

PRÉSENTATION DES MICROCONTRÔLEURS

Un microcontrôleur est un petit ordinateur complet intégré sur une seule [puce électronique](#).

Il regroupe les éléments essentiels d'un système informatique : un processeur (cerveau), des mémoires (pour stocker le programme et les données) et des interfaces d'entrées-sorties (pour communiquer avec des capteurs ou des moteurs).

Rôle et Fonctionnement

- **Traitement embarqué** : Il sert à automatiser des tâches précises et répétitives au sein d'un appareil.
- **Interaction physique** : Il lit des informations venues du monde réel (température, pression, appui sur un bouton) et agit en conséquence (allumer une lumière, faire tourner un moteur).
- **Faible consommation** : Contrairement au microprocesseur d'un ordinateur personnel, il consomme peu d'énergie et coûte peu.

Exemples d'utilisation

- **Objets du quotidien** : Machines à laver, micro-ondes, thermostats, brosses à dents électriques.
- **Prototypage** : Les célèbres cartes [Arduino](#) ou les puces STM32 reposent sur un microcontrôleur.

Différence avec un microprocesseur

La principale différence entre un microprocesseur et un microcontrôleur est que le premier ne contient que l'unité de calcul (le CPU), tandis que le second intègre sur une seule et même puce le CPU, la mémoire et les périphériques nécessaires pour fonctionner de manière autonome.

Quels sont les composants principaux d'un microcontrôleur PIC ?

D'après la structure matérielle et les architectures (notamment pour le PIC16F84A), les **composants principaux** d'un microcontrôleur PIC sont :

1. Le processeur et l'unité de calcul

- **Microprocesseur / UAL (Unité d'Arithmétique et de Logique)** : effectue les calculs et le traitement logiques et arithmétiques des données.
- **Registres associés** :
 - **Registre d'instructions et de décodage** : reçoit l'instruction pointée, l'analyse et la décode afin de déterminer l'opération à effectuer et piloter l'unité de calcul.
 - **Registre de travail (W Reg)** : enregistre temporairement le résultat issu des calculs de l'UAL.
 - **Registre d'état (Status Register)** : enregistre le dernier état du calculateur (indicateurs et résultats d'exécution).

- **Compteur de programme (PC)** : pointe vers la prochaine instruction à exécuter.
- **Compteur de programme (*Program Counter*)** : sert de pointeur pour sélectionner l'instruction à exécuter au sein de la mémoire programme.
- **Unité Arithmétique et Logique (UAL / ALU)** : exécute les opérations logiques et arithmétiques du programme.

2. Les mémoires et les bus

- **ROM / Flash ou EEPROM** : mémoire programme servant à stocker les instructions du code, accessibles en lecture seule durant l'exécution.
- **RAM** : mémoire vive volatile de données pour l'enregistrement temporaire des données et des variables de traitement lors du fonctionnement.
- **EEPROM** : mémoire de données non volatile qui permet d'inscrire et de conserver des données après une coupure d'alimentation.
- **Bus internes et multiplexeurs** : l'ensemble des bus (données, adresses, programme) et des multiplexeurs permettent la circulation et le routage des informations entre l'UAL, la mémoire et les registres.

3. Les ports d'Entrée/Sortie (E/S)

- **Interfaces parallèles (Port A, Port B, Port C...)** : permettent l'échange de signaux numériques avec l'extérieur (boutons, LED, capteurs, moteurs, relais...).
- **Registres de configuration associés (TRISX)** : permettent d'orienter chaque broche du port en entrée ou en sortie.

4. L'horloge et le contrôle du système

- **Horloge / Oscillateur** : connecté aux broches `osc1` et `osc2` pour rythmer les cycles d'exécution.
- **MCLR (*Master Clear*)** : entrée d'initialisation / réinitialisation (reset) du microcontrôleur.

