1. TP Prise en main

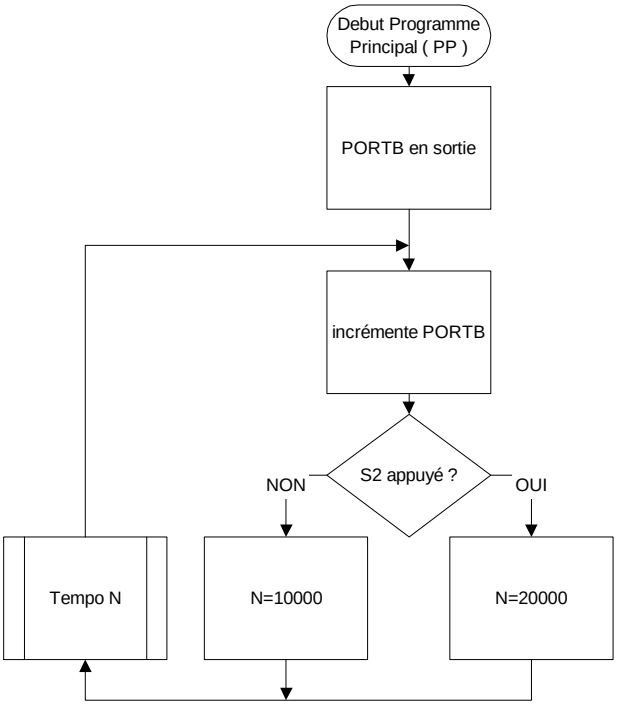
Ex1 :
Créer un projet « EMSEPIC18 » dans votre espace de travail personnel.
Processeur : PIC18F45K20
Langage : C18
Debugger : carte de test sur PICKIT3

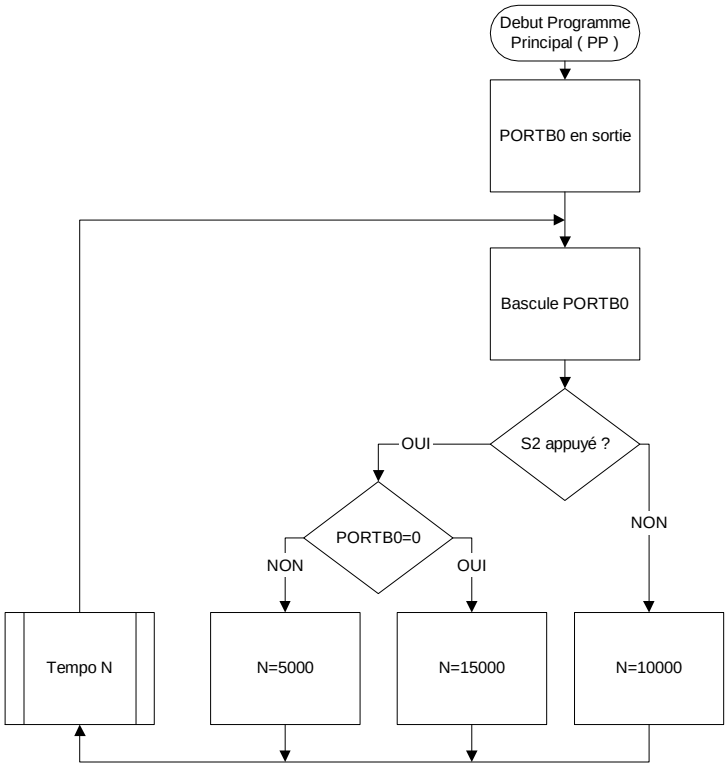
Recopier, analyser et tester le programme “bouton.c”
Visualiser dans une fenêtre « WATCH » les registres TRISD, PORTD, PORTB et leur évolution en « pas à pas »
(La carte test n’a pas d’oscillateur externe, l’oscillateur interne du PIC doit être activé)
Expliquer toutes les lignes #config (rechercher dans MPLAB help-topics-pic18 config settings)

Visualiser la mémoire RAM (appelée FILES REGISTER par MICROCHIP), rechercher le PORTD et comparer avec le contenu de la fenêtre « WATCH » (faire évoluer SW1)
Modifier le programme bouton.c en utilisant les structures définies dans le C18 et remplacer *PORTB & 0x10 par PORTBbits.RB0* dans le test du bouton.
Faire de même pour les valeurs placées dans RD0, remplacer *PORTD=1 et PORTD=0* par *PORTDbits.RD0=1 et PORTDbits.RD0=0*

Ex2 : modifier le programme led.c de manière à modifier la tempo (passer de 10000 à 20000) si S2 est appuyé. (Algorigramme de gauche ci dessous)
Créer un nouveau projet avec PROTEUS-ISIS comme simulateur et la carte
CARTE_TEST_PIC18F4620.DSN comme cible.(attention on change de microcontrôleur)
modifier led.c précédent (le bouton est S2 et la LED RB0)
Visualiser RB0 avec l’oscilloscope de ISIS et vérifier que la tempo double en fonction de la position de S2.

