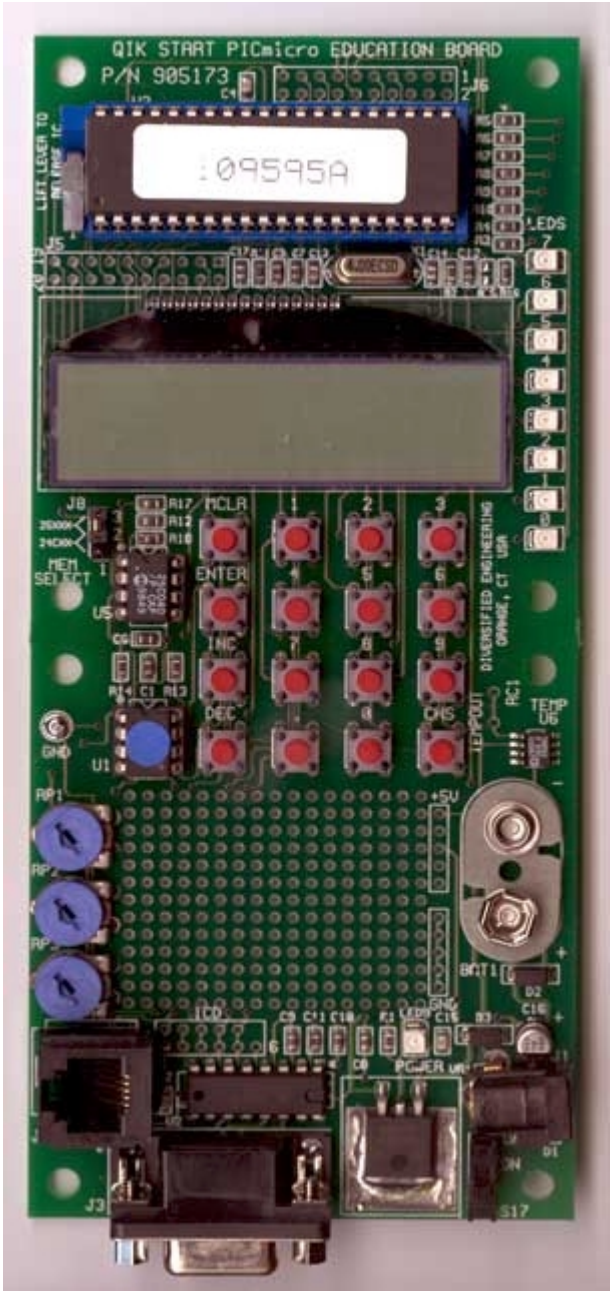
TP ATR :

Programmation du microcontrôleur PIC16F877

Sur la carte PICmicro Education Board II



Introduction :

Nous avons étudié en cours l'architecture des microcontrôleurs PIC et leur mode de programmation. L'objectif de ce TP est de mettre en pratique la partie théorique afin de concrétiser les concepts étudiés. A cette fin, nous tenterons d'utiliser la carte PICmicro Education Board II comme interface d'un programme que nous réaliserons pour le microcontrôleur.

Présentation du matériel :

Le Microcontrôleur :

Nous utiliserons le Microcontrôleur PIC16F877 construit par Microchip.

La carte d'interface :

La programmation d'un Microcontrôleur ne serait d'aucune utilité sans une carte d'interface permettant de concrétiser les actions du programme :

C'est la fonction de l'Education Board II, celle ci permet en effet de communiquer avec le PIC via un clavier de boutons, un ensemble de LEDS, un écran LCD, des potentiomètres et des capteurs de température.

Dans ce TP, nous utiliserons les LEDS et les boutons.

Attention : l'alimentation doit être comprise entre 7 et 12 volts mais nous avons constaté qu'en dessous de 10 ou si le courant n'est pas assez important, la carte a des difficultés à fonctionner.

Le programmeur :

La programmation d'un Microcontrôleur nécessite enfin une interface entre le PC et le PIC. Il en existe de plus ou moins sophistiqués, certains permettant même d'effectuer un debuggage temps réel, nous utiliserons le PIC-01 qui a l'avantage d'être le moins cher. (avantage certain en ces temps de disette pré-cloture du budget).

