

UNIVERSITÉ CHOUAIB DOUKKALI
Ecole Supérieure de Technologie
Sidi Bennour

Cours: Physique pour l'Informatique
« Programmation Assembleur » 'suite'

Unité 1: Informatique Industrielle programmation assembleur

← → ↻ 🔒 Sécurisé | https://www.google.fr/search?ei=lxsfWqnRAoKB6ATb-7vwDA&q=mplab+download&oq=mplab+d&gs_l=psy-ab.3.1.0i71k1l4.0.0.0.43... 📍 🔍 ☆ 🔔 📧 14

Google 🗣️ 🔍

Tous Vidéos Images Actualités Maps Plus Paramètres Outils

Environ 1.030.000 résultats (0,34 secondes)





✓ **MPLAB- X IDE | Microchip Technology Inc.**
www.microchip.com/mplab/mplab-x-ide ▼ Traduire cette page
Want **MPLAB®** X IDE on the go? Meet the Cloud based **MPLAB®** Xpress IDE. The award winning **MPLAB®** Development Environment is now more portable than ever! Simply navigate to mplabxpress.microchip.com from any compatible internet browser, and enjoy the streamlined **MPLAB** Xpress IDE without any **downloads** ...
[MPLAB- XC Compilers · DV164045 - MPLAB ICD 4 In ...](#)
Vous avez consulté cette page le 21/11/17.

✓ **MPLAB X v1.10 is available for download. | Microchip**
www.microchip.com > ... > [Development Tools] > **MPLAB X IDE** ▼ Traduire cette page
27 févr. 2012 - 1 message - 1 auteur
<http://ww1.microchip.com/downloads/mplab/X/index.html> MD5 e6515697068f3f1b2db2e19d7a07d875
mplabx-ide-v1.10-linux-installer.run 5e7c0ecf3a67fa41a9b4c4b9e8f835ef mplabx-ide-v1.10-osx-
installer.dmg 3c4548c2c4838efa8338e1a4beb85243 mplabx-ide-v1.10-...

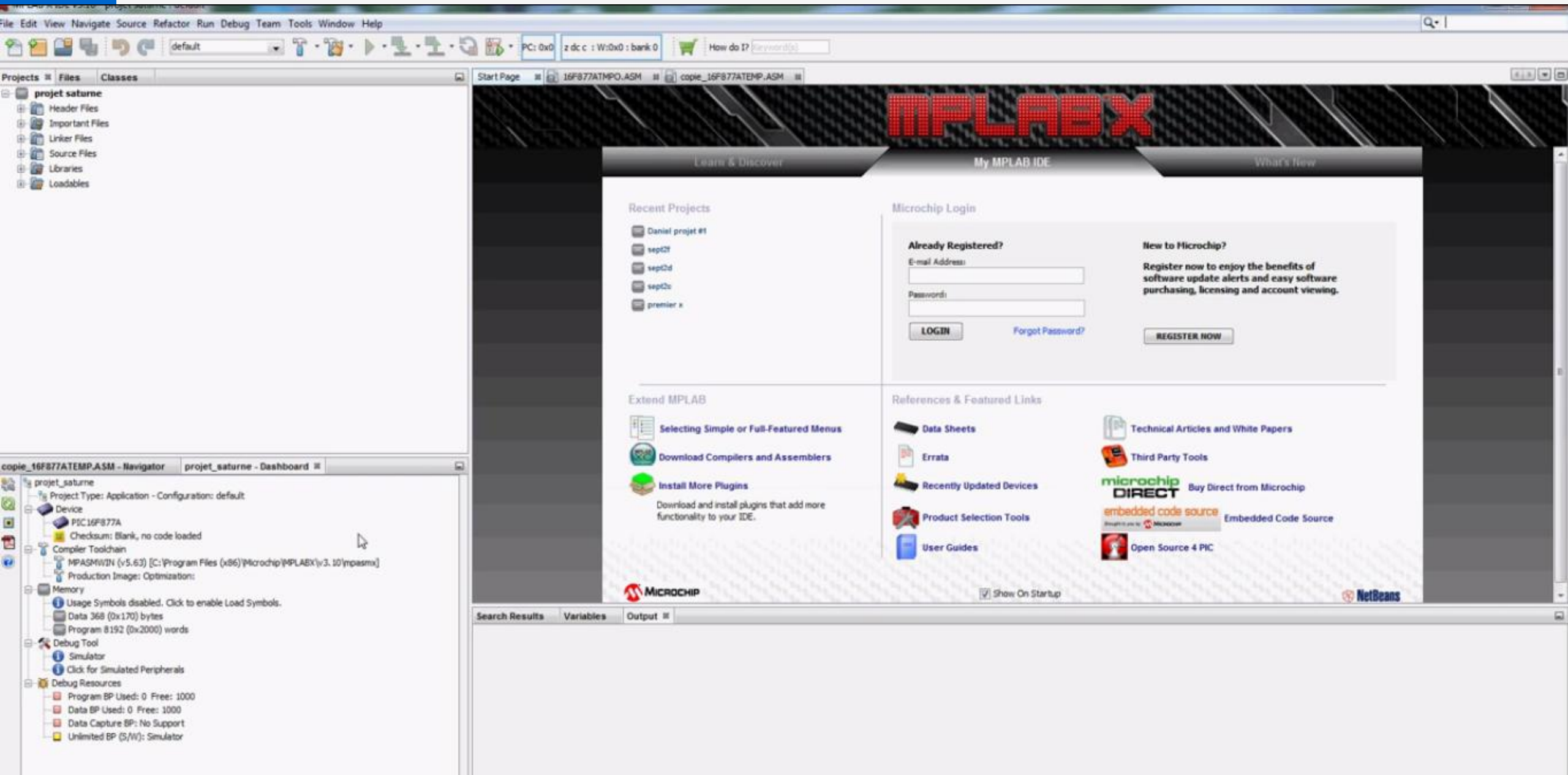
MPLAB X IDE v3.50 Released	1 message	27 oct. 2016
MPLAB X v3.40 broken download (Windows)?	9 messages	21 oct. 2016
MPLAB X v3.35 Released	1 message	22 juin 2016
MPLAB X v1.00 is available for download.	1 message	14 oct. 2011