Ex3 : Réaliser un programme faisant clignoter RB0 avec une période proche de 1s et un rapport cyclique ¼ si S2 est appuyé et ½ sinon (algorigramme de droite) Visualiser RB0 avec l’oscilloscope de ISIS.
Effectuer maintenant les essais sur la carte de test du PICKIT3





2. TP Horloge Interne

Sur carte de simulation CARTE_TEST_PIC18F4620.DSN, configurer la fréquence de l'oscillateur du PIC à 4Mhz (FOSC) (clic-droit sur le composant), quelle sera alors le temps de cycle machine (TCY)
Placer un oscilloscope sur la LED.
Adapter la temporisation du programme LED.C de manière à obtenir une période d'environ 1S pour la LED.
Modifier la fréquence de l'oscillateur à 1Mhz puis à 8Mhz, les résultats sur la période d'oscillation de la LED sont-ils cohérents ?

Mise en service de l'horloge interne
Rechercher dans MPLAB help-topics-PIC config settings les configurations des prama

#pragma config OSC =

#pragma config WDT =

#pragma config LVP =

#pragma config PBADEN =

#pragma config CCP2MX =

Active l'horloge interne avec PA6 et PA7 configuré comme des PORTS E/S
Desactiver le chien de garde
Desactiver le mode low voltage programming
Configurer le PORTB en numérique lors du RESET
Connecter CCP2 à PB3

Tester maintenant le programme LED.C, quelle est la période d'oscillation, est elle cohérente avec la configuration des prama.

Configurer maintenant le registre OSCON de manière à disposer d'une horloge interne à 8Mhz et mesurer la période sur la LED

Configurer maintenant le registre OSCON et OSCTUNE de manière à disposer d'une horloge interne à 32Mhz et mesurer la période sur la LED

3. TP I2C

A partir de :
DataSheet P18Fxx20 et PCF 8574
Cours MCC18
Doc ressource PIC18F
Programme : testI2C.c
Schema PCF8574.dsn

Ex1. Découverte du bus I²C
Créer un projet MPLAB (si nécessaire) avec comme seul fichier source : testI2C.c
Analyser ce programme et dessiner son algorigramme.
Tester le programme en plaçant un point d'arrêt à la ligne :"data++" et faire une simulation en utilisant le schéma PCF8574.dsn.
Observer les trames émises sur le bus I²C à l'aide du debugger I²C. Comparer les trames observées avec celles de la documentation du constructeur.

Ex2. Utilisation du PCF 8574 en E/S
Modifier le programme précédent pour que les broches :
- P4 à P7 fonctionnent en entrée,
- P0 à P3 fonctionnent en sortie.

Le programme permettra la recopie du contenu des entrées vers les sorties. Effectuer une simulation en utilisant le schéma PCF8574bis.dsn.

A partir de :
DataSheet P18Fxx20 et 24LC64
Cours MCC18
Doc ressource PIC18F
Schema 24lc64.dsn

Ex3. Mémoire E2PROM
A partir de la documentation de l'E2PROM écrire les fonctions suivantes :
- *void wr_oct(char adrs_sel ,int adresse, unsigned char donnee)*
  // Permet l'écriture de l'octet *donnee* à l'adresse : *adresse* de l'E2PROM d'adresse : *adrs_sel*
- *unsigned char rd_oct(char adrs_sel ,int adresse)*
  // Permet la lecture d'un octet à l'adresse : *adresse* de l'E2PROM d'adresse : *adrs_sel*
- *void wr_int(char adrs_sel ,int adresse, unsigned int *donnee)*
  // Permet l'écriture d'un entier *donnee* à l'adresse : *adresse* de l'E2PROM d'adresse : *adrs_sel*
- *unsigned int rd_int(char adrs_sel ,int adresse)*
  // Permet la lecture d'un entier à l'adresse : *adresse* de l'E2PROM d'adresse : *adrs_sel*