Les logiciels :

Nous utiliserons l'environnement de développement intégré MPLAB développé par Microchip pour programmer le PIC. Comme tout IDE qui se respecte, il permet de créer des projets, d'éditer les fichiers qu'il contient et de compiler et linker le tout.

MPLAB peut aussi permettre le debuggage du Microcontrôleur mais cette opération nécessite un programmeur plus évolué et nettement plus onéreux.

ICPROG sera utilisé pour charger les fichiers .hex générés par MPLAB.

Bon on va peut être pouvoir le commencer ce TP ?

Allumage d'une LED :

Préliminaires : (même pour Lulu)

- **Structure d'un programme :**

Il est tout d'abord nécessaire de présenter la structure d'un programme assembleur pour MPASM :

```
      LIST P=16F877

#include "P16F877.INC"

      __CONFIG _BODEN_ON&_CP_OFF&_WRT_ENABLE_ON&_PWRTE_ON&_WDT_OFF&_HS_OSC…
```

L'entête inclut la spécification du Microcontrôleur utilisé et des paramètres qui ne sont pas le sujet de ce TP.

```
        org     0x000                   ;Indique le début du programme
        goto    init

        org     0x004                   ;indique la marche à suivre en cas d'interruption
        goto    int
```

La fonction org permet de définir les vecteurs de démarrage du programme et d'exécution des interruptions, on utilise généralement des branchements.

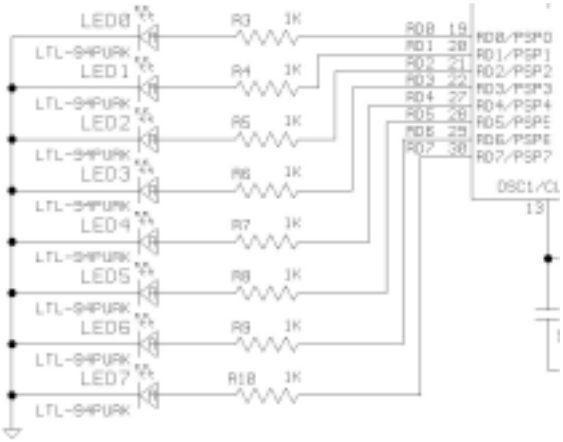
```
init    BSF     STATUS , RP0            ;Page d'Adresse 1 -> on a accès aux TRIS (CF M14)
        .
        .
        .
int     RETFIE
        END
```

S'ensuit le corps du programme fini par END.

Les instructions sont détaillées à partir de la page 26 du cours et les CF en fin de lignes renvoient au cours.

- **Fonctionnement des LEDS :**

Comme vous pourrez le voir sur le schéma de la carte, les huit LEDS sont montées en sortie des huit pattes du port :



Il suffit donc de mettre un bit sur le registre PORTC afin d'obtenir une tension de 5 volts sur la patte correspondante, mais pour cela, il faut tout d'abord le configurer en sortie car les ports sont configurés comme entrées par défaut.

Le registre TRISC est utilisé pour définir les entrées sorties des ports. Le nième bit du registre définit la qualité d'entrée ou sortie de la nième ligne du port correspondant, un 0 comme Output représentant une sortie et un 1 comme Input, une entrée (1 par défaut).

Le registre PORTC sera donc mis à 0 afin de pouvoir agir sur les LEDS.

Nous sommes maintenant assez balaises pour écrire le programme. Le premier groupe qui me rend un truc nickel aura la chance de voir son programme exécuté sur la carte sur sa propre paillasse. Ouaaaah.

