PIC LAB PRO / COURS DÉBUTANT / 01

Comprendre les PIC du bit au premier montage

PIC12F683, PIC16F877A et PIC18F4520 • Édition du 18 octobre 2026

Tu peux apprendre les PIC en partant d’un montage très simple. Le but de ce cours est de comprendre **ce que tu demandes au composant**, puis de reconnaître le registre qui porte cette demande.

Le fil conducteur
Choisir la fonction de la broche, choisir sa direction, puis lire ou commander son niveau. Une LED, un bouton et un capteur permettent de pratiquer ces trois idées.

Composant du cours	Ce qui le rend différent
PIC12F683	Petit boîtier de 8 broches ; GPIO et TRISIO ; oscillateur interne disponible.
PIC16F877A	Boîtier PDIP de 40 broches ; PORTA à PORTE et TRIS ; horloge externe pour nos exemples.
PIC18F4520	Boîtier PDIP de 40 broches ; PORT et LAT séparés ; ADCON2 pour le réglage ADC.

Le parcours

Pages	Ce que tu vas apprendre
2 à 4	Comprendre le programme, les bits et les registres.
5 à 7	Faire clignoter la même LED sur les trois PIC.
8	Lire un bouton et commander une LED.
9 à 10	Distinguer numérique et analogique, puis découvrir l’ADC.
11	Comprendre TRIG, ECHO et Timer1 avec le HC-SR04.
12 à 13	Faire les exercices, vérifier et consulter les sources.

Lis d’abord les pages 2 à 5 et fais le montage PIC12 dans le laboratoire. Compare ensuite avec les pages PIC16 et PIC18. Tu n’as pas à tout mémoriser en une fois.

PIC LAB PRO / COURS DÉBUTANT / 02

Un PIC : qu’est-ce qui travaille ?

Objectif : distinguer le programme, l’horloge, les registres et les broches.

Un microcontrôleur rassemble un processeur, de la mémoire et des périphériques. Ton programme est enregistré dans la mémoire programme. Le processeur suit ses instructions et agit sur les périphériques à travers des registres.

Élément	Son rôle dans ton montage
Programme	Décrit les actions : allumer, attendre, lire, comparer.
Horloge	Donne le rythme de fonctionnement du composant.
Registre	Contient des bits de configuration, des commandes ou des données.
Broche	Relie le PIC à une LED, un bouton ou un autre composant.
Périphérique	Effectue une tâche spécialisée : ADC, Timer1, communication...

Au démarrage, puis dans la boucle

```
void main(void) {
    initialiser();  // Préparer les fonctions et les broches
    while (1) {    // Recommencer tant que le PIC fonctionne
        // Lire les entrées et commander les sorties
    }
}
```

L’initialisation se fait normalement une fois. La boucle répète les actions. Si tu utilises des délais bloquants, le programme attend à cet endroit ; les autres actions de cette boucle attendent aussi.

L’horloge et les délais

Dans ce cours, les exemples utilisent **4 MHz**. Le PIC12 et le PIC18 emploient leur oscillateur interne ; le PIC16F877A utilise un quartz externe de 4 MHz. La valeur déclarée au compilateur doit correspondre à la fréquence réelle.

```
#define _XTAL_FREQ 4000000UL
__delay_ms(500);  // Demande environ une demi-seconde
```

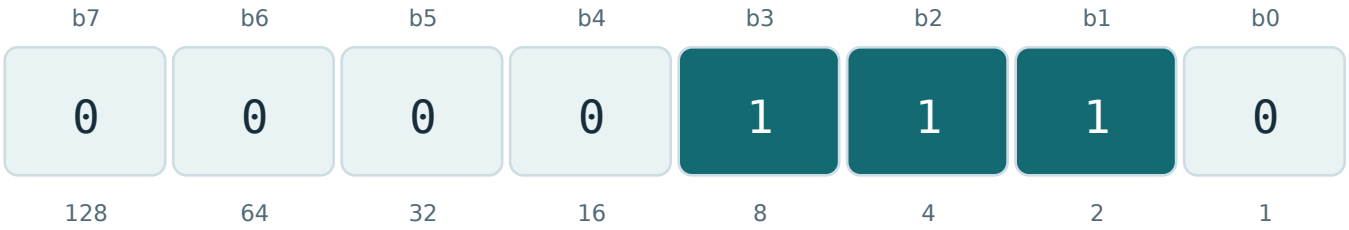
À ne pas confondre
_XTAL_FREQ aide le compilateur à calculer les délais. Cette ligne ne règle pas l’oscillateur. Les trois composants étudiés ont un processeur 8 bits : « PIC12 » ne signifie pas « processeur 12 bits ».

