الجمهورية الجزائرية الديمقراطية الشعبية
République Algérienne Démocratique et Populaire
وزارة التعليم العالي و البحث العلمي
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université Ziane Achour de Djelfa — جامعة الجلفة
Faculté des Sciences et de la Technologie — كلية العلوم و التكنولوجيا
Département de Génie Electrique — قسم الهندسة الكهربائية

Semestre: 1

Matières: Microprocesseurs et Microcontrôleurs

1er annee Master ELT(Electrotechnique)

Chapitre 5

Applications des microprocesseurs et microcontrôleurs

Contenu de la matière :

Interface LCD - Clavier Interface - Génération de signaux des ports Porte pour convertisseurs - Moteur- Contrôle - Contrôle des appareils DC / AC -mesure de la fréquence - système d'acquisition de données

Réalisé et présenté par :

Dr. Obeidi Thameur

Année Universitaire : 2023 / 2024

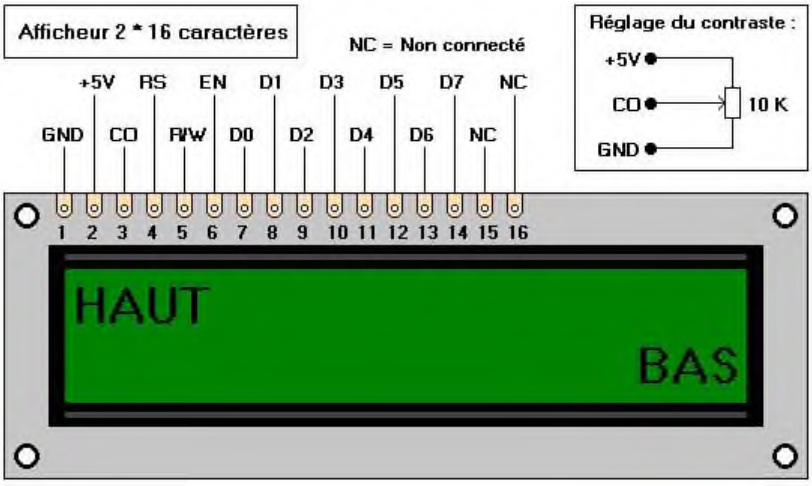
PROGRAMMATION DU PIC 16F877A

GESTION DE L'AFFICHAGAGE LCD 2X16 ET DU CLAVIER MATRICIEL

Objectif : A l'aide des microcontrôleurs, il est possible de réaliser des montages avec une véritable gestion de l'information et des protocoles pour la communication de données (entrées/sorties). Il devient possible d'utiliser, pour l'interface utilisateur, des périphériques "évolués" comme des claviers matriciels ou encore des afficheurs à cristaux liquides (LCD). Nous nous intéressons dans ce 3ème TP à la gestion du clavier matriciel et de l'afficheur LCD 2x16.

I- L'afficheur LCD 2x16

La figure ci-dessous montre le brochage d'un afficheur LCD 2x16 :



Le tableau suivant donne une description rapide de la fonction de chaque broche :

En observant le brochage de l'afficheur, on constate qu'il faut un minimum de 6 sorties pour le commander. En effet, si on utilise l'adressage sur 4 bits et que l'on se prive de la lecture dans l'afficheur (ce n'est pas nécessaire, donc on relie R/W à la masse), il nous faut commander les six broches : EN, RS, D4, D5, D6, et D7.

N°	Broche	Fonction
1	GND	Masse de l'alimentation
2	+5V	Alimentation 5V
3	CO	Réglage du contraste
4	RS	0 : Instruction 1 : Donnée
5	R/W	0 : Ecriture 1 : Lecture
6	EN	Validation
7	D0	Données Dans le cas de l'adressage en 4 Bits, seuls les bits D4...D7 sont utilisés. Les bits D0...D3 ne sont alors pas connectés.
8	D1	
9	D2	
10	D3	
11	D4	
12	D5	
13	D6	
14	D7	

Le compilateur MikroC utilise des fonctions (bibliothèques) pour simplifier aux utilisateurs l'exploitation des afficheurs LCD. On utilise souvent l'instruction "**sbit**" pour affecter chaque broche du LCD à une broche du PIC.

Remarque : Utiliser "**LCD Library**" à partir du help du MikroC.

Association des broches du LCD au port b du pic	définition du sens
sbit LCD_RS at RB4_bit;	sbit LCD_RS_Direction at TRISB4_bit;
sbit LCD_EN at RB5_bit;	sbit LCD_EN_Direction at TRISB5_bit;
sbit LCD_D7 at RB3_bit;	sbit LCD_D7_Direction at TRISB3_bit;
sbit LCD_D6 at RB2_bit;	sbit LCD_D6_Direction at TRISB2_bit;
sbit LCD_D5 at RB1_bit;	sbit LCD_D5_Direction at TRISB1_bit;
sbit LCD_D4 at RB0_bit;	sbit LCD_D4_Direction at TRISB0_bit;

Ainsi, d'autres fonctions peuvent être utilisées :

Lcd_Out : affiche un texte à la position spécifiée, exemple Lcd_Out (1, 1, "TP3")