Ecrire un programme permettant de tester ces fonctions ; puis faire une simulation en utilisant le schéma 24lc64.dsn

4. TP communication asynchrones

A partir de
DataSheet P18Fxx20
Cours MCC18
Doc ressource PIC18F
Programme : tstusart.c
Carte de simulation ISIS : carte_test.dsn
Simulateur PROTEUS/ISIS dans MPLAB

Ex1 :
Analyser le programme tstusart.c, expliquer chaque ligne et tracer son algorigramme

Justifier en particulier les lignes.

```
SPBRGH= 0x00;
SPBRG = 25;
TXSTA = 0b00100000;
RCSTA = 0b10010000;
BAUDCONbits.BRG16=0;
TXSTAbits.BRGH=1;
```

Expliquer le rôle des bits : `PIR1bits.RCIF et PIR1bits.TXIF.`

Tester le programme tstusart.c, à l'aide du terminal de la carte de simulation
Visualiser RX sur l'oscilloscope lors de la réception du caractère ASCII 'A', interpréter le protocole NRZ

Modifier la fréquence du quartz en 16 Mhz (clic-droit sur le PIC18 puis éditer propriétés
Rechercher dans le data sheet du PIC18F4620 les valeurs des registres et bits précédents
Pour configurer l'USART en 19200 bauds, no parity, 1 stop.
(Pensez à configurer également le terminal clic-droit éditer propriétés)

Ex2 :
Créer une nouvelle fonction void putchaine(rom char* chaine), permettant d'envoyer une chaine de caractère ASCII sur l'USART du PIC, la tester. Expliquer pourquoi on préfère placer la chaine émise est en rom.

Ex3 :
Le C18 dispose d'une bibliothèque stdio.h
Tester dans le programme précédent :

```
printf(« Bonjour \n\r ») ;
printf(« Decimal %d caractere %c hexa %X chaine %s »,41,41,41,ma_chaine) ;
```

Réaliser un programme additionnant deux chiffres entrés successivement sur le terminal

Réaliser un programme affichant sur l'afficheur 7 segments de gauche le chiffre entré sur le terminal (0à F). Pour cela créer un tableau de codes 7 segments.

Pour les plus rapides …
Tester le programme tst_sinus.c puis faire l'exercice « exponentielle » proposé sur le poly
A partir du programme horloge.c
Réaliser une horloge temps réel (RTC) heures, minutes, secondes sur les afficheurs 7 segments avec transfert sur USART lors de l'appui sur S2.

5. TP Interruptions et TIMER0

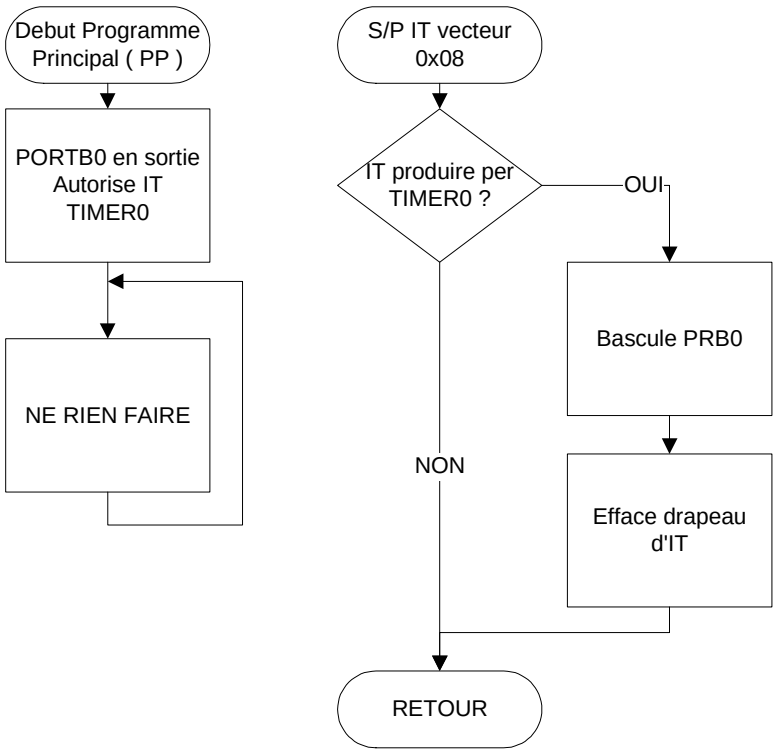
A partir de
DataSheet P18Fxx20
Cours MCC18
Doc ressource PIC18F
Programme : demo_it_rb0.c et flashIT.c
Debugger : PICKIT3 + carte de demo

