et le récepteur. Le système qui impose le signal d'horloge est dit maitre (master), et le second (qui reçoit ce signal) est dit esclave (slave).

Par conséquent pour un système donné quatre configurations sont possibles :
- réception synchrone (mode master)
- transmission synchrone (mode master)
- réception synchrone (mode slave)
- transmission synchrone (mode slave)

Dans le cas où le système est configure master (synchrone), la communication s'effectue en half-duplex. Cela signifie que la transmission et la réception ne peuvent pas avoir lieu en même temps (quand l'une des deux opérations est activée l'autre est automatiquement inhibée).

Dans le cas du PIC18F458 la pin **RC7/RX/DT/IO** est la broche de transfert de données, et **RC6/TX/CK** est la pin du signal d'horloge.

Les chronogrammes de transmission et de réception synchrone (mode master) sont illustrés respectivement dans les figures 6, et 7.

Remarque 22 *Dans le cas du mode slave (esclave) la seule différence est le fait que signal d'horloge qui cadence le transfert est fourni de l'extérieur (par le*

maitre) via la pin ***RC6/TX/CK****.*

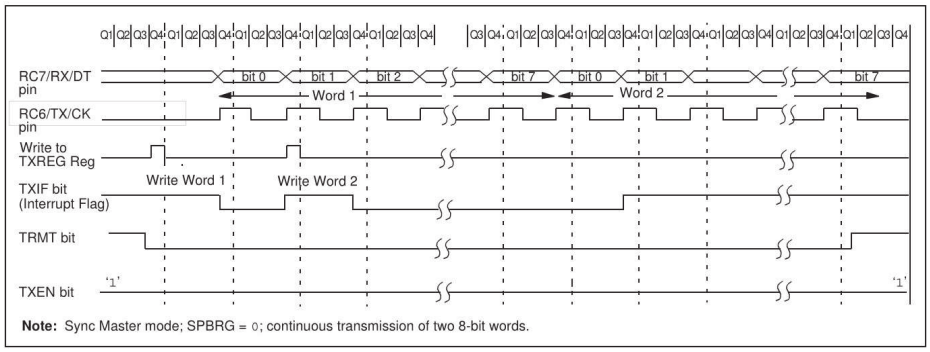


Figure 6 : chronogramme d'une transmission synchrone (mode master) [1]

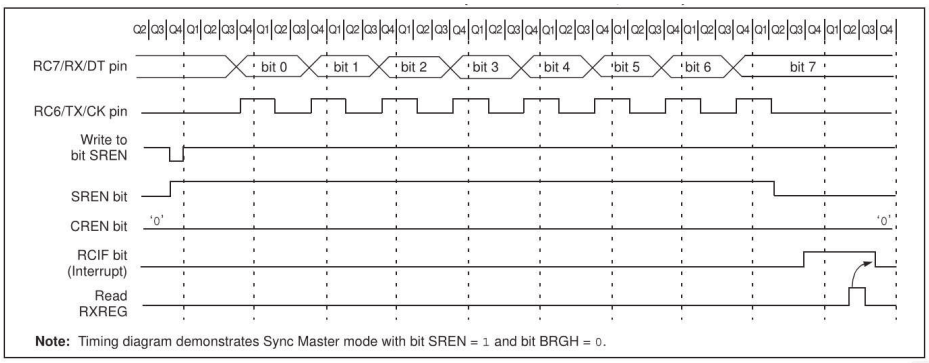


Figure 7 : chronogramme d'une reception synchrone (mode master) [1]

7.2.3 Le module : Master Synchronous Serial Port

Le module **MSSP** est une interface de communication serie utilisée en particulier dans la communication entre microcontroleur est circuits et périphériques tels que les EEPROM, les convertisseurs numérique/analogique DAC, les convertisseurs analogique/numériques ADC, et les afficheurs LCD.

Ce module de communication synchrone présente deux modes différents :

- Le mode SPI (Serial Peripheral Interface)
- Le mode I2C (Inter-Integrated Circuit)

Remarque 23

Trois registres sont utilisés pour le contrôle et la configuration du module MSSP.

Le registre d'état **SSPSTAT**, et deux registres de contrôle **SSPCON1** et **SSPCON2**

7.2.4 Le mode : Serial Peripheral Interface

Le **SPI** est un mode de communication série synchrone de type maitre/esclave (MASTER/SLAVE)[4].

Le circuit maitre génère le signal d'horloge **SCLK** qui cadence le transfert (figure 8), ainsi que les signaux de sélection du boitier esclave SS (slave sélection) concerné par le transfert en cours.

La communication peut s'effectué en mode half-duplex ou full-duplex vu que deux lignes d'entrée/sortie sont utilisées :

MOSI : Master Out Slave In

MISO : Master In Slave Out

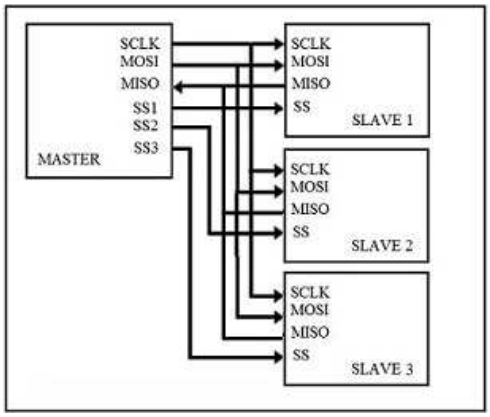


Figure 8 : Exemple d'une liaison SPI (un maitre /3 esclaves)

La figure 9 illustre l'implémentation du mode SPI dans le cas du PIC18F458.