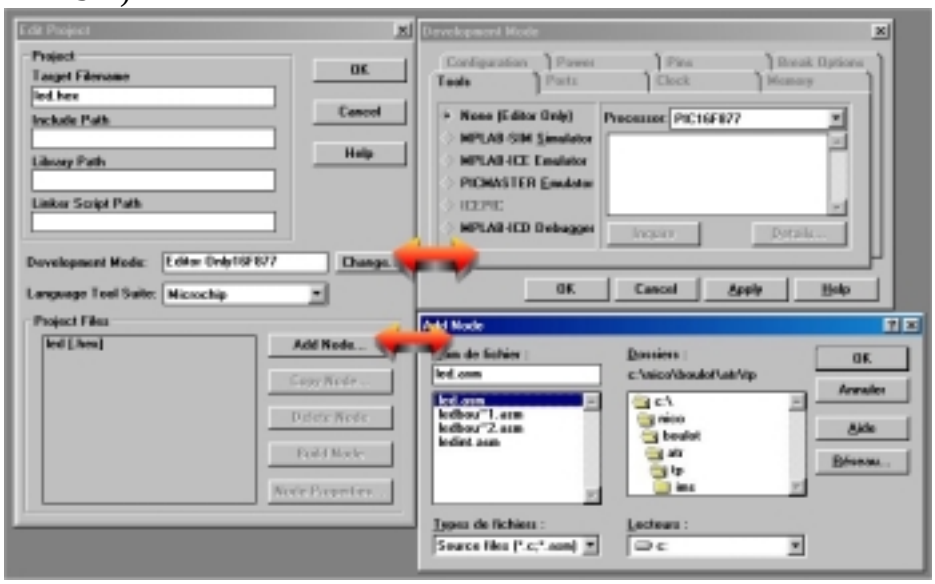
```
; Précise le PIC utilisé

        LIST P=16F877

#include "P16F877.INC"

  __CONFIG
_BODEN_ON&_CP_OFF&_WRT_ENABLE_ON&_PWRTE_ON&_WDT_OFF&_HS_OSC&_DEBUG_OFF&_CPD_OFF&_LVP_OFF

; Début du programme

            ORG     0x000                   ;Indique le début du programme
            GOTO    init

            ORG     0x004                   ;indique la marche à suivre en cas d'interruption
            GOTO    int

init        BSF     STATUS , RP0            ;Page d'Adresse 1 -> on a accès aux TRIS (CF M14)
            CLRF    INTCON                  ;Mise à 0 du vecteur d'interruption -> pas d'interruptions (CF M20)

            CLRF    TRISD                   ;Met le TRIS du port D à 0 -> port en mode Output

            BCF     STATUS , RP0            ;Page d'Adresse 0

            MOVLW   b'10000000'             ;Allume la première LED
            MOVWF   PORTD

debut       CLRWDT                          ;clear wdt
            GOTO    debut

int         RETFIE
            END                             ;!!!Ne pas oublier
```

- **Utiliser MPLAB :**

Nous avons un IDE de professionnel, autant ne pas le négliger, voici la marche à suivre pour éditer et compiler le programme :

1. Démarrez MPLAB : Menu Démarrer/Programmes/Microchip_MPLAB/MPLAB

2. Créez un nouveau projet : Project/New_Project…
   Je vous laisse l'initiative du nom et du répertoire, mais votre fichier .asm devra se trouver au même endroit.
   Attention n'oubliez pas de changer le mode de développement et d'ajouter comme nœud le fichier préalablement écrit.
   (soit avec notepad comme un vrai dur, soit avec fichier/Nouveau dans MPASM)



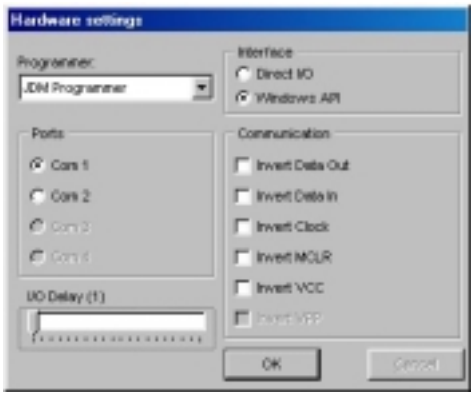
3. Project/Make et le fichier .hex est généré si tout se passe bien.