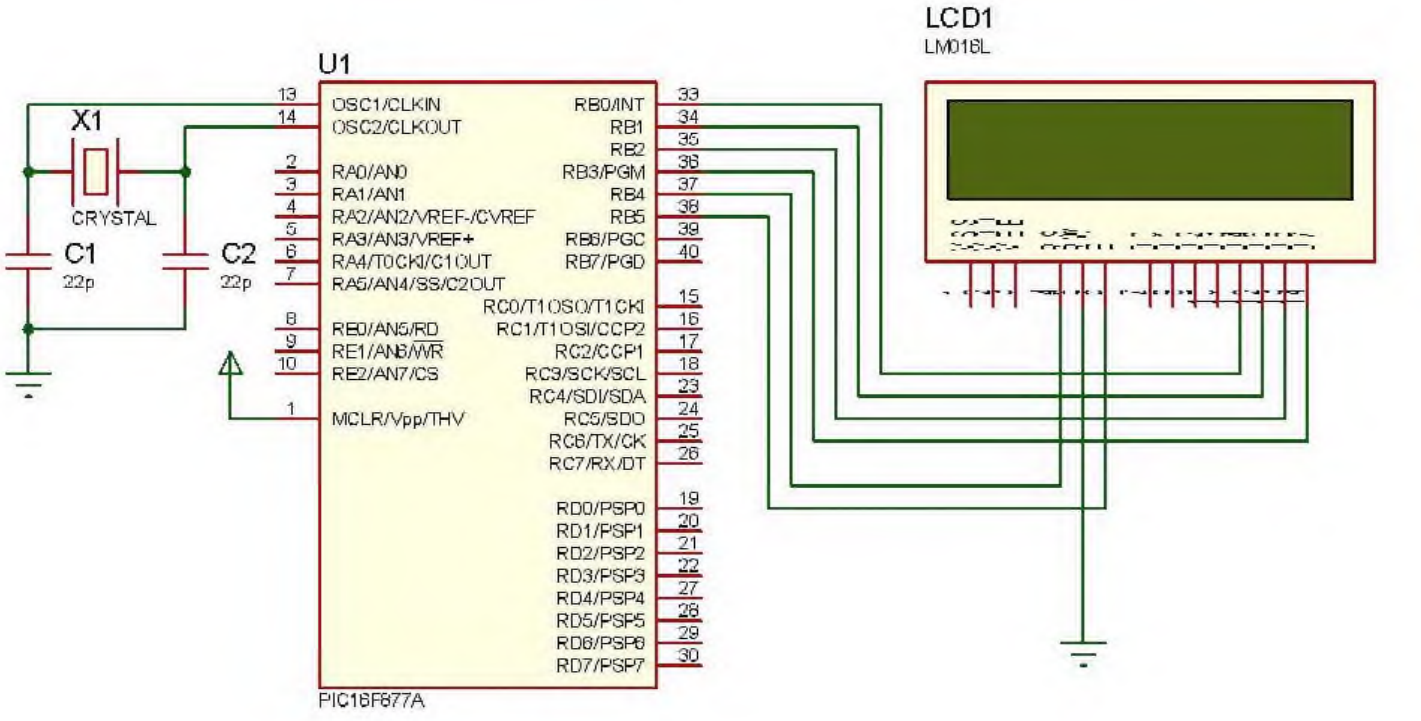
Lcd_Out_Cp : affiche un texte à la position courante, exemple Lcd_Out (1, 1, "ici")

Lcd_Chr : affiche un caractère à la position spécifiée, exemple Lcd_Chr (1, 3, "T")

Lcd_Cmd : envoi une commande au LCD, exemple Lcd_Cmd(_LCD_CLEAR) ;

II- Gestion de l'affichage d'un message via le PIC 16F877A et l'afficheur LCD 2x16

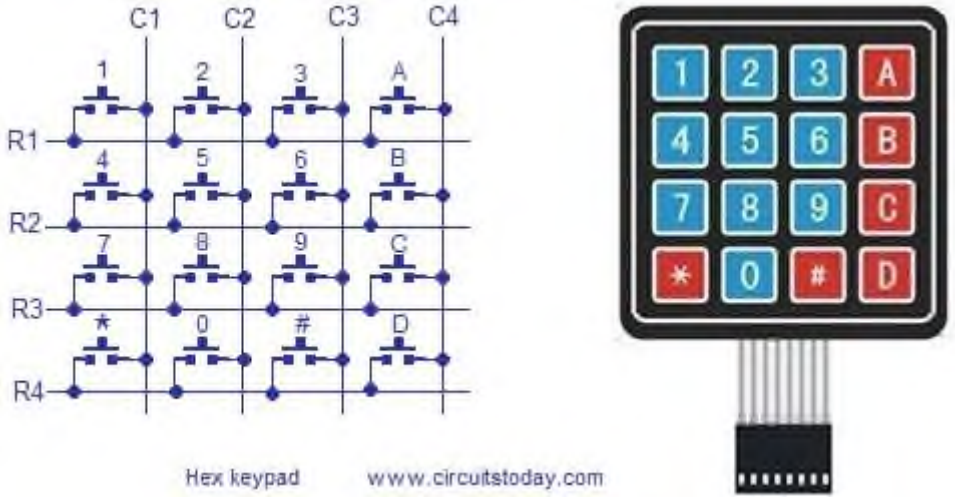
En utilisant le compilateur MikroC et Proteus ISIS, réaliser un programme qui permet d'afficher un message sur l'afficheur LCD 2x16. A titre d'exemple, réaliser le schéma suivant ci-dessous et la librairie LCD : MikroC → Help → Lcd Library.

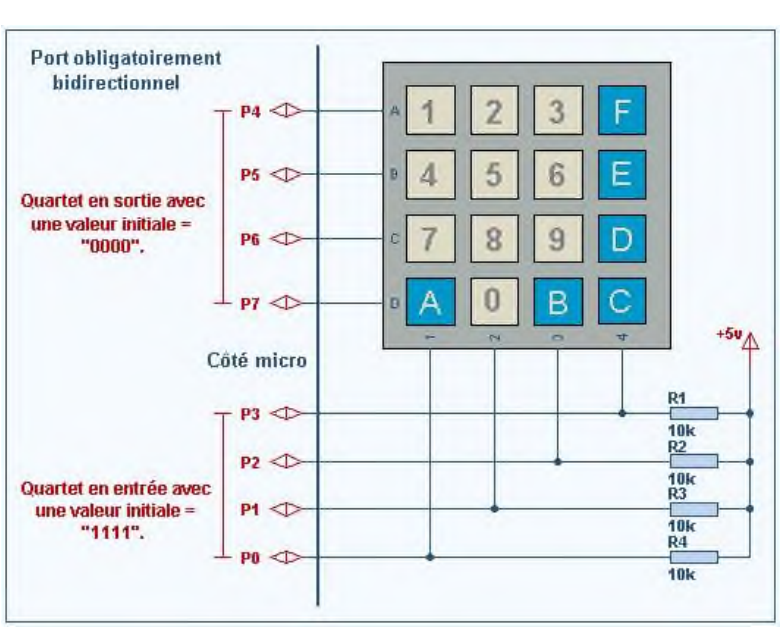



```
// les connections de l afficheur LCD 2x16
sbit LCD_RS at RB4_bit;
sbit LCD_EN at RB5_bit;
sbit LCD_D4 at RB0_bit;
sbit LCD_D5 at RB1_bit;
sbit LCD_D6 at RB2_bit;
sbit LCD_D7 at RB3_bit;
 //les directions
sbit LCD_RS_Direction at TRISB4_bit;
sbit LCD_EN_Direction at TRISB5_bit;
sbit LCD_D4_Direction at TRISB0_bit;
sbit LCD_D5_Direction at TRISB1_bit;
sbit LCD_D6_Direction at TRISB2_bit;
sbit LCD_D7_Direction at TRISB3_bit;
// fin des connections du module LCD
void main() {
Lcd_Init();    // initialisation du LCD
Lcd_Cmd(_LCD_CLEAR); // effacer l'écran

while(1)   {
Lcd_Out(1, 1, " MASTER1 TP SESTR ");
Lcd_Out(2, 1, " TP3 , 2019 ");
Lcd_Cmd(_LCD_CURSOR_OFF); // éliminer le curseur
delay_ms(1000);
Lcd_Out(1, 1, "Univ Constantine");
Lcd_Out(2, 1, "Merci bien ");
Lcd_Cmd(_LCD_CURSOR_OFF); // éliminer le curseur
delay_ms(1000);
//Lcd_Out_Cp("ICI");
//Lcd_Cmd(_LCD_CURSOR_OFF); // éliminer le curseur
}
}
```

III- Gestion de l'affichage via le PIC 16F877A, l'afficheur LCD 2x16 et un clavier matriciel (keypad)

A titre d'exemple, un clavier matriciel 4x4 est composé de 4 lignes (L1 L2 L3 L4) et de 4 colonnes (C1 C2 C3). Lorsqu'une touche est enfoncée, une connexion entre une ligne et une colonne est établi.