La figure 10 donne l'exemple de la configuration un maitre-un esclave. Les pins impliquées dans le protocole sont : les lignes de données (**SDO**, **SDA**), et l'horloge (**SCK**). Evidement la pin de sélection **SS** est optionnelle vu qu'il n'y a qu'un seul périphérique esclave

Remarque 24 *Diffèrent protocoles de communications SPI sont disponibles dans le PIC18F458 [1], et ne sont pas abordés dans ce document.*

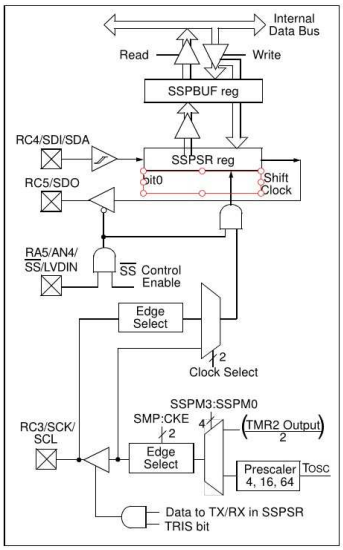


Figure 9 : le module MSSP
(mode SPI)

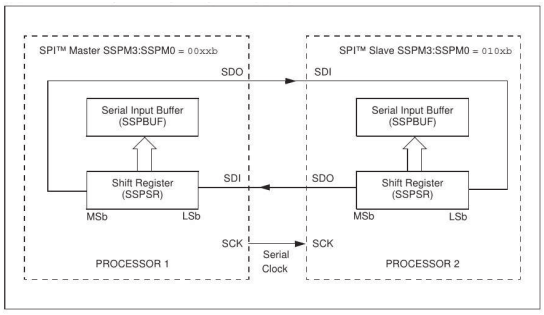


Figure 10 : liaison SPI (un maitre- un esclave)
du PIC18458

7.2.5 Le mode I2C

Le I2C (inter intergrated circuit) est aussi un bus de communication série synchrone de type maitre –esclave (figure 11) avec la particularité d'avoir une seule ligne e donnée et par conséquent fonctionne en half-duplex [1] ,[4].

La selection d'un boitier (esclave) par le maitre est réalisée par le dépos d'une adresse (8 bits) sur le bus, et c'est uniquement le circuit concerné par

cette adresse qui sera activé.

Les lignes de données **SDA** et d'horloge **SCL** sont des lignes drain ouvert d'où la nécessité d'ajouter des résistances de pull-up.

Remarque 25 - *généralement les circuits I2C (esclaves) sont dotés d'un ensembles de pins permettant de fixer une adresse particulière.*

La figure 12 illustre l'implémentation du mode I2C dans le PIC18F458.

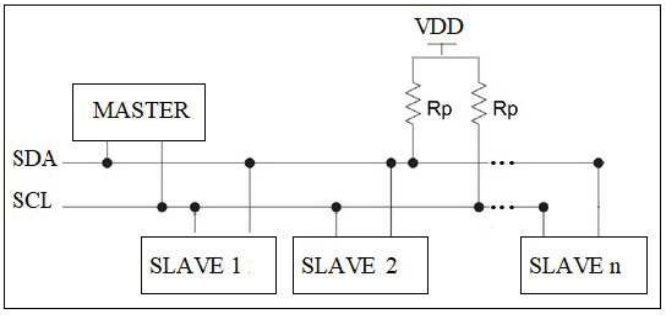


Figure 11 : Exemple d'une liaison I2C

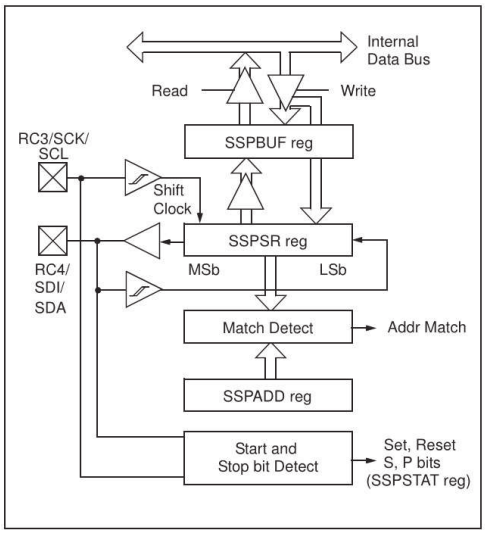


Figure 12 : le module MSSP (mode I2C)

7.3 Exercices

Dans les exercices suivants les microcontrôleurs sont cadencés avec un quartz de 4Mhz

Exercice 1 :

Deux microcontrôleurs (M1 et M2) de type PIC18F458 sont connectés via une liaison USART asynchrone.

- Donner l'organigramme et le programme permettant à M1 de transmettre 100 octets (placés à partir de l'adresse 0x10) vers M2 via cette liaison.

- Donner l'organigramme et le programme permettant à M2 la réception de cette information.

(Format de donnée : 8bit +1bit de stop, vitesse : 512 bit/s)

Exercice 2 :

Refaire l'exercice 1 en mode interruption

Exercice 3 :

Le circuit 25LC256 est une EEPROM série de 256 k octets qui peut être interfacer à un microcontrôleur via une liaison SPI.

En considérant le schéma de la figure 1 :

- Donner la séquence de configuration du microcontrôleur ?

- Donner l'organigramme et le programme permettant la lecture des 10 premiers octets de ce circuit memoire ?

- Vérifier sur le datasheet du circuit 25LC256 la validité du schéma et du programme proposés pour des opérations de lecture et écriture ?

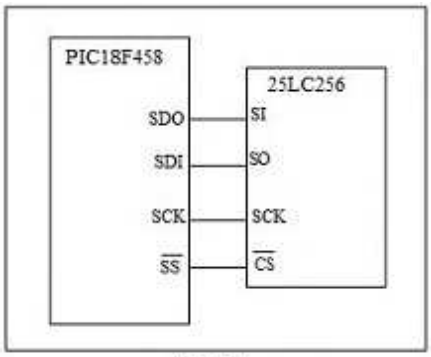


Figure 1

Exercice 4 :

Soit une communication SPI entre deux microcontrôleurs PIC18F458 (figure 2). Ou le master doit envoyer un message de 20 caractères stocké en mémoire (à partir de l'adresse 0x25).

Le second microcontrôleur (slave) doit afficher séquentiellement Ce message sur les LEDs connectées au port B.

- Donner la séquence de configuration de chaque microcontrôleur ?