- **Utiliser ICPROG :**

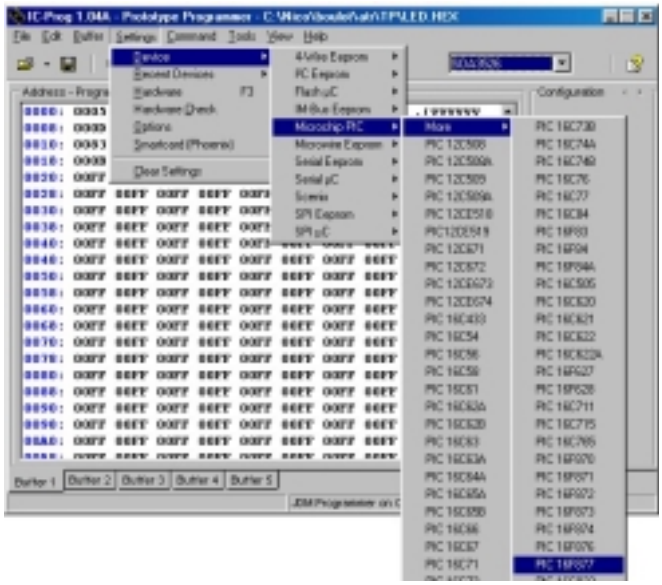
Attention, il vous faut d'abord insérer le PIC dans le programmeur :

1. Assurez-vous d'abord à fond d'avoir coupé toute source de courant susceptible d'alimenter le PIC avant de l'enlever ou de l'insérer dans un support.
2. Insérez-le maintenant dans la carte.
3. Allumez ensuite le courant.

Vous pouvez maintenant ouvrir le programme C:\PIC-01\ICPROG.EXE.



Configurez comme sur l'image.
Puis comme suit…



Puis ouvrez le fichier .hex généré précédemment.
Ensuit cliquez sur Command/Erase All
Et Command/Program All
Café !!!!

Une fois le PIC programmé remettez le sur la carte en prenant les précautions susmentionnées.

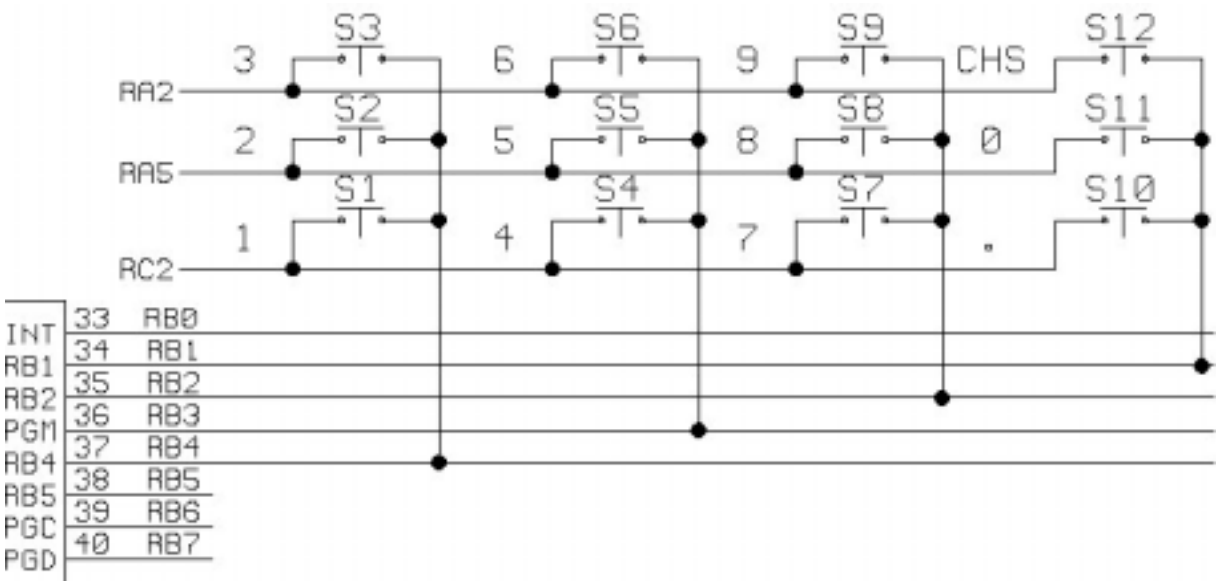
Voilà, ça fonctionne. Nous allons maintenant tenter d'allumer cette LED en appuyant sur un bouton.