5. Les périphériques intégrés

- **Timers / Compteurs** (par exemple le timer `TMR0`) : permettent le comptage d'impulsions via `RA4/TMR0CLK` ou le minutage.
- **Chien de garde (*Watchdog*)** : circuit de sécurité et antiplantage réinitialisant le PIC en cas de blocage.
- **Convertisseur Analogique-Numérique (CAN/ADC)** (*présent sur certains modèles*) : permettant de convertir des signaux analogiques en signaux numériques.

Comment se Déroule un cycle d'exécution d'une instruction

L'exécution d'une instruction s'effectue séquentiellement selon le cycle suivant :

1. Le **compteur de programme** pointe l'instruction à exécuter dans la **mémoire programme**.
2. L'instruction est transférée vers le **registre d'instructions**.
3. Le **registre de décodage** analyse l'instruction et transmet la commande à l'**unité de calcul**.
4. L'**UAL** réalise l'opération.
5. Le résultat est stocké dans le **registre de travail (W)**, puis orienté via les bus internes vers la **RAM** ou les registres cibles, tout en mettant à jour le **registre d'état**.

Comment configurer les registres TRISA et TRISB sur un microcontrôleur PIC ?

Pour configurer les registres **TRISA** et **TRISB** d'un microcontrôleur PIC, il faut définir l'orientation de chaque broche des ports respectifs (Port A et Port B) en **Entrée (E)** ou en **Sortie (S)**.

1. Rôle et principe de base

Les registres **TRISX** (TRISA pour le Port A, TRISB pour le Port B) sont les **registres de configuration des ports en Entrée/Sortie** :

- **1** = la broche est configurée en **Entrée (E) (Input)**.
- **0** = la broche est configurée en **Sortie (S) (Output)**.

2. Organisation des bits

Chaque bit du registre correspond à une broche du port :

- **TRISA** contrôle les broches $RA4, RA3, RA2, RA1, RA0$.
- **TRISB** contrôle les broches $RB7$ à $RB0$.

Exemple pour TRISA :

Si vous souhaitez configurer **RA4** et **RA3** en Entrée (1), **RA2** et **RA1** en Sortie (0), et **RA0** en Entrée (1) :

Représentation binaire : **0b11001**.

Exemples d'application :

- **Exemple 1 – Configuration du PORTA sur PIC 16F84A** : Pour affecter les broches du PORTA de la manière suivante :
 - RA4 = Entrée (1) (exemple bouton poussoir)
 - RA3 = Sortie (0) (exemple diode LED)
 - RA2 = Entrée (1) (exemple interrupteur)
 - RA1 = Sortie (0) (exemple moteur)
 - RA0 = Sortie (0) (exemple relais)

Le mot binaire à charger dans le registre **TRISA** est **00010100** (soit **14** en hexadécimal ou **20** en décimal).

- **Exemple 2 – Configuration du PORTB sur PIC 16F628** : Pour affecter les broches $RB7$ à $RB0$ avec le sens (E, S, S, S, E, E, S, S) :

- RB7 = 1, RB6 = 0, RB5 = 0, RB4 = 0, RB3 = 1, RB2 = 1, RB1 = 0, RB0 = 0

Le mot binaire du registre **TRISB** est 10001100 (soit 8C en hexadécimal ou 140 en décimal).

3. Formats d'écriture en programmation (ex. MikroC)

Vous pouvez affecter la valeur au registre sous plusieurs bases numériques :

- **Binaire** : TRISA = 0b11001; **OU** TRISB = 0b10111010;
- **Hexadécimal** : TRISA = 0x19; **OU** TRISB = 0xBA;
- **Décimal** : TRISA = 25; **OU** TRISB = 186;
- **Octal** : TRISA = 031; **OU** TRISB = 0272;

4. Configuration d'une seule broche

En langage MikroC Pro, il est également possible de modifier l'état d'un seul bit sans affecter le reste du registre :

- Pour mettre la broche **RB0** en entrée : TRISB.B0 = 1 ; TRISB.F0 = 1 ; TRISB.RB0 = 1 ; TRISB0_bit = 1 ;
- Pour mettre la broche **RB0** en sortie : TRISB.B0 = 0; TRISB.F0 = 0 ; TRISB.RB0 = 0 ; TRISB0_bit = 0 ;