Le brochage dit "matrice carrée" (comme le montre la figure) nécessite l'utilisation d'un port parallèle, obligatoirement bidirectionnel, complet (8 bits de P0 à P7). Le poids faible de ce port sera réservé pour les colonnes et initialisé en entrée avec la valeur "1111" tandis que son poids fort est affecté aux lignes et initialisé en sortie avec la valeur "0000".

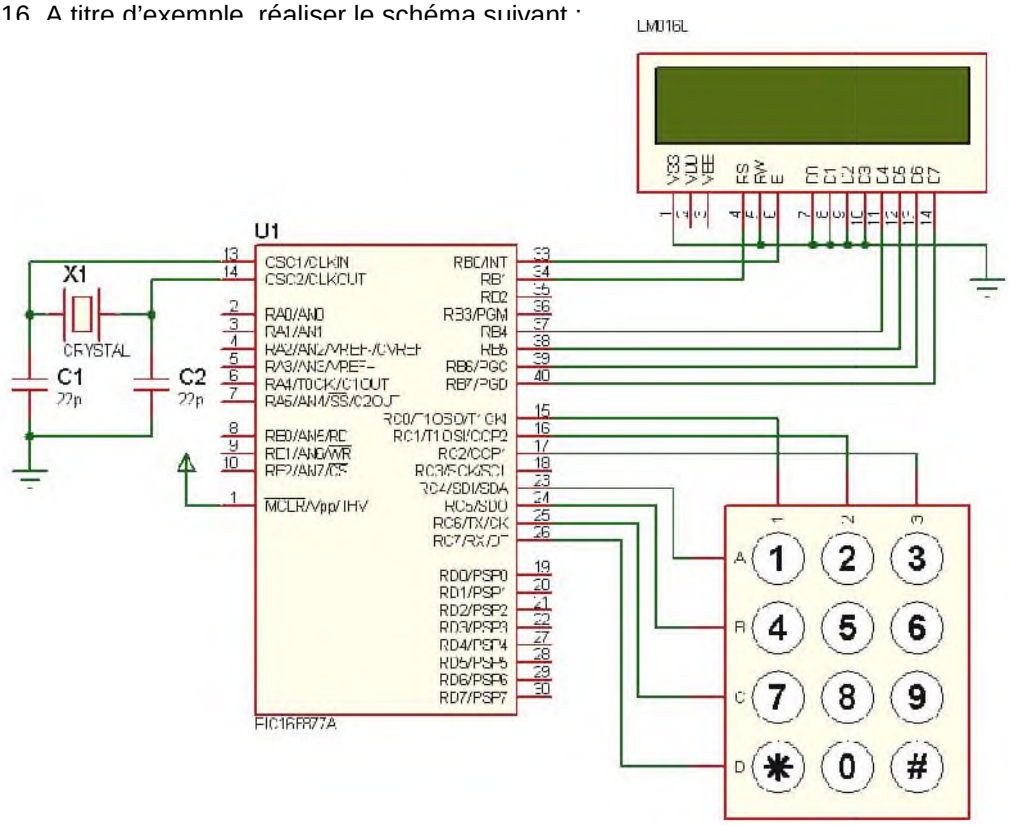
Codage du clavier matriciel : On sait déjà qu'une touche enfoncée provoque un court-circuit entre la ligne et la colonne correspondantes. Si ce court-circuit est matérialisé par un niveau logique 0 et tout le reste est à 1, on peut alors facilement, établir le tableau de la figure suivante :

	P7	P6	P5	P4	P3	P2	P1	P0	
Touches	**Lignes**				**Colonnes**				**Codes**
(Valeur ASCII)	**D**	**C**	**B**	**A**	**4**	**3**	**2**	**1**	**Hexa**
'1' => 49	1	1	1	0	1	1	1	0	**EE**
'2' => 50	1	1	1	0	1	1	0	1	**ED**
'3' => 51	1	1	1	0	1	0	1	1	**EB**
'A' => 65	1	1	1	0	0	1	1	1	**E7**
'4' => 52	1	1	0	1	1	1	1	0	**DE**
'5' => 53	1	1	0	1	1	1	0	1	**DD**
'6' => 54	1	1	0	1	1	0	1	1	**DB**
'B' => 66	1	1	0	1	0	1	1	1	**D7**

'7' => 55	1	0	1	1	1	1	1	0	BE
'8' => 56	1	0	1	1	1	1	0	1	BD
'9' => 57	1	0	1	1	1	0	1	1	BB
'C' => 67	1	0	1	1	0	1	1	1	B7
'*' => 42	0	1	1	1	1	1	1	0	7E
'0' => 48	0	1	1	1	1	1	0	1	7D
' # ' => 35	0	1	1	1	1	0	1	1	7B
'D' => 68	0	1	1	1	0	1	1	1	77