Allumage d'une LED à l'aide d'un bouton :

Une question peut être ? Comment fonctionnent ces boutons ? C'est très simple :

Ils sont câblés en matrice de trois colonnes (seulement les trois colonnes de droite, les autres ne sont pas câblés de la même façon) et de quatre lignes. Les sorties RA2 RA5 et RC2 sont multiplexées (0 = select) avec les entrées RB1 RB2 RB3 RB4 qui sont mises à 1 par un pull up interne. Le fait de presser un bouton provoque une sortie multiplexée qui met le courant à 0.

Heu un petit schéma et ça ira mieux vous verrez :



En gros, si RA2 est à 0 et S3 enfoncé alors RB4 est à 0 et c'est tout ce qu'on veut savoir pour l'instant.

Nous sommes maintenant en mesure de comprendre le programme suivant :

Afin de détecter le bouton, nous utiliserons le polling : méthode qui consiste à tester en boucle le registre correspondant.

```
; Précise le PIC utilisé

        LIST P=16F877

#include "P16F877.INC"

  __CONFIG
_BODEN_ON&_CP_OFF&_WRT_ENABLE_ON&_PWRTE_ON&_WDT_OFF&_HS_OSC&_DEBUG_OFF&_CPD_OFF&_LVP_OFF


; Dédut du programme

            ORG     0x000                   ;Indique le début du programme
            GOTO    init

            ORG     0x004                   ;indique la marche à suivre en cas d'interruption
            GOTO    int

init        BSF     STATUS , RP0            ;Page d'Adresse 1 -> on a accès aux TRIS (CF M14)
            CLRF    INTCON                  ;Mise à 0 du vecteur d'interruption -> pas d'interruptions (CF M20)
            BCF     OPTION_REG,7            ;Mise à 0 du bit  7 d'option (CF M19) (The big @#¤$ truc de fouine)

            CLRF    TRISD                   ;Met le TRIS du port D à 0 -> port en mode Output
```

```
            BCF     TRISC,2             ;Met le bit 2 du port C et les bits 2 et 5 du port A en sortie
            BCF     TRISA,2
            BCF     TRISA,5
            MOVLW   B'00011110'         ;Met les bits 1 à 4 du port B en entrée
            MOVWF   TRISB

            BCF     STATUS , RP0        ;Page d'Adresse 0

            BSF     PORTC,2             ;
            BSF     PORTA,5             ;
            BCF     PORTA,2             ;Teste la troisième colonne

debut       CLRWDT                      ;clear wdt
            BTFSS   PORTB,4             ;Teste l'appui du bouton
            GOTO    KeyOn               ;S'il est appuyé, on va à KeyOn
            MOVLW   b'00000000'         ;Sinon on eteint tout.
            MOVWF   PORTD
            GOTO    debut

KeyOn       MOVLW   B'11111111'
            MOVWF   PORTD
            GOTO    debut

int         RETFIE
            END
```

Le premier qui finit a encore une fois le droit d'exhiber, hum, son programme devant tout le monde. Et gagne un BN.

Allumage d'une LED à l'aide d'une interruption (CF M22):

Comme vous savez tout sur les interruptions, inutile de vous dire que dans notre cas suffisent les bits 3 et 7 d'INTCON soit le bit associé au PORTB et le général.