- Donner l'organigramme et le programme de chaque microcontrôleur ?

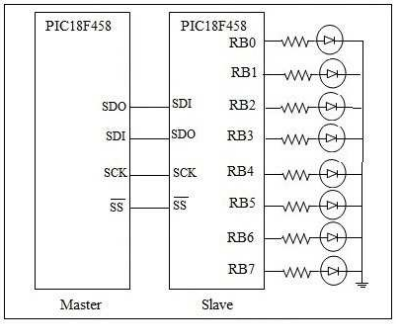


Figure 2

Exercice 5 :

Dans le schéma de la figure 3, le DAC0800 est un convertisseur numérique-analogique 8 bits, dont la sortie est configurée en mode bipolaire.

Le PIC18F458 est connecté à ce convertisseur via une liaison série SPI.

L'adaptation série/parelle est assurée par le registre à décalage 74HCT595 (figure 3). Ce circuit intègre un registre PIPO (parallel in parallel out) entre le registre à décalage est la sortie analogique.

La donnée série est envoyer vers le registre parelle après un flanc montant sur la pin RCK.

- En utilisant le datasheat du constructeur, donner et justifier la séquence de configuration du microcontrôleur ?

- Donner l'organigramme et le programme assembleur permettant la génération d'un signal carré ±5v de frequence donnée.

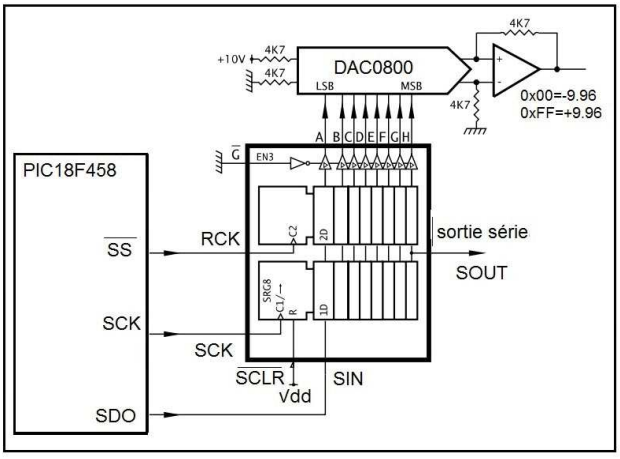


Figure 3

Exercice 6 :

Dans le schéma de la figure 4, le circuit 24LC64 est une EEPROM série de 64 ko connectée au PIC18F458 via une liaison série I2C.

- En utilisant les datasheats du constructeur, donner et justifier la séquence de configuration du microcontrôleur ?

- Donner l'organigramme et le programme assembleur permettant la mise à zéro du premier ko de cette EEPROM

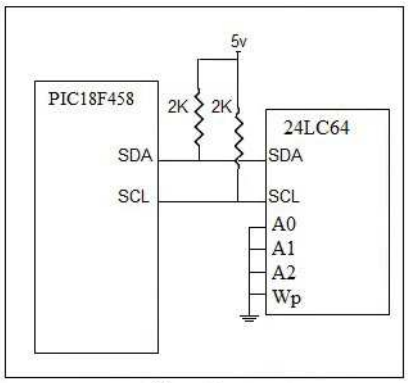


Figure 4

Chapitre 8

La conversion analogique/numerique

8.1 Introduction

Ce chapitre est consacré à la conversion analogique numérique, le principe de base est abordé brièvement, étant donné que la finalité est de décrire le module ADC du PIC18F458.

8.2 Le principe de base de la conversion

Un convertisseur analogique numérique CAN (analogue to digital converter ADC) est un circuit qui donne une correspondance linaire entre un signal analogique et une valeur numérique codée sur un nombre n de bits. Dans la réalité cette correspondance peut présenter un effet de non-linéarité (Figure 1).

Dans le cas d'une configuration unipolaire le signal analogique est borné par une certaine tension de référence ($0 \leq Vin \leq Vref$). Cette valeur de référence correspond à la valeur maximale possible sur N bits ($0 < N < 2^n - 1$). Par consequent la valeur numérique delivrée par un ADC pour une une entrée analogique Vin correspond à l'entier donnée par :

$val_ADC = (2^n - 1).Vin/Vref$

Ainsi la résolution d'un convertisseur (la valeur analogique minimale détectable) est liée d'une part au nombre de bits du convertisseur et la tension de référence :

1 LSB=Vref/ $(2^n - 1)$.

Remarque 26 *On note que l'ADC réalise une fonction définie pour l'ensemble d'entiers {0,.. , 2^n -1}*

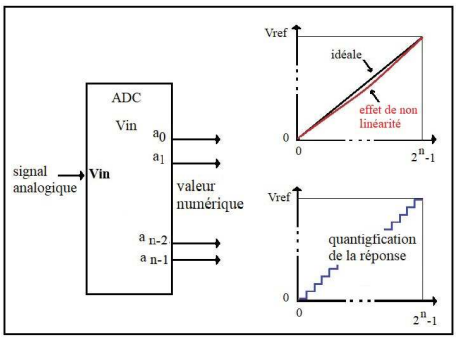


Figure 1 : schema bloc d'un ADC et sa caracteristique ideale, reelle, et quantifiee

Exemple 27

1. Un convertisseur 8bit de tension de référence 5v :
. Sa sortie numérique couvre l'ensemble d'entier : $[00\ FF]_{16} = [00\ 255]_{10}$
. Sa résolution : 5/255=19.607mv
. 2.5v correspond à : 255.2.5/ 5=127.5, donc le convertisseur délivrera l'entier 127 ou 128.

2. Un convertisseur 10bit de tension de référence 5v :
- Sa sortie numérique couvre l'ensemble d'entier : $[00\ 3F]_{16} = [00\ 1023]_{10}$
- Sa résolution : 5/1023=4.887mv
-2.5v correspond à : 1023.2.5/ 5=511.5, donc le convertisseur délivrera l'entier 511 ou 512.