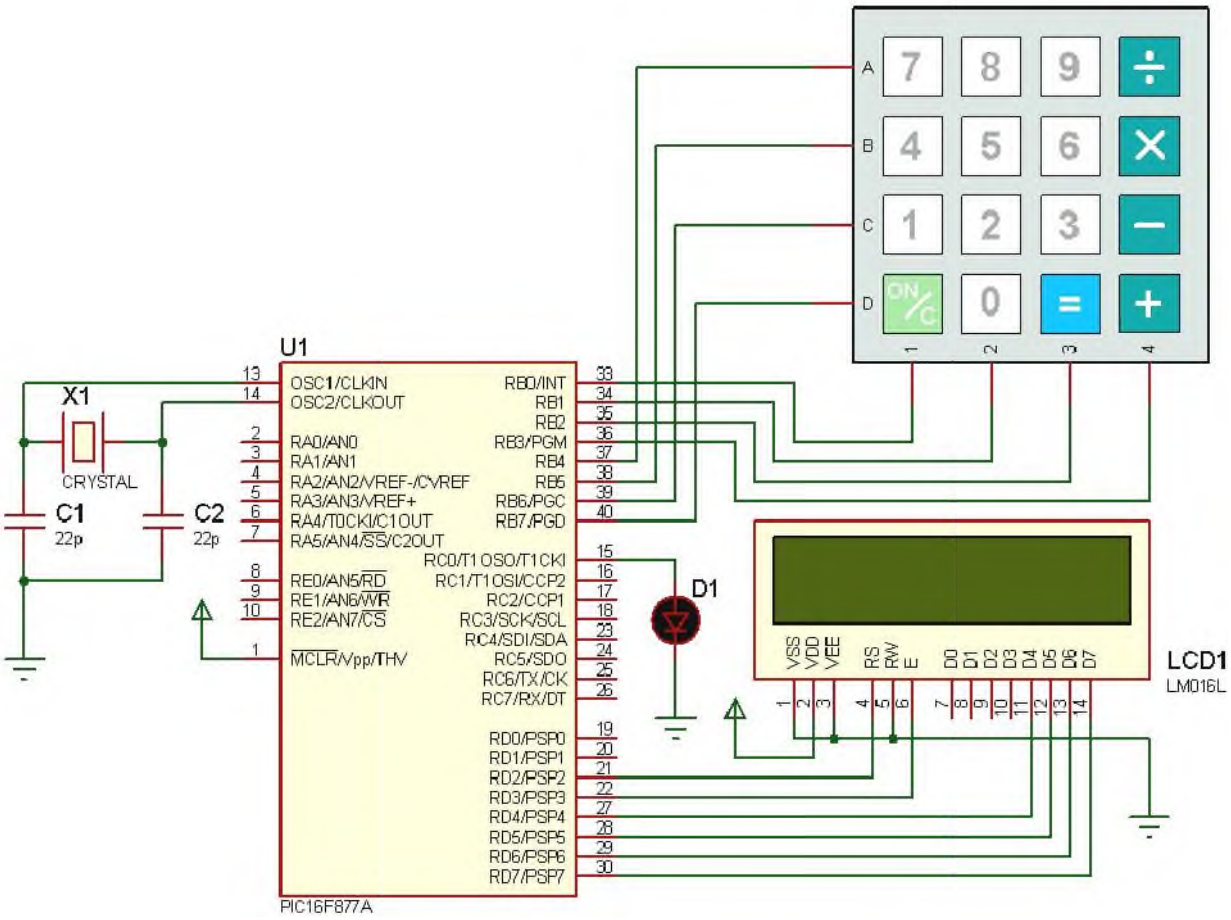
Décodage du clavier matriciel : L'acquisition d'une touche se fera par scrutation (examen), en effectuant une lecture permanente de l'état du clavier. A l'appui d'une touche, le microcontrôleur se dépêche pour explorer judicieusement les codes, déjà établis théoriquement, dans le tableau précédent.

En utilisant le compilateur MikroC et Proteus ISIS, réaliser un programme qui permet d'afficher le la touche enfoncée sur l'afficheur LCD 2x16. A titre d'exemple, réaliser le schéma suivant :



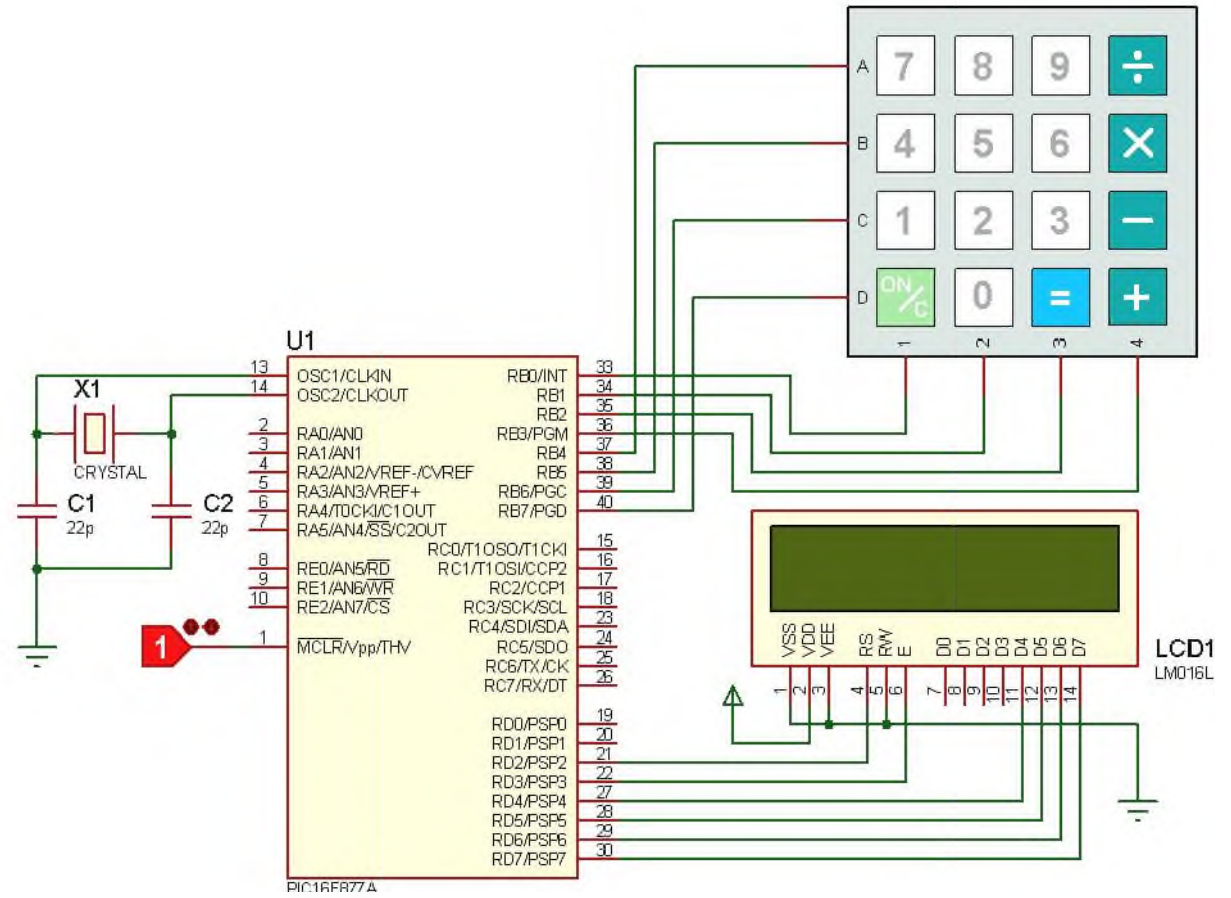

```
// connexions du module Keypad
char keypadPort at PORTC;
// connexions du module LCD
sbit LCD_RS at RB1_bit;
sbit LCD_EN at RB0_bit;
sbit LCD_D4 at RB4_bit;
sbit LCD_D5 at RB5_bit;
sbit LCD_D6 at RB6_bit;
sbit LCD_D7 at RB7_bit;
sbit LCD_RS_Direction at TRISB1_bit;
sbit LCD_EN_Direction at TRISB0_bit;
sbit LCD_D4_Direction at TRISB4_bit;
sbit LCD_D5_Direction at TRISB5_bit;
sbit LCD_D6_Direction at TRISB6_bit;
sbit LCD_D7_Direction at TRISB7_bit;

void main() {
   unsigned short kp = 0;
Keypad_Init(); // Initialisation du Keypad
Lcd_Init(); // Initialisation du Lcd
Lcd_Cmd(_LCD_CLEAR); // effacement de l'écran
Lcd_Cmd(_LCD_CURSOR_OFF); // Cursor off
Lcd_Out(1, 1, "Clavier 3x4");
Lcd_Out(2, 1, "Touche :"); // écrire un message dans LCD
do
{
kp = 0; // Reset la variable
// attendre que le bouton soit pressé et relâché
do
kp = Keypad_Key_Click(); // sauvegarder le code
while (!kp);
// transformation du code en valeur ASCII
switch (kp)
{
case  1: kp = 49; break; // 1
case  2: kp = 50; break; // 2
case  3: kp = 51; break; // 3
//case  4: kp = 65; break; // A
case  5: kp = 52; break; // 4
case  6: kp = 53; break; // 5
case  7: kp = 54; break; // 6
//case  8: kp = 66; break; // B
case  9: kp = 55; break; // 7
case 10: kp = 56; break; // 8
case 11: kp = 57; break; // 9
//case 12: kp = 67; break; // C
case 13: kp = 42; break; // *
case 14: kp = 48; break; // 0
case 15: kp = 35; break; // #
//case 16: kp = 68; break; // D
}
Lcd_Chr(2, 10, kp); // affichage de la valeur ASCII sur LCD
} while (1);
}
```