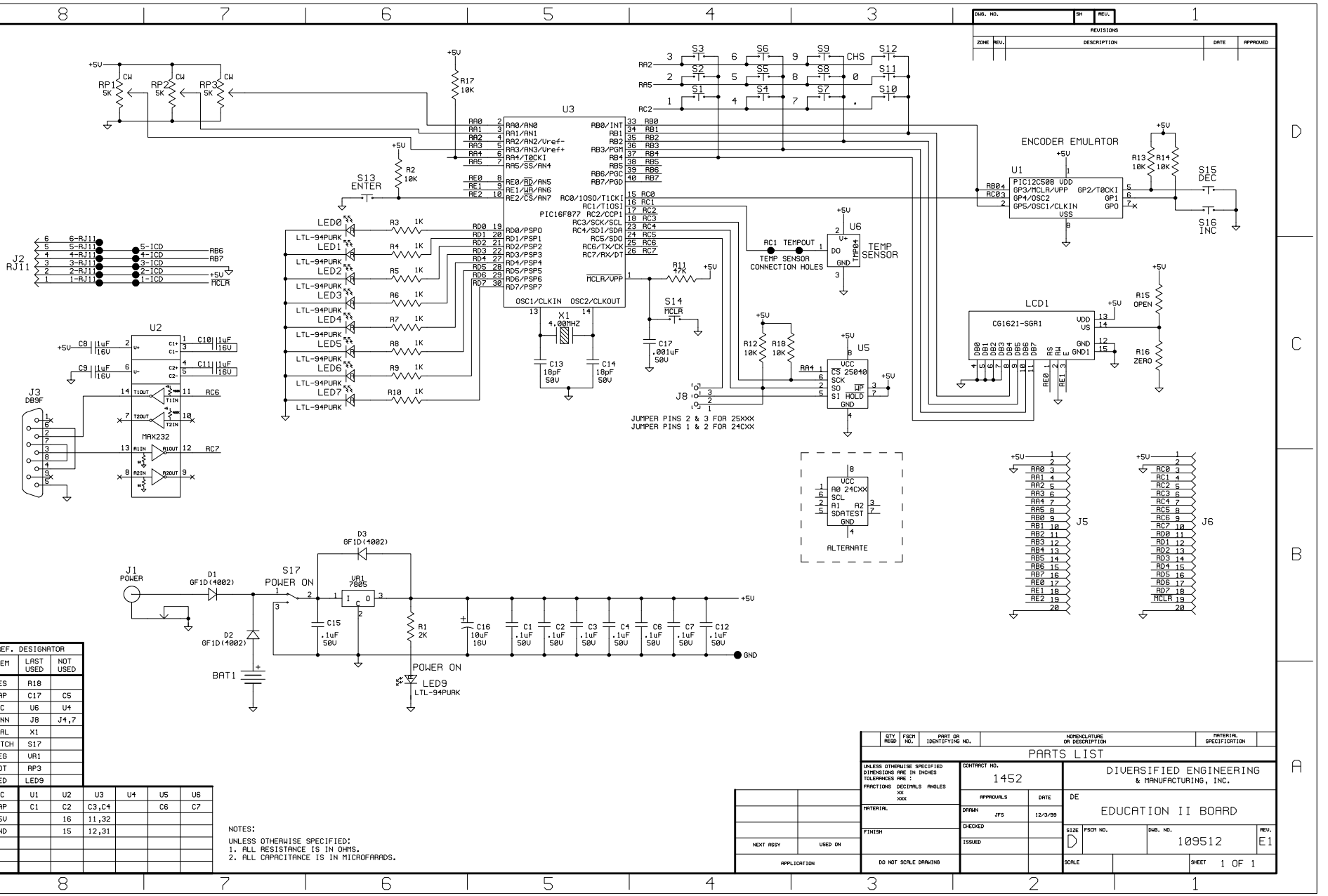
```
            ; Précise le PIC utilisé


        LIST P=16F877
        LIST r=dec,x=on,t=off

#include "P16F877.INC"

  __CONFIG
_BODEN_ON&_CP_OFF&_WRT_ENABLE_ON&_PWRTE_ON&_WDT_OFF&_HS_OSC&_DEBUG_OFF&_CPD_OFF&_LVP_OFF


; Variables

            cblock  H'20'
            W_TEMP                      ;Nous aurons besoin de variables afin de
            STATUS_TEMP                 ;stocker les registres susceptibles d'être modifiés lors de l'interruption
            endc

; Dédut du programme

            org     0000h
            goto    init

            ORG     0004h
            goto    int

init        BSF     STATUS , RP0        ;Page d'Adresse 1 -> on a accès aux TRIS

            BCF     OPTION_REG,7        ;Mise à 0 du bit  7 d'option (CF M19)

            CLRF    TRISD               ;Met le TRIS du port D à 0 -> port en mode Output
            BCF     TRISC,2             ;Met le bit 2 du port C et les bits 2 et 5 du port A en sortie
```

```
        BCF     TRISA,2
        BCF     TRISA,5
        movlw   B'00011110'             ;Met les bits 1 à 4 du port B en entrée
        movwf   TRISB


        BCF     STATUS , RP0            ;Page d'Adresse 0

        MOVLW   b'10001000'
        MOVWF   INTCON

boucle  clrwdt
        MOVLW   b'10101010'
        MOVWF   PORTD                   ;On allume 4 des LEDS
        BSF     PORTC,2                 ;
        BSF     PORTA,5                 ;
        BCF     PORTA,2                 ;Teste la troisième colonne
        goto boucle



int     clrwdt
        BTFSS   INTCON,0                ;On teste la provenance de l'interruption
        CLRF    INTCON                  ;on inhibe les IT
        movwf   W_TEMP
        swapf   STATUS,W                ;On sauve tout dans la RAM
        movwf   STATUS_TEMP             ;On peut le faire aussi ans la pile mais pas là :)

        MOVLW   b'01010101'
        MOVWF   PORTD
boucle2 clrwdt                          ;A vous de comprendre
        BTFSS   PORTB,4                 ;vous êtes pas des moules
        goto    boucle2

        swapf   STATUS_TEMP,W ;voilà c'est fini
        movwf   STATUS          ;on retablit tout même Z
        swapf   W_TEMP,F        ;
        swapf   W_TEMP,W        ;
        MOVLW   b'10001000'
        MOVWF   INTCON
fin     RETFIE                  ;On finit l'IT

        END
```