PIC LAB PRO / COURS DÉBUTANT / 03

Un registre : huit cases, pas huit broches

Objectif : lire un octet et reconnaître un champ de configuration.

Un registre 8 bits contient huit positions, de **b7 à b0**. Chaque position contient 0 ou 1. La position b0 se trouve à droite. Pour convertir un nombre binaire, additionne les poids des positions à 1.



Ici : 8 + 4 + 2 = **14**. Les autres positions valent zéro et n’ajoutent rien.

Binaire	Décimal	Hexadécimal
00000000	0	0x00
00000001	1	0x01
00001000	8	0x08
00001110	14	0x0E
11111111	255	0xFF

Le nombre reste le même ; son sens dépend du registre

La valeur 0x0E représente toujours le nombre 14. Mais écrire ce nombre dans un registre de direction ou dans une configuration ADC produit des effets différents.

Bit individuel ou champ ?
Dans TRIS, plusieurs bits commandent individuellement la direction des broches. En revanche, PCFG ou CHS se lisent comme un code complet : plusieurs bits forment une sélection. N’associe pas automatiquement « b2 » à « AN2 ».

Lire les préfixes dans le code

```
0b00001110  // Notation binaire
0x0E        // Notation hexadécimale
14          // Notation décimale
```

Mini-exercice : quels poids faut-il additionner pour lire 00000101 ? Quelle est sa valeur décimale ? Garde ta réponse pour la page de correction.

PIC LAB PRO / COURS DÉBUTANT / 04

Que faut-il régler, et que laisser tranquille ?

Objectif : ne plus chercher à mettre tous les registres à zéro.

Pour une première LED, commence par libérer la fonction numérique, puis choisis une sortie et enfin son niveau. Le comparateur et l'ADC sont des fonctions différentes : désactiver l'un ne désactive pas forcément l'autre.

Question	PIC12F683	PIC16F877A	PIC18F4520
Broches numériques	ANSEL	ADCON1 / PCFG	ADCON1 / PCFG
Comparateur	CMCON0	CMCON	CMCON
Direction	TRISIO	TRISA à TRISE	TRISA à TRISE
Niveau de sortie	Écrire GPIO	Écrire PORT	Écrire LAT de préférence
Lire une broche	Lire GPIO	Lire PORT	Lire PORT

Un zéro ne veut pas dire la même chose partout

Emplacement	Que signifie 0 ?
Bit TRIS de direction	La broche fonctionne en sortie.
Bit ANS de sélection analogique	La fonction d'entrée analogique concernée est désactivée.
Bit GPIO / LAT de sortie	Le verrou demande le niveau bas.
ADCON0.ADON	Le convertisseur ADC est arrêté.
Champ CM du comparateur	Il faut lire la table : CM=000 et CM=111 ne libèrent pas les broches de la même façon.

Les broches et registres inutilisés

Dans nos montages préparés, les GPIO libres et disponibles sont placées en sorties basses, sans autre composant connecté. GP3 du PIC12 reste une entrée. Sur une carte déjà câblée, vérifie les connexions avant de changer une direction.

Une autre stratégie possible est une entrée avec un niveau défini par un tirage. Une entrée numérique laissée flottante peut lire un état imprévisible. Ce n'est pas le mot « inutilisée » qui détermine à lui seul la bonne direction.

Dans PIC Lab Pro
Les positions grises sont préparées, réservées ou en lecture seule. Un x indique une donnée ou un état non déterminé. Ne modifie pas un résultat ADC inutilisé simplement pour faire disparaître ses x.

PIC LAB PRO / COURS DÉBUTANT / 05

PIC12F683 : ma première LED

Montage : GP0, broche 7, vers une LED et sa résistance 330 Ω en série, puis GND.