Ex1 : Découverte des interruptions
Créer un projet MPLAB (si nécessaire) avec comme seul fichier source : demo_it_rb0.c
Analyser ce programme, expliquer chaque ligne, dessiner son algorigramme.
Le tester sur la carte du PICKIT3

- En mode « run »
- Placer ensuite un point d'arrêt dans les sous-programme d'interruption.
- Visualiser en pas à pas dans une fenêtre « watch » `INTCONbits.INT0IF et cpt.`

Ex2 : TIMER 0 : programme flashit.c fonctionnant sur PICKIT3

- A partir du data sheet du P18F45K20, justifier la valeur 0x82 du registre T0CON
- Identifier les lignes de code C correspondants à l'algorigramme ci-dessous



-Tester dans le projet précédent le programme flashit.c

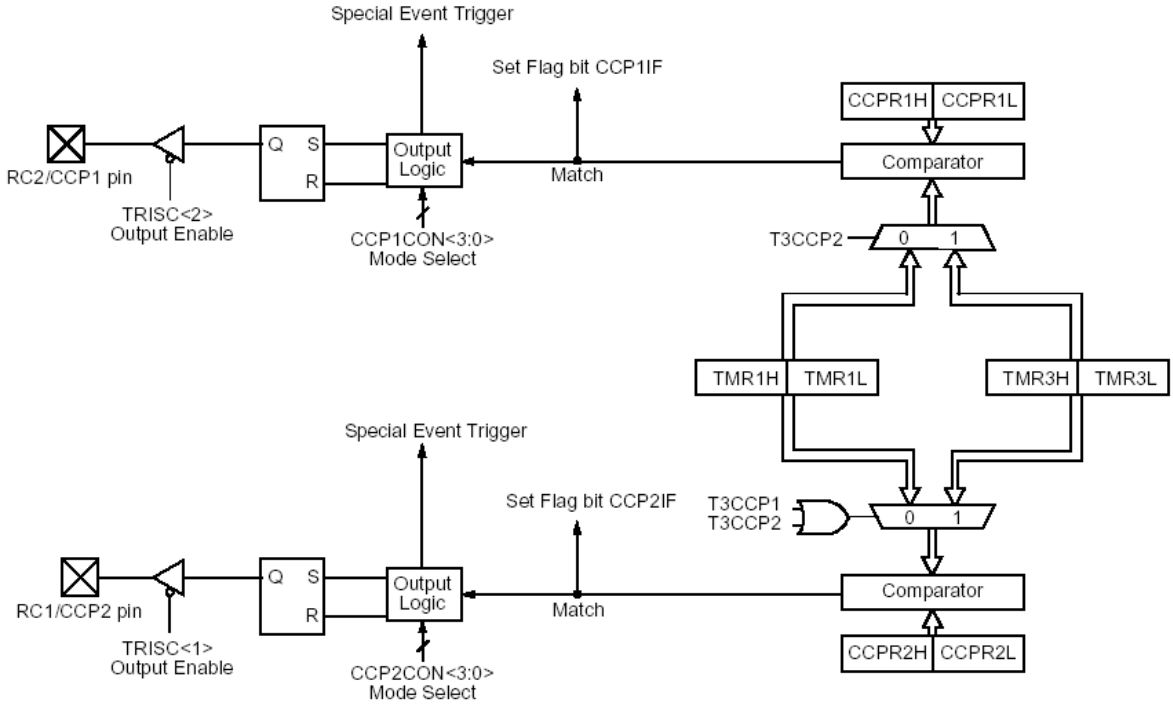
Ouvrir le projet précédemment crée pour la simulation sur PROTEUS ISIS
Simuler le programme sur la carte "CARTE_TEST_PIC18F4620.DSN"
- Placer un oscilloscope sur RB0, mesurer la période du signal la comparer à celle attendue et conclure par rapport à l'erreur mesurée.

Ex3 : Generateur d'implusions.
A partir du programme flashit.c :
Réaliser un programme générant sur RB0 un signal de fréquence 97,65625Hz et de rapport cyclique 1/10.
Les essais seront réalisés sur la carte PICDEM2+ with PIC18F4620.dsn avec l'oscilloscope ISIS.