Remarque 28 *Certains convertisseur permettent une configuration bipolaire ou l'entrée analogique est limitée entre deux seuils opposés* $(-Vref \leq Vin \leq Vref)$

8.3 L' ADC du PIC18F458

Le PIC 18F458 compte un seul convertisseur analogique numérique (10 bits), avec 8 entrées analogiques multiplexées :
- 5 entrées sur le PORTA (**AN0, AN1, AN2, AN3, AN4**)
- 3 entrées sur le PORTE (**AN5, AN6, AN7**)
Les tensions analogiques de références peuvent être internes (VDD, et VSS du microcontrôleur), ou externes via les pins : **RA2/AN2/VREF-**, et **RA3/AN3/VREF+** (Figure 02).

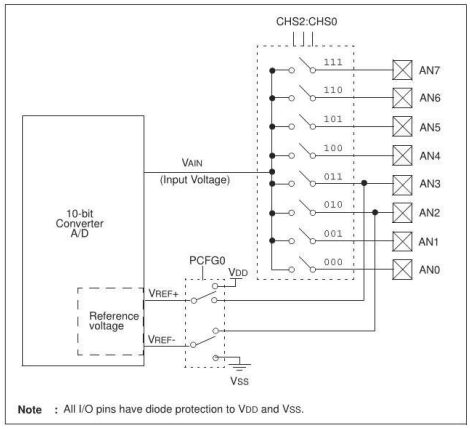


Figure 2 : schema bloc de l'ADC du PIC18F458

Le fonctionnement et les registres associés

Les dix bits du résultat de la conversion sont affectés à deux registres 8 bits (**ADRESH**, **ADRESL**), et par conséquent le résultat est soit justifié à droite (right justified) ou à gauche (left justified) (Figure 03).

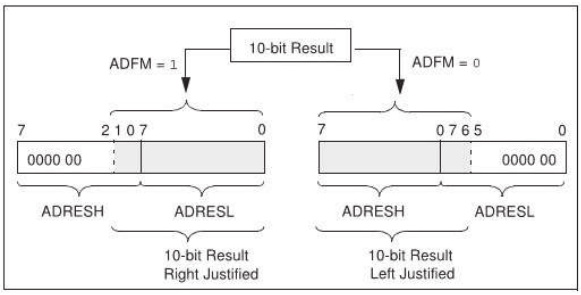


Figure 3 : Justification du resultat de la conversion

Les deux registres **ADCON0**, et **ADCON1** (fig 4) permettent la configuration el le contrôle du module ADC.

La configuration des pins des ports (A, E) est assurée par **ADCON1**, ainsi que le choix du mode de justification du résultat.

L'ensemble de bits du registre **ADCON0** permettent le contrôle de l'opération de conversion :

- **Adon** est le bit d'activation du module ADC.

- Les bits **CHS0-2** sélectionnent l'entrée analogique à convertir (Figure 02).
- Le bit **GO/($\overline{\text{DONE}}$)**permet d'une part le lancement de la conversion, et la détection de la fin de l'opération.

Remarque 29 *A la fin d'une opération de conversion le flag (d'interruption)* ***ADIF*** *est activé.*

8.3.1 Les étapes d'une conversion

Le constructeur impose une série d'étapes à respecter pour effectuer correctement une opération de conversion [1].

Ces étapes sont :

Configuration du module ADC

. Configuration des entrées analogiques et des références (**ADCON1**).
. Sélection de l'entrée analogique (**ADCON0**).
. Sélection de l'horloge du module ADC.
. Valider le module ADC (**ADCON0**).

Validation de l'interruption (si le mode interruption est utilisé)

. Mise à zéro du flag **ADIF**
. Mise à un (1) des bits **ADIE**, et **GIE**.

L'attente du temps d'acquisition nécessaire

Lancer la conversion : mise à un (1) du bit **GO/($\overline{\text{DONE}}$)**

L'attente de la fin de la conversion : soit par une scrutation du bit **GO/($\overline{\text{DONE}}$)** (sa remise à zéro), soit apres une interruption Due au module ADC.

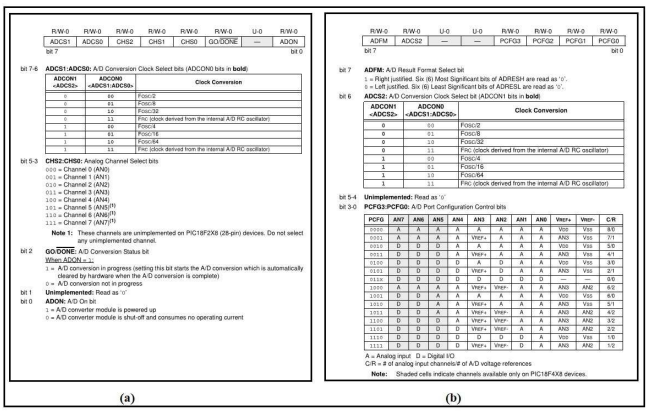


Figure 4 : Les registres de controle de l'ADC : (a) ADCON0, (b)ADCON1 [1]

8.4 Exercices

Dans les exercices suivants les microcontrôleurs sont cadencés avec un quartz de 4Mhz

Exercice 1 :

Soit le schéma de la figure 1, ou le résultat de la conversion analogique numérique est affiché sur les Leds du PORTB.

Donner un programme assembleur permettant :

- La configuration de l'ADC en mode 8 bits
- L'affichage séquentiel du résultat sur les LEDs du PORTB
- Proposer un sous-programme (à base du TIMER1) permettant une période d'échantillonnage d'une seconde.

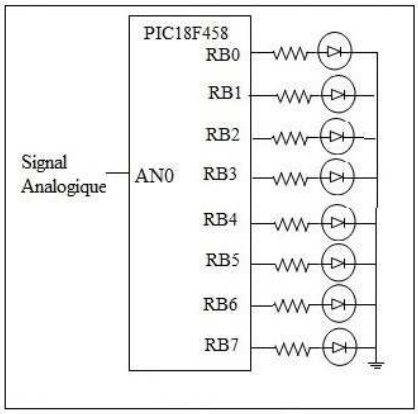


Figure 1

Exercice 2 :

Soit le montage de la figure 2, ou Le LM35 est un capteur de température dont la sensibilité est de 10mv/ degré Celsius (250mV à 25°).