Profil : 5 V, oscillateur interne 4 MHz, MCLR activé. VDD est la broche 1, VSS la broche 8 ; MCLR, broche 4, est maintenu à +5 V par 10 kΩ. Ajoute le découplage près du PIC. Les autres GPIO disponibles restent libres et à zéro.

```
#include <xc.h>
#define _XTAL_FREQ 4000000UL
#pragma config FOSC = INTOSCIO
#pragma config WDTE = OFF
#pragma config MCLRE = ON

#define LED GPIObits.GP0

void initialiser(void) {
    OSCCON = 0x61; // Oscillateur interne, 4 MHz
    ANSEL = 0x00; // Fonction analogique des broches
    CMCON0 = 0x07; // Comparateur
    ADCON0 = 0x00; // ADC inutile pour ce projet
    GPIO = 0x00;
    TRISIO = 0x08;
}

void main(void) {
    initialiser();
    while (1) {
        LED = 1; // Allumer
        __delay_ms(500);
        LED = 0; // Eteindre
        __delay_ms(500);
    }
}
```

ANSEL=0 enlève les sélections analogiques. **CMCON0=7** libère les fonctions comparateur. **TRISIO=0x08** garde GP3 en entrée et les GPIO disponibles en sorties. Le programme change ensuite GP0, sans refaire l’initialisation.

À faire dans le labo
Mode débutant → Ma première LED → PIC12F683 → Essayer → Démarrer. Puis change la durée à 1 seconde et observe. Une période complète compte un état allumé et un état éteint.

PIC LAB PRO / COURS DÉBUTANT / 06

PIC16F877A : le même raisonnement

Montage : RA0, broche 2, vers la résistance 330 Ω et la LED, puis GND.

Cet exemple demande un quartz externe de 4 MHz entre OSC1/OSC2, broches 13/14, avec sa charge adaptée. VDD : 11 et 32 ; VSS : 12 et 31. MCLR : broche 1 via 10 kΩ vers +5 V. Ne déclare pas 4 MHz si le quartz réel est à 10 ou 16 MHz.

```
#include <xc.h>
#define _XTAL_FREQ 4000000UL
#pragma config FOSC = HS
#pragma config WDTE = OFF
#pragma config LVP = OFF

#define LED PORTAbits.RA0

void initialiser(void) {
    ADCON1 = 0x06; // Fonction analogique des broches
    CMCON = 0x07; // Comparateur
    ADCON0 = 0x00; // ADC inutile pour ce projet
    PORTA = 0x00;
    PORTB = 0x00;
    PORTC = 0x00;
    PORTD = 0x00;
    PORTE = 0x00;
    TRISA = 0x00;
    TRISB = 0x00;
    TRISC = 0x00;
    TRISD = 0x00;
    TRISE = 0x00;
}

void main(void) {
    initialiser();
    while (1) {
        LED = 1; // Allumer
        __delay_ms(500);
        LED = 0; // Eteindre
        __delay_ms(500);
    }
}
```

ADCON1=0x06 choisit ici le numérique pour les entrées AN. **TRISA bit 0=0** autorise RA0 en sortie. La commande passe par **PORTA** : ce PIC ne possède pas les registres LAT du PIC18. Les autres ports sont mis à zéro car cet exemple suppose qu’ils sont libres.

PIC LAB PRO / COURS DÉBUTANT / 07

PIC18F4520 : comprendre PORT et LAT

Montage : RA0, broche 2, vers la résistance 330 Ω et la LED, puis GND.

Profil : 5 V et oscillateur interne 4 MHz. VDD : broches 11/32 ; VSS : 12/31 ; MCLR : broche 1 via 10 kΩ vers +5 V. Le code suppose les autres GPIO disponibles libres. Le profil d'horloge libère ici RA6 et RA7 comme GPIO.

```
#include <xc.h>
#define _XTAL_FREQ 4000000UL
#pragma config OSC = INTIO67
#pragma config WDT = OFF
#pragma config MCLRE = ON
#pragma config LVP = OFF
#pragma config PBADEN = OFF
#pragma config XINST = OFF

#define LED LATAbits.LATA0

void initialiser(void) {
    OSCCON = 0x62; // Oscillateur interne, 4 MHz
    ADCON1 = 0x0F; // Fonction analogique des broches
    CMCON = 0x07; // Comparateur
    ADCON0 = 0x00; // ADC inutile pour ce projet
    LATA = 0x00;
    LATB = 0x00;
    LATC = 0x00;
    LATD = 0x00;
    LATE = 0x00;
    TRISA = 0x00;
    TRISB = 0x00;
    TRISC = 0x00;
    TRISD = 0x00;
    TRISE = 0x00;
}

void main(void) {
    initialiser();
    while (1) {
        LED = 1; // Allumer
        __delay_ms(500);
        LED = 0; // Eteindre
        __delay_ms(500);
    }
}
```

ADCON1=0x0F choisit le numérique. **TRISA** règle toujours la direction. **LATA** contient le verrou de sortie ; **PORTA** sert à lire les niveaux des broches. Pour commander une sortie PIC18, LAT évite les ambiguïtés d'une lecture du port lors d'une modification.