Construire un programme en MikroC PRO qui permet la commande d'une serrure codée à l'aide de 3 chiffres ?



Exercice 2

Construire un programme en MikroC PRO qui permet de générer le fonctionnement d'une calculatrice simple ?

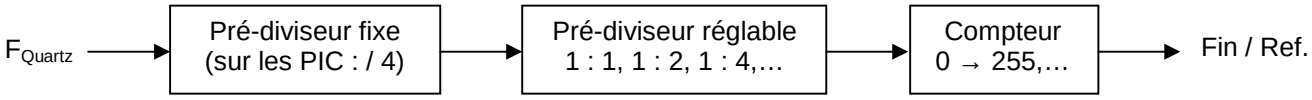


PROGRAMMATION DU PIC 16F877A
TIMER2, PWM ET CONTROLE DE LA VITESSE D'UN MOTEUR CC

Objectif : L'objectif capital de ce 2ème TP est de réaliser un programme en MikroC qui permet de configurer le Timer2 du PIC 16F877A pour générer une modulation de largeur d'impulsion PWM (Pulse Width Modulation). Le signal PWM généré est utilisé pour contrôler la vitesse d'un moteur à courant continu (MCC). La vérification est faite via Proteus ISIS.

I- PIC 16F877A et PWM mode

En bref, le PIC 16F877A contient 3 Timers internes : **Timer0**, **Timer2** sur 8 bits et **Timer1** sur 16 bits. Pour réaliser une temporisation, le principe de fonctionnement est basé sur l'utilisation d'un compteur qui s'incrémente à chaque front montant du signal qui lui est appliqué.



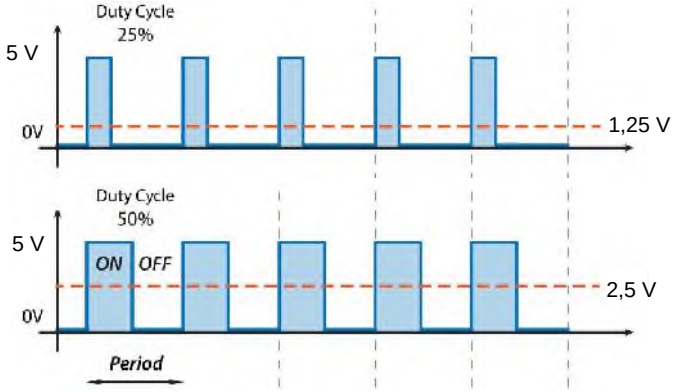
Un Timer doit pouvoir compter un temps défini par le programme (ex. 1 ms, 20 ms, 50 ms,…etc.). Deux paramètres peuvent être modifiés : la fréquence du signal appliqué au compteur et le nombre d'impulsions à compter.

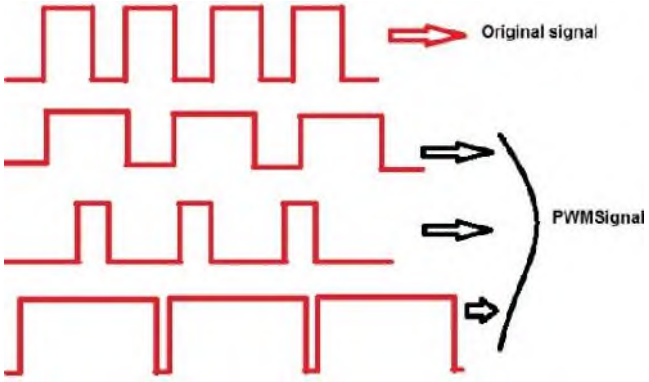
Les méthodes de configuration sont les suivantes :

- Modification de la fréquence du signal appliqué au compteur (pré-diviseur ou "prescalar" en anglais).
- Modification du nombre d'impulsions à compter (charger une valeur initiale pour changer le temps de comptage).

En effet, le Timer2 possède un pré-compteur fixe et un pré-compteur variable (1 : 1, 1 : 4 et 1 : 16).

PWM est un signal dont le rapport cyclique varie. Il est utilisé dans plusieurs applications (la commande des servomoteurs, la variation de la vitesse des moteurs à courant continu, la conversion numérique analogique,…etc.).





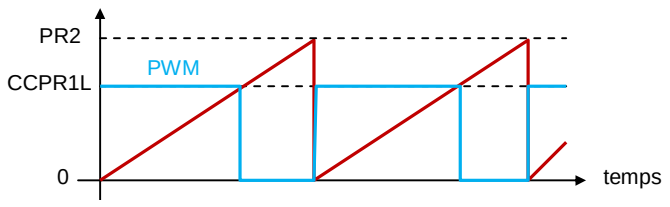
Le PIC 16F877A utilise le module CCP (Capture/Compare/PWM) en mode PWM. La broche RC2 doit être configurée en sortie. La valeur de la période est chargée dans le registre PR2 et la valeur du rapport cyclique dans le registre CCPR1L.