Unité 1: Informatique Industrielle

programmation assembleur

Features Downloads Documentation Webinars			
Title	Date Published	Size	D/L
Windows (x86/x64)			
MPLAB® X IDE v4.05	11/1/2017	695.1 MB	
MPLAB® X IDE Release Notes / User Guide v4.05	11/1/2017	6.6 KB	
Linux 32-Bit and Linux 64-Bit (Required 32-Bit Compatibility Libraries)			
MPLAB® X IDE v4.05	11/1/2017	646.7 MB	
MPLAB® X IDE Release Notes / User Guide v4.05	11/1/2017	6.6 KB	

Unité 1: Informatique Industrielle



Unité 1: Informatique Industrielle

programmation assembleur

Structure d'un programme en assembleur

Exemple d'un programme simple

Programmer un microcontrôleur consiste à écrire une suite de 0 et de 1 en mémoire programme.

Cette suite sera décrite dans un fichier « **.hex** » généré par MPLAB à partir de l'association de mnémoniques et de directive d'assemblage.

On trouvera ci-après un programme simple permettant de faire basculer la sortie RB7 du port B d'un microcontrôleur PIC16F877.

Unité 1: Informatique Industrielle

```
*****
;
; Ce programme génère une horloge en RB0
; de fréquence 24 fois plus faible que celle du quartz.
; il utilise la carte PicDem2Plus équipée d'un PIC16F877A
*****

LIST P=16F877A          ; directive qui définit le processeur utilisé
#include <P16F877A.INC>  ; fichier de définition des constantes

*****
;
; BITS DE CONFIGURATION
;
;
; __CONFIG __HS_OSC & __WDT_OFF & __CP_OFF & __CPD_OFF & __LVP_OFF

; _XS_OSC l'oscillateur est configuré en oscillateur à quartz haute fréquence
; _WDT_OFF le watchdog est désactivé
; _CP_OFF le code de protection de la mémoire programme est désactivé
; _CPD_OFF le code de protection de la mémoire EEPROM est désactivé
; _LVP_OFF la programmation basse tension est désactivée
; ces opérations sont nécessaires pour fonctionner en mode "debug"
*****

;
; DEMARRAGE SUR RESET
;
*****

org    0x0              ; Adresse de départ après reset
goto   debut

org    0x10             ; adresse de début du programme
debut  :

*****
;
; INITIALISATION
;
*****

; initialisation du PORTB en sortie (voir datasheet)
```