6. TP COMPARE/CAPTURE

A partir de
DataSheet P18Fxx20
Cours MCC18
Doc ressource PIC18F
Programme : itcomp.c
Debugger : PROTEUS-ISIS “CARTE_TEST_PIC18F4620.DSN”
Simulateur PROTEUS/ISIS dans MPLAB

Ex1 : COMPARE
Analyser le programme itcomp.c, expliquer chaque ligne et tracer son algorigramme
Tester la programme itcomp.c , visualiser le signal de sortie avec l’oscilloscope ISIS et interpréter les résultats.



Justifier lesvaleur placées dans les bits et registres suivants :

```
T1CONbits.RD16=0;
T1CONbits.TMR1CS=0;
T1CONbits.T1CKPS1=1;
T1CONbits.T1CKPS0=1;
T1CONbits.T1SYNC=1;
T1CONbits.TMR1ON=1;
T3CONbits.T3CCP2=0;
CCP1CON=0x0B;
CCPR1H=0x3d;
CCPR1L=0x09;
```

Ex2 : Modifier le programme précédant avec comme sortie RC2/CCP1 de manière à obtenir un basculement automatique. Quel est l’avantage de cette solution par rapport à celle du programme itcomp.c ?
Placer un oscilloscope sur RC2 et mesurer la période du signal, conclure

Ex3 : Modifier itcomp.c de manière à obtenir un rapport cyclique ¼ si S2 est enfoncé et ¾ sinon en fonction de TICTAC ou des valeurs dans CCPR1

Ex4 : CAPTURE
Analyser programme ITCAPT.C, justifier les valeurs placées dans les registres
T1CON, CCP1CON ainsi que les valeurs des bits T3CCP2, CCP1IE
Tester ce programme sur avec le simulateur ISIS et la carte TEST_USART_PWM_CCP.DSN. (placer un oscilloscope sur CCP1)
Expliquer la valeur présente sur le terminal ASCII.
Modifier la fréquence du générateur sur l'entrée CCP1 afin de déterminer les périodes min et max mesurables.

Ex5 :
Compléter le programme afin d'afficher la fréquence du signal sur CCP1

Ex6 : A partir du programme précédent, réaliser un programme affichant (en uS) la durée de l'état haut du signal sur CCP1.

7. TP PWM

Le programme TST_PWM.C réalise une démonstration de PWM sur la carte de démonstration du PICKIT3.

Tester ce programme.

Justifier les fonctions des registres et les valeurs placées dans les registres TCON2, PR2, CCPR1L, CCP1CON

Pourquoi tester le bit TMR2IF ?

Que se passe t il lorsque l'on modifie le registre CCPR1L ?

A partir de ce programme, réaliser un programme pour 18F4620 (horloge interne 32Mhz) sur le simulateur ISIS réalisant une PWM (sortie CCP2) avec une fréquence de 10Khz.
Le rapport cyclique sera modifiable par pas de 10% par appui sur un bouton. On visualisera le résultat sur un oscilloscope, afin de vérifier fréquence et rapport cyclique.

8. TP ADC

A partir de
DataSheet P18Fxx20
Cours MCC18
Doc ressource PIC18F
Programme : demo_ADC.C
Debugger : PICKIT3 + carte de demo

Ex1. Découverte du Convertisseur Analogique Digitale
Créer un projet MPLAB (si nécessaire) avec comme seul fichier source : demo_ADC.c
Analyser ce programme, expliquer chaque ligne, dessiner son algorigramme.
Tester le programme en plaçant un point d'arrêt à la ligne :" if ((400<result)&&(result<600))"
Quelle est la fonction réalisée par l'association de la structure et du programme

A partir de
DataSheet P18F1320
Schéma : Proteus : Thermomètre.dsn

Ex2. Mesure de température
On désire mesurer la température à partir d'une sonde PT100 et afficher cette dernière sur un afficheur LCD (voir schéma Thermomètre.dsn).
La température à mesurer est comprise entre -20℃ et 130℃.
Pour améliorer la précision, on utilise les références de tension externes (Vref+ et Vréf-) du convertisseur. Dans ce cas le quantum du convertisseur est donné par : $q = \frac{Vréf + Vréf -}{2^{n-1}}$

On utilise le convertisseur en mode 10 bits.

1. Créer un projet MPLAB si nécessaire.
2. Compléter le programme thermmètre.c
3. Tester votre programme (debugger = VSM)