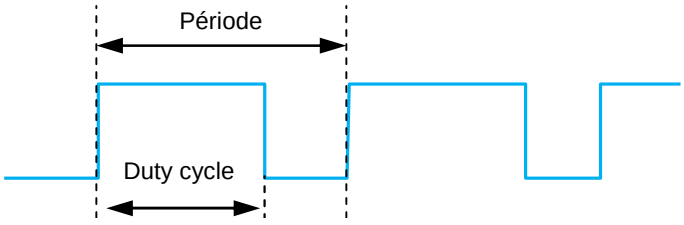
Le Timer2 (8 bits) compte l'horloge interne après une pré-division (prescaler) programmable par 1:1, 1:4 ou 1:16. Quand Timer2 atteint la valeur dans PR2, Timer2 passe à zéro et la broche PWM est mise à 1. Quand Timer2 atteint la valeur dans CCPR1L, la sortie PWM est mise à 0.

$$\text{La période} = 4 \times (PR2 + 1) \times Tosc \times \text{Valeur du pré-diviseur}$$

$$\text{La durée} = CCPR1L \times Tosc \times \text{Valeur du pré-diviseur}$$

$$\text{Avec : } Tosc = \frac{1}{\text{Fréquence horloge (Quartz)}}$$





En effet, pour générer le signal PWM, on peut utiliser ce principe qui agit directement sur les registres. Par conséquent, on peut utiliser les fonctions fournis par le compilateur MikroC PRO :

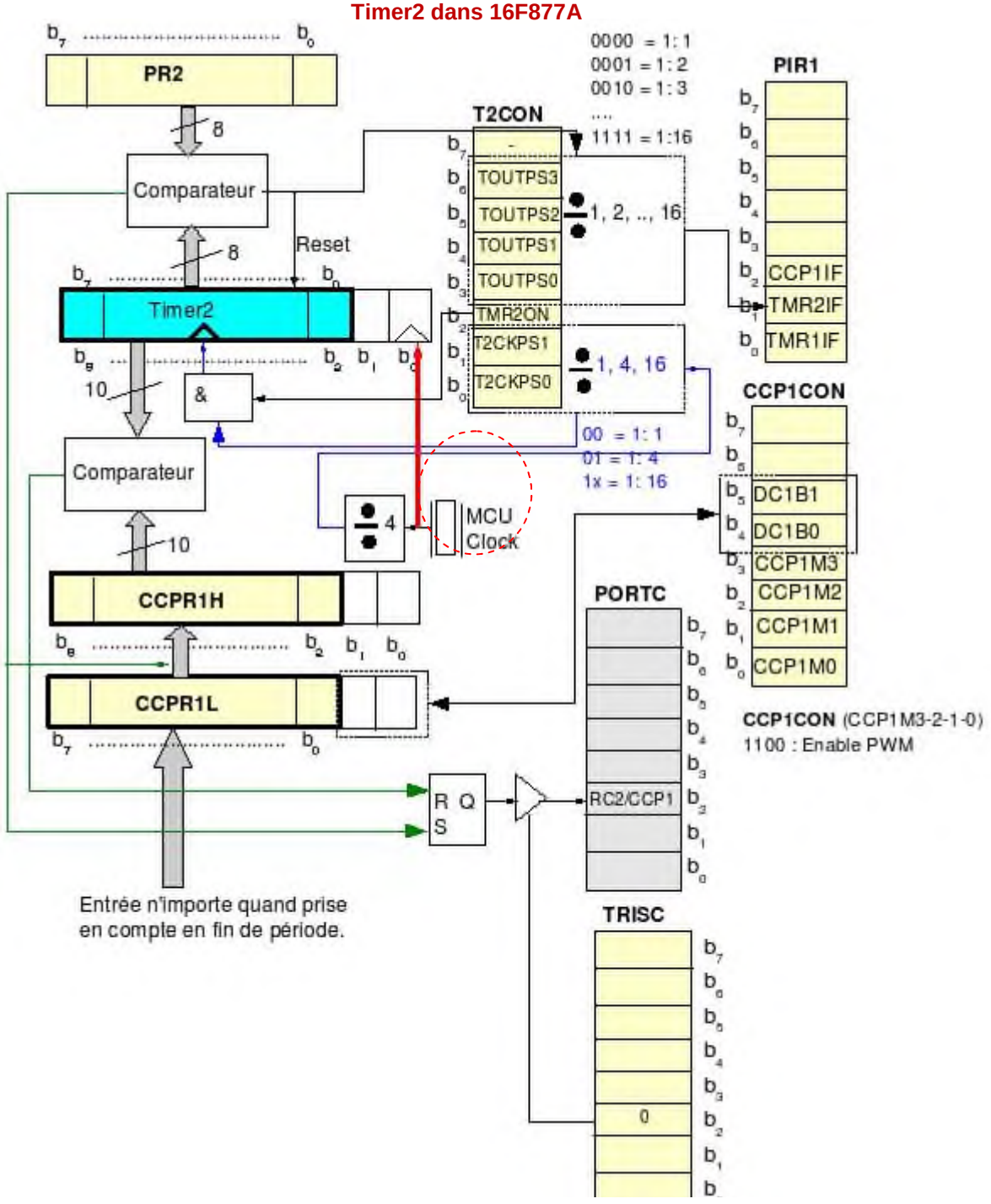
- Pwm_init(freq) : pour le choix de la fréquence en Hz.
- Pwm_Start() : pour le démarrage du module PWM.
- Pwm_Set_Duty(durée) : pour changer le rapport cyclique (de 0 pour 0% à 255 pour 100%).
- Pwm_Stop() : pour l'arrêt du module PWM.

II- Configuration du Timer2 et PWM (PIC 16F877A) : Configuration du CCP module

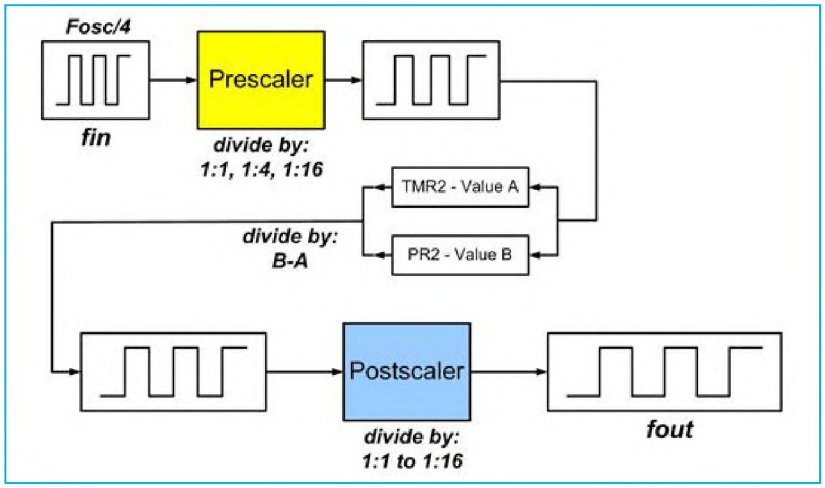
La configuration du Timer2 dans le PIC 16F877A est faite via les registres suivants : T2CON (TMR2), CCP1CON, PR2, CCPR1L. Ceci est réalisé à l'aide du comptage des cycles d'horloge (après la pré-division programmable).