commentaires

fichiers d'en-tête

configuration matérielle

adresses en ROM

étiquette

Unité 1: Informatique Industrielle

programmation assembleur

```
bcf    STATUS, RP0
bcf    STATUS, RP1      ; passage en banque 0
clrf   PORTB            ; RAZ des bascules D
bsf    STATUS, RP0      ; passage en banque 1
movlw  b'00000000'
movwf  TRISB            ; PORTB en sortie
bcf    STATUS, RP0      ; retour en banque 0
```

```
.*****
;
;   PROGRAMME PRINCIPAL                               *
;*****
```

boucle

```
bsf    PORTB,0          ; mise à 1 de la sortie
nop     ; 2 temps morts pour compenser le saut
nop
bcf    PORTB,0          ; mise à 0 de la sortie
goto   boucle           ; rebouclage
```

fin de programme

END

; directive signalant la fin du programme

```
.*****
;
```

Unité 1: Informatique Industrielle

programmation assembleur

Commentaires et étiquettes

- Les commentaires, après un « ; » sont essentiels pour la compréhension du programme et pour son débogage.
- Les étiquettes renvoient à des adresses en mémoire programme, sans avoir à se préoccuper du changement de cette adresse lorsqu'on ajoute ou retire des instructions.

Unité 1: Informatique Industrielle

programmation assembleur

Les fichiers d'en-tête (header files)

- Il faut dans un premier temps préciser au compilateur quel processeur nous allons utiliser ; c'est le rôle de la directive « **LIST** ».
- Le programme précédent utilise un certain nombre de constantes (STATUS, PORTB, TRISB...) associées à des registres du PIC, donc à des adresses ;
- Ces associations sont décrites dans le fichier « include » (par exemple « **P16F877A.INC** » pour un 16F877A). Ce fichier est localisé dans le dossier « **c:\Program Files\Microchip\MPASM suite** » et peut être lu avec un éditeur de texte.
- On trouvera ci-après un extrait de ce fichier où l'on peut voir la définition des adresses des registres, et la position des bits dans le registre (ici le registre STATUS).

;----- Register Files-----			;----- STATUS Bits -----		
INDF	EQU	H'0000'	IRP	EQU	H'0007'
TMR0	EQU	H'0001'	RP1	EQU	H'0006'
PCL	EQU	H'0002'	RP0	EQU	H'0005'
STATUS	EQU	H'0003'	NOT_TO	EQU	H'0004'
FSR	EQU	H'0004'	NOT_PD	EQU	H'0003'
PORTA	EQU	H'0005'	Z	EQU	H'0002'
PORTB	EQU	H'0006'	DC	EQU	H'0001'
PORTC	EQU	H'0007'	C	EQU	H'0000'
PORTD	EQU	H'0008'			

Unité 1: Informatique Industrielle

programmation assembleur

Les directives

La directive « Config »

- Elle permet de paramétrer le registre de configuration (ce registre est détaillé au chapitre « Special Features of CPU » de la documentation) par une instruction du type :

```
__Config __CP_OFF & __PWRTE_ON & __WDT_OFF & __HS_OSC
```

(Attention : il y a deux « _ » avant « **Config** »)

- Ce registre est écrit une fois pour toutes à la programmation et n'est plus accessible par la suite ; il permet de paramétrer matériellement le PIC :
 - configuration de l'oscillateur ;
 - mise en service ou non du chien de garde ;
 - mode de fonctionnement au démarrage...

Unité 1: Informatique Industrielle

programmation assembleur

Les directives

La directive « EQU » d'assignation

- Elle permet d'assigner des valeurs (comme dans le fichier « .inc »), par exemple :

```
MASQUE EQU 0X45
```

- Il s'agit à la fois d'une facilité d'écriture et un outils précieux pour la compréhension (et donc le « débogage ») d'un programme.

Unité 1: Informatique Industrielle

programmation assembleur

Les directives

Les définition par « #define »

- Fonctionne comme l'assignation mais pour un texte. Dans notre programme nous aurions pu écrire par exemple :

```
#DEFINE SORTIE PORTB,0
```

- et remplacer ensuite « bsf PORB,0 » par :

```
bsf    SORTIE
```

Unité 1: Informatique Industrielle

programmation assembleur

Les directives

Les macro-instructions

- Elles permettent de remplacer une série d'instruction par un texte ; attention ce n'est qu'une facilité d'écriture qui ne diminue pas la taille du programme comme le ferait un sous programme. ; exemple :

```
Macro1 macro  
    Suite d'instructions.....  
edm          ; fin de la macro
```

- L'exécution est cependant plus rapide que lors de l'appel d'un sous programme.
- Remarque** : il est possible de déclarer des étiquettes qui seront propre à la macro (et qui pourront donc avoir le même nom qu'une autre étiquette du programme sans risque de confusion) par l'instruction « **local** »
- exemple :

```
local  boucle
```