Donner un programme assembleur permettant :

- La configuration de l'ADC en mode 8 bits
- L'affichage séquentiel de la température sur l'afficheur LCD

(Utiliser la même temporisation de l'exercice précédent)

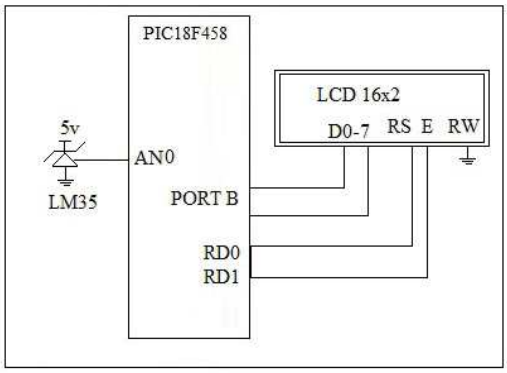


Figure 2

Exercice 3 :
Refaire l'exercice 2 en exploitant l'ADC en mode 10 bits.

Chapitre 9

Annexe

9.1 Annexe 1 :Rappel géneral

Dans cette annexe est donné un rappel bref sur les notions de base relatives à la représentation numérique, ainsi que la terminologie couramment utilisée en micro-informatique [2].

9.2 La représentation numérique

Le traitement d'un signal donné par un système micro-informatique nécessite impérativement sa quantification numérique.

En considérant une représentation pondérée (base b), un nombre N peut être écrit sous la forme :

$$N = \sum_{i=0}^{n-1} a_i b^i = a_0 + a_1 b + a_2 b^2 + ... + a_{n-1} b^{n-1}, \quad N[a_0 \quad a_1 ... a_{n-1}]$$

La base 2 permet l'utilisation de deux nombres {0,1}, qui peuvent etre implémentés via deux niveaux Electriques (exemple 0 et 5v).

Pour une compression et une meilleure lisibilité de l'information numérique, La base 16 est souvent utilisée. Dans ce cas l'ensemble de la représentation est : $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F\}$.

9.2.1 La représentation des entiers

Les entiers positifs

Une représentation sur n bits (base 2) couvre un intervalle de nombre N donnée par :

$0 \leq N \leq b^n - 1$

sur 8 bits :

$0 \leq N \leq b^8 - 1 = (255)_{10} = (FF)_{16}$

$(a_0\ a_1\ a_2\ a_3\ a_4\ a_5\ a_6\ a_7)_2$	$(h_0\ h_2)_{16}$	$(N)_{10}$
00000000	00	00
00000001	01	01
00000010	02	02
...	...	...
10000000	80	128
10000001	81	129
...	...	...
11111111	FF	255

sur 16 bits :
$0 \leq N \leq b^{16} - 1 = (65535)_{10} = (FFFF)_{16}$
Dans la représentation numérique, généralement la terminologie anglo-saxonne est la plus couramment utilisée :
- **Bit** : est la compression de l'expression "binary digit".
-**MSB** (Most Significant Bit) : est le bit de poids fort d'une séquence binaire donnée
- **LSB** (Least Significant Bit) : est le bit de poids faible de la séquence.
- **Byte** (ou octet en français) représente 8 bits.
- **Word** (ou mot en français) représente 16 bits.

Les entiers signés

La représentation module-signe

Cette représentation consiste à utiliser le MSB d'un nombre donné comme signe, et le reste des bits comme module.

a_{n-1}	a_{n-2}	...	a_0
signe		module	

Un MSB= 1 implique que le nombre est négatif,
Sinon (MSB= 0) le nombre est positif
L'inconvénient majeur du codage module-signe est le fait que le zéro possède deux représentations différentes (un zéro positif est un autre négatif).
sur 8 bits :
$(FF)_{16} = (-127)_{10} = (11111111)_2 \leq N \leq (01111111)_2 = (127)_{10} = (7F)_{16}$

La représentation en complément à 2

Dans ce cas un entier N est représenté avec une pondération base 2, ou le MSB possède une contribution négative :

$$N = \sum_{i=0}^{n-1} a_i b^i = -a_{n-1}b^{n-1} + a_{n-2}b^{n-2} + ...a_0 = -a_{n-1}b^{n-1} + \sum_{i=0}^{n-2} a_i b^i$$

Si dans ce cas le zéro possède une seule représentation, le nombre le plus négatif n'a pas de complément.

sur 8 bits :

$(80)_{16} = (-128)_{10} = (10000000)_2 \leq N \leq (01111111)_2 = (127)_{10} = (7F)_{16}$

$(a_0\ a_1\ a_2\ a_3\ a_4\ a_5\ a_6\ a_7)_2$	$(h_0\ h_2)_{16}$	$(N)_{10}$
00000000	00	00
00000001	01	01
00000010	02	02
...	...	...
01111111	7F	127
10000000	80	-128
10000001	81	-127
11111111	FF	-1

Sur 16 bits :

$(8000)_{16} = (-32768)_{10} = (10000000)_2 \leq N \leq (0111...1111)_2 = (32767)_{10} = (7FFF)_{16}$

- De même que la représentation module-signe, le MSB indique le signe :

si le MSB=1 l'entier est négatif,

sinon (MSB=0) l'entier est positif

9.2.2 La représentation des réels

Dans ce cas deux représentations sont utilisées avec différentes variantes liées en particulier à la taille des intervalles couverts et à la précision désirée.