La configuration du CCP module permet de générer un signal PWM de différentes formes ou la génération d'une temporisation bien déterminée ce qui permettra la commande d'un moteur CC, d'un servomoteur, …etc.

Timer2 dans 16F877A



La configuration du "post-scaler" sera étudiée dans un prochain TP.



III- Exemple descriptif de Configuration du Timer2

Ecrire un programme en MikroC qui permet de configurer le Timer2 pour générer d'un signal PWM de période maximale avec un rapport cyclique 50% (duty cycle). Visualiser le signal générer via Proteus ISIS. Vérifier les résultats obtenus ? Refaire la même chose pour les cas des rapports cycliques suivants : 50 %, 25% et 12,5% ?

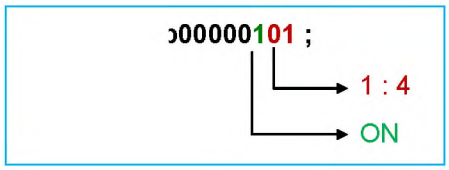
La période $= 4 \times (PR2 + 1) \times Tosc \times$ Valeur du pré-diviseur

Timer2 : registre 8 bits → période maximale pour PR2 = 256 – 1 = 255 (8 bits : comptage de 00000000 à 11111111).

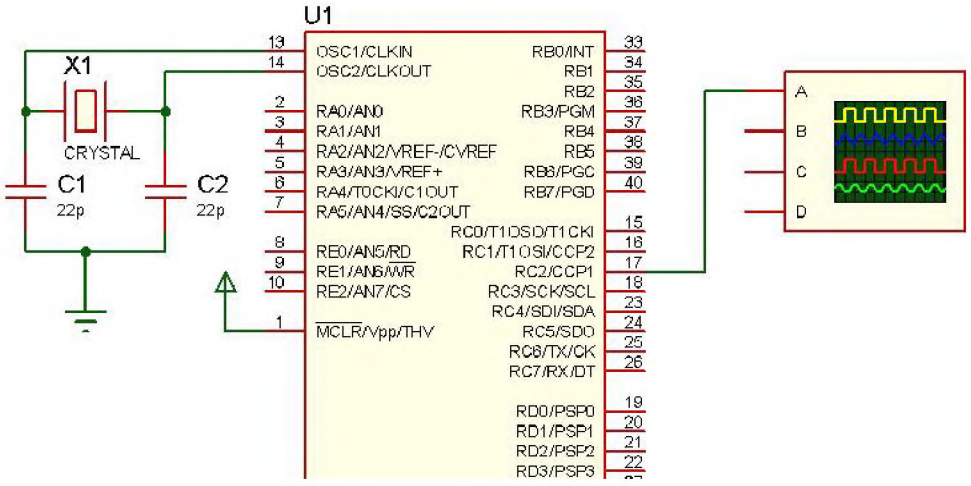
Soit, par exemple, un pré-diviseur **1 : 4** et un Oscillateur **8 MHz**. On obtient donc : CCPR1L = 256 / 2 = 128.

T_{PWM} = (PR2 +1) x (Tosc x 4 x $Timer2_{Prescaler}$) = (255 + 1) x ($1/8.10^6$) x 4 x 4 = 0,512 ms (soit f = 1,95 KHz).

```
void main() {
  unsigned char dc;  // dc : valeur pour duty cycle (rapport cyclique)
  TRISC=0;     // Port C en mode sortie
  PORTC=0;     // Clear port C
   // Configuration du CCP module à f_out  et  PWM Sortie
  PR2=255;    // Période Max pour Timer2 (compteur 8 bits)
  T2CON=0b00000101;  // 1 : TMR2 ON 00 -> 1:1  01 -> 1:4  1x -> 1:16
  CCP1CON=0b00001100;  // 1100 : Enable PWM
  dc=128;
  while(1)
  {
    CCPR1L=dc;
  }

}
```



Remarque : il est possible de configurer et d'utiliser CCP2CON, CCPR2L et RC1 (PIC broche 16) de la même manière.