PIC LAB PRO / COURS DÉBUTANT / 08

Un bouton : recevoir avant de commander

Objectif : lire un niveau numérique, sans utiliser l’ADC.

Le bouton relie l’entrée à +5 V quand tu appuies. Une résistance pull-down de 10 kΩ la relie à GND lorsqu’il est relâché. Le programme lit cette entrée, puis commande la LED.

Fonction	PIC12F683	PIC16F877A	PIC18F4520
LED en sortie	GP0, broche 7	RA0, broche 2	RA0, broche 2
Bouton en entrée	GP1, broche 6	RA1, broche 3	RA1, broche 3
Direction utilisée ici	TRISIO=0x0A	TRISA=0x02	TRISA=0x02
Lecture du bouton	GPIO.GP1	PORTA.RA1	PORTA.RA1

Les deux directions à comprendre

Le bit de direction de la LED vaut **0**. Le bit de direction du bouton vaut **1**. Les valeurs ci-dessus supposent les autres broches du port préparées en sorties basses, avec GP3 toujours en entrée sur PIC12.

Après l’initialisation numérique, dans la boucle

```
// PIC12F683
GPIObits.GP0 = GPIObits.GP1 ? 1 : 0;

// PIC16F877A
PORTAbits.RA0 = PORTAbits.RA1 ? 1 : 0;

// PIC18F4520
LATAbits.LATA0 = PORTAbits.RA1 ? 1 : 0;
```

L’expression « condition ? 1 : 0 » signifie : si la condition est vraie, utiliser 1 ; sinon utiliser 0. Tu peux écrire le même comportement avec un if et un else.

À faire dans le labo
Choisis Un bouton, une LED, puis démarre le scénario. Appuie et relâche. Mets ensuite en pause : l’entrée peut encore changer, mais la logique attend avant d’actualiser la LED. Une étape effectue une nouvelle lecture.

Un bouton réel peut rebondir. La lecture périodique de l’exemple n’est pas un anti-rebond complet. Le bouton du laboratoire est idéal.

PIC LAB PRO / COURS DÉBUTANT / 09

Numérique ou analogique ?

Objectif : distinguer un état logique et une mesure de tension.

Lecture numérique	Conversion analogique
Répond à une question : 0 ou 1 ?	Mesure une tension dans une plage de référence.
Exemple : bouton relâché ou appuyé.	Exemple : position d’un potentiomètre.
GPIO / PORT donne le niveau logique.	L’ADC produit un nombre sur 10 bits.
Ne demande pas de conversion ADC.	Demande une configuration et un lancement.

Un potentiomètre sur AN0

Les bornes extérieures sont reliées à VDD et GND. Le curseur fournit une tension variable à AN0. La broche correspondante doit être une entrée et sa fonction analogique doit être autorisée.

Tension, avec VREF-=0 V et VREF+=5 V	Code idéal du modèle
0 V	0
1,25 V	256
2,5 V	512
5 V	1023

Les trois ADC étudiés ont une résolution de **10 bits** : il existe 1 024 codes, numérotés de 0 à 1 023. Le modèle idéal utilise des pas de quantification ; ce n’est pas la précision garantie d’un capteur réel.

Pourquoi deux registres pour le résultat ?

Dix bits ne tiennent pas dans un seul registre de huit bits. Le résultat est réparti entre **ADRESH** et **ADRESL**. ADFM choisit son placement.

Valeur 512	ADRESH	ADRESL
Justification à droite, ADFM=1	0x02	0x00
Justification à gauche, ADFM=0	0x80	0x00

À observer
Tourner le potentiomètre ne remplace pas une mesure déjà terminée. Pour obtenir un nouveau résultat, il faut lancer une nouvelle conversion ou un scénario de lectures répétées.