Unité 1: Informatique Industrielle

programmation assembleur

Les directives

La déclaration des variables

- Permet de définir une zone de variables en RAM, l'emplacement et la taille de ces variables ; exemple :

```
CBLOCK      0x20    ; début de la zone variable
    Variable1 : 1    ; Variable1 a une taille de 1 octet
    Variable2 : 1    ; Variable2 a une taille de 1 octet
    tableau : 8      ; tableau est une variable de 8 octets
ENDC
```

- Il faut veiller à ce que l'endroit choisit ne soit pas déjà occupé par un registre. Par exemple, sur un PIC16F877A, on peut définir une zone de 96 octets à partir de l'adresse 20h.

Unité 1: Informatique Industrielle

programmation assembleur

Les directives

Les directive ORG et END

- La directive ORG permet de préciser à quel endroit en ROM son placées les lignes de programme qui suivent. Sur un PIC, après un RESET, le compteur programme pointe l'adresse 0x00 ; de plus à l'adresse 0X04 se trouve le vecteur d'interruption. Le programme commencera donc de la manière suivante :

```
org    0x00      ; adresse pointé après un reset ;  
                l'instruction qui va suivre se trouvera à cette adresse  
        goto début ;saut à l'étiquette marquant le début du programme  
  
org    0x10      ; cette adresse doit être supérieure à 0x04  
début      ; étiquette marquant le début du programme  
....  
end        ; fin du programme
```

Unité 1: Informatique Industrielle

programmation assembleur

Les règles d'écriture

L'écriture des variables

- MPLAB permet de définir un radix par défaut ; cette option peut être modifiée par «**Project \ build options \ project \ MPASM assembleur**» ; cependant, afin d'améliorer la portabilité des programmes, il est conseillé de toujours préciser le radix des paramètres.

0x1A	ou	H'1A'	ou	1Ah	: notation hexadécimale ;
B'10101111'					: notation binaire ;
D'12'	ou	.12			: notation décimale.
A'C'	ou	'C'			: code ASCII

Unité 1: Informatique Industrielle

programmation assembleur

Les règles d'écriture

La casse

- Par défaut MPLAB est sensible à la casse. Cette option peut être modifiée par « **project \ build options \ project \ MPASM assembleur** ».

Unité 1: Informatique Industrielle

programmation assembleur

Programmes classiques, astuces et limitations

Comparer deux nombres

- Le principe consiste à soustraire les deux nombres et observer les bits Z et C du registre d'état.

```
movf    nb1, w        ; mettre nb1 dans w
subwf   nb2, w        ; soustraire nb2-nb1
```

- Si $Z=1$, les nombres sont identiques ;
- si $Z=0$ et $C=1$, le résultat est positif, donc nb2 est supérieur à nb1,
- si $Z=0$ et $C=0$, le résultat est négatif, donc nb1 est supérieur à nb2,

Unité 1: Informatique Industrielle

programmation assembleur

Programmes classiques, astuces et limitations

Soustraire une valeur du registre de travail

- L'instruction « `sublw 8` » soustrait le contenu de « `w` » à 8, soit $(w)-8$, si on souhaite faire $8-(w)$, il faut ajouter le complément à 2 de 8 au contenu de `w`, soit « `addlw -8` ».

Unité 1: Informatique Industrielle

programmation assembleur

Programmes classiques, astuces et limitations

Pointer un tableau en mémoire programme

- L'instruction « retlw » revient d'un sous programme en plaçant un argument dans le registre « W ». On appelle dans un premier temps l'endroit où se trouve le tableau, tout en plaçant dans « W » le numéro de la ligne du tableau :

```
movf    n_ligne,W      ; W pointe la ligne du tableau
call    SUITE_Fbnc     ; recherche du contenu de la ligne
```

- Une fois arrivé à l'adresse « **SUITE_Fbnc** », on ajoute au compteur programme le contenu de « W », ce qui permet de sauter à la ligne qui nous intéresse, dont l'instruction renvoie l'argument au programme principal en le plaçant dans « W ».

```
SUITE_Fbnc
ADDWF    PCL,F
RETLW    d'01'          ; première ligne
RETLW    d'01'          ; seconde ligne
RETLW    d'02'          ; troisième ligne
RETLW    d'03'          ; quatrième ligne
RETLW    d'05'          ; cinquième ligne
RETLW    d'08'          ; sixième ligne
```