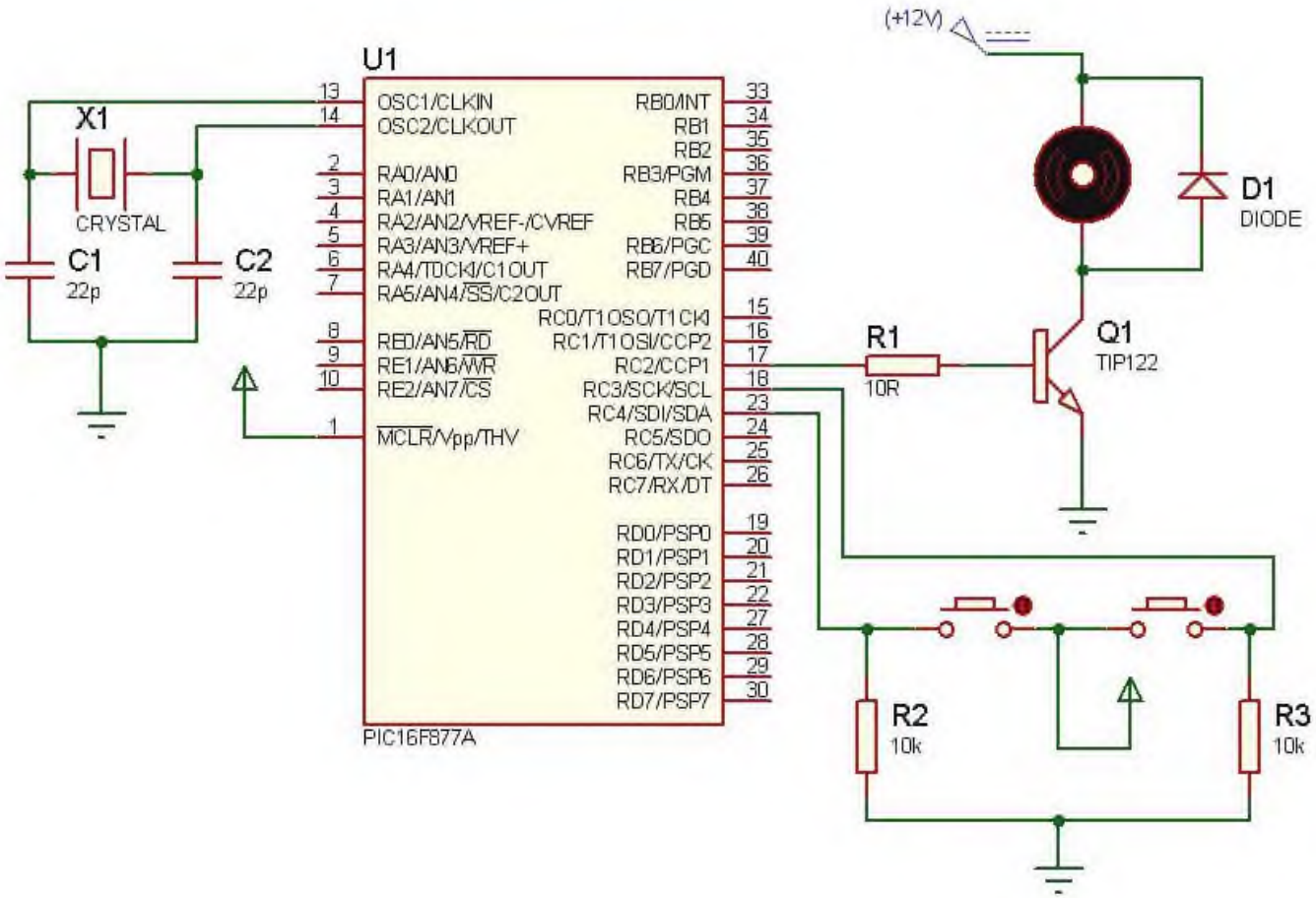
1) Construire un programme en MikroC PRO qui permet la génération d'un signal PWM carré de fréquence 1 KHz.

2) Construire un programme en MikroC PRO qui permet la génération d'un signal PWM de fréquence 2 KHz, dont le rapport cyclique est de 75%.

3) Ecrire un programme en MikroC PRO qui permet la génération d'un signal PWM de fréquence 2 KHz, dont le rapport cyclique varie continuellement entre 10% et 90% (pour ralentir la variation rapide, utiliser un palier de temps 100 ms).

4) Quelle est la temporisation maximale (T_{PWM_MAX}) qu'on peut générer avec Timer2 ?

5) Ecrire un programme en MikroC PRO qui permet la génération d'un signal PWM de fréquence 2 KHz, dont le rapport cyclique est modifié à l'aide de 2 boutons (augmentation et diminution). Vérifier le fonctionnement avec Proteus ISIS ?



Bibliographie

01. P. Andre. La liaison RS232. Dunod, Paris, 1998.
02. T.C. Bartee. Digital Computer Fundamentals. McGraw-Hill, Tokyo, 1981.
03. J. Campbell. L'interface RS-232. Sybex, Paris, 1984.
04. B. Fabrot. Assembleur pratique. Marabout Informatique, Alleur, Belgique, 1996.
05. *A*.B. Fontaine. Le microprocesseur 16 bits 8086/8088 - Matériel, logiciel, système d'exploitation. Masson, Paris, 1988.
06. *B*.Geoffrion. 8086 - 8088 - Programmation en langage assembleur. Editions Radio, Paris, 1986.
07. J.P. Hayes. Computer Architecture and Organization. McGraw-Hill, Tokyo, 1982. S. Leibson. Manuel des interfaces. McGraw-Hill, Paris, 1984.
08. H. Liien. Introduction a la micro-informatique - Du microprocesseur au microordinateur. Editions Radio, Paris, 1982.
09. H. Liien. 8088 et ses périphériques - Les circuits cles des IBM PC et compatibles. Editions Radio, Paris, 1985.
10. H. Liien. 8088 - Assembleur IBM PC et compatibles. Editions Radio, Paris, 1986.
11. H. Liien. Cours fondamental des microprocesseurs. Editions Radio, Paris, 1987.
12. H. Liien. Microprocesseurs - Du CISC au RISC. Dunod, Paris, 1995.
13. *G*.H. MacEwen. Introduction to Computer Systems. McGraw-Hill, Tokyo, 1981.
14. Mariatte. PC, modems et serveurs. P.S.I, Lagny, France, 1986.
15. P. Mercier. Assembleur facile. Marabout Informatique, Alleur, Belgique, 1989.
16. P. Mercier. La maîtrise du MS-DOS et du BIOS - Les interruptions - Organisation interne. Marabout Informatique, Alleur, Belgique, 1989.
17. P. Mercier. Les interruptions du MS-DOS. Marabout Informatique, Alleur, Belgique, 1990.
18. Osborne. Initiation aux micro-ordinateurs - Niveau 2. Editions Radio, Paris, 1981.
19. J.B. Peatman. Microcomputer Based Design. McGraw-Hill, Tokyo, 1981.
20. E. Pissaioux. Pratique de l'assembleur I80x86 - Cours et exercices. Herm's, Paris, 1994.
21. *H*.Schakei. Programmer en assembleur sur PC. Micro Application, Paris, 1995.

22. Tavernier. Modems. ETSF, Paris, 1993.
23. M. TiSCher et B. JennriCh. La bible PC - Programmation système. Micro Application, Paris, 1997.
24. R. Tourki. L'ordinateur PC - Architecture et programmation - Cours et exercices. Centre de Publication Universitaire, Tunis, 2002.
25. J. Tracy Kidder. The Soul of a New Machine. Atlantic-Little Brown, U.S.A, 1981.
26. J.M. Trio. Microprocesseurs 8086-8088 - Architecture et programmation. Eyrolles, Paris, 1984.
27. R. ZakS et A. WOLFE. Du composant au syst'me - Introduction aux microprocesseurs. Sybex, Paris, 1988.
28. Bigonoff (bigocours@hotmail.com) : "La programmation des PICs" - http://fribotte.free.fr/bdtech/cours/pic16f84/
29. Lycée Jacquard : "Le PIC 16FXX" - http://ejacquard.free.fr/dossier_lycee/Pic/cours_pic.htm
30. Lycée Jacquard : "MPLAB" - http://ejacquard.free.fr/dossier_lycee/Pic/cours_pic.htm
31. Microchip : "PIC16F8X – 18-pin Flash/EEPROM 8-bit micro-controllers" – DS30430C, http://www.microchip.com/1010/suppdoc/
32. Microchip : "PIC16F8X – EEPROM memory programming specification" – DS30262E, http://www.microchip.com/1010/suppdoc