# Guide des Opérations en Assembleur pour le PIC16F1789

## 1. Introduction au Langage Assembleur PIC

Le langage assembleur est un langage de bas niveau utilisé pour programmer directement les microcontrôleurs comme le PIC16F1789. Il repose sur des instructions simples qui interagissent directement avec le matériel.   
  
Structure générale d’un programme en assembleur pour le PIC16F1789 :

org 0x0000 ; Point de départ du programme  
 goto START ; Saut à l'étiquette START  
   
 org 0x004 ; Routine d'interruption (si utilisée)  
 retfie ; Retour après interruption  
   
 START:  
 ; Initialisation des registres  
 ; Corps principal du programme  
 end ; Fin du programme

## 2. Déclaration de Variables

En assembleur, les variables sont généralement stockées dans les registres de la mémoire RAM. Pour le PIC16F1789, les registres de la banque 0 à 3 peuvent être utilisés.   
  
Exemple :

cblock 0x20 ; Début du bloc de mémoire (adresse 0x20)  
 var1 ; Déclaration de la variable 1  
 var2 ; Déclaration de la variable 2  
 endc ; Fin du bloc

TEMP equ 08h

## 3. Opérations Arithmétiques

Les opérations arithmétiques de base incluent l’addition, la soustraction, la multiplication et la division.   
  
Exemples :

; Addition  
 movlw 0x05 ; Charger 5 dans W  
 movwf var1 ; Stocker W dans var1  
 movlw 0x03 ; Charger 3 dans W  
 addwf var1, f ; Ajouter W à var1 (résultat dans var1)  
   
 ; Soustraction  
 movlw 0x02 ; Charger 2 dans W  
 subwf var1, f ; Soustraire W de var1 (résultat dans var1)

## 4. Comparaisons et Logique

Les comparaisons et tests logiques s’effectuent via des instructions conditionnelles.   
  
Exemple pour tester une valeur :

btfsc STATUS, Z ; Tester le bit Zéro dans STATUS  
 goto LABEL\_TRUE ; Si Z=1, sauter à LABEL\_TRUE  
 goto LABEL\_FALSE ; Sinon, aller à LABEL\_FALSE

## 5. Structures de Contrôle

Les structures de contrôle comme if, else, while et for sont réalisées via des instructions de saut conditionnel et non conditionnel.   
  
Exemple d’un if/else :

movlw 0x01 ; Charger 1 dans W  
 subwf var1, w ; Comparer var1 avec W  
 btfsc STATUS, Z ; Si égal, sauter à IF\_TRUE  
 goto ELSE ; Sinon, aller à ELSE  
   
 IF\_TRUE:  
 ; Instructions pour le cas vrai  
 goto END\_IF ; Sauter à la fin de la structure  
   
 ELSE:  
 ; Instructions pour le cas faux  
   
 END\_IF:

Exemple d’une boucle while :

LOOP:  
 ; Instructions de la boucle  
 decfsz var1, f ; Décrémenter var1, sauter si zéro  
 goto LOOP ; Revenir au début de la boucle

## 6. Exemples Pratiques

Exemple : Clignotement d’une LED connectée au port RB0 :

START:  
 bsf PORTB, 0 ; Allumer la LED  
 call DELAY ; Appeler la routine de délai  
 bcf PORTB, 0 ; Éteindre la LED  
 call DELAY ; Appeler la routine de délai  
 goto START ; Recommencer  
   
 DELAY:  
 movlw 0xFF ; Charger un délai  
 movwf COUNT  
 DELAY\_LOOP:  
 decfsz COUNT, f ; Décrémenter le compteur  
 goto DELAY\_LOOP ; Boucler si non nul  
 return

## 7. Utilisation de l'ADC (Convertisseur Analogique-Numérique)

Le PIC16F1789 dispose d’un module ADC intégré permettant de convertir une tension analogique en une valeur numérique. L’ADC fonctionne en configurant plusieurs registres, notamment :   
- ADCON0 : Pour activer l’ADC et sélectionner le canal.  
- ADCON1 : Pour configurer la référence et l’horloge.

Exemple : Lire une tension sur AN0 et stocker la valeur dans un registre :

movlw 0x01 ; Sélectionner AN0 (canal 0)  
 movwf ADCON0 ; Configurer ADCON0  
 bsf ADCON0, GO ; Démarrer la conversion ADC  
 WAIT\_ADC:  
 btfsc ADCON0, GO ; Attendre la fin de la conversion  
 goto WAIT\_ADC  
 movf ADRESH, w ; Lire le résultat (partie haute)  
 movwf RESULT\_H ; Stocker le résultat (haut)  
 movf ADRESL, w ; Lire le résultat (partie basse)  
 movwf RESULT\_L ; Stocker le résultat (bas)

## 8. Utilisation du DAC (Convertisseur Numérique-Analogique)

Le module DAC du PIC16F1789 permet de convertir une valeur numérique en une tension analogique. Le DAC est configuré via le registre DACCON. On peut définir la sortie et la valeur avec :   
- DACCON0 : Activer le DAC et choisir les références.  
- DACCON1 : Définir la valeur numérique à convertir.

Exemple : Générer une tension analogique correspondant à une valeur numérique :

movlw 0x80 ; Activer le DAC, référence interne  
 movwf DACCON0 ; Configurer le DAC  
 movlw 0x3F ; Charger une valeur numérique (63/255)  
 movwf DACCON1 ; Définir la valeur de sortie

## 9. Utilisation du PWM (Modulation de Largeur d’Impulsion)

Le module PWM est souvent utilisé pour contrôler des moteurs ou varier l’intensité d’une LED. Le PWM est généré par le module CCP et nécessite la configuration de plusieurs registres, notamment :   
- CCPxCON : Configurer le mode PWM.  
- PR2 : Définir la période.  
- CCPRxL et CCPxCON<5:4> : Définir le rapport cyclique (duty cycle).

Exemple : Configurer un PWM sur CCP1 avec une fréquence donnée :

movlw 0x3F ; Définir la période PWM (PR2)  
 movwf PR2  
 movlw 0x0C ; Activer le PWM sur CCP1  
 movwf CCP1CON  
 movlw 0x1E ; Définir le rapport cyclique (partie haute)  
 movwf CCPR1L  
 bsf T2CON, TMR2ON ; Démarrer le Timer2 pour générer le PWM