La représentation en virgule fixe

Dans ce cas le nombre réel est représenté avec deux entités ; ou la première indique sa partie entière, et la seconde sa partie fractionnelle.

a_{n-1}	a_{n-2}	...	a_0	,	b_{n-1}	b_{n-2}	...	b_0
	partie	entiere				partrie	fractionnelle	

La partie entière peut être donnée en complément à 2 sur n bits $(-a_{n-1}b^{n-1} + \sum_{i=0}^{n-2} a_i b^i)$, et la partie fractionnelle sur m bits $(\sum_{i=0}^{m-1} a_i b^i)$

La représentation en virgule flottante

Dans ce cas un nombre réel N s'écrit sous la forme :

$N = \pm M.b^E$

Où :

M est la mantisse sur n bits

b : la base

E : l'exposant

Afin d'assurer une portabilité des produits, une normalisation du codage etait nécessaire. La première norme (IEEE754) date de 1985. Puis d'autres révisions de cette norme ont été proposées (la dernière est celle de 2008).

La version d'origine de la norme IEEE 754 définit quatre formats pour représenter les nombres en virgule flottante en base 2 :

Nom	**Taille**	**Signe**	**Exposant**	**Mantisse**
Simple précision	32 bits	1 bits	8 bits (de -126 à +127)	24 bits
Double précision	64 bits	1 bits	11 bits (de -1022 à +1023)	53 bits
Double précision étendue	79 bits	1 bits	15 bits (de -16382 à +16383)	64 bits

Remarque 30 *Le format double précision étendue est mis en oeuvre sur 80 bits*

9.2.3 Le code ASCII

Dans le cas général, une séquence numérique n'est pas toujours un nombre, et peut être une information non-numérique. Le code ASCII (American Standard Code for Information Interchange) proposé depuis le début des années soixante est pratiquement la norme de codage de l'information non-numérique.

Ce code a été étendu à plusieurs reprises pour comprendre outre les lettres, les symboles, et les différentes commandes.

"A" = (41)16 = (65)10 ; "B" = (42)16 = (66)10

"a" = (61)16 = (97)10 ; "b" = (62)16 = (98)10

"1" = (31)16 = (49)10 ; "2" = (32)16 = (50)10

"stx" = (02)16 = (02)10 (start of text)

"etx" = (03)16 = (03)10 (end of text)

"ack" = (06)16 = (02)10 (Acknowledge accuse de reception)

9.3 Annexe 2 :Les registres SFR (Special Function Register)

Address	Name	Address	Name	Address	Name	Address	Name
FFFh	TOSU	FDFh	INDF2(2)	FBFh	CCPR1H	F9Fh	IPR1
FFEh	TOSH	FDEh	POSTINC2(2)	FBEh	CCPR1L	F9Eh	PIR1
FFDh	TOSL	FDDh	POSTDEC2(2)	FBDh	CCP1CON	F9Dh	PIE1
FFCh	STKPTR	FDCh	PREINC2(2)	FBCh	ECCPR1H(5)	F9Ch	—
FFBh	PCLATU	FDBh	PLUSW2(2)	FBBh	ECCPR1L(5)	F9Bh	—
FFAh	PCLATH	FDAh	FSR2H	FBAh	ECCP1CON(5)	F9Ah	—
FF9h	PCL	FD9h	FSR2L	FB9h	—	F99h	—
FF8h	TBLPTRU	FD8h	STATUS	FB8h	—	F98h	—
FF7h	TBLPTRH	FD7h	TMR0H	FB7h	ECCP1DEL(5)	F97h	—
FF6h	TBLPTRL	FD6h	TMR0L	FB6h	ECCPAS(5)	F96h	TRISE(5)
FF5h	TABLAT	FD5h	T0CON	FB5h	CVRCON(5)	F95h	TRISD(5)
FF4h	PRODH	FD4h	—	FB4h	CMCON(5)	F94h	TRISC
FF3h	PRODL	FD3h	OSCCON	FB3h	TMR3H	F93h	TRISB
FF2h	INTCON	FD2h	LVDCON	FB2h	TMR3L	F92h	TRISA
FF1h	INTCON2	FD1h	WDTCON	FB1h	T3CON	F91h	—
FF0h	INTCON3	FD0h	RCON	FB0h	—	F90h	—
FEFh	INDF0(2)	FCFh	TMR1H	FAFh	SPBRG	F8Fh	—
FEEh	POSTINC0(2)	FCEh	TMR1L	FAEh	RCREG	F8Eh	—
FEDh	POSTDEC0(2)	FCDh	T1CON	FADh	TXREG	F8Dh	LATE(5)
FECh	PREINC0(2)	FCCh	TMR2	FACh	TXSTA	F8Ch	LATD(5)
FEBh	PLUSW0(2)	FCBh	PR2	FABh	RCSTA	F8Bh	LATC
FEAh	FSR0H	FCAh	T2CON	FAAh	—	F8Ah	LATB
FE9h	FSR0L	FC9h	SSPBUF	FA9h	EEADR	F89h	LATA
FE8h	WREG	FC8h	SSPADD	FA8h	EEDATA	F88h	—
FE7h	INDF1(2)	FC7h	SSPSTAT	FA7h	EECON2	F87h	—
FE6h	POSTINC1(2)	FC6h	SSPCON1	FA6h	EECON1	F86h	—
FE5h	POSTDEC1(2)	FC5h	SSPCON2	FA5h	IPR3	F85h	—
FE4h	PREINC1(2)	FC4h	ADRESH	FA4h	PIR3	F84h	PORTE(5)
FE3h	PLUSW1(2)	FC3h	ADRESL	FA3h	PIE3	F83h	PORTD(5)
FE2h	FSR1H	FC2h	ADCON0	FA2h	IPR2	F82h	PORTC
FE1h	FSR1L	FC1h	ADCON1	FA1h	PIR2	F81h	PORTB
FE0h	BSR	FC0h	—	FA0h	PIE2	F80h	PORTA

Note 1: Unimplemented registers are read as '0'.

2: This is not a physical register.

3: Contents of register are dependent on WIN2:WIN0 bits in the CANCON register.

4: CANSTAT register is repeated in these locations to simplify application firmware. Unique names are given for each instance of the CANSTAT register due to the Microchip header file requirement.

5: These registers are not implemented on the PIC18F248 and PIC18F258.