PIC LAB PRO / COURS DÉBUTANT / 10

L’ADC : ce qui change entre les trois PIC

Objectif : suivre une méthode commune avec des registres différents.

Réglage	PIC12F683	PIC16F877A	PIC18F4520
Autoriser l’analogique	ANSEL	ADCON1 / PCFG	ADCON1 / PCFG
Choisir le canal	ADCON0 / CHS	ADCON0 / CHS	ADCON0 / CHS
Activer le convertisseur	ADCON0 / ADON	ADCON0 / ADON	ADCON0 / ADON
Choisir ADFM	ADCON0	ADCON1	ADCON2
Démarrer GO/DONE	ADCON0 bit 1	ADCON0 bit 2	ADCON0 bit 1

Configurations AN0 proposées dans le labo à 4 MHz

```
// PIC12F683 : GP0 en entrée, comparateur désactivé
TRISIO = 0x09; ANSEL = 0x11; CMCON0 = 0x07; ADCON0 = 0x81;

// PIC16F877A : RA0 en entrée, AN0 seul autorisé
TRISA = 0x01; CMCON = 0x07; ADCON1 = 0x8E; ADCON0 = 0x81;

// PIC18F4520 : RA0 en entrée, AN0 seul autorisé
TRISA = 0x01; CMCON = 0x07; ADCON1 = 0x0E;
ADCON2 = 0xAA; ADCON0 = 0x01;
```

Ces lignes sont des initialisations ciblées à intégrer dans le profil d’horloge et d’alimentation du projet. Elles ne remplacent pas tout le programme. PCFG=1110 sélectionne AN0 dans les configurations indiquées ; PCFG n’est pas un masque « une broche par bit ».

L’ordre à comprendre

Choisir l’entrée et le canal ; laisser la tension être acquise ; lancer GO ; attendre la fin ; lire les deux octets. ADON=0 signifie que le convertisseur est arrêté. GO revient à zéro quand la conversion se termine.

```
// Si ADFM=1 : reconstruire le résultat à droite
mesure = ((unsigned int)ADRESH << 8) | ADRESL;
```

Les réglages d’horloge et d’acquisition comptent aussi. Ne reprends pas ADCON2 sur le PIC12 ou le PIC16 : ce registre n’y existe pas. Le PIC16 utilise ses propres positions ADCS et sa propre table PCFG.

PIC LAB PRO / COURS DÉBUTANT / 11

HC-SR04 : mesurer un temps, pas une tension

Objectif : comprendre le rôle d'une sortie, d'une entrée et d'un chronomètre.

Le capteur classique 5 V possède VCC, GND, TRIG et ECHO. Le PIC envoie une courte impulsion sur TRIG. Le module renvoie sur ECHO une impulsion numérique dont la largeur représente une distance.

Signal	Direction côté PIC	Dans nos projets
TRIG	Sortie	GP2 sur PIC12 ; RA2 sur PIC16/PIC18
ECHO	Entrée	GP1 sur PIC12 ; RA1 sur PIC16/PIC18
LED de proximité	Sortie	GP0 sur PIC12 ; RA0 sur PIC16/PIC18

Trois moments à observer

• Mettre TRIG à 1 pendant au moins 10 microsecondes, puis le remettre à 0.

• Quand ECHO passe à 1, commencer à compter le temps.

• Quand ECHO revient à 0, arrêter le compteur et interpréter la durée.

L'approximation utilisée par le projet est : **distance en cm = durée ECHO en microsecondes / 58**. Ainsi 5 800 µs représentent environ 100 cm. L'ADC ne sert pas à cette mesure.

Pourquoi Timer1 ?

Timer1 est un compteur 16 bits. Avec Fosc=4 MHz, une horloge interne Fosc/4 et un préscaler 1:1, un tick vaut 1 µs. Le compteur peut être lu dans TMR1H et TMR1L. Les champs de T1CON ne sont pas tous identiques sur les trois PIC.

```
// Profil proposé, horloge 4 MHz : 1 tick = 1 µs
T1CON = 0x00;          // Timer arrêté, configuration interne
TMR1H = 0; TMR1L = 0; // Remettre le compteur à zéro
T1CONbits.TMR1ON = 1; // Commencer à compter
```

Pourquoi un délai maximal ?
Sans écho, une attente infinie bloquerait le programme. Le projet impose un timeout de 30 ms. Son C prévoit aussi une attente supérieure à 60 ms entre les mesures. Le modèle du labo est idéal et les animations sont ralenties.