Maintenant vous savez tout et il vous reste un peu de temps pour mettre vos connaissances à profit :

Celui qui écrit le programme le plus original et le plus beau remporte le paquet de BN !!!

Par exemple : un programme pourra afficher différentes séquences sur les LEDS selon le bouton enfoncé.



REF. DESIGNATOR		
ITEM	LAST USED	NOT USED
RES	R18	
CAP	C17	C5
IC	U6	U4
CONN	J8	J4,7
XTAL	X1	
SWITCH	S17	
REG	VR1	
POT	RP3	
LED	LED9	

IC	U1	U2	U3	U4	U5	U6
CAP	C1	C2	C3,C4		C6	C7
+5V		16	11,32			
GND		15	12,31			

NOTES:
UNLESS OTHERWISE SPECIFIED:
1. ALL RESISTANCE IS IN OHMS.
2. ALL CAPACITANCE IS IN MICROFARADS.

QTY REQD	FSCM NO.	PART OR IDENTIFYING NO.	NOMENCLATURE OR DESCRIPTION	MATERIAL SPECIFICATION

PARTS LIST

NEXT ASSY	USED ON
APPLICATION	

UNLESS OTHERWISE SPECIFIED DIMENSIONS ARE IN INCHES
TOLERANCES ARE :
FRACTIONS DECIMALS ANGLES
XX
XXX
MATERIAL
FINISH
DO NOT SCALE DRAWING

CONTRACT NO. 1452

APPROVALS	DATE
DRAWN JFS	12/3/99
CHECKED	
ISSUED	

DIVERSIFIED ENGINEERING & MANUFACTURING, INC.
DE
EDUCATION II BOARD

SIZE	FSCM NO.	DWG. NO.	REV.
D		109512	E1

SCALE
SHEET 1 OF 1

DWG. NO.	SH	REV.

REVISIONS

ZONE	REV.	DESCRIPTION	DATE	APPROVED