9.4 Annexe 3 :Le jeu d'instructions des PIC18F

Le jeu d'instructions des PIC18F est donné dans les trois tableaux suivants :

Mnemonic, Operands		Description	Cycles	16-Bit Instruction Word MSb	LSb	Status Affected	Notes
BYTE-ORIENTED FILE REGISTER OPERATIONS							
ADDWF	f, d, a	Add WREG and f	1	0010 01da	ffff ffff	C, DC, Z, OV, N	1, 2
ADDWFC	f, d, a	Add WREG and Carry bit to f	1	0010 00da	ffff ffff	C, DC, Z, OV, N	1, 2
ANDWF	f, d, a	AND WREG with f	1	0001 01da	ffff ffff	Z, N	1,2
CLRF	f, a	Clear f	1	0110 101a	ffff ffff	Z	2
COMF	f, d, a	Complement f	1	0001 11da	ffff ffff	Z, N	1, 2
CPFSEQ	f, a	Compare f with WREG, skip =	1 (2 or 3)	0110 001a	ffff ffff	None	4
CPFSGT	f, a	Compare f with WREG, skip >	1 (2 or 3)	0110 010a	ffff ffff	None	4
CPFSLT	f, a	Compare f with WREG, skip <	1 (2 or 3)	0110 000a	ffff ffff	None	1, 2
DECF	f, d, a	Decrement f	1	0000 01da	ffff ffff	C, DC, Z, OV, N	1, 2, 3, 4
DECFSZ	f, d, a	Decrement f, Skip if 0	1 (2 or 3)	0010 11da	ffff ffff	None	1, 2, 3, 4
DCFSNZ	f, d, a	Decrement f, Skip if Not 0	1 (2 or 3)	0100 11da	ffff ffff	None	1, 2
INCF	f, d, a	Increment f	1	0010 10da	ffff ffff	C, DC, Z, OV, N	1, 2, 3, 4
INCFSZ	f, d, a	Increment f, Skip if 0	1 (2 or 3)	0011 11da	ffff ffff	None	4
INFSNZ	f, d, a	Increment f, Skip if Not 0	1 (2 or 3)	0100 10da	ffff ffff	None	1, 2
IORWF	f, d, a	Inclusive OR WREG with f	1	0001 00da	ffff ffff	Z, N	1, 2
MOVF	f, d, a	Move f	1	0101 00da	ffff ffff	Z, N	1
MOVFF	f_s, f_d	Move f_s (source) to 1st word f_d (destination) 2nd word	2	1100 ffff 1111 ffff	ffff ffff ffff ffff	None	
MOVWF	f, a	Move WREG to f	1	0110 111a	ffff ffff	None	
MULWF	f, a	Multiply WREG with f	1	0000 001a	ffff ffff	None	
NEGF	f, a	Negate f	1	0110 110a	ffff ffff	C, DC, Z, OV, N	1, 2
RLCF	f, d, a	Rotate Left f through Carry	1	0011 01da	ffff ffff	C, Z, N	
RLNCF	f, d, a	Rotate Left f (No Carry)	1	0100 01da	ffff ffff	Z, N	1, 2
RRCF	f, d, a	Rotate Right f through Carry	1	0011 00da	ffff ffff	C, Z, N	
RRNCF	f, d, a	Rotate Right f (No Carry)	1	0100 00da	ffff ffff	Z, N	
SETF	f, a	Set f	1	0110 100a	ffff ffff	None	
SUBFWB	f, d, a	Subtract f from WREG with borrow	1	0101 01da	ffff ffff	C, DC, Z, OV, N	1, 2
SUBWF	f, d, a	Subtract WREG from f	1	0101 11da	ffff ffff	C, DC, Z, OV, N	
SUBWFB	f, d, a	Subtract WREG from f with borrow	1	0101 10da	ffff ffff	C, DC, Z, OV, N	1, 2
SWAPF	f, d, a	Swap nibbles in f	1	0011 10da	ffff ffff	None	4
TSTFSZ	f, a	Test f, skip if 0	1 (2 or 3)	0110 011a	ffff ffff	None	1, 2
XORWF	f, d, a	Exclusive OR WREG with f	1	0001 10da	ffff ffff	Z, N	
BIT-ORIENTED FILE REGISTER OPERATIONS							
BCF	f, b, a	Bit Clear f	1	1001 bbba	ffff ffff	None	1, 2
BSF	f, b, a	Bit Set f	1	1000 bbba	ffff ffff	None	1, 2
BTFSC	f, b, a	Bit Test f, Skip if Clear	1 (2 or 3)	1011 bbba	ffff ffff	None	3, 4
BTFSS	f, b, a	Bit Test f, Skip if Set	1 (2 or 3)	1010 bbba	ffff ffff	None	3, 4
BTG	f, d, a	Bit Toggle f	1	0111 bbba	ffff ffff	None	1, 2

Note 1: When a Port register is modified as a function of itself (e.g., `MOVF PORTB, 1, 0`), the value used will be that value present on the pins themselves. For example, if the data latch is '1' for a pin configured as input and is driven low by an external device, the data will be written back with a '0'.

2: If this instruction is executed on the TMR0 register (and where applicable, d = 1), the prescaler will be cleared if assigned.

3: If Program Counter (PC) is modified or a conditional test is true, the instruction requires two cycles. The second cycle is executed as a NOP.

4: Some instructions are 2-word instructions. The second word of these instructions will be executed as a NOP unless the first word of the instruction retrieves the information embedded in these 16 bits. This ensures that all program memory locations have a valid instruction.

5: If the table write starts the write cycle to internal memory, the write will continue until terminated.