Le programme complet se trouve dans examples/PIC.../hcsr04.c de l'archive PIC Lab Pro. Les variantes sont propres à chaque composant.

PIC LAB PRO / COURS DÉBUTANT / 12

À toi de faire : cinq petits exercices

Objectif : expliquer le résultat avant de cliquer sur les bits.

1. Lire un nombre

Convertis 00000101 en décimal, puis en hexadécimal. Donne les poids qui participent au calcul.

2. Réparer une LED

Dans le projet LED du PIC12, le bit 0 de TRISIO vaut 1. La LED ne réagit pas comme prévu. Quel bit faut-il changer, dans quel registre et vers quelle valeur ? Faut-il modifier ADRESH ?

3. Choisir les directions

Tu veux une LED sur GP0 et un bouton sur GP1. Les autres GPIO disponibles sont libres et préparées en sorties basses. GP3 reste une entrée. Quelle valeur binaire et hexadécimale correspond à TRISIO ?

4. Comparer PIC16 et PIC18

Sur le PIC18F4520, tu lis un bouton sur RA1 et commandes une LED sur RA0. Quel registre lis-tu pour le bouton ? Quel registre utilises-tu de préférence pour écrire la sortie ? Pourquoi ne pas copier LATA dans le PIC16F877A ?

5. Interpréter une mesure

Avec une référence 0-5 V, le potentiomètre est à 2,5 V. Quel code ADC idéal attends-tu ? Avec le HC-SR04, ECHO reste haut pendant 2 900 µs. Quelle distance approximative attends-tu ? Les deux mesures utilisent-elles le même périphérique ?

Ta méthode dans le laboratoire
Choisis le projet, observe son câblage, lis les rôles des bits, modifie un réglage, explique l’effet, puis seulement lance le scénario. Le bouton « À toi de régler » crée une erreur de direction pour t’entraîner.

Pour commencer aujourd’hui, fais seulement les exercices 1 et 2. Tu peux poursuivre quand tu sais expliquer leur correction avec tes propres mots.

PIC LAB PRO / COURS DÉBUTANT / 13

Corrections et repères à garder

Des réponses courtes, puis les documents de référence.

Exercice	Réponse attendue
1	4 + 1 = 5 en décimal, soit 0x05.
2	TRISIO bit 0 doit passer à 0 pour une sortie. ADRESH ne sert pas à la LED.
3	TRISIO = 00001010 = 0x0A : GP1 et GP3 en entrée, les autres GPIO disponibles en sortie.
4	Lire PORTA.RA1. Écrire LATA pour la sortie PIC18. Le PIC16F877A n’a pas de registre LATA.
5	ADC idéal : 512. Ultrason : 2 900 / 58 ≈ 50 cm. Le premier utilise l’ADC, le second le chronométrage de l’impulsion.

Trois questions avant de changer un bit

• Quelle fonction a cette broche dans mon montage ?

• Doit-elle recevoir ou envoyer un signal ?

• Est-ce une commande à écrire ou une valeur que le matériel produit ?

Sources techniques

Microchip PIC12F683 - DS41211D : GPIO, oscillateur, comparateur, ADC et Timer1.

Microchip PIC16F87XA - DS39582C : brochage PIC16F877A, ports, ADC et Timer1.

Microchip PIC18F2420/2520/4420/4520 - DS39631E : PORT/LAT, ADC et Timer1.

HC-SR04 - fiche ElecFreaks distribuée par SparkFun : TRIG, ECHO et formule us/58.

Le cours, le labo et le véritable PIC

Le laboratoire joue des scénarios logiques JavaScript. Il ne compile ni n’exécute le C. Les exemples de l’archive ont reçu un contrôle de syntaxe C générique ; cela ne valide pas les pragmas XC8, le DFP, les temporisations réelles ou le montage physique. Vérifie le composant choisi et les noms de configuration dans MPLAB X.

Le cours vise les trois références indiquées et les profils de 5 V / 4 MHz. Il n’étend pas automatiquement ces réglages à tous les PIC12, PIC16 ou PIC18. Le bruit, les rebonds mécaniques et les phénomènes électriques réels ne sont pas entièrement reproduits par le modèle.