Mnemonic, Operands		Description	Cycles	16-Bit Instruction Word MSb LSb	Status Affected	Notes
CONTROL OPERATIONS						
BC	n	Branch if Carry	1 (2)	1110 0010 nnnn nnnn	None	
BN	n	Branch if Negative	1 (2)	1110 0110 nnnn nnnn	None	
BNC	n	Branch if Not Carry	1 (2)	1110 0011 nnnn nnnn	None	
BNN	n	Branch if Not Negative	1 (2)	1110 0111 nnnn nnnn	None	
BNOV	n	Branch if Not Overflow	1 (2)	1110 0101 nnnn nnnn	None	
BNZ	n	Branch if Not Zero	2	1110 0001 nnnn nnnn	None	
BOV	n	Branch if Overflow	1 (2)	1110 0100 nnnn nnnn	None	
BRA	n	Branch Unconditionally	1 (2)	1101 0nnn nnnn nnnn	None	
BZ	n	Branch if Zero	1 (2)	1110 0000 nnnn nnnn	None	
CALL	n, s	Call subroutine 1st word	2	1110 110s kkkk kkkk	None	
		2nd word		1111 kkkk kkkk kkkk		
CLRWDT	—	Clear Watchdog Timer	1	0000 0000 0000 0100	$\overline{TO}$, $\overline{PD}$	
DAW	—	Decimal Adjust WREG	1	0000 0000 0000 0111	C	
GOTO	n	Go to address 1st word	2	1110 1111 kkkk kkkk	None	
		2nd word		1111 kkkk kkkk kkkk		
NOP	—	No Operation	1	0000 0000 0000 0000	None	
NOP	—	No Operation	1	1111 xxxx xxxx xxxx	None	4
POP	—	Pop top of return stack (TOS)	1	0000 0000 0000 0110	None	
PUSH	—	Push top of return stack (TOS)	1	0000 0000 0000 0101	None	
RCALL	n	Relative Call	2	1101 1nnn nnnn nnnn	None	
RESET		Software device Reset	1	0000 0000 1111 1111	All	
RETFIE	s	Return from interrupt enable	2	0000 0000 0001 000s	GIE/GIEH, PEIE/GIEL	
RETLW	k	Return with literal in WREG	2	0000 1100 kkkk kkkk	None	
RETURN	s	Return from Subroutine	2	0000 0000 0001 001s	None	
SLEEP	—	Go into Standby mode	1	0000 0000 0000 0011	$\overline{TO}$, $\overline{PD}$	

Note 1: When a Port register is modified as a function of itself (e.g., `MOVF PORTB, 1, 0`), the value used will be that value present on the pins themselves. For example, if the data latch is '1' for a pin configured as input and is driven low by an external device, the data will be written back with a '0'.

2: If this instruction is executed on the TMR0 register (and where applicable, d = 1), the prescaler will be cleared if assigned.

3: If Program Counter (PC) is modified or a conditional test is true, the instruction requires two cycles. The second cycle is executed as a `NOP`.

4: Some instructions are 2-word instructions. The second word of these instructions will be executed as a `NOP` unless the first word of the instruction retrieves the information embedded in these 16 bits. This ensures that all program memory locations have a valid instruction.

5: If the table write starts the write cycle to internal memory, the write will continue until terminated.

Mnemonic, Operands		Description	Cycles	16-Bit Instruction Word MSb LSb	Status Affected	Notes
LITERAL OPERATIONS						
ADDLW	k	Add literal and WREG	1	0000 1111 kkkk kkkk	C, DC, Z, OV, N	
ANDLW	k	AND literal with WREG	1	0000 1011 kkkk kkkk	Z, N	
IORLW	k	Inclusive OR literal with WREG	1	0000 1001 kkkk kkkk	Z, N	
LFSR	f, k	Move literal (12-bit) 2nd word to FSRx 1st word	2	1110 1110 00ff kkkk 1111 0000 kkkk kkkk	None	
MOVLB	k	Move literal to BSR<3:0>	1	0000 0001 0000 kkkk	None	
MOVLW	k	Move literal to WREG	1	0000 1110 kkkk kkkk	None	
MULLW	k	Multiply literal with WREG	1	0000 1101 kkkk kkkk	None	
RETLW	k	Return with literal in WREG	2	0000 1100 kkkk kkkk	None	
SUBLW	k	Subtract WREG from literal	1	0000 1000 kkkk kkkk	C, DC, Z, OV, N	
XORLW	k	Exclusive OR literal with WREG	1	0000 1010 kkkk kkkk	Z, N	
DATA MEMORY ↔ PROGRAM MEMORY OPERATIONS						
TBLRD*		Table Read	2	0000 0000 0000 1000	None	
TBLRD*+		Table Read with post-increment		0000 0000 0000 1001	None	
TBLRD*-		Table Read with post-decrement		0000 0000 0000 1010	None	
TBLRD+*		Table Read with pre-increment		0000 0000 0000 1011	None	
TBLWT*		Table Write	2 (5)	0000 0000 0000 1100	None	
TBLWT*+		Table Write with post-increment		0000 0000 0000 1101	None	
TBLWT*-		Table Write with post-decrement		0000 0000 0000 1110	None	
TBLWT+*		Table Write with pre-increment		0000 0000 0000 1111	None	

Note 1: When a Port register is modified as a function of itself (e.g., `MOVF PORTB, 1, 0`), the value used will be that value present on the pins themselves. For example, if the data latch is '1' for a pin configured as input and is driven low by an external device, the data will be written back with a '0'.

2: If this instruction is executed on the TMR0 register (and where applicable, d = 1), the prescaler will be cleared if assigned.

3: If Program Counter (PC) is modified or a conditional test is true, the instruction requires two cycles. The second cycle is executed as a `NOP`.

4: Some instructions are 2-word instructions. The second word of these instructions will be executed as a `NOP` unless the first word of the instruction retrieves the information embedded in these 16 bits. This ensures that all program memory locations have a valid instruction.

5: If the table write starts the write cycle to internal memory, the write will continue until terminated.

Bibliographie

[1] PIC18FXX8 Data Sheet, 28/40-Pin High-Performance, Enhanced Flash Microcontrollers with CAN Module. Microchip Technology Inc.2024

[2] Lakhdari Fethi, Polycopié de Cours : Le Microprocesseur 16 bits "MC68000",Universite des Sciences et de la Technologie d'Oran (USTO),2020.

[3] Julio Sanchez, Microcontrollers High-Performance Systems and Programming, CRC PRESS. 2013

[4] Sid Katzen, The Essential PIC18Microcontroller, Springer-Verlag London, 2010.